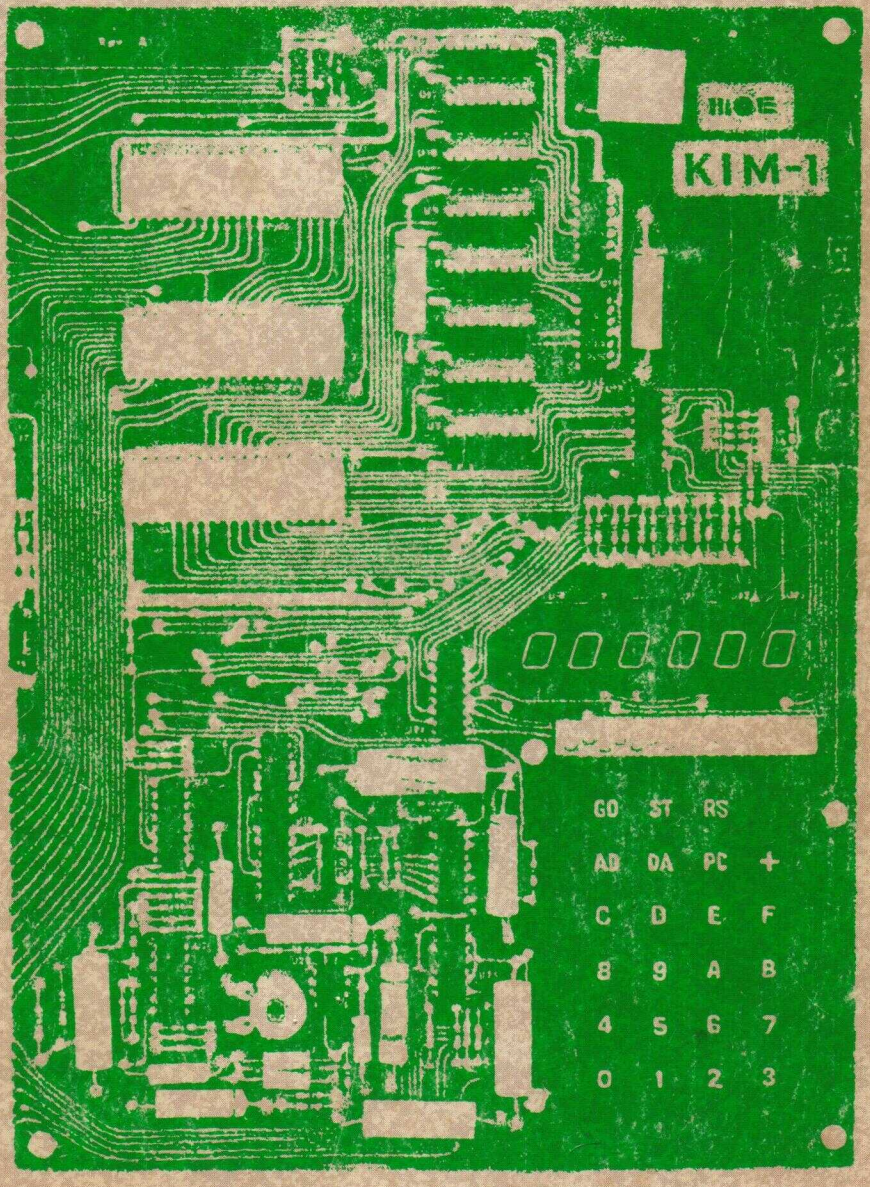
The First Book of KIM

Jim Butterfield, Stan Ockers, and Eric Rehnke

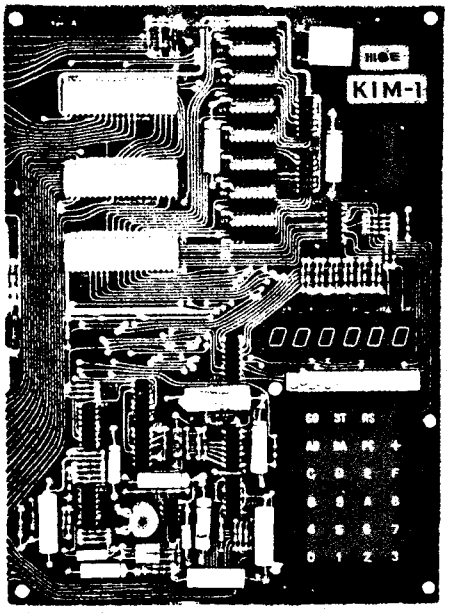


HAYDEN

The First Book of KIM

Edited by

JIM BUTTERFIELD • STAN OCKERS • ERIC REHNKE



HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

Dedicated to the person who just purchased a KIM-1 and doesn't know what to do with it . . .

Individual programs in this book were contributed by the various authors without copyright restrictions.

These programs may be used or copied without restriction. It is, however, common courtesy to quote author and source when copying; and a copy of any published material should be sent directly to the author.

In general, program authors welcome comments, suggestions or revisions to their programs. Depending on circumstances, they may not find it possible to reply to all correspondence.

If you develop a program that you'd like to share with other KIM users, send it in to KIM/6502 User Notes, 109 Centre Avenue, W. Norriton, Pennsylvania 19401. It might appear in User Notes . . . and even in a future *Book of KIM.*

ISBN 0-8104-5119-0
Library of Congress Catalog Card Number 78-53963

Printed in the United States of America

1 2 3 4 5 6 7 8 9 PRINTING

78 79 80 81 82 83 84 85 86 YEAR

IN THIS BOOK YOU'LL FIND:

A BEGINNER'S GUIDE TO KIM PROGRAMMING: 5

guidelines which take the absolute beginner, step by step, through the fundamentals of understanding and writing programs.

RECREATIONAL PROGRAMS: 23

dozens of programs including games, diversions and educational programs; fully detailed so that you can learn from the programming techniques as well as have fun. All programs run on the basic KIM-1 system.

DIAGNOSTIC AND UTILITY PROGRAMS: 114

to help you test your KIM computer — to help you test other devices, such as cassette recorders — and to make your KIM a more powerful machine.

EXPANDING YOUR KIM: 143

guidelines on how to expand your KIM from the basic small-but-powerful KIM-1 system to a huge-and-super-powerful machine; understanding the jargon; seeing what's available in both hardware and software.

CONNECTING TO THE WORLD: 155

an introduction to the methods by which KIM can read or sense other devices, and can in turn control other mechanisms.

POTPOURRI: 166

other useful pieces of information about your KIM system; reference material, hints, etc.

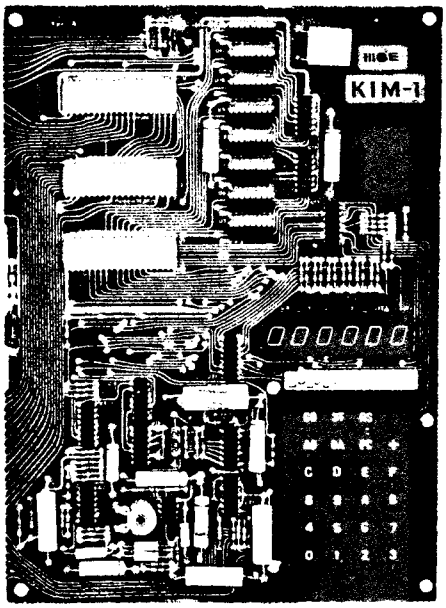
Acknowledgments

Thanks to all who have supported the KIM-1/6502 User Notes, from which much of this material was taken. A special thanks to Earl Nied for the use of his KIM-interfaced Selectric.

The KIM-1 microcomputer is manufactured by Commodore/MOS Technology, 950 Rittenhouse Road, Norristown, Pennsylvania 19401. It may be obtained directly from the manufacturer or from many hobbyist computer retail stores. At the time of writing, the complete KIM-1 system (less power supply) sells for $245.

All programs in this book run on the basic KIM-1 system; two require an audio amplifier.

A BEGINNER'S GUIDE TO KIM PROGRAMMING



A BEGINNER'S GUIDE TO KIM PROGRAMMING.

Running programs can be fun. But writing programs can be even more fun .. and exasperating, and exhilirating, too!

When you get the hang of it - and it will take time - you'll be able to create your own games, diversions, or useful routines. This section tries to introduce you to the mechanics of programming, so you can find your own way at your own speed.

Don't be afraid to use ideas from other parts of this book. If you like, try changing parts of a program or two and see what happens. And you can borrow whole sections of coding from another program if it does something you want.

LOOKING AT MEMORY

Random Access Memory.

If you've just turned your KIM system on, press the RS (Reset) button to get things started. Hit the following keys: AD (for ADDRESS) 0 0 0 0. You've just entered the address of memory cell 0000, the lowest numbered one in memory. The display will show 0000 (the number you entered) on the left. On the right, you'll see the contents of cell 0000: it will be a two digit number. That number might be anything to start with; let's change it.

Press key DA (for DATA). Now you're ready to change the contents of cell 0000. Key in 44, for example, and you'll see that the cell contents have changed to 44.

Hit the + button, and KIM will go to the next address. As you might have guessed, the address following 0000 is 0001. You're still in DATA mode (you hit the DA key, remember?), so you can change the contents of this cell. This time, put in your lucky number, if you have one. Check to see that it shows on the right hand part of the display.

This kind of memory - the kind you can put information into - is called RAM, which stands for Random Access Memory. Random access means this: you can go to any part of memory you like, directly, without having to start at the lowest address and working your way through. Check this by going straight up to address 0123 and looking at its contents (key AD 0 1 2 3); then address 0000 (key AD 0 0 0 0), which should still contain the value 44 that we put there.

Hexadecimal Numbers

Now that you're back at address 0000, let's step through several locations using the + key. Don't worry about contents too much. 0001 will still contain your lucky number, of course, but keep stepping with the + key until you reach 0009. What will the next address be? Most people would think that the next number should be 0010, and that would be correct if KIM used the familiar decimal numbering scheme. But KIM still has six more digits to go past 9, because it uses a computer numbering scheme called Hexadecimal. Hit the + key and you'll see address 000A come up.

Don't let the alphabetic confuse you - to KIM, A is just the digit that comes after 9. And there are more digits to come. Keep pressing the + button and you'll see that A is followed by B, C, D, E and F. Finally, after address 000F, you'll see address 0010 appear.

A word about pronunciation: don't call address 0010 "ten"; say "one zero" instead. After all, it isn't the tenth value after 0000; it's really the sixteenth (the word Hexadecimal means: based on sixteen).

If you don't understand why the letters appear, don't worry about it too much. Just understand, for the moment, that the alphabetics represent genuine numbers. So if you're asked to look at address 01EB, you'll know that it's a legitimate address number like any other. And if you're told to store a value of FA in there, go right ahead - you're just putting a number into memory.

When you get time, you'll find lots of books that explain Hexadecimal numbering in detail. There's even an appendix in your 6502 Programming Manual on the subject. It makes important and worth-while reading. But for now, just recognize that although the numbers may look a little funny, they are still exactly that: numbers.

Read Only Memory

So far, we've talked about one kind of memory, called RAM. You recall that we said that you can store numbers into RAM.

There's another kind of memory in KIM, but you can't store numbers there. It's called ROM, for Read Only Memory. This kind of memory contains fixed values that cannot be changed.

For example, let's look at address 1C3A (key AD 1 C 3 A). You'll see the value 18, and that value never changes. Try it: press DA 6 6 to try to change the contents to 66. See how it won't work?

