DEUTSCH

MOS

MICROCOMPUTERS

MICROCOMPUTERS

MICROCOMPUTERS

KIM-1 HANDBUCH

KIM 1

HANDBUCH

MICROCOMPUTER MODULE

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

INHALTSVERZEICHNIS

VERZEICHNIS DER ABBILDUNGEN

KAPITEL 1

IHR KIM-1 MIKROCOMPUTER MODUL

Willkommen in der neuen, interessanten Welt der Mikrocomputer. Als Besitzer eines KIM-1 Mikrocomputer Moduls verfügen Sie über einen sehr leistungsfähigen Digitalcomputer. Dieser sofort einsatzbereite, vollständig überprüfte Computer von MOS Technology entspricht dem neuesten Stand der Technik. Mit der Wahl des KIM-1 Moduls umgehen Sie alle Probleme die mit dem Aufbau und Austesten eines Mikrocomputersystems verbunden sind. Nach dem Studium der Systemfunktionen können Sie Ihre wertvolle Zeit zur Integration des KIM-1 Moduls in Ihr spezielles Interessengebiet nutzen. Wenn Sie einige in diesem Handbuch beschriebenen Prozeduren verfolgt haben, werden Sie in kürzester Zeit in der Lage sein, diese elementaren Beispiele zu erweitern.

Ihr KIM-1 Modul ist für zahlreiche Betriebsarten ausgelegt. Betreiben Sie das System mit dem Tastenfeld und der LED-Anzeige, beide sind bereits Bestandteile des Moduls. Als nächstes fügen Sie einen Nf-Kassettenrekorder hinzu. Damit verfügen Sie über eine preisgünstige Methode, um Ihre Programme abzuspeichern und wieder zuladen. Ein 8-Kanal Fernschreiber ist ebenfalls direkt anschließbar. Damit können Sie Befehle unmittelbar eingeben, Ergebnisse ausdrucken, Lochstreifen erstellen und diese wieder einlesen.

Das Herz Ihres KIM-1 Systems ist ein MCS 6502 Mikroprozessor, der mit zwei MCS 6530-Bausteinen zusammenarbeitet. Jeder MCS 6530 enthält 1 KByte ROM, 64 Byte RAM, 15 I/0-Anschlüsse und einen Zeitgeber. Die ROM-Zellen der MCS 6530 Bausteine speichern die von MOS Technology Inc. entwickelten Dienstprogramme. Diese wiederum ermöglichen die verschiedenen Betriebsarten des KIM-1 Systems.

Das KIM-1 System wurde entwickelt, um Ihnen ein leistungsfähiges Mikrocomputer-System in die Hand zu geben, mit dem Sie Ihre digitalen Steuerungsprobleme elegant lösen. Zu diesem Zweck enthält das System 1024 Bytes RAM zum Speichern von Daten und Anwenderprogrammen. Zusätzlich stehen 15 bidirektionale I/0-Anschlüsse zur Interface-Steuerung zur Verfügung. Und schließlich kann einer der Zeitgeber für Ihre spezielle Anwendung genutzt werden.

Sie erhalten das KIM-1 System komplett bestückt und getestet. Da es bereits einen 1 MHz Quarz enthält, verschwenden Sie auf die Taktversorgung der CPU keine Gedanken. Es gibt auch keine Interfaceprobleme. Schließen Sie nur die angegebenen Versorgungsspannungen an die dafür vorgesehenen Stifte an, und schon arbeiten Sie mit Ihrem KIM-1 System.

Wir empfehlen Ihnen dieses Handbuch ganz durchzulesen, bevor Sie Ihren KIM-1 Modul in Betrieb nehmen. Der Inhalt der folgenden Kapitel ist:

Kapitel 2 — Tips für die erste Inbetriebnahme

Kapitel 3 — Eine genaue Beschreibung von Hard-und Software des KIM-1 Systems

Kapitel 4 — Prozeduren für sämtliche Betriebsarten

Kapitel 5 — Ein Beispiel für einen typischen Anwendungsfall, welcher alle Elemente des KIM-1 Systems benutzt

Später können Sie Ihr KIM-1 System erweitern, d.h. die Speicherkapazität vergrößern, andere Speichertypen anschließen oder zusätzliche Ein- und Ausgabesports erhalten. Um eine solche Systemerweiterung möglichst einfach zu gestalten, sind bereits alle dazu erforderlichen Signale auf einen Stecker des Moduls geführt. Beachten Sie:

Kapitel 6 Anleitung zum Systemausbau — Speicher und I/O-Erweiterung

Im Anschluß an den Haupttext dieses Handbuches finden Sie einen Anhang, der Ihnen ausführliche Informationen zum Verständnis der Funktion des KIM-1 Systems gibt.

Da Ihnen dieses Handbuch nicht alles über Hardware und Programmierung des MCS 6502 sagen kann, fügen wir noch zwei Bücher bei. Das eine behandelt die Hardware; es beschreibt die verschiedenen Bausteine des Systems, elektrische Werte, Systemdaten und Zeitdiagramme. Das andere behandelt die Software; es vermittelt Ihnen die Informationen welche nötig sind, um mit dem Befehlsvorrat des MCS 6502 Programme zu schreiben. Soviel zur Einführung. Wollen wir nun beginnen und sehen wie Ihr KIM-1 Modul für Sie arbeiten kann.

KAPITEL 2

EINFÜHRUNG

Dieses Kapitel wird Ihnen die ersten wichtigen Schritte zur Durchführung der Grundoperationen Ihres KIM-1 Mikrocomputer-Moduls zeigen. Sie sollen zunächst einige Operationen ohne nähere Erklärung ausführen. In späteren Abschnitten dieses Handbuchs wird dann jeder Vorgang genau erklärt.

2.1 EINZELTEILE

Nach dem Öffnen der KIM-1 Verpackung finden Sie folgende Teile:

3 Bücher – KIM-1 User Manual
Hardware Manual
Programming Manual

1 Programmierkarte

1 Systemübersicht

1 KIM-1 Modul

1 Buchsenleiste (bereits am Modul aufgesteckt)

1 Hardwarepaket

1 Garantiekarte

Sie sollten den Behälter und das Verpackungsmaterial aufbewahren, falls Sie den KIM-1 Modul später einmal zurückschicken müssen.

2.2 VORSICHTSMASSNAHMEN

Achtung!

Ihr KIM-1 Modul enthält eine Reihe von integrierten MOS-Schaltungen. Alle Ein- und Ausgangsstifte dieser Bauelemente sind zwar durch chipinterne Beschaltung gegen statische Aufladung geschützt, trotzdem sollte man hohe Spannungen möglichst fernhalten.

Bevor Sie das Verpackungsmaterial von Ihrem KIM-1 Modul entfernen, beachten Sie bitte folgende Vorsichtsmaßnahmen:

1. Entfernen Sie statische Aufladung von Ihrem Körper indem Sie ein geerdetes Potential berühren; nehmen Sie erst jetzt Ihren KIM-1 Modul in die Hand. Diese Maßnahme ist besonders dann wichtig, wenn Sie sich in einem Raum mit Teppichboden befinden.

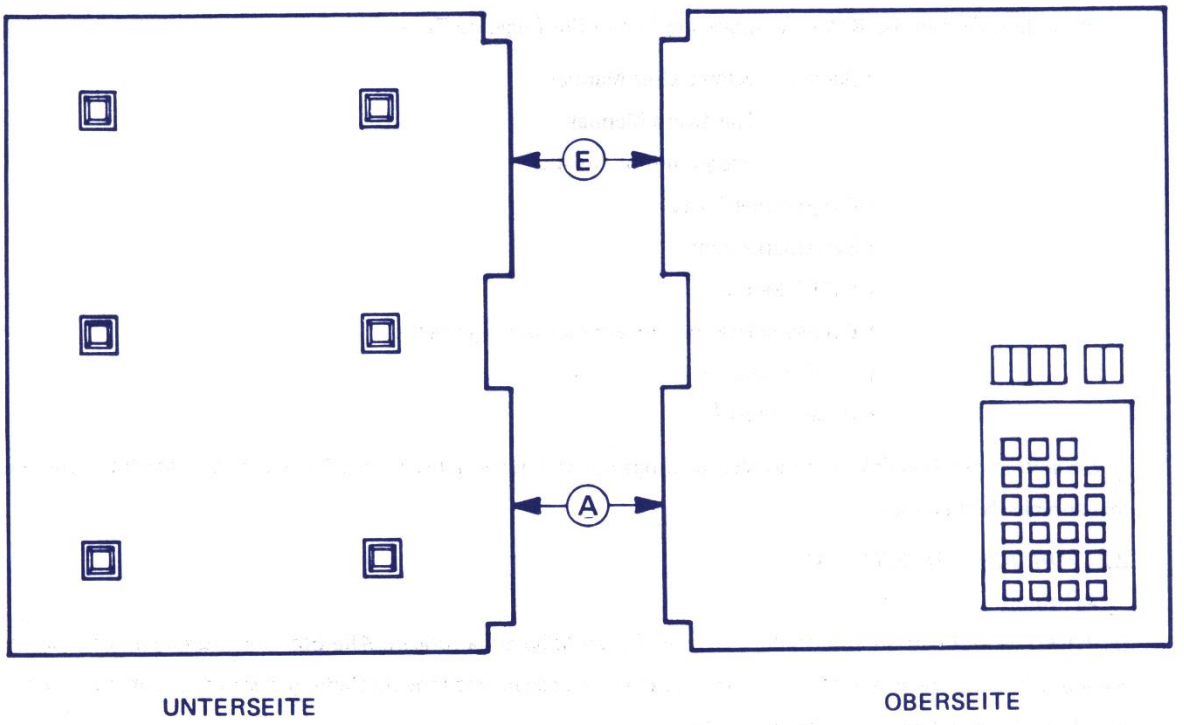
2. Überzeugen Sie sich davon, daß alle Lötkolben und verwendeten Meßgeräte geerdet sind.

Sie werden an Ihrem KIM-1 Modul ein Potentiometer entdecken. Dieses gehört zum Magnetband-Interface und wurde im Herstellerwerk abgeglichen. Sie sollten die Stellung dieses Potis nie verändern!

2.3 ERSTE SCHRITTE

Entnehmen Sie dem Hardwarepaket die Gummifüße und befestigen Sie diese auf der Unterseite des Moduls (siehe Bild 2.1). Sie dienen dazu, Abstand zwischen dem Modul und Ihrer Arbeitsplatte zu halten. Gleichzeitig nehmen sie den Druck auf, wenn eine Taste betätigt wird.

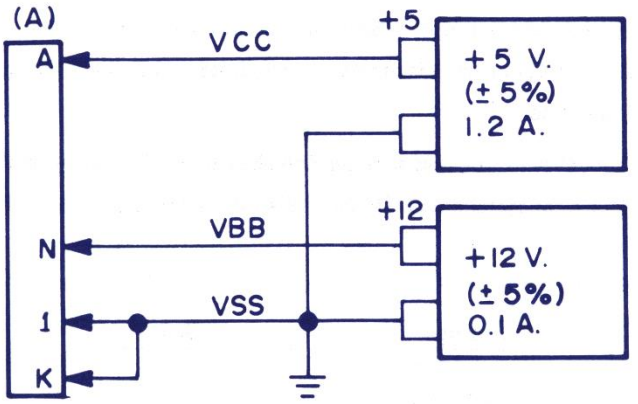
Legen Sie den Modul so, daß das Tastfeld vorne rechts ist und die beiden Stecker nach links zeigen. Der Stecker unten links ist der Applikations-Stecker (A). Eine Buchsenleiste mit 44 Kontakten wurde bereits aufgesteckt. Der oben links gelegene Stecker ist für spätere Systemerweiterung bestimmt, er wird Expansions-Stecker genannt.



KIM-1 Modul

Bild 2.1

Entfernen Sie die Buchsenleiste A vom Modul und beschalten Sie diese gemäß Bild 2.2.



Versorgungsspannungs-Anschlüsse

Bild 2.2

Stecken Sie nach Anschluß der Versorgungsspannungs-Leitungen die Buchsenleiste wieder auf den Stecker A. Überzeugen Sie sich von der Richtigkeit der PINbelegung! Beachten Sie:

1. Die +12 Volt-Spannung wird nur dann benötigt, wenn ein Kassetten-Rekorder angeschlossen werden soll.
2. Die Brücke von Stift A - K nach Vss (Stift A-1) ist wichtig für die Funktion des Systems. Bei Systemerweiterung muß diese Brücke entfernt werden. Wir sagen Ihnen später, wie dann der Stift A - K belegt wird.
3. Besitzen Sie noch keine geeignete Stromversorgung, dann sollten Sie die in Anhang D gezeigte Schaltung aufbauen. Auf jeden Fall müssen Ihre Spannungen elektronisch geregelt sein und die in Bild 2.2 angegebenen Ströme liefern können.

Schalten Sie die Versorgungsspannungen wieder ein und drücken Sie die RS-Taste (RESET). Es müssen alle Digits der LED-Anzeige aufleuchten. Das ist ein erstes Zeichen dafür, daß Ihr System betriebsbereit ist. Ist dies nicht der Fall, überprüfen Sie Ihren Aufbau oder schlagen Sie im Anhang C ("Wenn Schwierigkeiten auftreten") nach.

2.4 Ein einfaches Programm

Nach erfolgreichem Ablauf der bisherigen Schritte kann nun ein kleines Programm geladen werden, das die Funktion des Systems aufzeigt und Ihnen die Gewißheit gibt, daß alles richtig arbeitet. Wir verwenden zunächst

nur das Tastenfeld und die LED-Anzeige; beide befinden sich bereits auf dem Modul. In den beiden nächsten Abschnitten werden wir uns mit dem Fernschreiber und dem Kassetten-Rekorder beschäftigen.

In diesem ersten Beispiel wollen wir zwei 8-Bit Binärzahlen miteinander addieren und das Ergebnis anzeigen. Wir gehen davon aus, daß Sie mit der hexadezimalen Darstellung von Zahlen und den grundsätzlichen Gesetzen der Binärarithmetik vertraut sind.

Überzeugen Sie sich zunächst davon, daß sich der Schiebeschalter im Tastenfeld oben rechts in der linken Position befindet (SST Modus ausgeschaltet). Drücken Sie die Tasten gemäß dem folgenden Bild und prüfen Sie die Anzeige:

Drücke Taste	Anzeige	Schritt Nr.
AD	xxxx xx	1
0 0 0 2	0002 xx	2
DA	0002 xx	3
1 8	0002 18	4
+ A 5	0003 A5	5
+ 0 0	0004 00	6
+ 6 5	0005 65	7
+ 0 1	0006 01	8
+ 8 5	0007 85	9
+ F A	0008 FA	10
+ A 9	0009 A9	11
+ 0 0	000A 00	12
+ 8 5	000B 85	13
+ F B	000C FB	14
+ 4 C	000D 4C	15
+ 4 F	000E 4F	16
+ 1 C	000F 1C	17

Sie haben soeben ein Programm eingegeben und dieses in den RAM-Zellen 0002 bis 000F gespeichert. Beachten Sie die Bedeutung der Sondertasten:

AD = Wählt den Adress-Lademodus
DA = Wählt den Daten-Lademodus
\+ = Inkrementiert die Adresse ohne den Eingabemodus zu verändern
0 bis F = 16 Eingabetasten, hexadezimal kodiert für Adressen und Daten

Sie haben bereits festgestellt, daß Ihre LED-Anzeige 6 Digits besitzt. Die 4 Digits auf der linken Seite zeigen die Adresse hexadezimal an. Die beiden rechten Digits zeigen die unter diesen Adressen abgelegten Daten, ebenfalls hexadezimal. Wenn Sie die Taste AD drücken (Schritt 1) und danach 0002 (Schritt 2), so wählen Sie den Adress-Lademodus, geben die Adresse 0002 ein und stellen diesen Wert in den vier linken Digits dar. Das Zei-

chen x im Eingabeplan bedeutet, daß die Anzeige an dieser Stelle nicht definiert ist (Zufallswert).

Als nächstes haben Sie die Taste DA gedrückt (Schritt 3) und danach 18 (Schritt 4). Hier haben Sie den Daten-Lademodus angewählt und die Hexazahl 18 eingegeben, die natürlich an der zuvor geladenen Adresse 0002 gespeichert wurde. Infolgedessen wird die Zahl 18 in den beiden rechten Digits angezeigt.

Sie blieben im Daten-Lademodus, drückten die Taste + und danach eine zweistellige Zahl (Schritte 5 bis 17). Immer wenn die Taste + gedrückt wurde, erhöhte sich die Adress-Anzeige um den Wert 1. Die danach eingegebenen Hexazahlen werden im Daten-Anzeigefeld sichtbar. Diese Prozedur ist sehr bequem, wenn eine Reihe von aufeinanderfolgenden Speicherplätzen geladen werden soll. Ist Ihnen beim Drücken der Taste ein Fehler unterlaufen, so können Sie diesen ganz einfach korrigieren, indem Sie den falschen Wert durch nochmaliges Eingeben des richtigen überschreiben. Das soeben geladene Programm ist eine einfache Schleife, mit welcher zwei 8-Bit Binärzahlen miteinander addiert werden können. Das Ergebnis erscheint am Display.
Für den Programmierer sieht die Auflistung des Programms wie folgt aus:

```
POINTL          = $FA
POINTH          = $FB
START           = $1C4F
0000            VAL1
0001            VAL2
0002    18      PROG    CLC
0003    A5 00           LDA VAL1
0005    65 01           ADC VAL2
0007    85 FA           STA POINTL
0009    A9 00           LDA #ØØ
000B    85 FB           STA POINTH
000D    4C 4F 1C        JMP START
```

Dieses Programm löscht zunächst das Carry-Flag (CLC), lädt den Wert VAL1 in den Akkumulator (LDA VAL1), addiert dazu den Wert VAL2 (mit Carry) und speichert das Ergebnis in der Zelle POINTL (STA POINTL). Der Wert Null wird in die Zelle POINTH geladen (LDA 00 und STA POINT H), dann springt das Programm an eine Stelle welche die Bezeichnung START trägt (JMP START). Dieses (Fest-) Programm spricht das Display an und bewirkt, daß die Inhalte der Speicherzellen POINTH und POINTL im Adressfeld angezeigt werden. Beachten Sie, daß das Ergebnis der Addition bereits in der Zelle POINTL enthalten war.

Die Hexazahlen, die in der Auflistung neben den Adresswerten stehen, sind genau die Zahlen, welche Sie eingegeben haben. Wir nennen diese Zahlen den Maschinenkode. Zum Beispiel entspricht die Hexazahl 4C dem Mikroprozessor-Sprungbefehl JMP. Die beiden folgenden Bytes 1C und 4F geben das Sprungziel an.

Sie können das Programm noch nicht starten, weil die beiden Variablen (VAL1 und VAL2) noch nicht eingegeben sind. Versuchen wir ein Beispiel:

Drücke Taste	Anzeige	Schritt Nr.
AD	000F 1C	17A
0 0 F 1	00F1 xx	17B
DA 0 0	00F1 00	18
AD	00F1 00	19
0 0 0 0	0000 xx	20
DA 0 2	0000 02	21
+ 0 3	0001 03	22
+ GO	0002 18	23

Die Schritte 17A, 17B und 18 wählen die Binärarithmetik-Arbeitsweise des Mikropozessors. Die Schritte 19 bis 21 speichern die Hexazahl 02 in die Speicherzelle 0000 (VAL1). Der Schritt 22 speichert die Hexazahl 03 in Speicherzelle 0001 (VAL2). Nachdem auf die Startadresse 0002 erhöht wurde, können wir das Programm starten. Im Schritt 23 bewirkt die GO-Taste den Programmstart; das Ergebnis erscheint in den beiden rechten Digits des Adress-Anzeigefeldes. Obwohl das Problem trivial ist, zeigt es doch im Prinzip das Laden und Durchführen eines Programms und beweist, daß Ihr KIM-1 Modul mit großer Wahrscheinlichkeit ordnungsgemäß arbeitet. Führen Sie mit dem gespeicherten Programm noch ein Beispiel durch. Wiederholen Sie die Schritte 17A bis 23 indem Sie für VAL1 und VAL2 die Hexazahl FF in die Speicherzellen 0000 und 0001 laden. Wenn Sie jetzt die GO-Taste drücken (Startadresse 0002), muß auf dem Display der Wert 00FE xx erscheinen. Das Ergebnis ist richtig weil:

```
   FF = 1111 1111
+  FF = 1111 1111
   FE   1111 1110
```

Bevor wir fortfahren, können Sie noch einige Beispiele selbst durchführen.

2.5 Anschluß eines Nf-Kassetten-Rekorders

Im vorangegangenen Abschnitt haben Sie ein Programm geladen und ausgeführt. Wenn Sie die Stromversorgung Ihres Systems ausschalten, ist das Programm verloren, da es in RAMs geladen war. Benötigen Sie das gleiche Programm noch einmal, so müssen Sie die ganze Eingabeprozedur noch einmal durchführen.

Um Ihnen diese Mühe zu ersparen, ist der Anschluß eines Magnetbandgeräts an den KIM-1 Modul vorgesehen. Damit können Sie Programme und Daten speichern, auch wenn Ihr Modul ausgeschaltet ist.

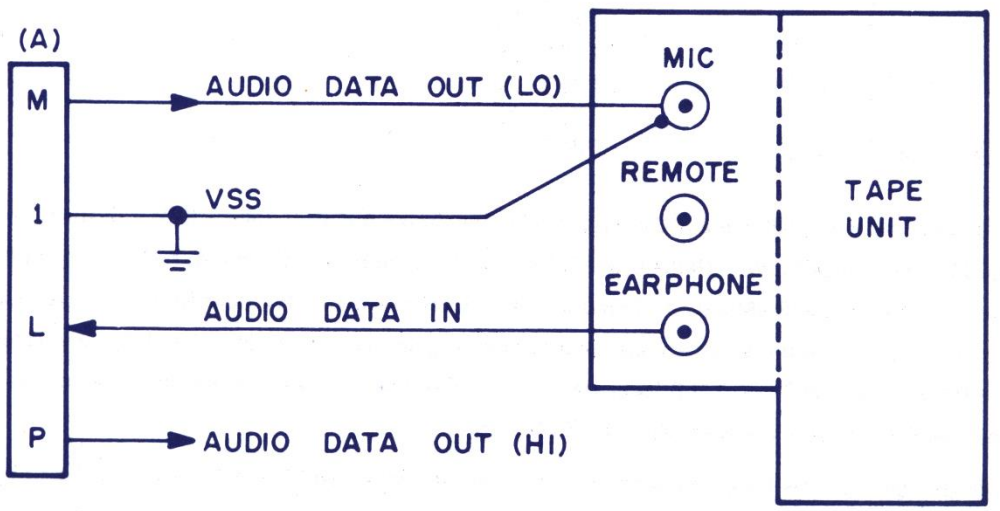
Die Kassette mit den aufgezeichneten Daten kann so oft gelesen werden, wie Sie es wünschen. Dieser Abschnitt behandelt Anschluß und Betrieb des Kassetten-Rekorders. Durch die vom KIM-1 System verwendete Aufzeichnungstechnik kann praktisch jeder Kassetten-Rekorder angeschlossen werden. Das System wurde mit einem Billigstgerät (< 20 $) aus einem Discountgeschäft getestet und arbeitete einwandfrei. Es ist auch möglich, zum Rücklesen der Daten einen anderen Rekorder zu benutzen als zum Aufzeichnen. Natürlich empfehlen wir Ihnen, den besten Kassetten-Rekorder und die besten Bänder zu verwenden, welche Sie zur Verfügung haben.

Wenn Sie einen Kassetten-Rekorder für den Einsatz im KIM-1 System neu anschaffen, so sollte dieser folgende Eigenschaften besitzen:

1. Eine Kopfhörer-Buchse, um die aufgezeichneten Daten in das KIM-1 System zurücklesen zu können.
2. Eine Mikrophon-Buchse zum Aufzeichnen der Daten vom KIM-1 System
3. Steuertasten für die Funktionen: Wiedergabe, Aufnahme, Rückspulen und Stop.

Verwenden Sie keine Diktiergeräte bei denen Mikrophon und Lautsprecheranschlüsse nicht herausgeführt sind. Solche Geräte müssen zum Einsatz im KIM-1 System erst umgebaut werden.

Um Ihr Magnetbandgerät an den KIM-1 Modul anzuschließen, schalten Sie zunächst die Versorgungsspannungen aus und entfernen Sie die Buchsenleiste A von dem Stecker am Modul. Verdrahten Sie diese wie in Bild 2.3 angegeben.



Anschluß der Magnetbandeinheit
Bild 2.3

Halten Sie die Anschlußdrähte so kurz wie möglich und vermeiden Sie es, diese in die Nähe von Leitungen zu bringen, welche größere Ströme führen. Die gezeigte Anschlußart gilt für tragbare Geräte. Das Nf-Ausgangssignal (LO) besitzt einen Scheitelwert von etwa 15 mV. Möchten Sie ein größeres Magnetbandgerät einsetzen, dann sollten Sie den Ausgangspin P mit dem "LINE"-Eingang dieses Gerätes verbinden. Der Ausgang P liefert ein Signal mit etwa 1 Volt Scheitelwert.

Stecken Sie die Buchsenleiste A wieder auf den KIM-1 Modul und schalten Sie die Versorgungsspannungen ein. Die folgende Prozedur prüft, ob die Zusammenschaltung richtig arbeitet.

1. Laden Sie das Programm aus dem letzten Kapitel neu und testen Sie es, damit Sie sicher sind, daß alles noch funktioniert.
2. Legen Sie dann eine Kassette ein und betätigen Sie die Rückspultaste.
3. Legen Sie eine Anfangs- und eine Endadresse für das zu speichernde Programm fest und fügen Sie eine Kennummer (ID) hinzu. Das geschieht folgendermaßen:

Drücke Taste	Anzeige	Schritt Nr.
AD	xxxx xx	1
0 0 F 1	00F1 xx	2
DA 0 0	00F1 00	3
AD	00F1 00	4
1 7 F 5	17F5 xx	5
DA 0 0	17F5 00	6
+ 0 0	17F6 00	7
+ 1 0	17F7 10	8
+ 0 0	17F8 00	9
+ 0 1	17F9 01	10
AD	17F9 01	11
1 8 0 0	1800 xx	12

Sie werden sich erinnern, daß das auf Band zu speichernde Programm im RAM des KIM-1 Systems an den Stellen 0000 bis 000F abgelegt war. Deshalb wählen wir die Anfangsandresse für die Aufzeichnung mit 0000 und laden diesen Wert in die RAM-Zellen 17F5 und 17F6 (Schritte 4 bis 7). Wir definieren die Endadresse für die Aufzeichnung, und zwar um eins mehr als unser letzter Programmschritt (000F + 1 = 0010) und laden diese in die RAM-Zellen 17F7 und 17F8 (Schritte 8 und 9). Dann nehmen wir eine beliebige Kennziffer (ID) z.B. 01 und speichern diese in die RAM-Zelle 17F9 (Schritt 10).

Die Anfangsadresse für das Lade-Programm ist 1800. Mit den Schritten 11 und 12 laden wir diesen Wert in das System. Würde man jetzt die Taste GO drücken, so könnte die Übertragung der Daten vom System auf das Magnetband beginnen. Aber wir müssen zuvor noch das Bandgerät starten!

4. Wählen Sie den Aufnahmemodus am Bandgerät und starten Sie das Band. Warten Sie einige Sekunden, drücken Sie jetzt die Taste GO.
5. Die Anzeige wird für eine kurze Zeit dunkel – wenn sie wieder aufleuchtet erscheint: 0000 xx.
6. Sobald die Anzeige wieder aufleuchtet ist die Übertragung beendet, und Sie können Ihren Kassettenrekorder anhalten.

Nun müssen Sie sich davon überzeugen, daß die Aufzeichnung fehlerfrei verlaufen ist. Das geschieht durch Lesen des eben aufgezeichneten Bandabschnitts. Gehen Sie dabei wie folgt vor:

1. Rückspulen des Bandes auf Startposition.
2. Schalten Sie die Versorgungsspannungen des Systems aus und nach einiger Zeit wieder ein.

Dieser Vorgang bewirkt, daß Ihr vorher in RAM gespeichertes Programm verloren geht.

3. Bereiten Sie das System zum Lesen des Bandes wie folgt vor:

Drücke Taste	Anzeige	Schritt Nr.
RS		
AD	xxxx xx	1
0 0 F 1	00F1 xx	2
DA 0 0	00F1 00	3
AD	00F1 00	4
1 7 F 9	17F9 xx	5
DA	17F9 xx	6
0 1	17F9 01	7
AD	17F9 01	8
1 8 7 3	1873 xx	9
GO	(Dark)	10

Das KIM-1 System wartet nun auf die Eingabe von Daten mit der Kennziffer 01. Erinnern Sie sich: dies ist die Kennziffer, welche wir vor der Datenaufzeichnung in den Speicher eingegeben haben.

4. Wenn Ihr Bandgerät einen Lautstärkeregeler besitzt, so stellen Sie ihn etwa auf die Hälfte.
5. Wenn Ihr Gerät eine Tonblende hat, dann drehen Sie diese voll auf.
6. Starten Sie jetzt das Band im Wiedergabemodus. Während das Band abläuft, übernimmt das System die aufgezeichneten Daten. Wenn das letzte Zeichen gelesen ist (Kennziffer 01), bringt die Anzeige den Wert 0000 xx.

Sie können das Band jetzt anhalten. Zeigt aber das Display den Wert

FFFF xx

so bedeutet das, daß beim Einlesen der Daten ein Fehler aufgetreten ist, die Kennziffer wurde jedoch erkannt. Drücken Sie in diesem Fall die Taste RS und wiederholen Sie den Einlesevorgang. Tritt danach immer noch FFFF xx auf, so wiederholen Sie die ganze Aufzeichnung und prüfen Sie jeden Schritt sorgfältig. Tritt der Fehler trotzdem auf, schlagen Sie in Anhang C nach.

Läuft das Band weiter, ohne daß das Display wieder aufleuchtet, so bedeutet dies, daß vom System keine Daten erkannt wurden. Wiederholen Sie auch in diesem Fall den gesamten Aufzeichnungs- und Wiedereinlesevorgang, indem Sie jeden Schritt sorgfältig prüfen. Läßt sich der Fehler nicht ausschalten, schlagen Sie in Anhang C nach.

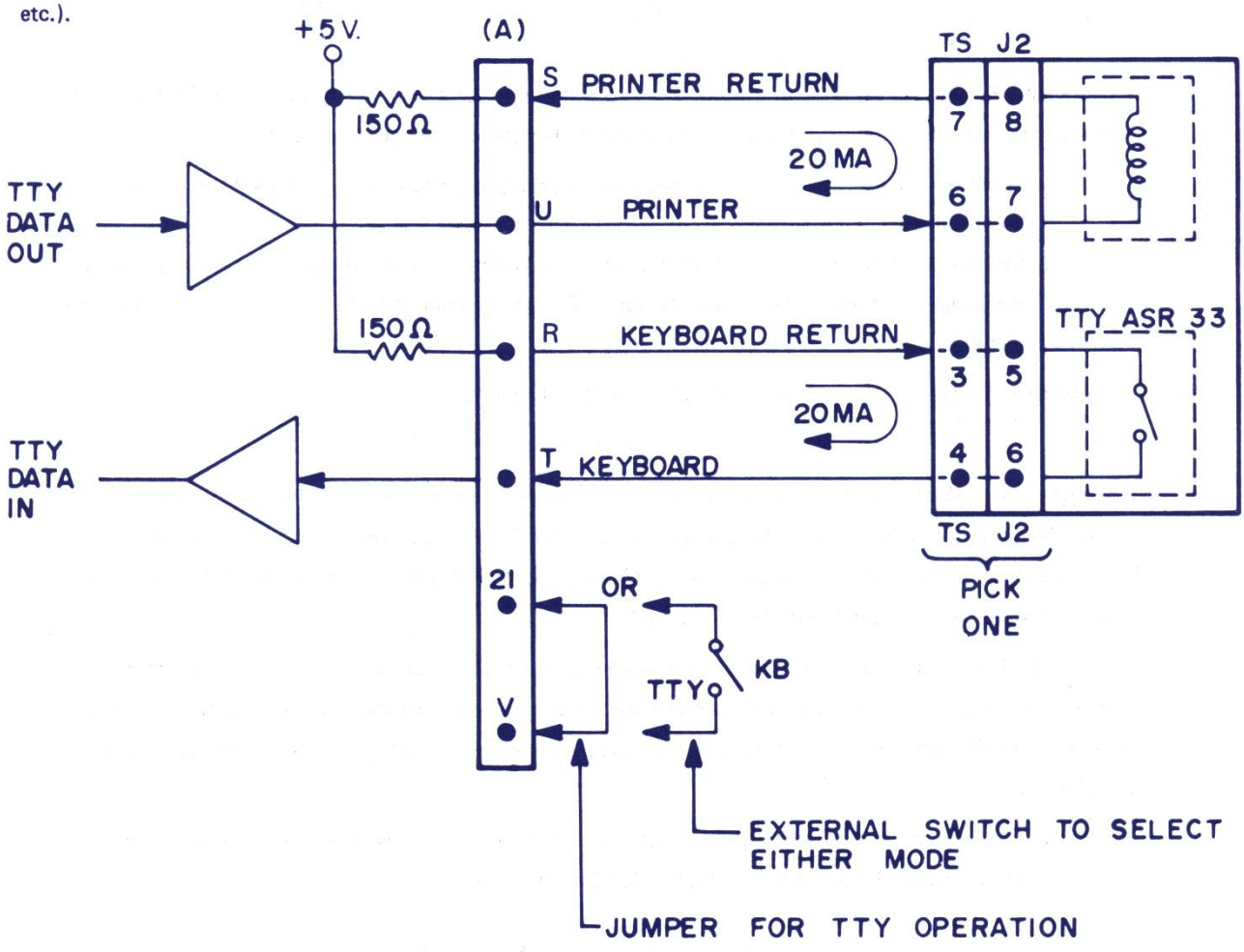
6. Angenommen der Wiedereinlesevorgang war erfolgreich, so überzeugen Sie sich davon, indem Sie ein einfaches Beispiel ausprobieren (z.B. 02 + 03 = 05).

Beachten Sie: Die Interface-Schaltung des KIM-1 Systems für den Magnetbandanschluß wird nicht abgeglichen. Wenn Sie der Beschreibung gefolgt sind, muß das System richtig arbeiten.

2.6 Anschluß eines 8-Kanal Fernschreibers

Steht Ihnen ein 8-Kanal Fernschreiber mit Serienschnittstelle zur Verfügung, so können Sie dieses Gerät mit geringstem Aufwand an Ihr KIM-1 System anschließen. Der bekannteste Vertreter solcher Geräte ist das Teletype Modell 33ASR. Wir beziehen uns in diesem Abschnitt immer auf dieses Gerät. Wenn Sie ein ähnliches Gerät besitzen, so können Sie die hier gemachten Ausführungen zum Anschließen trotzdem benutzen. Auf alle Fälle sollten Sie die Angaben des Herstellers beachten, wenn Sie Ihre Fernschreibmaschine anschließen.

Der KIM-1 Modul ist für eine Vierdraht-Schnittstelle ausgelegt. Es sollte ein Schleifenstrom von 20 mA verwendet werden. Prüfen Sie, ob Ihr Gerät entsprechend geschaltet ist. Wenn nicht, so können Sie dieses leicht von 60 mA auf 20 mA Schleifenstrom umstellen; befolgen Sie die Herstellerangaben! Ihr KIM-1 kann in beiden Betriebsarten (Halb- oder Vollduplex) verwendet werden. Auch sind Sie nicht auf eine bestimmte Übertragungsgeschwindigkeit festgelegt. Das KIM-1 System paßt sich automatisch an (10 Zeichen/sek 15, 30 etc.).



