THE APPLE HOUSE

HOW TO COMPUTERIZE YOUR HOME USING YOUR APPLE COMPUTER

JOHN BLANKENSHIP
Blankenship, John, 1948-
*853 1984 The Apple house

B1USINESS/SCIENCE/TECHNOLOGY DIVISION

TH 4812 Blankenship, John, 1948-
.B53 1984 The Apple house

B1USINESS/SCIENCE/TECHNOLOGY DIVISION

DATE BORROWER'S NAME

BUSINESS/Science/Technology Division

© THE BAKER & TAYLOR CO.
The Apple House

JOHN BLANKENSHIP
Senior Professor
DeVry Institute of Technology
Blankenship, John (date)
The Apple house.

Includes index.
TH4812.B53 1984 644'.028'54 83-46052
ISBN 0-13-038729-0
ISBN 0-13-038711-8 (pbk.)

Editorial/production supervision and interior design: Nancy Milnamow
Cover design: Jeannette Jacobs
Manufacturing buyer: Gordon Osbourne

© 1984 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-038729-0

Prentice-Hall of Australia Pty. Limited, Sydney
Prentice-Hall Canada Inc., Toronto
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Prentice-Hall of Southeast Asia Pte. Ltd., Singapore
Whitehall Books Limited, Wellington, New Zealand
## Preface

<table>
<thead>
<tr>
<th>Part I  THE GOAL</th>
<th>Chapter 1</th>
<th>What to Expect</th>
<th>Chapter 2</th>
<th>System Overview: Hardware</th>
<th>Chapter 3</th>
<th>System Overview: Software</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapter 4</td>
<td>The Voice Recognition System</td>
<td>Chapter 5</td>
<td>The Voice Synthesizer System</td>
<td>Chapter 6</td>
<td>The System Clock</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part II  THE HARDWARE</th>
<th>Chapter 4</th>
<th>The Voice Recognition System</th>
<th>Chapter 5</th>
<th>The Voice Synthesizer System</th>
<th>Chapter 6</th>
<th>The System Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapter 4</td>
<td>The Voice Recognition System</td>
<td>Chapter 5</td>
<td>The Voice Synthesizer System</td>
<td>Chapter 6</td>
<td>The System Clock</td>
</tr>
</tbody>
</table>

### Preface

### Part I  THE GOAL

Chapter 1  What to Expect

Chapter 2  System Overview: Hardware

Chapter 3  System Overview: Software

### Part II  THE HARDWARE

Chapter 4  The Voice Recognition System

Chapter 5  The Voice Synthesizer System

Chapter 6  The System Clock
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>The Failsafe System</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>The BSR Control System</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>The Telephone System</td>
<td>44</td>
</tr>
<tr>
<td>10</td>
<td>The Interrupt System</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>The Tone Generator</td>
<td>54</td>
</tr>
<tr>
<td>12</td>
<td>Input/Output (I/O) Ports</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Part III THE SOFTWARE</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The Software Organization</td>
<td>65</td>
</tr>
<tr>
<td>14</td>
<td>The Primary Tasks</td>
<td>72</td>
</tr>
<tr>
<td>15</td>
<td>The Initialization Module</td>
<td>82</td>
</tr>
<tr>
<td>16</td>
<td>The Voice Request Module</td>
<td>86</td>
</tr>
<tr>
<td>17</td>
<td>The Phone Control Module</td>
<td>98</td>
</tr>
<tr>
<td>18</td>
<td>The Security Module</td>
<td>103</td>
</tr>
<tr>
<td>19</td>
<td>The Event Timing Module</td>
<td>109</td>
</tr>
<tr>
<td>20</td>
<td>The Monitor Movement Module</td>
<td>114</td>
</tr>
</tbody>
</table>
Part IV THE SYSTEM

Chapter 21 Installation and Maintenance 120

Chapter 22 Expansions and Enhancements 123

APPENDICES

Appendix A Home Control Program (BASIC Portion) 125

Appendix B Home Control Program (Machine Language) 139

Appendix C Clock Set and View Program 143

Appendix D Time Table Editor 145

Appendix E Vocabulary Generator and Tester 148

Appendix F The HELLO Program 151

Appendix G Partial Product and Vendor Listing 152
The computer age is no longer coming. It’s here! The number of personal computers in the United States now numbers in the millions. Computer literacy is becoming commonplace. Even grade school students are learning to use and program computers.

But the really exciting news about computers is their steadily decreasing price and increasing performance. With some models selling for under $100, it is possible for anyone to own a computer. Unfortunately, getting a computer is the easy part. Once you have one you need to find ways of making it work for you. Hundreds of programs are available to make your computer a useful appliance. It can become a friendly machine and play games with you. It can help organize your life by cataloging your record collection or personal library. You might even save some money by putting your budget and checkbook under the supervision of your computer.

These features are wonderful indeed, but if you are like me, you will find them a little less than exciting. I don’t think a home computer is one that you use in the home. A home computer should control the home. It should handle security, supervise the lights, control the heat, answer the phone, and much much more.

The technology to computerize your home is available now. If the idea excites you, then prepare to enter the computer age and make your house the first APPLE HOUSE on the block.

JOHN BLANKENSHIP
Although computers are great for balancing checkbooks and formulating household budgets, this book was written for those persons that wish to extend the computer's power into far more exciting areas.

Imagine the convenience of having your "house" keep track of how many people are in each room and to turn off all the unnecessary lights. It does not take long till you take for granted that the lights should come on automatically when a room is entered.

Sometimes, of course, you will desire to take personal control and override such automation. Fantasize for a moment about the ability to just talk to your house to have your wishes carried out. And when you talk to the house, it will answer back—in its own voice.

I'm talking about a home that can also answer the phone, monitor security, and wake you up in the morning. And these are just a few of the capabilities you can expect.

All this is not some pie in the sky dream. Such computerization is a reality today in my home, and this book can make it a reality in yours.

The system described in this book is by no means the ultimate computerized home. It is, however, a complete and workable system that can have some very beneficial advantages. For example, you can expect to save a few dollars on your heating costs. Mine went down 30 percent.

You can save in other areas, too. Security systems and phone-answering devices are not cheap, and most will do far less for you than
your computerized home can. And my system is just the beginning. The only limit to your computerized home is your imagination.

I first started thinking about putting a computer in control of my home in 1976 when I bought my first Apple computer. It seemed so obvious to me that low-cost computing power would lead to residential automation.

Unfortunately, as the years passed it seemed that no one had both the motivation and the skill to tackle a project of such size and complexity. Finally, I gave up waiting and decided to do it myself.

I had five major goals in mind when I began the project. First, I wanted to use off-the-shelf equipment whenever possible. With the microcomputer becoming more of a consumer rather than a hobby item, I felt sure that many nontechnical people would like to duplicate my efforts.

The second goal was to devise a system that would be easy to install. I figured that even the most dedicated enthusiast would not want computerization enough to completely rewire a home in order to control the lights.

I also wanted the system to be easily expanded and customized. The same innovation and sense of adventure that would make a person want such a home would also make them want it his or her way. I decided to make the system as flexible as possible to ensure that subsystems could be added or deleted as necessary. Such a design would also make adapting the system to another brand of computer at least bearable.

Anytime that you add a lot of frills like voice recognition and speech synthesis to a computer project, you can expect the cost to increase. I wanted these frills though, and as my fourth goal I wanted to devise a system that would make such frills at least somewhat cost effective. The true value of some of the system's capabilities can only be measured in your desire to have them, but the increasing cost of energy has made any substantial reduction in the use of gas and electricity extremely exciting.

Finally, I wanted the system to be usable. Any system, no matter how cost effective, will not save anything if, because of the hassle of using it, it stays turned off. I wanted it to be friendly and helpful. I wanted a system that would be fondly missed the moment it broke down.

As you might guess, most people do not know what to expect from a computerized home, let alone what the requirements might be to build one. I would like to prepare you for what is to come and inform you of what knowledge and skills you will find useful if you decide to computerize your home.

If you wish to have your home computer set up exactly like mine, you might be able to do so with a relatively small amount of knowledge. You
can buy or build everything described in this book and then type in the programs provided.

I suspect, however, that very few, if any, individuals are going to run out and buy a microcomputer system just to control their home. It is much more likely that this book is being read by someone that already has a computer and is looking for a new and novel way of using it. Thus, I will assume that you have read the literature that came with your system and that you are reasonably familiar with programming in BASIC.

At least some knowledge of assembly language programming would also be nice, but unless you wish to change my system considerably it is not absolutely necessary.

If you feel you need help in any of the areas mentioned, many books are available to help you. *Microprocessors and Microcomputers* (2nd ed.) by R. Tocci and Laskowski can give you an in-depth look at both the hardware and software associated with the 6502. *Interface Projects for the Apple II* by R. Hallgren offers several specific and interesting examples. Both of these books are published by Prentice-Hall, Inc., Englewood Cliffs, N.J.

Radio Shack sells the *Engineers Notebook* by Forest Mims III. For a very modest price it will provide you with a wide variety of electronic hobby information.

Don't feel limited to these suggestions. You can find a wide selection of suitable books at almost every computer store. In your search, don't overlook the traditional bookstores in your local mall or shopping center. Even they are now catering to the needs of the computer hobbyist.

Many of the pieces of equipment needed can be purchased from various sources. If you happen to like construction projects, you can save money by building some sections yourself. Complete schematics are given for some of the more appropriate areas. There are a few things that just cannot be purchased anywhere, and if you want them you will have to make them yourself. For that reason, I suggest that for maximum benefit of this book you have some general knowledge of electronics.

My home is controlled by an Apple II Plus from Apple Computer, Inc. I chose this system because of the wide range of peripheral devices that can be purchased for it and because of the ease of connecting these devices to it.

This does not mean that if you own some other type of computer that this book will not be of interest to you. The majority of the home control system is written in BASIC and is easily used on almost any computer system. This text is designed to not just show you what I did, but to
explain how it was accomplished and why a particular approach was chosen.

I also will explain in as much detail as space permits why what I have done works. With this information, an industrious person should be able to not only implement this system on other microprocessor systems, but to customize it to meet precisely his or her requirements.

If you have not purchased a computer yet and wish to do so, I would strongly urge that you wait until you have finished the book as you will be better prepared to determine if a given system can be easily adapted to computerizing your home.

The home control system will have five basic functions. Although the modules for each of these functions are described in detail later in this book, I would like to discuss them briefly here. They are as follows:

Voice request
Phone control
Security management
Event timing
Monitor movement

The voice request module allows you to talk to the computer using a microphone. I use wireless mikes and have them throughout the house in strategic places. There are numerous submodules for this module that allow you to control house lighting, ask what time it is, turn on the security system, have the system place a phone call for you, shut off the stereo, and much more.

The phone control module serves as an intelligent answering machine. Not only can the computer answer the phone and record a message from the caller, but it will tell you when you get home how many messages you have waiting. This module also allows you to call the computer and request status information about the house or to control the lights or the heating system from a remote location.

The module for security management can watch for movement outside the home as well as monitor entry at doors and windows. If an intruder is detected, appropriate action can be taken. Since the system is intelligent, it can react differently if you are home or away or if the intruder is outside or inside. It can even call a neighbor and inform them of the situation.

Event timing provides a means to establish a predetermined sequence of events that needs to be done on an ongoing basis. For example, the
house might wake you at seven o'clock on weekdays but let you sleep till ten on weekends. It should control the house lighting automatically to give a lived in look whenever no one is home. The heat should be turned off or down whenever no one is home and turned on again prior to your arrival.

The heart of the event timing module is an **event table**. Each entry in the table represents an action to be taken and the time and conditions for it to occur. For example, one entry might indicate that the heat should be cut off at midnight, and another entry turns the heat on at seven o'clock the next morning, but only if someone is home. If no one is home, a different set of times and events is automatically used.

There will be a separate event or time table for each day of the week. Once you have created the proper tables, the house can handle the functions without further direction. The tables can be edited when necessary to reflect changes in your life-style.

The **internal movement** module watches various sensors to allow the system to keep track of where people are in the house. This information is used to decide what lights should be on and off.

The overall organization of the program provides that you may use any of these systems alone, all of them together, or even add some of your own. This means that if you cannot afford the hardware for all the modules at first, you may still enjoy the same functions now and easily add to them later.

I don't propose that the system here is perfect for everyone. There are certainly many unexplored possibilities available to the imaginative person. My hope is to provide you with one complete system with all the whats, whys, and hows so that you can expand and customize until your particular wants and needs are met.

This book is divided into four parts. Part I is an introduction and overview of the entire system. It is very important because it prepares you for the details to come by showing you the big picture.

Part II covers all the hardware required by the system. Things that cannot be purchased are examined in much more detail than off-the-shelf items. I chose to build several sections because of the financial incentives and also I greatly enjoy the additional learning experience. If you do not enjoy such tinkering do not be discouraged by the number of schematics in Part II, because commercial substitutes can be purchased in most cases.

Part III covers the actual program coding used to implement each program module. You will find the code well structured and easy to follow. The explanations of each module's operations will make your customizing easy.

Part IV shows how to implement the hardware and software into a
complete system. It shows how to initialize the main program and explains the utility programs that are used to maintain the required tables. The last chapter discusses some ideas for additions and expansions.

The appendixes provide a complete listing of the entire home control program and its related maintenance programs. They are listed directly from my computer in order to ensure the absence of errors. There is also a partial list of vendors to assist you in locating some of the items used in the home control system.

I have tried to organize the book in a manner that will provide you with the best possible understanding of how the entire system actually works. This is especially important because, as I have mentioned, it is doubtful that you will wish to duplicate exactly my work. I expect you to improve on the system in every way you can. In fact, by the time you are reading this I probably will have added several improvements to my own computerized home.

Wherever applicable, I try to use novel approaches in my computerization, especially if it makes the system more cost effective. You may choose to use some of these approaches if you customize my system, although perhaps in different ways. I hope that I have provided stimulation in enough areas to ensure that you can create a system that meets your specific needs.

The rest of Part I will provide you with a complete overview of the system, both the hardware and the software. Read these chapters more than once if you really are planning to add a computer to your home. Make sure that you truly understand how the system is organized and what each module’s responsibility is.

While you are reading, do not worry about the details of how a module accomplishes its task, but rather concentrate on how the combination of modules forms a means of intelligent control. As an example, let us assume we have a module that dials the phone. Do not be concerned with how the phone is actually dialed. That will come later. For now observe when and why the “dial phone” module is called to perform its task.

Not only will this approach provide you with a better understanding of the system, but it will make it easier to transfer this program to other computers or to implement it with other peripherals.

Let me cite one more example to help make my point. One of the modules in the system reads the electronic clock to find out what day it is and the time. Numerous clocks are available for the Apple, not to mention those for other computers. If you choose to buy a different clock from mine, you would only have to rewrite the “read clock” module, which is
very short. All the modules that use the “read clock” module could be used without modification.

Since low-level modules, such as the “read clock” module, have single specific objectives, they are generally easy to rewrite. In fact, the instruction manual for the peripheral involved should provide all the necessary information to do so.
Now that you have a good idea of where we are going, it's time to get started. In this chapter I will present an overview of the system hardware. Remember, try not to worry about details. Later sections will discuss in detail the schematics and/or sources for purchasing the items mentioned here.

It is also important to mention that you are by no means limited to using the hardware described in this book. The system is set up in such a manner that you can very easily use alternative devices in almost every case.

Figure 2.1 shows a block diagram of the entire home control system. I realize this diagram is complicated, but I feel it is necessary to get a good overview of the system. Later in the book a whole chapter will be devoted to each item mentioned here.

Let us examine each part, starting from the top of the figure. The voice recognition unit provides us with the primary means of communicating our commands to the home. It consists primarily of an analog-to-digital (A/D) converter that transforms the amplitude variations of your voice into patterns of 1's and 0's that the computer can analyze. It will be from this analysis that the computer can determine what word or words are spoken.

As nice as it is to speak to a computer, you will probably find, as I did, that the novelty wears off very fast if you have to walk halfway through the house to get to the microphone to turn on a light. There are
Figure 2.1 Detailed block diagram of the complete home control system.
many possible solutions to this problem, but I chose to use a wireless microphone, which is comprised of a small radio transmitter and receiver.

The wireless system must be of high quality or it will be difficult, if not impossible, for the computer to understand your commands. Fortunately, such systems are not as expensive as you might imagine. You may even decide to have several microphones throughout the house at strategic locations.

Naturally, if you are going to talk to the house, you will expect it to talk back. You will notice that I sometimes refer to the house rather than the computer. This transition may seem a bit harsh at the moment, but you will find it to be a very natural adjustment.

The speech synthesizer will produce a voice that can easily be sent to every room in the house. The speakers can be mounted in the ceiling or placed in any convenient location. I prefer to have mine behind a plant or otherwise camouflaged so that the illusion that the house is speaking is further enhanced.

As you will see in later sections, the house will take on a personality. The personality chosen for this book is meant to be somewhat neutral, but you will be able to tailor yours to suit yourself. It will be a combination of the personality and the fact that you can communicate with the house as you would with a person (with speech) that will soon convince you that the house and the computer have become one.

Some people might find such thoughts a bit disconcerting. I suspect, however, that the idea of machine intelligence is somewhat intriguing to you or you would not be reading this book to begin with.

Continuing down Figure 2.1, we come to the telephone portion of the system. The major element is an intercom telephone. The phone itself, as well as many of the associated controls, is interfaced to the computer through the Apple game connector.

The game connector provides an easy means to connect external devices to an Apple computer. It has three primary capabilities. They are paddle inputs, switch inputs, and annunciator outputs.

Game paddles and joysticks are the most common items plugged into the connector. For such devices, the paddle inputs convert the value of a variable resistance into a number between 0 and 255. The switch inputs are used to read the on/off condition of a switch, which usually controls a game function in some manner.

The annunciator outputs are seldom used by the commercial devices for the game connector. Since they are output pins, they can be used to turn on or off a transistor, relay, or almost anything you desire.

The switch inputs and annunciator outputs are nothing more than
one-bit input/output (I/O) ports. The home control system will use them to read from or to control external devices. If you are not using an Apple to computerize your house, you can easily substitute a standard bidirectional port for these pins. Chapter 12 discusses input/output ports in detail and will provide you with additional information.

The paddle inputs would be a little more difficult to simulate on another computer. For this reason, I have chosen not to make use of them in the home control system. Let’s look specifically now at the I/O pins of the game connector and how they will be used.

The Apple’s game connector has four output pins that can be used to control external devices, and it has three input pins for reading the status of external data. One of the output pins is used to turn the intercom phone on and off. This allows the computer to answer the phone for you and permits you to talk over the intercom from anywhere in the room. The phone dialer provides a means for the computer to call a requested party. Proper control of these functions can mean that you never again have to touch the phone.

A ring detector informs the computer when incoming phone calls require processing. One of the computer’s responsibilities will be to act as a phone-answering machine if you are not available to get to the phone. Rather than have an electrical connection directly to the phone lines, which requires special equipment and a notification to the phone company, I decided to let the computer communicate with callers just as you would—by speaking on the intercom phone.

The computer already has a voice, and it could easily be used to inform the caller of your message. Unfortunately, most voice synthesizers presently affordable by the average hobbyist take a little getting used to. This should not imply that they are extremely difficult to understand; on the contrary, after only a few minutes of practice most synthesizers become reasonably clear.

A person calling on the phone, however, does not expect to be greeted by an electronic voice and does not have time to get used to the computer’s peculiar accent. Add to this any distortion added by the intercom phone and you might have many of your callers simply hanging up rather than leaving messages.

However, I just happened to have a couple of cheap cassette recorders laying around from the days before disks and arrived at my final phone answering system as follows.

One of these recorders is set in the playback mode using a 30-second endless loop cassette. The other has a regular tape and is left in the record state. Two more of the game connector output pins are used to control the
on/off condition of the two tape recorders. When the phone rings, the computer can turn on the intercom, play the 30-second message, and then record any reply from the caller.

A sound detector is used to allow the computer to determine if there are sound changes or voices present. Using this capability, the system can record only as long as the caller is actually talking. This permits messages of any length and also keeps you from having a 30-second tape of a dial tone if someone hangs up.

For most calls, all you will want the computer to do is record the message. If it is you that is calling, however, you might want to communicate with the computer to issue special requests. The computer could then give you a status report for the house, this time in its own voice. You might wish to tell the house to turn on the heat, turn off a light, or anything else that the system is capable of handling.

Naturally, you will not want just anyone to be able to communicate directly with the system, so a way must be devised to inform the house that it is you calling. My method is simple. While the message is being recorded, the computer will continue to monitor the sound detector, watching for a particular sequence of sound. You can indicate it is you calling by playing the proper sequence into the phone, using a small hand-sized tone generator. As long as you have this generator, you can gain access to the computer anytime you call the house.

Another detector is used by the system to determine when you are trying to get the computer to respond to voice commands through the wireless mike. The process of understanding speech is very difficult and time consuming and will not be invoked unless it is required. Normally, the computer will not bother to analyze sounds present at the A/D converter. It will, however, watch the input from the voice detector, which is very easy. If you should pick up a wireless mike and say "computer" (or anything else for that matter), the computer will see the voice detect pin change and respond by saying something like "Yes sir, what can I do for you?" and then monitor the A/D converter for your response.

The last output pin on the game connector is connected to an ultrasonic transducer. This device is much like a loudspeaker for your stereo except that it is much smaller and can produce ultrasonic waves, which are sound waves above the range of human hearing.

To see the importance of such a capability, let’s discuss a commercial device called the BSR System X-10 Ultrasonic Command Consol. This device can be purchased in many department stores. Sears Roebuck even markets the same device under their own name.
The console plugs into any 110-volt outlet and generates coded pulses for each button pressed on the console. These pulses travel throughout your house on the existing house wiring. You can buy remote switches from BSR that can replace wall switches and receptacles at appropriate places in your house.

These remote switches contain electronic circuits that constantly monitor the power lines for pulse codes from the command console that can inform them to turn on or off. This allows you to control lights throughout the house simply by pressing buttons on the command console.

The BSR system provides a relatively inexpensive means to control many of your lights and appliances from a single location. The ultrasonic model of the command console has an additional feature that makes it especially attractive for our purposes.

Just as some TV sets can be controlled by aiming an ultrasonic unit at them, the BSR ultrasonic console can be manipulated using a cordless controller that generates ultrasonic pulses that can be received and interpreted by the command console.

Under program control, the house computer will generate the appropriate ultrasonic pulses using the game connector and the transducer previously mentioned. In this manner, the computer may control any device in the house that is equipped with a BSR remote switch.

Naturally, we will want the computer to control lights and appliances when verbally commanded to do so. In many cases, though, we will not wish to be bothered with trivial matters and will wish the house to handle them on its own.

Some examples of trivial actions would be turning off the heat every night at midnight and then back on at seven the next morning. Since these two actions occur on a regular basis, they should be handled automatically.

To provide a means for the house to handle such actions, a special clock will be interfaced to the computer. It will tell the house what time and what day it is. Using this clock, the house will be able to handle routine tasks without any human intervention at all.

There is another type of action that the house should handle for you automatically. As mentioned earlier, the system should monitor the movement of people throughout the house and control the lights accordingly. To provide the movement information, pressure-sensitive switches are placed at strategic positions under the carpet. These switches are connected to an input port so they may be read by the system.

This same port will monitor other switches and detectors that will provide the information required by the house to monitor security matters.
The closure of any switch will trigger an interrupt to get the computer’s attention. The computer’s reaction to interrupts is discussed in more detail in the next chapter.

The last piece of hardware to be mentioned now is the system sanity timer. It will ensure that the system software cannot fail even if unpredictable errors occur. A complete explanation of this subsystem will come later, but for now just accept the fact that if the system ever fails due to software faults the entire system will be reinitialized from disk and proper control resumed.

Now that you have had your first glimpse of what the home control computer will look like, let us start examining an overview of the software required to give the house its intelligence.
Now that we have an overall picture of the hardware needed to computerize your house, it's time to begin looking at how the programs will be organized. The importance of understanding the software cannot be overemphasized. Although the home control system presented in this book is complete and usable as described, it can be enlarged and customized in many different directions with very little if any additions to the hardware. To do so, however, will require an in-depth knowledge of the present program.

As with the overview on hardware, do not worry about details when reading this section. Later in the book we will find out exactly how to accomplish each function described here. At this point, however, you should be striving to see how each module relates to the overall operation of the program.

The flow chart in Figure 3.1 shows the primary organization of the home control program. The system is organized into five major modules. We will look at each separately.

The *initialization* module is only run when the system is powered up. It checks to see what day it is and sets up the appropriate files. It can adjust to the present status of events without human intervention. This is particularly important if you are out of town and the system has to restart itself after a temporary power failure.

Once initialization is complete, control passes to the four modules...
that handle the primary actions. The functions of these four modules were described in Chapter 1.

Usually, there is nothing happening that would require the attention of any of these modules. Control is continually passed from one to another, waiting, or if you prefer, watching for a situation to occur that requires attention. For example, if the voice detector does not show that someone is requesting to talk with the system, then control passes on to the phone module.

If the phone is not presently ringing, then the security module takes over. If conditions show that all is well, the time module processes the next item in the present time table. Afterward, the entire process starts over.

In this manner, the loop normally completes very quickly so that the system appears to be watching all the potential situations simultaneously. If, while the system is looping, any of the modules find that a task needs to be done, the system will spend the time necessary in that module to complete it. Once that task is completed, the looping will continue.