ROM contains pre-stored programs which do important things like lighting the display, detecting keyboard input, and reading or writing your cassette tape. These programs are called the Monitor. In fact, the name KIM stands for Keyboard Input Monitor in recognition of the importance of these programs. We'll talk briefly about the Monitor programs later.

Special Memory Locations

A few addresses in KIM are connected to things that aren't really memory at all. You can read up on them in the KIM User Manual when you're ready; we'll just point out a few examples here.

If you try to store a number into address 1700, for example, you might find that instead of storing the value, KIM will convert it to voltages and deliver these voltages to certain pins on your Application Connector at the edge of the board! Another example: address 1704 connects to a very fast timer - look at that address and you'll see "time going by" as a blur!

MINI-PROGRAM A: Swap the contents of two locations

This is our first beginner's program.

It doesn't do much: just exchanges the contents of locations 0010 and 0011. But it's a start, and you'll learn quite a few things about getting KIM programs going.

CAUTION: Before running this or any other program, be sure that you have set the contents of the KIM "vector" locations as follows:

Set address 17FA to 00
Set address 17FB to 1C
Set address 17FE to 00
Set address 17FF to 1C

The first two locations are needed so that your SST switch and ST key will work right. The last two make the BRK (break) instruction behave properly. YOU MUST ALWAYS SET UP THESE LOCATIONS AS SOON AS YOU TURN ON YOUR KIM SYSTEM.

Loading the Program

We'll take time to describe how the program works later. First, let's see how to load it. A listing usually looks something like this:

```
0200 A5 10   START LDA 10   address 10 to A
0202 A6 11         LDX 11   address 11 to X
0204 85 11         STA 11   A to address 11
0206 86 10         STX 10   X to address 10
0208 00            BRK      stop the program
```

The business end of the program - the part that goes into the computer - is the group of numbers on the left hand side. The stuff on the right helps explain what the program does.

If you look at the numbers on the left, you'll see that the first one, 0200, looks like an address. That's exactly what it is, and we can start by entering it with AD 0 2 0 0. The next number is A5, and that will be its contents. So hit DA A 5, and the display will confirm that we've put it in.

Keep going on the same line. Each line of the program listing may contain more than one value - for more than one address.

The next value is 10, and it needs to go into 0201. You don't need to enter the address. Just hit the + key and there you are - enter 1 0 and you've got it. Notice you didn't need to hit DA; you stay in Data mode until you press the AD key. Continue to the next line: just hit + A 6 + 1 1 and keep going until you've put the 00 in location 0208. Congratulations! You've loaded your first program. Now go back and check it for correctness. Hit AD 0 2 0 0 and use the + key to step through and check the values.

Now let's run the program and see if it works. First, look at the contents of addresses 0010 and 0011. Make a note of them; when the program runs, it will swap those two values.

Keep in mind that loading the program doesn't make anything happen. You have to run it to do the job - and that's what we'll do next.

Running the Program

Set address 0200. That's where the first instruction in the program is located - you may have noticed that it's marked START in the listing. Now the display shows 0200 A5, and we're ready to go. So - hit GO. And the program will run.

Doesn't take long, does it? The display will have changed to 020A xx. If the display shows any other address, something's wrong. Check that your SST switch is off (left), that the program is entered correctly, and that your vectors are OK.

Your program ran in less than a fifty thousandth of a second. No wonder you didn't see the display flicker.

Now check that the program did indeed run correctly by looking at the contents of locations 0010 and 0011. You'll see that they have been exchanged.

How it works

Inside the Central Processor (the heart of the computer) are several temporary storages called registers. You can LOAD many of these registers with the contents of memory; and you can STORE the contents of the registers into memory. The two registers we are using here are called A and X.

If we Load A from address 10, A now contains a copy of the contents of 0010. Location 0010 itself won't be changed; it will also contain that number. We do the same thing when we Load X from address 0011.

Now our A and X registers contain copies of the numbers in 0010 and 0011 respectively. If we Store A into address 0011, that address will now contain a copy of the value in A - which was originally the contents of address 0010, remember? Finally, we Store X into 0010 to complete the swap.

Look at the listing again. On the right hand side, we have the program exactly as we have described it, but abbreviated. You can see that LDA means Load A and so forth. The BRK (Break) at the end stops the program.

Step by Step

Let's go through the program a step at a time - literally. Maybe you're satisfied that it works. Even so, follow this procedure. It will show you how to test any KIM program.

First go back to addresses 0010 and 0011 and put a couple of brand new numbers there. This will help you see the computer operating.

Now set address 0200 again, but don't press GO yet.

We're going to "Single Step" our program, and see every instruction work. So slide the SST (Single STep) switch over to the right ... and then read the next section carefully.

Seeing the Registers

Registers A and X, plus quite a few we haven't talked about, are inside the 6502 microprocessor chip. There's no way you can view them - they are buried deep within the electronics.

To help you out, the KIM Monitor system will write out a copy of these registers into memory where you can inspect them. The contents of the A register may be seen at address 00F3, and the contents of the X register are at 00F5.

Don't be confused: These locations are not the actual registers, just copies made for your convenience. But it's a great convenience, for it allows you to see everything that's going on inside the microprocessor.

A Small Step for a Computer, but ...

If you're set up at location 0200 and your SST switch is on, hit the GO button once. The display will show 0202. That means: instruction at 0200 completed, ready to do the one at 0202.

Okay, let's check everything in sight. The first instruction was to load the A register, right? Enter address 00F3 and check that its contents (which correspond to the contents of A) are indeed the value from address 0010. If you like, look at 0010 and confirm that it hasn't changed.

Now for a clever KIM touch. If you're ready to proceed with the next instruction, hit PC (for Program Counter) and you'll find yourself back at address 0202, ready to perform the next instruction.

You've executed one instruction, performed one program step. Remember this: No matter how complex the program, it always operates one simple step at a time. And now you know how to check out each step, individually.

Hit GO and execute one more instruction. Check it out - remember that you'll find X at address 00F5.

From this point, find your own way through the last two instructions. Don't bother about the BRK (Break); it just stops the program. As the two registers are stored, you'll want to check that the memory addresses have been changed as expected.

Summary

The most important things that you've learned about coding are:

--the BRK (code 00) command stops the program;
--the SST switch causes a single instruction to be executed;
--the internal registers can be viewed.

BUT YOU MUST SET YOUR VECTORS PROPERLY (see the beginning of this section) OR NONE OF THE ABOVE WILL WORK!

A complete list of the register image addresses can be found in the KIM User Guide on page 39, Fig. 3-13 - when you need it.

From here on, you don't have to take anybody's word for any KIM operation. You can go to your KIM, set SST, and try it for yourself.

Exercises

1. Can you change the program so that it swaps the contents of locations 0020 and 0021?

2. Billy Beginner wrote the following program to swap the contents of locations 0010 and 0011:

```
0200 A5 10   START LDA 10   put 0010 into A
0202 85 11        STA 11   store A to 0011
0204 A6 11        LDX 11   put 0011 into X
0206 86 10        STX 10   store X to 0010
0208 00           BRK      stop
```

It didn't work. Can you see why?

3. Can you write a program to take the contents of address 0010 and place the same value in locations 0011, 0012, and 0013?

MINI-PROGRAM B: Setting many locations to zero

Here's the program:

```
0200 A9 00   START LDA #0     value 0 into A
0202 A2 09        LDX #9     start X at 9
0204 95 30   LOOP  STA 30,X   zero into 0030+X
0206 CA            DEX        decrease X by 1
0207 10 FB         BPL LOOP   back if X positv
0209 00            BRK        stop the program
```

This program, when you load and run it, will set the value of the ten locations from 0030 to 0039 to zero.

We can't give you a whole programming course here. Hopefully, you'll use the Programming Manual and the single-step feature to trace out exactly what the program does. But here are a few highlights:

When we load registers A and X in the first two instructions, we don't want to load the contents of a memory location. Instead, we want the actual values 0 and 9. To do this, we use a new kind of addressing called IMMEDIATE addressing.

Immediate addressing, when we use it, says "Don't go to memory - use this value." Immediate addressing can be spotted two ways. First, note the # sign that we use in writing the program: that signals that we are using immediate mode adressing. Secondly, you may have noticed that the computer instruction (called the Op Code) has changed: the previous program used code A5 to mean LDA; now we're using A9, which also means LDA but signals immediate addressing.

You can - and should - use the SST feature to check that immediate addressing works as advertised.

The instruction at 0204 uses the X register for INDEXING. That means that instead of storing the A value in address 30, the computer first calculates an effective address by adding the contents of the X register to the "base address" of 30. Since X contains 9 the first time through, the effective address will be 30+9 or 39 - and that's where we store our A value of 00. Later, X will be decreased to a value of 8, so we'll store into address 38.

Indexing seems complicated, but remember that it's a very powerful feature of KIM. Try to get the hang of it; it's well worth the effort.

The DEX instruction (Op Code CA) is the one that decreases X from 9 to 8 (and later to 7, 6, 5 and so on). Eventually, as this part of the program is automatically repeated, X will reach a value of 00. Finally, when we decrement X one more time, X will go to value FF, which KIM "sees" as a negative number, kind of like the value -1. KIM views all values in the range 80 to FF as negative - when you're ready, the Programming Manual will tell you more.

The BPL instruction at line 0207 is a CONDITIONAL TEST. BPL means Branch plus. If the result of our previous operation (Decrement X) gives us a positive, or plus, number, we will branch back to address 0204 and repeat the instructions from that point. The X values of 9, 8, 7 ... down through 0 are all positive or plus; so each time we'll go back and set one more location in memory to value zero. Finally, X becomes equal to value FF - a negative number. So in this case, BPL won't branch: the "plus" or "positive" condition isn't satisfied.

This last time, since BPL doesn't take us back, we proceed to the following instruction, BRK, which stops the program. That's OK because we've done our job of setting addresses 0030-0039 to value zero.

Single Step the program carefully, checking the value of X from time to time (location 00F5, remember?). Satisfy yourself that you can see it working.

By the way, that funny address on the branch instruction (FB) is a special kind of addressing mode called RELATIVE addressing. All branches use it; it's worth reading up on.

Exercises

1. Can you change the program to place value 55 in the above locations?

2. Can you change the program to place value 00 in locations 0030 to 0037?

3. Can you change the program to place value FF in locations 00A0 to 00BF?

INTERLUDE - PROGRAM TESTING

You've met one very powerful tool for checking out programs - the Single Step mode of operation. Let's review it and talk about a few others.

The SST mode is especially useful because you can pause between instructions and look at memory or registers. The register values are copied into memory locations from 00EF to 00F5, and while they are not real registers, just copies, they are just as good for testing purposes. Not only can you look at them, you can change them to new values. This ability to change a register can be handy in solving the "what if ... " type of question, or shortening testing of a loop.

For example, if you are single-stepping through mini-program B and you don't want to go around the loop a full ten times, you might use this trick. Go around a couple of times to get the loop started, and then change X (00F5) to a much lower value, say 1 or 2. Go back to single-stepping. A couple more turns around the loop, and you're out. Using this method, you won't have set the whole ten locations to zero, of course. But you will see that the loop itself is working right.

The Inserted BRK (Break)

Sometimes SST seems slow. You might have a long program, and you're sure that the first part is working. What you want is a way to run directly through the first bit, and then stop and single-step the rest.

It's not hard. Decide where you want the program to stop, so you can start single-stepping. Then put a BRK command, code 00, at that point.

You'll have to wipe out a live instruction, of course, but that's OK. You can put it back after the halt has happened.