TTY-Anschlüsse
Bild 2.4

Um ein Teletype (TTY) an das System anzuschließen, verfahren Sie wie folgt:

1. Schalten Sie die Versorgungsspannungen ab und entfernen Sie die Buchsenleiste A von Ihrem Modul.
2. Führen Sie die in Bild 2.4 gezeigten Verbindungen zwischen der Buchsenleiste A und dem entsprechenden Stecker am TTY aus.
3. Die Brücke zwischen A-21 und A-V sagt dem KIM-1 System, daß nur das TTY als Ein/Ausgabegerät angeschlossen ist. Wenn Sie außer dem TTY auch noch mit dem Tastenfeld und der LED-Anzeige arbeiten wollen, installieren Sie anstelle einer Drahtbrücke einen Schalter. Bei geöffneten Schalterkontakten können Sie nun mit dem Tastenfeld und der LED-Anzeige arbeiten, bei geschlossenen Schalterkontakten mit dem TTY.
4. Überzeugen Sie sich davon, daß die Verbindung A-21 nach a-V vorliegt. Stecken Sie die Buchsenleiste A wieder auf den Modul, schalten Sie die Versorgungsspannungen des Systems und das TTY ein.
5. Drücken Sie als erstes die RS-Taste auf Ihrem KIM-1 Modul, dann die Taste "RUB OUT" am TTY. Dieser Schritt ist sehr wichtig, da sich das KIM-1 System automatisch an die Übertragungsgeschwindigkeit der Serien-Schnittstelle anpassen muß. Das System benötigt diesen ersten Tastendruck, um sich einstellen zu können.

Wenn kein Fehler vorliegt, muß sofort am TTY folgende Ausgabe erscheinen:

KIM

Dieser Text sagt Ihnen, daß das TTY in Betrieb ist und das KIM-1 System bereit ist, Daten vom TTY-Tastfeld zu empfangen.

Sollte dieser Text nicht erscheinen, dann drücken Sie nochmals die RS-Taste am KIM-1 Tastenfeld und anschließend die Taste "RUB OUT" am TTY. Kommt immer noch keine Ausgabe "KIM", so überprüfen Sie alle Verbindungen, auch das TTY. Wiederholen Sie den Vorgang. Läßt sich der Fehler nicht beheben, so schlagen Sie im Anhang C nach.

6. Wenn das TTY betriebsbereit ist, so kann durch einige einfache Befehle die ordnungsgemäße Funktion des Systems geprüft werden.

Drücke Taste	Ausdruck	Schritt Nr.
	KIM	
	xxxx xx	1
0 0 0 2	0002	2
SPACE	0002 xx	3
1 8 •	18.	4
	0003 xx	5
A 5 •	A5.	6
	0004 xx	7
LF	0003 A5	8
RUB OUT	KIM	
	xxxx xx	9

In Schritt 1 steht noch das Wort "KIM" am Papier. Im Schritt 2 wird eine Adresse (0002) angewählt und im Schritt 3 eine Leertaste eingegeben. Daraufhin wird die eben eingegebene Adresse nocheinmal zusammen mit den an dieser Stelle gespeicherten Daten (xx) ausgedruckt. Schritt 4 zeigt den Befehl "Modifiziere Speicher". Die vor der Taste "·" eingegebene Hexazahl wird in die adressierte Speicherzelle eingeschrieben (18). In Schritt 5 sieht man, daß die Speicheradresse nach Drücken der Taste "·" um 1 erhöht wird (0003). Die Schritte 6 und 7 zeigen, daß die Daten in Zelle 0003 verändert werden, und das Fortschalten der Adresse nach 0004. Schritt 8 zeigt die Wirkung der Taste "LF". Die Adresse wird um eins zurückgesetzt und ein Ausdruck gebracht, der zeigt, daß die Eingabe von A5 in die Zelle 0003 korrekt erfolgt ist. Schritt 9 erklärt, wie das System auf die Taste "RUB OUT" reagiert: Das System wird in Grundstellung gebracht und der Text "KIM" ausgegeben. Beachten Sie, daß in diesem Beispiel ein Teil des Programms aus Kapitel 2.4 wiederholt wurde; allerdings erfolgte die Eingabe diesmal über das TTY.

Wenn alle Operationen fehlerfrei verlaufen sind, können Sie sicher sein, daß Ihr KIM-1 Modul mit dem TTY korrekt zusammenarbeitet. Alle weiteren TTY-Operationen werden in einem späteren Kapitel noch genauer beschrieben.

Wenn sich bis zu dieser Stelle keine Probleme ergeben haben, dann ist das System voll funktionsfähig. Unsere nächste Aufgabe wird es sein, mehr über das KIM-1 System und die Operationsprogramme zu lernen.

KAPITEL 3

DAS KIM-1 SYSTEM

Bis jetzt waren Sie damit beschäftigt Ihr KIM-1 System in Betrieb zu setzen und sich von dessen ordnungsgemäßer Funktion zu überzeugen. Nun ist es an der Zeit mehr über die verschiedenen Elemente des KIM-1 zu erfahren, wie sie im System zusammenarbeiten und wie die Steuerprogramme die einzelnen Systemkomponenten bedienen. Die in diesem Abschnitt angegebenen Diagramme, zusammen mit dem System-Blockschaltbild werden Ihnen helfen, die Einzelelemente Ihres KIM-1 Moduls zu verstehen.

3.1 KIM-1 System Beschreibung

Bild 3.1 zeigt ein komplettes Blockschaltbild des KIM-1 Systems. Das wichtigste Bauteil ist ein MCS 6502 Mikroprozessor, der als zentrales Steuerelement verwendet wird. Dieser 8-Bit Prozessor bedient die übrigen Systemkomponenten über drei voneinander unabhängigen Busleitungen. Ein 16-Bit Adress-Bus gestattet den direkten Zugriff auf einen Speicherbereich von 64 KBytes. Ein bidirektionaler 8-Bit Daten-Bus überträgt Daten vom MCS 6502 zu jeder Speicherzelle und von dort zurück in den Prozessor. Der dritte Bus ist der Steuerbus. Dieser übernimmt die Abwicklung des zeitlichen Ablaufs bei der Signalübertragung zwischen den einzelnen Systemkomponenten.

Unmittelbar neben dem MCS 6502 befindet sich ein 1 MHz Quarz, welcher mit einer auf dem Chip befindlichen Oszillatorschaltung zusammenarbeitet. Von diesem Quarzoszillator werden alle Zeitsignale des Systems abgeleitet. Das von dem 6502 erzeugte Signal $\emptyset_2$ wird allein oder mit anderen Steuersignalen von allen übrigen Bauelementen als Zeitbasis verwendet.

Der Mikroprozessor 6502 kann mit verschiedenen Speichertypen zusammenarbeiten. In dem KIM-1 System sind Schreib/Lesespeicher (RAM) und Nur-Lesespeicher (ROM) enthalten. Der ROM-Teil des Speichers enthält im wesentlichen die Systemsteuerprogramme. Beachten Sie die beiden mit 6530-002 und 6530-003 bezeichneten Bausteine. Jeder enthält 1024 Bytes ROM; dort ist das Steuerprogramm für das KIM-1 System gespeichert.

RAM stehen an drei verschiedenen Stellen des Speicherbereichs zur Verfügung. Zwei dieser Stellen sind wieder die 6530-Bausteine, von denen jeder 64 Bytes enthält. Diese werden vorwiegend zur Unterstützung der Steuerprogramme verwendet. Zusätzlich ist noch ein RAM-Block mit 1024 Byte Kapazität vorhanden. Dieser dient hauptsächlich dazu, Anwendungsprogramme und Daten zu speichern.

Ein/Ausgangs-Steuerleitungen (I/O) sind ebenfalls in den 6530-Bausteinen inbegriffen. Jeder enthält 15 I/O-Leitungen, wobei das Programm darüber entscheidet, ob diese als Ein- oder Ausgänge betrieben werden. Die I/O-Leitungen des 6530-002 sind für bestimmte Systemfunktionen verwendet, z.B. Tastenfeld, LED-Anzeige, TTY-Interface und Kassetten-Recorder Anschluß. Die 15 I/O-Leitungen des 6530-003 sind auf den Modulstecker herausgeführt und stehen dem Anwender uneingeschränkt zur Verfügung.

Schließlich enthält jeder MCS 6530-Baustein einen Interval-Timer, welcher mit Hilfe des System-Taktes exakte Zeitsignale erzeugt. Diese Zeitsignale sind vom Programm einstellbar. Der Timer des Bausteins 6530-003 wird nicht von den Dienstprogrammen benutzt; er steht dem Anwender uneingeschränkt zu Verfügung.

In Bild 3.1 ist ein Block gezeichnet, der mit Control-Logic bezeichnet ist. Dieser enthält auch einen Adressdekoder für Chip-Auswahlsignale der 6530 IC's und der statischen RAM's eine Entprellschaltung für die System-Reset und Stopptasten und nicht zuletzt eine Logik, die den Einzelschrittablauf des uComputers erlaubt, was sich speziell bei der Fehlersuche eines Programms sehr bewährt.

Außerdem zeigt das Bild 3.1 den Anschluß der Einheit Tastenfeld/LED-Anzeige an die I/O-Leitungen des Bausteins 6530-002 und die Interfaceschaltungen für das TTY und die Mangetband-Einheit.

Bild 3.2 zeigt im Detail die Verbindungen zwischen der CPU (MCS 6502) und den beiden MCS 6530-Bausteinen.

Bild 3.3 enthält eine Darstellung der Steuerlogik.

In Bild 3.4 sehen Sie den Aufbau des RAM-Speicherblocks.

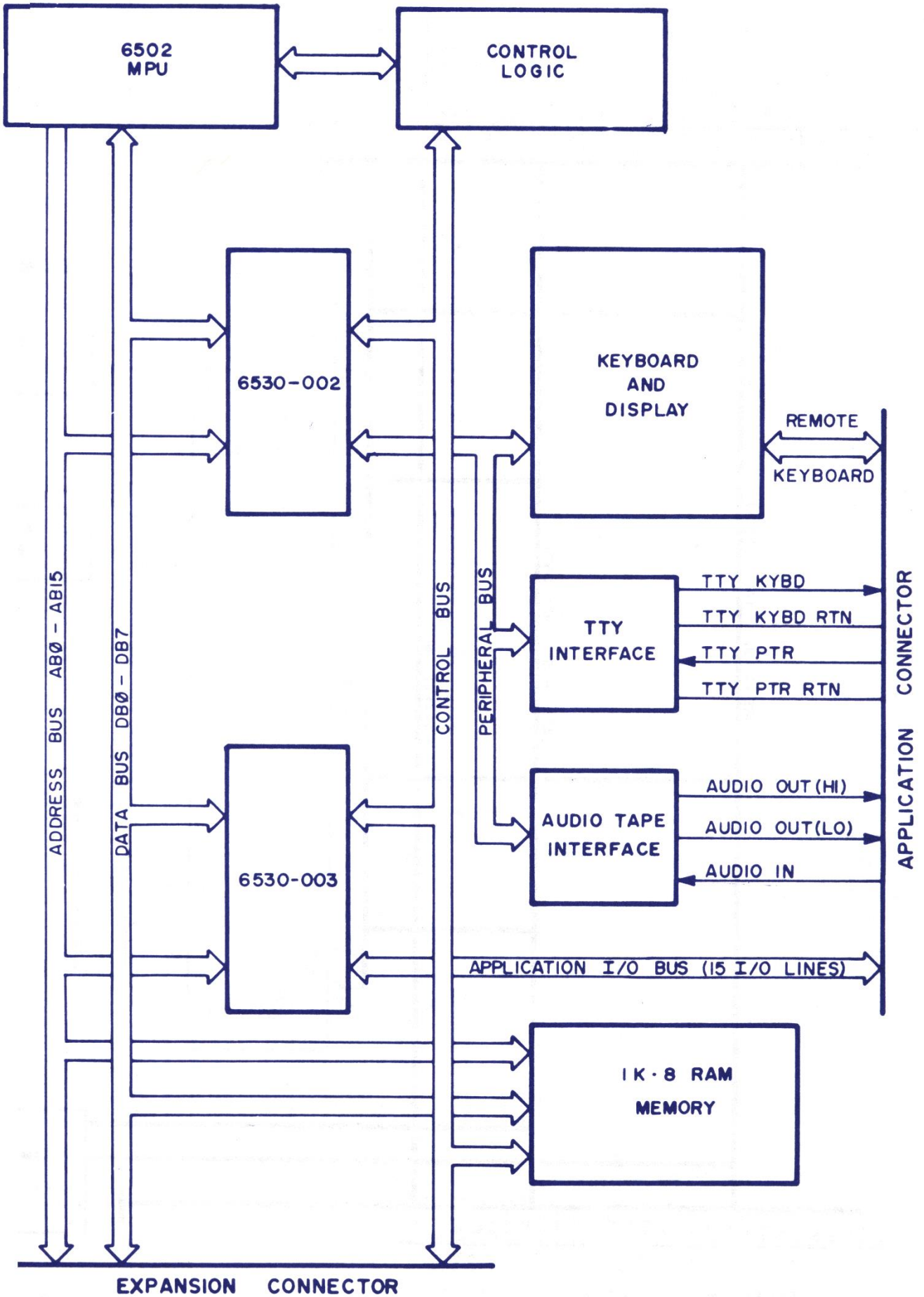
Die Bilder 3.5 und 3.6 zeigen die Schaltung des Tastenfeldes und der Display-Logik.

Bild 3.7 enthält die Schaltung des TTY-Interface und Bild 3.8 die des Magnetband-Interface.

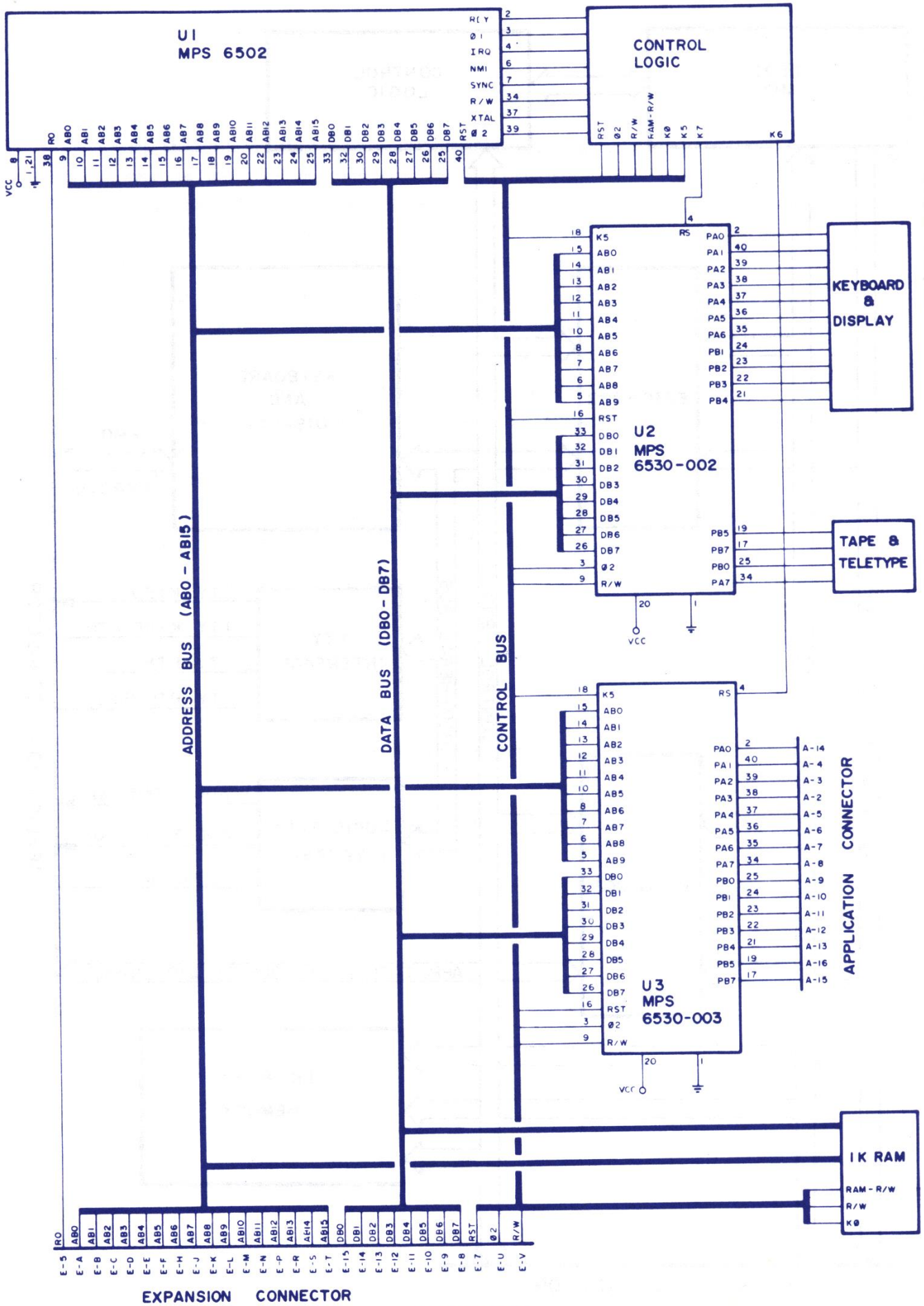
In den Bildern 3.9 und 3.10 ist die Belegung des Applikations- und des Expansions-Steckers wiedergegeben.

Der Übersichtsplan zeigt, wie alle Systemkomponenten miteinander verbunden sind sowie alle Signale, welche an den Steckern des Moduls herausgeführt wurden.

Wenn Sie ausführlichere Information über den MCS 6502 oder den MCS 6530 benötigen, dann schlagen Sie in dem Hardware-Handbuch nach, welches Ihrem KIM-1 Modul beiliegt.

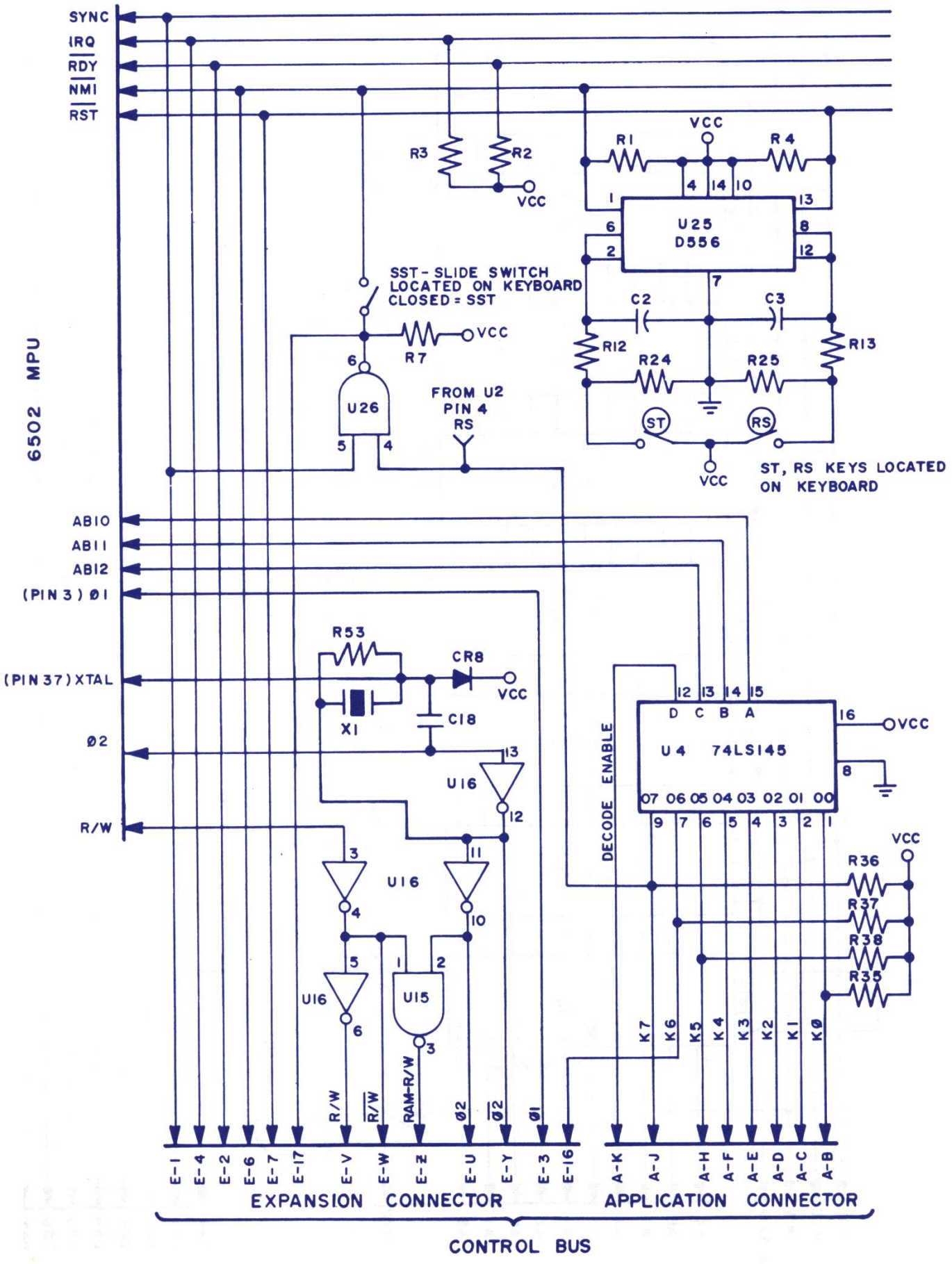


KIM-1 Blockschaltbild
Bild 3.1

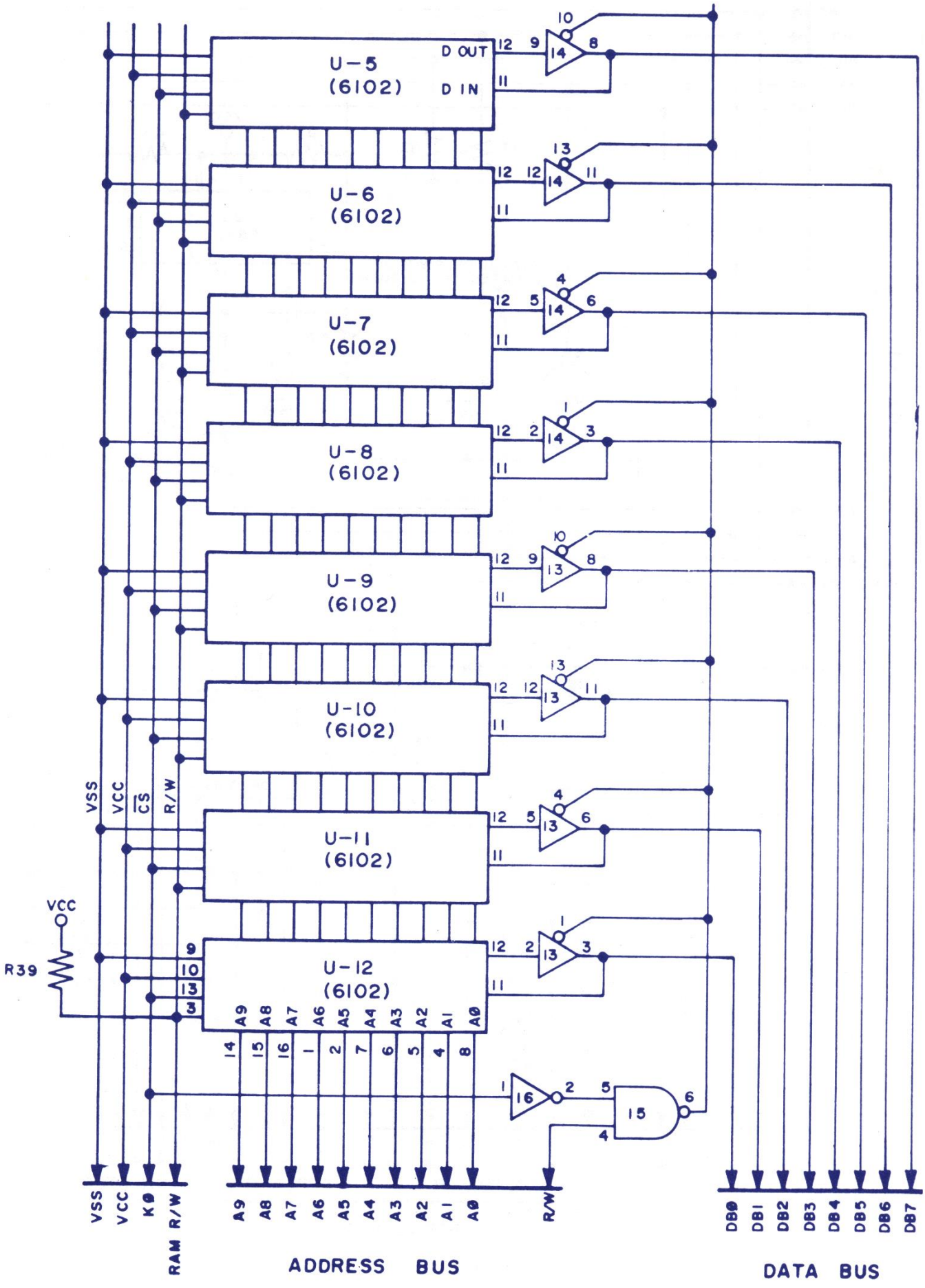


Blockschaltbild

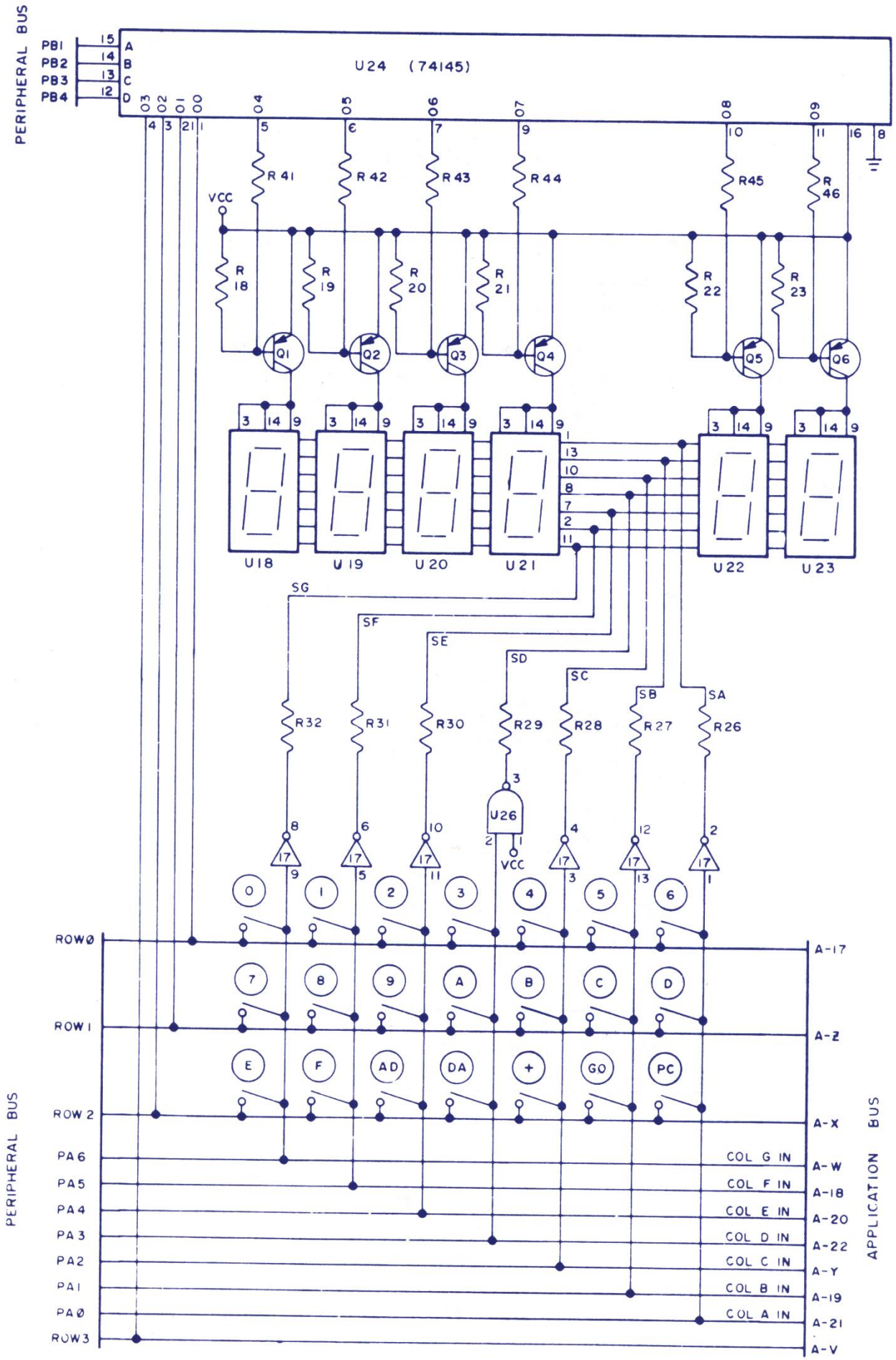
Bild 3.2



Zeitsteuerung
Bild 3.3

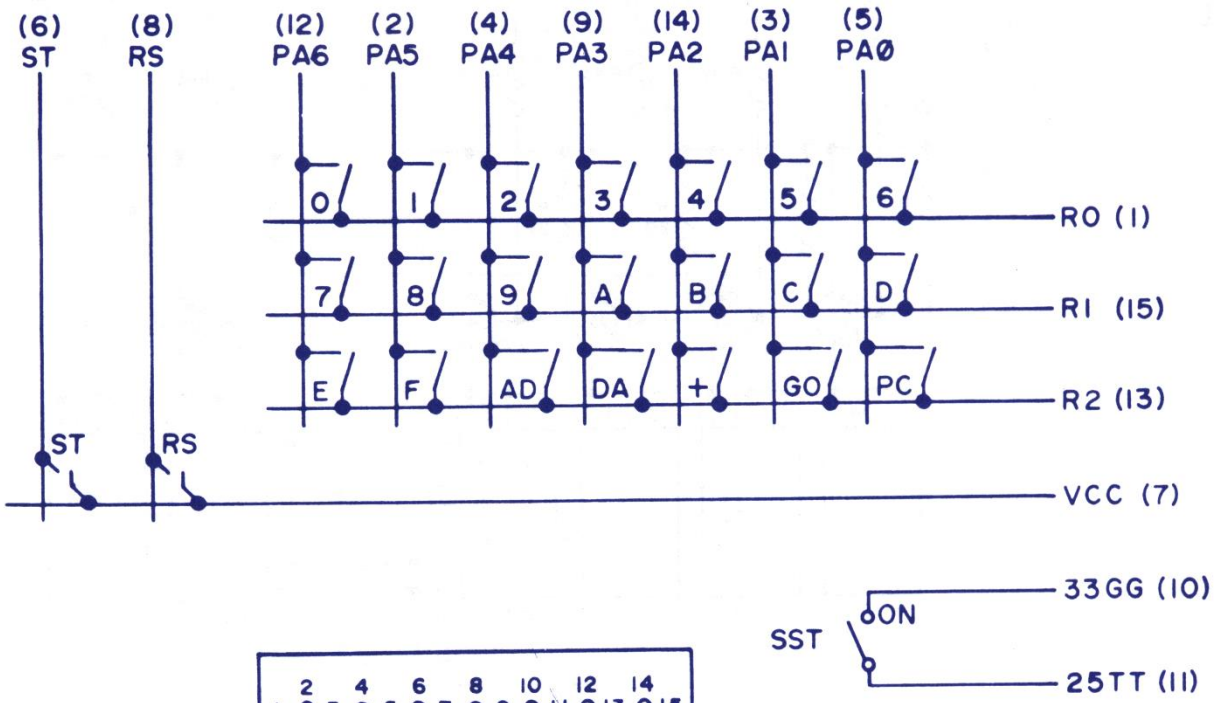


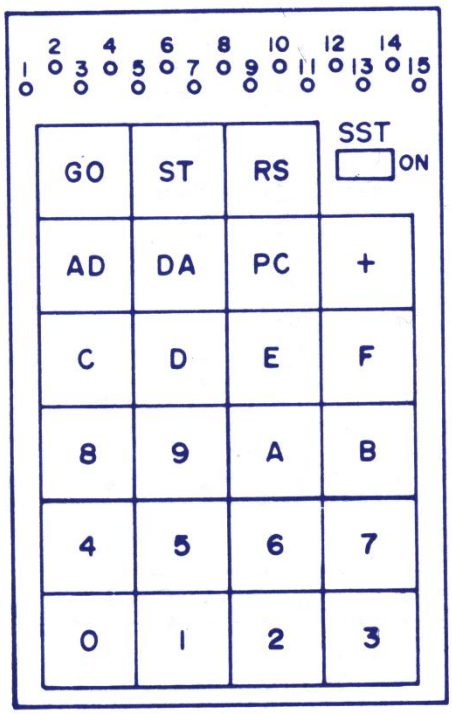
1Kx8 RAM
Bild 3.4



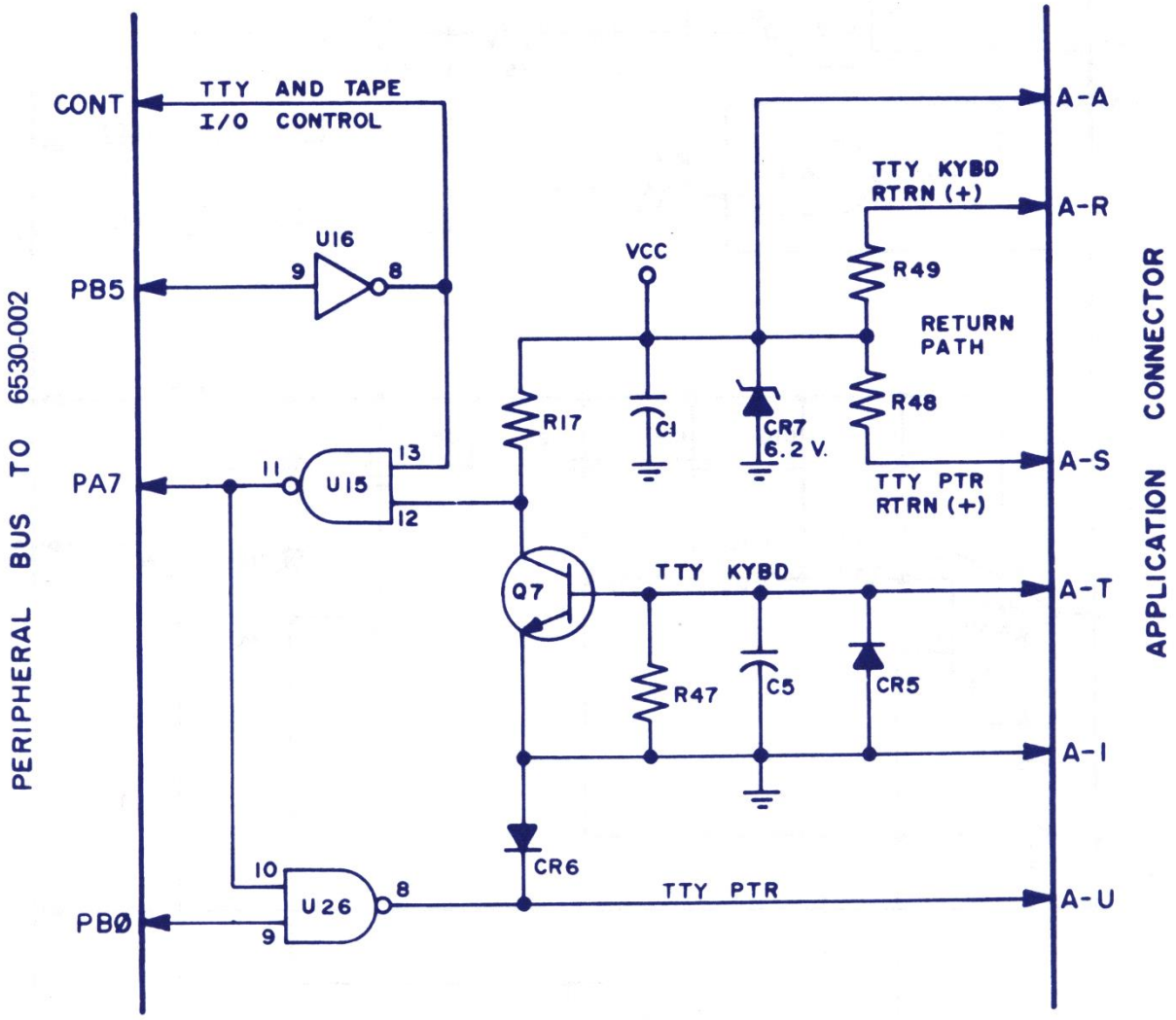
Tastenfeld und LED-Anzeige

Bild 3.5

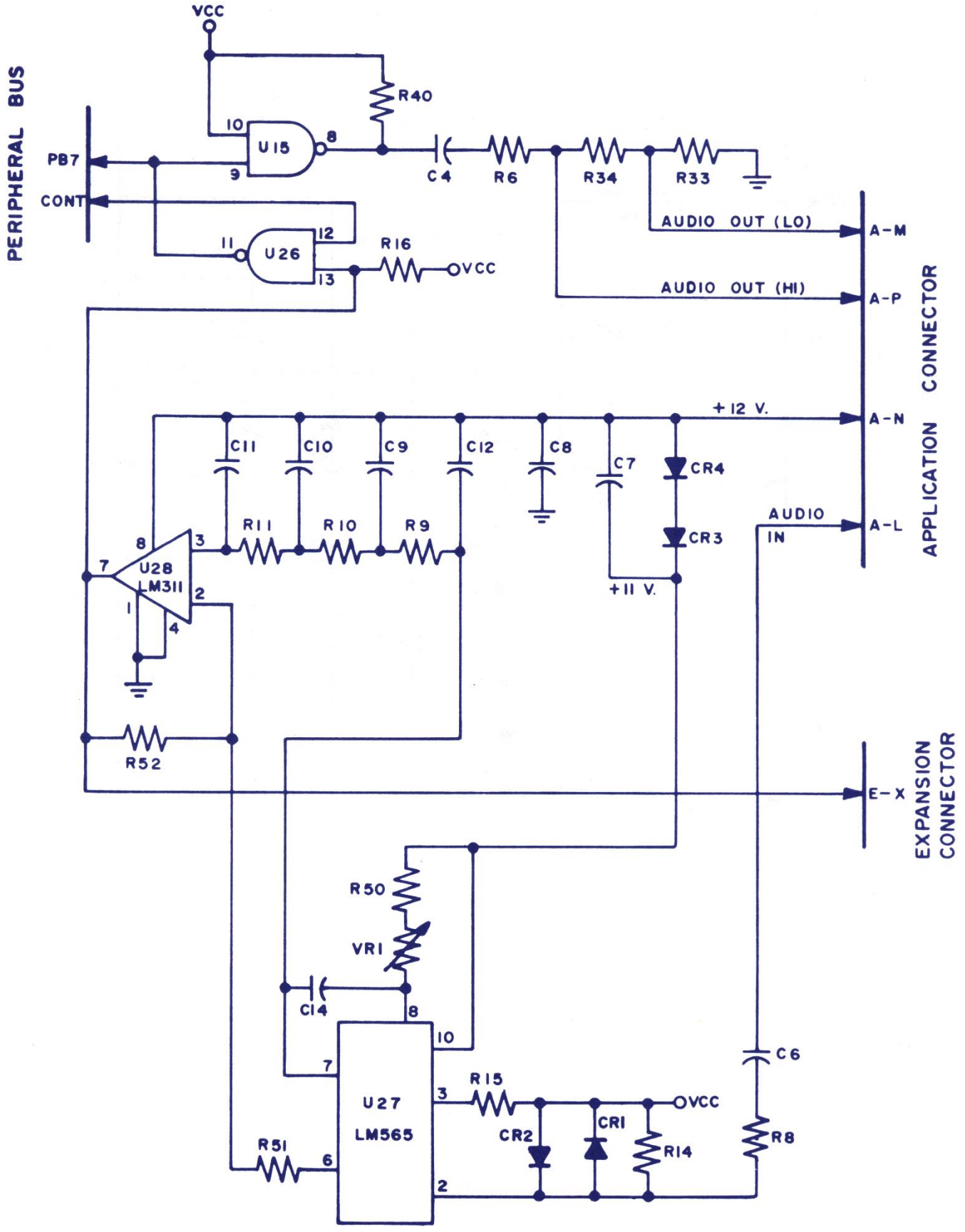




Tastenfeld im Detail
Bild 3.6



TTY-Interface
Bild 3.7



Magnetband-Interface
Bild 3.8

22	KB Col D
21	KB Col A
20	KB Col E
19	KB Col B
18	KB Col F
17	KB Row Ø
16	PB5
15	PB7
14	PAØ
13	PB4
12	PB3
11	PB2
10	PB1
9	PBØ
8	PA7
7	PA6
6	PA5
5	PA4
4	PA1
3	PA2
2	PA3
1	VSS GND