Let’s briefly look at lines 10 to 60 in Appendix A. As odd as it may seem, these lines represent the main home control program. Of course,
each of the subroutines called will have to be expanded and coded. I will
discuss each in detail as we move through the book.

Although the system can only process one thing at a time, usually it
will only take a second or so to respond to an external stimuli, such as an
intruder or the phone ringing. Such a small delay obviously presents no
real problem. If the system is busy in one of the modules though, it might
take much longer to respond.

At first glance, this possibility of lengthy delays appears to present a
real problem. Fortunately, there is very seldom a situation in which one
module needs to take immediate priority over another module. Furth­
more, in almost every instance that a valid conflict occurs, there is a
human present to decide on which module should get control.

Perhaps some examples of such conflict can make this clearer. Assume
for a moment that you are talking with the computer, thus making
the voice module busy. If the phone should ring and you want the house to
process it, you may simply say “goodbye.” This would cause the voice
module to terminate, and the system would continue through the loop and
process the phone call.

If you were not home to tell the computer “goodbye,” there would be
no problem to begin with, because you would not have been talking to the
computer and it would have answered the phone automatically.

For a second example, assume that the computer is answering the
phone during the time an event, such as turning off the heat, is scheduled
in the time table. The event timing module is designed so that, even if the
scheduled time has passed, the event will occur the next time the module is
run.

Thus even in the worst case an event would be processed within a
minute or two of its scheduled time. Furthermore, remember that such a
delay will only occur if the phone just happens to ring when you are not at
home and just when an event is to be processed. The infrequent occur­
rence of such situations does not, at least for me, deserve enough attention
to increase considerably the complexity of the system.

If we go through a similar process for each possible situation it can be
shown that there are two situations that truly represent a potential
problem. One problem is that the security system is inactive while the
computer is acting as a phone-answering system. In this case, even a one­
minute delay might determine if an intruder could be easily scared away by
turning on appropriate lights and the radio. As before, no problem exists
unless just the proper set of events happens at just the right times. Even
so, I value security enough to see this as a major concern.
To solve this dilemma, the phone module will also monitor a security flag (which indicates some kind of security problem exists) and, if necessary, end the phone call and continue looping. The security module would then react appropriately to whatever situation had occurred.

As you can see, because it is seldom that two tasks are required at exactly the same time, the entire system can appear to be handling all the tasks simultaneously even though the program is sequential in nature. It is this sequential format that makes the system so easy to program.

The second problem is one that cannot be handled on the first come, first served priority arrangement that I have shown so far. You may have noticed that, although I mentioned earlier that the house would monitor the movement of people, there is no module in the main loop to provide this function. For this function the system needs the ability to respond immediately, no matter what it is doing. After all, if you have to wait even a few seconds for the lights to come on when entering a room, this module loses its usability.

To provide this immediate action, all the floor switches as well as
security detectors will cause a computer interrupt to be generated. An interrupt simply tells the computer to abandon what it is presently doing and process some other program. When this second program is completed, control is passed back to the original program at the point where the interrupt occurred.

A functional flow chart of the interrupt program is shown in Figure 3.2. When an interrupt occurs, the computer will read the switches to determine if it is a security problem or if residents are moving inside the house. If it is a security situation, certain flags will be set (including the security flag) to convey that information to the main program.

If it was internal movement of the residents that caused the interrupt, the appropriate lights will be turned on and/or off. Since this interrupt program always requires less than two seconds to run, it is unlikely that any module presently active will even notice the interruption.

If you should feel unsure of how all this is going to be accomplished, please don’t be discouraged. So far we have only touched the surface. Rest assured that the details are coming.

By now you should, however, have a better idea of what the computerized house can be expected to do. You should know what general types of hardware will be necessary to allow the computer to gather information and to communicate with the outside world.

You should also be starting to see how the program will handle the many different types of tasks that will be required. If you are unsure of any of these areas, you may wish to reread some parts of this section. Otherwise, let’s move on to Part II and start examining the hardware in detail.
In this and the following chapters in this section, we will be looking at the hardware required to automate your home. There are several things that should be emphasized before we begin.

First, all the hardware discussed in this section is only a suggestion. True, I have tested this particular combination of hardware and have solved the problems related to it. In doing so though, I also tried to develop a system that was very modular.

Each section of the hardware is interfaced to the system only through its function. This means that any equipment that can provide the correct function should be easily adaptable for use in the home control system.

As you read this section, you should also realize that the intelligence in the home control system is found in the software and not the hardware. This section is not intended to explain how the system works. What it will do is to provide a thorough understanding of the environment that the software must run in. With this in mind, let’s look first at the system that processes voice requests.

Giving a computer the power to recognize speech is far from an easy task. Fortunately, several companies have done a great deal of research in this area and have produced commercial products for personal computers that can aid us tremendously.

One such company was Heuristics. They produced two low-cost voice recognition systems. Their Speech Lab could recognize 32 words
and their Speech Link was capable of 64. Unfortunately, Heuristics went out of business just after I had completed the computerizing of my home using a Speech Link.

Some dealers may have Speech Links still in stock. If you cannot find one, there is no need to worry. It will be very easy to interface virtually any speech recognition system made for your computer to my software. I will discuss equipment substitution in detail later.

The Speech Link is a user-trainable, isolated word recognition system with a maximum standard vocabulary of 64 words. User-trainable means that the prospective user must initially "say" each word for the system so that it can create a template in memory of what each word looks like. Each time a sound or word is to be recognized, a new template will be formed and it will be compared against the templates in memory to find the best match.

Isolated word recognition simply means that the system can only recognize a word by itself. This is opposed to being able to pick a word out of a sentence. If you were to read this paragraph out loud, it would become quickly apparent that in normal speech there are very few pauses between words. This lack of pauses is one of the major obstacles to be overcome if a machine is to effectively understand human speech. To stay cost effective, we will have to settle, at least for the present, for isolated word recognition. You will see later that the proper software can make this restriction much less binding.

The Speech Link is very easy to use. It effectively allows the Apple command "INPUT A$" to wait, not for a keyboard entry, but for a word to be spoken. This word may actually be a phrase of several words as long as there is no significant pause between the words.

The software explained in Chapter 16 will utilize the contents of the string variable (in this example A$) to determine the action required. For now, however, we need to examine the special hardware requirements of the system.

We will not examine the Speech Link itself in any detail. For such information you should consult a Speech Link manual. Instead we will be looking at some hardware that will make the Speech Link much more usable for our application.

There are two major problems with most commercial speech recognition units. First, once the system is directed to recognize something, it will, as you would suspect, wait until some sound has occurred and then analyze it. On the surface, this appears very usable.

To see the problem, consider the structure of our home control system. It normally loops from one module to another. If there is no verbal
input, we want the system to continue on through the loop to perform the next task. If, during each loop, we ask the Speech Link to check for a sound, it becomes quickly apparent that if no sound occurs the system will "hang" in that module.

To solve this problem, we simply need some means for the main program to know when someone wants to speak to the computer. We might do this by simply connecting a switch to an input pin. Depending on the position of the switch, the computer would get either a 1 or a 0 when reading that pin. The computer could use this information to decide whether or not to turn control over to the Speech Link.

We will return to the first problem in just a moment. Another problem with using voice control of a house is that you may not be beside the computer, or more specifically beside the microphone, when you wish to issue a command. This problem can be solved in a variety of ways.

If you only have a couple of places in the house that you wish to issue verbal commands from, you might actually run microphone cables to those spots. A mixer could then be used to combine the signals before sending them to the Speech Link.

Although this method can be very acceptable, I decided to add more flexibility by using a radio transmitter and receiver. In this way, the microphones are completely wireless and permit use anywhere in the home.

The method chosen to solve the first problem will effect the solution to the second. If, for example, you run cables to each mike, you could also run wires for switches to tell the program when to turn control over to the Speech Link. If you are using wireless mikes, though, you lose all their flexibility if you must go to a specific place to press a switch in order to get the computer's attention.

Since I wanted to use wireless mikes, I solved this problem by having a special circuit watch for sound changes at the receiver and place the appropriate 1 or 0 on one of the Apple's game connector input pins. There are three of these pins on the Apple game connector, two of which are normally connected to switches on the game paddles.

Figure 4.1 shows the schematic for the sound detection circuitry. Note that the output of the circuit is connected to the game connector so that a program could decide if there was any sound present by reading the status of the appropriate address. This would be done in BASIC with the PEEK command.

The operation of the circuit is straightforward. The output of the receiver is sent to the first operational amplifier, which is used as a buffer to prevent loading. The gain of the second amplifier is controlled by the
Figure 4.1 This circuit serves as a voice detector so that the computer can quickly determine if someone wants to issue verbal commands to the system.
A wireless microphone allows you to talk to the computer from anywhere in the house.

200K potentiometer. The amplifier output is turned upside down by the 7404 inverter. The pulse from the inverter is stretched by the 555 in order to make it easier to recognize.

Whenever a sound is present at the wireless mike, the game connector pin will be held high. The main program can monitor this pin to determine if it should stop and carry out verbal commands.

The wireless mike must be of a fairly high quality or the Speech Link will not operate satisfactorily. I tried several units with very limited success. Finally, I found one that seems to be perfect. It is a model WMS-49 wireless microphone system by Mura. It is pictured in Figure 4.2 and comes with a microphone and a receiver, both operated from 9-volt batteries. Since the receiver needs to be left on continuously, I use a battery eliminator.

Another nice feature of the Mura unit is that it has two outputs on the receiver. The microphone output connects to the microphone input on the Speech Link while I use the auxiliary output to drive the voice detection circuitry as shown in Figure 4.1.

If you use some other wireless microphone system, you should consider having the following features. The receiver and transmitter should be crystal controlled to ensure drift-free reception. The frequency response should be as wide as possible. The Mura unit is 30 to 18,000 hertz
(Hz). A microphone output (in addition to an auxiliary output) is really necessary to ensure a proper impedance match to the Speech Link.

The gain of the receiver should be adjusted to give the best results with the Speech Link. The only way to make this adjustment is through trial and error. Once that is done, you will need to adjust the 200K potentiometer (Figure 4.1) so that the LED glows only when you speak into the mike.

This completes the description of the hardware required to allow voice command of the house computer. Even though I am using the Speech Link, almost any voice recognition unit may be used. In later chapters we will see how the proper software can give intelligence to the recognition capability discussed here.
Being able to talk to your computer is very nice. If the “conversation” is to be complete though, we will have to give the house the power to talk for itself. With today’s technology, we have many methods to choose from.

We could utilize an A/D (analog-to-digital) converter to sample an actual voice. These samples could be stored in the computer’s memory and then played back at any time by using a D/A converter. Using this method, you can get excellent quality speech, but at a very high cost. What you must give up is memory. It is not unreasonable to use well in excess of 4000 bytes of memory for each second of speech using this method.

Linear predictive coding (LPC) is another method for reproducing speech. This technique is used in the Texas Instruments’ Speak & Spell. The data compression technology used permits a second of speech to be stored using only a few hundred bytes of memory, and the quality of speech is still very good.

The problem with LPC is that the data used to generate the speech are very complex and are provided to the user in the form of read-only memory (ROM). There are many words and phrases offered, but you are, for the most part, limited to a severely restricted vocabulary.

As I designed the home control system, it became quickly apparent
that I needed a nearly unlimited vocabulary if the system was to communicate in the manner I was planning. I also needed a method for storing speech that was extremely efficient if my home was to say more than a few sentences.

A company called Votrax has been utilizing a technology that differs distinctly from the two methods just discussed. First, they do not copy or reproduce a human voice. Instead they have developed a set of filters that can simulate the human vocal track.

By sending codes to their system, you can instruct it to produce any of the basic sounds used in the English language. These sounds are called phonemes. Since each phoneme requested requires only one byte of memory, and since most words require less than ten phonemes, the amount of memory used is almost trivial.

In addition, you can have the computer say anything you wish, as long as you can decide on the basic phonemes required for each word. As you would expect, you do have to give up something in exchange for all these pluses. The quality of the speech is somewhat less than with the other methods.

At first it may be difficult to understand. My experience has been that after only a little practice you get used to the peculiar "accent" found in the Votrax synthesizers. Recently, Votrax introduced a single integrated circuit that has all the filters on it. This has spurred the release of a new type of product from several companies.

The new products not only have the capability to produce phonemes on command, but they have a built-in microprocessor with the software to convert standard English text into its appropriate phonemes. Some of these products also add inflection to the speech, which makes it much easier to understand. Any of these devices satisfy all the major requirements I had for giving a voice to my home.

I own two such voice synthesizers, a Votrax Type 'N Talk and a Microvox from Micromint. They are shown in Figure 5.1. Either can perform satisfactorily for your computerized home.

Although I have not done any scientific studies, I think the Type 'N Talk makes fewer errors when it translates text to speech, although there is not really much difference.

The Microvox is much faster and has less pause between the words spoken. It also offers you the ability to control the pitch on individual words or even syllables so that the quality of the speech can be greatly improved. With a little work, you can even have it waking you each morning to a song.
There are many speech synthesizers suitable for giving your home a voice.

The Type 'N Talk requires a serial port to connect it to a computer. You may use either serial or parallel with the Microvox. If you really enjoy construction projects, as I do, you will appreciate the fact that the Microvox is available in kit form.

Whether you choose an assembled unit or the kit, the Microvox comes with complete schematics and numbered parts. This is not true with the Type 'N Talk.

Either system is about as easy to use as a printer. In fact, I connect them to my computer using a standard RS-232C printer interface. You activate it exactly as if it were a printer (PR#1 for example), and the system simply says anything that would normally be printed.

Figure 5.2 shows the hardware organization for using a synthesizer. A RS-232C serial card connects the unit to the Apple. The connecting cable may need to be slightly different for different serial boards, but any qualified dealer should be able to assist you.

There is an amplifier built into the speech synthesizers, but it is small and only meant to drive one or two speakers. I connected the synthesizer output to a 5-watt transistor amplifier that had been gathering dust in the garage for years. Since I did not want a booming voice from the house, the 5 watts was plenty to drive several more speakers.

To keep the speaker load as close to 8 ohms as possible, you should use a series–parallel arrangement such as the one in Figure 5.2. Depending on the number of speakers you require, you may not be able to get an exact match, but you should at least keep it in the right ball park.

The Type 'N Talk has a frequency control that allows you to control the pitch of the voice manually. At the high end the computer sounds like a chipmunk, and at the low end it is more like a 45 record playing at the 33 speed. Experiment a little with the frequency until you find a sound you like.
Figure 5.2 An additional amplifier allows the synthesized voice to be heard throughout the house.
The Microvox defaults on power up to a mode similar to the Type 'N Talk, that is, monotone speech. If you wish, you can insert your own pitch variations under software control or even choose a mode where inflection is added automatically, although somewhat arbitrarily.

Remember, this only provides the computer with the ability to speak. The software is what will give it the intelligence to know what to say.
One of the most important capabilities you can have in a home control computer is the ability to tell time. This capability is not only desirable but essential for many of the tasks we expect our home to perform. Let’s look again at some of those tasks.

Whenever the system is first turned on, we will want it to initialize itself. Part of this initialization is to establish if anyone is home. We will see later that it will simply ask (verbally, of course) to see if anyone answers. If a short power failure were to cause a system reinitialization in the middle of the night though, we would not want the house to wake us just to ask if we were home.

In such a case the system will be able to ascertain whether it should check to see if anyone is home by reading the system clock. If the time falls within a given range, the house will make some appropriate assumptions and carry on without disturbing anyone.

To handle the preceding situation, we need a system clock with two capabilities. First, as with any clock, it must be able to keep track of the time. It does not need to be extremely accurate. Hours, minutes, and seconds will be more than ample for our needs. We will also need a means to determine if it is A.M. or P.M.

The second requirement is the ability to operate in the event of a power failure. This necessitates some form of battery backup. Let’s look
The Hardware Part II

<table>
<thead>
<tr>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>USE</th>
<th>RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LSD · SECONDS</td>
<td>(0-9)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>MSD · SECONDS</td>
<td>(0-5)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>LSD · MINUTES</td>
<td>(0-9)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>MSD · MINUTES</td>
<td>(0-5)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LSD · HOURS</td>
<td>(0-9)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>MSD · HOURS</td>
<td>(0-9)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DAY OF WEEK</td>
<td>(0-6)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LSD · DATE</td>
<td>(0-9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>MSD · DATE</td>
<td>(0-3)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>LSD · MONTH</td>
<td>(0-9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>MSD · MONTH</td>
<td>(0-1)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>LSD · YEAR</td>
<td>(0-9)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>MSD · YEAR</td>
<td>(0-9)</td>
</tr>
</tbody>
</table>

*D1 - D0 USED FOR HOURS
D2 - 0 FOR AM OR 1 FOR PM
D3 - 0 FOR 12 HOURS OR 1 FOR 24

Figure 6.1 Functions available on the 5832 clock chip.

at some of the other expected uses of our clock to see what other functions will be needed.

The event timing module was mentioned briefly earlier. It will give us the capability to arrange for events to occur at prescheduled times. These events will be stored in time tables on the disk. For them to be effective, there will be a separate table for each day of the week. You could, for example, schedule different wake-up calls on different days or to turn the TV off at 9 P.M. on weeknights.

For our system to provide us with such flexibility, we must provide it with the ability to know what day it is. Our clock then should also include a calendar. Actually, we can see that for our purposes we do not need a complete date, but only the day of the week. There will be other uses for the clock in our home control program, but the capabilities listed so far will cover these uses.

To provide our system with such a clock, we only need to visit a local computer store or flip through the pages of a personal computer magazine. Many clock boards are available for the Apple computer, and nearly all provide at least the capabilities that we have mentioned. Most in fact are vastly superior to our modest needs.

Because of this and due to my love of tinkering with hardware, I decided to build my own clock. After comparing several of the possible alternatives, I decided to center my efforts on the MSM5832 clock/calendar chip from OKI. It had all the necessary requirements, including the fact that it was inexpensive.

Some of its features are a crystal-controlled oscillator on the chip, a single 5-volt supply, a 12- or 24-hour format, and a low power dissipation for battery backup. The entire clock is packaged in an 18-pin DIP.

Data are sent to and read from the 5832 over four data lines. These
data lines are connected internally to the clock chip’s 13 major registers. The function of these registers is shown in Figure 6.1. If, for example, you wish to find out what day it is, you could have a program read register 6. The chip uses four address lines to select the register to be used.

For some functions you need to read two registers. The seconds are a good example. Suppose the time was 28 seconds. Register 1 would hold a 2 and register 0 would be 8. By manipulating the address lines and then reading the data lines, we can have access to the correct time and date. We would set the time of course by writing to the same registers.

In addition to the address and data lines, four other lines control the 5832. These lines are the chip select, hold, read, and write. Figure 6.2 shows how all these lines will be connected to a peripheral interface adapter (PIA) I/O chip. The PIA provides a convenient means to connect something to a computer. Chapter 11 will discuss the PIA in more detail. For now, just assume that the computer can read or write over any of the port lines.

The four least significant bits of port A connect to the clock chip’s

![Figure 6.2](image)

Figure 6.2 The 5832 clock chip can easily and economically provide the house with the time and day of the week.
data lines. Reading these will give the contents of one of the 13 registers, depending on the address lines. In my case, I tied the most significant address line to ground (pin 7). This means that only the registers 0 to 7 are available to be read from or written to. These eight registers are enough to provide all the capability we discussed earlier.

I did not require the ability to set the time by writing to the clock’s registers without my intervention, so I used a switch to manually select either the read or the write line. This uses up all the lines on port A. Unfortunately, we also need to control the chip select and the hold line. These two lines can be connected together and will allow us to effectively stop the clock momentarily. This is important during a read, because if a register is changing at that time, extraneous data might result.

Since I had to use more than one port to control the clock, you might wonder why I did not go ahead and use the second port to control the read and write lines separately. I could also have used a line for address 3 rather than grounding it. Such a method would certainly give the clock more flexibility and power. My method results in all the capability I need, and it leaves seven lines free on port B, which is enough to drive many peripherals, such as a printer, or for expansion of the home control system.

A crystal is connected between pins 16 and 17 to ensure accuracy. Even so, you will need to make some minor adjustments with the associated capacitors to fine tune the clock speed.

The battery backup circuit that I used is a little simpler than others that I have seen. It is meant to be used with two ordinary nonrechargeable batteries. I have been using AA cells for nearly a year with no need for replacement. You should make sure the diodes used are germanium rather than silicon in order to keep the supply voltage within tolerances. There is one point that may be obvious, but it should be mentioned anyway. The ground for the clock (and all peripherals) should be connected to the Apple’s ground.

As simple as it looks, that’s all that is required to make a clock for our system. Of course it will take the right software to read it and set it. This software will be discussed in detail in later chapters, and I hope you will find that it will be easier than you think.

I do wish to remind you that the home control program is designed to work with just about any clock available for the Apple. As we will see, only a few minor modifications will be necessary.
If I am going to have a computer control my home, I want that computer to work—always! It must be able to contend with every reasonable situation that I can anticipate and perhaps even some that I cannot.

There will of course be some limitations. We cannot reasonably expect it to work when someone unplugs it or attacks it with a hammer. We can, however, make it intelligent enough to handle power outages and perhaps even recover from its own software errors and intermittent hardware problems.

Perhaps you are wondering why I am worrying about such a "trivial" problem. For those of you that fall into this category, a short explanation can be beneficial.

When you put the computer in charge of your home, you are delegating tremendous responsibility. Let's examine some of the functions given to the computer. One is to act as your alarm clock. Imagine how inconvenient it would be if you could not depend on the house to get you up at the correct time. Have you ever had a clock that was not dependable? How long did it take you to get rid of it?

Consider getting up in the winter to a cold house because the computer has failed to properly control the heat. Don’t forget how failures in the phone answering or security system could affect you.

These and other functions represent very important tasks. If the
home control system is to really be valuable, it must be given responsible tasks. If we are going to trust part of our daily activity to a computer, we must make sure that it can react appropriately to problems that might occur.

I don't mean to imply that personal computers are not reliable or that software cannot be debugged completely. On the contrary, microcomputer systems are extremely reliable and my software has gone through considerable testing.

Even so, you never know when one of the thousands of tiny transistors inside just one of the chips is going to fail. You never know when just the right pattern of events will put the software through a sequence that was never anticipated. It is unlikely that either of these situations will occur, but neither you nor I can guarantee that they will not.

Furthermore, there are other potential failures that are nearly impossible to predict. They all center around the power lines. No matter how good a power company you have in your area, the line voltage is constantly fluctuating because of changing loads like air conditioners and other large appliances. In addition, there may occasionally be large noise spikes on the line due to line switching at the power company, lightning striking a transformer, or even a shorted vacuum cleaner at your neighbor's home. To keep things simple, we will group these types of power line failures into two categories: (1) lengthy failures that cause the computer (and everything else in your home) to be inoperative, and (2) short, but large fluctuations in the line voltage.

Let's examine each of these failures. Lengthy delays are probably the easiest to deal with. The Apple computer has a power on reset and special initialization software in its auto-start ROM. This may sound complicated, but it boils down to the fact that an Apple computer will automatically RUN the "HELLO" program on whatever disk is in the drive whenever the Apple is turned on.

Thus whenever a lengthy power failure occurs, the "HELLO" program will take control as soon as power is restored. To restart the system, we need only have two things happen. First, the "HELLO" program should perform any special initialization and then RUN the main home control program. The home control program then has the responsibility of finding out what is going on. It has to determine if anyone is home. It can do that by just asking and seeing if anyone answers. You would not want the system to wake you in the middle of the night just to ask if you were home, though, so we will expect it to have some intelligence.

The house should check the time and determine if you should be
asleep. If so, we will want it to make certain assumptions until such time as they can be verified. We will discuss the actual assumptions in later chapters, but I hope you get the idea. The computer has got to be able to intelligently restart itself anytime there is a massive power failure.

If you have been following closely, you are probably wondering about the short noise spikes that might occur. You can, and probably should, purchase a noise suppressor at your local computer store. Such devices are relatively cheap and will solve all but the largest spike problems. Those will, unfortunately, cause us many problems when they occur.

Notice I said when, not if. With the computer controlling your home, it will be on 24 hours a day. Eventually, some spike is bound to happen, and when it does the result usually is that a few memory locations will be changed. It is possible, depending on the locations changed, that everything will continue on as if nothing had happened; more likely, the system will "hang" somewhere.

In this case the hardware is fine, but the software has failed. It has become insane for all practical purposes. We need some means to prevent such insanity from wrecking the integrity of our system. I wanted a simple and inexpensive method to achieve the desired results. I realize that no system can be perfect, but because of the responsibility delegated to this system, I wanted it as failsafe as possible.

My solution was a System Sanity Timer. Functionally, it works like this. The timer is nothing more than a resettable counter that is always counting forward from zero. If it ever gets to 15, it will cause a system reset. A reset, as we just described, will cause the system to rerun the home control program. Our assumption is that the counter will never get to 15 unless the software is insane.

We will attempt to ensure this by having commands throughout the program that will reset the counter to zero. As long as everything is going well, the program will reset the counter before it reaches 15. If anything causes the system to hang up, the counter will time out and the system will completely reinitialize itself.