Let's try doing that on mini-program B. Let's say we want to run straight through to the BPL instruction at 0207, and then single-step from that point on.

Change 0207 (previously 10) to value 00, the BRK command. Now go to the beginning of the program (0200), be sure SST is off, and hit GO. You'll see 0209 00 on the display, which tells you that the halt at 0207 has worked. Now go back to 0207, put the value of 10 (for BPL) back in, set the SST switch on, and you're ready to step. Easy? You bet - and you can save lots of time this way in testing big programs.

No Operation (NOP, code EA)

It sounds funny, but a very handy instruction is one that doesn't do anything. When the microprocessor encounters Op Code EA (NOP), it does nothing - just passes on to the next instruction.

The biggest use of the NOP instruction is to take out another instruction that you don't want any more; or to leave room in the coding to add another instruction later if you need to.

Some programmers write their programs in sections, and at first they put a BRK instruction between each section. That way, when they are testing, the program will stop after each part, and they can check to see that each part runs OK. When they are finished testing, they change the BRK's to NOP's and the program will run straight through.

The ST (Stop) Key

When everything is under control in program testing, you won't need the ST key. But sometimes the program 'gets away' on you - and the only way to find out what it's doing is to use this key.

Let's wreck mini-program B by wiping out the DEX instruction. We'll do this by replacing it with a NOP; so write value EA into location 0206. What will happen?

When we run the program, the X register will never change from its starting value of 9 because we don't have a DEX instruction. So the program will keep branching back to LOOP forever, and it will never stop. We've created this situation artificially, of course, but it could have happened by oversight when we were writing the program.

Set address 0200, SST off, and hit GO. Everything goes dead. Our program is running but it will never stop. Meanwhile, the display is dark. This time we know why it's happening. But if we didn't, how would we solve it?

Press ST - stop - and the computer will freeze. The display will light showing the next instruction we were about to execute. If we wanted to pinpoint the trouble, we could flip over to SST now and track the problem down, step by step.

A last comment on the ST button: If the display goes dark and pressing ST doesn't relight it, the computer has a different problem. It has gone berserk due to a completely illegal Op Code. Press the RS (Reset) button; now you'll need to start over and use the BRK and SST features to track down the trouble.

MINI-PROGRAM C: Displaying values

KIM has a 6-digit display. You can show information on the display quite easily, if you know how.

In the KIM Monitor programs there are several packages called subroutines that you can call upon to do certain jobs. You could write the same coding for these jobs yourself; but use the Monitor subroutines to save time and trouble.

When you give the command JSR SCANDS (coded 20 1F 1F), the Monitor will briefly light the display with the data it finds in addresses 00FB, 00FA, and 00F9. That's three locations, each displaying as two digits, so the full six-digit display is filled.

"Briefly" means exactly that. The display lights for a split second. To get a steady display, you must repeat the JSR SCANDS command over and over again. Use a loop, of course; no point in filling up your program with JSR SCANDS instructions.

You should also know that when you call this Monitor subroutine, the contents of your registers are wiped out. So if you have something important in the A register that you will want to use after giving JSR SCANDS, be sure to put it safely somewhere in memory or you'll lose it. The same goes for other registers like X and Y.

Here's a simple program to show 0000 00 on the display. Note that we must put the value 00 into addresses FB, FA, and F9 before we call JSR SCANDS.

```
0200 A9 00    START  LDA #0       zero into A
0202 85 FB           STA POINTH   first 2 digits
0204 85 FA           STA POINTL   next 2 digits
0206 85 F9           STA INH      last 2 digits
0208 20 1F 1F  LOOP  JSR SCANDS   light up!
020B 4C 08 02        JMP LOOP     do it again
```

This program never ends, so eventually you'll have to stop it with the RS or ST keys. See how the last instruction jumps back to address 0208 so the display is lit continuously? Another interesting point: see how the jump address at 020B is "backwards" - 08 02 instead of 0208? This is called "low order first" addressing and you'll see a lot of it on the KIM system.

The single-step feature doesn't work too well on Monitor subroutines. That's normal, and it's not serious. These subroutines are well tested and dependable, so you shouldn't need to use SST with them.

Exercises

1. Can you change the program to make the display show 5555 55?

2. Can you write a program to make the display show 1234 56?

3. How about a program to show the word EFFACE? or FACADE? or COOCOO?

MINI-PROGRAM D: reading the keypad

To read the KIM pushbuttons you have another Monitor subroutine called GETKEY. You "call" it with JSR GETKEY (20 6A 1F). This subroutine will give you the identity of the key that is being pressed at that moment as a value in the A register. You can continue by using this value any way you want. If no key is being pressed at the time, you'll get a value of 15 in A.

There are a couple of cautions on the use of JSR GETKEY. First, you must not be in Decimal Mode. If you're not sure about this, give a CLD (D8) instruction at the beginning of your program. Secondly, before giving JSR GETKEY, you must "open up the channel" from the keyboard with either one of two subroutines: JSR SCANDS or JSR KEYIN. You've met JSR SCANDS before: it's used to light the display. If you don't want to light the display, use JSR KEYIN (20 40 1F) before using JSR GETKEY.

This program reads the keyboard and displays what it sees:

```
0200 D8       START CLD         clr dc mode
0201 A9 00          LDA #0      zero into A
0203 85 FB    STORE STA POINTH
0205 85 FA          STA POINTL
0207 85 F9          STA INH
0209 20 1F 1F       JSR SCANDS  light display
020C 20 6A 1F       JSR GETKEY  test keys
020F 4C 03 02       JMP STORE
```

Exercises

1. Do you think that the instruction at 0201 is really needed? Try removing it (change 0201 and 0202 to EA) and see.

2. What values do you get for the alphabetic keys? For keys like PC and GO? Are there any keys that don't work with JSR GETKEY?

3. Try running in decimal mode (change 0200 to SED, code F8). What happens? Is it serious? How about key F?

4. Can you change the program so that only the last digit of the display changes with the keyboard?

CONCLUSION

You've reached the end of our little Beginner's Guide. But you've only started on the road towards understanding programming.

Use the tools we have given you here to forge your own path. KIM is a very rich machine. You have 56 Op Codes to choose from, and many powerful addressing combinations. You don't need to learn them all right away, but when you need them, they'll be there.

The KIM Programming Manual makes good reading. Don't try to go through the whole thing at one sitting. Stop and try a few things; you have the Single Step feature to help you understand what each instruction really does.

Try leafing through - or stepping through - other people's programs, to understand what makes them tick. Change the coding, if you like, to see what happens. When you see a program that does something you want to do, borrow the coding - you don't need to re-invent the wheel.

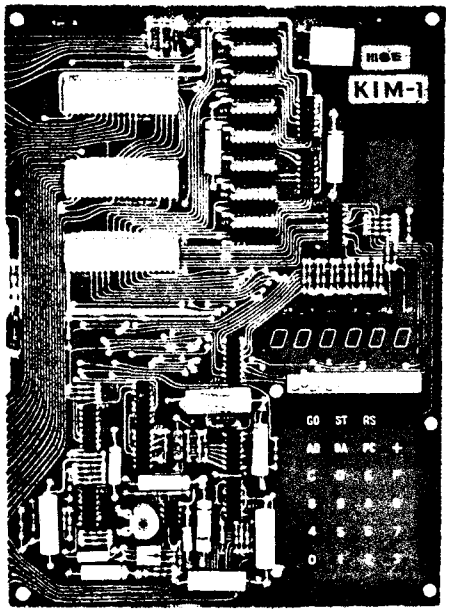
Don't be discouraged when your program doesn't work on the first try. Even experts have to spend time getting the "bugs" out of their coding. It's part of the game: Think of yourself as Sherlock Holmes, methodically tracking down the elusive villains.

A proverb says that a journey of a thousand miles starts with the first step. In the same way, the biggest programs still operate one step at a time.

So forge ahead at your own speed. Communicate with other KIM owners; you'll have a lot of information to swap.

But most of all: have fun.

RECREATIONAL PROGRAMS



Clear Decimal Mode: Set 00F1 to 00 before running these programs.

ADDITION

BY JIM BUTTERFIELD

DIRECTIONS -

HERE'S A HANDY LITTLE ADDING MACHINE PROGRAM. KIM BECOMES A SIX DIGIT ADDER. "GO" CLEARS THE TOTAL SO YOU CAN START OVER. THEN ENTER A NUMBER AND HIT THE PLUS KEY TO ADD IT TO THE PREVIOUS TOTAL. IF YOU MAKE A MISTAKE IN ENTERING A NUMBER, JUST HIT THE "0" KEY SEVERAL TIMES AND ROLL THE BAD NUMBER OUT BEFORE ENTERING THE CORRECTION. NO OVERFLOW INDICATOR, AND NO SUBTRACTION OR MULTIPLICATION - MAYBE YOU WOULD LIKE TO TRY YOUR HAND AT ADDING THESE. THE PROGRAM IS FULLY RELOCATABLE.

```
0200 20 1F 1F START   JSR SCANDS       light display
0203 20 6A 1F         JSR GETKEY       read keyboard
0206 C5 60           CMP PREV          same as last time?
0208 F0 F6           BEQ START         yes, skip
020A 85 60           STA PREV          no, save new key
020C C9 0A           CMP #$0A          numeric key?
020E 90 29           BCC NUM           yes, branch
0210 C9 13           CMP #$13          GO key?
0212 F0 18           BEQ DOGO          yes, branch
0214 C9 12           CMP #$12          + key?
0216 D0 E8           BNE START         no, invalid key
0218 F8 18           SED CLC           prepare to add
021A A2 FD           LDX #$FD          minus 3; 3 digits
021C B5 FC    ADD    LDA POINTH+1,X     display digit
021E 75 65           ADC ACCUM+3,X      add total
0220 95 FC           STA POINTH+1,X     total to display
0222 95 65           STA ACCUM+3,X      & to total accum
0224 E8              INX               next digit
0225 30 F5           BMI ADD           last digit?
0227 86 61           STX FLAG          flag total-in-display
0229 D8              CLD
022A 10 D4           BPL START         return to start
022C A9 00    DOGO   LDA #0            set flag for
022E 85 61           STA FLAG          total-in-display
0230 A2 02           LDX #2            for 3 digits...
0232 95 F9    CLEAR  STA INH,X         clear display
0234 CA              DEX               next digit
0235 10 FB           BPL CLEAR         last digit?
0237 30 C7           BMI START         finished, back to go
0239 A4 61    NUM    LDY FLAG          total-in-display?
023B D0 0F           BNE PASS          no, add new digit
023D E6 61           INC FLAG          clear t-i-d flag
023F 48              PHA               save key
0240 A2 02           LDX #2            3 digits to move
```

```
0242 B5 F9    MOVE   LDA INH,X       get display digit
0244 95 62           STA ACCUM,X     copy to total Accum
0246 94 F9           STY INH,X       clear display
0248 CA              DEX             next digit
0249 10 F7           BPL MOVE        last digit?
024B 68              PLA             recall key
024C 0A 0A    PASS   ASL A ASL A     move digit..
024E 0A 0A           ASL A ASL A     ..into position
0250 A2 04           LDX #4          4 bits
0252 0A       SHIFT  ASL A           move bit from A
0253 26 F9           ROL INH         ..to INH..
0255 26 FA           ROL POINTL      ..to rest of
0257 26 FB           ROL POINTH          display
0259 CA              DEX             next bit
025A D0 F6           BNE SHIFT       last bit?
025C F0 A2           BEQ START       yes. back to start
```

***** HEX DUMP - ADDITION *****

```
0200 20 1F 1F 20 6A 1F C5 60 F0 F6 85 60 C9 0A 90 29
0210 C9 13 F0 18 C9 12 D0 E8 F8 18 A2 FD B5 FC 75 65
0220 95 FC 95 65 E8 30 F5 86 61 D8 10 D4 A9 00 85 61
0230 A2 02 95 F9 CA 10 FB 30 C7 A4 61 D0 0F E6 61 48
0240 A2 02 B5 F9 95 62 94 F9 CA 10 F7 68 0A 0A 0A 0A
0250 A2 04 0A 26 F9 26 FA 26 FB CA D0 F6 F0 A2
```

NOTE: WHENEVER SPACE PERMITS, A HEX DUMP OF THE PROGRAMS LISTED WILL BE GIVEN. THESE DUMPS WERE TAKEN FROM ACTUAL RUNNING PROGRAMS. SO, IF THERE IS A DISCREPANCY BETWEEN THE LISTING AND THE DUMP, THE LISTING IS MOST PROBABLY IN ERROR.