Z	KB Row 1
Y	KB Col C
X	KB Row 2
W	KB Col G
V	KB Row 3
U	TTY PTR
T	TTY KYBD
S	TTY PTR RTRN(+)
R	TTY KYBD RTRN(+)
P	AUDIO OUT HI
N	+12v
M	AUDIO OUT LO
L	AUDIO IN
K	DECODE ENAB
J	K7
H	K5
F	K4
E	K3
D	K2
C	K1
B	KØ
A	VCC +5v

Applikations-Stecker
Bild 3.9

22	VSS GND
21	VCC +5
20	
19	
18	
17	SST OUT
16	K6
15	DBØ
14	DB1
13	DB2
12	DB3
11	DB4
10	DB5
9	DB6
8	DB7
7	RST
6	NMI
5	RO
4	IRQ
3	Ø1
2	RDY
1	SYNC

Z	RAM/R/W
Y	$\overline{Ø2}$
X	PLL TEST
W	$\overline{R/W}$
V	R/W
U	Ø2
T	AB15
S	AB14
R	AB13
P	AB12
N	AB11
M	AB10
L	AB9
K	AB8
J	AB7
H	AB6
F	AB5
E	AB4
D	AB3
C	AB2
B	AB1
A	ABØ

Expansions-Stecker
Bild 3.10

3.2 KIM-1 Speicherbelegung

Wie bereits erwähnt, kann der im System verwendete Mikroprozessor MCS 6502 einen Speicherbereich von 64 kByte adressieren. Das KIM-1 System besitzt selbstverständlich nur einen Teil dieses Bereichs. In diesem Abschnitt wird gezeigt, welchen Speicherbereich das System enthält und welche Adressen ihm zugeordnet sind.

Jede Speicherzelle besteht aus 8 Bit (= ein Byte). Jede adressierbare Zelle in diesem System hat vier Funktionen:

1. Ein ROM-Byte; dort sind die Steuer- und Dienstprogramme gespeichert.
2. Ein RAM; Schreib-/Lese-Speicher zum Ablegen von Daten oder Anwenderprogrammen.
3. Ein I/O-Port; solche Zellen enthalten sog. Richtungs-Register welche jeden I/O-Pin zum Ein- oder Ausgang erklären, außerdem Datenbuffer zur Übertragung der gesendeten bzw. empfangenen Daten. Jede I/O-Zelle hat ihre eigene Adresse und wird wie eine RAM-Zelle behandelt.
4. Einen Timer-Bereich; eine Reihe von Adressen sind für die internen Timer reserviert. Sie können Daten an die Timer senden und damit die Verzögerungszeit einstellen, oder den aktuellen Zählerstand lesen.

Bild 3.11 zeigt alle Speicherblöcke des KIM-1 Systems. Bild 3.12 enthält eine Speicherliste, welche alle verwendeten Speicherzellen des Systems und deren Lage zueinander zeigt.

Beachten Sie die für Speichererweiterungen vorgesehenen Bereiche.

Kapitel 6 befaßt sich eingehend mit der Speichererweiterung. Schließlich enthält Bild 3.13 eine Zusammenstellung aller wichtigen Speicherzellen. Beim Schreiben Ihrer eigenen Programme werden Sie häufig dort nachschlagen.

Der in Bild 3.12 dargestellte Speicherbelegungsplan stellt einen Block von 8192 Speicherzellen dar, die alle im unteren Teil des 64 K-Bereichs liegen. Dieser Block ist noch weiter unterteilt, und zwar in Unterblöcke zu je 1024 Speicherzellen. Jeder Unterblock besteht aus 4 Pages zu je 256 Speicherzellen. Die mit dem Buchstaben K bezeichneten Unterblöcke beziehen sich auf den in der Logik enthaltenen Adressdekoder. Der Name Page kennzeichnet eine bestimmte Gruppe von 256 Adressen. Für einige Adressen ist an den wichtigen Stellen des Bereichs der zugehörige Hexakode angegeben.

Wenn Sie das obere Ende des Speicherblocks betrachten, sehen Sie, daß der erste Unterblock (K7) dem ROM 6530-002 und der zweite (K6) dem ROM 6530-003 zugeordnet ist. Das gesamte Stuerprogramm des KIM-1 Systems ist in diesen beiden Unterblöcken gespeichert.

Ein Teil des folgenden Unterblocks (K5) wird von den RAM's, den I/O-Ports und den Timern der beiden 6530-Bausteinen belegt. Eine genauere Aufteilung dieses Bereichs gibt das Detailbild daneben. Beachten Sie, daß die RAM-Adressen des 6530-002 (Hexaadressen 17EC bis 17FF) von dem System-Steuerprogramm benutzt werden; sie dürfen in keinem Anwenderprogramm erscheinen. Das Gleiche gilt für die Adressen der I/O-Ports und des Timers im Baustein 6530-002.

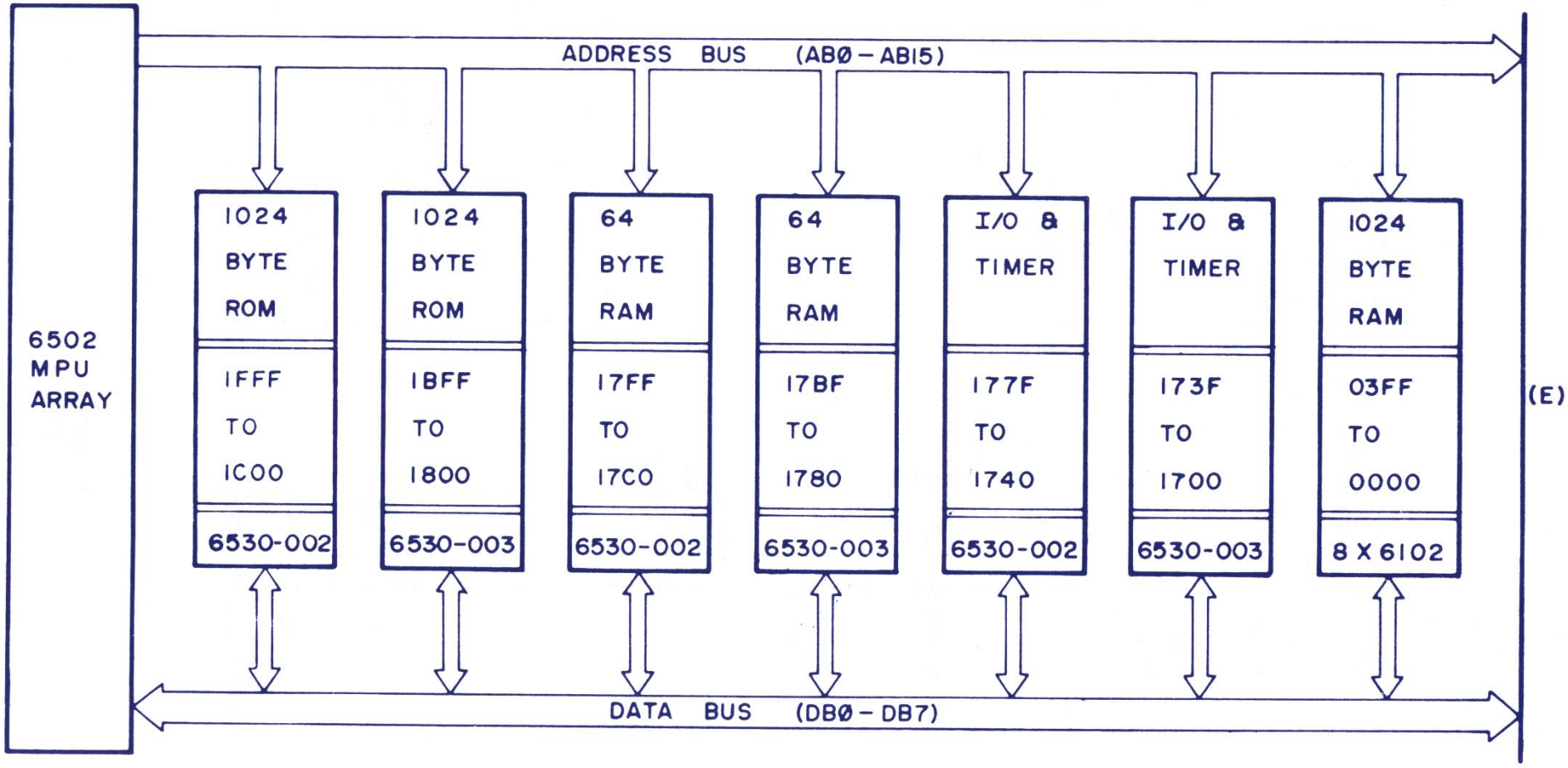
Die vier nächsten Unterblöcke (K4, K3, K2 und K1) sind für zusätzliche Speicher in einem erweiterten System vorgesehen. Im Kapitel 6 wird die Speichererweiterung behandelt.

Der niedrigste Unterblock (KO) ist mit statischen RAM's bestückt, die das KIM-1 System enthält. Beachten Sie, daß innerhalb dieses Bereichs Page 0 und Page 1 eine besondere Bedeutung besitzen. Page 1 wird als System-Stack benutzt; dort werden bei der Ausführung von Unterprogrammen die Return-Adressen aufbewahrt und im Fall von Interrupts der Maschinenstatus zwischengespeichert. Page 0 hat eine besondere Bedeutung für eine bestimmte Adressierungsart des 6502-Mikroprozessors.

Bild 3.12 bezeichnet den Bereich von Page 0 und Page 1 genauer. Beachten Sie, daß 17 Speicherzellen (00EF bis 00FF) noch von dem Systemsteuerprogramm mitbenutzt werden. Diese RAM-Zellen sind nicht mit Anwenderprogrammen zu belegen. Auch sollte für den Stack der entsprechende Platz freigehalten werden. Acht Zellen benötigt das Systemsteuerprogramm, um Interrupts zu bedienen.

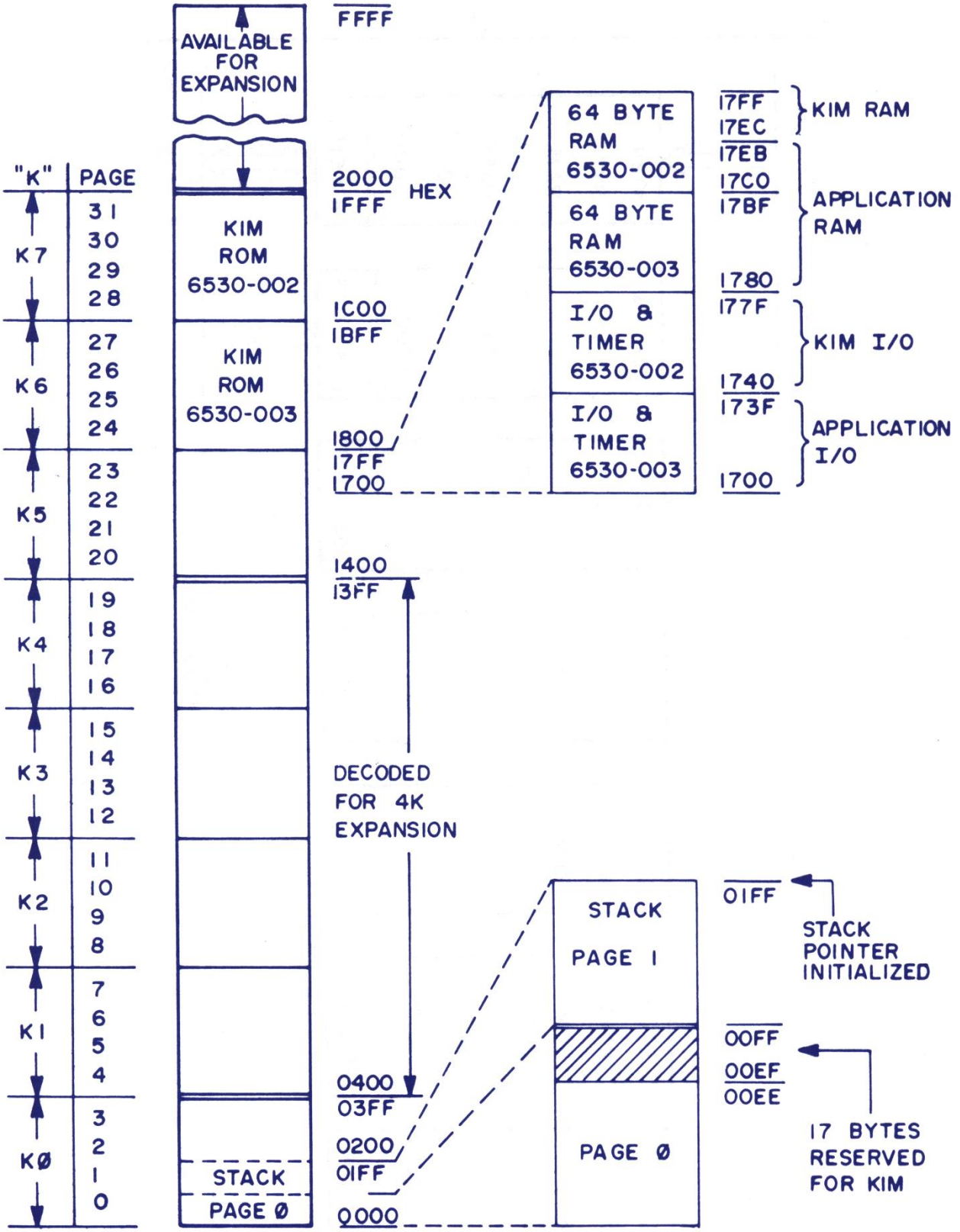
Im Folgenden noch eine Zusammenstellung der Speicherbereiche, welche der Anwender belegen darf:

1. Page 0 ganz, bis auf 00EF bis 00FF
2. Page 1 ganz, bis auf 8 Byte (für Stack)
3. Page 2 und Page 3 ohne Einschränkung
4. Page 23:
   Alle I/O-Ports von 1700 bis 173F
   Alle 64 Bytes RAM von 1780 bis 17BF
   Zusätzlich noch 44 Bytes RAM von 1700 bis 17EB



Speicher-Block Schaltbild

Bild 3.11



Speicherbelegung

Bild 3.12

ADRESSE	FELDBEZEICHNUNG	SYMBOL	FUNKTION
00EF	↑	PCL	Programmzähler-niederes Byte
00F0		PCH	Programmzähler-höheres Byte
00F1	Maschinen-	P	Status Register
00F2	Register	SP	Stack Pointer
00F3	Speicher	A	Akkumulator
00F4		Y	Index-Register X
00F5	↓	X	Index-Register Y
1700	↑	PAD	6530-003 A Daten Register
1701	Application	PADD	6530-003 A Datenrichtungs-Register
1702	I/O	PBD	6530-003 B Daten Register
1703	↓	PBDD	6530-003 B Datenrichtungs-Register
1704	↑		6530-003 Interval Timer
↓	Interval Timer		s. Hardware-Handbuch Kapitel 1.6
170F	↓		
17F5	↑	SAL	Anfangsadresse-niederes Byte
17F6	Magnetband	SAH	Anfangsadresse-höheres Byte
17F7	Load & Dump	EAL	Endadresse-niederes Byte
17F8		EAH	Endadresse-höheres Byte
17F9	↓	ID	Kenn-Nummer (Band)
17FA	↑	NMIL	NMI-Vektor niederes Byte
17FB		NMIH	NMI-Vektor höheres Byte
17FC	Interrupt	RSTL	RST-Vektor niederes Byte
17FD	Vectors	RSTH	RST-Vektor höheres Byte
17FE	↓	IRQL	IRQ-Vektor niederes Byte
17FF		IRQH	IRQ-Vektor höheres Byte
1800	↑ Magnetband ↓	DUMPT	Anfangsadresse Band Lesen
1873		LOADT	Anfangsadresse Band Schreiben
1C00	↑ Stoptaste und SST ↓		Startadresse für NMI Maschinenstatus retten (Laden nach 17 FA und 17 FB)
17F7	↑ Lochstreifen Lesen (Q) ↓	EAL	Endadresse niederes Byte
17F8		EAH	Endadresse höheres Byte

Spezielle Speicheradressen

Bild 3.13

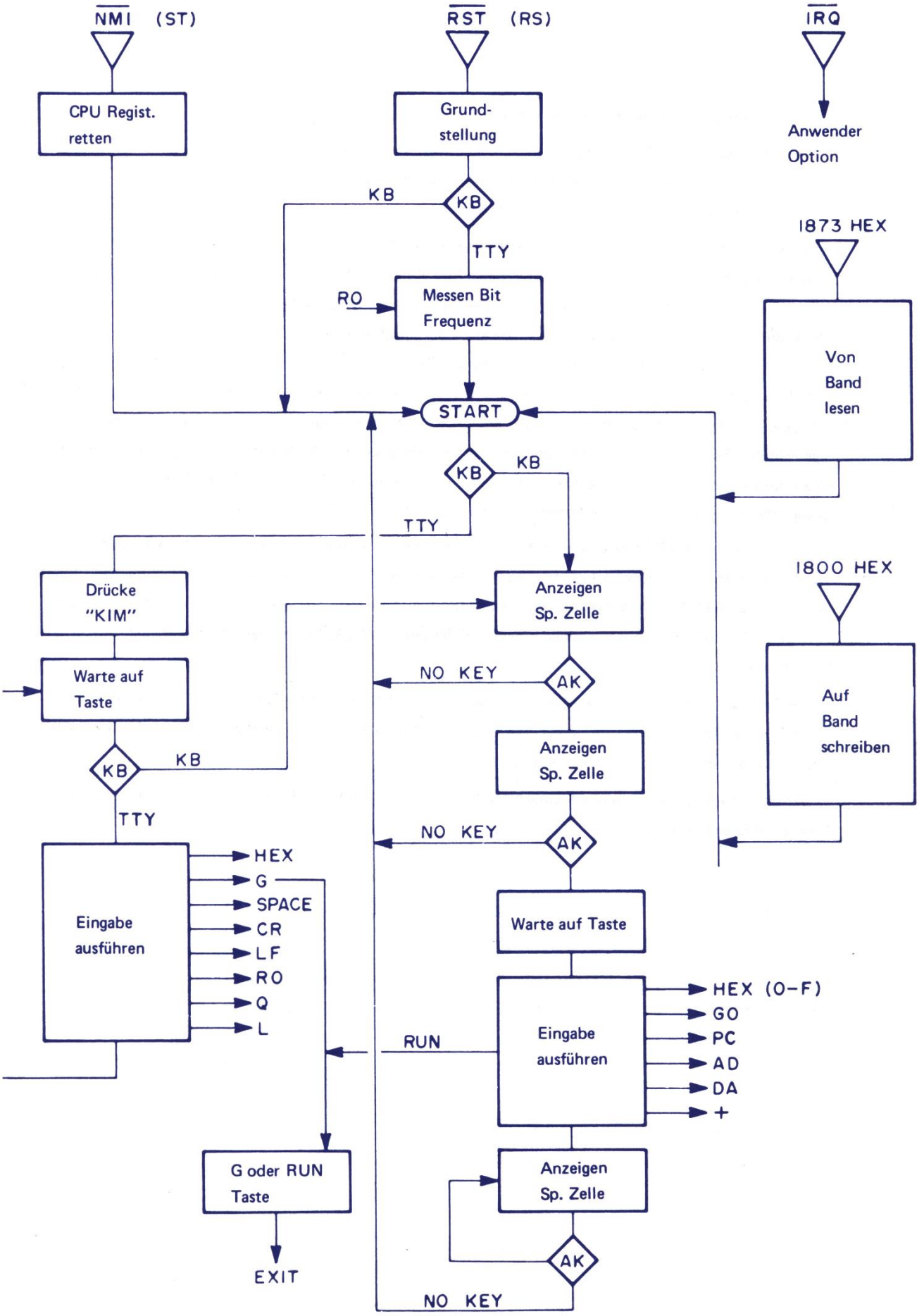
3.3 KIM-1 Steuerprogramme (Dienst-Programme)

Bild 3.14 zeigt ein vereinfachtes Flußdiagramm der KIM-1 Steuerprogramme. Dieser Abschnitt gibt Erklärungen, damit Sie die verschiedenen Betriebsarten des Systems verstehen. Wenn Sie die Versorgungsspannungen des Systems einschalten und die Taste RESET (RS) drücken, wird automatisch das System-Steuerprogramm aufgerufen. Das gilt immer, wenn Sie die RESET-Taste drücken.

Bei jedem Drücken der RESET-Taste wird das System in Grundstellung gebracht. Das bedeutet, daß der Stack-Pointer eingestellt wird, die Ports als Ein- oder Ausgang deklariert und bestimmte Flags gesetzt werden. Als Nächstes entscheidet das Programm, ob das System mit dem TTY oder mit dem Tastenfeld und Display auf dem Modul zusammenarbeiten soll.

Wurde die Betriebsart TTY gewählt, so wartet das Programm auf einen ersten Tastendruck am TTY (Taste RUBOUT). Wurde das Drücken dieser Taste erkannt, so nimmt das Programm automatisch eine Messung der Übertragungsgeschwindigkeit vor und speichert das Ergebnis, um bei dem folgenden seriellen Datenaustausch die Signale richtig bewerten zu können. Diese Messung wird nach jedem Tastendruck (RS) durchgeführt.

Als Nächstes sendet das Programm eine Kennung "KIM" zum TTY. Dann verweilt es in der Schleife "Warte auf Zeichen". Jeder nun vom TTY gesendete Kode wird in der Routine "Eingabe Ausführen" ausgewertet. Das Drücken der jeweiligen Taste veranlaßt das Programm, in entsprechende Unterprogramme zu springen; dort werden die Eingaben interpretiert. Nach Ausführung der zugehörigen Routine, kehrt das Programm an die Stelle "Warte auf Zeichen" zurück und ist bereit, den nächsten Tastendruck auszuwerten.



Flußdiagramm
Bild 3.14

Die TTY-Programmschleife kann unter folgenden Bedingungen verlassen werden:

1. Drücken der RESET-Taste
2. Drücken der G-Taste; bewirkt den Start eines Anwender-Programms
3. Betriebsartwechsel von TTY auf Tastenfeld/Display

Wenn nach dem Drücken der Reset-Taste die Betriebsart Tastenfeld/Display erkannt wird, so beginnt das System-Steuerprogramm mit der Bedienung dieser Elemente. Das Display zeigt den in der adressierten Speicherzelle enthaltenen Wert an ("Anzeigen Speicherzelle"), solange keine der Tasten gedrückt wurde (Abfrage TA?). Wird ein Tastendruck erkannt, so entscheidet das Programm während eines zusätzlichen Anzeigezyklus, ob die Daten gültig sind. Wird ein echter Tastendruck erkannt, so führt das Programm die Routine "Warte auf Taste" aus; dort entschlüsselt es, welche Taste gedrückt wurde. Als Nächstes durchläuft das Programm die Routine "Eingabe Ausführen". Durch Unterteilung in entsprechende Unterprogramme werden die den einzelnen Tasten zugeordneten Funktionen ausgeführt. Danach kehrt das Programm in die Routine "Anzeigen Speicherzelle" zurück und wartet darauf, daß die Taste losgelassen wird. Ist das der Fall, dann erfolgt ein Sprung in die ursprüngliche Routine "Anzeigen Speicherzelle". Es wird der nächste Tastendruck erwartet.

In der Betriebsart TTY wie auch in KB können Routine aufgerufen werden, welche Schreiben oder Lesen auf bzw. vom Magnetband bewirken. In beiden Fällen werden bestimmte Befehle der Eingabemedien verwendet. Auf jeden Fall wird nach Beendigung einer Magnetband-Routine zur Start-Position zurückgekehrt, was bewirkt, daß je nach Betriebsart entweder am TTY der Ausdruck "KIM" erfolgt oder das Display wieder eine Anzeige bringt.

Das Betätigen der Stop-Taste bewirkt einen nichtmaskiertbaren Interrupt (NMI) des Prozessors. Beim Drücken dieser Taste wird ein laufendes Programm unterbrochen, der Maschinenstatus in den Stack gerettet und das Systemsteuerprogramm wieder gestartet.

Der andere vorhandene Interrupt-Eingang (IRQ) ermöglicht Sprünge an jede vom Anwender gewünschte Adresse im Speicherbereich.

KAPITEL 4

BEDIENUNG DES KIM-1 SYSTEMS

Nachdem Sie jetzt eine bessere Vorstellung davon haben, was Ihr KIM-1 System enthält und wie es arbeitet, möchten wir Sie ausführlicher mit den Systemoperationen vertraut machen. Wir wollen die möglichen Systemoperationen in drei Gruppen einteilen: Betrieb nur mit Tastenfeld und Display, mit Magnetband-Rekorder und mit TTY.

4.1 Tastenfeld und Display

Ein kurzer Blick auf das Tastenfeld zeigt, daß es 23 Tasten und einen Schiebeschalter gibt. Die Funktion der einzelnen Tasten ist wie folgt:

- 0 bis F – 16 Tasten, hexadezimal kodiert zur Eingabe von Daten oder Adressen
- AD – Wählt den Adresseingabe-Modus
- DA – Wählt den Dateneingabe-Modus
- \+ – Erhöht die Adresse um eins, verändert den Eingabemodus nicht
- PC – Stellt den Inhalt des Programmzähler dar
- GO – Bewirkt den Start eines Programms ab der im Display angezeigten Adresse
- ST – Beendet die Ausführung eines Programms und bewirkt einen Rücksprung in das System-Steuerprogramm.

Wie Sie bereits aus den vorhergegangenen Kapiteln wissen, besteht das 6-Digit Display aus einem 4-Digit Anzeigefeld für Adressen und einem 2-Digit Anzeigefeld für Daten.

Mit dem KIM-1 Tastenfeld und dem Display können Sie folgende Operationen durchführen:

1. Adresse anwählen

   Drücken Sie die Taste AD und danach vier beliebige Hexatasten. Der so eingegebene Wert erscheint im Adressfeld des Display. Haben Sie eine falsche Eingabe gemacht, geben Sie solange weiter ein bis der gewünschte Wert auf dem Display erscheint. Im Datenfeld des Displays wird der zur jeweiligen Adresse gehörige Speicherzelleninhalt angezeigt.

2. Daten modifizieren

   Nach Eingabe der gewünschten Adresse soll nun die Taste DA gedrückt werden. Wenn Sie jetzt zwei Hexawerte eingeben, so stellen diese den neuen Speicherinhalt unter der aktuellen Adresse

dar. Die eingegebenen Daten erscheinen im Datenfeld des Displays. Damit erhalten Sie eine Quittung dafür, daß ihre Eingabe an der richtigen Stelle angekommen ist.
Beachten Sie, daß es möglich ist eine ROM-Adresse anzuwählen, oder eine Speicherzelle die im System überhaupt nicht existiert. In einem solchen Fall wird es Ihnen nicht gelingen den Inhalt der Datenanzeige zu verändern, da es unmöglich ist, in eine ROM-Zelle oder eine nicht vorhandene Zelle zu schreiben.

3. Adresse inkrementieren
   Durch Drücken der Taste + wird die angezeigte Adresse automatisch um eins erhöht. Infolgedessen erscheint der Inhalt der neuen Speicherzelle im Datenfeld des Displays. Dieser Ablauf ist nützlich, wenn man eine Reihe von aufeinanderfolgenden Speicherplätzen lesen oder modifizieren will. Das Drücken der + Taste ändert den Eingabemodus nicht. Hatten Sie zuvor AD gedrückt, so bleiben Sie im Adresseingabe-Modus, war DA gedrückt, dann fahren Sie im Dateneingabe-Modus fort.

4. Abrufen des Programmzählers
   Immer wenn der NMI-Interrupt-Eingang des 6502 Mikroprozessors aktiviert ist, wird der normale Programmablauf unterbrochen und die interenen CPU-Register an einer bestimmten Stelle im Speicher abgelegt; anschließend erfolt ein Rücksprung ins Systemsteuer-Programm. Im KIM-1 System kann NMI-Interrupt durch das Drücken der Stop-Taste (ST) auftreten, oder, wenn Einzelschrittbetrieb eingestellt ist, nach Drücken der GO-Taste.
   Die PC-Taste gestattet es, den Programmzählerstand zur Zeit des Interrupts abzurufen. Dabei wird dieser Programmzählerinhalt im Adressfeld des Diplays dargestellt. Sie können dann im Programm fortfahren, indem Sie die GO-Taste drücken.

5. Ausführen eines Programms
   Wählen Sie die Anfangsadresse des gewünschten Programms. Drücken Sie jetzt die GO-Taste und schon startet das Programm bei der im Adressfeld des Displays angezeigten Adresse.

6. Beenden eines Programms
   Die Stop-Taste hat die Aufgabe ein Programm zu unterbrechen. Wie bereits erwähnt, aktiviert die Stop-Taste den NMI-Interrupteingang des 6502 Mikroprozessors.
   Beachten Sie: Die Taste ST arbeitet nur dann richtig, wenn Sie am Anfang den richtigen Interrupt-Vektor in die Speicherzellen 17FA und 17FB geladen haben. Für die meisten Betriebsfälle ist es zweckmäßig den Wert 1C00 in diese Zellen zu laden. Das geschieht folgendermaßen:

   Drücke Taste:

AD			
1	7	F	A
DA		0	0
+		1	C

Wenn jetzt ein NMI-Interrupt auftritt, springt das Programm an die Stelle 1C00, sichert alle CPU-Register und tritt erst dann in das Systemsteuerprogramm ein.

Sie sollten nach jeder Unterbrechung der Versorgungsspannung den Interrupt-Vektor neu laden. Wenn das System auf die Stop-Taste falsch reagiert, so liegt das häufig daran, daß der Interrupt-Vektor nicht geladen wurde.

7. Einzelschritt Programmausführung

   Bei der Fehlersuche in einem neuentwickelten Programm wird Ihnen diese Betriebsart sehr helfen. Schalten Sie den SST Schalter in die ON-Position. Drücken Sie dann die GO-Taste jedesmal, wenn Sie einen Programmschritt weiterfahren möchten. Das Display zeigt jetzt die Adresse und den Maschinenkode des auszuführenden Befehls an. Beachten Sie, daß beim Durchtasten eines Programms einige Adressen scheinbar übersprungen werden. Das hängt damit zusammen, daß es ein-, zwei- und drei-Byte Befehle gibt. Bei Mehrbyte-Befehlen verweilt das Einzelschritt-Programm nur beim ersten Byte eines Befehls.

   Die Betriebsart SST (Einzelschritt) verwendet ebenfalls den NMI-Interrupt des Mikroprozessors. Deshalb ist es auch hier wichtig, den Interrupt-Vektor richtig zu laden.

Damit sind alle Standard-Operationen behandelt, die Sie vom Tastenfeld aus durchführen können. Durch Kombinationen können Sie bestimmte Sonderoperationen vornehmen:

1. Laden des IRQ-Interrupt-Vektors

   Erinnern Sie sich, daß das System noch einen zusätzlichen Interrupt besitzt (IRQ), welcher ebenfalls direkt auf den Mikroprozessor einwirkt. Um auch mit diesem zu arbeiten, laden Sie zuvor den zugehörigen Interrupt-Vektor. Dieser muß in den Speicherzellen 17FE und 17FF abgelegt werden.