Figure 7.1 shows the circuitry required to implement a system sanity timer. A 74191 counter chip is the basis for the timer. Any pulse on the load line (pin 11) will cause the counter to clear by forcing a parallel load of the data in, which is zero. The counter is driven from a 555 oscillator set to produce one pulse every 3 seconds or so. This means we have around 45 seconds before the counter must be reset again.

If the switch S1 is open, the Apple's reset line is not connected to the
counter and the sanity timer is not functional. Since most programs will not reset the counter, you will want to leave the switch open unless you are running the home control software.

When the switch is closed, the sanity timer is enabled. Pin 13 produces a low pulse whenever the counter reaches 15. This will reset the Apple and also the counter. The counter can also be cleared by pushing the reset button on the Apple or by pulsing the restart input. This can be connected to any output pin on the PIA. We will see how this is handled in the chapter on I/O ports.

Under constant control of intelligent software, the system sanity timer makes our home control computer almost invincible.
A primary capability needed in our home control computer is to be able to manipulate lights and appliances throughout the house. There are many approaches from which to choose.

One option would be to rewire the house. Each light could be controlled using a relay or a solid-state switch such as an SCR or TRIAC. Not only would we have to add these devices to each circuit we wished to control, but we would also have to run control lines from one of the computer's output ports to each device. This approach is certainly feasible, but hardly cost effective.

As mentioned in Chapter 2, the BSR System X-10 remote control system provides the solution for us. Figure 8.1 shows some of the elements in a BSR system. Let's look at it now in more detail.

The center of the system is the command console. It has 16 buttons to designate one of 16 lights to control and 6 additional buttons to indicate what should be done to the lights. These six functions are ALL LIGHTS ON, EVERYTHING OFF, DIM, BRIGHT, ON, and OFF. Let's look at each function in more detail.

The first thing you probably noticed is that the ALL ON refers to lights only and the ALL OFF means everything. This distinction can be made because the remote switches can be designated for use with appliances or with lights. As we will see later, this can be very beneficial in our home control system.
The remote switches come in a variety of packages. The easiest to install are just small boxes that plug into any standard outlet, just like a mechanical timer. You may then plug a lamp or appliance into the receptacle on the box. For those persons not wishing to have unsightly boxes, there is a BSR receptacle that can completely replace your present one. If you wish to control overhead lights, you may replace the normal room switch with the appropriate BSR model designed for that purpose.

Each BSR remote has a concealed small switch that allows you to specify its operating number. This has to be between 1 and 16, just like the buttons on the command console. There is also a house code switch. Its purpose is to make sure you and any neighbors that also have a BSR will not interfere with each other’s units.

All the remotes, as well as the console, have some very sophisticated electronics in them. There are no extra wires connecting the console to any of the remotes. Each unit is attached only to the house wiring. When a button is pressed on the console, a specific high-frequency code sequence is injected into your house wiring. The amplitude of this signal is such that it will be totally ignored by normal lights and appliances.

The BSR remotes have a special filter to trap out this special frequency. Once isolated, the frequency is amplified and then decoded to
determine its content. Let’s use an example to see how things are controlled. Suppose the button marked 3 was pressed on the console. The coded signal (indicating 3) would be added to the power lines. Every remote would analyze that code. Only the module that finds a match between its switch code and the one from the power lines will activate itself.

If you were to press several other number buttons, let’s say 6 and 7, on the console, there would be a total of three remotes activated. If we now press the ON button, each of the active remotes will turn itself on. Similarly, if we hold down the DIM button, they will each dim the light they control. For the safety of your appliances, the appliance modules will not respond to a DIM command.

This product alone can provide your home with a great deal of flexibility. Imagine the things you could do if the computer could just press the buttons on the command console. BSR makes a device that will help us to do just that. The command console also comes in an ultrasonic model. It is identical to the standard model except that it can be controlled from across the room using a BSR ultrasonic controller.

The ultrasonic controller sends coded signals to the command console in much the same way as some TV sets are controlled remotely. If we can produce the same signals as the controller, we can easily take control of the command console and indirectly have control of up to 16 functions in the home.

To simulate the signals from the controller, we will have to know a little more about them. I obtained information about the BSR signals from an article by Steve Ciarcia in the January, 1980 issue of Byte magazine. They are made up of short bursts of a 40-kilohertz tone. To make things easier, let’s assume that the actual code is made up of 1’s and 0’s. A 1 will be 160 cycles of the 40-kilohertz tone followed by 4 milliseconds of silence. A 0 will be 48 cycles, followed this time by a 6.8-millisecond pause. Figure 8.2 shows this more clearly.

Figure 8.3 shows how the 1’s and 0’s just described can be put together to form the code word sent by the ultrasonic controller. The first

![Figure 8.2](image_url)
The bit in the word will always be a 1. It could be thought of as a start bit. The function of this bit is to indicate the start of a new transmission so that the receiving circuitry will be ready for it.

The next five bits make up the actual code indicating which button was pressed. To prevent extraneous errors due to noise on the lines, the same button code is transmitted again but in reverse order. The end of the word is marked by 640 cycles of the 40-kilohertz tone, and there should be at least 24 milliseconds of delay before the next word is sent.

Figure 8.4 lists the five-bit codes for each BSR button. With this information, all we need do is figure out a way to control a 40-kilohertz tone as previously described. We could do it with hardware, but I am one of those people who would rather resort to software. Not only is it usually cheaper, but I would rather change a program than unsolder and resolder wires.

If you own an Apple or know someone that does, you have probably heard it play music by toggling the output line going to the speaker. Obviously, we can produce tones in such a manner. After a few calculations, I determined that it was possible to produce a 40-kilohertz tone using the same technique.

Of course, I could not use a speaker owing to its low frequency

<table>
<thead>
<tr>
<th>BUTTON</th>
<th>CODE</th>
<th>BUTTON</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>ALL OFF</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>ALL L. ON</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>ON</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>OFF</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>DIM</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>BRIGHT</td>
<td>26</td>
</tr>
</tbody>
</table>

ALL CODES ARE IN DECIMAL

Figure 8.4 The table shows the data codes required for each of the BSR buttons.
response. I connected a 40-kilohertz transducer to the Apple game connector, as shown in Figure 8.5. The proper software can easily generate all the possible codes and allow the computer to control the BSR, and thus anything that can be plugged into an outlet. The software for generating the codes will be discussed further in Chapter 20.
The ability to control the telephone is a very important part of the home control system. The telephone is used in many of the modules. You may, for example, verbally request the computer to call someone or to answer the phone when you are home.

When you are not home, the house should be able to answer the phone and record messages without any human intervention. You should also be able to call the computer and get a status report or to control lights in the house. If a security problem occurs, you should expect your house to call you or a neighbor and report the problem.

As you can see, without the ability to control the phone, the capabilities of the home control system would be significantly reduced. In keeping with my overall philosophy, I will keep the hardware to a minimum. I also will use commercially available equipment whenever possible. This will be especially true when we actually connect something to the phone lines.

The phone company has many regulations governing what may be connected to their lines. This may seem unfair to you since you rent the lines, but look at it from their point of view. The phone company has a large quantity of expensive equipment that might be damaged if you were to inject improper signals. Perhaps even more importantly, the service of other phone company subscribers could be impaired or even eliminated for a time if improper equipment is used.
Any manufacturer may apply to the FCC for permission to produce a device for direct connection to the phone lines. Since there are many devices on the market that are approved for direct connection, we will utilize them for our interface. Although you are allowed to connect approved items to the lines, you are required to notify the phone company of the connection and of the FCC registration number, which should be included either on the device itself or in the owner’s manual.

The first of these devices that we are going to use is the D.C. Hayes Micromodem. A modem (modulator demodulator) is a device that converts the digital signals in the computer to analog tones that can be easily transmitted over the phone lines to another computer. In effect, a modem makes it just as easy to talk to a computer in another state as it is to talk to your printer.

The reasons for owning a modem are increasing everyday. Many computer services are becoming available over the phone lines. In some areas you may get your bank balance or order merchandise using a personal computer equipped with a modem. You can also subscribe to networks that contain enormous data bases that have the potential of allowing a home computer to make truly useful decisions.

A D.C. Hayes Micromodem can provide your Apple with all these capabilities. It also has some very special features that will aid us tremendously in our home control telephone system. The Micromodem has auto-dial and auto-answer, which means that it has the firmware and hardware required to dial a number for you and to connect itself to the phone lines whenever it detects a ring.

We will use these features whenever the home control program needs to call someone or to determine when the phone is ringing. Since the Micromodem is FCC approved for direct connection to the phone line, we have these capabilities without any hassles with the phone company.

In addition to detecting rings and dialing the phone, we must also be able to connect a cassette recorder and player to the phone lines if the computer is to act as a phone answering system. Again I solved the problem by going to a commercially available device. The Duofone 101 is an electronic telephone amplifier from Radio Shack.

The amplifier connects directly to the phone lines and allows the phone to perform like an intercom. Since the amplifier is powered from the phone lines, it can be controlled very easily from the computer. All we need do is replace the on/off switch on the Duofone with the contacts of a relay. Anytime the computer wants the amplifier on, it only needs to activate the relay.

Since the amplifier provides intercom-type operation, all our phone
company interfacing is solved. The computer can monitor the Micromodem to determine if the phone is ringing. If you are home, the Duophone can be activated so that you can talk from anywhere in the room.

If you are not home, the computer can activate the amplifier first and then the cassette player. The player would play a 30-second endless loop tape with a message for the caller. At the end of the tape the computer should turn off the player and start the recorder. As long as someone is talking, the recorder should be left on.

To perform all the functions just mentioned, we will need to build some special hardware. Figure 9.1 shows the circuit we will need. We will use three of the game connector outputs to control the telephone amplifier and the two tapes.

The 7404 inverters are used to buffer the game connector outputs. Each inverter controls a relay by turning on or off a transistor. The telephone amplifier relay should have its contacts substituted for the amplifier's switch.

Most cassette recorders have a remote jack next to the microphone jack. The relay contacts should be connected to plugs that will mate with the remote jacks on the recorders. When they are plugged in and the play and record buttons are depressed, your computer will have complete control of both tape machines.

Controlling the telephone amplifier and the tape recorders is essential for our system, but it is not enough. If the computer is to know when to hang up on a caller, it must be able to "hear" when the call is over. Figure 9.2 shows a circuit to give the computer ears.

You might be wondering why we just don't use the Speech Link for this purpose. There are two reasons. First, the Speech Link is connected into the computer by way of a wireless mike and is not active unless the wireless mike is on. Second, we do not need to know what words are actually being said. If we can just determine if there is a changing sound present, we will have all the information that we need.

The circuit of Figure 9.2 is essentially the same as the one found in Chapter 4 except that a crystal microphone is used as the input. As long as a sound is present, the game connector pin will be a logic 1. The 200K potentiometer can be used to control the sensitivity.

If we place the crystal microphone in front of the telephone amplifier, someone leaving a message will cause the game connector pin to alternate somewhat randomly between a 1 and a 0. If the person does not say anything for a period of time, the input will remain 0. If the person hangs up and produces a dial tone, the pin will remain high.
Figure 9.1 Using a driver circuit and small relays, the computer can control two tape recorders and a telephone amplifier.
Figure 9.2 A sound detector provides a means for the computer to listen to phone calls.
The home control program will sample this input pin on the game connector and determine when to hang up on a caller that is leaving a message. This completes the principal hardware required if the home control program is to take charge of your phone. The only thing left out is the ability to control your house by calling from a remote location. This addition will be explored in Chapter 11.
As was mentioned earlier, the majority of the home control system is sequential in nature. Each task is usually completed before the next is started. Some tasks, though, such as watching for movement throughout the house, require that the controlling module be running at the time of the event that is being monitored.

The Apple’s microprocessor is a 6502, and it, as well as other microprocessors, has an interrupt line. Actually, the 6502 has two interrupt lines, a nonmaskable (NMI) and an interrupt request (IRQ). My system will use the IRQ. When this line is pulsed low, the 6502 will abandon the program that it is presently running and jump to a second program. When the second program is complete, the original program is continued as if nothing had happened.

The program to handle the special events that cause interrupts will be covered later, but right now we want to look at the hardware required to generate the interrupts at the appropriate times. To better understand the requirements of this hardware, we need to examine each event that will cause an interrupt.

One such event is the movement of people throughout the house. We will detect this movement by using strategically placed switches under the carpet. Refer to Figure 10.1 to see my switch arrangement, which handles four rooms. Anytime someone walks from one room to another, the person will step on two of the switches. The first switch pressed indicates
the room the person came from and the second switch tells the computer which room was entered.

You can buy pressure-sensitive switches for this purpose, but I found it cheaper to make mine by attaching tape switch to a 12-inch square of aluminum. Tape switch looks like a thick piece of tape, but it has two conductors that touch whenever a few pounds of pressure is applied. You can decrease the sensitivity by placing a sheet of foam rubber between the aluminum and the floor. This will make sure that pets will not activate the switches.

If you do not have carpet, you can still utilize the home control program, but your switches will be much more expensive. Instead of a physical switch at each doorway, you will need some kind of beam and a way to detect if it is broken. You could use either a light or an ultrasonic system, depending on your preference.

Regardless of what type of switch arrangement we use, we will need a means for the computer to determine both when a switch is pressed and which one it is. If we were to connect each of the four switches to four bits of an eight-bit input port, the computer could read that port and determine which was pressed. If the system could constantly monitor the port, it could also tell when a switch was pressed.

Such monitoring would require so much attention that the system would not be able to get much else done. If we could somehow generate an interrupt whenever any of the switches is pressed, then the main program would not have to keep checking and the interrupt program could easily determine which switch caused the interrupt by reading the input port.

Figure 10.2 shows a circuit that will solve our problems. As you can see from the drawing, there are eight switches instead of the four we have been using. More on this later. Each switch is connected to a network of two resistors and a capacitor that helps eliminate bounce (noise) that
Internal and external movement switches are connected to a computer port, as well as to the interrupt line. When an interrupt occurs, the port can be read to determine the exact cause.

occurs when mechanical contact is made. This network also ensures that even if a switch is held closed for extended periods of time the pulse generated will be only momentary.

All the inputs are sent to the input port. They are also functionally ORed together using a NAND gate and an inverter. If any switch is closed, the 555 monostable multivibrator will be triggered. It will produce a short pulse, which is inverted again and sent to the Apple’s interrupt line.

Whenever any switch is stepped on, an interrupt is generated. This will cause the interrupt program to be run. This program, which is discussed in Chapter 20, will read the port and utilize the information
obtained to determine the actual movement. Once the system determines which lights should be altered, it will produce the required ultrasonic tones to activate the BSR console.

The remaining four switches are intended for use in the security system. A closure of any of these switches will also run the interrupt program. If it determines that it was one of these switches that was pressed, the program will set and clear some memory locations that BASIC can PEEK at to determine what security problem has occurred.

One of my security inputs is connected to magnetic reed switches on the doors. Since the doors are heavily deadbolted, activation of these switches will imply that the owner of the house has returned. A second input is connected to window switches in series so that any open window will activate the pin. Such a signal will indicate that someone is entering the house without proper authority.

My third security input is used to indicate movement outside the house. Many commercially available sensors can provide such an indication. The least expensive are usually infrared beams or ultrasonic movement detectors. If you wish to spend a little more, there are devices that react to body heat. The actual type of sensor or sensors you use should be based on the physical conditions around your house.

The fourth and last input for the security port is simply labeled miscellaneous and generally is not used in the system described in this book. Activation of this input will set a flag so that your software can react as you see fit. A potential use for this input that I am exploring is a driveway sensor. This would allow the system to determine when a car enters or leaves the driveway.

As with most of the hardware in the home control system, the real potential will not be apparent until the software is explained. We will soon see how the software can utilize the limited information gained from a few switches and create a system that can react intelligently to movement both inside and outside the house.
In Chapter 9 we discussed most of the telephone system. The only capability left out in that chapter was the ability to control lights and other items in the house when you are away by calling your home computer.

To provide a simple means to communicate with the computer by phone, we will need some special hardware. We cannot use the Speech Link system for two reasons. First, the Speech Link is connected through the radio receiver and would not be able to listen to phone conversations. Second, the distortion and noise on the phone line make the Speech Link much less effective.

This does not cause us any real problems. We just need to analyze our needs. The first thing the system must be able to do is to recognize the difference between a standard caller trying to leave a message and you wishing to control something in the house or wanting a status report.

Your system must also be able to understand what you want once it has determined it is you calling. Without the luxury of using a speech recognition system, we must severely limit the commands that we expect the system to understand. One of my initial ideas was to use the tones on a push-button phone. Each tone was to be used as a separate command. It sounded like a good idea until I tried building the required tone decoder.

It turns out that such a tone decoder is either very unreliable or more expensive than I was willing to accept. My next step was to consider decoding only two tones, in order to keep the cost down. One of these
Chap. 11 The Tone Generator

Tones would be used to answer ‘yes’ and the other for ‘no.’ I figured that the system could ask the proper questions, using the speech synthesizer, and I could answer either yes or no.

It occurred to me that the system really did not need to be able to recognize ‘yes’ and ‘no.’ Just ‘yes’ would be enough. A ‘no’ would simply be the absence of a ‘yes.’ I also decided that if I used a short tone sequence rather than the tone itself I would be able to recognize a ‘yes’ using the sound detection hardware covered in Chapter 9. Some software would be required, but the thought of avoiding additional decoding hardware was very appealing.

We will need one piece of additional hardware, a tone sequence generator. One of my objectives for the design of the generator was that the code could be easily changed. I might want to change my own code from time to time, and I felt certain that anyone duplicating my system would want a code of his or her own.

Let’s look at the circuit I finally arrived at. It is shown in Figure 11.1. It is made of two 555 timers acting as oscillators, an eight-switch DIP, and a 74165 shift register. The principle of operation is really rather simple. We want the speaker to produce a sequence that the computer can detect. Figure 11.2 shows a possible sequence.

There is a short burst of tone first, followed by a pause. The second burst is a little longer than the first. The length of time required for the two bursts and the pause is labeled I, J, and K. It will be the responsibility of the detecting software to measure these lengths and compare them to a predetermined sequence for ‘yes.’ Any other sequence will be taken for the word ‘no.’

The circuit of Figure 11.1 is used to produce the desired sequence. The 555 timer on the left outputs a tone from pin 3 to the speaker. This tone can be turned on or off by placing a logical 1 or 0 on pin 4. This controlling signal will come from the serial output of the eight-bit shift register. The on/off sequence of the tone will be equivalent to the one/zero pattern in the shift register. The second 555 timer is used to clock the shift register. It will run much slower than the first timer.

The actual frequencies of the two timers are not critical. The software can adapt to almost any reasonable combination we wish to use. The leftmost timer should produce a tone within the bandwidth of the phone lines. Somewhere between 1000 and 2000 hertz is ideal.

The second timer needs to be slow enough to create a recognizable sequence, and yet fast enough to complete the entire sequence within a second or so. This implies a frequency of 5 to 10 hertz. The fact that your frequencies are different from mine will just help make your system more
Figure 11.1 The DIP switches create a unique code that only your computer can understand. Using this signal you can control your home from any telephone.
secure. In addition to the frequencies, though, we will want to have control of the tone sequence.

The sequence, remember, is reflective of the pattern in the shift register. When pin 1 on the shift register is pulsed low, the shift register is loaded with the data controlled by the eight dip switches. The open switches will produce a tone. Likewise, closed switches will cause a pause.

Any pattern can be detected, but my software will expect a tone, a pause, and another tone. The length of each of these can be easily varied by the setting of the DIP switches. As we will see in later chapters, the software can be easily modified for any size of tones and pauses. It will even be easy to have another pause and a third tone if you wish.

I use four AA cells as my power supply. If you use a small speaker, the entire generator will fit easily in the palm of your hand. To use it, just hold the unit so that the speaker is close to the telephone mouthpiece and press the push button. The tone sequence, when played, will be "heard" by the computer at the other end of the phone line.

As mentioned earlier, the circuitry used to allow the computer to "listen" will be the same circuit described in Chapter 9. The circuit was used there to determine when a caller stopped talking so that the computer could hang up at the end of the message. As you remember, the circuit worked by sending the computer a logic 1 whenever the sound level exceeded a preset threshold.

This same capability is all we need for the tone detector. When the tone sequence is played over the phone, it will trigger the sound detector. The software can monitor how long each tone and pause lasts. These times can be compared to see if they are within tolerance. For these measurements to be valid, the computer must know when to start taking readings.

The computer will not continually monitor the sound detector for the presence of the tone sequence. The first time it watches for the sequence will be when the caller starts to record his or her message. If the sequence
is not the first sound heard, the system will assume it is a standard call and process it appropriately.

If the sequence is detected, though, the recorder will be turned off and the computer will communicate using the speech synthesizer. It will ask if a status report is desired. If the next sound conforms to the proper tone sequence, the computer will interpret it as a "yes" answer. Any other sound will be a "no." The most obvious sound to use for "no" is to actually say "no." Of course, any word, even "yes," would be interpreted as a negative response since it will not match the expected tone sequence.

After the status report the system will continue to ask questions and act on the answers accordingly. We will see later that the software will ask the questions in a manner that will provide easy remote control of the home.
In previous chapters several devices were connected to a computer using some form of parallel input or output port. There are many such ports on the market, but I chose to build my own. Even if you plan to purchase your ports, this chapter may be of value to you as it discusses the general software required to transfer data to and from a port.

To begin with, you need to realize that a port is nothing more than a memory location made up of discrete components. Let’s look at an example. Each memory chip in the Apple contains 16,384 memory cells. If these cells were made out of individual gates or flip-flops, you could attach wires to the flip-flop (F/F) outputs. If that wire controlled a transistor, which in turn controlled a relay, then external devices could be turned on and off by simply storing a 1 or 0 into that particular memory cell.

Using similar reasoning, external switches could be connected to the set and clear inputs of a flip-flop cell. Software could easily be used to read the state of that cell and thus determine the state of the external switch. This simple example should demonstrate that there is nothing mysterious about I/O ports.

Of course, we don’t build memories out of discrete flip-flops. Not only would such memories be very expensive and consume much more power than LSI memories, but they would also take up much more space.
In addition, all the discrete wiring would provide potential faults and make such a system less reliable.

We can build an output port using F/Fs if we wish. We would need one F/F for each data line. Whenever the 6502 executes a STORE instruction to a particular address, the address and data lines will simultaneously hold the data and the address of where those data are to be stored. All we need do is to detect or decode the proper address and use that signal to gate the data into the flip-flops at the end of the cycle.

It is equally simple to build an input port. Whenever the processor is reading from a given address, it expects to find the proper data on the data bus. If we again detect the appropriate address, we can use that signal to let the data on the bus. This enabling will usually be done using tristate buffers. When enabled, the tristates transfer their inputs to their outputs, and when disabled the outputs have an infinite impedance.

Figure 12.1 shows simple output and input ports for an Apple computer. The address decoding for the Apple is simple because 16 addresses have already been partially decoded for each slot on the mother board. Any of these addresses will activate the device-select pin on the slot. In this case we have not decoded the device-select any further. This means that any of the 16 addresses could be used when addressing either the input or the output port. A decoder chip could have been used to allow 15 more ports to be added to this one slot. The read/write line is used to determine whether we are reading from or writing to the port.

You can certainly use I/O ports like those just discussed. There are chips on the market, though, that offer several advantages over constructing the port out of discrete gates and F/Fs. These chips are called PIAs (peripheral interface adapters) and VIAs (versatile interface adapters). Since we don’t need the advanced features found in VIAs, let’s examine how to build a port using a PIA.

The 6820 PIA from Motorola has all the buffering and tristate capabilities built in. It has six registers, of which two are used as the actual input and output ports. The decoding required to select one of the six registers is unique, as we will soon see, and is also built into the chip itself.

Another advantage of the PIA is that the pins of each eight-bit port can be individually programmed to be either an output or an input. This makes it extremely flexible and adds to the usability of the ports. Figure 12.2 shows an I/O design for an Apple that will give you four 8-bit ports, any of which can be either input or output.

The 7442 is a decoder chip that aids in the selection of one of the 16
Figure 12.1 Input/output ports on an Apple are nothing more than special memory locations.
Figure 12.2 PIA chips provide an efficient and versatile means to build input/output ports.
addresses reserved for each Apple slot. Any of these addresses will activate the device select, which in turn activates the decoder.

Once enabled by the device select, the decoder uses address lines A2 and A3 to force one of its pins (1, 2, 3 or 4) to go low. These pins can be used to enable four additional devices. Each device may have up to four addresses, which should be selected with A0 and A1.

Two of the four possible devices will be PIAs. Pin 2 selects the top PIA, while pin 3 enables the bottom PIA. Each PIA requires four addresses for access to its internal registers. A0 and A1 are used for this purpose. These internal registers will be discussed momentarily.

This leaves two pins for other devices. Pin 1 is used to restart the system sanity timer. This failsafe system was explained in Chapter 7. Since we did not use the lower two address lines in this case, there will be four different addresses that will restart the sanity timer.

The last pin, pin 4, can be used for expansion if you want to experiment. It can handle any device with four addresses, so you could easily add another PIA or even a serial port, using something like a 6850 communications chip.

For full use of the 6820 PIA chips, you should consult a Motorola specification sheet. I will only discuss the necessary information required for using the PIA in the context of the home control system.

There have been many needs in previous chapters for a parallel I/O port. The system clock, for example, needed a full port and one pin from another. The interrupt system also needed an eight-bit input port. My voice output system used a serial port, but your particular choice might require a parallel port. In all these cases, you can utilize one of the PIA ports just described.