ASTEROID

BY STAN OCKERS

YOU ARE PILOTING YOUR SPACECRAFT BETWEEN MARS AND JUPITER WHEN YOU ENCOUNTER A DENSE PORTION OF THE ASTEROID BELT. PRESS KEY ZERO TO MOVE LEFT, THREE TO MOVE RIGHT. WHEN YOUR CRAFT IS HIT THE DISPLAY WILL GIVE A NUMBER TO INDICATE HOW SUCESSFUL YOU WERE. THE PROGRAM STARTS AT 0200.

```
0200  A9 00          LDA #$00      ...INITIALIZE COUNTER...
0202  85 F9          STA 00F9
0204  85 FA          STA 00FA
0206  85 FB          STA 00FB
0208  A2 06          LDX #$06      ...INITIALIZE 00E2-00E8
020A  BD CE 02 INIT  LDA 02CE,X
020D  95 E2          STA 00E2,X
020F  CA             DEX
0210  10 F8          BPL INIT
0212  A5 E8    TOGG  LDA 00E8      ...TOGGLE 00E8...
0214  49 FF          EOR #$FF
0216  85 E8          STA 00E8      (FLASHER FLAG)
0218  A2 05          LDX #$05      DELAY BETWEEN FLASHES
021A  20 48 02 LITE  JSR DISP      DISPLAY AND..
021D  20 97 02       JSR CHEK      CHECK FOR MATCH
0220  CA             DEX
0221  D0 F7          BNE LITE
0223  20 40 1F       JSR KEYIN     SET DIRECTIONAL REGS.
0226  20 6A 1F       JSR GETKEY    GET KEYBOARD ENTRY
0229  C9 15          CMP #$15      A VALID KEY?
022B  10 E5          BPL TOGG      NO
022D  C9 00          CMP #$00      KEY 0?
022F  F0 06          BEQ LEFT      YES, GO LEFT
0231  C9 03          CMP #$03      KEY 3?
0233  F0 0A          BEQ RT        YES, GO RIGHT
0235  D0 DB          BNE TOGG      NOT A VALID KEY
0237  06 E7    LEFT  ASL 00E7      SHIFT CRAFT LEFT
0239  A9 40          LDA #$40      LEFT HAND EDGE?
023B  C5 E7          CMP 00E7
023D  D0 D3          BNE TOGG      NO, RETURN
023F  46 E7    RT    LSR 00E7      SHIFT RIGHT
0241  D0 CF          BNE TOGG      NOT RIGHT SIDE, RETURN
0243  38             SEC           OFF EDGE, RETURN TO
0244  26 E7          ROL 00E7      RIGHT SIDE
0246  D0 CA          BNE TOGG      RETURN
                  **** DISPLAY SUBROUTINE ****
0248  A9 7F    DISP  LDA #$7F      PORT TO OUTPUT
024A  8D 41 17       STA 1741
024D  A9 09          LDA #$09      INIT. DIGIT
024F  8D 42 17       STA 1742
0252  A9 20          LDA #$20      BIT POSITION TO
0254  85 E0           STA 00E0      6TH BIT
0256  A0 02    BIT   LDY #$02      3 BYTES
0258  A9 00          LDA #$00      ZERO CHARACTER
025A  85 E1          STA 00E1
```

```
025C  B1 E2      BYTE     LDA (00E2),Y   GET BYTE
025E  25 E0               AND 00E0       NTH BIT = 1?
0260  F0 07               BEQ NOBT       NO, SKIP
0262  A5 E1               LDA 00E1       YES, UPDATE
0264  19 E4 00            ORA 00E4,Y     CHARACTER
0267  85 E1               STA 00E1
0269  88                  DEY
026A  10 F0               BPL BYTE       NEXT BYTE
026C  A5 E1               LDA 00E1       CHAR. IN ACCUM.
026E  C4 E8               CPY 00E8       SHIP ON?
0270  D0 08               BNE DIGT       NO, SKIP
0272  A4 E0               LDY 00E0       IS THIS SHIP
0274  C4 E7               CPY 00E7       DIGIT?
0276  D0 02               BNE DIGT       NO, SKIP
0278  09 08               ORA #$08       ADD IN SHIP
027A  8D 40 17  DIGT      STA 1740       LIGHT DIGIT
027D  A9 30               LDA #$30       DELAY (DIGIT ON)
027F  8D 06 17            STA 1706
0282  AD 07 17  DELA      LDA 1707       TIME UP?
0285  F0 FB               BEQ DELA       NO
0287  A9 00               LDA #$00       TURN OFF SEGMENTS
0289  8D 40 17            STA 1740
028C  EE 42 17            INC 1742       SHIFT TO NEXT DIGIT
028F  EE 42 17            INC 1742
0292  46 E0               LSR 00E0       SHIFT TO NEXT BIT
0294  D0 C0               BNE BIT        MORE BITS
0296  60                  RTS
                  ***** CHECK SUBROUTINE *****
0297  C6 E9      CHEK     DEC 00E9       DEC. TIMES THRU COUNT
0299  D0 1A               BNE MORE       SKIP IF NOT 48TH TIME
029B  A9 30               LDA #$20       RESET TIMES THRU COUNT
029D  85 E9               STA 00E9
029F  8A                  TXA            SAVE X
02A0  48                  PHA
02A1  A2 FD               LDX  #$FD      NEGATIVE 3 IN X
02A3  F8                  SED            DECIMAL MODE
02A4  38                  SEC            (TO ADD ONE)
02A5  B5 FC      NXTB     LDA 00FC,X     ..INCREMENT COUNTER
02A7  69 00               ADC #$00       WHICH IS MADE OF BYTES
02A9  95 FC               STA 00FC,X     IN DISPLAY AREA (00F9-
02AB  E8                  INX            00FB)..
02AC  D0 F7               BNE NXTB       NEXT BYTE
02AE  D8                  CLD
02AF  68                  PLA            RETURN X
02B0  AA                  TAX
02B1  E6 E2               INC 00E2       ..SET UP FOR NEXT GROUP
02B3  A5 E2               LDA 00E2       OF BYTES..
02B5  C9 30      MORE     CMP #$30       ALL GROUPS FINISHED?
02B7  F0 09               BEQ RECY       YES, RECYCLE ASTR. FIELD
02B9  A0 00      MATCH    LDY #$00       SHIP - ASTEROID MATCH?
02BB  A5 E7               LDA 00E7       LOAD CRAFT POSITION
02BD  31 E2               AND 00E2,Y     AND WITH ASTEROID BYTE
02BF  D0 07               BNE FIN        IF MATCH, YOU'VE HAD IT
02C1  60                  RTS            EXIT MATCH SUBROUTINE
```

```
02C2  A9 00     RECY     LDA #$00       GO THRU ASTEROID FIELD
02C4  85 E2              STA 00E2       AGAIN
02C6  F0 F1              BEQ MATCH      UNCONDITIONAL BRANCH
02C8  20 1F 1F  FIN      JSR SCANDS     DISPLAY COUNT
02CB  4C C8 02           JMP FIN        CONTINUOUSLY

02CE  D5                 LOW POINTER, ASTEROID BELT
02CF  02                 HIGH POINTER, ASTEROID BELT
02D0  08                 MASK, BOTTOM SEGMENT
02D1  40                 MASK, MIDDLE SEGMENT
02D2  01                 MASK, TOP SEGMENT
02D3  04                 CRAFT POSITION
02D4  FF                 FLAG (SHIP ON)
```

******* ASTEROID FIELD *******

```
02D5-   00 00 00 04 00 08 00 06 12 00 11 00 05 00 2C 00
02E5-   16 00 29 00 16 00 2B 00 26 00 19 00 17 00 38 00
02F5-   2E 00 09 00 1B 00 24 00 15 00 39 00 0D 00 21 00
0305-   10 00 00
```

******* HEX DUMP - ASTEROID *******

```
0200-   A9 00 85 F9 85 FA 85 FB A2 06 BD CE 02 95 E2 CA
0210-   10 F8 A5 E8 49 FF 85 E8 A2 05 20 48 02 20 97 02
0220-   CA D0 F7 20 40 1F 20 6A 1F C9 15 10 E5 C9 00 F0
0230-   06 C9 03 F0 0A D0 DB 06 E7 A9 40 C5 E7 D0 D3 46
0240-   E7 D0 CF 38 26 E7 D0 CA A9 7F 8D 41 17 A9 09 8D
0250-   42 17 A9 20 85 E0 A0 02 A9 00 85 E1 B1 E2 25 E0
0260-   F0 07 A5 E1 19 E4 00 85 E1 88 10 F0 A5 E1 C4 E8
0270-   D0 08 A4 E0 C4 E7 D0 02 09 08 8D 40 17 A9 30 8D
0280-   06 17 AD 07 17 F0 FB A9 00 8D 40 17 EE 42 17 EE
0290-   42 17 46 E0 D0 C0 60 C6 E9 D0 1A A9 30 85 E9 8A
02A0-   48 A2 FD F8 38 B5 FC 69 00 95 FC E8 D0 F7 D8 68
02B0-   AA E6 E2 A5 E2 C9 30 F0 09 A0 00 A5 E7 31 E2 D0
02C0-   07 60 A9 00 85 E2 F0 F1 20 1F 1F 4C C8 02 D5 02
02D0-   08 40 01 04 FF 00 00 00 04 00 08 00 06 12 00 11
02E0-   00 05 00 2C 00 16 00 29 00 16 00 2B 00 26 00 19
02F0-   00 17 00 38 00 2E 00 09 00 1B 00 24 00 15 00 39
0300-   00 0D 00 21 00 10 00 00
```

CHANGES -
YOU CAN MAKE YOUR OWN ASTEROID FIELD STARTING AT 02D5. THE GROUP COUNT, (02B6), WILL HAVE TO BE CHANGED IF THE FIELD SIZE DIFFERS. THE SPEED OF THE CRAFT MOVING THROUGH THE FIELD IS CONTROLLED BY 027E. WHAT ABOUT A VARYING SPEED, SLOW AT FIRST AND SPEEDING UP AS YOU GET INTO THE FIELD? WHAT ABOUT A FINAL "DESTINATION COUNT" AND A SIGNAL TO INDICATE YOU HAVE REACHED YOUR DESTINATION? HOW ABOUT ALLOWING A HIT OR TWO BEFORE YOU ARE FINALLY DISABLED?