2. Maschinenstatus abfragen

   Durch einen NMI-Interrupt, der z.B. von der Stop-Taste oder vom Betrieb im SST-Modus herrührt, werden alle CPU-internen Register in bestimmten Speicherzellen abgelegt. Wenn Sie diese Zellen auslesen möchten, finden Sie diese bei folgenden Adressen:

   00EF = Programmzähler niederes Byte (PCL)
   00F0 = Programmzähler höheres Byte (PCH)
   00F1 = Status-Register (P)
   00F2 = Stack-Pointer (SP)
   00F3 = Akkumulator (A)
   00F4 = Index-Register Y
   00F5 = Index-Register X

4.2 Nf-Kassetten Recorder

Mit dem Nf-Kassetten Recorder sind zwei Operationen möglich: Daten aus dem Speicher des KIM-1 System auf dem Magnetband aufzeichnen, oder bereits auf dem Magnetband aufgezeichnete Daten im Speicher Ihres KIM-1 Systems (RAM) ablegen.

Aufzeichnungen von Daten auf das Magnetband.

Dazu gehen Sie wie folgt vor:

1. Wählen Sie für den Datenblock, den Sie aufzeichnen möchten, eine Kenn-Nummer (ID). Diese zwei Digit-Zahl speichern Sie unter der Adresse 17F9. Verwenden Sie nicht 00 oder FF.
2. Laden Sie die Anfangsadresse des zu übertragenden Bereichs in folgende Speicheradressen:
   17F5 = Anfangsadresse niederes Byte (SAL)
   17F6 = Anfangsadresse höheres Byte (SAH)
3. Verwenden Sie als Endadresse einen Wert, der um ein LSB größer ist, als die Adresse des letzten zu übertragenden Bytes. Laden Sie diese Zahl in folgende Speicheradressen:
   17F7 = Endadresse niederes Byte (EAL)
   17F8 = Endadresse höheres Byte (EAH)

Dazu ein Beispiel: Angenommen Sie möchten einen Datenblock von Adresse 0200 bis einschließlich 03FF auf Band übertragen (Page 2 und 3). Ihr Block soll die Kenn-Nummer 06 erhalten. Laden Sie die Daten wie folgt mittels des Tastenfeldes in den Speicher Ihres KIM-1:

```
17F5 = 00   (SAL)
17F6 = 02   (SAH)
17F7 = 00   (EAL)  } = 03FF + 1
17F8 = 04   (EAH)  }
17F9 = 06   (ID)
```

Beachten Sie, daß die Endadresse größer als die Anfangsadresse sein muß.

4. Wenn Sie eine neue, unbeschriebene Kassette verwenden, genügt es, diese in das Gerät einzulegen und auf die Startposition zurückzuspulen.
5. Stellen Sie die Anfangsadresse des Programms für Magnetbandaufzeichnung ein; diese Adresse ist 1800.
6. Starten Sie das Band in der Stellung Aufnahme und warten Sie, bis das Band seine Nenngeschwindigkeit erreicht hat (einige Sekunden).
7. Drücken Sie jetzt die Taste GO und schon beginnt der Aufzeichnungsvorgang. Die Anzeige erlischt kurz und leuchtet dann wieder auf, wenn der Übertragungsvorgang abgeschlossen ist.
8. Warten Sie jetzt noch einige Sekunden und halten Sie dann das Band an.

Zurücklesen von Daten vom Magnetband.

Gehen Sie bitte wie folgt vor:

1. Wählen Sie die Kenn-Nummer des Blocks, den Sie vom Band lesen möchten, und laden Sie diesen Wert in die Speicherzelle mit der Adresse 17F9.
2. Stellen Sie die Anfangsadresse des Programms für Magnetbandeinlesen ein; diese Adresse ist 1873.
3. Drücken Sie jetzt die Taste GO. Das KIM-1 System wartet nun auf Daten vom Magnetband.
4. Legen Sie die Kassette ein und bringen Sie diese in Anfangsposition. Überzeugen Sie sich davon, daß der Lautstärkeregeler auf Mittenposition ist.
5. Starten Sie das Magnetbandgerät in der Stellung Wiedereingabe und überzeugen Sie sich davon, daß sich das Band bewegt.
6. Das Display erlischt während der Übertragung. Wenn es wieder aufleuchtet, muß es den Wert 0000 xx bringen. Ist das der Fall, so ist der gesuchte Datenblock gefunden und korrekt im RAM des KIM-1 Systems abgelegt worden. Zeigt das Display FFFF xx, so ist zwar der richtige Block gefunden worden, aber bei der Übertragung ist ein Fehler aufgetreten. Läuft das Band weiter und die Anzeige leuchtet nicht wieder auf, so wurde der gesuchte Datenblock nicht gefunden.
7. Wenn Sie in Schritt 1 die Kenn-Nummer ID = 00 verwendet haben, wird die am Band aufgezeichnete Kenn-Nummer ignoriert, und das System übernimmt den ersten gültigen Daten-Block. Dabei werden die Daten an den vom Band gelesenen Adressen im RAM des KIM-1 Systems gespeichert.
8. Hatten Sie in Schritt 1 die Kenn-Nummer ID = FF geladen, so wird wie in 7 der erste gültige Daten-Block angenommen. Allerdings erfolgt jetzt die Speicherung im KIM-1 System nicht an der vom Band gelesenen Adresse, sondern an der Stelle, welche in den Speicherzellen 17F5 und 17F6 (SAL, SAH) abgelegt ist.

Spezielle Magnetband-Operationen

Das KIM-1 System bewirkt die Aufzeichnung von Daten auf Magnetband nach dem in Anhang E beschriebenen Format. Jeder Datenblock am Band beginnt mit einer Gruppe von Synchronisationszeichen, gefolgt von der Kenn-Nummer ID. Die Blöcke dürfen beliebig lang sein. Wenn Sie bei der Aufzeichnung sorgfältig vorgehen, wird es Ihnen nicht schwer fallen, mehr als einen Datenblock auf eine Kassette zu laden. Um Blöcke nacheinander aufzuzeichnen, ohne dazwischen das Band zurückzuspulen, müssen Sie lediglich vor jedem neuen Block die Kenn-Nummer ID, SAL, SAH EAL und EAH neu eingeben.

Möchten Sie auf ein teilweise beschriebenes Band weitere Daten aufzeichnen, gehen Sie wie folgt vor: Band an die Startposition zurückspulen, Parameter des zuletzt aufgezeichneten Blocks eingeben und das Bandgerät im Wiedergabe-Modus starten. Wenn Ihr System den eingestellten Block gefunden hat, halten Sie das Bandgerät an. Sie befinden sich jetzt an einer Stelle an welcher Sie neue Daten aufzeichnen können.

Ist es auch möglich, zwischen den Datenblocks x gesprochene Texte einzufügen. Diese werden vom KIM-1 System überlesen. Allerdings ist es nötig einen Hörer zum Dateneingang des KIM-1 Systems parallel zu schalten, um diese Texte zu erkennen.

Wir empfehlen Ihnen, Daten nur an Stellen des Bandes aufzuzeichnen, die mit Sicherheit gelöscht sind. Beim Überlappen von Blöcken treten möglicherweise Fehler auf.

4.3. 8-Kanal Fernschreiber

Der Anschluß eines 8-Kanal Fernschreibers mit Serienschnittstelle (z.B. Teletype Modell ASR 33) an das KIM-1 System bringt eine Vielzahl neuer Betriebsmöglichkeiten. In allen Fällen bekommen Sie auch für Ihre Eingabe ein Protokoll. Ist Ihr TTY mit einem Lochstreifenleser/stanzer ausgerüstet, können Sie mit dem KIM-1 System Lochstreifen erzeugen und diese auch wieder in das System einlesen. Der TTY ermöglicht folgende Operationen:

Anwählen einer Adresse

Drücken Sie vier Hexatasten (0 bis F), um dem Gerät die gewünschte Adresse mitzuteilen. Dann drücken Sie die SPACE-Taste.

Der Drucker antwortet auf diese Eingabe mit den vier eingegebenen Zeichen und einem Zweidigit-Hexakode, welcher dem Speicherinhalt der adressierten Zelle entspricht.

Drücken Sie:	1234 SPACE
Drucker antwortet:	1234 AF

Das bedeutet, daß unter der Adresse 1234 das Datenwort AF gespeichert ist.

Modifizieren von Daten

Geben Sie wie im vorhergehenden Abschnitt eine Adresse ein. Drücken Sie dann zwei Hexa-Zeichen; das sind die neu einzugebenden Daten. Betätigen Sie jetzt die "."-Taste; damit wird der alte Zelleninhalt mit den neuen Daten überschrieben.

Drücken Sie:	1234 SPACE
Drucker antwortet:	1234 AF
Drücken Sie:	6D"."
Drucker antwortet:	1235 B7

Beachten Sie: die angewählte Adresse 1234 wurde mit 6D überschrieben, das System schreitet automatisch zur nächsten Adresse und gibt die dort gespeicherten Daten aus (B7).

Vornullen sind weder bei Adressen noch bei Daten einzugeben. Dazu ein Beispiel:

Drücken Sie:	EF SPACE
Es wird die Adresse 00EF angewählt	
Drücken Sie:	E SPACE
Es wird die Adresse 000E angewählt	

Drücken Sie:	A "."	
Es werden die Daten OA eingegeben		
Drücken Sie:	"."	
Es werden die Daten 00 eingegeben.		

Adresse inkrementieren

Drücken Sie "CR" (Wagenrücklauf) wenn Sie die Adresse um eins erhöhen möchten, ohne die eingestellte Speicherzelle zu verändern.

Letzter Ausdruck:	1234 AF	
Drücken Sie:		CR
Drucker antwortet:	1235 B7	
Drücken Sie:		CR
Drucker antwortet:	1236 C8	

Adresse dekrementieren

Drücken Sie "LF" Zeilenwechsel), wenn Sie die vorhergehende Adresse erreichen wollen:

Letzter Ausdruck:	1234 AF	
Drücken Sie:		LF
Drucker antwortet:	1233 9D	
Drücken Sie:		LF
Drucker antwortet:	1232 8E	

Abbrechen der Operation

Um den Ablauf einer Operation zu beenden, drücken Sie "RUB OUT" (Löschen). Als Ergebnis dieses Tastendrucks bringt das System automatisch die Ausgabe "KIM". Das bedeutet, daß eine neue Operation eingeleitet werden kann.

Drücken Sie:	1264	RUB OUT
Drucker antwortet:	KIM	
	xxxx xx	
Drücken Sie:	1234	SPACE
Drucker antwortet	1234 AF	

In diesem Beispiel wurde die RUB OUT-Taste verwendet, um eine fehlerhafte Adresseingabe zu korrigieren. Beachten Sie: Die RUB OUT-Taste muß jedesmal gedrückt werden, wenn die RESET-Taste am KIM-1 Modul betätigt wurde, damit sich das System auf die Übertragungsgeschwindigkeit des TTY einstellen kann.

Lochstreifen laden

Lochstreifen mit einem Datenformat wie in Anhang F beschrieben, können wie folgt eingelesen werden:

1. Lochstreifen in den Leser einlegen
2. Drücken Sie "L"
3. Starten Sie den Lochstreifen-Leser

Die auf dem Lochstreifen gespeicherten Daten werden in den Speicher des KIM-1 Systems geladen. Dabei zeugt der TTY-Drucker ein Protokoll des Lochstreifen-Inhalts.

Während des Einlesevorgangs wird eine Prüfsumme der Daten gebildet (Check-sum), die nach Ende der bertragung mit der am Lochstreifen bereits enthaltenen, verglichen wird. Sind beide Summen nicht identisch, folgt der Ausdruck einer Fehlermeldung.

ochstreifen stanzen

Das KIM-1 System kann dazu benutzt werden, Lochstreifen nach dem in Anhang F beschriebenen Format ı stanzen. Man gehe dazu wie folgt vor:

1. Wählen Sie Anfangs- und Endadresse der auf den Lochstreifen zu stanzenden Daten
2. Legen Sie ungestanztes Lochstreifenmaterial ein und starten Sie den Stanzer.

Drücken Sie:		1 7 F 7 SPACE
Drucker antwortet:	17F7 xx	
Drücken Sie:		F F .
Drucker antwortet:	17F8 xx	
Drücken Sie:		0 3 .
Drucker antwortet:	17F9 xx	
Drücken Sie:		2 0 0 SPACE
Drucker antwortet:	0200 xx	

Die Endadresse 03FF wurde in die Zellen 17F7 (EAL) und 17F8 (EAH) abgespeichert. Die Anfangsadresse 200 wurde wie gezeigt angewählt.

3. Drücken Sie jetzt Q

   Der Stanzvorgang beginnt; gleichzeitig wird vom TTY-Drucker ein Protokoll erstellt.

usdrucken eines Programms

Es ist möglich, einen Teil des Speichers des KIM-1 Systems auszudrucken. Der Vorgang ist der gleiche wie eim Erstellen eines Lochstreifens, nur wird jetzt der Stanzer nicht eingeschaltet.

Programm ausführen

Um ein Programm vom TTY-Tastenfeld aus zu starten ist folgende Prozedur nötig:

1. Laden Sie die Startadresse des Programms
2. Drücken Sie G

   Wenn Sie ein Programm an der Stelle 0200 starten möchten, verfahren Sie wie folgt:

   Drücken Sie: (2)(0)(0) [SPACE]

   Drucker antwortet: 0200 xx

   Drücken Sie: (G)

Das Programm startet bei 0200 und läuft so lange, bis Sie die Taste ST oder RS am KIM-1 Modul drücken. Sie können auch im Einzeltakt-Modus arbeiten.

KAPITEL 5

EIN ANWENDUNGSBEISPIEL

Es ist nicht der Sinn dieses Handbuchs, jede mögliche Anwendung oder Programmiertechnik zu beschreiben. Wir wollen daher, nachdem Sie bereits mit den Grundelementen des KIM-1 Systems vertraut sind, lediglich das bisher Behandelte in einem einfachen, aber typischen Anwendungsfall zusammenfassen.

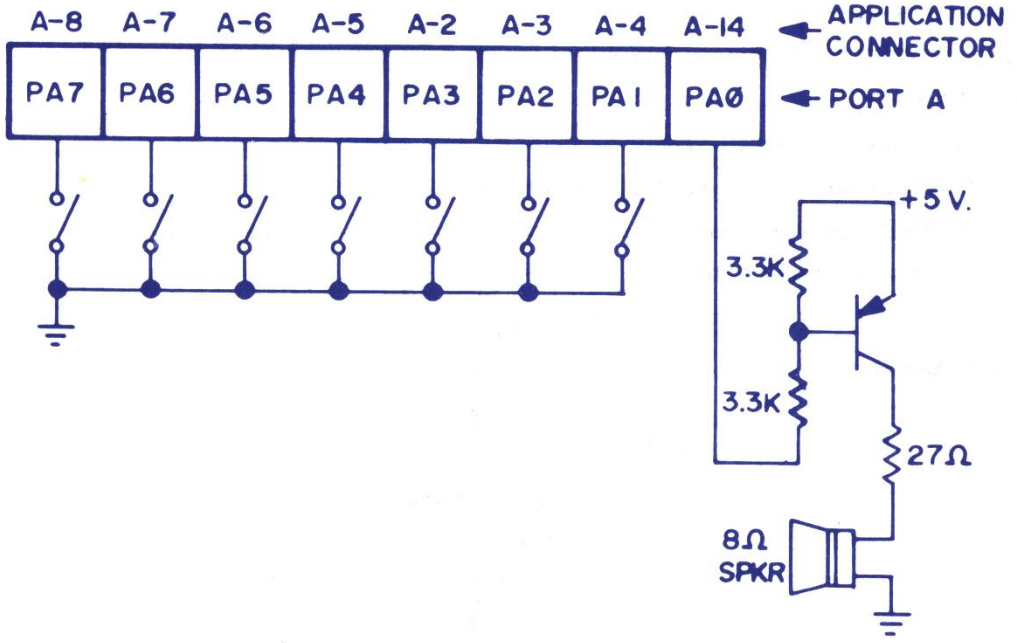
Die Aufgabe ist, eine Rechteckspannung mit veränderbarer Frequenz zu erzeugen und diese in einem Lautsprecher hörbar zu machen. Zur Einstellung der Frequenz sollen 7 Kippschalter verwendet werden. Wir wollen zur Lösung des Problems folgendermaßen vorgehen: zunächst muß ein Interface entworfen werden, dann wird das Programm geschrieben, geladen und ausgeführt. Schließlich behandeln wir eine Reihe von Fehlersuchmethoden, welche auch zum Austesten späterer Programme sehr nützlich sind.

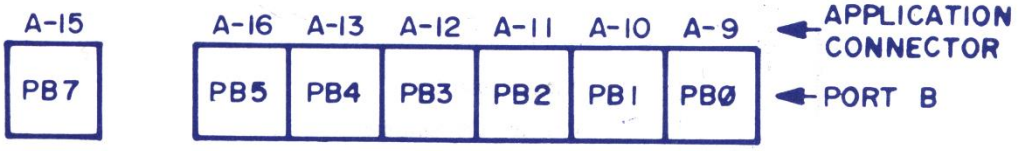
5.1 Interface-Auslegung

Sie erinnern sich sicher daran, daß 15 I/O-Leitungen von dem Baustein 6530-003 auf den Applikationsstekker herausgeführt sind. Die Logik- und Schaltungsdetails dieser I/O-Leitungen sind im Anhang H und im Abschnitt 1.6 des Hardware-Handbuchs beschrieben.

Für unser Anwendungsbeispiel verwenden wir nur acht Leitungen. Eine davon wird als Ausgang deklariert (PAØ). Sie steuert den Lautsprecher über einen Treiber. Die übrigen 7 Leitungen werden mit den Kippschaltern verbunden und als Eingänge deklariert. In Bild 5.1 ist das Schaltbild wiedergegeben. Die sieben Anschlüsse von Port B werden in diesem Beispiel nicht verwendet.

Ein geschlossener Schalter soll log. 1 bedeuten, ein offener Schalter log. 0. Wenn wir die gemeinsame Verbindung der Schalter auf Masse legen, erzeugen wir gerade den umgekehrten Sinn. Das ist bei der Entwicklung der Software zu berücksichtigen: d.h., das von den Schaltern gelesene Datenwort muß nach dem Einlesen in die CPU komplementiert werden. Wir legen weiterhin fest: der an Leitung PA1 angeschlossene Schalter soll die geringste Bewertung besitzen (LSB = 2^0). Man kann also mit den sieben Schaltern eine Binärzahl von 0 bis 127 darstellen.





(THE B PORT IS NOT USED IN THIS EXAMPLE APPLICATION)

Verwendung eines Lautsprechers

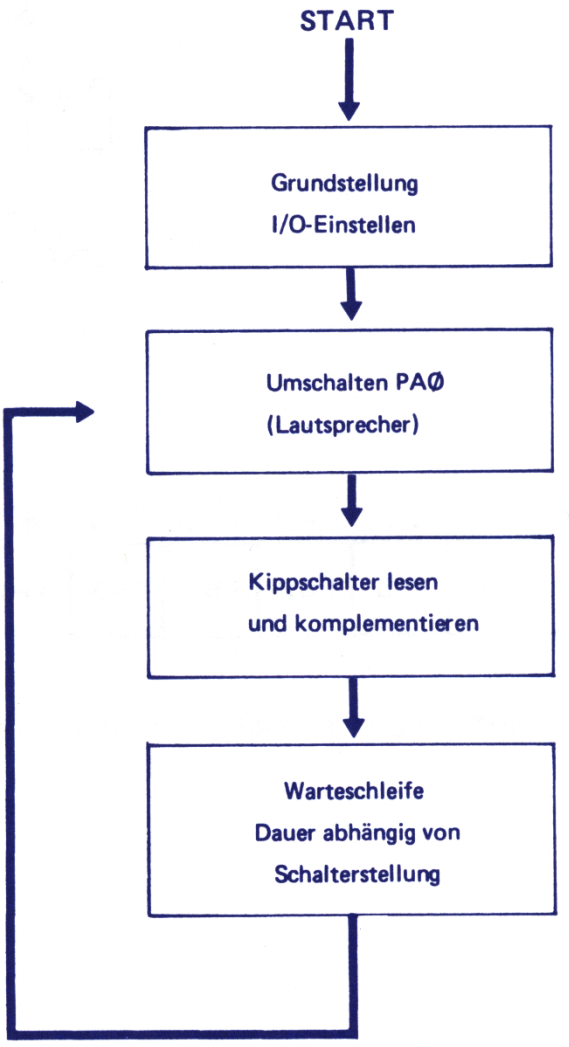
Bild 5.1

5.2 Schreiben des Programms

Nachdem das Interface festgelegt ist, muß als nächstes das Programm geschrieben werden. Dabei gehen wir in vier Schritten vor:

1. Entwerfen eines Flußdiagramms
2. Entwerfen des Programms in Assemblersprache
3. Überprüfen des Programms
4. Übersetzen in Maschinensprache

Flußdiagramm



Als ersten Schritt zeigt unser Flußdiagramm die Grundstellung des Systems. In diesem Schritt muß auch festgelegt werden, daß der Stift PAØ als Ausgang und die restlichen I/O-Leitungen als Eingänge behandelt werden.

Danach beginnen wir eine Programmschleife, indem der Zustand des Ausgangs PAØ invertiert wird. Dann lesen wir den Zustand der Kippschalter und komplementieren den gelesenen Wert, um den richtigen Sinn zu bekommen. Der gelesene Wert bestimmt die Länge einer Verzögerungsschleife, nach deren Ablauf das Programm zurückspringt, um den Zustand der Leitung PAØ erneut zu invertieren. Es ist bereits erkennbar, daß dieses Programm am Ausgang PAØ eine Rechteckspannung erzeugt, deren Frequenz durch die Stellung der Kippschalter gegeben ist.

Assemblerprogramm:

Unsere nächste Aufgabe ist es, dieses einfache Flußdiagramm in ein Programm umzusetzen. Dieses soll zunächst in Assemblersprache geschrieben werden. Studieren Sie Ihr Programmier-Handbuch, um mit den möglichen Befehlen vertraut zu werden, welche die CPU (6502) verarbeiten kann. Eine Zusammenstellung der wichtigsten Befehle finden Sie auch im Anhang B. Bild 5.2 zeigt das in Assemblersprache geschriebene Programm.

Marke	Operations Kode	Operand	Zyklus-zahl	Bemerkungen
INIT	LDA	#$01	2	Deklarieren I/O: O = Ein, 1=Aus
	STA	PADD	4	Datenrichtungsreg. Port A
START	INC	PAD	6	Umschalten PAØ ; PA1-PA7 bleiben unverändert
READ	LDA	PAD	4	Lesen Schalter in Akkumul.
	EOR	#$FF	2	Komplementiere Daten
	LSR	A	2	Akkumul. 1 Bit rechts schieben
	TAX		2	Ergebnis nach Indexreg. X
DELAY	DEX		2	Verzögerungsschleife, Dauer
	BPL	DELAY	3,2	nach Inhalt Indexreg. X
	BMI	START	3	Springe nach Start
PADD	=$1701			Absolute Adresse Datenrichtungs-register A
PAD	=$1700			Absolute Adresse Datenregister A

Assembler-Liste

Bild 5.2

Wie Sie sehen, ist jede Zeile des Programms in mehrere Felder unterteilt:

1. Ein Markenfeld, in dem Sie einer bestimmten Programmzelle einen Namen zuordnen können.
2. Ein Operationskode-Feld, in welches der auszuführende Befehl eingetragen wird.
3. Ein Operanden-Feld. Dort werden solche Daten hinterlegt, welche vom Operationskode direkt zur Verarbeitung benötigt werden, über die Adressierungsart Auskunft geben oder das Datenformat festlegen. In den Handbüchern werden folgende Symbole benutzt:

#	Folgebyte-Adressierung (Immediate)
$	Hexa-Kode
@	Oktal-Kode
%	Binär-Kode
'	ASCII-Zeichen
=	Zuordnung einer Marke zu einem Wert

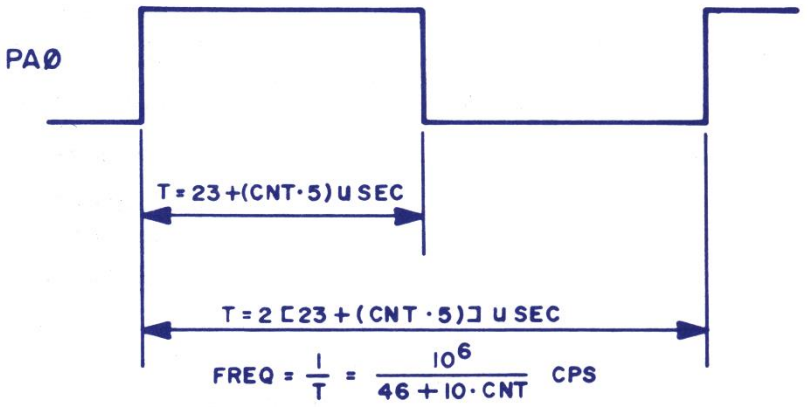
4. Ein Maschinenzyklus-Feld, das die Anzahl der erforderlichen Zyklen angibt, um einen Befehl abzuarbeiten. Siehe auch Anhang B des Programmier-Handbuchs.
5. Ein Kommentar-Feld, in dem der Programmierer die Bedeutung der Operation vermerken kann.

Überprüfung des Programms

Die Information über die Anzahl der benötigten Maschinenzyklen gestattet, den exakten Zeitbedarf des Programms zu ermitteln. Beachten Sie, daß das KIM-1 System von einem 1 MHz-Oszillator gespeist wird; jeder Maschinenzyklus dauert daher 1.10^{-6} sek. Ein Befehl wie z.B. "INC PAD", welcher 6 Maschinenzyklen benötigt, dauert demnach 6.10^{-6} Sek.

Wenn man die Anzahl der Maschinenzyklen ermittelt, welche zwischen dem zweimaligen Ansprechen der Leitung PAØ liegen, kann man eine Formel für die Frequenz der erzeugten Rechteckspannung aufstellen. Die Frequenz kann ermittelt werden, aus der Schalterstellung der Anzahl von Maschinenzyklen zwischen zwei Signalwechseln an PAØ und der Periodendauer des Systemtaktes im KIM-1 System. Bild 5.3 zeigt die Kurvenform des Signals an PAØ und die Formeln, aus denen die Frequenz ermittelt wird.

CNT entspricht dem Wert des Indexregisters X; dieser Wert ist von der Stellung der Schalter abhängig.



Rechtecksignal Ausgang

Bild 5.3

Kodierung in Maschinensprache

Unser nächstes Problem ist es, das in Assemblersprache geschriebene Programm in den Maschinenkode umzusetzen. Die schnellste und sicherste Methode besteht darin, den von MOS Technology Inc. erstellten Assembler zu benutzen. Wenn Sie diesen Weg nicht gehen wollen, müssen Sie Ihr Programm mit Hilfe einer Übersetzliste bearbeiten.

Es ist zweckmäßig, zunächst eine Tabelle zu erstellen (Bild 5.4).

	Kode			Befehl		
ADDRESS	BYTE 1	BYTE 2	BYTE 3	LABEL	OP CODE	OPERAND
Ø2ØØ	A9	Ø1		INIT	LDA	#$01
Ø2Ø2	8D	Ø1	17		STA	PADD
Ø2Ø5	EE	ØØ	17	START	INC	PAD
Ø2Ø8	AD	ØØ	17	READ	LDA	PAD
Ø2ØB	49	FF			EOR	#$FF
Ø2ØD	4A				LSR	A
Ø2ØE	AA				TAX	
Ø2ØF	CA			DELAY	DEX	
Ø21Ø	1Ø	FD			BPL	DELAY
Ø212	3Ø	F1			BMI	START
Ø214						

Maschinenkode Tabelle

Bild 5.4

Als erstes wird der Inhalt des in Assemblersprache geschriebenen Programms aus Bild 5.2 in diese Tabelle eingetragen. In die erste Spalte tragen Sie die absolute Adresse ein, bei welcher Ihr Programm beginnen soll. Die Befehls-Spalte enthält soviel Platz, um Ein-, Zwei- und Drei-Byte Befehle einzutragen. Den ensprechenden Operationskode finden Sie im Anhang B Ihres Programmierhandbuchs oder in der Programmierkarte.

Beispielsweise lautet der erste Befehl LDA #$01, das bedeutet: lade den Akkumulator mit dem Inhalt der nächsten Programmspeicher-Zelle (01). Es handelt sich also um den Immediate-Adressiermodus, welcher durch das Zeichen # gekennzeichnet ist. Der Operationskode für LDA # ist A9. Dieser Wert muß nun in die erste Befehlsspalte eingetragen werden. Das Folgebyte hexa 01 wird in die nächste Befehlsspalte geschrieben. Die Anfangsadresse für Ihr Programm tragen Sie in die Adreßspalte ein, z.B. hexa 0200. Der gesamte Befehl LDA #$01 belegt also die Speicherzellen 0200 und 0201.

Die nächste Adresse 0202 wird nun in die Adreßspalte eingetragen und dem nächsten Befehl zugeordnet. Fahren Sie in dieser Weise fort, indem Sie alle folgenden Ein-, Zwei- und Drei-Bytebefehle in den Maschinenkode übersetzen und in die Befehlsspalte eintragen. Dabei enthält die Adreßspalte immer nur die Adresse, die zum Operationskode gehört.

Ist der Operand ein Symbol, muß der diesem Symbol zugeordnete Wert (absolute Adresse) eingetragen werden. Dazu ein Beispiel: Der Befehl INC PAD bedeutet: erhöhe den Inhalt der Speicherzelle mit der symbolischen Adresse PAD um eins. In unserem Assembler-Programm wurde dem Symbol PAD die absolute Adresse 1700 zugeordnet. Deshalb müssen Sie in die zweite und dritte Befehlsspalte den Wert 1700 eintragen. Beachten Sie, daß nach dem Operationskode immer zunächst das Adreßbyte mit der geringeren Bewertung eingetragen wird, danach das mit der höheren Bewertung.

Wenn Sie mit Verzweigungsbefehlen arbeiten, müssen Sie zuerst die genaue Sprungweite vor- bzw. rückwärts ermitteln. Schlagen Sie auch in Ihrem Programmier-Handbuch Kapitel 4.1.1 (Grundlagen der relativen Verzweigung) nach. In unserem Beispiel benötigt der Befehl BMI START (s. Bild 5.2 und Bild 5.4) eine Rückwärtssprungweite von −15, um die mit START gekennzeichnete Adresse zu erreichen. Der Sprung erfolgt von Adresse 0213 rückwärts nach Adresse 0205 einschließlich.

Das Zweierkomplement von −15 ist in Hexazahl F1; diese tragen Sie in die Zelle 0212 ein. Handelt es sich um einen Vorwärtssprung, so müssen Sie anstelle des Zweierkomplements den Zahlenwert selbst eintragen.

5.3 Laden des Programms

Wenn Sie Ihr Programm soweit übersetzt haben, können Sie die in Bild 5.4 eingetragenen Adressen und Kodewerte nach der in Kapitel 2.4 angegebenen Prozedur in Ihr KIM-1 System laden. Der Ablauf ist wie folgt:

Drücke Taste	Anzeige
AD 0 2 0 0	0200 xx
DA A 9	0200 A9
+ 0 1	0201 01
+ 8 D	0202 8D
+ 0 1	0203 01
+ 1 7	0204 17
+ E E	0205 EE
+ 0 0	0206 ØØ
+ 1 7	0207 17
+ A D	0208 AD
+ 0 0	0209 ØØ

Drücke Taste			Anzeige
+	1	7	020A 17
+	4	9	020B 49
+	F	F	020C FF
+	4	A	020D 4A
+	A	A	020E AA
+	C	A	020F CA
+	1	0	0210 10
+	F	D	0211 FD
+	3	0	0212 30
+	F	1	0213 F1

Tastenfolge Programm laden

Bild 5.5

5.4 Ausführen des Programms

Nachdem Sie das Programm geladen haben, können Sie mit der Ausführung beginnen. Falls der NMI-Vektor noch nicht deklariert wurde, machen Sie zunächst noch folgende Eingabe:

Drücke Taste					Anzeige
AD	1	7	F	A	17FA xx
DA			0	0	17FA 00
+			1	C	17FB 1C

Diese Prozedur bewirkt, daß das Programm mit der ST-Taste richtig beendet wird. Stellen Sie jetzt noch die Startadresse Ihres Programms (0200) ein und beginnen Sie mit der Ausführung:

Drücke Taste					Anzeige
AD	0	2	0	0	0200 A9
GO					(Dark)

Ihr Programm läuft jetzt ab. Wenn alle sieben Kippschalter offen sind, werden Sie noch keinen Ton im Lautsprecher hören, weil die Frequenz zu hoch ist. Schließen Sie alle Schalter; nun muß der tiefste Ton, den das Programm erzeugen kann, zu hören sein. Mit anderen Schalterstellungen können Sie verschiedene Tonhöhen erhalten.

Wenn Sie die Taste ST drücken, halten Sie Ihr Programm an, der Ton im Lautsprecher verschwindet und die LED-Anzeige auf Ihrem KIM-1 Modul leuchtet wieder auf. Die angezeigte Adresse wäre als nächste ausgeführt worden. Wahrscheinlich 020F oder 0210, da das Programm den größten Teil der Verarbeitungszeit an diesen Adressen verbringt (Verzögerungsschleife!).

5.5 Austesten und Verändern des Programms

Wenn Ihr Programm nicht zufriedenstellend gelaufen ist, müssen Sie eine Fehlersuch-Prozedur durchführen, welche aus den folgenden Schritten besteht:

Schritt 1: Programm auflisten

Überzeugen Sie sich zunächst davon, daß alle Programmschritte fehlerfrei geladen wurden. Wählen Sie die Startadresse: AD0200, und schauen Sie nach, ob der Wert A9 angezeigt wird. Durch Verwendung der + Taste können Sie jede folgende Programmspeicherzelle zur Anzeige bringen.

Schritt 2: Programmausführung im Einzelschritt

Nehmen Sie die in Kapitel 5.4 beschriebene Prozedur noch einmal vor, aber stellen Sie den Schiebeschalter SST in die Stellung Ein, bevor Sie die Taste GO drücken. Nach Drükken von GO wird der erste Befehl ausgeführt. Die Anzeige erscheint wieder und zeigt Ihnen, daß das System sich erneut im Steuerprogramm befindet. Die angezeigte Adresse gehört zum ersten Byte des nächsten Befehls. Sie können jetzt durch weiteres Drücken von GO im Programm fortschreiten oder Änderungen bei den im Speicher abgelegten CPU-Register vornehmen. Die in Bild 5.6 beschriebene Prozedur gibt einen guten Überblick über die im SST-Modus durchführbaren Operationen.

Schritt 3: I/O-Operationen testen

Wenn der Programmablauf im SST-Modus fehlerfrei zu sein scheint, müssen Sie sich von der ordnungsgemäßen Funktion Ihrer I/O-Anordnung überzeugen.

Erinnern Sie sich: Lesen von oder Schreiben in eine Speicherzelle ist völlig gleichbedeutend mit dem Aussenden oder Einlesen von Daten über I/O-Leitungen. Wenn Sie demnach die Adresse eines Leseports einstellen, zeigt Ihnen die LED-Anzeige auf Ihrem KIM-1 Modul genau den Zustand der zugehörigen I/O-Leitungen an.

Die verwendete Adresse für den I/O-Port ist in unserem Beispiel 1700. Drücken Sie: AD 1 7 0 0 und die LED-Anzeige bringt den Hexakode der zur eingestellten Schalterkonfiguration gehört. Wenn Sie einen Schalter verändern, wird sich auch der im Datenfeld der Anzeige dargestellte Wert ändern.

Belassen Sie die eingestellte Adresse und drücken Sie die DA-Taste. Wenn Sie jetzt eine der Tasten 0 bis F betätigen, schreiben Sie Daten in den Port 1700. Da aber sieben Bits dieses Ports als Eingänge daklariert sind, wird nur ein Bit (PAØ) auf Ihre Eingabe reagieren. Wenn Sie in rascher Folge die Tasten 0 und 1 betätigen, müssen Sie im Lautsprecher ein Geräusch wahrnehmen.