As I have mentioned, there are four addresses used for each PIA. Since I also stated there are six different registers in the PIA, you should be wondering how they all can be accessed. To explore this more efficiently, we need to realize that each PIA has two separate ports. They are referred to as port A and port B. There are two addresses and three registers used for each.

Let's examine port A for a moment. One address is for the control register. Most of the bits in this register are used to control the handshake lines. Since these lines are not used in the home control system, they will not be discussed here. There is one bit in the control register, however, that we will need to use. It allows the second address for port A to access either one of the two remaining registers.

The important bit we are dealing with is data line 2 or D2. The data
lines are labeled from D0 through D7, where D7 is the most significant bit. When D2 is set to a 1 in the control register, the second address of port A becomes the I/O register. If bit D2 is cleared in the control register, the second address provides access to the data direction register (DDR).

The DDR provides the PIA I/O port with its flexibility. Each individual bit in the DDR controls whether the corresponding bit in the I/O register is to be used as an input or an output. A 1 indicates an output pin whereas a 0 represents an input. The eight pins of the I/O port can be made all inputs, all outputs, or any combination that you require.

The software must initialize the DDR before it tries to use the I/O registers. We will see how this is done in Part III when we explore the programs that make up the home control system. You can use any commercially available I/O board as long as you follow the instructions provided for its use. You may also use an I/O port instead of the Apple game connector. This would be required if you prefer to keep your paddles connected or if you are using some computer other than an Apple.

This completes our discussion of the hardware required for implementing the home control system. Part III will cover the software that will bring the hardware to life.
We are finally to the software, the heart of the home control system. I hope you have read the previous two parts of the book first. I realize the tendency, especially for those of you that detest hardware, will be to begin your reading here. Even if you are not planning to actually computerize your home and are just reading for information, I urge you to start at the beginning.

I think my treatment of the hardware has been straightforward. I tried not to get bogged down in detailed discussions of transistors and integrated circuits. You should have found my discussions informative and easy to follow. I have tried to center the discussion around what the hardware is going to do for the software. And that is exactly why reading the book in order is important.

In this chapter I have several objectives in mind. First, I want to make sure you understand the principles of structured programming. Second, I want to show you the flow charting symbols I will be using in the chapters to follow. And, last, this chapter will serve as an overview of the total software architecture of the system. A thorough understanding of how the home control program is organized is very important if the remaining chapters are to be easily digested.

Structured programming is the best thing to happen in the computer industry in a long time. It has turned programming from an art into a science. You hear all sorts of key words and phrases in a discussion of
structured programming. Two of the most important are modular and top down design.

Actually, structured programming is really so logical that after you have been exposed to it you will probably find it very simple and obvious. The real key to building a structured program is to plan the entire program completely before any code is written. To plan completely, you must fully define the problem.

In the past two sections of the book, I have tried to define the problem of home control, or at least my approach to it. Since the problem has been broken down into its fundamental parts, it will be easy to write the program in modules called subroutines. Each subroutine will solve the problems found in each of the fundamental parts.

This principle of structured programming does not stop here though. Each subroutine should be further divided into simpler problems that will also be solved by additional subroutines. And those subroutines can be further divided, and so on. This process, starting at the top and working down, should continue until the tasks to be solved are trivial.

Figure 13.1 shows the symbols used to indicate tasks. Simple tasks are represented by just a box. If the box has vertical lines on each side, it indicates a subroutine. The subroutine boxes should be described in more detail by using a separate flow chart. A slanted box indicates an input or output operation.

The process previously described will yield a program made up of fundamental actions that are linked together to build larger and more complex actions. This linking is accomplished with logical control structures. We will call these constructs for short. Figures 13.2 through 13.5 show the primary constructs we will be using. Modules A, B, and C represent tasks and are used for example purposes only.

The IF-THEN-ELSE construct is used to decide which of two

![Figure 13.1 These symbols will be used to identify functions in the home control flow charts.](image-url)
modules, or groups of modules, is to be executed based on some condition. Many BASIC interpreters have such a statement. Applesoft only has an IF-THEN, but it can be used along with a GOTO to perform the IF-THEN-ELSE.

A special case of the IF statement occurs if we wish to do one of several things depending on the value of some variable. Figure 13.3 describes such a situation using IF statements. Figure 13.4 shows the much easier to follow construct, the CASE of X. In both of these situations, module A is performed if $X = 1$, module B is performed when $X = 2$, and module C is performed when $X = 3$.

Some of the most powerful constructs are based on a loop structure. There are three forms of loops. These are the WHILE, the UNTIL, and the ITERATE loop. Although any of the three loops can be created using only IF and GOTO statements, Apple BASIC only has ITERATE loops. Applesoft implements them using FOR/NEXT statements. To keep things as simple as possible for the reader, I have chosen to use only ITERATE loops in this book. The symbol I will use for a loop is shown in Figure 13.5. The dotted line is shown here only for clarity. Normally it will be omitted.

The architecture of the home control program is going to be top down, as I have described. At the bottom of the pyramid of tasks will be those that are very basic and fundamental. As it turns out, these fundamental tasks are very important for several reasons.

They are important primarily because they form the center core of the program. Knowing how they work will provide you with a true understanding of how larger tasks can be created by linking smaller ones.
Figure 13.3 The IF-THEN-ELSE can be used to select more than just two alternatives.

Figure 13.4 Special constructs provide diagrams that are easier to read and follow than combinations of IF-THEN-ELSE blocks.
Figure 13.5 To keep the flow charts from looking like spaghetti, loops will be drawn as shown, but without the dotted line.

together. They are also important because they provide an easy means to transfer the home control program to a system other than an Apple computer.

Examples of the fundamental tasks are dialing the phone, recognizing a spoken word, pausing for a specific period of time, and pushing a button

<table>
<thead>
<tr>
<th>LINE NOS.</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-99</td>
<td>MAIN PROGRAM</td>
</tr>
<tr>
<td>100-149</td>
<td>CHECK FOR PHONE RINGING</td>
</tr>
<tr>
<td>150-199</td>
<td>CHECK FOR SOUND CHANGE</td>
</tr>
<tr>
<td>200-249</td>
<td>DELAY 5C SECONDS</td>
</tr>
<tr>
<td>250-299</td>
<td>CHECK FOR OWNER CODE</td>
</tr>
<tr>
<td>300-374</td>
<td>SAY MESSAGE NUMER ME</td>
</tr>
<tr>
<td>375-399</td>
<td>INITIALIZE SPEECH LINK</td>
</tr>
<tr>
<td>400-449</td>
<td>DELAY BASE ON DE</td>
</tr>
<tr>
<td>450-474</td>
<td>CHECK FOR WIRELESS MIKE ON</td>
</tr>
<tr>
<td>475-499</td>
<td>LOAD TIME TABLE</td>
</tr>
<tr>
<td>500-524</td>
<td>TURN THINGS ON BY PHONE</td>
</tr>
<tr>
<td>525-599</td>
<td>TURN THINGS OFF BY PHONE</td>
</tr>
<tr>
<td>600-649</td>
<td>READ TIME</td>
</tr>
<tr>
<td>650-699</td>
<td>DIAL PHONE</td>
</tr>
<tr>
<td>700-749</td>
<td>STATUS REPORT</td>
</tr>
<tr>
<td>750-799</td>
<td>PUSH BUTTON BU ON BSR</td>
</tr>
<tr>
<td>800-849</td>
<td>RECOGNIZE VERBAL YES/NO</td>
</tr>
<tr>
<td>850-899</td>
<td>RECOGNIZE A WORD</td>
</tr>
<tr>
<td>900-949</td>
<td>MAIN INITIALIZATION</td>
</tr>
<tr>
<td>950-999</td>
<td>VARIABLE INITIALIZATION</td>
</tr>
<tr>
<td>1000-1999</td>
<td>PROCESS VOICE COMMANDS</td>
</tr>
<tr>
<td>2000-2999</td>
<td>PROCESS PHONE CALLS</td>
</tr>
<tr>
<td>3000-3999</td>
<td>PROCESS SECURITY FLAGS</td>
</tr>
<tr>
<td>4000-4999</td>
<td>PROCESS TIME TABLE</td>
</tr>
<tr>
<td>5000-5999</td>
<td>TIME TABLE SUBROUTINES</td>
</tr>
</tbody>
</table>

Figure 13.6 This table summarizes where each module in the home control program is located.
on the BSR unit. Each of these routines is very short and is explained individually in Chapter 14. If any of the major modules wishes to perform a fundamental task, it simply calls that subroutine. If your computer or peripheral operates differently than mine, you can change the appropriate fundamental subroutines and have the program working on your system with minimal effort.

Figure 13.6 shows the line numbers reserved for different areas of the home control program. The main program lies between lines 1 and 99. The fundamental subroutines occupy from 100 to 999. The major modules are each given 1000 possible lines. There is nothing special about my choices in this matter. Other line numbers could just as easily have been used.

In addition to the BASIC program, we will also need a machine language program to process interrupts and an area of memory to store the vocabulary templates for the Speech Link. Both of these situations will be discussed in detail in later chapters. At this point, however, you can refer

![Figure 13.7](image) This memory map shows where each section of the home control program resides. Note that the normal Apple LOMEM has been altered to make room for special programs and data.
to the memory map in Figure 13.7 to see the areas of memory reserved for
different parts of the home control system.

You should now have a good overview of how the software is
organized. The main program consists primarily of a few GOSUBS that
call the major modules in the proper order. These major modules use
constructs to link together the fundamental tasks in such a manner that an
intelligent system is created. In the next chapter we will look at the
fundamental tasks in detail.
A primary task in the home control system is one of the fundamental functions used to build larger modules. These simple, almost trivial actions form the foundation of the entire system. The tremendous power of the home control program comes from the intelligent linking of subroutines that perform the actions required to solve trivial tasks.

In this chapter I wish to cover all the primary tasks in considerable detail. A good understanding of these tasks and how the computer performs them will aid you greatly when we look at the major modules in later chapters. The name of each task and its starting line number were given in Figure 13.6. The actual program listings can be found in Appendix A. Let's look at each of them separately.

The first module, as its name implies, checks to see if the phone is ringing. Anytime the main program wants to know such information, it will call this subroutine. This subroutine, as well as several others, will use the variable YES to pass status information back to the calling program. In this case, when the program returns, the variable YES will equal a 1 if the phone is ringing and a 0 if it is not.

As previously mentioned, you can refer to Appendix A to see the code for this module. It begins at line 100. As you can see, it is very short, so short in fact that no flow charts will be used to describe this or any of the fundamental tasks. The subroutine is very simple. First, the phone is assumed not to be ringing, so YES is set to 0. A FOR loop causes the
system to check a location of the Micromodem card 30 times. The variable RING has been set equal to the actual address of this location.

This Micromodem location is special. It was designed to contain a number smaller than 128 whenever the phone is ringing. The program checks for such a situation and sets YES equal to 1 when it exists.

At this point, I hope you are thinking, “Boy, he wasn’t kidding when he said the fundamental tasks would be simple.” Remember, if you happen to be using a different commercial device for detecting a ring, you will need to change this routine. Once that is done, any module that needs to know if the phone is ringing will GOSUB 100 and check the value of the variable YES upon return.

If you do not wish to buy special hardware, you should refer to back issues of such magazines as *Byte*, *Microcomputing*, and *Micro*. These magazines are filled with many interesting and informative articles that can aid those of you that wish to design and construct circuits of your own. Of course, my hardware chapters have given you some ideas, but I suspect many will wish to alter my initial ideas to match particular objectives.

You should also check the book section of your local computer store. There are many books available on subjects ranging from telephone projects to interfacing the Apple computer. I point all this out because you should now realize how simple it is to adapt my program to any hardware you might have or want to build.

For completeness, I should discuss the magic number 128. Many peripheral input pins, such as the location RING and the Apple game connector input pins, are each connected only to the most significant bit of an input port. The remaining 7 pins are left unconnected and therefore appear to be random 1’s and 0’s when read. Any external event forcing the most significant bit to 0 will appear, when read, as a number smaller than 128, no matter what state the rest of the bits are in.

The second fundamental task is to check for a changing sound from the microphone and its associated amplifier discussed in Chapter 9. This subroutine also passes its answer in the variable YES, like the ring detection module. It is not quite as easy though to detect a change as it is a simple presence.

The problem is as follows. Usually, when no one is talking, the input pin will be zero. For example, if someone called and just quit talking, the input pin would stay low. If they hung up, though, the dial tone would be a continuous sound that would result in a 1 on the input pin. To ensure that someone is still talking, the computer must check to see if the input pin is alternating between 1 and 0.

To do this, the subroutine starts by assuming there is no change,
which again means that YES is set to 0. The location SOUND has been set to
the address of one game connector input pin. Depending on whether
SOUND contains a number greater or less than 127, the temporary
variable S(6) is set to a 1 or a 0. For a couple of seconds the system loops
and checks to see if the location SOUND still contains the same infor-
mation. If it does not, a change has occurred and the variable YES is set to 1.

This subroutine does not return as soon as it detects a sound, but
always checks for the same period of time. This will provide an easy
means to monitor how long a call is taking. Using this information, the
phone module (Chapter 17) can easily impose a maximum limit on the
length of recorded messages. The listing for this fundamental task, as for
all the others, is in Appendix A.

There are two delay routines, one at 200 and another at 400. Let’s
look at both of them now. The one at 200 will delay a number of seconds
equal to the contents of the variable SC. It performs this task by setting up
the variable DE and calling the general-purpose delay routine at 400.

You might be wondering why I did not use the clock for these
routines. For many situations the clock would be excellent. Remember,
however, that my clock is only accurate to 1 second. The delay at 400 is
accurate to one cycle of the FOR loop, about 1/400 of a second. This
accuracy was needed to time the 30-second message tape. If the tape
stopped a half-second early each time, it would not take many phone calls
to have it starting up in the middle of the message.

There is one line that I did not mention in the explanation. It PEEKs
at the location SANITY. This statement will occur throughout the home
control system. Its purpose is to reset the system sanity timer as discussed
in Chapter 7.

At line 250 we have a subroutine that checks for the special tone
sequence described in Chapter 11. The principle of its operation is not very
difficult. The program monitors the same microphone as the subroutine at
line 150, which detects sound changes. The difference is that this module
will measure how long the first sound, the first pause, and finally the
second sound last. The relative lengths of these three periods are stored in
the variables I, J, and K. This is done in lines 262 through 286.

At line 292 the measured values are checked to see if they are within
tolerance of the code values. If they are, then YES is set to 1; otherwise it
is set to 0. The lines prior to 262 are used to wait for a sound to start. If it
does not start within a few seconds, the subroutine will return with YES
equal to 0.

The PRINT statement at line 293 is not required for normal operation.
If you are changing your code or starting up your system for the first time,
you may use this line to help you decide on your ranges for I, J, and K. The easiest way to do this is to simply call your computer while you are out running an errand and play your tone sequence. Since it will not match, the computer will hang up on you; but when you get home, it will have printed the values it found on the screen and you can alter lines 294 through 298.

The Say a Message module at line 300 may appear to be long at first, but the program itself is really very short. Most of the listing is made up of the messages. To use the message module, you need only to set the variable MESSAGE equal to a number between 1 and 64 and call this subroutine. Not only will the corresponding message be said, but the delay routine will be called to force a pause so that the message and the program will stay in sync.

This need to stay in sync may not be very obvious, so let me discuss it further. Suppose the computer asks you a question such as, "What do you wish me to do to the floodlights?" You would be expected to respond by saying either "Turn it on" or "Shut it off."

Once the computer has sent the message to the speech synthesizer, it will then enable the speech recognition unit in order to hear your reply. If the system begins "listening" too early, it will hear itself talking. This is possible because of the buffer in the voice synthesizer. If the wait is too long, you might begin speaking before the system is ready to listen. The solution to this problem is to have the computer pause whenever it speaks for just the right amount of time.

Line 303 is used to enable the speech synthesizer. Depending on the variable MESSAGE, one of the print statements will send a phrase to the synthesizer. If MESSAGE is 1, line 311 will be executed. If MESSAGE is 2, line 312 will be spoken. Each print statement must end with a RETURN in order to send the program flow back to line 306.

In line 306 the variable SC is set to an element in the array MT. The variable MESSAGE is used to select which element is to be used. Each element in the array has been initialized to equal the number of seconds required to say each corresponding message. When the subroutine at 200 is called, a pause of SC seconds will be generated. Note that the delay routine we are using is also one of the fundamental tasks.

By reading through the messages, you can see the phrases that my house can say. You can easily customize these to give your home the personality you desire. If you wish, you can add additional messages in lines 365 through 374. If you need more than ten additional messages to customize your home, you will have to expand line 305.

The last thing done in this module is to turn off the speech synthesizer using a PR#0 command.
The next task is to initialize Speech Link. The Speech Link voice recognition system needs to have memory set aside for work space. It also needs to know where the word templates are stored. Once all this has been done, a control “I” must be sent to the slot holding the Speech Link board.

If you own a Speech Link, you can get a full description of what is happening in this module from your manual. If you are using some other recognition system, I will not bother you with the details about mine.

The subroutine at line 450 checks to see if the wireless mike is on. It looks at the Apple game connector pin that is connected to the wireless mike (see Chapter 4) and sets the variable YES equal to 1 or 0 depending on the value found. Whenever the wireless mike is off, the RF radiation from my early model Apple keeps the input pin high, which means that the reading will be 128 or greater. If the mike is turned on, the receiver locks onto the carrier, and the reading will drop to 127 or lower. Because of this, I can get the computer’s attention by just turning on the microphone.

If you are using a later model Apple or any computer that is well shielded, or if you have a different type of wireless mike than mine, you may have to alter this routine somewhat, although I doubt it. If you should need to make changes, you will just have to experiment a little with your equipment. In general, though, if my subroutine is not suitable, the Check For Sound Change routine at line 150 should work if the variable SOUND is replaced with the variable VOICE. Copy it at line 450 and make the appropriate changes.

Don’t forget that the sensitivity of the hardware discussed in Chapter 4 can be varied using a potentiometer.

At line 475 there is a subroutine to perform another fundamental task. In this case it loads a two-dimensional array from disk. This array contains the time table information that the main program will use to control the automatic actions preprogrammed by you. A complete discussion of the contents of this array will come later. The listing for this subroutine is relatively short. The variable D$ is equal to a control “D” as required by the Apple for recognition of disk commands.

We need to skip ahead just a little in the program listing to the module at line 750. This routine utilizes a machine language program (details in Chapter 20) to “push” a button on the BSR unit. To use this subroutine, the variable BU should be set to the appropriate code for the button desired. The codes were listed in Figure 8.4.

The subroutines at 500 and 525 are used to turn things ON and OFF using the telephone. They both use other fundamental tasks to help
perform their functions. These subroutines are only called if the main program has determined that something is to be controlled. Let’s look first at the Turn On By Phone module.

The program immediately goes to the subroutine at 540. Using the Say Message task, the computer will say, “Give me a yes to each valid device” (message 38). A FOR loop is then used to say the seven items that my program can control from a telephone. After each is said, the caller can respond by saying the word “no” or playing the proper tone sequence for a “yes.” The subroutine at 250 is used to check for the code. If a YES was given (YES = 1), the correct button on the BSR will be pressed.

When all seven items have been said, a RETURN sends the program back to line 510. At this point the ON button is pressed, which will turn on every item whose button was pressed earlier. The only difference in the OFF subroutine is that the OFF button is pressed in line 534. The power of using fundamental tasks to create more powerful functions should be becoming very clear. If it is not clear at this point, I urge you to study the listings in Appendix A for the modules discussed so far.

At line 600 there is a routine that will read the time from the clock circuit discussed in Chapter 6. This time is stored in two ways. The first way uses the array T(X), where X ranges from 0 to 6. When this subroutine is exited, T(1) and T(0) will contain the present time in seconds. T(0) contains the least significant digit. Likewise, T(3) and T(2) contain the minutes information. The hours are found in T(5) and T(4). T(6) will contain a number from 0 to 6 indicating the day of the week.

Each of these digits is read using the subroutine at line 620. It sets up the proper data direction on the I/O port, sends the proper address, and reads in the digit desired. During the read operation the clock is disabled from counting so that false reads will not occur. The variables used here are A, B, CA, and CB, which represent data registers A and B and their associated control registers.

A small machine language subroutine (Appendix B) is called from line 630. Its purpose is to mask the top four bits of the digit being read to 0.

Line 616 converts the present time in minutes and hours to a single number, which is stored in the variable TM. Since my clock chip adds 4 to the most significant digit of hours if the time is P.M., each time will have a unique TM. If you are using a different clock board, simply replace this routine with one that reads your clock and puts the appropriate data into the variables T(X) and TM. If you do write your own routine, don’t forget to indicate A.M. and P.M. by adding 4 to the most significant hours digit to indicate P.M.
The Dial the Phone module does exactly as its name implies. Before calling this routine, PN$ should be set equal to the desired phone number. Line 655 activates the D.C. Hayes Micromodem, which is in slot MSLOT. The phone is actually dialed by sending a control Q, the phone number, and a line feed to the modem. When the modem receives this sequence, it will dial and attempt to communicate with a computer at the other end of the line.

Since we don’t want such a communication to take place, the modem must be instructed to hang up by sending it a control Z. This would normally disconnect us from the phone line. To solve this problem, we activate the intercom phone in line 665 before the control Z is sent. The variable TN is the game connector pin controlling the phone. TN stands for telephone on. Location TF turns the phone off.

After the modem is completely turned off with a PR#0, the variable S5 is set to a 1. This serves as a flag to other modules so they will know that the phone is off the hook.

At line 700 there is a module to verbally report the present status of the house. Variables S8 and S9 keep track of the number of outside and inside triggers of the security sensors. This module uses the Message module to notify you of any security problems if S8 or S9 are not equal to 0. It also uses the variable NC (number of calls) to let you know how many phone calls have occurred that day.

Three modules are associated with the Speech Link voice recognition unit. The initialization routine at line 375 was discussed earlier. The other two are at lines 800 and 850. Let’s look at 850 first.

This routine is highly dependent on the Speech Link. If you own one, the code should be obvious. If not, you will have to create a similar program to perform the same functions. Lines 858 to 865 cause the Speech Link to wait until a word is spoken. If the word is recognized, it is stored in the variable W$. If no match is found, W$ will be an empty or null string.

If recognition does not occur, the computer will respond (line 870) by asking you to repeat your request. The rest of this routine is a little peculiar and will require a little more information about the Speech Link system.

The Speech Link not only puts the recognized word in W$, but it also puts the number of the word recognized in a location we can PEEK at. Since I said each word three times during training, I needed to map three different word numbers into the desired number. This is accomplished by the formula in line 875. Essentially, if the peeked location contains a 0, 1,
or 2, the word number (WN) will be 1. If it is 3, 4, or 5, then WN will be 2. This continues for all the possible words.

You could set WN using IF-THEN statements, but this feature of the Speech Link is much more efficient. A second feature of the Speech Link is selective recognition. This means that the word to be recognized can be compared to less than all the words in the vocabulary. This is not always a useful feature. In many cases, however, an intelligent program can determine that the next input should be one of several particular words.

For example, if the command given to the house is a room name, like kitchen, the system will ask, "What do you wish me to do to the kitchen?" It will expect you to give only one of the following four commands.

Turn it on
Shut it off
Dim it
Nevermind

The first three obviously control the light. Notice that I have tried to make each command sound as different as possible. The last command is used to get out of the sequence. It is used primarily to cancel a misunderstood command. Suppose you said "kitchen" and the house replied, "What do you wish me to do to the bedroom," because it misunderstood the word "kitchen." By replying "nevermind" the command will be aborted.

Since only four different words are expected, the system can be much more accurate if it compares a spoken word to only the four possible and not to all 63 in the vocabulary. To select which words to use during the recognition process, the Speech Link has a table that starts at the location specified by the variable MASK.

Lines 880 to 890 set each element of the table equal to zero so that the corresponding word will be ignored during the recognition process. Any module wanting to recognize a word will be required to set up the MASK table for the words it expects before calling this module. Since the home control program is intelligent, it will always know what type of command or response it expects from you.

I should remind you that this chapter makes no attempt at explaining how actions such as the preceding one are accomplished. The intelligence of the system will be explained in detail in the chapters that follow. All you are expected to understand now is what each fundamental task does.
The last task associated with the Speech Link resides at line 800. It is very similar to the word recognition module, except a control V is sent to the Speech Link before and after a word is recognized. The control V is used to toggle the speech system into and out of a special form of recognition. In this mode it can recognize only two words, "yes" and "no."

As in other modules with such a response, the variable YES will be set to a 1 for "yes" and a 0 for "no." This mode is particularly nice because it is speaker independent and does not require training.

At line 950 there is a module that initializes all the major variables. Refer to the listing in Appendix A as I discuss the use for each variable.

In lines 955 to 960, PF, PN, RF, RN, TF, and TN stand for tape PLAYER off/on, tape RECORDER off/on, and PHONE off/on. Any access of these variables, such as a PEEK or a POKE, will perform the named operation. Refer to the appropriate hardware chapters for more information on the use of these game connector locations.

The variable VOICE is equal to the location that indicates whether the wireless mike is off or on. Similarly, SOUND is a location that indicates if a noise exceeds the preset threshold. Lines 965 to 967 establish which slots the Speech Link, the Voice Synthesizer, and the Modem are in. RING is the Micromodem location that indicates if the phone is ringing.