BAGELS

BY JIM BUTTERFIELD

DIRECTIONS -

THE COMPUTER HAS CHOSEN FOUR LETTERS, ALL OF WHICH ARE A,B,C,D,E, OR F. LETTERS MAY BE REPEATED - FOR EXAMPLE, THE COMPUTER'S "SECRET" COMBINATION MIGHT BE CACF OR BBBB.

YOU GET TEN GUESSES. EACH TIME YOU GUESS, THE COMPUTER WILL TELL YOU TWO THINGS: HOW MANY LETTERS ARE EXACTLY CORRECT (THE RIGHT LETTER IN THE RIGHT PLACE); AND HOW MANY LETTERS ARE CORRECT, BUT IN THE WRONG POSITION.

FOR EXAMPLE, IF THE COMPUTER'S SECRET COMBINATION IS CBFB AND YOU GUESS BAFD, THE TWO NUMBERS WILL BE 1 AND 1 (THE F MATCHES EXACTLY; THE B MATCHES BUT IN THE WRONG PLACE). THESE NUMBERS WILL SHOW ON THE RIGHT HAND SIDE OF THE DISPLAY; THE CODE YOU ENTERED WILL APPEAR ON THE LEFT.

MAKE A NOTE OF YOUR GUESSES AND THE COMPUTER'S RESPONSE. WITH A LITTLE MENTAL WORK, YOU SHOULD BE ABLE TO BREAK THE CODE EXACTLY IN SEVEN OR EIGHT WORDS. A CORRECT GUESS WILL PRODUCE A RESPONSE OF 4 - 0. IF YOU DON'T GUESS RIGHT IN TEN MOVES, THE COMPUTER WILL GIVE YOU THE ANSWER.

AFTER A CORRECT GUESS, OR AFTER THE COMPUTER TELLS YOU THE ANSWER, IT WILL START A NEW GAME (WITH A NEW SECRET CODE) THE INSTANT YOU TOUCH A NEW KEY.

```
0200 E6 16     GO      INC RND+4    randomize
0202 20 40 1F          JSR KEYIN     on pushbutton delay
0205 D0 F9             BNE GO
0207 D8                CLD
0208 A9 0A     NEW     LDA #$0A     ten guesses/game
020A 85 18             STA COUNT    new game starting
020C A9 03             LDA #3       create 4 mystery codes
020E 85 10             STA POINTR
0210 38        RAND    SEC          one plus...
0211 A5 13             LDA RND+1    ...three previous
0213 65 16             ADC RND+4       random numbers
0215 65 17             ADC RND+5
0217 85 12             STA RND      =new random value
0219 A2 04             LDX #4
021B B5 12     RLP     LDA RND,X    move random numbers over
021D 95 13             STA RND+1,X
021F CA                DEX
0220 10 F9             BPL RLP
0222 A6 10             LDX POINTR
0224 A0 C0             LDY #$C0     divide by 6
0226 84 11             STY MOD      keeping remainder
0228 A0 06             LDY #6
022A C5 11     SET     CMP MOD
022C 90 02             BCC PASS
022E E5 11             SBC MOD
0230 46 11     PASS    LSR MOD
0232 88                DEY
0233 D0 F5             BNE SET      continue division
0235 18                CLC
0236 69 0A             ADC #$0A     random value A to F
```

```
0238 95 00           STA SECRET,X
023A C6 10           DEC POINTR
023C 10 D2           BPL RAND
023E C6 18    GUESS  DEC COUNT  new guess starts here
0240 30 7A           BMI FINISH ten guesses?
0242 A9 00           LDA #0
0244 A2 0C           LDX #$0C   clear from WINDOW...
0246 95 04    WIPE   STA WINDOW,X    ...to  POINTR
0248 CA              DEX
0249 10 FB           BPL WIPE
                     ;
                     ;      WAIT FOR KEY TO BE DEPRESSED
                     ;
024B 20 CE 02 WAIT   JSR SHOW
024E F0 FB           BEQ WAIT
0250 20 CE 02        JSR SHOW
0253 F0 F6           BEQ WAIT   debounce key
0255 A5 08           LDA WINDOW+4    new guess?
0257 F0 08           BEQ RESUME      no, input digit
0259 29 60           AND #$60
025B 49 60           EOR #$60   previous game finished?
025D F0 A9           BEQ NEW    ...yes, new game;
025F D0 DD           BNE GUESS  ...no, next guess
0261 20 6A 1F RESUME JSR GETKEY
0264 C9 10           CMP #$10   guess must be in
0266 B0 E3           BCS WAIT     range A to F
0268 C9 0A           CMP #$0A
026A 90 DF           BCC WAIT
026C A8              TAY
026D A6 10           LDX POINTR  zero to start
026F E6 10           INC POINTR
0271 B9 E7 1F        LDA TABLE,Y   segment pattern
0274 95 04           STA WINDOW,X
0276 98              TYA
0277 D5 00           CMP SECRET,X  exact match?
0279 D0 03           BNE NOTEX
027B E6 0E           INC EXACT
027D 8A              TXA        destroy input
027E 95 0A    NOTEX  STA INPUT,X
0280 A5 07           LDA WINDOW+3  has fourth digit arrived?
0282 F0 31           BEQ BUTT      ...no
0284 A0 03           LDY #3        ...yes, calculate matches
0286 B9 oA 00 STEP   LDA INPUT,Y   for each digit:
0289 29 18           AND #$18      ..has it already been
028B F0 12           BEQ ON          matched?
028D B9 00 00        LDA SECRET,Y
0290 A2 03           LDX #3        if not, test
0292 D5 0A    LOOK   CMP INPUT,X   ...against input
0294 F0 05           BEQ GOT
0296 CA              DEX
0297 10 F9           BPL LOOK
0299 30 04           BMI ON
029B E6 0F    GOT    INC MATCH  increment counter
029D 16 0A           ASL INPUT,X   and destroy input
029F 88       ON     DEY
02A0 10 E4           BPL STEP
```

```
02A2 A2 01           LDX #1         display counts
02A4 B4 0E    TRANS  LDY EXACT,X
02A6 B9 E7 1F        LDA TABLE,Y
02A9 95 08           STA WINDOW+4,X
02AB CA              DEX
02AC 10 F6           BPL TRANS
02AE 20 CE 02 DELAY  JSR SHOW      long pause for debounce
02B1 E6 0F           INC MATCH
02B3 D0 F9           BNE DELAY
02B5 20 CE 02 BUTT   JSR SHOW      wait for key release
02B8 D0 FB           BNE BUTT
02BA F0 8F           BEQ WAIT
                     ;
                     ;       TEN GUESSES MADE - SHOW ANSWER
                     ;
02BC A2 03    FINISH LDX #3
02BE B4 00    FIN2   LDY SECRET,X
02C0 B9 E7 1F        LDA TABLE,Y
02C3 95 04           STA WINDOW,X
02C5 CA              DEX
02C6 10 F6           BPL FIN2
02C8 A9 E3           LDA #$e3    'square' flag
02CA 85 08           STA WINDOW+4
02CC D0 E0           BNE DELAY  unconditional jmp


                     ;       SUBROUTINE TO DISPLAY
                     ;       AND TEST KEYBOARD
                     ;
02CE A0 13    SHOW   LDY #$13
02D0 A2 05           LDX #5
02D2 A9 7F           LDA #$7F
02D4 8D 41 17        STA PADD
02D7 B5 04    LITE   LDA WINDOW,X
02D9 8D 40 17        STA SAD
02DC 8C 42 17        STY SBD
02DF E6 11    POZ    INC MOD     pause loop
02E1 D0 FC           BNE POZ
02E3 88              DEY
02E4 88              DEY
02E5 CA              DEX
02E6 10 EF           BPL LITE
02E8 20 40 1F        JSR KEYIN
02EB 60              RTS
                     END
```

Program notes:

1. Program enforces a pause of about 4 seconds after displaying counts or answer. This guards against display being 'missed' due to bounce, hasty keying.
2. After count displayed, or at end of game(s), user can blank display, if desired, by pressing GO or any numeric key. Game operation is not affected, but user may feel it 'separates' games better.

3. When a digit from the user's guess is matched, it is destroyed so that it will not be matched again. There are two significantly different types of 'destruction', however (at 27D and 29D); the test at label STEP is sensitive to which one is used.

```
                ;       LINKAGES TO KIM MONITOR
                ;
                KEYIN   =$1F40
                GETKEY  =$1F6A
                TABLE   =$1FE7
                PADD    =$1741
                SBD     =$1742
                SAD     =$1740
                ;
                ;       WORK AREAS
                ;
0000            SECRET  *=*+4    computer's secret code
0004            WINDOW  *=*+6    display window
000A            INPUT   *=*+4    player"s input area
000E            EXACT   *=*+1    # of exact matches
000F            MATCH   *=*+1    # of other matches
0010            POINTR  *=*+1    digit being input
0011            MOD     *=*+1    divisor/delay flag
0012            RND     *=*+6    random number series
0018            COUNT   *=*+1    number of guesses left
```

****** HEX DUMP - BAGELS ******

```
0200 E6 16 20 40 1F D0 F9 D8 A9 0A 85 18 A9 03 85 10
0210 38 A5 13 65 16 65 17 85 12 A2 04 B5 12 95 13 CA
0220 10 F9 A6 10 A0 C0 84 11 A0 06 C5 11 90 02 E5 11
0230 46 11 88 D0 F5 18 69 0A 95 00 C6 10 10 D2 C6 18
0240 30 7A A9 00 A2 0C 95 04 CA 10 FB 20 CE 02 F0 FB
0250 20 CE 02 F0 F6 A5 08 F0 08 29 60 49 60 F0 A9 D0
0260 DD 20 6A 1F C9 10 B0 E3 C9 0A 90 DF A8 A6 10 E6
0270 10 B9 E7 1F 95 04 98 D5 00 D0 03 E6 0E 8A 95 0A
0280 A5 07 F0 31 A0 03 B9 0A 00 29 18 F0 12 B9 00 00
0290 A2 03 D5 0A F0 05 CA 10 F9 30 04 E6 0F 16 0A 88
02A0 10 E4 A2 01 B4 0E B9 E7 1F 95 08 CA 10 F6 20 CE
02B0 02 E6 0F D0 F9 20 CE 02 D0 FB F0 8F A2 03 B4 00
02C0 B9 E7 1F 95 04 CA 10 F6 A9 E3 85 08 D0 E0 A0 13
02D0 A2 05 A9 7F 8D 41 17 B5 04 8D 40 17 8C 42 17 E6
02E0 11 D0 FC 88 88 CA 10 EF 20 40 1F 60
```

Label Table for Program BAGELS

ADDRESS	LABEL	WHERE USED
02B5	BUTT	0282 02B8
0018	COUNT	020A 023E
02AE	DELAY	02B3 02CC
000E	EXACT	027B 02A4
02BE	FIN2	02C6
02BC	FINISH	0240
1F6A	GETKEY	0261
0200	GO	0205
029B	GOT	0294
023E	GUESS	025F
000A	INPUT	027E 0286 0292 029D
1F40	KEYIN	0202 02E8
02D7	LITE	02E6
0292	LOOK	0297
000F	MATCH	029B 02B1
0011	MOD	0226 022A 022E 0230 02DF
0208	NEW	025D
027E	NOTEX	0279
029F	ON	0299
1741	PADD	02D4
0230	PASS	022C
0010	POINTR	020E 0222 023A 026D 026F
02DF	POZ	02E1
0210	RAND	023C
0261	RESUME	0257
021B	RLP	0220
0012	RND	0200 0211 0213 0215 0217 021B 021D
1740	SAD	02D9
1742	SBD	02DC
0000	SECRET	0238 0277 028D 02BE
022A	SET	0233
02CE	SHOW	024B 0250 02AE 02B5
0286	STEP	02A0
1FE7	TABLE	0271 02A6 02C0
02A4	TRANS	02AC
024B	WAIT	024B 0253 0266 026A 02BA
0246	WIPE	0249
0004	WINDOW	0246 0255 0274 0280 02A9 02C3 02CA 02D7

Label tables, when available, are often useful for studying a program. For each label (alphabetically arranged) you can see, on the left, the address belonging to the label; and on the right, where the label is used in the program.