Diese Methode, das KIM-1 Tastenfeld und die Anzeige zu verwenden, um I/O-Ports zu testen, ist sehr nützlich, wenn Sie den Hardware-Teil Ihres Systems zu untersuchen.

Drücke Taste	Anzeige	Bedeutung
AD 0 2 0 0	0200 A9	Wähle Adresse des ersten Befehls
SST (ON)	0200 A9	SST in Stellung EIN; alle Schalter offen!
GO	0202 8D	Lade Akku mit §01
GO	0205 EE	Lade PADD
GO	0208 AD	PAØ umschalten
GO	020B 49	Schalter (PA1-PA7) lesen
GO	020D 4A	Akku Inhalt komplementieren
GO	020E AA	Akku Inhalt um 1 Bit nach rechts schieben
AD 0 0 F 3	00F3 xx	Akku Inhalt anzeigen (Display)
+	00F4 xx	Anzeigen Indexreg. Y
+	00F5 00	Anzeigen Indexreg. X
PC	020E AA	Programmzähler zurückladen
GO	020F CA	Lade Indexreg. X mit Inhalt des Akku
AD 0 0 F 3	00F3 00	Anzeigen Inhalt Akku
+	00F4 xx	Anzeigen Indexreg. Y
+	00F5 00	Anzeigen Indexreg. X
PC	020F CA	Programmzähler zurückladen
GO	0210 10	DEX beendet
AD 0 0 F 5	00F5 FF	Anzeigen Indexreg. X
PC	0210 10	Programmzähler zurückladen
GO	0212 30	Keine Verzweigung da Ergebnis von DEX nicht positiv!
GO	0205 EE	Verzweigung da Ergebnis von DEX negativ

Operation im SST-Modus

Bild 5.6

KAPITEL 6

ERWEITERUNG DES SYSTEMS

In den vorangegangenen Abschnitten wurde bereits erwähnt, daß der Mikroprozessor MCS 6502 auf einen Speicherbereich von 64 KByte direkt zugreifen kann. In diesem Abschnitt wollen wir uns damit befassen, wie man den Speicherbereich bzw. die Anzahl der I/O-Ports erweitert. Danach behandeln wir die Handhabung von Interrupts in einem erweiterten System.

6.1 Speicher und I/O-Erweiterung

Im KIM-1 System werden Speicherzellen und I/O-Ports völlig gleich behandelt. Es gibt keine speziellen I/O-Befehle. Die Übertragung von Daten findet zwischen der CPU und bestimmten Registern statt, angeordnet zwischen dem Daten-Bus und dem zu bedienenden I/O-Anschluß. Der Baustein 6530 enthält solche Register. Diese besitzen eine Adresse im System, genauso wie eine Speicherzelle. Wenn wir daher von Systemerweiterung sprechen, meinen wir Methoden, die den echten Speicherbereich (RAM, ROM, PROM) und die Anzahl der I/O-Ports vermehren, da beide hinsichtlich der Adressierung gleich behandelt werden.

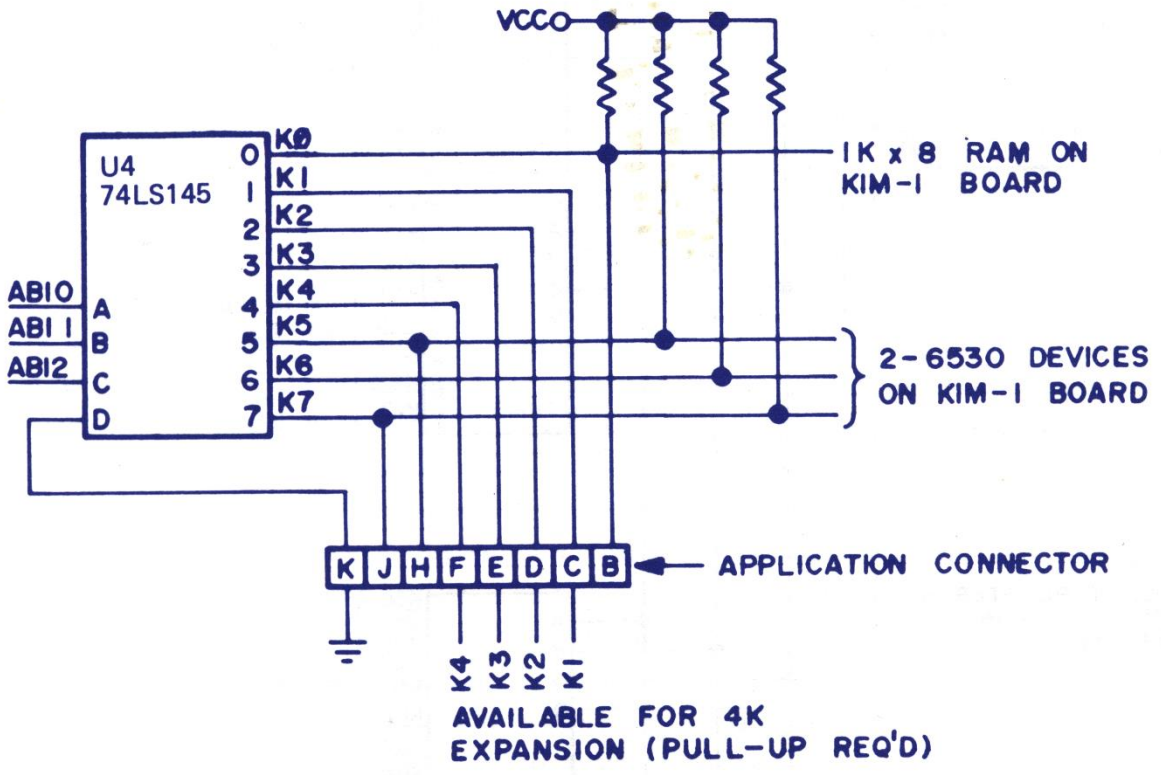
Die erste und am einfachsten durchzuführende Speichererweiterung ist das Hinzufügen eines Speicherbereichs von 4 KByte. Sie werden sich sicher erinnern, daß der niedrigste 8 KByte-Block durch einen bereits auf dem KIM-Modul untergebrachten Adreßdekoder angewählt wird. (Baustein U4 im Schaltbild.) Jeder der 8 Ausgänge des Dekoders (KØ bis K7) ist einem 1 KByte-Block im niedrigsten 8 KByte Speicher-Bereich zugeordnet. Drei der Ausgänge (K5, K6 und K7) werden dazu benutzt im Baustein 6530 enthaltene ROM, RAM, I/O und Timer-Zellen auszuwählen. Der vierte Dekoderausgang (KØ) adressiert die ebenfalls auf dem Modul angeordneten 1024 Byte RAM. Die verbleibenden Dekoderausgänge (K1, K2, K3, K4) werden vom KIM-1 Modul nicht benutzt und sind auf den Expansionsstecker herausgeführt.

Bild 6.1 zeigt eine geeignete Methode, um mit Hilfe dieser vier Dekodersignale weitere 4 KByte Speicherplätze zu adressieren. Beachten Sie, daß der Dekodereingang D auf den Applikationsstecker herausgeführt ist. Diesen Anschluß haben Sie im Kapitel 2 auf Masse gelegt. Solange er dort verweilt, wird der Dekoder immer nur Adressen im unterem 8 KByte-Block ansprechen – unabhängig vom Zustand der Leitungen AB13, AB14 und AB15.

Wollen Sie den Speicherbereich über 8 KByte hinaus erweitern, so müssen Sie dafür sorgen, daß das Massesignal von diesem Dekoder weggenommen wird. Gleichzeitig ist es nötig, ein weiteres Dekoderauswahlsignal

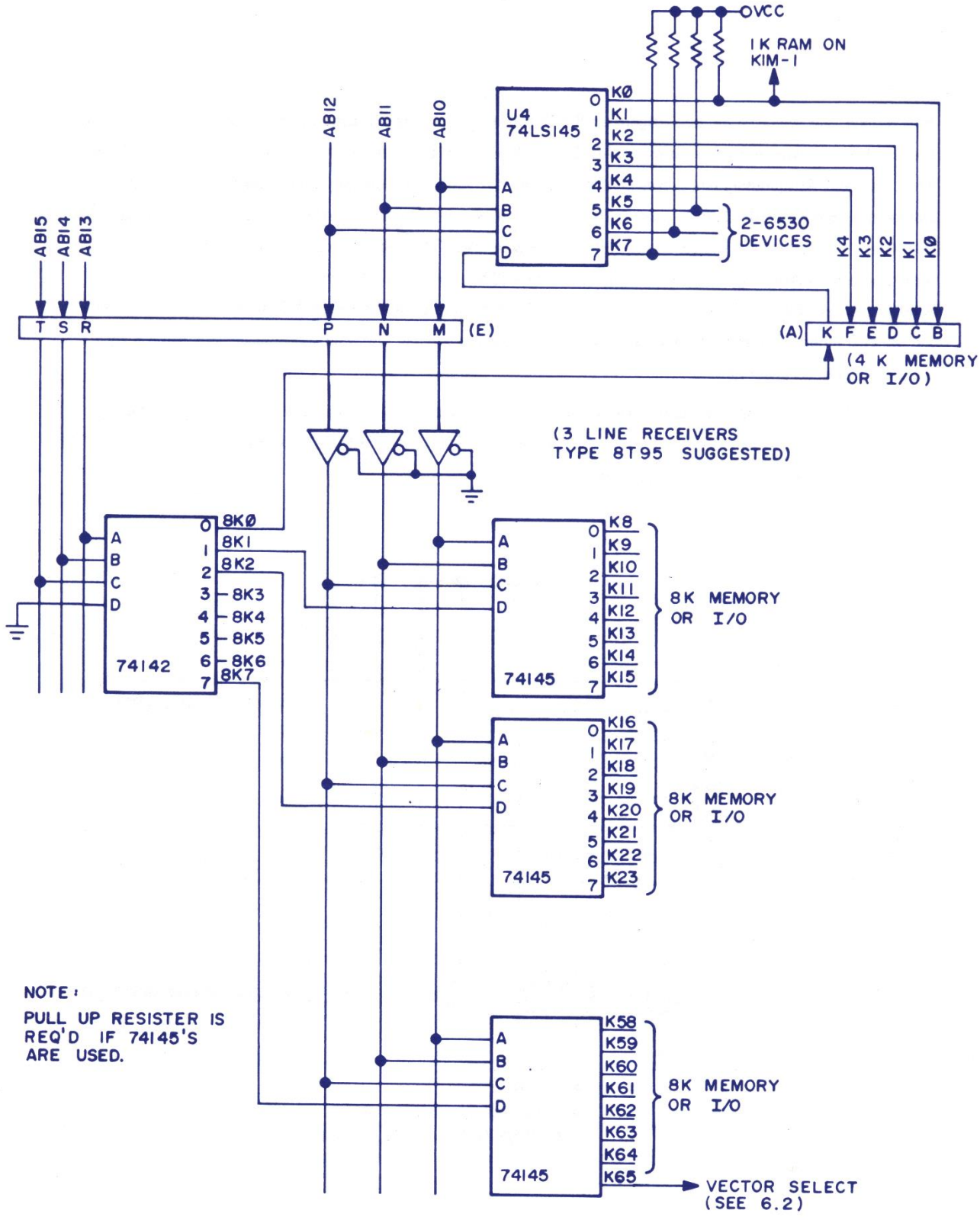
zu erzeugen, mit welchen Sie den folgenden 8 KByte Bereich aktivieren. Eine Methode ist in Bild 6.2 gezeigt. Die drei höchstbewerteten Adreßsignale AB13, AB14 und AB15 werden auf einen zusätzlichen Dekoder geführt. Die acht Ausgangssignale dieses Dekoders teilen den Gesamtadressbereich von 64 KByte in acht Unterblöcke zu je 8 KByte (8KØ, 8K1, usw.). Der Ausgang 8KØ wird nun mit dem freigewordenen D-Eingang des Dekorders U4 verbunden. Damit ist die richtige Auswahl des KIM-1 Speicherblocks wieder sichergestellt. Die restlichen sieben Ausgänge des zusätzlichen Dekoders (8K1, bis 8K7) können nun, wie in Bild 6.2 gezeigt, benutzt werden. Für jeden 8K-Speicherblock, den Sie dem ursprünglichen KIM-1 System hinzufügen, benötigen Sie also einen zusätzlichen Dekoder. Sie müssen aber nur soviele Dekoder installieren, wie Sie 8 KByte-Blöcke tatsächlich anschließen.

Eines sollten Sie jedoch bei Speichererweiterungen grundsätzlich beachten. Die Adreßbus-Leitungen der CPU dürfen nur mit einer TTL-Last beschaltet werden. Deshalb sind in Bild 6.2 drei Leitungsempfänger für die Signale AB10, AB11 und AB12 vorgesehen.



4K Speichererweiterung

Bild 6.1



Speichererweiterung 65 K

Bild 6.2

Bevor Sie sich für eine Methode der Systemerweiterung entschließen, empfehlen wir Ihnen zunächst alle Belastungsbedingungen im KIM-1 System sorgfältig zu studieren. Sie werden mit Sicherheit weitere Bausteine benötigen, wenn Ihr erweitertes System zufriedenstellend arbeiten soll.

6.2 Interruptvektor-Verarbeitung

Wir haben bereits in früheren Kapiteln auf die Interrupt-Verarbeitung beim Mikroprozessor 6502 hingewiesen. Wir empfehlen Ihnen jetzt das Kapitel 9 Ihres Programmierhandbuchs sorgfältig zu lesen; es behandelt die RESET und Interrupt-Verarbeitung.

Insgesamt gibt es drei mögliche Arten von Interrupts: RESET, NMI und IRQ, ausgelöst bei Betätigung einer der Leitungen RST, NMI oder IRQ am Mikroprozessor 6502. Der Prozessor holt daraufhin aus einem bestimmten Paar von Speicherzellen einen Wert und lädt diesen in den Programmzähler. Die Adressen dieser Speicherzellen sind durch die Hardware festgelegt und nicht vom Programmierer beeinflußbar. Die zu den einzelnen Interrupts gehörigen Adressen sind:

FFFA, FFFB - $\overline{\text{NMI}}$ Vector
FFFC, FFFD - $\overline{\text{RST}}$ Vector
FFFE, FFFF - $\overline{\text{IRQ}}$ Vector

Diese sechs Adressen liegen am oberen Ende des 64 KByte-Speicherbereichs.

Im KIM-1 System sind die drei Adreßbits AB13, AB14 und AB15 nicht mit zur Adreßdekodierung herangezogen. Wenn daher der 6502 den RST-Vektor von den Zellen FFFC und FFFD holen will, liest er in Wirklichkeit die Adressen 1FFC und 1FFD. Das bedeutet, alle Interrupt-Vektoren werden vom oberen Ende des 8 KByte Speicherblocks geholt, der im KIM-1 Grundsystem bereits dekodiert ist.

Es ist üblich, Interruptvektoren in ROM abzulegen, damit sie nach Einschalten der Versorgungsspannung sofort zur Verfügung stehen. Jedoch sollte der Programmierer das Interrupt-Sprungziel beeinflussen können. Infolgedessen enthält die Interruptzelle einen indirekten Sprungbefehl. Dieser Befehl verarbeitet den Inhalt einer bestimmten RAM-Zelle als Interrupt-Vektor, und zwar für beide Interrupts. Im KIM-1 System liegt der NMI-Vektor an der Stelle 17FA und 17FB und der IRQ-Vektor bei 17FE und 17FF. Der RST-Vektor wird nicht auf diese Weise verarbeitet; er führt das System immer in das Grundsteuerprogramm, wie nach dem Einschalten der Versorgungsspannung. Was aber geschieht, wenn wir das System über den 8 KByte Speicherbereich hinaus erweitern? Wir müssen jetzt auch AB13, AB14 und AB15 dekodieren. Die Interruptvektor-Adressen sind nicht mehr in dem mit K7 dekodierten Speicherblock enthalten. Die CPU will jetzt die Vektoren von Ihren eigentlichen Adressen (z.B. FFFA) holen. Das gleiche Problem hätte man sogar bei einem nicht erweiterten System, wenn man die Behandlung des RST Vektors und den Einsprung in die Grundstellung anders behandeln möchte, als es im KIM-1 gelöst wurde, oder wenn der RST Vektor in einem anderen Block als K7 gespeichert wäre.

Die Lösung dieses Problems besteht darin, durch eine Logikschaltung ein Signal "Interruptauswahl" zu erzeugen. Bild 6.2 verwendet dafür das Auswahlsignal für den höchsten 1 KByte-Block. Wenn ein Interrupt ausge-

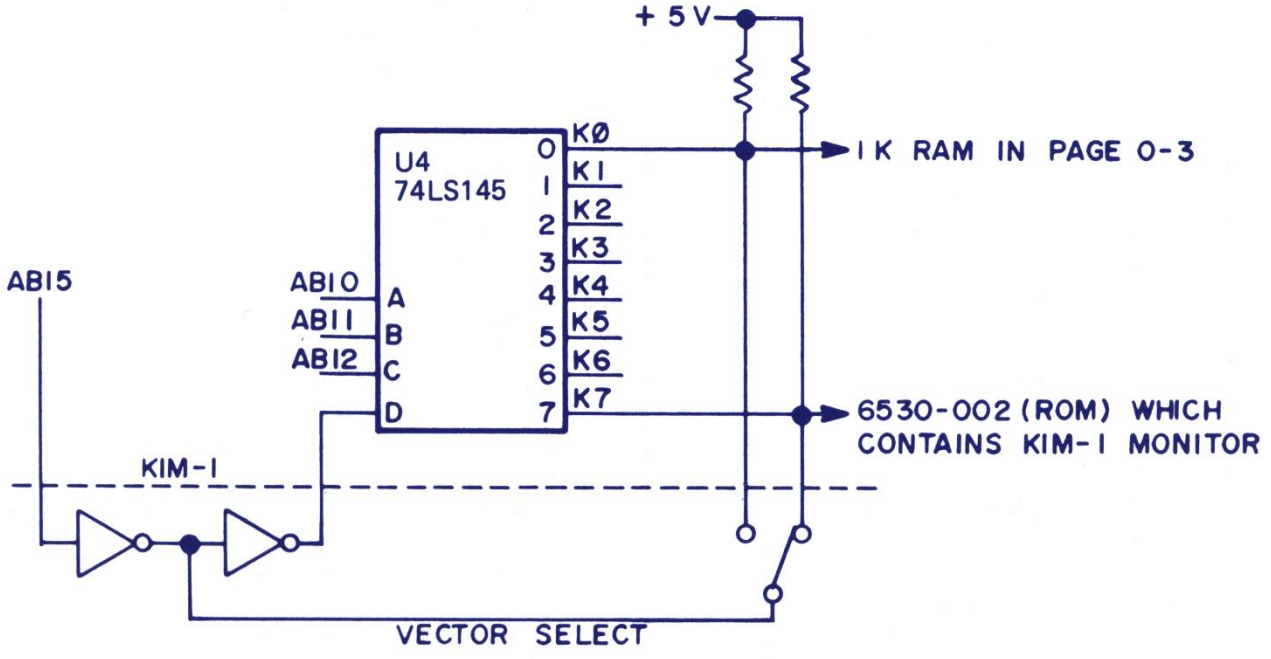
löst wird, geht diese Leitung nach log. Ø . Enthält die Position K64 den Vektor nicht, so kann man dieses Signal zusätzlich auf den K-Ausgang schalten (wire or), welcher verwendet werden soll.

Nehmen wir an, Sie hätten den Dekoderausgang K64 auf KØ geschaltet. Wenn das Signal RST auftritt, erzeugt die CPU ein Signal, welches Daten von den Adressen FFFC und FFFD holen möchte. Das Signal K64 geht nach log. Ø. Die Brücke nach KØ setzt auch diesen Dekoderausgang nach log. Ø . Dadurch wird der Interrupt-Vektor effektiv von den Adressen 03FC und 03FD gelesen. Verbinden Sie K64 mit K7, so holen Sie den Interrupt-Vektor von 1FFC und 1FFD.

Auf diese Weise werden die höchsten sechs Bytes eines jeden 1 K-Speicher-Blocks zu Interrupt-Vektoren deklariert. Es ist möglich, zur Auswahl des Speicherblocks, von welchem Sie den Vektor holen möchten, einen Schalter zu benutzen. Diese Methode ist aber nur so lange anwendbar, wie Sie einen Dekoder mit "Open Collector"- Ausgängen, z.B. den 74145, verwenden.

Bild 6.3 zeigt ein noch einfacheres Beispiel. Es geht davon aus, daß das System nur 8 KByte Speicher enthält. Der Adreßdekoder U4 wird immer dann abgeschaltet, wenn ein Interrupt auftritt, weil dann das Signal AB15 aktiv ist (AB15 entspricht obere Hälfte des Gesamtspeicherbereichs!). Das Komplement des Signals AB15 wird mit Hilfe eines Stufenschalters auf die Dekoderausgänge KØ bis K7 gelegt. Abhängig von der Stellung dieses Stufenschalters können die Interruptvektoren jetzt wahlweise von den höchsten sechs Bytes der Positionen KØ bis K7 geholt werden.

KØ entspricht im KIM-1 System einem RAM-Block, K7 ist der ROM in dem Baustein 6530-002 (Systemsteuerprogramm). Auf diese Weise besitzen Sie zwei verschiedene Interrupt-Vektoren, die Sie mit einem einfachen Kippschalter auswählen können.



Vektor-Auswahl

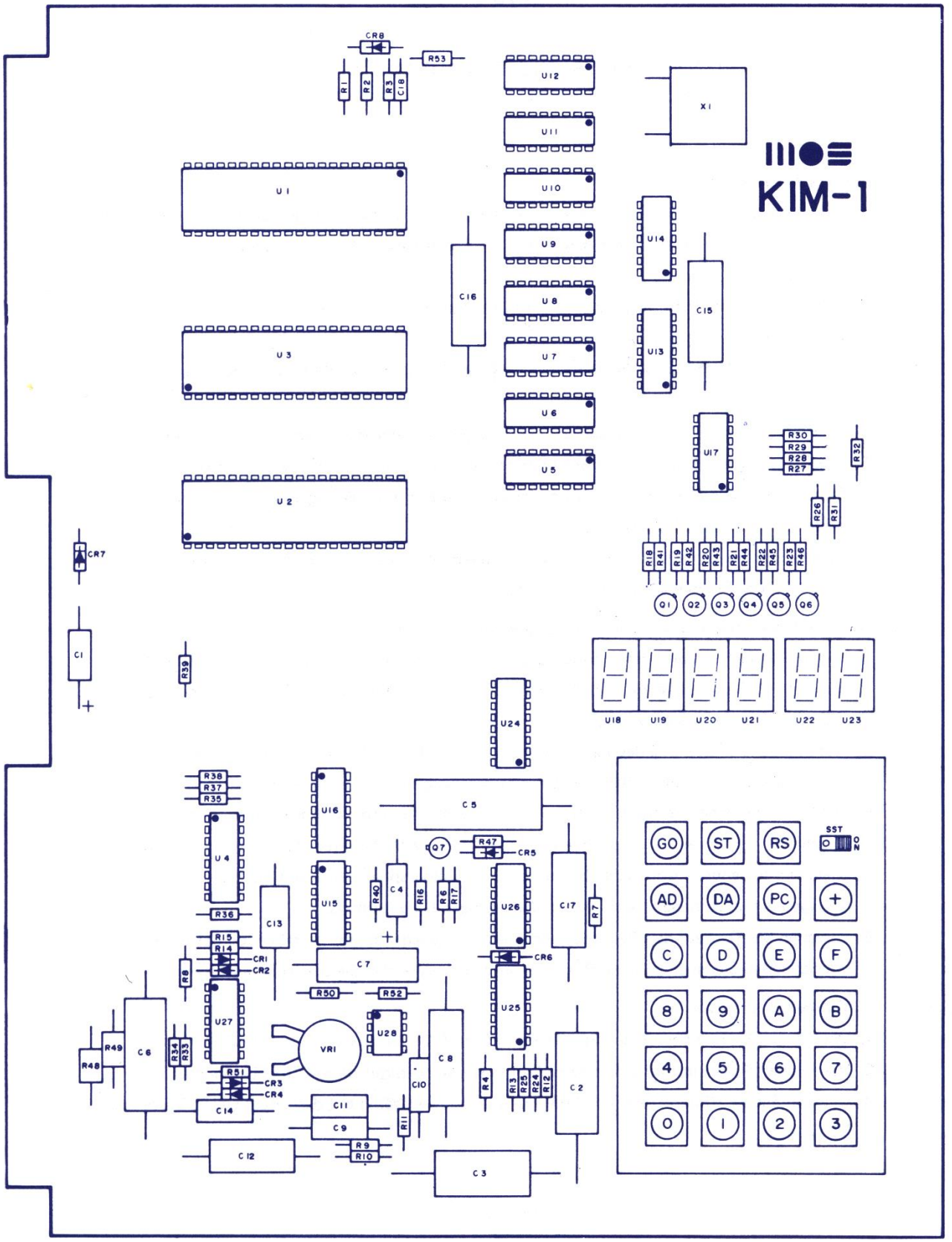
Bild 6.3

ANHANG A

ITEM	BAUTEIL	STCK.	BESCHREIBUNG
1.	U1	1	6502 Microprocessor
2.	U2	1	6530 ROM RAM I/O Chip-02
3.	U3	1	6530 ROM RAM I/O Chip-03
4.	U5 through U12	8	6102 RAM 500ns Acc,Øns
5.	U18 through U23	6	7 SEG .3" Red Display
6.	U25	1	556 Timer IC
7.	U27	1	565 Phase Lock Loop
8.	U28	1	311 Comparator
9.	U24	1	74145 BCD Decoder IC
10.	U13 & U14	2	74125 TRI STATE Buffer
11.	U15	1	7400 Quad Nand IC
12.	U16	1	7404 Hex Inverter IC
13.	U17	1	7406 Hex Inv. O/C IC
14.	U26	1	7438 Quad Nand O/C IC
15.	CR1,2,3,4,&8	5	20 MA. 50v Diode - IN914
16.	CR5, CR6	2	1A 50v Diode - IN4001
17.	CR7	1	6.2v ½w Z. Diode - IN4735
18.	Q7	1	NPN Transistor B>20, VCE>12 - 2N5371
19.	Q1 through Q6	6	PNP Transistor B>20, VCE>6 - 2N5375
20.	R24 & R25	2	47KΩ ±10% ¼w Resistor
21.	R1,2,3,4, & 6	5	3.3KΩ ±10% ¼w Resistor
22.	R34 & R50	2	2.2KΩ ±10% ¼w Resistor
23.	R12-R17, R41-R46	12	1.0KΩ ±10% ¼w Resistor
24.	R35 through R40	6	560Ω ±10% ¼w Resistor
25.	R18-R23, R47	7	220Ω ±10% ¼w Resistor
26.	R33	1	47Ω ±10% ¼w Resistor
27.	R52	1	5 Meg. ±10% ¼w Resistor
28.	R51	1	30KΩ ±5% ¼w Resistor
29.	R7,R8,R9,R10&R11	5	10KΩ ±5% ¼w Resistor
30.	R48, R49	2	150Ω ±5% ½w
31.	R26 through R32	7	82Ω ±5% ¼w
32.	VR1	1	5KΩ Potentiometer
33.	C2, C3, C6	3	.22±10% uf.>12 wv. cap
34.	C1, C4	2	1uf+80-10%>12WV Cap
35.	C5	1	.33 uf±10%>12WV Cap
36.	C7,C8,C15,C16,C17	5	.1uf+80-10%>12WV Cap
37.	C9, C10, C11	3	.0068uf±10%>12WV
38.	C12	1	.047uf±10%>12WV
39.	C13	1	.022uf±10%>12WV
40.	C14	1	.001uf±10%>12WV
41.		1	44 Pin Edge Conn.
42.	X1	1	1 MHz XTAL
43.		1	PCB.
44.		1	24 Key KBD
45.		6	Rubber Pads
46.		1	Shipping Bag (Static Free)
47.		1	Shipping Box
48.		1	Hardware Manual
49.		1	Software Manual
50.		1	KIM Manual
51.		1	Warranty Card
52.		1	Wall Chart
53.		2	#2 x ¼ SS Screws (Keyboard)
54.		1	Program Card
55.	C18	1	10pf CAP
56.	R53	1	330K ¼w Resistor
57.	U4	1	74LS145 BCD Decoder IC

ANHANG B

BESTÜCKUNGSPLAN KIM-1



ANHANG C

WENN SCHWIERIGKEITEN AUFTRETEN

Symptom: **Anzeige leuchtet nicht**

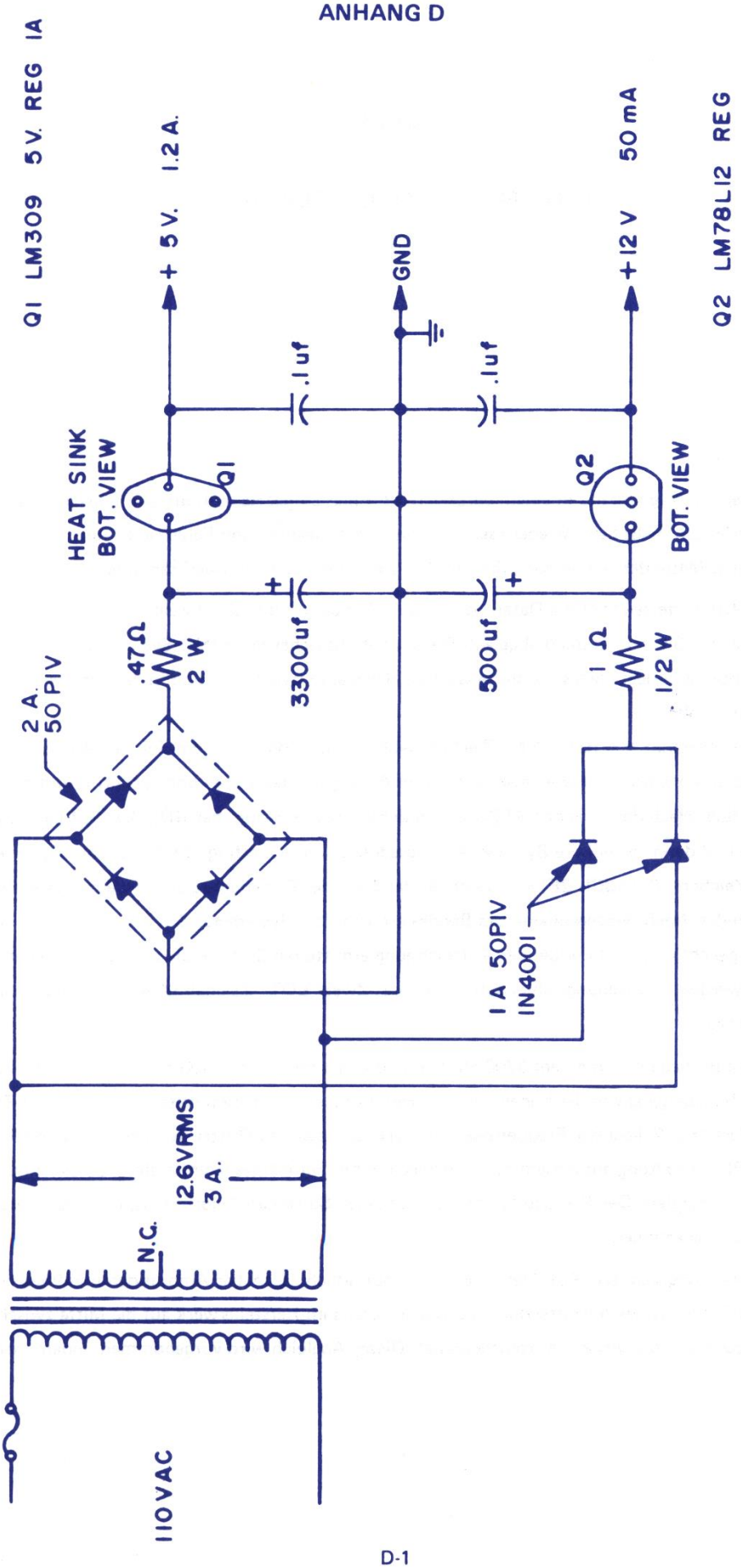
1. Prüfen Sie die 5 Volt Spannungsversorgung. Messen Sie, ob zwischen den Stiften E-21 und E-22, außerdem zwischen den Stiften A-A und A-1, +5 Volt liegen. Die Systemspannung sollte innerhalb +5 V $\pm$ 5% liegen.
2. Prüfen Sie die Verdrahtung nach Bild 2.4 (TTY-Tastenfeld). Stift A-21 darf nicht mit Stift A-V verbunden sein.
3. Untersuchen Sie ob der Dekoder angewählt ist. Prüfen Sie gemäß Bild 2.2, ob Stift A-K auf log. Ø liegt.
4. Drücken Sie die RESET-Taste und untersuchen Sie, ob keine der übrigen Tasten klemmt.
5. Schalten Sie einen Spannungsmesser zwischen die Anschlüsse E-21 (+5 V) und Stift E-7 (RESET). Drücken Sie die RESET-Taste kurz und lassen Sie wieder los. Die angezeigte Spannung muß zwischen +1 V und +4 V wechseln.
6. Überzeugen Sie sich mit Hilfe eines Oszillographen, ob an dem Stift E-V (System-Takt) ein Signal mit der Frequenz von 1 MHz auftritt.

Symptom: **Keine Datenübertragung zum oder vom Magnetband möglich**

1. Prüfen Sie die +12 Volt Spannungsversorgung. Messen Sie, ob zwischen den Anschlüssen A-N und A-1 +12 Volt auftreten. Stift A-1 muß log. Ø entsprechen. Stellen Sie die Spannung auf +12 $\pm$ 5% Volt ein (siehe Bild 2.2).
2. Prüfen Sie die Einstellung des Lautstärkereglers – stellen Sie ihn auf Mittenstellung.
3. Überzeugen Sie sich davon, daß Sie den richtigen Ausgangs-Pin benutzen (s. Bild 2.3).
4. Prüfen Sie die Band-Interfaceschaltung, indem Sie den Kassetten-Rekorder abstecken und die Anschlüsse A-P mit A-L kurzschließen. Starten Sie den KIM-Monitor, damit er einen Teil des Speichers aussendet. Beobachten Sie mit einem Oszillographen den Anschluß E-X (PLL TEST). Schlagen Sie im Anhang E nach, wegen des richtigen Datenformats.
5. Sprechen Sie Text auf den Rekorder und spielen Sie wieder ab, um zu sehen ob das Gerät arbeitet. Schließen Sie einen anderen Rekorder an, oder verwenden Sie eine andere Kassette.

Symptom: **TTY-Interface Problem**

1. Überzeugen Sie sich davon, daß Stift A-V mit Stift A-21 verbunden ist (s. Bild 2.4).
2. Vergleichen Sie die in Bild 2.4 dargestellten Verbindungen mit Ihrem TTY-Handbuch.
3. Drücken Sie die RESET-Taste am KIM-Tastenfeld abwechselnd mit der "RUB OUT"-Taste am TTY.


Q1 LM309 5V. REG 1A
+ 5 V. 1.2 A.
GND
+12 V 50 mA
Q2 LM78L12 REG
.1uf
.1uf
HEAT SINK
BOT. VIEW
Q1
Q2
BOT. VIEW
+
+
3300uf
500uf
.47Ω
2 W
1 Ω
1/2 W
2 A.
50 PIV
1A 50PIV
IN4001
12.6VRMS
3 A.
N.C.
110VAC

ANHANG E

DATENFORMAT MAGNETBAND

Die Daten werden am Magnetband in einem besonderen Format aufgezeichnet, um ein fehlerfreies Wiedererkennen zu gewährleisten. Falls beim Wiederlesen von Daten vom Magnetband Fehler auftreten, wurden mehrere Fehlererkennungs-Methoden vorgesehen, die den Anwender rechtzeitig aufmerksam machen.