At line 969 the array SP contains the Security Phone numbers. These numbers will be called in case of an emergency. Lines 970 to 971 set up the addresses for the PIA ports, A and B, and their respective control registers. The next line sets SANITY equal to the address of the system sanity timer. There will be accesses of the sanity timer throughout the program so that it will not cause a reset.

The array MT stands for message time. It is initialized to the number of seconds required to say each message in the lines 973 to 975. Next, N1$ and N2$ are set to the names of the two people for whom you have set up vocabularies. The respective vocabularies must be saved on disk under the same names.

Lines 979 to 980 set up the array DAY$ as the days of the week. PN$ in line 981 holds the phone numbers you can ask the computer to call. If you have single-digit dialing on your phone, you may increase the dialing speed by using a one-digit code.

Lines 984 to 986 establish the variables TEEN$ and TN$. These will be used to allow the Speech Synthesizer to announce the time correctly. Lines 987 to 992 set up the addresses for the tables, work space, and control characters used by the Speech Link.
Line 993 dimensions the time table array TT. This table will contain the events that should occur automatically, as well as their scheduled times. The appropriate codes mentioned in Chapter 8 for the BSR buttons are put in the array BU at lines 995 to 996.

The control codes used by the Modem are established in line 997. Line 998 makes sure that the phone and the two tapes are all off.

This completes the fundamental tasks required to control the hardware for the computerized home. All that is left to do is to see how these tasks can be linked together to create an intelligent system.
In this and the following five chapters I am going to discuss the major modules that actually form the home control program. As you recall from earlier chapters, these modules are initialization, voice request, phone control, security, event timing, and monitor movement. The initialization module is run only once and then control is passed sequentially among each of the other modules until conditions indicate that a particular function is needed.

These chapters will be presented differently than Chapter 14, which covered the fundamental tasks. Each of the tasks was small and, although important, they were lacking in overall complexity. The tasks are also the most probable sections of the program to be rewritten if your system is not identical to mine. For these reasons, the fundamental tasks were discussed in the context of their actual code.

The major modules are much larger and more complex. Rather than exploring their code, I will discuss them in terms of flow charts. Even though they are more complex, you should not hesitate to examine them. As you will see, their increased complexity does not necessarily mean that they must appear more complicated. Remember, these modules are little more than a series of control structures that link the fundamental tasks together in such a manner that a larger goal is accomplished.

Figure 15.1 shows the flow chart for the initialization module. The
Figure 15.1 The initialization routine makes appropriate choices, even if no one is home.
The first two items are calls to fundamental tasks to initialize the variables and the Speech Link hardware. Next the module loads the machine language portions of the home control program, including the vocabulary templates for the Speech Link. The final disk transfer is used to load the appropriate time table (depending on what day it is).

Except for enabling interrupts, the rest of this module is used to initialize some very special variables that are used as flags or status indicators. The contents of these flags provide a means for the system to remember important status information. The primary status variables will be $S_1$, $S_2$, ..., $S_9$. I will define each of them as they are initialized in the flow chart.

$S_1$ can be one of three values. If it is a 2, it means someone is home. A 3 means the computer thinks the house is empty and a 1 means that the status is unknown. At this point in the initialization, $S_1$ is set to 1, thus making the assumption that no one is home.

The variable $S_2$ can only be a 1 or a 0. A 0 means that everyone is asleep, while a 1 indicates they are awake. The initial assumption is that they are asleep.

Both $S_3$ and $S_4$ are alarm indicators. $S_3$ is for alarm clock or wake-up calls, while $S_4$ refers to the security or burglar alarm. In both cases a 0 means the alarm is on, whereas a 1 indicates off.

$S_5$ is a 0 whenever the phone is hung up and a 1 if it is in use. $S_8$ and $S_9$ keep track of how many outside and inside security breaches have occurred while the house was empty. A special variable NC is used to keep track of the number of phone calls. $S_6$ and $S_7$ are reserved for temporary variables and do not have any global significance.

Once the computer makes the initial assumptions and sets the flags accordingly, it then attempts to confirm or change as many of the assumptions as possible. It will do this primarily by asking verbally if anyone is home. To prevent the system from waking you in the middle of the night just to confirm its decision, I added an additional check.

If the clock shows that people should be sleeping, the computer will not make the announcement. Instead, it will alter only the $S_1$ flag to show that the status is unknown. If the clock indicates that people should not be sleeping, it proceeds to ask if anyone is there.

If someone responds to the question by turning on the wireless mike, four of the flags are changed. The house will now assume someone is home, that they are awake, and that no alarms have been set.

The last portion of this module POKES the security alarm value of $S_4$
into a memory location called ALARM. This location is used as a flag to the machine language portion of the home control system and causes it to act differently depending on the value. The actions of the machine language program will be discussed in Chapter 20.

You can refer to Appendix A to see the actual code used to implement this and the other major modules of the home control program. The initialization module begins at line 900.
The purpose of the voice request module is to provide a means to control the house using verbal commands. Because of its complexity, much of this module is broken down into submodules, which are in turn made up of primary tasks.

Figure 16.1 shows the first-level flow chart for the main module. Let's examine the chart in detail. You can refer to Appendix A for a complete listing of the program if you wish to see how the flow chart is implemented in code.

The voice request module begins by using a fundamental task to determine if the wireless microphone is on or off. If the mike is off, the control returns to the main program loop. If the mike is on, it means that someone wants to verbally command the house to do something.

Assuming someone turned the microphone on, the flags are set to show that someone is home and that they are awake. The security alarm is shut off, as is the wake-up call function. These operations are representative of how the home control program is intelligent. The user does not have to “tell” the computer that he is home or that she is no longer in bed. These conclusions are “inferred” by the system.

If, for example, you wake up early and ask the computer to turn on the bedroom light, it will automatically, because of the flags, act differently. It will not try to wake you up later as it was scheduled to do. It won’t
Figure 16.1 This section of the voice control module passes control to one of seven submodules.
mind if inside security sensors are triggered because it will now assume it is you and not an intruder.

The next item in the flow chart is to check to see if the phone is off the hook. This needs a little explanation before we look further at the flow chart. Later in this chapter and in Chapter 17 we will find that you can command the computer to dial or answer the phone for you and place the call on the intercom phone. Once that is done the program will set a flag to indicate that the phone is off the hook, and it will proceed to handle other tasks.

The assumption will be that if you are talking on the phone you will not be giving verbal commands to the house. Consequently, whenever the phone is off the hook and the computer sees the microphone on, it will assume that the phone call is over and it will terminate the call. This may seem complicated now, but it will seem simpler after we have discussed it in more detail later.

For now, we can see from the flow chart that if the microphone is turned on while the phone is in use, then the computer will hang up, say it has hung up, and set the flag to indicate that the phone is no longer in use.

At this point in the flow chart we can assume that someone really wants to give the computer a verbal command. The command must be one of the 13 that is expected, so the mask is set to only allow recognition of the proper words. A fundamental task is called to do the actual recognition. This task, as discussed in Chapter 14, will set the variable WN (word number) equal to the appropriate number between 1 and 13.

Depending on WN, control is passed to one of seven secondary or submodules that process the command given. We will look at each of these modules in a moment. For the sake of continuity, though, let’s complete our discussion of Figure 16.1. After the appropriate secondary module performs its task, the flow proceeds down the flow chart.

If a flag shows the phone is off the hook, which would indicate that the system has just been requested verbally to make a phone call, then the program returns to continue processing other tasks. Otherwise, the computer asks if there is anything else you want and waits for a verbal yes or no. If YES, it says that it is ready and waits for a new command. If NO, the system will say goodbye and return to loop until another situation requires attention.

You should note that the computer always responds verbally to the commands you give. When a human talks to a machine, some type of feedback to the user is very important in order to convey that the machine is understanding.

Let’s look at each of the seven secondary modules that execute the
verbal commands. The first of these is the control module. It is called when the word spoken was one of the items that can be controlled, such as office, stereo, or bedroom. The flow chart for this module is shown in Figure 16.2.

In the first block of the flow chart we can see that the variable BU is set to the correct button code for the BSR unit. The button codes were stored in the BU array during initialization. The word number WN is used to select the correct code.

The computer uses a fundamental task to ask what you wish to do to the item specified. Since the name of the item being controlled is stored in the variable W$, the house can call the item by name. The mask is then prepared to recognize only one of four words. The acceptable words are "turn it on," "shut it off," "dim it," and "nevermind." Another fundamental task is called to determine what you responded with. I should point out that the words "on" and "off" were not used alone because the recognition system had trouble telling them apart.

This is a good time to reemphasize the importance of the fundamental task subroutines. They handle all the drudgery associated with the task. In this case, for example, if the human response is not recognized, the task subroutine will ask you to repeat the answer. This verbal response from the computer is handled by still another task, which also takes care of all the little details associated with its job. With the tasks taking care of all the details, the major modules become not only simple to understand but also very easy to write.

Let's continue with the flow chart. If the user's response was "nevermind," the program simply returns to the main module. Otherwise, we can be sure that WN is 14, 15, or 16, because those were the only words that were acceptable. Since the item in question is going to be turned ON, OFF, or DIMMED, a fundamental task first "presses" a button to select the item.

Control is then passed to one of three sections of the module to press the ON or OFF button or in the case of DIM to press the button 50 times. This completes the control module.

If the verbal command is "Time Please," the time of day submodule is called. Figure 16.3 shows the organization of this routine. It first calls the read time task in order to set the array T to the individual digits.

These digits are analyzed, converted to their English equivalents, and announced verbally by the speech synthesizer. You can refer to the flow chart to see how this is accomplished. The reason for it should be obvious. Suppose the time is 10945. If the synthesizer read those numbers, you would have to translate them yourself.
Figure 16.2  This submodule allows you to control lights using verbal commands.
Figure 16.3 The numeric time from the clock must be converted to its string equivalent so that the voice synthesizer can properly announce it.
The program will convert 10945 into "Monday 9 forty five A.M." This string of characters will be used by the voice output task to announce the time.

If the input word was "telephone," control is passed to the submodule shown in Figure 16.4. The computer begins by asking who you wish to call. Since there are only five acceptable answers, four names or places plus "nevermind," the mask is set to allow only words with numbers 17 to 21.

The task to recognize a word is called again, and if the answer is not "nevermind," then message task states that the phone call is being made. The number is looked up in the array PN$ and then dialed by a

![Figure 16.4](image-url) Using this module, you can ask the computer to call someone for you.
fundamental task. Finally, a flag is set to convey the information that the phone is off hook to other modules.

The *conversation mode* is entered in response to the command "are you busy?" This module simulates a conversation between you and the house. All the questions asked by the computer expect a YES or NO answer, or some short phrase that is ignored. There is no real intelligence displayed here except that the computer’s response will be appropriate for YES or NO answers. This mode was added more for fun than to serve a useful purpose. If you don’t want such a mode, you can easily add another command or extra phone numbers for verbal calling. I think if you try it though you might find it has some value.

I find it most useful when someone visits that has heard about my "computerized home" and yet knows nothing about computers. They seldom are impressed when it dials the phone or turns on lights, but they never cease to be amazed when it appears to be conversing with me.

This module is actually very simple as it is made up only of a couple of fundamental tasks. Refer to Figure 16.5 to see how the conversation is created.

The next submodule allows you to change vocabularies while the program is running. Since the Speech Link must be trained for each user, I placed two vocabularies on the disk. In my case, one of these is saved under the name JOHN and the other as WANDA. Chapter 21 discusses some utility programs, one of which can create these vocabulary files.

A couple of techniques are used here that make the house seem intelligent. Initially, the variables N1$ and N2$ are set equal to JOHN and WANDA, respectively. Whenever the house is talking, it refers to the person as N1$. For example, it might say, "What can I do for you "N1$"?" This makes the comments much more personal.

When training my vocabulary, I said all my commands but one. I had Wanda say the phrase, "This is Wanda." Likewise, in her vocabulary I said the phrase "This is John." If when the computer says, "What can I do for you "N1$"?" the variable N1$ is JOHN, then Wanda can respond with "This is Wanda." The recognition of that phrase as a command word will send control to the *change vocabulary* submodule shown in Figure 16.6.

As you can see from the flow chart, the computer will apologize to Wanda (or the alternate person) and then load her vocabulary. Afterward the contents of N1$ and N2$ are exchanged so that everything works equally well for either person.

The next command is "goodnight." When this command is issued,
"I KNOW HOW YOU FEEL"

"THAT'S GREAT"

"I'M SORRY"

"DID SOMETHING SPECIAL HAPPEN"

"DID YOU HAVE A GOOD DAY"

"COULD WE GET A COMPUTER"

"WHAT WAS IT"

"THAT'S INTERESTING"

"PLEASE"

"GOT TO GO NOW"

"GREAT"

"THAT'S INTERESTING"

"YES/NO"

"YES/NO"

"YES/NO"

"YES/NO"

"YES/NO"

"YES/NO"

RETURN

The simulated conversation provided by this module can really impress a nontechnical user.
Figure 16.6 My system will allow two separate vocabularies to alternate automatically, depending on who is talking to the computer.

Figure 16.7 Telling the house "goodnight" tells the system to adjust the status of the alarms.
the system will assume you are going to bed. As you can see from Figure 16.7, it will first tell you that it will watch the house for you. This means that the security system is enabled.

The flags are set to indicate that you are asleep and the security alarm is on. The machine language program that handles security (Chapter 20) is notified of the change by poking ALARM with the value of S4. By

![Flowchart](image)

**Figure 16.8** When you leave, the house will set flags so that it will answer the phone and handle security.
enabling the interrupts we are insured that the machine language program is operational.

The system will ask if you wish to be awakened in the morning. Depending on your answer, the wake-up flag is altered and the appropriate verbal response is given.

The computer will continue to remember that you are asleep and the status of the wake-up alarm until one of two things happens. If the house was instructed to wake you, it will verbally try to arouse you at the correct time. This will be discussed in detail in Chapter 19. At that time the system will assume that you are awake and adjust all the appropriate flags.

The second way that the flags can be altered is for someone to turn on the wireless mike and try to issue a command. As mentioned earlier, such an action will cause the house to assume that you have gotten up early.

The last verbal command is “bye-bye.” I used this instead of “good-bye” because it was easier for the system to tell it apart from “good-night.” Figure 16.8 shows the flow chart for its implementation. It begins by asking you if you are leaving. If your answer is “no,” it turns off the security alarm and returns.

If you said “yes” you were leaving, then it sets up a flag to indicate that no one is home. This flag will be used primarily by the telephone answering module and the automatic event scheduler. These modules will be discussed in detail in later chapters.

The house will then ask if you want the security alarm on. If not, it announces that it will handle phone calls, sets the appropriate flags, and returns. If you answered “yes,” it acknowledges your request and waits for 45 seconds so that you can get out of the house before it turns on the alarm.

This concludes the discussion on the modules that process the verbal commands. Although you should now understand how the commands are carried out, I expect that some of the interactions this module has with other modules might still be a little confusing. I suggest that you read on and get the details on the other major modules first and then return to this chapter if you are still unsure of some details.
We saw in the last chapter that you could verbally command the phone to place a call for you. In this chapter we are going to examine the module that handles incoming phone calls.

Figure 17.1 shows how this module begins. A major task is used to determine if the phone is ringing. If not, the program returns and continues the major loop. If the phone is ringing, two things happen.

First, a variable that keeps track of the number of phone calls is incremented. This variable will be used later in the status report module to tell you how many calls have been received.

The system must now decide what to do about the call. This depends on the status flag S1. Depending on its value, control will be passed to one of three sections of this module. These three sections handle three different situations, which are “someone home,” “no one home,” and “unsure if anyone is home.” This last situation could occur only if there had been a power failure and the system had to be reinitialized.

Let’s look at each of these situations individually. I will start with section A, which is shown in Figure 17.2. It handles the phone when no one is home, which means it must function like an answering machine. It begins by turning on the intercom phone and the tape player. This is done by POKEing the appropriate game connector addresses. The system then waits in a loop for just enough time for the endless loop message tape to
Chap. 17 The Phone Control Module

Figure 17.1 When the phone rings, the phone may be handled in one of three ways.

finish. My timing loop is based on my tapes, which may be different from yours. Experiment with the delay a little to get it just right for your system.

At the time when the tape should be finished, the player is turned off and the recorder is turned on. Immediately, the system begins watching for the special code that indicates that you are calling. If it does not get the proper code as the first sound, it records the caller’s message and turns everything off. Although a maximum time is allotted for each message, the system will hang up early if there are no changes from the sound sensor.

If it was determined that it was you calling, the system reacts by turning off the recorder and asking if you want a status report. Again it checks for the special code. If it does not occur, the status report task is not called.

It then will ask you if you want to turn something on. If you give the code for YES, a task module is called to let you select the items to be turned on. Similarly, you are given the choice to turn off items in the house. When all this is complete, the house will say “good-bye” and hang up.

Section B of this module is shown in Figure 17.3. It handles calls when you are at home. It begins by asking you if you would like for it to get
Figure 17.2 If no one is home, the house becomes an answering machine that can also accept commands from you.
the phone for you. The system will wait about 3 seconds for your reply, which in this case is simply turning on the wireless microphone.

If you fail to turn it on in the prescribed time, the house will assume you answered the phone yourself and will tend to other tasks. If the mike did come on, the system will turn on the intercom phone so that you may talk with the caller. A flag is set so that all modules will know the phone is presently turned on.

Figure 17.4 shows how section C handles a call if the system is unsure if anyone is home. It waits for several seconds to give you a chance to answer the phone. If it is still ringing, it assumes that no one is home and proceeds to section A to handle the phone call in the proper manner.

Even though it assumes you are not home if the phone is still ringing, it will not assume the opposite if the ringing has stopped. The reason for this is that it is possible that the caller just hung up. This means that, although the house can assume incorrectly at times, I have tried to make the default assumptions the best choice.

In this case, for example, an assumption of you being home when you are not would cause the burglar alarm to be deactivated. Since I find this inexcusable, my system will not allow it. The opposite case, however, assuming you are away when you are really home, is acceptable. If you do happen to be home and you set off the alarm, you can easily deactivate it.
Remember that this type of assumption is not an everyday occurrence. It can only happen if there is a power failure at night and you receive a phone call before morning that hangs up early. If the power failure happened during the day, the system would have verbally inquired to see for sure if anyone was home. After all, your house should be polite and not wake you in the middle of the night just to see if you are home—even if that means a potentially incorrect assumption now and then.
Many of the capabilities of the home control system can be considered a bit frivolous. After all, how many people do you know with a house that talks to them? The security aspect of the system, though, is very practical. If you price a full-featured commercial security system for your home, you may find the computerized home to have some attractive economical advantages, especially when you consider all its other capabilities.

I don’t wish to imply that the security program offered here will provide total security. With custom changes in the program, however, its capabilities can be increased to almost any degree that you wish.

You might be asking yourself why I did not publish the “perfect” security module. The answer is really very simple. First, as you see from the module given, I think the best security must be designed to function in a particular environment. The best for my home would probably not be the best for yours.

The second and probably the most important reason for not publishing the perfect security module is that, if you can read it to see how it works, then an intruder can also read it to see how to overcome it. My intention will be to show you a workable security system. Using it as a guide, I hope you will customize it to meet your particular needs.

In my system there will be several groups of sensors that can indicate a security problem. The doors and windows should have some kind of
magnetic or contact switches. Refer to Figure 18.1 to see how these switches may be connected to an input port on the computer.

Most input ports will "float" high, which means they will read as a 1 if nothing is connected to them. If this happens not to be true for your ports, you can force it to happen by using a 5K pull-up resistor as shown. Such a resistor is always a good idea, even for ports that float high.

Since the port will normally be high, any closure of a switch will short it to ground and cause a 0. Sensory switches may be purchased in normally open or normally closed varieties. This means that you can have a 0 indicate that the door is open or closed, depending on your choice of switches.

You can add an inverter between the switch and the port if you cannot find the type of switch that you prefer. In all my examples, a 0 on the port will indicate a security problem. You should also notice from the figure that you do not need a separate input pin for each switch. By paralleling all the window switches, for example, any window opening will force a 0 at that pin.

Another type of sensor should be used to indicate outside movement. This could be an ultrasonic motion detector or an infrared beam that triggers if broken. There are even sophisticated sensors on the market that can detect the body heat of someone near your home. The type of sensor or sensors you choose will depend on your home and its surroundings. I will assume that the appropriate external sensors have been paralleled so that we have a single pin indicating external movement.

Similar types of sensors could also be used inside the house to

![Figure 18.1](image-url) The security switches may be combined in many combinations as required.
Figure 18.2 If the security flag has been set, the indicator flags will be checked and an appropriate action taken. (a) When you come home, you are asked if you want a status report, which summarizes the day's activity, including such things as the number of phone calls received. (b) An inside problem requires drastic measures. The computer will call you and your neighbors and inform them of the situation. (c) In response to an outside problem, the house will attempt to scare off an intruder by turning on lights and playing the radio.

indicate internal movement. Since I have floor switches already installed, I decided to let them act as the internal sensors when the security alarm was on.

All the security port inputs will be ORed together so that any of them can trigger an interrupt. This will run a machine language program that will set up several memory locations to indicate whether there is a security problem and, if so, if it was a door or window or if the problem was internal or external.

The details of the machine language program and the interrupt circuitry will be discussed in Chapter 20. For now, let's just assume we can make such determinations. The flow chart in Figure 18.2 shows the security module.
A **PEEK** at the proper location will first determine if any security problems have occurred. If not, the program returns to its normal state of looping. If a problem is indicated, another location is checked to see if it is a door opening. Since my doors are heavily deadbolted, the house will assume that a door opening means that I have come home.

The house will react to my presence by turning on a light and asking if it is me. If the wireless mike is not turned on within 10 seconds, it will assume that I am an intruder. More on this in a moment. If the mike is turned on, the house asks if I want a status report. Depending on my answer, either one is given or the system says good-bye. Finally, all the flags are set to appropriate conditions. This includes such things as someone being home, that there is one person in the den, which is where I normally enter the house, and that the alarm is now off.
Figure 18-2C

Backing up the flow chart, we can see what would have happened if the security problem had not been a door. The system will then check to see if outside movement caused the indication. If so, it turns on various lights and the radio in the hope that an intruder will assume someone is home and leave.

In this example, the same thing is done if the intrusion is in the daytime or at night, if you are home or if you are away. I figured the radio would work either day or night and that if I was home I would certainly be warned of an external intruder by all the lights flashing, even if I was asleep.

You could easily modify this module so that special sequences could occur for different situations. Remember, there are task modules that can
easily turn on or off any light, wait for a specific amount of time, dial the phone, or do almost anything else that you would want to do.

Going back to the original flow chart, we can see that if the problem was not the door or an outside movement then it is assumed that an intruder is inside the house. In this case, several lights are turned on and phone calls are placed sequentially to the phone numbers established during initialization.

Since the intruder and the person called on the intercom phone can both hear the message, we will want something that will not only inform the person being called that a break-in is occurring, but that may also serve as a deterrent to the intruder. My message indicates that the owner has already been contacted and that now someone else is being called. This type of activity should convince almost anyone that some other house is better suited for a robbery than yours.

Please don’t hesitate to customize this section to meet your needs. Carefully study the code that implements these charts and I think you will see how simple programming is when you use task modules.
The event timing module adds the ability to automatically schedule events such as waking you up in the morning or controlling the lights to give a lived-in look while you are away. I have 20 events in my program, but you can easily add more or alter mine to your preference.

The heart of this module is the time table array. It is stored in the program under the variable name TT(I,J). Figure 19.1 shows how this two-dimensional array is organized. The index J allows access to any row in the table. Each row contains the information for scheduling one event.

The index I provides access to one of three columns in each entry. Column 1 holds the time that this event should occur. Column 2 always contains the number 1, 2, or 3, which indicates if this event should be done always, only if someone is home, or only if no one is home. The third column contains the number of the actual event to be scheduled, which in my case is between 1 and 20.

A utility program is described in Chapter 22 for creating and editing the time tables. You will need to create seven tables, one for each day of the week. These tables are stored on diskette and are named TT.0 for Sunday's table, TT.1 for Monday's, and so on. This allows you to have total flexibility when scheduling events. Such flexibility will not only create a more lived-in look when you are away, but it can also add to your convenience when you are home.

For example, the house could be used to remind you on Thursday
mornings at 8:15 that you should pick up doughnuts for the office. It might announce the bedtime for the kids at 8:30 on weeknights and 9:30 on weekends. Even the morning wake-up calls are handled using an event in the time table array. The possibilities are endless. Use your imagination.

One very economical use I found for the time table was controlling my heating system. Since I already had a usable BSR system for remote control, I decided to use it to control the heat. An inspection of my thermostat showed that the OFF switch simply disconnected it from the furnace. I used a 12-volt multipole relay to perform the disconnection when activated. A 12-volt wall transformer provides the power for the relay. The secret of my control is that the wall transformer is just plugged into a BSR remote unit. This means I can enable the furnace by pressing buttons on the BSR command console, or the computer can do it using my BSR interface.

Using this method, you cannot really turn down the heat. Either its off completely or it runs normally based on the thermostat. I found, though, that I could leave the heat off entirely from the time I went to bed until 15 minutes before I got up and never notice the drop in temperature. Granted, my house is well insulated and I use an electric blanket. If you have problems with the temperature dropping too low, you can easily schedule the heat to come back on several times during the night for 10 or 15 minutes.