BANDIT

JIM BUTTERFIELD

Start the program at 0200 and on the right, you'll see the $25 that KIM has given you to play with. The funny symbols on the left are your "wheels" - hit any key and see them spin.

Every time you spin the wheels by hitting a key it costs you $1. When the wheels stop, you might have a winning combination, in which case you'll see money being added to your total on the right. Most of the time, you'll get nothing ... but that's the luck of the game.

The biggest jackpot is $15: that's three bars across the display. Other combinations pay off, too; you'll soon learn to recognize the "cherry" symbol, which pays $2 every time it shows in the left hand window.

There's no house percentage, so you can go a long time on your beginning $25. The most you can make is $99; and if you run out of money, too bad: KIM doesn't give credit.

1 cherry 1+1 4pts

777 5pts

BBB 10pts

BANDIT MICRO-WARE ASSEMBLER 65XX-1.0 PAGE 01

```
0010:
0020:                   **********************************
0030:                   ***** ONE ARMED BANDIT   *****
0040:                   ***** BY JIM BUTTERFIELD *****
0050:                   **********************************
0060:
0070: 02D1              WINDOW *      $0000  DISPLAY WINDOW
0080: 02D1              AMT    *      $0005  CASH CACHE
0090: 02D1              ARROW  *      $0006
0100: 02D1              RWD    *      $0007  REWARD
0110: 02D1              STALLA *      $0008  WAIT WHILE
0120: 02D1              TUMBLE *      $0009
0130:
0140:                   LINKAGES TO KIM
0150:
0160: 02D1              KEYIN  *      $1F40  IS KEY DEPRESSED?
0170: 02D1              PADD   *      $1741
0180: 02D1              SAD    *      $1740
0190: 02D1              SBD    *      $1742
0200: 02D1              TABLE  *      $1FE7  HEX:7 SEG
```

```
0210:
0220:                    MAIN PROGRAM
0230:
0240: 0200               BANDIT ORG   $0200
0250: 0200 A9 25         GO     LDAIM $25     GIVE HIM $25
0260: 0202 85 05                STA   AMT     TO START WITH
0270: 0204 20 BA 02             JSR   CVAMT   AND SHOW IT TO HIM.
0280: 0207 A9 00                LDAIM $00     RESET ARROW.
0290: 0209 85 06                STA   ARROW
0300:
0310:                    MAIN DISPLAY LOOP
0320:
0330: 020B 20 8D 02      LPA    JSR   DISPLY  DISPLAY UNTIL
0340: 020E D0 FB                BNE   LPA     [GO] IS RELEASED.
0350: 0210 E6 09         ROLL   INC   TUMBLE  RANDOMIZE TUMBLE.
0360: 0212 20 8D 02             JSR   DISPLY  DISPLAY UNTIL
0370: 0215 F0 F9                BEQ   ROLL    A KEY IS HIT.
0380:
0390: 0217 A9 03                LDAIM $03
0400: 0219 85 06                STA   ARROW
0410: 021B F8                   SED
0420: 021C 38                   SEC
0430: 021D A5 05                LDA   AMT
0440: 021F E9 01                SBCIM $01     CHARGE ONE BUCK.
0450: 0221 85 05                STA   AMT
0460: 0223 20 BA 02             JSR   CVAMT   CONVERT FOR LED.
0470: 0226 26 09                ROL   TUMBLE
0480:
0490: 0228 20 8D 02      LPB    JSR   DISPLY
0500: 022B C6 08                DEC   STALLA  DISPLAY A WHILE.
0510: 022D D0 F9                BNE   LPB
0520: 022F A6 06                LDX   ARROW
0530: 0231 A5 09                LDA   TUMBLE     MAKE A
0540: 0233 29 06                ANDIM $06        RESULT
0550: 0235 09 40                ORAIM $40
0560:
0570: 0237 95 01                STAAX WINDOW +01
0580: 0239 46 09                LSR   TUMBLE
0590: 023B 46 09                LSR   TUMBLE  DO ALL
0600: 023D C6 06                DEC   ARROW   3 WINDOWS.
0610: 023F D0 E7                BNE   LPB
0620:
0630:                    ALL WHEELS STOPPED - COMPUTE PAYOFF
0640:
0650: 0241 A5 04                LDA   WINDOW +04
0660: 0243 C5 03                CMP   WINDOW +03  CHECK FOR
0670: 0245 D0 37                BNE   NOMAT          A MATCH.
0680: 0247 C5 02                CMP   WINDOW +02
0690: 0249 D0 33                BNE   NOMAT
0700: 024B A2 10                LDXIM $10
0710: 024D C9 40                CMPIM $40     PAY $15 FOR 3 BARS
0720: 024F F0 0D                BEQ   PAY
0730: 0251 A2 0B                LDXIM $08
0740: 0253 C9 42                CMPIM $42     PAY $10 FOR 3 UPS
0750: 0255 F0 07                BEQ   PAY
0760: 0257 A2 06                LDXIM $06
0770: 0259 C9 44                CMPIM $44     PAY $5 FOR 3 DOWNS
0780: 025B F0 01                BEQ   PAY
0790: 025D CA                   DEX
```

```
0800:
0810:                    A WIN!!! PAY AMOUNT IN X
0820:
0830: 025E 86 07      PAY    STX   RWD     HIDE REWARD
0840: 0260 A9 80      PAX    LDAIM $80
0850: 0262 85 08             STA   STALLA
0860: 0264 20 8D 02   LPC    JSR   DISPLY    DISPLAY
0870: 0267 C6 08             DEC   STALLA    FOR A HALF
0880: 0269 D0 F9             BNE   LPC        A WHILE.
0890: 026B C6 07             DEC   RWD
0900: 026D F0 9C             BEQ   LPA
0910: 026F 18                CLC             SLOWLY ADD
0920: 0270 F8                SED             THE PAYOFF
0930: 0271 A5 05             LDA   AMT       TO THE AM'T.
0940: 0273 69 01             ADCIM $01
0950: 0275 B0 94             BCS   LPA
0960: 0277 85 05             STA   AMT
0970: 0279 20 BA 02          JSR   CVAMT
0980: 027C D0 E2             BNE   PAX
0990:
1000:                    WHEELS NOT ALL THE SAME - CHECK FOR SMALL PAYOFF
1010:
1020: 027E A2 03      NOMAT  LDXIM $03
1030: 0280 C9 46             CMPIM $46      A CHERRY?
1040: 0282 F0 DA             BEQ   PAY
1050: 0284 20 8D 02   LOK    JSR   DISPLY
1060: 0287 A5 05             LDA   AMT      CAN'T PLAY
1070: 0289 D0 80             BNE   LPA      WITH NO DOUGH!
1080: 028B F0 F7             BEQ   LOK
1090:
1100:
1110:
1120:
1130:                    DISPLAY SUBROUTINE
1140:
1150: 028D A6 06      DISPLY LDX   ARROW
1160: 028F 10 02             BPL   INDIS       ROLL
1170: 0291 F6 02      OVER   INCAX WINDOW +02 THE DRUM
1180: 0293 CA         INDIS  DEX
1190: 0294 10 FB             BPL   OVER
1200: 0296 A9 7F             LDAIM $7F
1210: 0298 8D 41 17          STA   PADD
1220: 029B A0 0B             LDYIM $08
1230: 029D A2 04             LDXIM $04
1240: 029F B5 00      LITE   LDAAX WINDOW LIGHT
1250: 02A1 8C 42 17          STY   SBD      ALL THE
1260: 02A4 8D 40 17          STA   SAD      WINDOWS
1270: 02A7 D8                CLD
1280: 02A8 A9 7F             LDAIM $7F
1290: 02AA E9 01      ZIP    SBCIM $01
1300: 02AC D0 FC             BNE   ZIP
1310: 02AE 8D 42 17          STA   SBD
1320: 02B1 C8                INY
1330: 02B2 C8                INY
```

```
1340:  02B3 CA                 DEX
1350:  02B4 10 E9              BPL    LITE
1360:  02B6 20 40 1F           JSR    KEYIN
1370:  02B9 60                 RTS
1380:
1390:                    AMOUNT CONVERSION
1400:
1410:  02BA A5 05        CVAMT LDA    AMT
1420:  02BC 29 0F              ANDIM $0F      TRANSLATE
1430:  02BE AA                 TAX            AMOUNT
1440:  02BF BD E7 1F           LDAAX TABLE    TO LED
1450:  02C2 85 00              STA    WINDOW   CODE.
1460:  02C4 A5 05              LDA    AMT
1470:  02C6 4A                 LSRA
1480:  02C7 4A                 LSRA
1490:  02C8 4A                 LSRA
1500:  02C9 4A                 LSRA
1510:  02CA AA                 TAX
1520:  02CB BD E7 1F           LDAAX TABLE
1530:  02CE 85 01              STA    WINDOW +01
1540:  02D0 60                 RTS
```

```
SYMBOL TABLE 3000 30A2
AMT     0005    ARROW   0006    BANDIT  0200    CVAMT   02BA
DISPLY  028D    GO      0200    INDIS   0293    KEYIN   1F40
LITE    029F    LOK     0284    LPA     020B    LPB     0228
LPC     0264    NOMAT   027E    OVER    0291    PADD    1741
PAX     0260    PAY     025E    ROLL    0210    RWD     0007
SAD     1740    SBD     1742    STALLA  0008    TABLE   1FE7
TUMBLE  0009    WINDOW  0000    ZIP     02AA
```

You'll notice that the listing for BANDIT looks a little different from others in this book. That's because it is the output of a resident assembler operating in an expanded KIM system. See the section on expansion for a further discussion of assemblers.

You might like to change the payouts so that there is a "house percentage". That way, visitors will eventually run out of money if they play long enough. This has two possible advantages: it will teach them the evils of gambling, and they won't hog your KIM all day playing this game.

BITZ

BY JIM BUTTERFIELD

A teaching program which drills you on binary and hexadecimal numbering schemes. It's kind of fun just as a speed test.

Start the program at 0200 and you'll see eight bits on the left side of the display. Some of the bits are in the lower position, meaning 'off' or zero. Others will be in the top row, where they mean 'on' or logic one.

All you have to do is translate those bits into hexadecimal notation, and enter the hex value. For example, if all bits happen to be 'on', the number you'd enter is FF; or if all the bits were 'off', you'd enter 00. KIM rewards a correct answer with another problem.