Das Band im ASCII-Kode zeichnet die Daten seriell auf. Aus dem 7-Bit ASCII-Zeichen wird ein Parity-Bit gewonnen und als achtes Bit mit am Band abgelegt. Die aus dem Speicher des KIM-1 Systems gelesenen Bytes werden in zwei Halbbytes zerlegt. Jedes Halbbyte wird als Hexazahl interpretiert und in das entsprechende 7-Bit ASCII-Zeichen verwandelt.

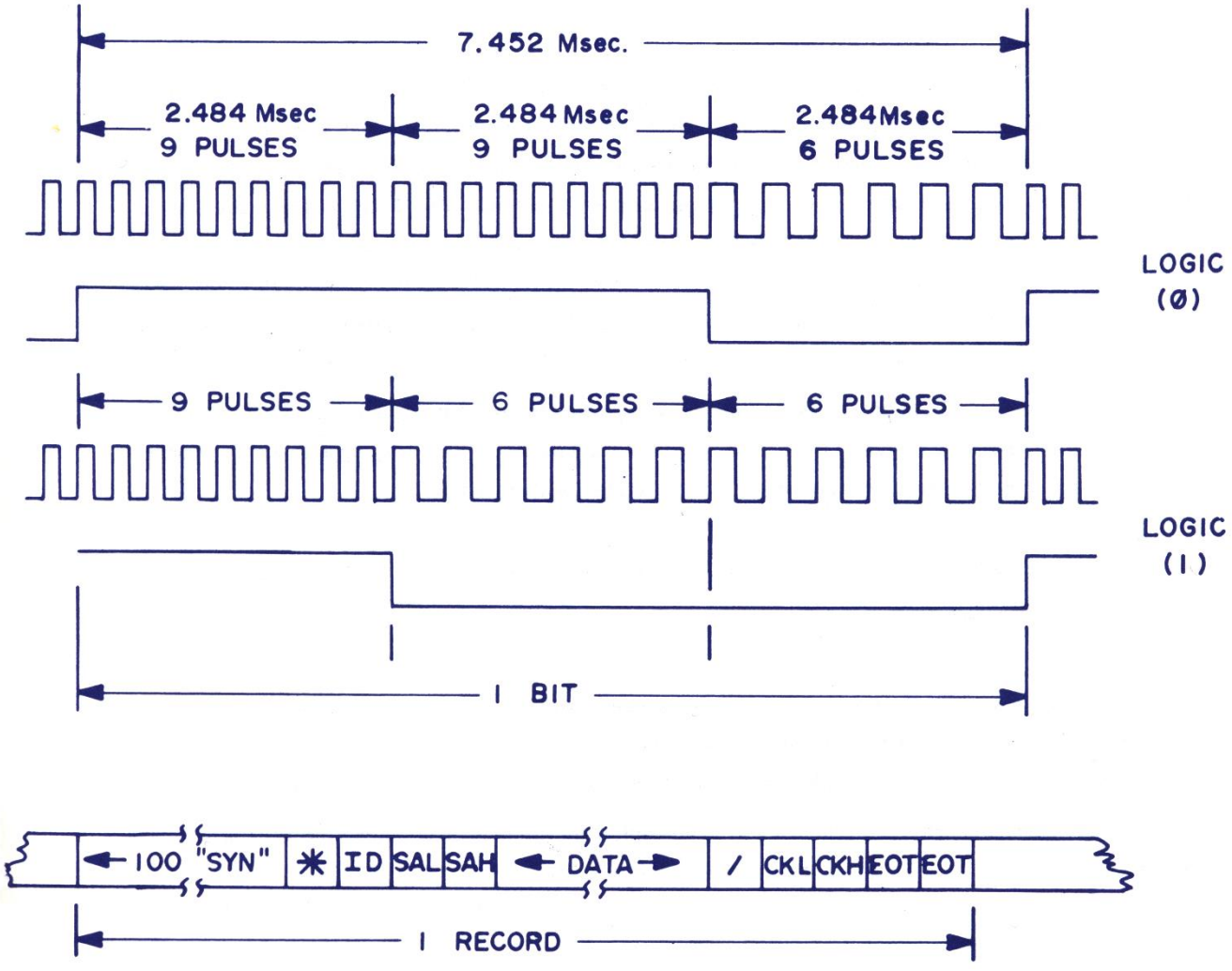
Jede Aufzeichnung beginnt mit 100 "SYN"-Zeichen (ASCII 16), gefolgt von dem Zeichen ASCII 2A. Bei Einlesen des Bandes erkennt der Mikroprozessor somit den Anfang gültiger Daten und synchronisiert sich auf den seriellen Datenfluß. Nach dem Zeichen ASCII 2A sendet er die Kenn-Nummer (ID), das niedrigere Byte der Anfangsadresse und dann das höhere Byte der Anfangsadresse. Danach erfolgt die Aufzeichnung der eigentlichen Daten. Das Zeichen "/" (ASCII 2F) kennzeichnet das Ende des Datenteils. Danach werden zwei Prüfsummen-Bytes gesendet. Nach Wiedereinlesen des Bandes ermittelt das System aus den erkannten Daten eine Prüfsumme und vergleicht diese mit der bei der Aufzeichnung ermittelten Summe, die auf dem gelesenen Band steht. Damit werden Übertragungsfehler sofort erkannt. Zwei "EOT"-Zeichen (ASCII 04) bilden das Ende der Übertragung.

Jedes übertragene Bit beginnt mit einem 3700 Hz Ton und endet mit einem 2400 Hz Ton. Bei Bits mit dem Wert log. 1 erfolgt der Übergang von der hohen zur niedrigen Frequenz nach dem ersten Drittel der Bit-Zeit. Bei Bits mit dem Wert Log. Ø liegt der Frequenzwechsel nach dem zweiten Drittel der Bit-Zeit. Beim Wiederlesen erkennt eine PLL-Schaltung aus diesem Frequenzwechsel mit Hilfe eines Komparators, ob es sich um eine log. Ø oder log. 1 handelt. Der Mikrocomputer benutzt einen Sofrtware-Algorithmus, um diese Signale in 8-Bit Datenworte zu übertragen.

Die zur Datenerkennung benutzte PLL-Methode ist unempfindlich gegenüber Amplituden und Phasenschwankungen. Die Frequenz des freilaufenden PLL wurde bereits im Herstellerwerk auf die Mitte zwischen den beiden Eckfrequenzen abgeglichen (Mittenfrequenz). Dieser Abgleich wird vorgenommen, indem man

die Anschlüsse A-P mit A-L verbindet. Ein Programm, das bei 1A6B beginnt, erzeugt die Mittelfrequenz und gestattet das Abgleichen der Regelschleife mit dem Potentiometer VR1. Der Anschluß E-X wird dazu mit einem Voltmeter verbunden und das Poti solange verändert, bis die Voltmeteranzeige von +5 Volt auf 0 Volt umpringt (bzw. von 0 Volt nach +5 Volt je nach Drehsinn).

Dieser Abgleich wurde bereits im Herstellerwerk vorgenommen; er muß nur nach Auswechseln eines Bauelements wiederholt werden.



Datenformat Magnetband

Bild E-1

ANHANG F

DATENFORMAT LOCHSTREIFEN

Die Lochstreifen Lade- und Stanz-Routinen verwenden ein bestimmtes Datenformat, welches einen fehlerfreien Betrieb gewährleistet. Jedes übertragene Zeichen wird in zwei Halbbytes zerlegt. Diese Halbbytes werden als Hexazeichen interpretiert, in das entsprechende ASCII-Zeichen verwandelt und dann in den Lochstreifen gestanzt. Jede Aufzeichnung beginnt mit dem Zeichen ";" (ASCII 3B). Das als nächstes übertragene Zeichen enthält die Anzahl der zu übertragenden Zeichen (18 Hex). Die Übertragung startet mit Adresse "low" (1 Byte, 2 Zeichen) dann mit Adresse "high" (1 Byte, 2 Zeichen). Danach folgen die Daten (18 Bytes, 36 Zeichen). Jede Aufzeichnung wird von einer Prüfsumme beendet (2 Byte, 4 Zeichen), dann folgen ein Wagenrücklauf (ASCII Ø D), ein Zeilenvorschub (ASCII Ø A) und 6 "NULL"-Zeichen (ASCII ØØ).

Der zuletztgesendete Block enthält keine Datenbytes (gekennzeichnet durch ;ØØ). Das Endadressenfeld wird durch eine vierstellige Hexazahl ersetzt, welche die Anzahl der in der Übertragung enthaltenen Blöcke angibt. Danach folgt die Prüfsumme und schließlich als letztes Zeichen "XOFF".

Beispiel:

```
;180000FFEEDDCCBBAA00998877665544332211223344556677889 90AFC
;0000010001
```

Beim Einlesen eines Lochstreifens werden alle Daten überlesen bis das Zeichen ";" erscheint. Wird ein Nicht-ASCII-Zeichen oder eine falsche Prüfsumme gelesen, so erkennt das KIM-1 System diesen Fehler. Die Prüfsumme wird ermittelt, indem das Programm alle Zeichen außer dem Zeichen ";" addiert.

Das beschriebene Lochstreifenformat ist kompatibel mit allen von MOS Technology Inc. gelieferten Lochstreifen.

ANHANG G

6502 CHARAKTERISTIK

Takt-Signale ($\emptyset_1$, $\emptyset_2$)

Der Taktgenerator des MCS 6502 befindet sich auf dem Chip. Die Frequenz ist quarzgesteuert.

Adreß-Bus (A_0 bis A_{15})

Diese Ausgänge sind TTL-kompatibel und können eine TTL-Last bei 130 pF treiben.

Daten-Bus (D_0 bis D_7)

Der Datenbus besitzt acht Leitungen. Er ist bidirektional und überträgt Signale von der CPU zur Peripherie und umgekehrt. Alle Ausgänge sind Tristate-Buffer; sie treiben eine TTL-Last bei 130 pF.

Ready (RDY)

Dieses Eingangssignal gestattet dem Benutzer, den Prozessor in Einzelzyklus-Mode zu betreiben, ausgenommen bei den Schreibzyklen. Eine negative Flanke während der Taktzeit von $\emptyset_1$ hält den Prozessor an; die Adreßleitungen zeigen dabei die Adresse des soeben gelesenen Wertes. Dieser Zustand dauert solange an, bis während eines Taktes $\emptyset_2$ das RDY-Signal wieder auf log. 1 geht. Dadurch ist es möglich, auch mit sehr langsamen Speichern zusammenzuarbeiten, und "DMA"-Operationen durchzuführen. Wenn Ready in einem Schreibzyklus log. $\emptyset$ ist, wird es bis zum nächsten Lesezyklus nicht ausgewertet.

Interrupt-Anforderung (IRQ)

Dieser TTL-kompatible Eingang bewirkt den Start eines Interrupt-Zyklus innerhalb der CPU. Der Prozessor beendet den Befehl, den er vor Eintreffen des Interruptsignals begonnen hat. Dann wird das Interrupt-Maskenbit im Status-Koderegister abgefragt. Ist das Interrupt-Maskenflag nicht gesetzt, so beginnt der Prozessor mit der Abarbeitung des Interrupts. Der Programmzähler und das Status-Register werden im Stack abgelegt. Jetzt setzt der Prozessor das Interrupt-Masken-Flag, damit kein weiterer Interrupt ausgelöst wird. Danach wird das niedere Byte des Programmzählers vom Speicherplatz FFFE und das höhere Byte vom Speicherplatz FFFF geladen. Das Programm fährt an der Stelle fort, die durch den in diesen Speicherzellen enthaltenen Vektor gegeben ist. Damit ein Interrupt ausgelöst werden kann, muß das RDY-Signal log. 1 sein. Soll dieser Eingang als "wired or" betrieben werden, so wird ein "Latch up"-Widerstand von 3 KOhm empfohlen.

Nichtmaskierbarer Interrupt (NMI)

Eine negative Flanke an diesem Eingang bewirkt das Auslösen eines nichtmaskierbaren Interrupts.

NMI ist ein unbedingter Interrupt. Die Operation verläuft grundsätzlich wie unter IRQ, es wird nur das Interrupt-Maskenflag nicht ausgewertet und der Vektor von den Adressen FFFA und FFFB geholt. Auch hier wird im Fall von "wired or"-Betrieb ein "Latch up"-Widerstand von 3 KOhm empfohlen.

Die Eingänge IRQ und NMI sind Hardware-Interrupts, die während $\emptyset_2$ abgefragt werden und bei $\emptyset_1$ beginnen, sobald der letzte Befehl abgearbeitet ist.

Überlauf-Flag setzen (S.O)

Dieser TTL-kompatible Eingang gestattet es, auf das Überlauf-Flag im Status-Koderegister zuzugreifen.

SYNC

Dieses Ausgangssignal zeigt an, wenn der Prozessor einen neuen Befehl einliest. Die SYNC-Leitung geht auf log. 1 während eines $\emptyset_1$-Taktes innerhalb einer Befehlsholphase und bleibt in diesem Zustand für den Rest des Zyklus. Wenn die RDY-Leitung während des $\emptyset_1$-Taktes nach log. Ø gebracht wird, in dem SYNC nach log. 1 geht, hält der Prozessor sofort an und verweilt dort bis RDY wieder nach log. 1 geschaltet wird. Daher kann das SYNC Signal dazu benutzt werden, um die RDY-Leitung zu steuern.

RESET

Dieser Eingang dient dazu den Prozessor nach dem Einschalten der Versorgungsspannungen zu starten. Solange diese Leitung log. Ø bleibt, ist jeglicher Datenaustausch mit dem Prozessor unterbunden. Erkennt der Prozessor eine positive Flanke an diesem Eingang, beginnt er sofort einen RESET-Zyklus.

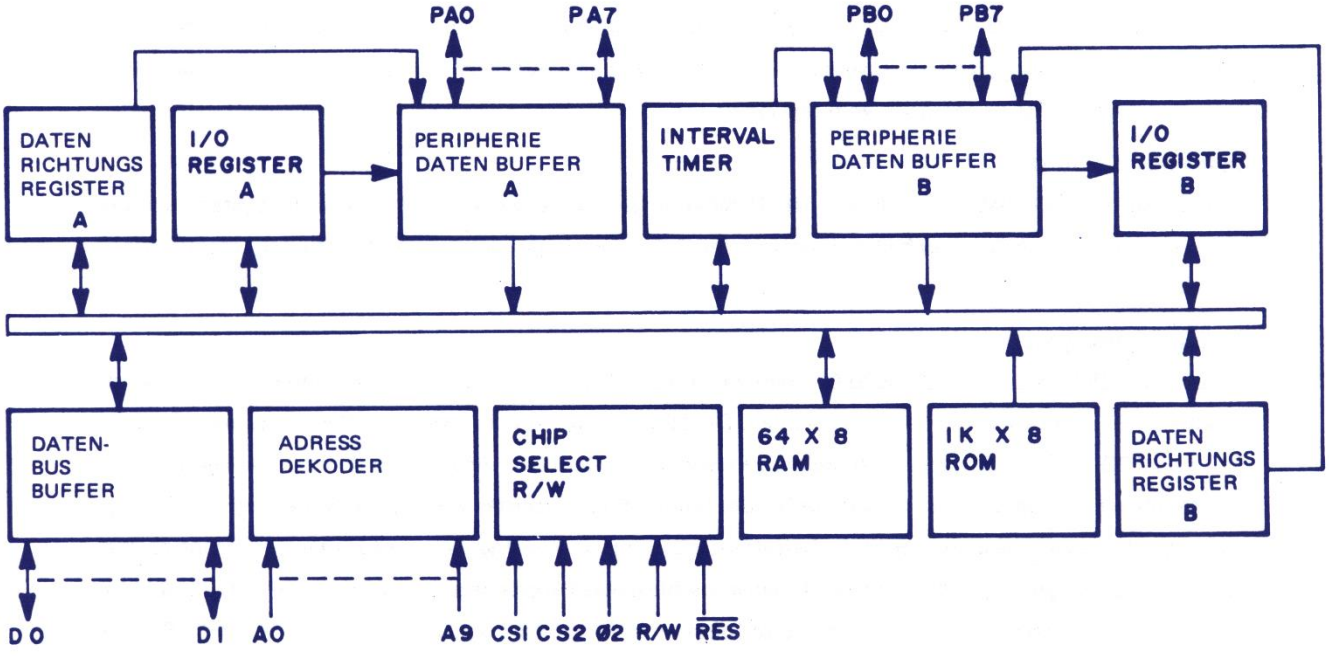
Nach einer Einlaufzeit von 6 Takten, setzt der Prozessor das Interrupt-Maskenflag und lädt den Programmzähler aus den Speicherzellen FFFC und FFFD. Dort steht der Startvektor für die Programmsteuerung. Hat die Versorgungsspannung nach dem Einschalten des Versorgungsnetzes 4,75 Volt erreicht, muß der RESET-Eingang noch für zwei Takte auf log. Ø gehalten werden.

Hat das RESET-Signal nach diesen beiden Takten log. 1 erreicht, beginnt der Prozessor mit der oben beschriebenen RESET-Prozedur.

ANHANG H

6530-CHARAKTERISTIK

Der MCS 6530 ist ein Mehrzweckbaustein zum Einsatz in der MCS 650X-Mikroprozessor-Familie. Er enthält einen maskenprogrammierbaren ROM mit der Kapazität 1K x 8 bits, einen statischen RAM mit der Kapazität von 64 x 8 bits, zwei softwaregesteuerte bidirektionale 8-Bit I/O-Ports und einem softwaregesteuerten Timer mit Interruptauslösung für 1 bis 262144 Taktperioden.



MCS 6530 Blockschaltbild

Bild H-1

RESET (RES)

Wird das System in Grundstellung gebracht, bewirkt eine log. Ø am RES-Eingang das Löschen aller vier I/O-Register. Das wiederum schaltet alle Datenports als Eingänge. So verhindert man, daß externe Geräte unerlaubt angesteuert werden. Die Treiber für den Datenbus sind dann ausgeschaltet. Auch die Interrupterkennung ist unterbrochen. Das RES-Signal muß für mindestens eine Taktperiode auf log. Ø gehalten werden.

Takt-Eingang

An den Takteingang ist der $Ø_2$-Systemtakt anzuschließen. Dabei genügt ein Hub von 0,4 Volt bis 2,4 Volt; aber auch ein Signal mit höherem Pegel (0,2 Volt bis +5 Volt $^{+0,3}_{-0,2}$) ist zulässig.

Lesen/Schreiben (R/W)

Der Prozessor liefert das RW Signal. Es dient zur Datenübertragung in beide Richtungen. Ein log. 1 auf der RW-Leitung gestattet es dem Prozessor, Daten vom 6530 zu lesen, sofern die Adresse richtig gewählt wurde. Log. Ø auf der RW-Leitung erlaubt es dem Prozessor, Daten in den 6530 zu schreiben.

Interrupt-Anforderung *(IRQ)*

Der IRQ-Anschluß gehört zu dem Timer. Dieser Stift kann, wenn er nicht zur Interruptauslösung verwendet wird, als zusätzlicher I/O-Kanal eingesetzt werden. Bei Verwendung als Interruptpin muß er vom Datenrichtungsregister als Eingang deklariert werden. Der Interruptpin liegt normalerweise auf log. 1; geht er nach log. Ø, so wertet das die CPU als Interrupt vom MCS6530.

Daten-Bus (D_0 bis D_7)

Der Baustein 6530 hat acht bidirektionale Datenleitungen. Diese werden mit dem Bus des Systems verbunden und gestatten den Austausch von Daten mit der CPU. Die Ausgangspuffer werden nur bei Leseoperationen aktiv.

Peripherie-Datenports

Der MCS 6530-002 und MCS 6530-003 besitzen je 15 I/O-Leitungen. Jede Leitung kann durch Programmbefehle als Ein- oder Ausgang deklariert werden. Die 15 Leitungen sind in zwei 8-Bit Ports unterteilt, PAØ-PA7 und PBØ- PB7. PB6 wird als Chipselekt verwendet und steht dem Anwender nicht zur Verfügung. Das Einschreiben einer log. Ø in das zugehörige bit des Datenrichtungsregister deklariert eine Leitung als Eingang. Wenn log, 1 eingeschrieben ist, wirkt die betreffende Leitung als Ausgang. Ist eine Leitung als Eingang geschaltet, so liegt am zugehörigen Ausgang log. 1; ein angeschlossenes Peripheriegerät wird mit einer TTL-Last belegt. Bei einer Leseoperation liest der Mikroprozessor direkt den Peripherieanschluß. Umgekehrt, wenn die Peripherie vom MCS 6530 angesteuert wird, sind die Daten in einem Register zwischengespeichert. Der Prozessor liest sicher den richtigen Wert, solange am Eingang des 6530 für log. 1 mehr als 2 Volt zur Verfügung stehen und für log. Ø weniger als 0,8 Volt. Alle Peripherieanschlüsse sind TTL-kompatibel. Die Anschlüsse PAØ und PBØ können bei 1,5 Volt 3mA abgeben und direkt zur Ansteuerung von Darlington-Transistoren verwendet werden. Die Leitung PB7 besitzt keinen internen "Pull-Up-Widerstand und dient daher z.B. für "wired-or"-Schaltungen.

Adreß-Leitungen (AØ- A9)

Es sind 10 Adreßleitungen und ein ROM-SELECT-Anschluß vorhanden. Diese finden immer als Adreßleitungen Verwendung. Außerdem gibt es noch zwei Stifte, die über die Maske entweder für allgemeine Verwendung einzeln, oder als weitere Chipselect-Leitungen gemeinsam verwendet werden. Sie heißen: PB5 und PB6. Sollen sie als Peripherieanschlüsse dienen, so sind sie keine Chipselect-Leitungen. Hier wird PB5 als Datenpin und PB6 als Chipselect verwendet. Ein Blockschaltbild wird in Bild H-1 gezeigt. Der MCS 6530 besteht aus vier Grundeinheiten: RAM, ROM I/O und Timer. Der RAM und ROM sind direkt mit dem Mikroprozessor über dessen Daten und Adreßbus verbunden. Der I/O-Bereich besteht aus zwei 8-Bit Hälften. Jede Hälfte enthält ein Datenrichtungsregister (DDR) und ein I/O-Register.

ROM 1KByte

Der ROM besitzt ein Format von 1024 x 8 bit. Die Adreßleitungen AØ-A9 und RSØ werden benötigt, um den ganzen ROM anzusprechen. Wenn man CS1 und CS2 dazu nimmt, können sieben MCS 6530 adressiert werden, das entspricht 7168 x 8 bit ROM.

RAM 64 Bytes

Ein statischer RAM mit der Organisation 64 x 8 ist in dem MCS 6530 enthalten. Er wird adressiert über AØ -A5 (Byte-select) RSØ, A6, A7, A8, A9 und CS1.

Interne Peripherie-Register

Es gibt vier interne Register: zwei Daten-Richtungsregister, und zwei I/O-Daten-Register. Die Datenrichtungsregister (Seite A und Seite B) legen fest, welche Leitungen Eingänge und welche Ausgänge darstellen. Eine log. 1 deklariert den zugehörigen Pin als Ausgang. Deshalb erscheint alles, was in dieses Register geschrieben wird, am entsprechenden Peripherieanschluß. Einschreiben von log. Ø in das Datenrichtungsregister bewirkt, daß alle Registerausgänge zur Peripherie abgeschaltet werden. Ein Beispiel: Wenn wir in das Datenrichtungsregister A Position 3 ein log. 1 laden, dann schalten wir den Peripherieanschluß PA3 als Ausgang. Hätte man eine log. Ø geladen, wäre PA3 ein Dateneingang. Die beiden I/O-Register werden zur Zwischenspeicherung der Prozessorinformation verwendet, bis das Peripheriegerät diese Daten übernehmen kann.

Während einer Leseoperation liest der Prozessor nicht diese Register, sondern direkt die Peripherieanschlüsse. Der Prozessor kann auch die Anschlüsse lesen, welche als Ausgänge deklariert sind. Die Daten der I/O-Register können nur durch eine Schreiboperation des Prozessors geändert werden. Der Inhalt des I/O-Registers wird durch eine Leseoperation des Prozessors nicht verändert.

Der Timer

Der Timer-Bereich des MCS 6530 enthält drei Teile: einen Frequenzteiler, ein programmierbares 8-Bit-Register und eine Interrupt-Logik. Diese Teile sind in Bild 4 dargestellt.

Der Timer zählt bis 256 Zeitintervalle. Jedes Zeitintervall kann sein: 1T, 8T, 64T, 1024T. Dabei ist T eine Periode des Systemtaktes. Wenn ein programmiertes Zeitintervall abgelaufen ist, wird ein Interrupt-Flag gesetzt. Danach zählt der Timer abwärts bis maximal −255T. Auf diese Weise kann eine Routine, die

mit dem Setzen des Interrupt-Flags gestartet wurde, durch Lesen des Timerinhalts feststellen, wieviel Zeit seit dem Setzen des Interrupt-Flags vergangen ist.

Der 8-Bit System Daten-Bus dient zum Laden und Lesen des Timers. Zum Zählen von 52 Zeitintervallen muß das Bitmuster 00110100 in das Register des Timers geschrieben werden.

Sofort wenn die Daten in das Timer-Register geschrieben werden, übernimmt der Timer von den Adreß-Leitungen AØ und A1 die gewünschten Zeitintervalle (1, 8, 64, 1024T). Während einer Lese- oder Schreib-Operation steuert die Adreßleitung A3 die Interrupt-Leitung PB7. A3 = log. 1 bereitet IRQ über PB7 vor; A3 = log. Ø hebt den Interrupt auf. Wenn PB7 zusammen mit dem Timer als Interrupt-Flag dient, ist es als Eingang zu programmieren. Wenn PB7 durch A3 vorbereitet ist und ein Interrupt auftritt, geht PB7 auf Log. Ø. Wird der Timer gelesen, bevor das Interrupt-Flag gesetzt ist, erkennt der Prozessor die noch verbliebenen Zeitintervalle (51, 50, 49, usw.).

Hat der Timer bis auf 00000000 gezählt, wird beim nächsten Takt ein Interrupt ausgelöst und der Zähler geht in die Stellung 11111111. Nach dem Interrupt wird der Zählerstand linear-abwärts gezählt. Liest der Prozessor den Timer nach Auftreten des Interrupts, und erkennt er den Wert 11100100, so ist seit dem Interrupt eine Zeit von 28T vergangen. Der gelesene Wert ist das Zweierkomplement.

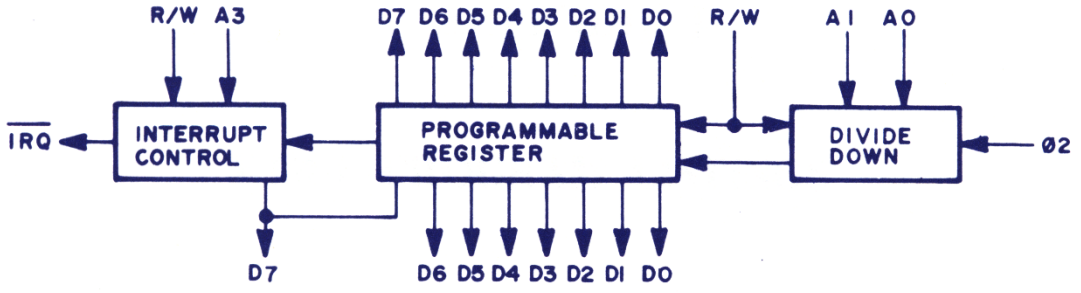
Gelesener Wert:	11100100
Komplement:	00011011
Addiere 1	00011100 = 28!

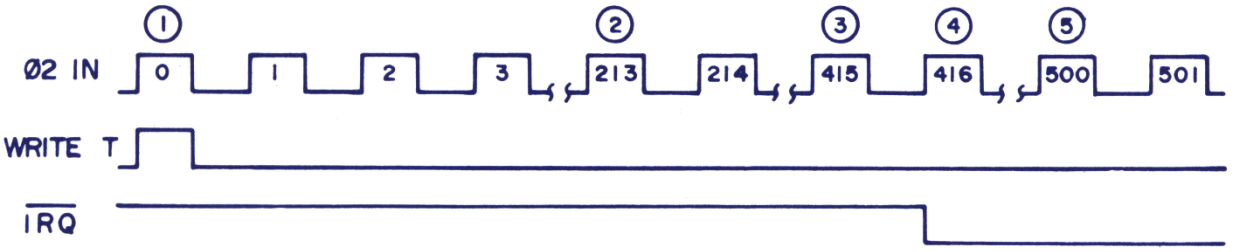
Sie erhalten also die im Ganzen verstrichene Zeit durch eine Zweierkomplement-Addition mit dem ursprünglich in den Timer geladenen Wert.

Dazu ein weiteres Beispiel:

Die in den Timer geschriebene Zeit sei 00110100 (= 52). Mit einer Teilung in Achterschritten ist die gesamte Zeit bis zum Interrupt (52 x 8) + 1 = 417T. Wenn nach dem Interrupt der Wert 11100100 gelesen wurde, dann ist die bis dahin verstrichene Zeit: 416T + 28T = 444T.

Wenn der Timer nach dem Interrupt gelesen oder neu geladen wird, erfolgt ein Rücksetzen der Interruptmarke. Wenn jedoch der Timer zur gleichen Zeit gelesen wird, zu der der Interrupt auftritt, wird die Interruptmarke nicht zurückgesetzt.





1. In den Timer geschriebener Wert: $00110100 = 52_{dez}$
2. Aus dem Timer gelesener Wert: $00011001 = 25_{dez}$

   $52 - \frac{213}{8} - 1 = 52 - 26 - 1 = 25$
3. Aus dem Timer gelesener Wert: $00000000 = 0_{dez}$

   $52 - \frac{415}{8} - 1 = 52 - 51 - 1 = 0$
4. Interrupt wurde bei Ø 2-Takt 416 ausgelöst

   Aus dem Timer gelesener Wert: 11111111
5. Aus dem Timer gelesener Wert: 10101100

   Zweierkomplement: $01010100 = 84_{dez}$

   $84 + (52 \times 8) = 500_{dez}$

Soll der Timer nach dem Interrupt gelesen werden, muß A3 log. Ø sein, um damit IRQ zurückzusetzen. Dadurch werden weitere Interrupts vermieden bevor neue Daten in den Timer eingeschrieben sind.