Figure 19.2 shows the list of my events. You can see that they are very easily implemented because they are almost entirely made up of calls to the primary tasks. You should also notice that the wake-up call
Chap. 19 The Event Timing Module

<table>
<thead>
<tr>
<th>EVENT NUMBER</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TURN DEN LIGHT ON</td>
</tr>
<tr>
<td>2</td>
<td>TURN DEN LIGHT OFF</td>
</tr>
<tr>
<td>3</td>
<td>TURN KITCHEN LIGHT ON</td>
</tr>
<tr>
<td>4</td>
<td>TURN KITCHEN LIGHT OFF</td>
</tr>
<tr>
<td>5</td>
<td>TURN FLOODLIGHTS ON</td>
</tr>
<tr>
<td>6</td>
<td>TURN FLOODLIGHTS OFF</td>
</tr>
<tr>
<td>7</td>
<td>TURN OFFICE LIGHTS ON</td>
</tr>
<tr>
<td>8</td>
<td>TURN OFFICE LIGHTS OFF</td>
</tr>
<tr>
<td>9</td>
<td>TURN ALL LIGHTS ON</td>
</tr>
<tr>
<td>10</td>
<td>TURN ALL LIGHTS OFF</td>
</tr>
<tr>
<td>11</td>
<td>TURN STEREO ON</td>
</tr>
<tr>
<td>12</td>
<td>TURN STEREO OFF</td>
</tr>
<tr>
<td>13</td>
<td>TURN HEAT ON</td>
</tr>
<tr>
<td>14</td>
<td>TURN HEAT OFF</td>
</tr>
<tr>
<td>15</td>
<td>WAKE UP CALL</td>
</tr>
<tr>
<td>16</td>
<td>ENABLE INTERRUPTS</td>
</tr>
<tr>
<td>17</td>
<td>ANNOUNCE TIME</td>
</tr>
<tr>
<td>18</td>
<td>TURN BEDROOM LIGHT ON</td>
</tr>
<tr>
<td>19</td>
<td>TURN BEDROOM LIGHT OFF</td>
</tr>
<tr>
<td>20</td>
<td>DISABLE INTERRUPTS</td>
</tr>
</tbody>
</table>

Figure 19.2 The events in the time table can be chosen from the list of Figure 19.1. You can, of course, create any activity that your needs dictate.

subroutine is conditional based on the flags S2 and S3, which indicate if someone is awake and if the sleep alarm is on. The sleep alarm, remember, was set when you used the verbal command "goodnight."

There are a couple of requirements concerning the construction of a time table. First, a special numbering system must be used to indicate the time an event is scheduled (column 1). Figure 19.3 shows some example translations you should use if you are using my clock. The reason for these is that my clock, as discussed in Chapter 6, adds 4 to the most significant digit of the hours to indicate P.M.

My event timing algorithm requires that the times from my clock from midnight to midnight be in numeric order. The automatic addition of 4000 to P.M. numbers by my clock helps to make this true since P.M. numbers will always be larger than A.M. numbers. To have proper order, I also had to convert 12:00 to zero or 4000, depending on whether it was A.M. or P.M.

What this all boils down to is that the time 8:12 A.M. should be

<table>
<thead>
<tr>
<th>ACTUAL TIME</th>
<th>TABLE ENTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDNIGHT</td>
<td>0</td>
</tr>
<tr>
<td>12:15 AM</td>
<td>15</td>
</tr>
<tr>
<td>8:00 AM</td>
<td>800</td>
</tr>
<tr>
<td>8:12 AM</td>
<td>812</td>
</tr>
<tr>
<td>11:30 AM</td>
<td>1130</td>
</tr>
<tr>
<td>NOON</td>
<td>4000</td>
</tr>
<tr>
<td>12:30 PM</td>
<td>4030</td>
</tr>
<tr>
<td>1:00 PM</td>
<td>4100</td>
</tr>
<tr>
<td>8:12 PM</td>
<td>4812</td>
</tr>
<tr>
<td>11:20 PM</td>
<td>5120</td>
</tr>
</tbody>
</table>

Figure 19.3 Standard times must be converted for use in the time table. Some example entries are shown.
entered into the time table as the number 812. If it was 8:12 P.M. you would use 4812. If your clock has a 24-hour format, you can eliminate much of the preceding. I decided to write my program for a 12-hour clock to provide more information for those using other hardware.

Figure 19.4 Each item in the time table is processed sequentially. A table pointer (TP) keeps track of the present position.
The second requirement of the time table is that all entries have to be in chronological order. The reason for this is that my algorithm does not search the table to find events that need executing. This would be too time consuming for my sequential system. If the events are in order, it need only check to see if it is time for the next event. If not, it loops again to see if any other modules require attention. Once an item is executed, a pointer (the variable TP) is moved on to the next entry in the table.

Figure 19.4 shows the flow chart for the event timing module. If all the items in the present table have been completed, the system returns without any further checking. If the time is zero (midnight), then it is time to load a new table. The system clock is checked to see what day it is, and the corresponding table is transferred from the disk.

If the present time is smaller than the time of the item presently being considered (pointed to by TP), nothing happens. Once the time equals or exceeds the scheduled time, the item is processed depending on whether the present “at home status” matches column 2 of the table. In any event, the pointer TP is incremented to the next line of the table.
For the average reader of this book, this module will probably be the hardest to understand. It is not that this section is really any more difficult, but because of the speed requirements it is written entirely in machine language. Even if you are not familiar with machine language, though, I think you can understand the principles here by following the flow charts.

You will recall from Chapter 18 that all the security and under-the-carpet switches are connected to the Apple interrupt pin. When any of these switches are activated, a machine language program is automatically run. This chapter discusses that program. Even though this program can be viewed pretty much as an independent package, it must communicate with the other major modules in the home control system.

The major information passed to this module from the main program is whether the security alarm is on or off. Location 822H will serve as a flag to the machine language program and contains a 0 when the alarm is on. A 1 will indicate it is off. BASIC can turn the alarm on and off by POKING location 2082, which is the decimal equivalent of 822H.

The machine language program has the responsibility of performing one of two different tasks depending on the status of the alarm. If the alarm is on, the input port must be read and the cause of the interrupt determined. Four memory locations are used to keep track of the number of violations that have occurred at a door, window, outside, or at the miscellaneous switch mentioned earlier.
Whenever the program determines that a window has been opened, for example, it will increment the location associated with the window violations. In this case the location is 820H or 2080 decimal. If a door switch caused the interrupt, location 81FH would be incremented.

In addition to a location for a door, window, and so on, a second location will also be incremented. This location is 81DH. Its use was discussed in Chapter 18. The main program will not take the time to check each flag for the doors, windows, and so on, unless the security flag (81DH) is greater than 1. It is the responsibility of the main program to clear each of these locations after the security problem has been attended to.

The second task given to this module is to handle interrupts when the alarm is off. In this case, the switches to be monitored are those under the carpets. Refer to Figure 10.1 for the placement of my switches. They were arranged by trial and error to get a spot that would always get stepped on when someone walks between rooms.

It takes the tripping of two switches to determine movement. For example, if the computer determines that switch 1 was pressed followed by closure at switch 2, it should be obvious that someone left the den and went to the bedroom. A pulse from switch 2 followed by switch 3 would indicate movement from the bedroom to the office.

There are of course some other possibilities. Two people could have tried to enter the hall from different rooms at the same time. Such actions will definitely confuse my house. In practice, though, I found that such occurrences were very rare. I can easily tolerate such errors, especially since a foolproof system would be immensely more complicated.

The computer must keep track of the number of people remaining in each room and tend to the lights accordingly. To control lights, a submodule of this program is used for producing the BSR codes as described in Chapter 8.

Figure 20.1 shows the flow chart for the complete machine language module. When an interrupt occurs, the program first saves all the 6502's registers. It then waits for about 1 millisecond to let any contact bounce from the switches subside. Then the switches are read from the port. If none are pressed, the program assumes that noise caused the interrupt and returns to the original program by way of DONE.

If a switch is pressed, the program handles it in one of two ways, depending on the status of the alarm byte. If the alarm byte is a 0, meaning on, the security section is performed. If it is a 1, the internal switches are checked.

Let's look first at the flow chart for the security section in Figure
20.2. Each bit in the input byte is checked to see which switches are pressed. Everytime a closed switch is found, the appropriate indicator byte is incremented. Finally, a byte is incremented to indicate to BASIC that some kind of security problem has occurred. The program then returns through DONE.
If the alarm is off, only the internal switches need monitoring. Figure 20.3 shows the details of this section. If one of the internal switches is not pressed, the program returns. The normal action, though, is for the program to determine which switch (0, 1, 2, or 3) is pressed and save it as the FROM room. The program then waits until that switch is released so that it will not be read as the second switch.

If someone is actually moving from one room to another, a second switch should be pressed fairly quickly. I have my program watch for the second switch for about 2 seconds. If the second press does not occur in the allotted time, a return is forced.

The second switch is stored as TO. We now have the room number that someone came from and the room they went to. Four locations are used as counters to keep track of how many people are in each room. A room that is entered for the first time will have its light turned on, while a room that is empty will have its light turned off. As an effort to correct errors on its own, the system will not decrement the number in a room below 0.

A special subroutine called SSONIC is used to "press" the buttons on the BSR controller. This routine presses each button twice to ensure

![Diagram](image_url)

Figure 20.2 Security problems cause flags (memory locations) to be changed. The main program monitors these flags and takes the appropriate actions.
Figure 20.3  Internal movement is monitored so that the system can turn lights on and off as required.
reliability. Typically, SSONIC sends the code in the accumulator. BASIC also uses this routine by POKEing the code to be sent into location 2093 and CALLing 2300. Refer to Chapter 8 to review what the requirements of SSONIC are.

Although the BASIC portion of my home control program does not utilize all the information created here (such as the number of people in each room), you could use PEEK statements to retrieve it if you should want such data for your modifications.

As I said at the beginning of this chapter, this portion of the program looks more complicated than it is simply because it is written in machine language. If you take the time to go through it using the flow chart, I think you will find it very easy to follow, even if you are not a programming whiz.

The principles used here can be utilized for most houses, especially if you only want to automate the lights in three or four major rooms. Actually, a few rooms are usually very acceptable. If you do need more rooms, you will have to make extensive modifications to the program, but the basic principles still apply.

The key thing to remember here is that the only things implemented in machine language are the things that absolutely require it. The security flags are a good example of this. The machine language program only sets the flags. BASIC still is used to determine what action is to be taken. This approach provides the speed and flexibility of machine language interrupts and still maintains the ease of modification of BASIC.

The computer’s capability to monitor movement, either inside or outside the house, is contingent on interrupts. Because of this, you can easily stop the monitoring by telling the Apple to ignore interrupts. For the convenience of persons requiring this action, I have provided two routines, which are also listed in Appendix B. To use them from BASIC, just CALL 2058 to enable interrupts and CALL 2075 to disable them.
In previous chapters I have covered each module of the home control program. Unlike many pieces of software, you cannot just type these programs into the computer and run them. Many of the modules require special data tables or machine language sections.

Not only do we need a means to create and maintain these items, but we also have to reserve sections of memory for them to fit in. This chapter will offer a number of programs that can aid us in these tasks.

First, refer to Chapter 8, where we discussed the hardware needed to build your own clock. Although the home control program contains the subroutines needed to use the clock, you will also need a program that will allow you to set the time and to test the clock by observing it in operation.

Appendix C contains a listing of a program to perform these tasks. When it is run, you are asked if you wish to set or observe the clock. The program is very short and is well commented to help you see how each section works.

The next program maintains the speech recognition system. As was discussed in Chapter 4, the Speech Link has to be trained by each user. The words and their templates can be stored on the disk so that programs can utilize speech input without being retrained each time they are run.

The program in Appendix E can be used to create these templates, as well as to perform several other useful tasks. For instance, it provides a
means for you to test different words in a vocabulary. This will allow you
to try substitutes for a word that is hard to recognize and then to retrain
and test the vocabulary. The program also serves as a structured tutorial
for many of the functions of the Speech Link.

You can try new words by altering the DATA statements. I use each
word three times to increase the recognition capabilities. Each word must
be different if Speech Link is to differentiate among them, so I create two
additional words by adding a period and an exclamation point to the
original word.

If you retrain the words without reinitializing, the Speech Link will
average the present templates with the new word. Further information can
be obtained from the Speech Link manual for those of you that plan to use
it in your home control program.

We also need to create the time tables used for the event timing
module. The program to do this is shown in Appendix D. Like the other
maintenance programs, this program is menu driven and self-prompting to
make it easy to use. It not only allows you to create but also to edit the
time tables specified in Chapter 19.

Each of these programs is well commented and almost self-explanatory. Because of this, as well as the fact that they are only maintenance
programs, I will not go into lengthy explanations of their operations. I
think you will find them pleasantly easy to follow if you decide to examine
them in detail.

Now that we have looked at how the data sections for the home
control program are created, let’s see how the space for these sections is
reserved. To put the initialization procedure into proper perspective, we
need to pause for a moment and reexamine the requirements of a system
start-up.

If you refer to Chapter 7, you will remember that the failsafe system
will force a reset whenever the software becomes so “insane” that it can
no longer keep retriggering the system sanity timer. This reset must cause
the disk to reboot. The HELLO program, the one that runs automatically
after a boot, must set up everything required and then run the home
control program itself.

This HELLO program is our next topic. You can find a complete
listing for it in Appendix F. The program is actually very short, but it
performs several vital functions. First, it must POKE a 0 into the power-up
byte, which is decimal 1012. Doing this will make an Apple with an
autostart ROM reboot if a reset occurs.

The program must now reserve the space for the machine language
sections of the program. It does this by POKEing a new LOMEM address of 6600 into locations 103 and 104. In addition, a 0 must be stored into 6600 to meet the requirements of Applesoft.

When this is done, the area from 2048 to 6600 is no longer used by BASIC and is free for our machine language programs and data. You can refer to your Apple documentation for more details concerning the movement of the start of a BASIC program.

The last thing the HELLO program must do is to RUN the home control program, which will in turn load all the appropriate data and machine language programs into the space just created. The sanity timer will be triggered just before each disk operation to ensure that a reset will not occur during the transfer.

At this point you should be able to install the home control system software in your Apple. First, initialize a diskette with the HELLO program just discussed. That diskette should also contain all the programs listed in the appendixes. Use the maintenance programs to create the required data tables and you are all set.

For the convenience of those persons not wishing to spend their weekends typing, you may receive a diskette prepared as just described by sending $12.50 to

John Blankenship
P.O. Box 47934
Doraville, GA 30362.
In the last 21 chapters I have described a complete home control system that can automate your home. I have tried to create a system that offers as much power per dollar as possible. Wherever I could, I used commercially available equipment so that tinkering with hardware would not be a major undertaking when computerizing your home.

I have tried to design the system so that it is not only cost effective but also usable. No matter how many features I add or how many ways I come up with to make your home easier to live in, I suspect that each of you will want to change one little item here or add something there.

And because of that, my most important goal when creating the system was to make it easy to alter and expand. Everything is modular, and although interaction between modules is essential, I hope you will be able to make your modifications as painlessly as possible.

Although I believe that modifications can be relatively easy, there is no way around the fact that you must understand my system in order to make them. And you must understand it in detail. I believe the proper use of this text can provide you with that understanding.

There are many areas open to you for expanding the system, and this book would not be complete if it did not at least mention a few. One of the easiest to implement would be to add an automatic lawn sprinkler system. BSR has recently added an electric valve to their list of accessories. With it, some plastic pipe, and the proper entries into the time tables, you have
established the basis for the system. To make it intelligent, you might connect some type of moisture-sensitive resistance to the Apple paddle ports. Your program could then decide if the grass needs watering and then use the clock to decide when during the day to actually perform the operation.

The Apple paddle inputs could also be used to make my suggested heating system more intelligent. For example, a thermistor could enable the software to determine the temperature in the house and therefore provide a more elaborate basis for controlling the heating system.

If you interfaced a smoke detector to an input pin, your house could deal with a fire in a manner similar to how security problems are handled. There is no way to put a price tag on the safety and peace of mind such a system could offer.

If you have an electric garage door opener, your computerized home can decrease the possibility of someone else being able to open it. The opener could be plugged into a BSR remote. The computer would only energize the remote during the time you are expected home. Once this is done, even if someone had the transmitter, they would not be able to activate the door except for the brief periods during the day when the computer is programmed to permit entry.

You might also consider wiring your doorbell to an input pin. The computer could then handle someone at the door in much the same manner as it monitors the telephone. The possibilities for this one enhancement can be almost limitless.

In fact there are almost unlimited possibilities in almost every area you can imagine. So let your imagination run wild. Don’t rule out anything you are interested in having. Anything and everything is possible. You might have to make compromises or develop a creative strategy, but in the end any goal can be met.

Computerizing my home has been one of my most exciting and rewarding projects. I hope you find the same satisfaction as you computerize yours.
HOME CONTROL PROGRAM

APPLE II HOME CONTROL

BY

JOHN BLANKENSHIP

COPYRIGHT JUNE 1982

LIST

1 REM !=================================
2 REM ! APPLE II HOME CONTROL
3 REM ! PROGRAM
4 REM !
5 REM ! JOHN BLANKENSHIP
6 REM ! COPYRIGHT JUNE 1982
7 REM !=================================
8 REM
9 REM
10 COSUB 900 : REM INITIALIZATION
11 COSUB 1000 : REM PROCESS VOICE REQUESTS
12 COSUB 2000 : REM PROCESS INCOMING PHONE CALLS
13 COSUB 3000 : REM PROCESS SECURITY INFORMATION
14 COSUB 4000 : REM PROCESS NEXT TIME TABLE ENTRY
15 GOTO 20 : REM LOOP BACK AND DO IT AGAIN
16 REM !---------------------------------
17 REM ! CHECK FOR PHONE RINGING
18 REM ! YES = 1 IF RINGING
19 REM ! YES = 0 IF NOT
20 REM !---------------------------------
21 LET YES = 0
22 FOR I = 1 TO 30
23 IF PEEK (RING) < 128 THEN YES = 1: RETURN
24 NEXT I
25 NEXT I
26 RETURN
150 REM !-------------------------!
151 REM ! CHECK FOR SOUND CHANGE !
152 REM ! FOR PERIOD OF 4 SEC. !
153 REM ! YES = 1 IF CHANGE !
154 REM !-------------------------!
155 LET YES = 0
160 LET S6 = 0
165 IF PEEK (SOUND) > 127 THEN S6 = 1
170 FOR I1 = 1 TO 400
175 LET S7 = 0
180 IF PEEK (SOUND) > 127 THEN S7 = 1
185 IF S6 < S7 THEN YES = 1
190 NEXT I1
195 RETURN
200 REM !-------------------------!
202 REM ! DELAY OF 5C SECONDS !
204 REM !-------------------------!
205 DE = 420
210 FOR S7 = 1 TO 5C
215 LET Z = PEEK (SANITY)
220 GOSUB 400
225 NEXT S7
230 RETURN
250 REM !-------------------------!
251 REM ! CHECK FOR OWNER CODE !
252 REM !-------------------------!
253 LET YES = 0
254 FOR I = 1 TO 2000
256 IF PEEK (SOUND) > 127 THEN 262: REM SOUND STARTED
258 NEXT I
260 RETURN: REM NO TONE IN TIME
262 FOR I = 1 TO 200
264 IF PEEK (SOUND) < = 127 THEN 270: REM SOUND STOPPED
266 NEXT I
268 RETURN: REM TONE TOO LONG
270 FOR J = 1 TO 30
272 IF PEEK (SOUND) > 127 THEN 278: REM SOUND RESTARTED
274 NEXT J
276 RETURN: REM PAUSE TOO LONG
278 FOR K = 1 TO 100
280 IF PEEK (SOUND) < = 127 THEN 286: REM SOUND STOPPED
282 NEXT K
284 RETURN: REM 2ND TONE TOO LONG
286 REM AT THIS POINT I, J, & K =
288 REM LENTH OF TONE1, LENTH OF PAUSE
290 REM AND LENTH OF TONE2 RESPECTIVELY
292 LET YES = 1
293 PRINT I, J, K
294 IF I < 95 OR I > 110 THEN YES = 0
296 IF J < 13 OR J > 21 THEN YES = 0
298 IF K < 55 OR K > 70 THEN YES = 0
299 RETURN
300 REM !-------------------------!
301 REM ! SAY MESSAGE # ME !
302 REM !-------------------------!
303 PRINT: PRINT D$"PR$"VSLOT
306 SC = HT(MESSAGE): REM SET DELAY TIME
307 GOSUB 200
308 PRINT D$"PR•O"
309 IF MESSAGE < 50 THEN Z = PEEK (SAN ITY)
310 RETURN
311 PRINT "YES "N1%" HOW CAN I HELP YOU": RETURN
312 PRINT "CAN I DO ANYTHING ELSE FOR YOU"
313 PRINT "I AM READY": RETURN
314 PRINT "O K GOOD BYE FOR NOW": RETURN
315 PRINT "WHAT DO YOU WISH ME TO DO TO TH E "N%": RETURN
316 PRINT "THE "WT%" IS NOW ON": RETURN
317 PRINT "THE "WT%" IS NOW OFF": RETURN
318 PRINT "THE "WT%" IS NOW DIMMED": RETURN
319 PRINT "WHO DO YOU WISH TO CALL": RETURN
320 PRINT I 'AM NOW CALLING "W%": RETURN
321 PRINT "PLEASE EXCUSE ME "N2%" I'LL CHA NGE VOCABULARIES": RETURN
322 PRINT "THE PHONE IS NOW HUNG UP": RETURN
323 PRINT "NOT RIGHT NOW DID Y OU HAVE A GOOD DAY": RETURN
324 PRINT "THAT'S REALLY GREAT": RETURN
325 PRINT "I'M SORRY ABOUT THAT": RETURN
326 PRINT "DID SOMETHING SPECIAL HAPPEN": RETURN
327 PRINT "WHAT WAS IT": RETURN
328 PRINT "I NO HOW YOU FEEL NOTHING SPECIAL HAPPENED HERE EITHER" : RETURN
329 PRINT "THAT'S REALLY INTERESTING": RETURN
330 PRINT "SAY "N1%" DO YOU THINK WE COULD GET ANOTHER COM PERVER": RETURN
331 PRINT "DON'T SAY NO WITHOUT THINKING I T OVER EVEN A LITTLE ONE WO ULD BE O K": RETURN
332 PRINT "BOY THAT'S GREAT I WOULD LOVE TO HAVE SOME COM PAN E WHEN YOU ARE NOT HERE": RETURN
333 PRINT "WELL I'D BETTER GET BACK TO WOR K I HAVE LOTS OF THINGS PILI NG UP GOOD BYE FOR NOW": RETURN
334 PRINT "GOODNIGHT I WILL WATCH THE H OUSE FOR YOU": RETURN
335 PRINT "SHAL I WAKE YOU IN THE MORNING": RETURN
336 PRINT "I WILL LET YOU SLEEP IN THEN": RETURN
337 PRINT "I'LL WAKE YOU AT THE CORRECT TI ME": RETURN
338 PRINT "ARE YOU LEAVING": RETURN
339 PRINT "SHAL I HANDLE SECURITY FOR YOU": RETURN
340 PRINT "O K I'LL WATCH THE PHONE FOR YOU TILL YOU GET BACK": RETURN
341 PRINT "HAVE A GOOD TIME I WILL TAK E CARE OF EVERYTHING": RETURN
PRINT "DO YOU WISH A STATUS REPORT" : RETURN
PRINT "DO YOU WISH SOMETHING TURNED ON" : RETURN
PRINT "DO YOU WISH SOMETHING TURNED OFF" : RETURN
PRINT "IS THAT YOU MASTER" : RETURN
PRINT "PLEASE EXCUSE ME DUE TO A TECHNICAL PROBLEM I AM REINITIALIZING IS ANY ONE HOME" : RETURN
PRINT "WARNING WARNING THE OWNER HAS ALREADY BEEN CALLED AND TOLD YOU ARE ENTERING THE HOUSE I AM NOW TRYING TO CALL SOMEONE ELSE FOR YOUR OWN SAFETY IT WOULD BE BEST FOR YOU TO LEAVE IMMEDIATELY" : RETURN
PRINT "GIVE ME A YES FOR EACH VALID ANSWER" : RETURN
PRINT "OFFICE" : RETURN
PRINT "STEREO" : RETURN
PRINT "BEDROOM" : RETURN
PRINT "DEN" : RETURN
PRINT "KITCHEN" : RETURN
PRINT "OUTSIDE" : RETURN
PRINT "HEAT" : RETURN
PRINT "THERE HAVE BEEN NO SECURITY PROBLEMS" : RETURN
PRINT "THERE HAS BEEN "58" OUTSIDE WARNINGS" : RETURN
PRINT "THERE HAS BEEN "59" INSIDE WARNINGS" : RETURN
PRINT "THERE HAS BEEN "NC" PHONE CALLS " : RETURN
PRINT "PLEASE REPEAT THAT" : RETURN
PRINT "IT IS "DAY$(T(6))" THE TIME IS "HR$" "MIN$" "M$" : RETURN
PRINT "WAKE UP WAKE UP IT IS TIME TO GET UP YOU CANNOT STAY IN BED ALL DAY WAKE UP " : RETURN
PRINT "DO YOU WANT ME TO GET THE PHONE " : RETURN
PRINT "I'M SORRY BUT I MUST GO NOW GOOD BYE" : RETURN
REM |------------------------|
REM | INIT. SPEECHLINK |
REM |------------------------|
LET HI = INT (WAA / 256)
LET LO = WAA - HI * 256
POKE (1144 + Sslot), LO
POKE (1272 + Sslot), HI
LET HI = INT (TAA / 256)
LET LO = TAA - HI * 256
POKE (WAA + 2), LO
POKE (WAA + 3), HI
395 PRINT D$"PRESSLOT
396 PRINT 1$
397 PRINT D$"PR0"
398 RETURN
400 REM !---------------------!
402 REM ! DELAY BASED ON DE !
404 REM !---------------------!
410 FOR D = 1 TO DE
420 NEXT D
430 RETURN
450 REM !---------------------!
452 REM ! WIRELESS MIKE ON ? !
454 REM !---------------------!
455 LET YES = 0
460 IF PEEK (VOICE) < 128 THEN YES = 1
470 RETURN
475 REM !---------------------!
476 REM ! LOAD TIME TABLE !
477 REM !---------------------!
480 GOSUB 600
482 PRINT D$"OPEN TT."T(6)
484 PRINT D$"READ TT."T(6)
486 FOR J = 1 TO 20
488 FOR I = 1 TO 3
490 INPUT TT(I,J)
492 NEXT J,J
494 PRINT D$"CLOSE"
496 LET TP = 1
498 RETURN
500 REM !---------------------!
502 REM ! TURN ON BY PHONE !
504 REM !---------------------!
505 GOSUB 540
510 LET BU = 20: GOSUB 750
515 RETURN
525 REM !---------------------!
527 REM ! TURN OFF BY PHONE !
529 REM !---------------------!
530 GOSUB 540
534 LET BU = 28: GOSUB 750
538 RETURN
540 REM ----SUBR. FOR ON/OFF--
541 LET MESSAGE = 38: GOSUB 300
542 FOR M = 1 TO 7
543 LET MESSAGE = M + 38: GOSUB 300
544 GOSUB 250
545 IF YES THEN BU = BU(M): GOSUB 750
546 NEXT M
547 RETURN
600 REM !---------------------!
602 REM ! READ TIME T(X) !
603 REM ! & TM !
604 REM !---------------------!
605 FOR Y = 0 TO 8
610 GOSUB 620
612 LET T(Y) = X
614 NEXT Y
618 LET TM = T(2) + T(3) * 10 + T(4) * 100 + T(5) * 1000
618 RETURN
620 REM ----SUBR. TO READ T(X)--
621 POKE CB,0: POKE B,128: REM SET DDR
622 POKE CB,4: POKE B,0: REM HOLD CLOCK F OR READING
624 POKE CA,0: POKE A,240: POKE CA,4: REM P ORT A READY FOR READ
POKE B, 128: REM CHIP ON
POKE A, (128 + Y * 16): REM Y IS ADDRESS
POKE 2048, PEEK (A): CALL 2049: X = PEEK (2048)
POKE A, 0: REM RESET READ LINE
POKE B, 0: REM TURN CHIP OFF
RETURN
REM !-------------------!
REM !-------------------!
REM !-------------------!
REM
REM !-------------------!
REM !-------------------!
REM !-------------------!
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
REM
REM !-------------------!
LET WN = INT (((PEEK (WAA + 6)) / 3) + 1)
FOR X = MASK TO MASK + 63
POKE X, 0
NEXT X
LET Z = PEEK (SANITY)
RETURN
REM !----------------------------------!
REM ! MAIN INITIALIZATION !
REM !-------------------------!
GOSUB 950: REM INIT VARIABLES
GOSUB 375: REM INIT SPEECH LAB
LET Z = PEEK (SANITY)
PRINT DS "$BLOAD HOME.ML"
LET Z = PEEK (SANITY)
PRINT DS "$BLOAD JOHN": REM VOCABULARY
LET Z = PEEK (SANITY)
GOSUB 475: REM LOAD TIME TABLE
LET Z = PEEK (SANITY)
GOSUB 600: REM READ TIME
IF TM < 1300 AND TM > 1200 THEN TM = TM - 1200
IF TM < 5300 AND TM > 5200 THEN TM = TM - 1200
LET S1 = 3: S2 = 0: S3 = 0: S4 = 0: S5 = 0:
S6 = 0: S9 = 0: NC = 0
IF TM ) 5100 OR TM < 900 THEN S1 = 1: GOTO 940
LET MESSAGE = 36: GOSUB 300
FOR I = 1 TO 400
GOSUB 450
IF YES THEN S1 = 2: S2 = 1: S3 = 1: S4 = 1: MESSAGE = 4: GOSUB 300: GOTO 940
NEXT I
POKE 2082, S4: REM ALARM FLAG
CALL 2050: REM ENABLE INTERRUPTS
RETURN
REM !----------------------------------!
REM ! VARIABLE INITIALIZATION!
REM !----------------------------------!
LET PF = 49240
LET PN = 49241
LET RF = 49242
LET RN = 49243
LET TF = 49244
LET TN = 49245
LET VOICE = 49249
LET SOUND = 49251
LET SSLOT = 1
LET VSLOT = 7
LET MLSLOT = 2
LET RING = 16 * MLSLOT - 16251
LET SP$(1) = "399-8030": SP$(2) = "452-1562"
LET A = 49400: CA = A + 1
LET B = A + 2: CB = A + 3
LET SANITY = 49393
DIM MT(64)
FOR I = 1 TO 64: READ MT(I): NEXT I
DATA 3, 3, 1, 2, 4, 4, 4, 1, 3, 4, 2, 6, 1, 1, 3, 1
7, 1, 4, 8, 8, 9, 3, 2, 1, 1, 1, 2, 4, 5, 2, 2, 2, 1, 1
1, 28, 2, 1, 1, 1, 1, 1, 1, 1, 3, 2, 3, 2, 1, 5, 9, 2, 2
0, 0, 0, 0, 0, 0, 0, 0, 0
LET N$ = "JOHN"
LET N2$ = "WANDA"