If you're not yet at ease with the concept of bits and how they relate to hexadecimal numbering, a few runs of this program will help a lot.

```
0200 D8       START  CLD
0201 A9 01           LDA #1   Set FLAG2 ..
0203 85 1D           STA FLAG2   .. to new problem
0205 20 40 1F MAIN   JSR KEYIN  set directnl reg
0208 20 6A 1F        JSR GETKEY   get key input
020B C5 14           CMP PREV   same as last time?
020D F0 50           BEQ LIGHT   yes, skip
020F 85 14           STA PREV   record new input
0211 C9 15           CMP #$15    no key?
0213 F0 1C           BEQ NOKEY  yes, brnch
0215 A6 1C           LDX FLAG1  first digit found?
0217 D0 0C           BNE DIG1   yes, check second
0219 C5 16           CMP SEED1  first digit match?
021B D0 42           BNE LIGHT  no, ignore input
021D AA              TAX
021E BD E7 1F        LDA TABLE,X  change to segment
0221 85 1C           STA FLAG1     ..store..
0223 D0 3A           BNE LIGHT        .. and exit
0225 C5 17    DIG1   CMP SEED2   second digit match?
0227 D0 36           BNE LIGHT   no, ignore input
0229 AA              TAX
022A BD E7 1F        LDA TABLE,X   change to segment
022D 85 1D           STA FLAG2
022F D0 2E           BNE LIGHT
0231 A6 1D    NOKEY  LDX FLAG2    problem solved?
0233 F0 2A           BEQ LIGHT    not yet, skip
0235 A9 00           LDA #0      Clear..
0237 85 1C           STA FLAG1    ..for new problem
0239 85 1D           STA FLAG2
023B AD 04 17        LDA TIMER   get random value
023E AA              TAX
023F 29 0F           AND #$0F    extract last digit
0241 85 17           STA SEED2    .. and store
```

```
0243 8A                 TXA
0244 4A 4A              LSRA LSRA  Extract first digit
0246 4A 4A              LSRA LSRA
0248 85 16              STA SEED1   ..and store
024A 86 15              STX SEED   Store whole number
024C A2 FC              LDX #$FC   Minus 4 for window
024E A9 00     PATT     LDA #0     Clear Accum
0250 26 15              ROL SEED    ..then roll in..
0252 2A                 ROL A        ..two bits..
0253 26 15              ROL SEED      ..and..
0255 2A                 ROL A          ..convert..
0256 A8                 TAY             ..to..
0257 B9 7B 02           LDA TAB,Y         ..segments
025A 95 1C              STA FLAG1,X
025C E8                 INX          next segment
025D D0 EF              BNE PATT
025F A9 7F     LIGHT    LDA #$7F   Set directional..
0261 8D 41 17           STA SADD     ..registers
0264 A0 09              LDY #9
0266 A2 FA              LDX #$FA   Minus 6
0268 B5 1E     SHOW     LDA FLAG2+1,X  Window contents
026A 8D 40 17           STA SAD
026D 8C 42 17           STY SBD
0270 C6 11     WAIT     DEC MOD
0272 D0 FC              BNE WAIT
0274 C8 C8              INY INY
0276 E8                 INX
0277 30 EF              BMI SHOW
0279 10 8A              BPL MAIN
027B 14 12     TAB      .BYTE $14,$12,$24,$22
027D 24 22
               ; end
```

xxxxxx HEX DUMP - BITZ xxxxxx

```
0200-   D8 A9 01 85 1D 20 40 1F 20 6A 1F C5 14 F0 50 85
0210-   14 C9 15 F0 1C A6 1C D0 0C C5 16 D0 42 AA BD E7
0220-   1F 85 1C D0 3A C5 17 D0 36 AA BD E7 1F 85 1D D0
0230-   2E A6 1D F0 2A A9 00 85 1C 85 1D AD 04 17 AA 29
0240-   0F 85 17 8A 4A 4A 4A 4A 85 16 86 15 A2 FC A9 00
0250-   26 15 2A 26 15 2A A8 B9 7B 02 95 1C E8 D0 EF A9
0260-   7F 8D 41 17 A0 09 A2 FA B5 1E 8D 40 17 8C 42 17
0270-   C6 11 D0 FC C8 C8 E8 30 EF 10 8A 14 12 24 22
```

BLACKJACK

BY JIM BUTTERFIELD

Description:

KIM uses a 'real' deck of cards in this game. So when you've seen four aces going by, you know that there will be no more - until the next shuffle.

BLACKJACK starts at address 0200. You'll see the cards being shuffled - the word SHUFFL appears on the display - and then KIM will ask how much you want to bet.

You'll start with an initial amount of $20. Your balance is always shown to the right of the BET? question, so on the first hand, you'll see BET? 20 on the display.

You may bet from $1 to $9, which is the house limit. The instant you hit key 1 to 9 to signal your bet, KIM will deal. Of course, you can't bet more money than you have ... and KIM ignores freeloaders who try to bet a zero amount.

After the deal, you'll see both your cards on the left of the display, and one of KIM's cards on the right. (KIM's other card is a "hole" card, and you won't see it until it's KIM's turn to play). Aces are shown as letter A, face cards and tens as letter F, and other cards as their value, two to nine. As always, Aces count value 1 or 11 and face cards count 10.

You can call for a third card by hitting the 3 button .. then the fourth card with the 4 button, and so on. If your total goes over 21 points, KIM will ungrammatically say BUSTED, and you'll lose. If you get five cards without exceeding 21 points, you'll win automatically. If you don't want any more cards, hit key 0. KIM will report your point total, and then will show and play its own hand. KIM, too, might go BUSTED or win on a five-card hand. Otherwise, the most points wins.

From time to time, KIM will advise SHUFFL when the cards start to run low.

Remember that you have a good chance to beat KIM at this game. Keep track of the cards that have been dealt (especially aces and face cards), and you're likely to be a winner!

```
0200 A2 33     START  LDX #51       52 cards in deck
0202 8A        DK1    TXA           Create deck
0203 95 40            STA DECK,X     by inserting cards
0205 CA               DEX             into deck
0206 10 FA            BPL DK1           in sequence
0208 A2 02            LDX #2        Set up 3 locations
020A BD BB 03 INLOP   LDA INIT,X       ..into..
020D 95 75            STA PARAM,X       zero page
020F CA               DEX           addresshi/ dpt/ amt
```

```
0210 10 F8              BPL INLOP
0212 AD 04 17           LDA TIMER    use random timer
0215 85 80              STA RND       to seed random chain
0217 D8        DEAL     CLD          main loop repeats here
0218 A6 76              LDX DPT      next-card pointer
021A E0 09              CPX #9       less than 9 cards?
021C B0 34              BCS NOSHUF   9 or more, don't shuffl
                ; shuffle deck
021E A0 D8              LDY #SHUF-$300  Set up SHUFFL msg
0220 20 57 03           JSR FILL         put in WINDOW
0223 A0 33              LDY #51         ripple 52 cards
0225 84 76              STY DPT      set full deck
0227 20 30 03 SHLP      JSR LIGHT    illuminate display
022A 38                 SEC
022B A5 81              LDA RND+1    Generate
022D 65 82              ADC RND+2     new
022F 65 85              ADC RND+5      random
0231 85 80              STA RND         number
0233 A2 04              LDX #4
0235 B5 80     RMOV     LDA RND,X    move over
0237 95 81              STA RND+1,X   the random
0239 CA                 DEX             seed numbers
023A 10 F9              BPL RMOV
023C 29 3F              AND #$3F     Strip to 0-63 range
023E C9 34              CMP #52      Over 51?
0240 B0 E5              BCS SHLP     yes, try new number
                ; swap each card into random slot
0242 AA                 TAX
0243 B9 40 00           LDA DECK,Y   get next card
0246 48                 PHA          save it
0247 B5 40              LDA DECK,X   get random card
0249 99 40 00           STA DECK,Y    into position N
024C 68                 PLA          and the original card
024D 95 40              STA DECK,X    into the random slot
024F 88                 DEY          next in sequence
0250 10 D5              BPL SHLP     bck for next card
                ; ready to accept bet
0252 A0 DE     NOSHUF   LDY #MBET-$300   Set up BET? msg
0254 20 57 03           JSR FILL         put in WINDOW
0257 A5 77              LDA AMT      display balance
0259 20 A6 03           JSR NUMDIS     .. put in WINDOW
025C 20 30 03 BETIN     JSR LIGHT    illuminate display
025F C9 0A              CMP #10      not key 0 to 9?
0261 B0 F9              BCS BETIN    nope, ignore
0263 AA                 TAX
0264 86 79              STX BET      store bet amount
0266 CA                 DEX
0267 30 F3              BMI BETIN    zero bet?
0269 E4 77              CPX AMT      sufficient funds?
026B B0 EF              BCS BETIN    no, refuse bet
                ; bet accepted - deal
026D A2 0B              LDX #11      Clean WINDOW and
026F A9 00              LDA #0           card counters
0271 95 90     CLOOP    STA WINDOW,X
0273 CA                 DEX
0274 10 FB              BPL CLOOP
```

```
                        ; here come the cards
0276 20 78 03           JSR YOU      one for you..
0279 20 8F 03           JSR ME       & one for me..
027C 20 78 03           JSR YOU      another for you..
027F 20 64 03           JSR CARD     put my second card..
0282 86 7A              STX HOLE      ..in the hole
0284 20 28 03           JSR WLITE    wait a moment
                        ; deal complete - wait for Hit or Stand
0287 20 30 03 TRY       JSR LIGHT
028A AA CA              TAX DEX      key input?
028C 30 11              BMI HOLD     zero for Stand?
028E E4 96              CPX UCNT     N for card #n?
0290 D0 F5              BNE TRY      nope, ignore  key
                        ; Hit - deal another card
0292 20 78 03           JSR YOU      deal it
0295 C9 22              CMP #$22     22 or over?
0297 B0 40              BCS UBUST    yup, you bust
0299 E0 05              CPX #5       5 cards?
029B F0 53              BEQ UWIN     yup, you win
029D D0 E8              BNE TRY      nope, keep going
                        ; Stand - show player's total
029F A5 95    HOLD      LDA WINDOW+5   save KIM card
02A1 48                 PHA              on stack
02A2 A2 00              LDX #0       flag player ..
02A4 20 0F 03           JSR SHTOT     .. for total display
02A7 A2 04              LDX #4
02A9 A9 00              LDA #0
02AB 95 90    HLOOP     STA WINDOW,X   clean window
02AD CA                 DEX
02AE 10 FB              BPL HLOOP
                        ; restore display card and hole card
02B0 68                 PLA          display card
02B1 85 95              STA WINDOW+5  back to display
02B3 A6 7A              LDX HOLE     get hole card
02B5 20 6D 03           JSR CREC     rebuild
02B8 20 92 03           JSR MEX      play and display
                        ; KIM plays here
02BB 20 28 03 PLAY      JSR WLITE    pause to show cards
02BE A5 9A              LDA MTOT     point total
02C0 C9 22              CMP #$22      ..22 or over?
02C2 B0 29              BCS IBUST      yup, KIM bust
02C4 65 9B              ADC MACE     add 10 for aces?
02C6 A6 91              LDX WINDOW+1  five cards?
02C8 D0 18              BNE IWIN       yes, KIM wins
02CA C9 22              CMP #$22     22+ including aces?
02CC 90 02              BCC POV      nope, count ace high
02CE A5 9A              LDA MTOT     yup, ace low
02D0 C9 17    POV       CMP #$17     17 or over?
02D2 B0 2C              BCS HOLD2     yes, stand..
02D4 20 8F 03           JSR ME        no, hit..
02D7 D0 E2              BNE PLAY     unconditional Branch
                        ; KIM wins here
02D9 20 28 03 UBUST     JSR WLITE    show player's hand..
02DC 20 55 03           JSR BUST     make BUST message..
02DF 20 28 03           JSR WLITE    ..and show it
```

```
02E2 A5 77     IWIN   LDA AMT      decrease balance
02E4 F8 38            SED SEC
02E6 E5 79            SBC BET      ..by amount of bet
02E8 85 77     JLINK  STA AMT      store new balance
02EA 4C 17 02  XLINK  JMP DEAL     next play
               ; Player wins here
02ED 20 55 03  IBUST  JSR BUST     make BUST message..
02F0 20 28 03  UWIN   JSR WLITE    display pause
02F3 A5 77     ADD    LDA AMT      increase balance
02F5 F8 18            SED CLC
02F7 65 79            ADC BET      by amount of bet
02F9 A0 99            LDY #$99     $99 maximum..
02FB 90 01            BCC NOFLO    have we passed it?
02FD 98               TYA          yes, restore $99
02FE D0 E8            BNE JLINK    unconditional branch
               ; KIM stands - compare points
0300 A2 03     HOLD2  LDX #3       flag KIM..
0302 20 0F 03         JSR SHOTOT    .. for total display
0305 A5 9A            LDA MTOT     KIM's total..
0307 C5 97            CMP UTOT     vs. Player's total..
0309 F0 DF            BEQ XLINK    same, no score;
030B B0 D5            BCS IWIN     KIM higher, wins;
030D 90 E4            BCC ADD      KIM lower, loses.

               ; subroutines start here
               ; SHTOT shows point totals per X register
030F B5 97     SHTOT  LDA UTOT,X   player's or KIM's total
0311 F8 18            SED CLC
0313 75 98            ADC UACE,X   try adding Ace points
0315 C9 22            CMP #$22     exceeds 21 total?
0317 B0 02            BCS SHOVER   yes, skip
0319 95 97            STA UTOT,X   no, make permanent
031B D8        SHOVER CLD
031C B5 97            LDA UTOT,X   get revised total
031E 48               PHA          save it
031F A0 E2            LDY #TOT-$300  set up TOT- msg
0321 20 57 03         JSR FILL      put in WINDOW
0324 68               PLA          recall total
0325 20 A6 03         JSR NUMDIS   insert in window
               ; display pause, approx 1 second
0328 A0 80     WLITE  LDY #$80     timing constant
032A 20 30 03  WDO    JSR LIGHT    illuminate screen
032D 88               DEY          countdown
032E D0 FA            BNE WDO
               ; illuminate display
0330 84 7F     LIGHT  STY YSAV     save register
0332 A0 13            LDY #$13
0334 A2 05            LDX #$5      6 digits to show
0336 A9 7F            LDA #$7F
0338 8D 41 17         STA PADD     set directional reg
033B B5 90     DIGIT  LDA WINDOW,X
033D 8D 40 17         STA SAD      character segments
0340 8C 42 17         STY SBD      character ID
0343 E6 7B     WAIT   INC PAUSE
```

```
0345 D0 FC            BNE WAIT    wait loop
0347 88 88            DEY DEY
0349 CA               DEX
034A 10 EF            BPL DIGIT
034C 20 40 1F         JSR KEYIN   switch Dir Reg
034F 20 6A 1F         JSR GETKEY  test keyboard
0352 A4 7F            LDY YSAV    restore Y value
0354 60               RTS
                  ; fill WINDOW with BUST or other message
0355 A0 E6    BUST    LDY #$BST-$300
0357 84 74    FILL    STY POINTR
0359 A0 05            LDY #5      six digits to move
035B B1 74    FILLIT  LDA (POINTR),Y  load a digit
035D 99 90 00         STA WINDOW,Y    put in window
0360 88               DEY
0361 10 F8            BPL FILLIT
0363 60               RTS
                  ; deal a card, calc value & segments
0364 A6 76    CARD    LDX DPT     Pointer in deck
0366 C6 76            DEC DPT     Move pointer
0368 B5 40            LDA DECK,X  Get the card
036A 4A 4A            LSRA LSRA   Drop the suit
036C AA               TAX         0 to 12 in X
036D 18       CREC    CLC         no-ace flag
036E D0 01            BNE NOTACE  branch if not ace
0370 38               SEC         ace flag
0371 BD BE 03         LDA VALUE,X  value from table
0374 BC CB 03         LDY SEGS,X   segments from table
0377 60               RTS
                  ; card to player, including display & count
0378 20 64 03 YOU     JSR CARD    deal card
037B E6 96            INC UCNT    card count
037D A6 96            LDX UCNT    use as display pointer
037F 94 8F            STY WINDOW-1,X   put card in Wndw
0381 A0 10            LDY #$10    ten count for aces
0383 90 02            BCC YOVER   no ace?
0385 84 98            STY UACE    ace, set 10 flag
0387 18 F8    YOVER   CLC SED
0389 65 97            ADC UTOT    add points to..
038B 85 97            STA UTOT     ..point total
038D D8               CLD
038E 60               RTS
                  ; card to KIM, including display & counts
038F 20 64 03 ME      JSR CARD     deal card
0392 C6 99    MEX     DEC MCNT     inverted count
0394 A6 99            LDX MCNT     use as (r) display pontr
0396 94 96            STY WINDOW+6,X   into window
0398 A0 10            LDY #$10    ten count for aces
039A 90 02            BCC MOVER   no ace?
039C 84 9B            STY MACE    ace, set 10 flag
039E 18 F8    MOVER   CLC SED
03A0 65 9A            ADC MTOT    add points to..
03A2 85 9A            STA MTOT      .. point total
03A4 D8               CLD
03A5 60               RTS
```

```
                ; transfer number in A  to display
03A6 48       NUMDIS PHA          save number
03A7 4A 4A           LSRA LSRA    extract left digit
03A9 4A 4A           LSRA LSRA
03AB A8              TAY
03AC B9 E7 1F        LDA TABLE,Y  convert to segments
03AF 85 94           STA WINDOW+4
03B1 68              PLA          restore digit
03B2 29 0F           AND #$0F     extract right digit
03B4 A8              TAY
03B5 B9 E7 1F        LDA TABLE,Y  convert to segments .
03B8 85 95           STA WINDOW+5
03BA 60              RTS
                ; tables in hex format
03BB 03 00 20  01 02 03 04 05 06 07 08 09 10 10 10 10
03CB F7 DB CF  E6 ED FD 87 FF EF F1 F1 F1 F1
03D8 ED F6 BE F1 F1 B8  FC F9 F8 D3
03E2 F8 DC F8 C0   FC BE ED 87 F9 DE
```