Elemente des Timers

Bild H.2

APPENDIX I

KIM-1 PROGRAM LISTINGS

```
CARD # LOC     CODE        CARD
     3                    ;                666666   555555   333333   000000
     4                    ;                6        5             3   0     0
     5                    ;                6        5             3   0     0
     6                    ;                666666   555555   333333   0     0
     7                    ;                6     6        5        3   0     0
     8                    ;                6     6        5        3   0     0
     9                    ;                666666   555555   333333   000000
    10                    ;
    11                    ;
    12                    ;
    13                    ;                         000000   000000   333333
    14                    ;                         0     0  0     0        3
    15                    ;                ------   0     0  0     0        3
    16                    ;                ------   0     0  0     0   333333
    17                    ;                ------   0     0  0     0        3
    18                    ;                         0     0  0     0        3
    19                    ;                         000000   000000   333333
    20                    ;
    21                    ;
    22                    ;
    23                    ;
    24                    ;
    25                    ;       COPYRIGHT
    26                    ;       MOS TECHNOLOGY, INC
    27                    ;       DATE OCT 18 1975 REV D
    28                    ;
    29                    ;
    30                    ;
    31                    ;       6530-003 IS AN AUDIO CASSETT TAPE
    32                    ;       RECORDER ENTENSION OF THE BASIC
    33                    ;       KIM MONITOR
    34                    ;
    35                    ;       IT FEATURES TWO BASIC ROUTINES
    36                    ;       LOADT-LOAD MEM FROM AUDIO TAPE
    37                    ;       DUMPT-STOR MEM ONTO AUDIO TAPE
    38                    ;
    39                    ;       LOADT
    40                    ;         ID=00      IGNORE ID
    41                    ;         ID=FF      IGN. ID USE SA FOR START ADDR
    42                    ;         ID=01-FE   IGN.ID USE ADDR ON TAPE
    43                    ;
    44                    ;       DUMPT
    45                    ;         ID=00      SHOULD NOT BE USED
    46                    ;         ID=FF      SHOULD NOT BE USED
    47                    ;         ID=01-FE   NORMAL ID RANGE
    48                    ;         SAL        LSB STARTING ADDRESS
    49                    ;         SAH        MSB
    50                    ;         EAL        LSB ENDING ADDRESS
    51                    ;         EAH        MSB
    52                    ;
```

```
CARD #  LOC   CODE      CARD
    54                ;
    55                ;       EQUATES
    56                ;       SET UP FOR 6530-002 I/O
    57                ;
    58                SAD     =$1740          6530 A DATA
    59                PADD    =$1741          6530 A DATA DIRECTION
    60                SBD     =$1742          6530 B DATA
    61                PBDD    =$1743          6530 B DATA DIRECTION
    62                CLK1T   =$1744          DIV BY 1 TIME
    63                CLK8T   =$1745          DIV BY 8 TIME
    64                CLK64T  =$1746          DIV BY 64 TIME
    65                CLKKT   =$1747          DIV BY 1024 TIME
    66                CLKRDI  =$1747          READ TIME OUT BIT
    67                CLKRDT  =$1746          READ TIME
    68                ;
    69  0000                  *=$00EF
    70                ;       MPU REG. SAVX AREA IN PAGE 0
    71                ;
    72  00EF          PCL     *=*+1 PROGRAM CNT LOW
    73  00F0          PCH     *=*+1 PROGRAM CNT HI
    74  00F1          PREG    *=*+1 CURRENT STATUS REG.
    75  00F2          SPUSER  *=*+1 CURRENT STACK POINT
    76  00F3          ACC     *=*+1 ACCUMULATOR
    77  00F4          YREG    *=*+1 Y INDEX
    78  00F5          XREG    *=*+1 X INDEX
    79                ;
    80                ;       KIM FIXED AREA IN PAGE 0
    81                ;
    82  00F6          CHKHI   *=*+1
    83  00F7          CHKSUM  *=*+1
    84  00F8          INL     *=*+1 INPUT BUFFER
    85  00F9          INH     *=*+1 INPUT BUFFER
    86  00FA          POINTL  *=*+1 LSB OF OPEN CELL
    87  00FB          POINTH  *=*+1 MSB OF OPEN CELL
    88  00FC          TEMP    *=*+1
    89  00FD          TMPX    *=*+1
    90  00FE          CHAR    *=*+1
    91  00FF          MODE    *=*+1
    92                ;
    93                ;       KIM FIXED AREA IN PAGE 23
    94                ;
    95  0100                  *=$17E7
    96  17E7          CHKL    *=*+1
    97  17E8          CHKH    *=*+1           CHKSUM
    98  17E9          SAVX    *=*+3
    99  17EC          VEB     *=*+6           VOLATILE EXECUTION BLOCK
   100  17F2          CNTL30  *=*+1           TTY DELAY
   101  17F3          CNTH30  *=*+1           TTY DELAY
   102  17F4          TIMH    *=*+1
   103  17F5          SAL     *=*+1           LOW STARTING ADDRESS
   104  17F6          SAH     *=*+1           HI STARTING ADDRESS
   105  17F7          EAL     *=*+1           LOW ENDING ADDRESS
```

```
CARD #  LOC     CODE        CARD
  106   17F8              EAH     *=*+1             HI ENDING ADDRESS
  107   17F9              ID      *=*+1
  108                     ;
  109                     ;       INTERRUPT VECTORS
  110                     ;
  111   17FA              NMIV    *=*+2             STOP VECTOR (STOP=1C00)
  112   17FC              RSTV    *=*+2             RST VECTOR
  113   17FE              IRQV    *=*+2             IRQ VECTOR (BRK= 1C00)
  114                     ;
```

```
CARD #  LOC     CODE        CARD
  116   1800                      *=$1800
  117                     ;
  118                     ;       INIT VOLATILE EXECUTION BLOCK
  119                     ;       DUMP MEM TO TAPE
  120                     ;
  121   1800  A9 AD       DUMPT   LDA   #$AD        LOAD ABSOLUTE INST
  122   1802  8D EC 17            STA   VEB
  123   1805  20 32 19            JSR INTVEB
  124                     ;
  125   1808  A9 27               LDA   #$27        TURN OFF DATAIN PB5
  126   180A  8D 42 17            STA   SBD
  127   180D  A9 BF               LDA   #$BF        CONVERT PB7 TO OUTPUT
  128   180F  8D 43 17            STA   PBDD
  129                     ;
  130   1812  A2 64               LDX   #$64        100 CHARS
  131   1814  A9 16       DUMPT1  LDA   #$16        SYN CHAR'S
  132   1816  20 7A 19            JSR   OUTCHT
  133   1819  CA                  DEX
  134   181A  D0 F8               BNE   DUMPT1
  135                     ;
  136                     ;
  137   181C  A9 2A               LDA   #'*         START CHAR
  138   181E  20 7A 19            JSR   OUTCHT
  139                     ;
  140   1821  AD F9 17            LDA   ID          OUTPUT ID
  141   1824  20 61 19            JSR   OUTBT
  142                     ;
  143   1827  AD F5 17            LDA   SAL         OUTPUT STARTING
  144   182A  20 5E 19            JSR   OUTBTC      ADDRESS
  145   182D  AD F6 17            LDA   SAH
  146   1830  20 5E 19            JSR   OUTBTC
  147                     ;
  148   1833  AD ED 17  DUMPT2    LDA   VEB+1       CHECK FOR LAST
  149   1836  CD F7 17            CMP   EAL         DATA BYTE
  150   1839  AD EE 17            LDA   VEB+2
  151   183C  ED F8 17            SBC   EAH
  152   183F  90 24               BCC   DUMPT4
  153                     ;
  154   1841  A9 2F               LDA   #'/         OUTPUT END OF DATA CHR
  155   1843  20 7A 19            JSR   OUTCHT
  156   1846  AD E7 17            LDA   CHKL        LAST BYTE HAS BEEN
  157   1849  20 61 19            JSR   OUTBT       OUT PUT   NOW OUTPUT
  158   184C  AD E8 17            LDA   CHKH        CHKSUM
  159   184F  20 61 19            JSR   OUTBT
  160                     ;
  161                     ;
  162   1852  A2 02               LDX   #$02        2 CHAR'S
  163   1854  A9 04       DUMPT3  LDA   #$04        EOT CHAR
  164   1856  20 7A 19            JSR   OUTCHT
  165   1859  CA                  DEX
  166   185A  D0 F8               BNE   DUMPT3
  167                     ;
```

```
CARD #  LOC     CODE        CARD
  168   185C  A9 00           LDA   #$00        DISPLAY 0000
  169   185E  85 FA           STA   POINTL      FOR NORMAL EXIT
  170   1860  85 FB           STA   POINTH
  171   1862  4C 4F 1C        JMP   START
  172                 ;
  173   1865  20 EC 17  DUMPT4 JSR   VEB         DATA BYTE OUTPUT
  174   1868  20 5E 19        JSR   OUTBTC
  175                 ;
  176   186B  20 EA 19        JSR   INCVEB
  177   186E  4C 33 18        JMP   DUMPT2
  178                 ;
  179                 ;       LOAD MEMORY FROM TAPE
  180                 ;
  181                 ;
  182   1871  0F 19     TAB    .WORD LOAD12
  183   1873  A9 8D     LOADT  LDA   #$8D        INIT VOLATILE EXECUTION
  184   1875  8D EC 17        STA   VEB         BLOCK WITH STA ABS.
  185   1878  20 32 19        JSR   INTVEB
  186                 ;
  187   187B  A9 4C           LDA   #$4C        JUMP TYPE RTRN
  188   187D  8D EF 17        STA   VEB+3
  189   1880  AD 71 18        LDA   TAB
  190   1883  8D F0 17        STA   VEB+4
  191   1886  AD 72 18        LDA   TAB+1
  192   1889  8D F1 17        STA   VEB+5
  193                 ;
  194   188C  A9 07           LDA   #$07        RESET PB5=0 (DATA IN)
  195   188E  8D 42 17        STA   SBD
  196                 ;
  197   1891  A9 FF     SYNC   LDA   #$FF        CLEAR SAVX FOR SYNC AREA
  198   1893  8D E9 17        STA   SAVX
  199                 ;
  200   1896  20 41 1A  SYNC1  JSR   RDBIT       GET A BIT
  201   1899  4E E9 17        LSR   SAVX        SHIFT BIT INTO CHAR
  202   189C  0D E9 17        ORA   SAVX
  203   189F  8D E9 17        STA   SAVX
  204   18A2  AD E9 17        LDA   SAVX        GET NEW CHAR
  205   18A5  C9 16           CMP   #$16        SYN CHAR
  206   18A7  D0 ED           BNE   SYNC1
  207                 ;
  208   18A9  A2 0A           LDX   #$0A        TEST FOR 10 SYN CHARS
  209   18AB  20 24 1A  SYNC2  JSR   RDCHT
  210   18AE  C9 16           CMP   #$16
  211   18B0  D0 DF           BNE   SYNC        IF NOT 10 CHAR  RE-SYNC
  212   18B2  CA              DEX
  213   18B3  D0 F6           BNE   SYNC2
  214                 ;
  215                 ;
  216   18B5  20 24 1A  LOADT4 JSR   RDCHT       LOOK FOR START OF
  217   18B8  C9 2A           CMP   #'*         DATA CHAR
  218   18BA  F0 06           BEQ   LOAD11
  219   18BC  C9 16           CMP   #$16        IF NOT *  SHOULD BE SYN
```

```
CARD #  LOC     CODE        CARD
  220   18BE  D0 D1            BNE   SYNC
  221   18C0  F0 F3            BEQ   LOADT4
  222                    ;
  223   18C2  20 F3 19  LOAD11 JSR   RDBYT     READ ID FROM TAPE
  224   18C5  CD F9 17         CMP   ID        COMPARE WITH REQUESTED ID
  225   18C8  F0 0D            BEQ   LOADT5
  226   18CA  AD F9 17         LDA   ID        DEFAULT 00 READ RECORD
  227   18CD  C9 00            CMP   #$00      ANYWAY
  228   18CF  F0 06            BEQ   LOADT5
  229   18D1  C9 FF            CMP   #$FF      DEFAULT FF IGNOR SA ON
  230   18D3  F0 17            BEQ   LOADT6    TAPE
  231   18D5  D0 9C            BNE   LOADT
  232                    ;
  233   18D7  20 F3 19  LOADT5 JSR   RDBYT     GET SA FROM TAPE
  234   18DA  20 4C 19         JSR   CHKT
  235   18DD  8D ED 17         STA   VEB+1     SAVX IN VEB+1,2
  236   18E0  20 F3 19         JSR   RDBYT
  237   18E3  20 4C 19         JSR   CHKT
  238   18E6  8D EE 17         STA   VEB+2
  239   18E9  4C F8 18         JMP   LOADT7
  240                    ;
  241   18EC  20 F3 19  LOADT6 JSR   RDBYT     GET SA BUT IGNORE
  242   18EF  20 4C 19         JSR   CHKT
  243   18F2  20 F3 19         JSR   RDBYT
  244   18F5  20 4C 19         JSR   CHKT
  245                    ;
  246                    ;
  247   18F8  A2 02     LOADT7 LDX   #$02      GET 2 CHARS
  248   18FA  20 24 1A  LOAD13 JSR   RDCHT     GET CHAR(X)
  249   18FD  C9 2F            CMP   #'/       LOOK FOR LAST CHAR
  250   18FF  F0 14            BEQ   LOADT8
  251   1901  20 00 1A         JSR   PACKT     CONVERT TO HEX
  252   1904  D0 23            BNE   LOADT9    Y=1 NON-HEX CHAR
  253   1906  CA               DEX
  254   1907  D0 F1            BNE   LOAD13
  255                    ;
  256   1909  20 4C 19         JSR   CHKT      COMPUTE CHECKSUM
  257   190C  4C EC 17         JMP   VEB       SAVX DATA IN MEMORY
  258   190F  20 EA 19  LOAD12 JSR   INCVEB    INCREMENT DATA POINTER
  259   1912  4C F8 18         JMP   LOADT7
  260                    ;
  261   1915  20 F3 19  LOADT8 JSR   RDBYT     END OF DATA COMPARE CHKSUM
  262   1918  CD E7 17         CMP   CHKL
  263   191B  D0 0C            BNE   LOADT9
  264   191D  20 F3 19         JSR   RDBYT
  265   1920  CD E8 17         CMP   CHKH
  266   1923  D0 04            BNE   LOADT9
  267   1925  A9 00            LDA   #$00      NORMAL EXIT
  268   1927  F0 02            BEQ   LOAD10
  269                    ;
  270   1929  A9 FF     LOADT9 LDA   #$FF      ERROR EXIT
  271   192B  85 FA     LOAD10 STA   POINTL
```



```
CARD #  LOC     CODE        CARD
  272   192D  85 FB            STA   POINTH
  273   192F  4C 4F 1C         JMP   START
  274                    ;
```

```
CARD # LOC     CODE        CARD
  276                      ;
  277                      ;       SUBROUTINES FOLLOW
  278                      ;
  279                      ;       SUB TO MOVE SA TO VEB+1,2
  280                      ;
  281   1932  AD F5 17     INTVEB  LDA   SAL
  282   1935  8D ED 17             STA   VEB+1
  283   1938  AD F6 17             LDA   SAH
  284   193B  8D EE 17             STA   VEB+2
  285   193E  A9 60                LDA   #$60        RTS INST
  286   1940  8D EF 17             STA   VEB+3
  287   1943  A9 00                LDA   #$00        CLEAR CHKSUM AREA
  288   1945  8D E7 17             STA   CHKL
  289   1948  8D E8 17             STA   CHKH
  290   194B  60                   RTS
  291                      ;
  292                      ;       COMPUTE CHKSUM FOR TAPE LOAD
  293                      ;       RTN USES Y TO SAVX A
  294                      ;
  295   194C  A8           CHKT    TAY
  296   194D  18                   CLC
  297   194E  6D E7 17             ADC   CHKL
  298   1951  8D E7 17             STA   CHKL
  299   1954  AD E8 17             LDA   CHKH
  300   1957  69 00                ADC   #$00
  301   1959  8D E8 17             STA   CHKH
  302   195C  98                   TYA
  303   195D  60                   RTS
  304                      ;
  305                      ;       OUTPUT ONE BYTE USE Y
  306                      ;       TO SAVX BYTE
  307                      ;
  308   195E  20 4C 19     OUTBTC  JSR   CHKT        COMP CHKSUM
  309   1961  A8           OUTBT   TAY               SAVX DATA BYTE
  310   1962  4A                   LSR   A           SHIFT OFF LSD
  311   1963  4A                   LSR   A
  312   1964  4A                   LSR   A
  313   1965  4A                   LSR   A
  314   1966  20 6F 19             JSR   HEXOUT      OUT PUT MSD
  315   1969  98                   TYA
  316   196A  20 6F 19             JSR   HEXOUT      OUT PUT LSD
  317   196D  98                   TYA
  318   196E  60                   RTS
  319                      ;
  320                      ;       CONVERT LSD OF A TO ASCII
  321                      ;       AND OUTPUT TO TAPE
  322                      ;
  323   196F  29 0F        HEXOUT  AND   #$0F
  324   1971  C9 0A                CMP   #$0A
  325   1973  18                   CLC
  326   1974  30 02                BMI   HEX1
  327   1976  69 07                ADC   #$07
```

```
CARD #  LOC     CODE        CARD
  328   1978  69 30     HEX1   ADC   #$30
  329                    ;
  330                    ;      OUTPUT TO TAPE ONE ASCII
  331                    ;      CHAR  USE SUB'S ONE + ZRO
  332                    ;
  333   197A  8E E9 17   OUTCHT STX   SAVX
  334   197D  8C EA 17           STY   SAVX+1
  335   1980  A0 08              LDY   #$08        START BIT
  336   1982  20 9E 19   CHT1    JSR   ONE
  337   1985  4A                LSR   A           GET DATA BIT
  338   1986  B0 06              BCS   CHT2
  339   1988  20 9E 19           JSR   ONE         DATA BIT=1
  340   198B  4C 91 19           JMP   CHT3
  341   198E  20 C4 19   CHT2    JSR   ZRO         DATA BIT=0
  342   1991  20 C4 19   CHT3    JSR   ZRO
  343   1994  88                DEY
  344   1995  D0 EB              BNE   CHT1
  345   1997  AE E9 17           LDX   SAVX
  346   199A  AC EA 17           LDY   SAVX+1
  347   199D  60                RTS
  348                    ;
  349                    ;
  350                    ;      OUTPUT 1 TO TAPE
  351                    ;      9 PULSES 138 MICROSEC EACH
  352                    ;
  353   199E  A2 09       ONE    LDX   #$09
  354   19A0  48                PHA               SAVX A
  355   19A1  2C 47 17   ONE1    BIT   CLKRDI      WAIT FOR TIME OUT
  356   19A4  10 FB              BPL   ONE1
  357   19A6  A9 7E              LDA   #126
  358   19A8  8D 44 17           STA   CLK1T
  359   19AB  A9 A7              LDA   #$A7
  360   19AD  8D 42 17           STA   SBD         SET  PB7=1
  361   19B0  2C 47 17   ONE2    BIT   CLKRDI
  362   19B3  10 FB              BPL   ONE2
  363   19B5  A9 7E              LDA   #126
  364   19B7  8D 44 17           STA   CLK1T
  365   19BA  A9 27              LDA   #$27
  366   19BC  8D 42 17           STA   SBD         RESET PB7=0
  367   19BF  CA                 DEX
  368   19C0  D0 DF              BNE   ONE1
  369   19C2  68                 PLA
  370   19C3  60                 RTS
  371                    ;
  372                    ;
  373                    ;      OUTPUT 0 TO TAPE
  374                    ;      6 PULSES 207 MICROSEC EACH
  375                    ;
  376   19C4  A2 06       ZRO    LDX   #$06
  377   19C6  48                 PHA               SAVX A
  378   19C7  2C 47 17   ZRO1    BIT   CLKRDI
  379   19CA  10 FB              BPL   ZRO1
```

```
CARD #  LOC    CODE        CARD
   380  19CC   A9 C3           LDA   #195
   381  19CE   8D 44 17         STA   CLK1T
   382  19D1   A9 A7           LDA   #$A7
   383  19D3   8D 42 17         STA   SBD          SET PB7=1
   384  19D6   2C 47 17  ZRO2   BIT   CLKRDI
   385  19D9   10 FB            BPL   ZRO2
   386  19DB   A9 C3           LDA   #195
   387  19DD   8D 44 17         STA   CLK1T
   388  19E0   A9 27           LDA   #$27
   389  19E2   8D 42 17         STA   SBD          RESET PB7=0
   390  19E5   CA               DEX
   391  19E6   D0 DF            BNE   ZRO1
   392  19E8   68               PLA                RESTORE A
   393  19E9   60               RTS
   394                     ;
   395                     ;       SUB TO INC VEB+1,2
   396                     ;
   397  19EA   EE ED 17  INCVEB INC   VEB+1
   398  19ED   D0 03            BNE   INCVE1
   399  19EF   EE EE 17         INC   VEB+2
   400  19F2   60        INCVE1 RTS
   401                     ;
   402                     ;       SUB TO READ BYTE FROM TAPE
   403                     ;
   404  19F3   20 24 1A  RDBYT  JSR   RDCHT
   405  19F6   20 00 1A          JSR   PACKT
   406  19F9   20 24 1A  RDBYT2 JSR   RDCHT
   407  19FC   20 00 1A          JSR   PACKT
   408  19FF   60                RTS
   409                     ;
   410                     ;       PACK A=ASCII INTO SAVX
   411                     ;       AS HEX DATA
   412                     ;
   413  1A00   C9 30      PACKT  CMP   #$30
   414  1A02   30 1E             BMI   PACKT3
   415  1A04   C9 47             CMP   #$47
   416  1A06   10 1A             BPL   PACKT3
   417  1A08   C9 40             CMP   #$40
   418  1A0A   30 03             BMI   PACKT1
   419  1A0C   18                CLC
   420  1A0D   69 09             ADC   #$09
   421  1A0F   2A        PACKT1 ROL   A
   422  1A10   2A               ROL   A
   423  1A11   2A               ROL   A
   424  1A12   2A               ROL   A
   425  1A13   A0 04            LDY   #$04
   426  1A15   2A        PACKT2 ROL   A
   427  1A16   2E E9 17          ROL   SAVX
   428  1A19   88               DEY
   429  1A1A   D0 F9            BNE   PACKT2
   430  1A1C   AD E9 17          LDA   SAVX
   431  1A1F   A0 00            LDY   #$00         Y=0 VALID HEX CHAR
```

```
CARD #  LOC     CODE        CARD
  432   1A21   60             RTS              Y=0 VALID HEX
  433   1A22   C8          PACKT3 INY          Y=1 NOT HEX
  434   1A23   60             RTS
  435                       ;
  436                       ;       GET 1 CHAR FROM TAPE AND RETURN
  437                       ;       WITH CHAR IN A  USE SAVX+1 TO ASM CHAR
  438                       ;
  439   1A24   8E EB 17    RDCHT   STX   SAVX+2
  440   1A27   A2 08               LDX   #$08        READ 8 BITS
  441   1A29   20 41 1A    RDCHT1  JSR   RDBIT       GET NEXT DATA BIT
  442   1A2C   4E EA 17            LSR   SAVX+1      RIGHT SHIFT CHAR
  443   1A2F   0D EA 17            ORA   SAVX+1      OR IN SIGN BIT
  444   1A32   8D EA 17            STA   SAVX+1      REPLACE CHAR
  445   1A35   CA                  DEX
  446   1A36   D0 F1               BNE   RDCHT1
  447                       ;
  448   1A38   AD EA 17            LDA   SAVX+1      MOVE CHAR INTO A
  449   1A3B   2A                  ROL   A           SHIFT OFF PARITY
  450   1A3C   4A                  LSR   A
  451   1A3D   AE EB 17            LDX   SAVX+2
  452   1A40   60                  RTS
  453                       ;
  454                       ;       THIS SUB GETS ONE BIT FROM
  455                       ;       TAPE AND RETURNS IT IN SIGN OF A
  456                       ;
  457   1A41   2C 42 17    RDBIT   BIT   SBD         WAIT FOR END OF START BIT
  458   1A44   10 FB               BPL   RDBIT
  459   1A46   AD 46 17            LDA   CLKRDT      GET START BIT TIME
  460   1A49   A0 FF               LDY   #$FF        A=256-T1
  461   1A4B   8C 46 17            STY   CLK64T      SET UP TIMER
  462                       ;
  463   1A4E   A0 14               LDY   #$14
  464   1A50   88          RDBIT3  DEY               DELAY 100 MICROSEC
  465   1A51   D0 FD               BNE   RDBIT3
  466                       ;
  467   1A53   2C 42 17    RDBIT2  BIT   SBD
  468   1A56   30 FB               BMI   RDBIT2      WAIT FOR NEXT START BIT
  469                       ;
  470   1A58   38                  SEC
  471   1A59   ED 46 17            SBC   CLKRDT      (256-T1)-(256-T2)=T2-T1
  472   1A5C   A0 FF               LDY   #$FF
  473   1A5E   8C 46 17            STY   CLK64T      SET UP TIMER FOR NEXT BIT
  474                       ;
  475   1A61   A0 07               LDY   #$07
  476   1A63   88          RDBIT4  DEY               DELAY 50 MICROSEC
  477   1A64   D0 FD               BNE   RDBIT4
  478                       ;
  479   1A66   49 FF               EOR   #$FF        COMPLEMENT SIGN OF A
  480   1A68   29 80               AND   #$80        MASK ALL EXCEPT SIGN
  481   1A6A   60                  RTS
```

```
CARD # LOC      CODE        CARD
  483                       ;
  484                       ;       DIAGNOSTICS
  485                       ;          MEMORY
  486                       ;          PLLCAL
  487                       ;
  488                       ;
  489                       ;
  490                       ;       PLLCAL OUTPUT 166 MICROSEC
  491                       ;       PULSE STRING
  492                       ;
  493  1A6B  A9 27      PLLCAL LDA   #$27
  494  1A6D  8D 42 17          STA   SBD          TURN OFF DATIN PB5=1
  495  1A70  A9 BF             LDA   #$BF         CONVERT PB7 TO OUTPUT
  496  1A72  8D 43 17          STA   PBDD
  497                       ;
  498  1A75  2C 47 17  PLL1    BIT   CLKRDI
  499  1A78  10 FB             BPL   PLL1
  500  1A7A  A9 9A             LDA   #154         WAIT 166 MICRO SEC
  501  1A7C  8D 44 17          STA   CLK1T
  502  1A7F  A9 A7             LDA   #$A7         OUTPUT PB7=1
  503  1A81  8D 42 17          STA   SBD
  504                       ;
  505  1A84  2C 47 17  PLL2    BIT   CLKRDI
  506  1A87  10 FB             BPL   PLL2
  507  1A89  A9 9A             LDA   #154
  508  1A8B  8D 44 17          STA   CLK1T
  509  1A8E  A9 27             LDA   #$27         PB7=0
  510  1A90  8D 42 17          STA   SBD
  511  1A93  4C 75 1A          JMP   PLL1
  512                       ;
  513                       ;
  514                       ;       INTERRUPTS PAGE 27
  515                       ;
  516  1A96                        *=*+$0164   RESERVED FOR TEST
  517  1BFA  6B 1A      NMIP27 .WORD PLLCAL
  518  1BFC  6B 1A      RSTP27 .WORD PLLCAL
  519  1BFE  6B 1A      IRQP27 .WORD PLLCAL
  520                       ;
```

```
CARD # LOC      CODE        CARD
  522                     ;
  523                     ;
  524                     ;
  525                     ;
  526                     ;                 666666    555555    333333    000000
  527                     ;                 6         5              3    0     0
  528                     ;                 6         5              3    0     0
  529                     ;                 666666    555555    333333    0     0
  530                     ;                 6     6         5         3    0     0
  531                     ;                 6     6         5         3    0     0
  532                     ;                 666666    555555    333333    000000
  533                     ;
  534                     ;
  535                     ;
  536                     ;                           000000    000000    222222
  537                     ;                           0     0   0     0         2
  538                     ;                 ------    0     0   0     0         2
  539                     ;                 ------    0     0   0     0   222222
  540                     ;                 ------    0     0   0     0   2
  541                     ;                           0     0   0     0   2
  542                     ;                           000000    000000    222222
  543                     ;
```

```
CARD # LOC   CODE      CARD
  545                ;
  546                ;
  547                ;
  548                ;             COPYRIGHT
  549                ;             MOS TECHNOLOGY INC.
  550                ;             DATE  OCT 13 1975    REV E
  551                ;
  552                ;       KIM   :TTY INTERFACE
  553                ;             :KEYBOARD INTERFACE
  554                ;             :7 SEG 6 DIGIT DISPLAY
  555                ;
  556                ;
  557                ;       TTY CMDS:
  558                ;             G   GOEXEC
  559                ;             CR  OPEN NEXT CELL
  560                ;             LF  OPEN PREV. CELL
  561                ;             .   MODIFY OPEN CELL
  562                ;             SP  OPEN NEW CELL
  563                ;             L   LOAD (OBJECT FORMAT)
  564                ;             Q   DUMP  FROM OPEN CELL ADDR TO HI LIMIT
  565                ;             RO  RUB OUT  - RETURN TO START (KIM)
  566                ;                 ((ALL ILLEGAL CHAR ARE IGNORED))
  567                ;
  568                ;       KEYBOARD CMDS:
  569                ;           ADDR  SETS MODE TO MODIFY CELL ADDRESS
  570                ;           DATA  SETS MODE TO MODIFY DATA IN OPEN CELL
  571                ;           STEP  INCREMENTS TO NEXT CELL
  572                ;           RST   SYSTEM RESET
  573                ;           RUN   GOEXEC
  574                ;           STOP  $1C00 CAN BE LOADED INTO NMIV TO
  575                ;                 USE STOP FEATURE
  576                ;           PC    DISPLAY PC
  577                ;
  578                ;           CLOCK IS NOT DISABLED IN SIGMA 1
  579                ;
  580                ;
  581                ;
  582                ;
```

```
CARD #  LOC     CODE        CARD
  584   1C00                    *=$1C00
  585                   ;
  586                   ;
  587   1C00  85 F3     SAVE    STA   ACC        KIM ENTRY VIA STOP (NMI)
  588   1C02  68                PLA               OR BRK (IRQ)
  589   1C03  85 F1             STA   PREG
  590   1C05  68        SAVE1   PLA               KIM ENTRY VIA JSR  (A LOST)
  591   1C06  85 EF             STA   PCL
  592   1C08  85 FA             STA   POINTL
  593   1C0A  68                PLA
  594   1C0B  85 F0             STA   PCH
  595   1C0D  85 FB             STA   POINTH
  596   1C0F  84 F4     SAVE2   STY   YREG
  597   1C11  86 F5             STX   XREG
  598   1C13  BA                TSX
  599   1C14  86 F2             STX   SPUSER
  600   1C16  20 88 1E          JSR   INITS
  601   1C19  4C 4F 1C          JMP   START
  602                   ;
  603   1C1C  6C FA 17  NMIT    JMP   (NMIV)      NON-MASKABLE INTERRUPT TRAP
  604   1C1F  6C FE 17  IRQT    JMP   (IRQV)      INTERRUPT TRAP
  605                   ;
  606   1C22  A2 FF     RST     LDX   #$FF        KIM ENTRY VIA RST
  607   1C24  9A                TXS
  608   1C25  86 F2             STX   SPUSER
  609   1C27  20 88 1E          JSR   INITS
  610                   ;
  611                   ;
  612   1C2A  A9 FF     DETCPS  LDA   #$FF        COUNT START BIT
  613   1C2C  8D F3 17          STA   CNTH30      ZERO CNTH30
  614   1C2F  A9 01             LDA   #$01        MASK HI ORDER BITS
  615   1C31  2C 40 17  DET1    BIT   SAD         TEST
  616   1C34  D0 19             BNE   START       KEYBD SSW TEST
  617   1C36  30 F9             BMI   DET1        START BIT TEST
  618   1C38  A9 FC             LDA   #$FC
  619   1C3A  18        DET3    CLC               THIS LOOP COUNTS
  620   1C3B  69 01             ADC   #$01        THE START BIT TIME
  621   1C3D  90 03             BCC   DET2
  622   1C3F  EE F3 17          INC   CNTH30
  623   1C42  AC 40 17  DET2    LDY   SAD         CHECK FOR END OF START BIT
  624   1C45  10 F3             BPL   DET3
  625   1C47  8D F2 17          STA   CNTL30
  626   1C4A  A2 09             LDX   #$08
  627   1C4C  20 6A 1E          JSR   GET5        GET REST OF THE CHAR
  628                   ;                         TEST CHAR HERE
  629                   ;
  630                   ;
  631                   ;
  632                   ;
  633                   ;
  634                   ;       MAKE TTY/KB SELECTION
  635                   ;
```

```
CARD #  LOC     CODE        CARD
  636   1C4F  20 8C 1E  START  JSR   INIT1
  637   1C52  A9 01             LDA   #$01
  638   1C54  2C 40 17          BIT   SAD
  639   1C57  D0 1E             BNE   TTYKB
  640   1C59  20 2F 1E          JSR   CRLF       PRT CR LF
  641   1C5C  A2 0A             LDX   #$0A       TYPE OUT KIM
  642   1C5E  20 31 1E          JSR   PRTST
  643   1C61  4C AF 1D          JMP   SHOW1
  644                     ;
  645   1C64  A9 00      CLEAR  LDA   #$00
  646   1C66  85 F8             STA   INL        CLEAR INPUT BUFFER
  647   1C68  85 F9             STA   INH
  648   1C6A  20 5A 1E  READ   JSR   GETCH      GET CHAR
  649   1C6D  C9 01             CMP   #$01
  650   1C6F  F0 06             BEQ   TTYKB
  651   1C71  20 AC 1F          JSR   PACK
  652   1C74  4C DB 1D          JMP   SCAN
  653                     ;
  654                     ;       MAIN ROTINE FOR KEY BOARD
  655                     ;       AND DISPLAY
  656                     ;
  657   1C77  20 19 1F  TTYKB  JSR   SCAND      IF A=0 NO KEY
  658   1C7A  D0 D3             BNE   START
  659   1C7C  A9 01     TTYKB1 LDA   #$01
  660   1C7E  2C 40 17          BIT   SAD
  661   1C81  F0 CC             BEQ   START
  662   1C83  20 19 1F          JSR   SCAND
  663   1C86  F0 F4             BEQ   TTYKB1
  664   1C88  20 19 1F          JSR   SCAND
  665   1C8B  F0 EF             BEQ   TTYKB1
  666                     ;
  667   1C8D  20 6A 1F  GETK   JSR   GETKEY
  668   1C90  C9 15             CMP   #$15
  669   1C92  10 BB             BPL   START
  670   1C94  C9 14             CMP   #$14
  671   1C96  F0 44             BEQ   PCCMD      DISPLAY PC
  672   1C98  C9 10             CMP   #$10       ADDR MODE=1
  673   1C9A  F0 2C             BEQ   ADDRM
  674   1C9C  C9 11             CMP   #$11       DATA MODE=1
  675   1C9E  F0 2C             BEQ   DATAM
  676   1CA0  C9 12             CMP   #$12       STEP
  677   1CA2  F0 2F             BEQ   STEP
  678   1CA4  C9 13             CMP   #$13       RUN
  679   1CA6  F0 31             BEQ   GOV
  680   1CA8  0A        DATA   ASL   A          SHIFT CHAR INTO HIGH
  681   1CA9  0A               ASL   A          ORDER NIBBLE
  682   1CAA  0A               ASL   A
  683   1CAB  0A               ASL   A
  684   1CAC  85 FC            STA   TEMP       STORE IN TEMP
  685   1CAE  A2 04            LDX   #$04
  686   1CB0  A4 FF     DATA1  LDY   MODE       TEST MODE 1=ADDR
  687   1CB2  D0 0A            BNE   ADDR       MODE=0 DATA
```

```
CARD #  LOC     CODE        CARD
  688   1CB4  B1 FA            LDA   (POINTL),Y GET DATA
  689   1CB6  06 FC            ASL   TEMP       SHIFT CHAR
  690   1CB8  2A               ROL   A          SHIFT DATA
  691   1CB9  91 FA            STA   (POINTL),Y STORE OUT DATA
  692   1CBB  4C C3 1C         JMP   DATA2
  693                       ;
  694   1CBE  0A          ADDR  ASL   A          SHIFT CHAR
  695   1CBF  26 FA            ROL   POINTL     SHIFT ADDR
  696   1CC1  26 FB            ROL   POINTH     SHIFT ADDR HI
  697   1CC3  CA          DATA2 DEX
  698   1CC4  D0 EA            BNE   DATA1      DO 4 TIMES
  699   1CC6  F0 08            BEQ   DATAM2     EXIT HERE
  700                       ;
  701   1CC8  A9 01       ADDRM LDA   #$01
  702   1CCA  D0 02            BNE   DATAM1
  703                       ;
  704   1CCC  A9 00       DATAM LDA   #$00
  705   1CCE  85 FF       DATAM1 STA  MODE
  706   1CD0  4C 4F 1C    DATAM2 JMP  START
  707                       ;
  708   1CD3  20 63 1F    STEP  JSR   INCPT
  709   1CD6  4C 4F 1C         JMP   START
  710                       ;
  711   1CD9  4C C8 1D    GOV   JMP   GOEXEC
  712                       ;
  713                       ;
  714                       ;       DISPLAY PC BY MOVING
  715                       ;       PC TO POINT
  716                       ;
  717   1CDC  A5 EF       PCCMD LDA   PCL
  718   1CDE  85 FA            STA   POINTL
  719   1CE0  A5 F0            LDA   PCH
  720   1CE2  85 FB            STA   POINTH
  721   1CE4  4C 4F 1C         JMP   START
  722                       ;
  723                       ;       LOAD PAPER TAPE FROM TTY
  724                       ;
  725   1CE7  20 5A 1E    LOAD  JSR   GETCH      LOOK FOR FIRST CHAR
  726   1CEA  C9 3B            CMP   #$3B       SMICOLON
  727   1CEC  D0 F9            BNE   LOAD
  728   1CEE  A9 00       LOADS LDA   #$00
  729   1CF0  85 F7            STA   CHKSUM
  730   1CF2  85 F6            STA   CHKHI
  731                       ;
  732   1CF4  20 9D 1F         JSR   GETBYT     GET BYTE CNT
  733   1CF7  AA               TAX              SAVE IN X INDEX
  734   1CF8  20 91 1F         JSR   CHK        COMPUTE CHKSUM
  735                       ;
  736   1CFB  20 9D 1F         JSR   GETBYT     GET ADDRESS HI
  737   1CFE  85 FB            STA   POINTH
  738   1D00  20 91 1F         JSR   CHK
  739   1D03  20 9D 1F         JSR   GETBYT     GET ADDRESS LO
```

```
CARD #  LOC     CODE        CARD
  740   1D06  95 FA           STA   POINTL
  741   1D08  20 91 1F        JSR   CHK
  742                      ;
  743   1D0B  8A             TXA                 IF CNT=0 DONT
  744   1D0C  F0 0F           BEQ   LOAD3        GET ANY DATA
  745                      ;
  746   1D0E  20 9D 1F  LOAD2  JSR   GETBYT       GET DATA
  747   1D11  91 FA           STA   (POINTL),Y    STORE DATA
  748   1D13  20 91 1F        JSR   CHK
  749   1D16  20 63 1F        JSR   INCPT         NEXT ADDRESS
  750   1D19  CA              DEX
  751   1D1A  D0 F2           BNE   LOAD2
  752   1D1C  E8              INX                 X=1  DATA RECORD
  753                      ;                     X=0  LAST RECORD
  754   1D1D  20 9D 1F  LOAD3  JSR   GETBYT       COMPARE CHKSUM
  755   1D20  C5 F6           CMP   CHKHI
  756   1D22  D0 17           BNE   LOADE1
  757   1D24  20 9D 1F        JSR   GETBYT
  758   1D27  C5 F7           CMP   CHKSUM
  759   1D29  D0 13           BNE   LOADER
  760                      ;
  761   1D2B  8A             TXA                 X=0  LAST RECORD
  762   1D2C  D0 B9           BNE   LOAD
  763                      ;
  764   1D2E  A2 0C     LOAD7  LDX   #$0C         X-OFF KIM
  765   1D30  A9 27     LOAD8  LDA   #$27
  766   1D32  8D 42 17         STA   SBD          DISABLE DATA IN
  767   1D35  20 31 1E         JSR   PRTST
  768   1D38  4C 4F 1C         JMP   START
  769                      ;
  770   1D3B  20 9D 1F  LOADE1 JSR   GETBYT       DUMMY
  771   1D3E  A2 11     LOADER LDX   #$11         X-OFF ERR KIM
  772   1D40  D0 EE           BNE   LOAD8
  773                      ;
  774                      ;       DUMP TO TTY
  775                      ;       FROM OPEN CELL ADDRESS
  776                      ;       TO LIMHL,LIMHH
  777                      ;
  778   1D42  A9 00     DUMP   LDA   #$00
  779   1D44  85 F8            STA   INL
  780   1D46  85 F9            STA   INH          CLEAR RECORD COUNT
  781   1D48  A9 00     DUMP0  LDA   #$00
  782   1D4A  85 F6            STA   CHKHI        CLEAR CHKSUM
  783   1D4C  85 F7            STA   CHKSUM
  784                      ;
  785   1D4E  20 2F 1E  DUMP1  JSR   CRLF         PRINT CR LF
  786   1D51  A9 3B            LDA   #$3B          PRINT SMICOLON
  787   1D53  20 A0 1E         JSR   OUTCH
  788   1D56  A5 FA            LDA   POINTL       TEST POINT GT OR ET
  789   1D58  CD F7 17         CMP   EAL           HI LIMIT GO TO EXIT
  790   1D5B  A5 FB            LDA   POINTH
  791   1D5D  ED F8 17         SBC   EAH
```

```
CARD # LOC      CODE        CARD
  792  1D60  90 18           BCC   DUMP4
  793                      ;
  794  1D62  A9 00           LDA   #$00        PRINT LAST RECORD
  795  1D64  20 3B 1E        JSR   PRTBYT      0 BYTES
  796  1D67  20 CC 1F        JSR   OPEN
  797  1D6A  20 1E 1E        JSR   PRTPNT
  798                      ;
  799  1D6D  A5 F6           LDA   CHKHI       PRINT CHKSUM
  800  1D6F  20 3B 1E        JSR   PRTBYT      FOR LAST RECORD
  801  1D72  A5 F7           LDA   CHKSUM
  802  1D74  20 3B 1E        JSR   PRTBYT
  803  1D77  4C 64 1C        JMP   CLEAR
  804                      ;
  805  1D7A  A9 18      DUMP4 LDA   #$18        PRINT 24 BYTE CNT
  806  1D7C  AA              TAX                SAVE AS INDEX
  807  1D7D  20 3B 1E        JSR   PRTBYT
  808  1D80  20 91 1F        JSR   CHK
  809  1D83  20 1E 1E        JSR   PRTPNT
  810                      ;
  811  1D86  A0 00      DUMP2 LDY   #$00        PRINT 24 BYTES
  812  1D88  B1 FA           LDA   (POINTL),Y GET DATA
  813  1D8A  20 3B 1E        JSR   PRTBYT      PRINT DATA
  814  1D8D  20 91 1F        JSR   CHK         COMP CHKSUM
  815  1D90  20 63 1F        JSR   INCPT       INCREMENT POINT
  816  1D93  CA              DEX
  817  1D94  D0 F0           BNE   DUMP2
  818                      ;
  819  1D96  A5 F6           LDA   CHKHI       PRINT CHKSUM
  820  1D98  20 3B 1E        JSR   PRTBYT
  821  1D9B  A5 F7           LDA   CHKSUM
  822  1D9D  20 3B 1E        JSR   PRTBYT
  823  1DA0  E6 F8           INC   INL         INCREMENT RECORD CNT
  824  1DA2  D0 02           BNE   DUMP3
  825  1DA4  E6 F9           INC   INH
  826  1DA6  4C 48 1D  DUMP3 JMP   DUMP0
  827                      ;
  828  1DA9  20 CC 1F  SPACE JSR   OPEN         OPEN NEW CELL
  829  1DAC  20 2F 1E  SHOW  JSR   CRLF         PRINT CR LF
  830  1DAF  20 1E 1E  SHOW1 JSR   PRTPNT
  831  1DB2  20 9E 1E        JSR   OUTSP       PRT SPACE
  832  1DB5  A0 00           LDY   #$00        PRINT DATA SPECIFIED
  833  1DB7  B1 FA           LDA   (POINTL),Y BY POINT AD = LDA EXT
  834  1DB9  20 3B 1E        JSR   PRTBYT
  835  1DBC  20 9E 1E        JSR   OUTSP       PRT SPACE
  836  1DBF  4C 64 1C        JMP   CLEAR
  837                      ;
  838  1DC2  20 63 1F  RTRN  JSR   INCPT       OPEN NEXT CELL
  839  1DC5  4C AC 1D        JMP   SHOW
  840                      ;
  841  1DC8  A6 F2      GOEXEC LDX  SPUSER
  842  1DCA  9A              TXS
  843  1DCB  A5 FB           LDA   POINTH      PROGRAM RUNS FROM
```

```
ARD # LOC    CODE        CARD
 844  1DCD  48           PHA                 OPEN CELL ADDRESS
 845  1DCE  A5 FA        LDA   POINTL
 846  1DD0  48           PHA
 847  1DD1  A5 F1        LDA   PREG
 848  1DD3  48           PHA
 849  1DD4  A6 F5        LDX   XREG          RESTORE REGS
 850  1DD6  A4 F4        LDY   YREG
 851  1DD8  A5 F3        LDA   ACC
 852  1DDA  40           RTI
 853                ;
 854  1DDB  C9 20     SCAN  CMP   #$20        OPEN CELL
 855  1DDD  F0 CA           BEQ   SPACE
 856  1DDF  C9 7F           CMP   #$7F        RUB OUT (KIM)
 857  1DE1  F0 1B           BEQ   STV
 858  1DE3  C9 0D           CMP   #$0D        NEXT CELL
 859  1DE5  F0 DB           BEQ   RTRN
 860  1DE7  C9 0A           CMP   #$0A        PREV CELL
 861  1DE9  F0 1C           BEQ   FEED
 862  1DEB  C9 2E           CMP   #'.         MODIFY CELL
 863  1DED  F0 26           BEQ   MODIFY
 864  1DEF  C9 47           CMP   #'G         GO EXEC
 865  1DF1  F0 D5           BEQ   GOEXEC
 866  1DF3  C9 51           CMP   #'Q         DUMP FROM OPEN CELL TO HI LIMIT
 867  1DF5  F0 0A           BEQ   DUMPV
 868  1DF7  C9 4C           CMP   #'L         LOAD TAPE
 869  1DF9  F0 09           BEQ   LOADV
 870  1DFB  4C 6A 1C        JMP   READ        IGNORE ILLEGAL CHAR
 871                ;
 872  1DFE  4C 4F 1C  STV   JMP   START
 873  1E01  4C 42 1D  DUMPV JMP   DUMP
 874  1E04  4C E7 1C  LOADV JMP   LOAD
 875                ;
 876  1E07  38        FEED  SEC
 877  1E08  A5 FA           LDA   POINTL      DEC DOUBLE BYTE
 878  1E0A  E9 01           SBC   #$01        AT POINTL AND POINTH
 879  1E0C  85 FA           STA   POINTL
 880  1E0E  B0 02           BCS   FEED1
 881  1E10  C6 FB           DEC   POINTH
 882  1E12  4C AC 1D  FEED1 JMP   SHOW
 883                ;
 884  1E15  A0 00     MODIFY LDY   #$00        GET CONTENTS OF INPUT BUFF
 885  1E17  A5 F8           LDA   INL         INL AND STOR IN LOC
 886  1E19  91 FA           STA   (POINTL),Y  SPECIFIED BY POINT
 887  1E1B  4C C2 1D        JMP   RTRN
 888                ;
 889                ;       END OF MAIN LINE
```

```
ARD # LOC     CODE        CARD
 891                      ;       SUBROUTINES FOLLOW
 892                      ;
 893                      ;
 894                      ;
 895                      ;       SUB TO PRINT POINTL,POINTH
 896                      ;
 897  1E1E  A5 FB         PRTPNT LDA   POINTH
 898  1E20  20 3B 1E             JSR   PRTBYT
 899  1E23  20 91 1F             JSR   CHK
 900  1E26  A5 FA                LDA   POINTL
 901  1E28  20 3B 1E             JSR   PRTBYT
 902  1E2B  20 91 1F             JSR   CHK
 903  1E2E  60                   RTS
 904                      ;
 905                      ;       PRINT STRING OF ASCII CHAR FROM
 906                      ;       TOP+X TO TOP
 907                      ;
 908  1E2F  A2 07         CRLF   LDX   #$07
 909  1E31  BD D5 1F      PRTST  LDA   TOP,X
 910  1E34  20 A0 1E             JSR   OUTCH
 911  1E37  CA                   DEX
 912  1E38  10 F7                BPL   PRTST       STOP ON INDEX ZERO
 913  1E3A  60            PRT1   RTS
 914                      ;
 915                      ;       PRINT 1 HEX BYTE AS TWO ASCII CHAR'S
 916                      ;
 917  1E3B  85 FC         PRTBYT STA   TEMP
 918  1E3D  4A                   LSR   A           SHIFT CHAR RIGHT 4 BITS
 919  1E3E  4A                   LSR   A
 920  1E3F  4A                   LSR   A
 921  1E40  4A                   LSR   A
 922  1E41  20 4C 1E             JSR   HEXTA       CONVERT TO HEX AND PRINT
 923  1E44  A5 FC                LDA   TEMP        GET OTHER HALF
 924  1E46  20 4C 1E             JSR   HEXTA       CONVERT TO HEX AND PRINT
 925  1E49  A5 FC                LDA   TEMP        RESTORE BYTE IN A AND RETURN
 926  1E4B  60                   RTS
 927                      ;
 928  1E4C  29 0F         HEXTA  AND   #$0F        MASK HI 4 BITS
 929  1E4E  C9 0A                CMP   #$0A
 930  1E50  18                   CLC
 931  1E51  30 02                BMI   HEXTA1
 932  1E53  69 07                ADC   #$07        ALPHA HEX
 933  1E55  69 30         HEXTA1 ADC   #$30        DEC HEX
 934  1E57  4C A0 1E             JMP   OUTCH       PRINT CHAR
 935                      ;
 936                      ;       GET 1 CHAR FROM TTY
 937                      ;       RETURN FROM SUB WITH CHAR IN A
 938                      ;       X IS PRESERVED AND Y RETURNED = FF
 939                      ;
 940  1E5A  86 FD         GETCH  STX   TMPX        SAVE X REG
 941  1E5C  A2 08                LDX   #$08        SET UP 8 BIT CNT
 942  1E5E  A9 01                LDA   #$01
```

```
CARD # LOC     CODE        CARD
   943  1E60  2C 40 17  GET1   BIT   SAD
   944  1E63  D0 22           BNE   GET6
   945  1E65  30 F9           BMI   GET1       WAIT FOR START BIT
   946  1E67  20 D4 1E        JSR   DELAY      DELAY 1 BIT
   947  1E6A  20 EB 1E  GET5   JSR   DEHALF     DELAY 1/2 BIT TIME
   948  1E6D  AD 40 17  GET2   LDA   SAD        GET 8 BITS
   949  1E70  29 80           AND   #$80       MASK OFF LOW ORDER BITS
   950  1E72  46 FE           LSR   CHAR       SHIFT RIGHT CHARACTER
   951  1E74  05 FE           ORA   CHAR
   952  1E76  85 FE           STA   CHAR
   953  1E78  20 D4 1E        JSR   DELAY      DELAY 1   BIT TIME
   954  1E7B  CA              DEX
   955  1E7C  D0 EF           BNE   GET2       GET NEXT CHAR
   956  1E7E  20 EB 1E        JSR   DEHALF     EXIT THIS RTN
   957                 ;
   958  1E81  A6 FD           LDX   TMPX
   959  1E83  A5 FE           LDA   CHAR
   960  1E85  2A              ROL   A          SHIFT OFF PARITY
   961  1E86  4A              LSR   A
   962  1E87  60        GET6  RTS
   963                 ;
   964                 ;      INITIALIZATION FOR SIGMA
   965                 ;
   966  1E88  A2 01     INITS  LDX   #$01       SET KB MODE TO ADDR
   967  1E8A  86 FF            STX   MODE
   968                 ;
   969  1E8C  A2 00     INIT1  LDX   #$00
   970  1E8E  8E 41 17         STX   PADD       FOR SIGMA USE SADD
   971  1E91  A2 3F            LDX   #$3F
   972  1E93  8E 43 17         STX   PBDD       FOR SIGMA USE SBDD
   973  1E96  A2 07            LDX   #$07       ENABLE DATA IN
   974  1E98  8E 42 17         STX   SBD        OUTPUT
   975  1E9B  D8               CLD
   976  1E9C  78               SEI
   977  1E9D  60               RTS
   978                 ;
   979                 ;      PRINT 1 CHAR  CHAR=A
   980                 ;      X IS PRESERVED  Y RETURNED = FF
   981                 ;      OUTSP   PRINTS 1 SPACE
   982                 ;
   983  1E9E  A9 20     OUTSP  LDA   #$20
   984  1EA0  85 FE     OUTCH  STA   CHAR
   985  1EA2  86 FD            STX   TMPX
   986  1EA4  20 D4 1E         JSR   DELAY      10/11 BIT CODE SYNC
   987  1EA7  AD 42 17         LDA   SBD        START BIT
   988  1EAA  29 FE            AND   #$FE
   989  1EAC  8D 42 17         STA   SBD
   990  1EAF  20 D4 1E         JSR   DELAY
   991  1EB2  A2 08            LDX   #$08
   992  1EB4  AD 42 17  OUT1   LDA   SBD        DATA BIT
   993  1EB7  29 FE            AND   #$FE
   994  1EB9  46 FE            LSR   CHAR
```

```
D #  LOC     CODE        CARD
95   1EBB  69 00           ADC   #$00
96   1EBD  8D 42 17        STA   SBD
97   1EC0  20 D4 1E        JSR   DELAY
98   1EC3  CA              DEX
99   1EC4  D0 EE           BNE   OUT1
00   1EC6  AD 42 17        LDA   SBD      STOP BIT
01   1EC9  09 01           ORA   #$01
02   1ECB  8D 42 17        STA   SBD
03   1ECE  20 D4 1E        JSR   DELAY    STOP BIT
04   1ED1  A6 FD           LDX   TMPX     RESTORE INDEX
05   1ED3  60              RTS
06                 ;
07                 ;       DELAY 1 BIT TIME
08                 ;       AS DETERMEND BY DETCPS
09                 ;
 0   1ED4  AD F3 17  DELAY  LDA   CNTH30  THIS LOOP SIMULATES THE
11   1ED7  8D F4 17         STA   TIMH    DETCPS SECTION AND WILL DELAY
12   1EDA  AD F2 17         LDA   CNTL30  1 BIT TIME
13   1EDD  38        DE2    SEC
14   1EDE  E9 01     DE4    SBC   #$01
15   1EE0  B0 03            BCS   DE3
16   1EE2  CE F4 17         DEC   TIMH
17   1EE5  AC F4 17  DE3    LDY   TIMH
18   1EE8  10 F3            BPL   DE2
19   1EEA  60               RTS
20                 ;
21                 ;                     DELAY HALF BIT TIME
22   1EEB  AD F3 17  DEHALF LDA   CNTH30  DOUBLE RIGHT SHIFT OF DELAY
23   1EEE  8D F4 17         STA   TIMH    CONSTANT FOR A DIV BY 2
24   1EF1  AD F2 17         LDA   CNTL30
25   1EF4  4A               LSR   A
26   1EF5  4E F4 17         LSR   TIMH
27   1EF8  90 E3            BCC   DE2
28   1EFA  09 80            ORA   #$80
29   1EFC  B0 E0            BCS   DE4
30                 ;
31                 ;       SUB TO DETERMINE IF KEY IS
32                 ;       DEPRESSED OR COMDITION OF SSW
33                 ;             KEY NOT DEP OR TTY MODE     A = 0
34                 ;             KEY DEP OR KB MODE       A NOT ZERO
35                 ;
 6                 ;
37   1EFE  A0 03     AK     LDY   #$03    3 ROWS
38   1F00  A2 01            LDX   #$01   DIGIT 0
39                 ;
40   1F02  A9 FF     ONEKEY LDA   #$FF
41   1F04  8E 42 17  AK1    STX   SBD     OUTPUT DIGIT
42   1F07  E8               INX           GET NXT DIGIT
43   1F08  E8               INX
44   1F09  2D 40 17         AND   SAD     INPUT SEGMENTS
45   1F0C  88               DEY
46   1F0D  D0 F5            BNE   AK1
```

```
CARD # LOC    CODE        CARD
 1047                    ;
 1048  1F0F   A0 07            LDY   #$07
 1049  1F11   8C 42 17         STY   SBD
 1050                    ;
 1051  1F14   09 80            ORA   #$80
 1052  1F16   49 FF            EOR   #$FF
 1053  1F18   60               RTS
 1054                    ;
 1055                    ;       SUB  OUTPUT TO 7 SEGMENT DISPLAY
 1056                    ;
 1057  1F19   A0 00      SCAND  LDY   #$00        GET DATA SPECIFIED
 1058  1F1B   B1 FA            LDA   (POINTL),Y BY POINT
 1059  1F1D   85 F9            STA   INH        SET UP DISPLAY BUFFER
 1060  1F1F   A9 7F      SCANDS LDA   #$7F       CHANGE SEG
 1061  1F21   8D 41 17         STA   PADD       TO OUTPUT
 1062                    ;
 1063  1F24   A2 09            LDX   #$09       INIT DIGIT NUMBER
 1064  1F26   A0 03            LDY   #$03       OUTPUT 3 BYTES
 1065                    ;
 1066  1F28   B9 F8 00  SCAND1 LDA   INL,Y       GET BYTE
 1067  1F2B   4A               LSR   A          GET MSD
 1068  1F2C   4A               LSR   A
 1069  1F2D   4A               LSR   A
 1070  1F2E   4A               LSR   A
 1071  1F2F   20 48 1F         JSR   CONVD      OUTPUT CHAR
 1072  1F32   B9 F8 00         LDA   INL,Y      GET BYTE AGAIN
 1073  1F35   29 0F            AND   #$0F       GET LSD
 1074  1F37   20 48 1F         JSR   CONVD      OUTPUT CHAR
 1075  1F3A   88               DEY              SET UP FOR NXT BYTE
 1076  1F3B   D0 EB            BNE   SCAND1
 1077  1F3D   8E 42 17         STX   SBD        ALL DIGITS OFF
 1078  1F40   A9 00            LDA   #$00       CHANGE SEG
 1079  1F42   8D 41 17         STA   PADD       TO INPUTS
 1080  1F45   4C FE 1E         JMP   AK         GET ANY KEY
 1081                    ;
 1082                    ;       CONVERT AND DISPLAY HEX
 1083                    ;       USED BY SCAND ONLY
 1084                    ;
 1085  1F48   84 FC      CONVD  STY   TEMP       SAVE Y
 1086  1F4A   A8               TAY              USE CHAR AS INDEX
 1087  1F4B   B9 E7 1F         LDA   TABLE,Y    LOOK UP CONVERSION
 1088  1F4E   A0 00            LDY   #$00       TURN OFF SEGMENTS
 1089  1F50   8C 40 17         STY   SAD
 1090  1F53   8E 42 17         STX   SBD        OUTPUT DIGIT ENABLE
 1091  1F56   8D 40 17         STA   SAD        OUT PUT SEGMENTS
 1092                    ;
 1093  1F59   A0 7F            LDY   #$7F       DELAY 500 CYCLES APPROX.
 1094  1F5B   88        CONVD1 DEY
 1095  1F5C   D0 FD            BNE   CONVD1
 1096                    ;
 1097  1F5E   E8               INX              GET NEXT DIGIT NUM
 1098  1F5F   E8               INX              ADD 2
```

```
CARD #  LOC    CODE        CARD
 1099   1F60   A4 FC             LDY   TEMP        RESTORE Y
 1100   1F62   60               RTS
 1101                    ;
 1102                    ;              SUB TO INCREMENT POINT
 1103                    ;
 1104   1F63   E6 FA     INCPT  INC   POINTL
 1105   1F65   D0 02            BNE   INCPT2
 1106   1F67   E6 FB            INC   POINTH
 1107   1F69   60       INCPT2 RTS
 1108                    ;
 1109                    ;       GET KEY FROM KEY BOARD
 1110                    ;       RETURN WITH A=KEY VALUE
 1111                    ;       A GT. 15 THEN ILLEGAL OR NO KEY
 1112                    ;
 1113                    ;
 1114   1F6A   A2 21     GETKEY LDX   #$21        START AT DIGIT 0
 1115   1F6C   A0 01     GETKE5 LDY   #$01        GET 1 ROW
 1116   1F6E   20 02 1F          JSR   ONEKEY
 1117   1F71   D0 07            BNE   KEYIN       A=0 NO KEY
 1118   1F73   E0 27            CPX   #$27        TEST FOR DIGT 2
 1119   1F75   D0 F5            BNE   GETKE5
 1120   1F77   A9 15            LDA   #$15        15=NO KEY
 1121   1F79   60               RTS
 1122   1F7A   A0 FF     KEYIN  LDY   #$FF
 1123   1F7C   0A       KEYIN1 ASL   A           SHIFT LEFT
 1124   1F7D   B0 03            BCS   KEYIN2      UNTIL Y=KEY NUM
 1125   1F7F   C8               INY
 1126   1F80   10 FA            BPL   KEYIN1
 1127   1F82   8A       KEYIN2 TXA
 1128   1F83   29 0F            AND   #$0F        MASK MSD
 1129   1F85   4A               LSR   A           DIV. BY 2
 1130   1F86   AA               TAX
 1131   1F87   98               TYA
 1132   1F88   10 03            BPL   KEYIN4
 1133   1F8A   18       KEYIN3 CLC
 1134   1F8B   69 07            ADC   #$07        MULT (X-1) TIMES A
 1135   1F8D   CA       KEYIN4 DEX
 1136   1F8E   D0 FA            BNE   KEYIN3
 1137   1F90   60               RTS
 1138                    ;
 1139                    ;       SUB TO COMPUTE CHECK SUM
 1140                    ;
 1141   1F91   18       CHK    CLC
 1142   1F92   65 F7            ADC   CHKSUM
 1143   1F94   85 F7            STA   CHKSUM
 1144   1F96   A5 F6            LDA   CHKHI
 1145   1F98   69 00            ADC   #$00
 1146   1F9A   85 F6            STA   CHKHI
 1147   1F9C   60               RTS
 1148                    ;
 1149                    ;       GET 2 HEX CHAR'S AND PACK
 1150                    ;       INTO INL AND INH
```

```
CARD # LOC      CODE        CARD
 1151                       ;       X PRESERVED  Y RETURNED = 0
 1152                       ;       NON HEX CHAR WILL BE LOADED AS NEAREST HEX EQU
 1153                       ;
 1154   1F9D   20 5A 1E     GETBYT  JSR   GETCH
 1155   1FA0   20 AC 1F             JSR   PACK
 1156   1FA3   20 5A 1E             JSR   GETCH
 1157   1FA6   20 AC 1F             JSR   PACK
 1158   1FA9   A5 F8                LDA   INL
 1159   1FAB   60                   RTS
 1160                       ;
 1161                       ;       SHIFT CHAR IN A INTO
 1162                       ;       INL AND INH
 1163                       ;
 1164   1FAC   C9 30        PACK    CMP   #$30        'CHECK FOR HEX
 1165   1FAE   30 1B                BMI   UPDAT2
 1166   1FB0   C9 47                CMP   #$47        NOT HEX EXIT
 1167   1FB2   10 17                BPL   UPDAT2
 1168   1FB4   C9 40        HEXNUM  CMP   #$40        CONVERT TO HEX
 1169   1FB6   30 03                BMI   UPDATE
 1170   1FB8   18           HEXALP  CLC
 1171   1FB9   69 09                ADC   #$09
 1172   1FBB   2A           UPDATE  ROL   A
 1173   1FBC   2A                   ROL   A
 1174   1FBD   2A                   ROL   A
 1175   1FBE   2A                   ROL   A
 1176   1FBF   A0 04                LDY   #$04        SHIFT INTO I/O BUFFER
 1177   1FC1   2A           UPDAT1  ROL   A
 1178   1FC2   26 F8                ROL   INL
 1179   1FC4   26 F9                ROL   INH
 1180   1FC6   88                   DEY
 1181   1FC7   D0 F8                BNE   UPDAT1
 1182   1FC9   A9 00                LDA   #$00        A=0 IF HEX NUM
 1183   1FCB   60           UPDAT2  RTS
 1184                       ;
 1185   1FCC   A5 F8        OPEN    LDA   INL         MOVE I/O BUFFER TO POINT
 1186   1FCE   85 FA                STA   POINTL
 1187   1FD0   A5 F9                LDA   INH         TRANSFER INH- POINTH
 1188   1FD2   85 FB                STA   POINTH
 1189   1FD4   60                   RTS
 1190                       ;
 1191                       ;
 1192                       ;       END OF SUBROUTINES
```

```
CARD # LOC      CODE        CARD
 1194                       ;
 1195                       ;       TABLES
 1196                       ;
 1197  1FD5  00            TOP     .BYTE $00,$00,$00,$00,$00,$00,$0A,$0D,'MIK'
 1197  1FD6  00
 1197  1FD7  00
 1197  1FD8  00
 1197  1FD9  00
 1197  1FDA  00
 1197  1FDB  0A
 1197  1FDC  0D
 1197  1FDD  4D 49 4B
 1198  1FE0  20                .BYTE ' ',$13,'RRE',' ',$13
 1198  1FE1  13
 1198  1FE2  52 52 45
 1198  1FE5  20
 1198  1FE6  13
 1199                       ;
 1200                       ;               TABLE HEX TO 7 SEGMENT
 1201                       ;               0   1   2   3   4   5   6   7
 1202  1FE7  BF            TABLE   .BYTE $BF,$86,$DB,$CF,$E6,$ED,$FD,$87
 1202  1FE8  86
 1202  1FE9  DB
 1202  1FEA  CF
 1202  1FEB  E6
 1202  1FEC  ED
 1202  1FED  FD
 1202  1FEE  87
 1203                       ;               8   9   A   B   C   D   E   F
 1204  1FEF  FF                .BYTE $FF,$EF,$F7,$FC,$B9,$DE,$F9,$F1
 1204  1FF0  EF
 1204  1FF1  F7
 1204  1FF2  FC
 1204  1FF3  B9
 1204  1FF4  DE
 1204  1FF5  F9
 1204  1FF6  F1

CARD # LOC      CODE        CARD
 1206                       ;
 1207                       ;
 1208                       ;
 1209                       ;
 1210                       ;       INTERRUPT VECTORS
 1211                       ;
 1212  1FF7                        *=$1FFA
 1213  1FFA  1C 1C       NMIENT .WORD NMIT
 1214  1FFC  22 1C       RSTENT .WORD RST
 1215  1FFE  1F 1C       IRQENT .WORD IRQT
 1216                             .END


END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 4
NUMBER OF ERRORS =    0,  NUMBER OF WARNINGS =    0
```