FOR I = 0 TO 6: READ DAY$(I): NEXT I

DATA SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY

FOR I = 1 TO 4: READ PN$(I): NEXT I

DATA "452-1954", "390-6030", "7", "8"

FOR I = 0 TO 9: READ TEEN$(I): NEXT I

DATA TEN, ELEVEN, TWELVE, THIRTEEN, FOURTEEN, FIFTEEN, SIXTEEN, SEVENTEEN, EIGHTEEN, NINETEEN

FOR I = 2 TO 5: READ TN$(I): NEXT I

DATA TWENTY, THIRTY, FORTY, FIFTY

LET WAA = 3000

LET TAA = WAA + 557

LET MASK = TAA + 2177

LET D$ = "": REM CTL D

LET I$ = "": REM CTL I

LET V$ = "": REM CTL V

DIM TT(3, 20)

FOR I = 1 TO 7: READ BU(I): NEXT I

DATA 6, 7, 4, 5, 0, 9, 10

LET Q$ = CHR$(17): LF$ = CHR$(10): Z$ = "": REM CTL Z

POKE TF, 0: POKE PF, 0: POKE RF, 0

RETURN

REM !-------------------------!!!

REM ! PROCESS VOICE COMMANDS !

REM !-------------------------!!!

GOSUB 450: REM IS MIKE ON

IF NOT YES THEN RETURN

LET S1 = 2: S2 = 1: S3 = 1: S4 = 1: POKE 2082, S4

IF S5 = 1 THEN POKE TF, 0: S5 = 0: MESSAGE = 12: GOSUB 300: RETURN

LET MESSAGE = 1: GOSUB 300

FOR I = MASK TO MASK + 38: POKE I, 1: NEXT I

GOSUB 850: REM RECOG. A WORD

GOSUB 1100, 1100, 1100, 1100, 1100, 1100, 1100, 1100, 1100, 1100, 1100, 1100

GOSUB 800

IF YES THEN MESSAGE = 3: GOSUB 300: GOTO 1010

FOR I = MASK TO MASK + 39 TO MASK + 50: POKE I, 1: NEXT I

GOSUB 850

IF WN = 17 THEN RETURN

GOSUB 750: REM PRESS BUTTON BU

GOSUB 750: REM 'ON'

LET MESSAGE = 6: GOSUB 300

RETURN

LET BU = 28: GOSUB 750: REM 'OFF'

LET MESSAGE = 7: GOSUB 300

RETURN
LET BU • 18
FOR I = 1 TO 50
GOSUB 750 : REM PRESSES 'DIM' ONCE
LET MESSAGE • 8 : GOSUB 300
RETURN
REM !-------------------!
REM ! ANNOUNCE THE TIME !
REM !-------------------!
GOSUB 600 : REM READ TIME
LET M$ = " A M "
IF T(5) > 1 THEN M$ = " P M ": T(5) = T(5) - 4
LET HR$ = ""
IF T(4) > 0 THEN HR$ = STR$(T(4))
LET M$ = ""
IF T(3) + T(2) = 0 THEN MIN$ = " O C L OCK ": GOTO 1232
IF T(2) > 0 THEN MIN$ = STR$(T(2))
IF T(3) = 1 THEN MIN$ = TEEN$(T(2)): GOTO 1232
IF T(3) > 1 THEN MIN$ = TEN$(T(3)) + M
IN$;
GOSUB 300
RETURN
REM ! VOICE CTL OF PHONE !
REM !-------------------!
LET MESSAGE • 8 : GOSUB 300
LET M$ = ""
IF T < 5 THEN M$ = ""
IF T < 4 THEN HRS = TEENS <T<4>>
IF T < 3 THEN MINS • TEN$ <T<2>>: COTO
LET MESSAGE • 8 : GOSUB 300
LET MESSAGE • 10 : GOSUB 300
LET PNS$ = PNs(WN - 17)
LET FN$ = FN$(WN - 17)
GOSUB 650 : REM DIAL PHONE
LET S$ = " PHONE OFF HOOK
RETURN
REM !-------------------!
REM ! "CONVERSATION" MODE !
REM !-------------------!
LET MESSAGE = 9 : GOSUB 300
FOR I = MASK + 40 TO MASK + 62: POKE
I, 1 : NEXT I
GOSUB 850 : REM RECOG. A WORD
IF WN = 17 THEN RETURN
LET MESSAGE = 10 : GOSUB 300
LET PNS$ = PNS$(WN - 17)
GOSUB 650 : REM DIAL PHONE
LET S$ = 1 : REM PHONE OFF HOOK
RETURN
REM !-------------------!
REM !-------------------!
LET MESSAGE = 13 : GOSUB 300
GOSUB 800
IF YES THEN MESSAGE = 14 : GOSUB 300 : GOTO 1430
LET MESSAGE = 15 : GOSUB 300
LET MESSAGE = 16 : GOSUB 300
GOSUB 800
IF NOT YES THEN MESSAGE = 18 : GOSUB 300 : GOTO
LET MESSAGE = 17 : GOSUB 300
GOSUB 800
LET MESSAGE = 19 : GOSUB 300
LET MESSAGE = 20 : GOSUB 300
GOSUB 800
IF YES THEN MESSAGE = 22 : GOSUB 300 : GOTO 1480
LET MESSAGE = 21 : GOSUB 300
LET MESSAGE = 23 : GOSUB 300
RETURN
REM !-------------------!
REM ! CHANGE VOCABULARIES !
REM !-------------------!
LET MESSAGE = 11 : GOSUB 300
1620 PRINT D$"LOAD "N2$  
1630 LET T$ = N1$  
1640 LET N1$ = N2$  
1650 LET N2$ = T$  
1660 RETURN  
1700 REM !------------------!  
1701 REM ! HANDLE GOODNIGHT !  
1702 REM !------------------!  
1710 LET MESSAGE = 24: GOSUB 300  
1715 LET S2 = 0:S4 = 0 : POKE 2082,S4  
1716 CALL 2050: REM ENABLE INTERRUPTS  
1720 LET MESSAGE = 25: GOSUB 300  
1725 GOSUB 600  
1730 IF YES THEN 1750  
1735 LET S3 = 1  
1740 LET MESSAGE = 26: GOSUB 300  
1745 RETURN  
1750 LET S3 = 0  
1755 LET MESSAGE = 27: GOSUB 300  
1760 RETURN  
1800 REM !------------------!  
1801 REM ! HANDLE BYE BYE !  
1802 REM !------------------!  
1810 LET MESSAGE = 28: GOSUB 300  
1815 GOSUB 600  
1820 IF NOT YES THEN S4 = 1 : POKE 2082,S4 : RETURN  
1821 NC = 0: REM RESET NUMBER OF CALLS  
1822 LET S1 = 3: REM SOMEONE HOME  
1825 LET MESSAGE = 29: GOSUB 300  
1830 GOSUB 600  
1835 IF NOT YES THEN MESSAGE = 30: GOSUB 300:S4 = 1: POKE 2082,S4: RETURN  
1840 LET MESSAGE = 31: GOSUB 300  
1845 LET SC = 45: GOSUB 200  
1850 LET S4 = 0 : POKE 2082,S4  
1851 CALL 2050: REM ENABLE INTERRUPTS  
1855 RETURN  
2000 REM !==================================================================  
2010 REM ! PROCESS PHONE CALLS !  
2020 REM !==================================================================  
2025 LET Z = PEEK (SANITY)  
2030 GOSUB 100: REM CHECK FOR RING  
2040 IF NOT YES THEN RETURN  
2045 LET NC = NC + 1  
2050 ON S1 GOTO 2060,2340,2110  
2060 REM ------STATUS UNKNOWN------  
2070 LET SC = 3: GOSUB 200: REM 3 SEC DE  
LAV  
2080 GOSUB 100: REM RING CHECK  
2090 IF NOT YES THEN RETURN : REM STILL UNSURE  
2100 LET S1 = 3: REM ASSUME NO ONE HOME  
2110 REM ------NO ONE HOME--------  
2115 POKE TN,0  
2120 POKE PN,0: REM PLAY MESSAGE  
2125 LET Z = PEEK (SANITY)  
2130 LET DE = 11900: GOSUB 400: REM PLAYER DELAY  
2135 LET Z = PEEK (SANITY)  
2140 POKE PF,0  
2150 POKE RN,0  
2160 GOSUB 250: REM CHECK FOR OWNER  
2170 IF YES THEN 2260  
2180 FOR I = 1 TO 15: REM MAX OF 80 SEC. MESSAGE
LET Z = PEEK (SANITY)
GOSUB 150: REM 3 SEC SOUND CHECK
IF NOT YES THEN 2230
IF PEEK (2077) < 0 THEN MESSAGE = 54: GOSUB 300: GOTO 2230
NEXT I
POKE RF, 0
POKE TF, 0
RETURN
REM -----OWNER CALLING--------
POKE RF, 0
LET MESSAGE = 32: GOSUB 300
GOSUB 250: IF YES THEN GOSUB 700
LET MESSAGE = 33: GOSUB 300
GOSUB 250: IF YES THEN GOSUB 500
LET MESSAGE = 34: GOSUB 300
GOSUB 250: IF YES THEN GOSUB 525
LET MESSAGE = 4: GOSUB 300
POKE TF, 0
RETURN
REM ---SOMEONE HOME--------
LET MESSAGE = 53: GOSUB 300
FOR I = 1 TO 500: REM SEVERAL SECOND
POKE TN, 0
LET SS 1: REM FLAG FOR PHONE ON
RETURN
REM PROCESS SECURITY FLAGS
LET Z = PEEK (SANITY)
IF PEEK (2077) = 0 THEN RETURN: REM NO SECUR. PROB.
IF PEEK (2079) > 0 THEN 3100: REM DOOR OPEN -------
LET MESSAGE = 35: GOSUB 300
GOSUB 5000: REM DEN ON
FOR I = 1 TO 700
LET Z = PEEK (SANITY)
GOSUB 450
POKE 2085, I: REM 1 PERSON IN DEN
POKE 2086, 0: POKE 2087, 0: POKE 2098, 0
RETURN
REM ----SOMEONE OUTSIDE-----
GOSUB 5850: GOSUB 5500
LET SC 15: GOSUB 200
IF PEEK (2078) > 0 THEN 3200: REM 0
GOTO 3300
REM ---- DOOR OPEN -------
LET MESSAGE = 35: GOSUB 300
GOSUB 5000: REM DEN ON
FOR I = 1 TO 700
LET Z = PEEK (SANITY)
GOSUB 450
IF YES THEN 3140
NEXT I
GOTO 3300
LET MESSAGE = 32: GOSUB 300
GOSUB 800
IF YES THEN GOSUB 700: GOTO 3160
LET MESSAGE = 4: GOSUB 300
GOSUB 3400
POKE 2082, 1: REM 1 PERSON IN DEN
POKE 2086, 0: POKE 2087, 0: POKE 2088, 0
LET S1 = 2: S2 = 1: S3 = 1: S4 = 1: S8 = 0: S9 = 0: NC = 0
POKE 2082, S4
RETURN
REM ---SOMEONE OUTSIDE-----
GOSUB 5850: GOSUB 5500
LET SC = 15: GOSUB 200
3215  GO SUB 200
3220  LET SC = 15: GO SUB 200
3225  GO SUB 5400
3230  GO SUB 3400
3235  LET SC = 20: GO SUB 200
3236  LET SC = 20: GO SUB 200
3237  LET SC = 20: GO SUB 200
3240  GO SUB 5450
3245  LET S0 = S0 + 1
3250  RETURN
3300  REM ------WINDOW OR INSIDE------
3305  GO SUB 5050: GO SUB 5850
3310  GO SUB 5150: GO SUB 5300
3315  FOR I = 1 TO 2
3320  LET PN(I) = SP(I)
3325  GO SUB 650
3330  LET MESSAGE = 37: GO SUB 300
3335  POKE TF,0: S5 = 0
3340  LET SC = 5: GO SUB 200
3345  NEXT I
3350  GO SUB 3400
3355  LET S0 = S0 + 1
3360  RETURN
3400  REM ------CLEAR ML FLAGS--------
3405  POKE 2077,0: REM SECURITY FLAG
3410  POKE 2078,0: REM OUTSIDE FLAG
3415  POKE 2079,0: REM DOOR FLAG
3420  POKE 2080,0: REM WINDOW FLAG
3425  POKE 2081,0: REM MISC. FLAG
3430  RETURN
4000  REM !=================================================================!
4001  REM ! PROCESS NEXT ITEM   !
4002  REM ! IN TIME TABLE   !
4003  REM !=================================================================!
4015  GO SUB 600: REM READ TIME
4016  IF TM > 1200 AND TM < 1300 THEN TM = TM - 1200
4017  IF TM > 5200 AND TM < 5300 THEN TM = TM - 1200
4018  IF TM < 0 THEN 4020
4020  GO SUB 475
4022  LET Z = PEEK (SANITY)
4025  GO SUB 600
4028  IF TM = 0 THEN 4022
4029  IF TP > 20 THEN RETURN
4030  IF TM < TT(1,TP) THEN RETURN
4032  IF TT(2,TP) = 1 THEN 4035
4033  IF TT(2,TP) = 0 THEN RETURN
4035  ON (TT(3,TP)) GO SUB 5000,5050,5100,51 50,5200,5250,5300,5350,5400,5450,5500, 5550,5600,5650,5700,5750,5800,5850,590 0,5950
4040  LET TP = TP + 1
4050  RETURN
4997  REM !------------------------------------------------------------------!
4998  REM ! TIME TABLE SUBROUTINES   !
4999  REM !------------------------------------------------------------------!
5000  REM ------- DEN ON ---------
5010  LET BU = 5: GO SUB 750
5020  LET BU = 20: GO SUB 750
5030  RETURN
5050  REM ------- DEN OFF ---------
5060  LET BU = 5: GO SUB 750
5070  LET BU = 20: GO SUB 750
5080  RETURN
5100  REM -------- KITCHEN ON --------
5110  LET BU = 8: GOSUB 750
5120  LET BU = 20: GOSUB 750
5130  RETURN
5150  REM -------- KITCHEN OFF --------
5160  LET BU = 8: GOSUB 750
5170  LET BU = 20: GOSUB 750
5180  RETURN
5200  REM -------- FLOODLIGHTS ON ----
5210  LET BU = 9: GOSUB 750
5220  LET BU = 20: GOSUB 750
5250  RETURN
5250  REM -------- FLOODLIGHTS OFF ----
5260  LET BU = 9: GOSUB 750
5270  LET BU = 28: GOSUB 750
5280  RETURN
5300  REM -------- OFFICE ON -----------
5310  LET BU = 6: GOSUB 750
5320  LET BU = 20: GOSUB 750
5330  RETURN
5350  REM -------- OFFICE OFF -----------
5360  LET BU = 6: GOSUB 750
5370  LET BU = 28: GOSUB 750
5380  RETURN
5400  REM -------- ALL ON ---------------
5410  LET BU = 24: GOSUB 750
5420  RETURN
5450  REM -------- ALL OFF ---------------
5460  LET BU = 18: GOSUB 750
5470  RETURN
5500  REM -------- STEREO ON-----------
5510  LET BU = 7: GOSUB 750
5520  LET BU = 20: GOSUB 750
5530  RETURN
5550  REM -------- STEREO OFF---------
5560  LET BU = 7: GOSUB 750
5570  LET BU = 28: GOSUB 750
5580  RETURN
5600  REM -------- HEAT ON --------------
5610  LET BU = 10: GOSUB 750
5620  LET BU = 28: GOSUB 750
5630  RETURN
5650  REM -------- HEAT OFF------------
5660  LET BU = 10: GOSUB 750
5670  LET BU = 20: GOSUB 750
5680  RETURN
5700  REM -------- WAKE UP CALL -------
5705  IF S3 = 1 OR S2 = 1 THEN RETURN
5710  LET MESSAGE = 52: GOSUB 300
5715  GOSUB 5850
5720  GOSUB 600
5730  LET MESSAGE = 51: GOSUB 300
5735  LET S2 = 1: S3 = 1: S4 = 1: POKE 2082, S
4 3
5740  RETURN
5750  REM ------- ENABLE INTERRUPTS-----
5755  CALL 2058
5760  RETURN
5800  REM ------- ANNOUNCE TIME ------
5810  GOSUB 1200
5830  RETURN
5850  REM ------- BEDROOM ON ----------
5860  LET BU = 4: GOSUB 750
5870  LET BU = 20: GOSUB 750
5880 RETURN
5900 REM ---- BEDROOM OFF -----
5910 LET BU = 4: GOSUB 750
5920 LET BU = 28: GOSUB 750
5930 RETURN
5950 REM ---- DISABLE INTERRUPTS ----
5955 CALL 2075
5960 RETURN
Home Control Program
(Machine Language)

Appendix B

:ASM

1000 *
1010 * UPPER NIBBLE MASK ROUTINE FOR
1020 * USE WITH THE 5832 CLOCK CHIP
1030 *

0800-00 1040 TEMCLK .DA #0 CLOCK DATA
0801-AD 00 08 1050 CLOCK LDA TEMCLK GET DATA
0804-29 0F 1060 AND #$0F MASK HIGH NIBBLE
0806-8D 00 08 1070 STA TEMCLK REPLACE DATA
0809-60 1080 RTS

0900 *
1100 * ROUTINES TO CONTROL INTERRUPTS
1110 *

0A0A-A9 30 1120 ENABLE LDA #INTER SET UP INTERRUPTS
0B0C-6D FE 03 1130 STA #$FF VECTOR
0B0F-A9 08 1140 LDA /INTER
0B11-6D FF 03 1150 STA #$FF
0B14-A9 04 1155 LDA #4 INITIALIZE THE
0B16-6D F7 CO 1156 STA SWPORT+1 PIA PORT
0B19-58 1160 CLI ENABLE INTERRUPTS
0B1A-60 1170 RTS
0B1B-76 1180 DISABL SEI DISABLE INTERRUPTS
0B1C-60 1180 RTS

1200 *
1210 * INTERRUPT ROUTINE FOR HANDLING
1220 * SWITCHPLATE INPUTS AND CONTROL-
1230 * IING LIGHTS THROUGH THE BSR
1240 * ULTRA SONIC COMMAND CONCOL
1250 *
1260 * JOHN BLANKENSHIP
1270 *

FC00-

1280 WAIT .EQ $FC00 MONITOR DELAY
COF6- 1290 SWPORT .EQ $COF6 INPUT PORT
0810- 00 1300 SECDF .DA $0 SECURITY FLAG
0811- 00 1310 OUTCF .DA $0 OUTSIDE FLAG
0812- 00 1320 DOORF .DA $0 DOOR FLAG
0820- 00 1330 WINDF .DA $0 WINDOW FLAG
0821- 00 1340 INSDF .DA $0 MIS. FLAG
0822- 00 1350 ALARM .DA #1 STATUS OF ALARM
0823- 00 1360 TO .DA $0 ROOM MOVEMENT TO
0824- 00 1370 FROM .DA $0 ROOM MOVEMENT FROM
0825- 00 00 00 00
0828- 00 1380 NUM .HS 00000000 NUMBER IN EACH ROOM
0828- 05 04 06
082C- 07 1390 TABLE .HS 05040607 BSR CODE TABLE
082D- 00 1400 DATA .DA #0 PERM CODE HOLDER
082E- 00 1410 DATA1 .DA #0 TEMP CODE HOLDER
082F- 00 1420 DATA2 .DA #0 TIMES TO SEND EACH CODE
0830- A5 45 1430 INTER LDA #45 RESTORE ACC
0832- 49 1440 PHA SAVE REGISTERS
0833- 8A 1450 TXA
0834- 48 1460 PHA
0835- 98 1470 TYA
0836- 48 1480 PHA
0837- A9 04 1490 LDA #4 WAIT .1 MSEC
0839- 20 A8 FC 1500 JSR WAIT DEBOUNCE
083C- AD F6 C0 1510 LDA SWPORT
083F- C9 FF 1520 CMP #$FF CK FOR NONE PRESSED
0841- F0 3D 1530 BEQ DONE
0843- AC 22 09 1540 LDY ALARM SEE 1F ALARM SET
0846- FO 4A 1550 BEQ SECUR DO SECURITY STUFF
0848- 09 FO 1560 ORA #$F0 MASK TOP NIBBLE
084A- A2 FF 1570 LDX #$FF SET COUNTER
084C- E8 1580 AGAIN INX
084D- 4A 1590 LSR CK NEXT ROOM
084E- 90 06 1600 BCC FOUND ROOM 15 IN X
0850- C9 00 1610 CMP #$0 DONE LOOKING
0852- F0 2C 1620 BEQ DONE YES AND NONE FOUND
0854- DO F6 1630 BNE AGAIN NO-KEEP LOOKING
0856- BE 24 08 1640 FOUND STX FROM SAVE ROOM FROM
0859- AD F6 C0 1650 PAUSE LDA SWPORT WAIT TILL SWITCHES RELEASED
085C- C9 FF 1660 CMP #$FF
085E- D0 F9 1670 BNE PAUSE
0860- A9 04 1670 LDA #$4 WAIT .1 MSEC
0862- 20 A8 FC 1690 JSR WAIT FOR DEBOUNCE
0865- 20 DD FB 1700 JSR #$F0DD BELL
0868- A2 FF 1710 SECOND LDX #$FF FOR 5 SECONDS LOOK
086A- A0 FF 1720 OVER LDY #$FF FOR 2ND SWITCH
086C- AD F6 C0 1730 MORE LDA SWPORT READ SWITCHES
086F- 29 DF 1740 AND #$F MASK HIGH BYTE
0871- C9 00 1750 CMP #$F CK FOR PRESSED
0873- D0 3B 1760 BNE ONEWAS
0875- A9 04 1770 LDA #$4 WAIT .1 MSEC
0877- 20 A8 FC 1780 JSR WAIT FOR DEBOUNCE
087A- 88 1790 DEX LOOP UNTIL
087B- D0 EF 1800 BNE MORE 2 SECONDS
087D- CA 1810 DXE ARE UP
087E- D0 EA 1820 BNE OVER
0880- AD F6 C0 1830 DONE LDB SWPORT WAIT TILL RELEASED
0883- C9 FF 1840 CMP #$FF
0885- D0 F9 1850 BNE DONE
0887- A9 04 1860 LDA #$4 ANOTHER .1 MSEC
0889- 20 A8 FC 1870 JSR WAIT DEBOUNCE
088C- 68 1880 PLA RESTORE REGISTERS
088D- A8 1890 TAY
088E- 68 1900 PLA
140
088F- AA 1910 TAX
0890- 68 1920 PLA
0891- 40 1930 RTI
0892- 0A 1940 SECUR ASL
0893- B0 03 1950 BCS DOOR
0895- EE 1E 08 1960 INC OUTF
0896- 0A 1970 DOOR ASL
0897- B0 03 1980 BCS WINDOW
0898- 0A 1990 INC DOORF
0899- EA 03 2000 WINDOW ASL
089A- B0 03 2010 BCS INSIDE
089B- EE 21 08 2020 INC WINDF
089C- 4C 0D 08 2030 JMP DONE
089E- AF FF 2080 ONEWAS LDX #FF
089F- B0 03 2090 NEXT INX
08A0- 4A 2100 LSR
08A1- 90 02 2110 BCC GOTOIT
08A2- B0 FA 2120 BCS NEXT
08A3- 6E 23 08 2130 GOTIT STX TO
08A4- FE 25 08 2140 INC NUM,X INC # IN ROOM TO
08A5- AE 24 08 2150 LDX FROM
08A6- DE 25 08 2160 DEC NUM,X DEC # IN ROOM FROM
08A7- 10 0F 2170 BPL OK
08A8- A9 00 2180 LDA #10 NEVER LET # BE NEGATIVE
08A9- 0B 2190 STA NUM,X
08AA- AE 23 08 2200 LDX TO SEE IF 1ST
08AB- BD 25 08 2210 LDA NUM,X PERSON TO
08AC- C9 01 2220 CMP #1 ENTER
08AD- DE 00 2230 BNE OFF
08AE- AE 23 08 2240 OK LDX TO
08AF- BD 28 08 2250 LDA TABLE,X CONVERT TO CODE
08B0- 20 FF 08 2260 JSR SSONIC TURN ON LIGHT
08B1- A9 14 2270 LDA #20 CODE FOR "ON"
08B2- 20 FF 08 2280 JSR SSONIC
08B3- AE 24 08 2290 OFF LDX FROM
08B4- BD 25 08 2300 LDA NUM,X
08B5- 25 08 2310 BNE DONE IS EMPTY
08B6- BD 29 08 2320 LDA TABLE,X CONVERT TO CODE
08B7- 20 FF 08 2330 JSR SSONIC TURN OFF LIGHT
08B8- A9 1C 2340 LDA #28 CODE FOR "OFF"
08B9- 20 FF 08 2350 JSR SSONIC
08BA- 4C 0D 08 2360 JMP DONE
08BB- 0D 2D 08 2370 SSONIC STA DATA
08BC- A0 03 2380 BEGIN LDY #$3 KEY CODE
08BD- 03 2400 TWO DEC DATA2 SEND EACH TWICE
08BE- 2F 08 2410 BNE MAIN
08BF- CE 2F 08 2420 LDY #1 LONG WAIT
08C0- D0 0B 2430 WT LDA #FF
08C1- CA 0A 2440 JSR WAIT
08C2- FF 08 2450 DEX
08C3- D0 FC 2460 BNE WT
08C4- D0 60 2470 RTS
08C5- A2 01 2480 MAIN LDX #$1 START BIT
08C6- 20 55 09 2490 JSR ONE
08C7- AD 2D 08 2500 LDA DATA PREPARE TO SEND DATA BITS
08C8- 0D 2E 08 2510 STA DATA1 PREPARE FOR COMPLEMENT DATA
08C9- 20 41 09 2520 JSR OUT
08CA- AD 1D 08 2530 LDA DATA
08CB- 49 1F 2540 EOR #$1F
08CC- 0D 2E 08 2550 STA DATA1

141
0927-  20 41 09  2560  JSR OUT
092A-  A0 FF  2370  LDY #$FF  STOP BITS
092C-  20 66 09  2580  JSR PULSES
092F-  A0 FF  2390  LDY #$FF
0931-  20 66 09  2600  JSR PULSES
0934-  A0 82  2610  LDY #$82
0936-  20 66 09  2620  JSR PULSES
0939-  A9 60  2630  LDA #$60  PAUSE BETWEEN
093B-  20 A8 FC  2640  JSR WAIT
093E-  4C 01 09  2650  JMP TWO
2660  *-----------------------------------------------------------------------
2670  * SUBROUTINE 'OUT' OUTPUTS
2680  *  FIVE BITS IN DATA1
2690  *-----------------------------------------------------------------------
0941-  A2 05  2700  OUT  LDX #$5  GET READY
0943-  4E 2E 06  2710  OUT1  LSR DATA1  GET NEXT BIT
0946-  B0 0D  2720  BCS ONE  SEND A ONE
0948-  A0 30  2730  ZERO  LDY #$30  OR A ZERO
094A-  20 66 08  2740  JSR PULSES  (48 PULSES)
094D-  A9 31  2750  LDA #$31
094F-  20 A8 FC  2760  JSR WAIT  6.83 MSEC
0952-  4C 64 09  2770  JMP MOR  CONTINUE
0955-  A0 A0  2780  ONE  LDX #$A0  (160 PULSES)
0957-  20 66 09  2790  JSR PULSES
095A-  A9 24  2800  LDA #$24  3.82 MSEC
095C-  20 A8 FC  2810  JSR WAIT
095F-  A9 05  2820  LDA #$05  .146 MSEC
0961-  20 A8 FC  2830  JSR WAIT
0964-  CA  2840  MOR  DEX
0965-  D0 DC  2850  BNE OUT1  KEEP GOING
0967-  60  2860  RTS  DONE HERE
2870  *-----------------------------------------------------------------------
2880  * OUTPUT Y PULSES AT 40 KHZ
2890  *-----------------------------------------------------------------------
0968-  CD 5E CO  2900  PULSES  CMP #$C05E  TRANSDUCER ON
096B-  EA  2910  NOP
096C-  EA  2920  NOP
096D-  EA  2930  NOP
096E-  EA  2940  NOP
096F-  CD 5F CO  2950  CMP #$C05F  TRANSDUCER OFF
0972-  CD FF FF  2960  CMP #$FFFF  JUST A DELAY
0975-  88  2970  DEY
0976-  D0 F0  2980  BNE PULSES  KEEP GOING
0979-  60  2990  RTS  DONE HERE
3000  .EN

SYMBOL TABLE

084C- AGAIN
0822- ALARM
08FC- BEGIN
0801- CLOCK
08AA- CONT
082D- DATA
082E- DATA1
082F- DATA2
081B- DISABL
0860- DONE
0898- DOOR
081F- DOORF
080A- ENABLE
0856- FOUND
0824- FROM
08B8- GOTTIT
0859- PAUSE
08A4- INSIDE
0830- INTER
0911- MAIN
0964- MOR
086C- MORE
08B2- NEXT
0825- NUM
08E3- OFF
08D5- OK
0855- ONE
08B0- ONEWAS
0941- OUT
0943- OUT1
081E- OUTF
086A- OVER
0850- PAUSE
0868- PULSES
081D- SEC F
0868- SECOND
0892- SECUR
08F9- SSONIC
0CF6- SWPORT
0829- TABLE
0800- TEMCLK
0823- TO
0901- TWO
08B9- WAIT
0820- WINDF
089E- WINDOW
0908- WT
0848- ZERO
Appendix C

Clock Set and View Program

1 LIST
0 REM FIRST USE POKES TO
1 REM LOAD SMALL MACHINE LANGUAGE PROGRAM
2 FOR I = 1 TO 9: READ X: POKE 768 + I, X: NEXT I
9 REM SET UP PORTS AND DDR ADDRESSES
10 A = 12 * 16 + 3 + 15 * 16 + 0: C = A + 1
20 AB = A + 2: CB = C + 2
21 REM !-----------------------!
22 REM ! MAIN PROGRAM STARTS!
23 REM ! HERE
24 REM !-----------------------!
25 HOME
30 PRINT "CLOCK SET AND VIEW PROGRAM"
35 VTAB 5
40 PRINT "1. VIEW CLOCK"
45 PRINT "2. SET CLOCK"
50 PRINT : INPUT "WHICH "; NUM
55 ON NUM GOTO 1000, 2000
60 GOTO 25
80 REM ------------------------------
90 REM GENERAL PURPOSE SUBROUTINE TO
91 REM READ AND WRITE
92 REM BASIC IS SLOW ENOUGH TO
93 REM PROVIDE THE CORRECT TIMING
94 REM FOR THE 5832 CHIP
95 REM ------------------------------
100 POKE CB, 0: POKE AB, 128: POKE CB, 4: POKE
101 AB, 0: REM SET UP HOLD PORT
110 POKE C, 0: REM DDR
111 IF Z = 0 THEN POKE A, 255: REM WRITE
112 IF Z = 1 THEN POKE A, 240: REM READ

143
113 POKE C,4: REM BACK TO OUTPUT
120 POKE AB,128: REM TURN CHIP ON
130 POKE A,(128 + Y * 16 + D): REM Y=ADDR
   D=WRITE DATA (0 IF READ)
140 POKE 768, PEEK (A): CALL 769: X = PEEK
   (768): REM MASK UPPER NIBBLE TO ZERO
145 POKE A,0: REM RESET READ
150 POKE AB,0: REM TURN CHIP OFF
170 RETURN
800 REM ! ------------------------!
810 REM !
920 REM !----- - ------------------!
1000 REM READ ALL ADDRESSES
1001 VTAB 10
1002 PRINT "D HR MI SC"
1005 Z = 1; D = 0
1010 FOR Y = 0 TO 6
1020 GOSUB 100
1030 T(Y) = X
1040 NEXT Y
1050 VTAB 20
1060 PRINT T(6); T(4); T(3);T(2); T(1); T(0); ""
1070 GOTO 1005
1900 REM !------------------------!
1910 REM ! SET THE TIME !
1920 REM ! NOTE: SECONDS CANNOT !
1930 REM ! BE SET WITH !
1940 REM ! THIS CHIP !
1950 REM !------------------------!
2000 REM SET TIME
2001 PRINT : PRINT "MOVE READ/WRITE SWITCH TO WRITE POS."
2002 PRINT : PRINT "USE 0 FOR SUNDAY, 1 FOR MONDAY ETC."
2003 PRINT : PRINT "ADD 4 TO H10 FOR PM"
2004 PRINT : PRINT "ENTER 0-9 FOR OTHER DIGITS"
2005 PRINT : PRINT "SEE CHAPTER 6 FOR MORE DETAILS ON CLOCK"
2010 Z = 0: REM SET UP WRITE
2020 PRINT "INPUT DAY.H10,H1,M10,M1"
2030 FOR Y = 6 TO 0 STEP - 1
2035 IF Y < 2 THEN D = 0: GOTO 2050
2040 INPUT D
2050 GOSUB 100
2060 NEXT Y
2070 Z = 1; D = 0
2080 END
3000 DATA 173,0,3,41,15,141,0,3,86,200
Time Table Editor

Appendix

1
2 LIST

1 D$ = CHR$(4)
5 DIM TT(3,20)
10 REM !-----------------------------!
12 REM ! TIME TABLE MAINTENANCE !
14 REM ! PROGRAM BY !
16 REM ! JOHN BLANKENSHPH !
18 REM !-----------------------------!
20 REM
25 REM
80 REM !-----------------------------!
90 REM ! MAIN PROGRAM MENUE !
95 REM !-----------------------------!
100 TEXT
110 HOME
120 PRINT "TIME TABLE MAINTENANCE"

130 VTAB 5
140 PRINT "1. LOAD A TABLE"
145 PRINT "2. EDIT TABLE IN MEMORY"
150 PRINT "3. SAVE PRESENT TABLE"
155 PRINT "4. CREATE A NEW TABLE"
157 PRINT "5. END"
160 VTAB 20: PRINT ""
165 VTAB 20: PRINT "WHICH ": GET A$
170 A = VAL(A$)
171 IF A = 5 THEN HOME : END
175 IF A < 1 OR A > 4 THEN 160
180 ON A GOSUB 1000,2000,3000,4000
190 GOTO 100
1000 REM !---------------------------------!
1005 REM ! LOAD A TABLE FROM DISK !
1010 REM !---------------------------------!
1020 HOME
1030 PRINT " LOAD TABLE"
1040 VTAB 3
1050 INPUT "WHAT TABLE NAME (JUST RETURN TO VOID) ";TN$
1055 IF TN$ = "" THEN RETURN
1060 PRINT DS"OPEN "TN$
1070 PRINT DS"READ "TN$
1080 FOR J = 1 TO 20
1085 FOR I = 1 TO 3
1090 INPUT TT(I, J)
1100 NEXT I, J
1110 PRINT DS"CLOSE"
1120 RETURN
1130 HOME
2000 REM !---------------------------------!
2005 REM ! EDIT PRESENT TABLE !
2100 REM !---------------------------------!
2105 HOME
2110 IF A$ = "N" THEN RETURN
2120 GOTO 2100
3000 REM !---------------------------------!
3005 REM ! SAVE TABLE TO DISK !
3100 REM !---------------------------------!
3105 HOME
3110 IF A$ = "N" THEN RETURN
3120 GOTO 3000
4000 REM !---------------------------------!
4005 REM ! CREATE A NEW TABLE !
4100 REM !---------------------------------!
4200 HOME
4230 PRINT "WE MUST ENTER THREE NUMBERS FOR EACH LINE IN THE TABLE": PRINT
4240 PRINT "THE ENTRIES ARE"
4250 PRINT " 1. EVENT TIME (24 HR FOR HAT)"
4260 PRINT " 2. USE (1-Both 2-Home 3-Away)"
4270 PRINT " 3. TASK NUMBER (1-20)"
4280 PRINT : PRINT
4282 PRINT "ENTER EVENTS IN CHRONOLOGICAL ORDER"
4084 PRINT "INPUT ZERO'S FOR UNUSED ELEMENTS"
4086 PRINT : PRINT
4088 FOR J = 1 TO 20
4100 PRINT "LINE "J" (ET, U, T) "
4105 INPUT TT(1, J), TT(2, J), TT(3, J)
4110 NEXT J
4120 RETURN
5000 REM !-----------------------------!
5005 REM ! PRINT PRESENT ARRAY !
5010 REM !-----------------------------!
5020 TEXT : HOME
5030 POKE 34, 21
5035 INVERSE
5040 PRINT "1 2 3"
5045 NORMAL
5050 FOR J = 1 TO 20
5060 GOSUB 7000
5070 PRINT
5080 NEXT J
5090 RETURN
6000 REM !-----------------------------!
6005 REM ! PRINT A WITH LENGTH K !
6010 REM !-----------------------------!
6020 A$ = STR$ (A)
6030 IF LEN (A$) < K THEN A$ = "" + A$: GOTO
6030
6040 PRINT A$;
6050 RETURN
7000 REM !-----------------------------!
7004 REM ! PRINT LINE J OF TABLE !
7008 REM !-----------------------------!
7009 INVERSE
7010 K = 2: A = J: GOSUB 8000
7011 NORMAL
7020 FOR I = 1 TO 3
7030 K = 10: A = TT(I, J): GOSUB 8000
7040 NEXT I
7050 RETURN
Appendix

Vocabulary
Generator and
Tester

1 LIST
2 TEXT
3 IF PEEK (104) < > 25 THEN POKE 103, 20
   1: POKE 104, 25: POKE 6600, 0: PRINT CHR$ (4) "RUN VOCAB.CREAT$&SAY"
4 REM LINE 2 MAKES SURE LOMEM IS SET TO 6
5 600
6 SLO$ = 1
7 D$ = "": REM CTL D
8 I$ = "": REM CTL I
9 V$ = "": REM CTL V
10 HOME: PRINT "SPEECH LINK TEST BY JOHN
      BLANKENSHIP"
11 VTAB (5)
12 PRINT "1. INITIALIZ$"
13 PRINT "2. TRAIN WORDS"
14 PRINT "3. RECOGNIZE WORD$"
15 PRINT "4. SAVE VOCABULARY"
16 PRINT "5. YES AND NO"
17 PRINT "6. END"
18 INPUT "WHICH "; A
19 ON A COSUB 1000, 2000, 3000, 3500, 3800, 989
20 GOTO 10
300 REM **********************
301 REM * TRAIN W$ *
302 REM **********************
310 PRINT W$
320 PRINT D$""FR0""SLOT
330 PRINT W$
340 PRINT D$""FR0""
ERR = PEEK (WAA + 9)
RETURN
REM *********************************************
REM * RECOGNIZE $*
REM *********************************************
PRINT D$"IN#"SLOT
INPUT W$
PRINT D$"IN#0"
ERR = PEEK (WAA + 9)
RETURN
999 END
REM *********************************************
REM * INITIALIZATION *
REM *********************************************
WAA = 3000
HI = INT (WAA / 256)
LO = WAA - HI * 256
POKE (1144 + SLOT),LO
POKE (1272 + SLOT),HI
TAA = WAA + 557
HI = INT (TAA / 256)
LO = TAA - HI * 256
POKE (WAA + 2),LO
POKE (WAA + 3),HI
PRINT D$"PR#"SLOT
PRINT I$
PRINT D$"PR#0": REM SAME FOR D$
RETURN
REM *********************************************
REM * TRAIN WORDS *
REM *********************************************
PRINT "PLEASE SAY THE FOLLOWING WORDS
FOR I = 1 TO 1000: NEXT I
RESTORE
READ W$
IF W$ = "END" THEN 2060
GOSUB 300
GOTO 2030
RETURN
REM *********************************************
REM * RECOGNIZE WORDS *
REM *********************************************
PRINT "PLEASE SAY SOME WORDS (10) AND
I WILL PRINT WHAT I THOUGHT YOU SAID"
FOR I = 1 TO 10
GOSUB 400
PRINT W$
NEXT I
RETURN
REM *********************************************
REM * SAVE VOCABULARY *
REM *********************************************
PRINT "WHAT NAME DO YOU WISH TO SAVE
THIS VOCABULARY UNDER (JUST RETURN
N TO VOID)"
PRINT : INPUT NAME$
IF LEN (NAME$) = 0 THEN RETURN
PRINT D$"BSAVE "NAME$",A"TAA",L3000"
3800 REM ********************
3805 REM * YES AND NO  *
3810 REM ********************
3812 HOME
3814 PRINT "THIS WILL PERFORM 10 TESTS OF"  
3815 PRINT "THE YES AND NO INPUT"
3820 PRINT D$"PR#"SLOT
3830 PRINT V$
3835 PRINT D$"PR#0"
3840 FOR I = 1 TO 10
3845 GOSUB 400
3850 PRINT W$
3855 NEXT I
3860 PRINT D$"PR#"SLOT
3865 PRINT V$
3870 PRINT D$"PR#0"
3900 RETURN
4000 REM ********************
4004 REM *WORDS FOR TRAINING*
4005 REM * LAST WORD = END  *
4010 REM ********************
4020 DATA OFFICE, OFFICE, OFFICE!
4030 DATA STEREO, STEREO, STEREO!
4040 DATA BEDROOM, BEDROOM, BEDROOM!
4050 DATA DEN, DEN, DEN!
4060 DATA KITCHEN, KITCHEN, KITCHEN!
4070 DATA FLOODLIGHTS, FLOODLIGHTS, FLOODLIGHTS!
4080 DATA HEAT, HEAT, HEAT!
4090 DATA TIME PLEASE, TIME PLEASE, TIME PLEASE!
4100 DATA TELEPHONE, TELEPHONE, TELEPHONE!
4110 DATA ARE YOU BUSY, ARE YOU BUSY, ARE YOU BUSY!
4120 DATA THIS IS WANDA, THIS IS WANDA, TH
4130 DATA GOODNIGHT, GOODNIGHT, GOODNIGHT!
4140 DATA BYE BYE, BYE BYE, BYE BYE!
4150 DATA TURN IT ON, TURN IT ON, TURN IT ON!
4160 DATA SHUT IT OFF, SHUT IT OFF, SHUT IT OFF!
4170 DATA DIM IT, DIM IT, DIM IT!
4180 DATA NEVER MIND, NEVER MIND, NEVER MIND!
4190 DATA BILL, BILL, BILL!
4200 DATA WANDA, WANDA, WANDA!
4210 DATA TOCCO HILLS, TOCCO HILLS, TOCCO HILLS!
4220 DATA VILLAGE TWIN, VILLAGE TWIN, VILLAGE TWIN!
4230 DATA END
The HELLO Program

```
10 HOME
20 PRINT "HOME CONTROL PROGRAM DISKETTE"
30 PRINT "COPYRIGHT BY"
40 PRINT "JOHN BLANKENSHIP"
50 POKE 1012,0 : REM RESET POWER UP BYTE
60 POKE 103,201 : POKE 104,25 : POKE 6600,0
70 REM LOMEM NOW SET TO 6600
80 POKE 49393,0 : REM RESET SANITY TIMER
90 PRINT
100 PRINT CHR$(4)"RUN HOME CONTROL"
```
Micromodem II
Hayes Microcomputer Products
5835 Peachtree Corners East
Norcross Ga 30092

Type 'N Talk Speech Synthesizer
Votrax
500 Stephenson Highway
Troy MI 48084

Microvox Speech Synthesizer
40-Khz Ultrasonic Transducer
Micromint
561 Willow Avenue
Cedarhurst NY 11516

5832 Clock chip and crystal
Miscellaneous semiconductors and electronic parts
Concord Computer Products
1971 So. State College
Anaheim CA 92806
and
JDR Microdevices
1224 S. Bascom Avenue
San Jose CA 95128

Mura WMS-49 Wireless Microphone
Herbach & Rademan
401 East Avenue
Philadelphia PA 19134
and
Mura
177 Cantiague Rock Road
Westbury NY 11590
Yours can be the first APPLE house on the block!

Here is a book that will show you how to save time and money by using your computer to control your home: the security, the lights, the heat, the telephone, and much more.

With John Blankenship's system, your house can accept verbal commands and respond with its own voice. It does not need human instruction and performs many useful tasks on its own. Once you get used to an intelligent house you will wonder how you ever got along without one.

Even though most items in the APPLE house can be readily purchased, the author shows how you can save even more money by building some from scratch. He also points out that you can substitute equipment you already own because of the system's modularity.

Although written with an APPLE computer in mind, the principles discussed can easily be transferred to other computer systems.