SYMBOL TABLE

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES
ACC	00F3	76	587 851
ADDR	1CBE	694	687
ADDRM	1CC8	701	673
AK	1EFE	1037	1080
AK1	1F04	1041	1046
CHAR	00FE	90	950 951 952 959 984 994
CHK	1F91	1141	734 738 741 748 808 814 899 902
CHKH	17E8	97	158 265 289 299 301
CHKHI	00F6	82	730 755 782 799 819 1144 1146
CHKL	17E7	96	156 262 288 297 298
CHKSUM	00F7	83	729 758 783 801 821 1142 1143
CHKT	194C	295	234 237 242 244 256 308
CHT1	1982	336	344
CHT2	198E	341	338
CHT3	1991	342	340
CLEAR	1C64	645	803 836
CLKKT	1747	65	****
CLKRDI	1747	66	355 361 378 384 498 505
CLKRDT	1746	67	459 471
CLK1T	1744	62	358 364 381 387 501 508
CLK64T	1746	64	461 473
CLK8T	1745	63	****
CNTH30	17F3	101	613 622 1010 1022
CNTL30	17F2	100	625 1012 1024
CONVD	1F48	1085	1071 1074
CONVD1	1F5B	1094	1095
CRLF	1E2F	908	640 785 829
DATA	1CA8	680	****
DATAM	1CCC	704	675
DATAM1	1CCE	705	702
DATAM2	1CD0	706	699
DATA1	1CB0	686	698
DATA2	1CC3	697	692
DEHALF	1EEB	1022	947 956
DELAY	1ED4	1010	946 953 986 990 997 1003
DETCPS	1C2A	612	****
DET1	1C31	615	617
DET2	1C42	623	621
DET3	1C3A	619	624
DE2	1EDD	1013	1018 1027
DE3	1EE5	1017	1015
DE4	1EDE	1014	1029
DUMP	1D42	778	873
DUMPT	1800	121	****
DUMPT1	1814	131	134
DUMPT2	1833	148	177
DUMPT3	1854	163	166
DUMPT4	1865	173	152
DUMPV	1E01	873	867
DUMP0	1D48	781	826
DUMP1	1D4E	785	****

```
SYMBOL  VALUE  LINE DEFINED        CROSS-REFERENCES

DUMP2    1D86         811   817
DUMP3    1DA6         826   824
DUMP4    1D7A         805   792
EAH      17F8         106   151  791
EAL      17F7         105   149  789
FEED     1E07         876   861
FEED1    1E12         882   880
GETBYT   1F9D        1154   732  736  739  746  754  757  770
GETCH    1E5A         940   648  725 1154 1156
GETK     1C8D         667  ****
GETKEY   1F6A        1114   667
GETKE5   1F6C        1115  1119
GET1     1E60         943   945
GET2     1E6D         948   955
GET5     1E6A         947   627
GET6     1E87         962   944
GOEXEC   1DC8         841   711  865
GOV      1CD9         711   679
HEXALP   1FB8        1170  ****
HEXNUM   1FB4        1168  ****
HEXOUT   196F         323   314  316
HEXTA    1E4C         938   922  924
HEXTA1   1E55         933   931
HEX1     1978         328   326
ID       17F9         107   140  224  226
INCPT    1F63        1104   708  749  815  938
INCPT2   1F69        1107  1105
INCVEB   19EA         397   176  258
INCVE1   19F2         400   398
INH      00F9          85   647  760  825 1059 1179 1187
INITS    1E88         966   600  609
INIT1    1E8C         969   636
INL      00F8          84   646  779  823  885 1066 1072 1158 1178 1185
INTVEB   1932         281   123  185
IRQENT   1FFE        1215  ****
IRQP27   1BFE         519  ****
IRQT     1C1F         604  1215
IRQV     17FE         113   604
KEYIN    1F7A        1122  1117
KEYIN1   1F7C        1123  1126
KEYIN2   1F82        1127  1124
KEYIN3   1F8A        1133  1136
KEYIN4   1F8D        1135  1132
LOAD     1CE7         725   727  762  874
LOADER   1D3E         771   759
LOADE1   1D3B         770   756
LOADS    1CEE         728  ****
LOADT    1873         183   231
LOADT4   18B5         216   221
LOADT5   18D7         233   225  228
LOADT6   18EC         241   230
LOADT7   18F8         247   239  259
LOADT8   1915         261   250
LOADT9   1929         270   252  263  266
```

```
SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

LOADV    1E04         874   869
LOAD10   192B         271   268
LOAD11   18C2         223   218
LOAD12   190F         258   182
LOAD13   18FA         248   254
LOAD2    1D0E         746   751
LOAD3    1D1D         754   744
LOAD7    1D2E         764   ****
LOAD8    1D30         765   772
MODE     00FF          91   686  705  967
MODIFY   1E15         884   863
NMIENT   1FFA        1213   ****
NMIP27   1BFA         517   ****
NMIT     1C1C         603  1213
NMIV     17FA         111   603
ONE      199E         353   336  339
ONEKEY   1F02        1040  1116
ONE1     19A1         355   356  368
ONE2     19B0         361   362
OPEN     1FCC        1185   796  828
OUTBT    1961         309   141  157  159
OUTBTC   195E         308   144  146  174
OUTCH    1EA0         984   787  910  934
OUTCHT   197A         333   132  138  155  164
OUTSP    1E9E         983   831  835
OUT1     1EB4         992   999
PACK     1FAC        1164   651 1155 1157
PACKT    1A00         413   251  405  407
PACKT1   1A0F         421   418
PACKT2   1A15         426   429
PACKT3   1A22         433   414  416
PADD     1741          59   970 1061 1079
PBDD     1743          61   128  496  972
PCCMD    1CDC         717   671
PCH      00F0          73   594  719
PCL      00EF          72   591  717
PLLCAL   1A6B         493   517  518  519
PLL1     1A75         498   499  511
PLL2     1A84         505   506
POINTH   00FB          87   170  272  595  696  720  737  790  843  881  897
                           1106 1188
POINTL   00FA          86   169  271  592  688  691  695  718  740  747  785
                           812  833  845  877  879  886  900 1058 1104 1186
PREG     00F1          74   589  847
PRTBYT   1E3B         917   795  800  802  807  813  820  822  834  898  901
PRTPNT   1E1E         897   797  809  830
PRTST    1E31         909   642  767  912
PRT1     1E3A         913   ****
RDBIT    1A41         457   200  441  458
RDBIT2   1A53         467   468
RDBIT3   1A50         464   465
RDBIT4   1A63         476   477
RDBYT    19F3         404   223  233  236  241  243  261  264
RDBYT2   19F9         406   ****
RDCHT    1A24         439   209  216  248  404  406
RDCHT1   1A29         441   446
```

```
SYMBOL   VALUE  LINE DEFINED        CROSS-REFERENCES

READ     1C6A         648   870
RST      1C22         606  1214
RSTENT   1FFC        1214  ****
RSTP27   1BFC         518  ****
RSTV     17FC         112  ****
RTRN     1DC2         838   859  887
SAD      1740          58   615  623  638  660  943  948 1044 1089 1091
SAH      17F6         104   145  283
SAL      17F5         103   143  281
SAVE     1C00         587  ****
SAVE1    1C05         590  ****
SAVE2    1C0F         596  ****
SAVX     17E9          98   198  201  202  203  204  333  334  345  346  427
                            430  439  442  443  444  448  451
$BD      1742          60   126  195  360  366  383  389  457  467  494  503
                            510  766  974  987  989  992  996 1000 1002 1041
                           1049 1077 1090
SCAN     1DDB         854   652
SCAND    1F19        1057   657  662  664
SCANDS   1F1F        1060  ****
SCAND1   1F28        1066  1076
SHOW     1DAC         829   839  882
SHOW1    1DAF         830   643
SPACE    1DA9         828   855
SPUSER   00F2          75   599  608  841
START    1C4F         636   171  273  601  616  658  661  669  706  709  721
                            768  872
STEP     1CD3         708   677
STV      1DFE         872   857
SYNC     1891         197   211  220
SYNC1    1896         200   206
SYNC2    18AB         209   213
TAB      1871         182   189  191
TABLE    1FE7        1202  1087
TEMP     00FC          88   684  689  917  923  925 1085 1099
TIMH     17F4         102  1011 1016 1017 1023 1026
TMPX     00FD          89   940  958  985 1004
TOP      1FD5        1197   909
TTYKB    1C77         657   639  650
TTYKB1   1C7C         659   663  665
UPDATE   1FBB        1172  1169
UPDAT1   1FC1        1177  1181
UPDAT2   1FCB        1183  1165 1167
VEB      17EC          99   122  148  150  173  184  188  190  192  235  238
                            257  282  284  286  397  399
XREG     00F4          77   597  849
YREG     00F5          78   596  850
ZRO      19C4         376   341  342
ZRO1     19C7         378   379  391
ZRO2     19D6         384   385
```

```
INSTRUCTION  COUNT

     ADC          13
     AND           9
     ASL           7
     BCC           4
     BCS           5
     BEQ          26
     BIT          12
     BMI           9
     BNE          44
     BPL          15
     BRK           0
     BVC           0
     BVS           0
     CLC           8
     CLD           1
     CLI           0
     CLV           0
     CMP          38
     CPX           1
     CPY           0
     DEC           2
     DEX          14
     DEY           8
     EOR           2
     INC           7
     INX           5
     INY           2
     JMP          31
     JSR         115
     LDA         108
     LDX          29
     LDY          25
     LSR          22
     NOP           0
     ORA           6
     PHA           5
     PHP           0
     PLA           5
     PLP           0
     ROL          18
     RTI           1
     RTS          28
     SBC           5
     SEC           3
     SED           0
     SEI           1
     STA          81
     STX          14
     STY           7
     TAX           3
     TAY           3
     TSX           1
     TXA           3
     TXS           2
     TYA           4

# SYMBOLS =  204 (LIMIT =  400)      # BYTES = 1690 (LIMIT = 4096)
# LINES = 1242 (LIMIT = 1500)      # XREFS =  646 (LIMIT =  900)
STOP          0
```

NOTIZEN

NOTIZEN

commodore

Commodore Büromaschinen GmbH
Frankfurter Straße 171-175
6078 Neu-Isenburg
Telefon (0 61 02) * 80 03 · Telex 4185663 como d