******* HEX DUMP - BLACKJACK *******

```
0200 A2 33 8A 95 40 CA 10 FA A2 02 BD BB 03 95 75 CA
0210 10 F8 AD 04 17 85 80 D8 A6 76 E0 09 B0 34 A0 D8
0220 20 57 03 A0 33 84 76 20 30 03 38 A5 81 65 82 65
0230 85 85 80 A2 04 B5 80 95 81 CA 10 F9 29 3F C9 34
0240 B0 E5 AA B9 40 00 48 B5 40 99 40 00 68 95 40 88
0250 10 D5 A0 DE 20 57 03 A5 77 20 A6 03 20 30 03 C9
0260 0A B0 F9 AA 86 79 CA 30 F3 E4 77 B0 EF A2 0B A9
0270 00 95 90 CA 10 FB 20 78 03 20 8F 03 20 78 03 20
0280 64 03 86 7A 20 28 03 20 30 03 AA CA 30 11 E4 96
0290 D0 F5 20 78 03 C9 22 B0 40 E0 05 F0 53 D0 E8 A5
02A0 95 48 A2 00 20 0F 03 A2 04 A9 00 95 90 CA 10 FB
02B0 68 85 95 A6 7A 20 6D 03 20 92 03 20 28 03 A5 9A
02C0 C9 22 B0 29 65 9B A6 91 D0 18 C9 22 90 02 A5 9A
02D0 C9 17 B0 2C 20 8F 03 D0 E2 20 28 03 20 55 03 20
02E0 28 03 A5 77 F8 38 E5 79 85 77 4C 17 02 20 55 03
02F0 20 28 03 A5 77 F8 18 65 79 A0 99 90 01 98 D0 E8
0300 A2 03 20 0F 03 A5 9A C5 97 F0 DF B0 D5 90 E4 B5
0310 97 F8 18 75 98 C9 22 B0 02 95 97 D8 B5 97 48 A0
0320 E2 20 57 03 68 20 A6 03 A0 80 20 30 03 88 D0 FA
0330 84 7F A0 13 A2 05 A9 7F 8D 41 17 B5 90 8D 40 17
0340 8C 42 17 E6 7B D0 FC 88 88 CA 10 EF 20 40 1F 20
0350 6A 1F A4 7F 60 A0 E6 84 74 A0 05 B1 74 99 90 00
0360 88 10 F8 60 A6 76 C6 76 B5 40 4A 4A AA 18 D0 01
0370 38 BD BE 03 BC CB 03 60 20 64 03 E6 96 A6 96 94
0380 8F A0 10 90 02 84 98 18 F8 65 97 85 97 D8 60 20
0390 64 03 C6 99 A6 99 94 96 A0 10 90 02 84 9B 18 F8
03A0 65 9A 85 9A D8 60 48 4A 4A 4A 4A A8 B9 E7 1F 85
03B0 94 68 29 0F A8 B9 E7 1F 85 95 60 03 00 20 01 02
03C0 03 04 05 06 07 08 09 10 10 10 10 F7 DB CF E6 ED
03D0 FD 87 FF EF F1 F1 F1 F1 ED F6 BE F1 F1 B8 FC F9
03E0 F8 D3 F8 DC F8 C0 FC BE ED 87 F9 DE
```

BLACK MATCH

by Ron Kushnier (modified by the editors)

Description -

There are 21 matches. Each player must take 1,2, or 3 matches per turn. The player who winds up with the last match loses. The player plays against the computer and goes first. Starting address - 0200, press "GO". Player enters a number on the keyboard; the left two digits display the players number. The centre digits display the computer's choice after some "think time". The rightmost digits display a running total of matches left. The computer has an I.Q. and will become dumber if you lose, smarter if you win.

```
0200 A9 21     START  LDA #$21      initial IQ
0202 85 ED            STA IQ
0204 A9 21     NEW    LDA #$21      21 matches
0206 85 F9            STA INH         to start game
0208 A9 00     PLAY   LDA #0        clear player's move
020A 85 FB            STA POINTH
020C 20 1F 1F         JSR SCANDS    light display
020F 20 6A 1F         JSR GETKEY      and test keys
0212 C9 04            CMP #4        key 4 or over?
0214 10 F2            BPL PLAY        go back
0216 C9 00            CMP #0        key 0? go back
0218 F0 EE            BPL PLAY
021A 85 FB            STA POINTH    record move
021C A9 00            LDA #0        wipe last KIM move
021E 85 FA            STA POINTL
0220 F8               SED           decimal mode
0221 38               SEC
0222 A5 F9            LDA INH       get total matches
0224 E5 FB            SBC POINTH    subtract move
0226 30 E0            BMI PLAY      not enough matches?
0228 85 F9            STA INH       OK, new total
022A A9 08            LDA #8
022C 85 EE            STA SLOW      set slow counter
022E A9 FF     TIME   LDA #$FF      slowest count into..
0230 8D 07 17         STA CLOCK      ..slowest KIM timer
0233 20 1F 1F  DISP   JSR SCANDS
0236 2C 07 17         BIT CLOCK
0239 10 F8            BPL DISP
023B C6 EE            DEC SLOW
023D D0 EF            BNE TIME
023F 18               CLC
0240 A5 F9            LDA INH       get total
0242 F0 26            BEQ DEAD      player loses?
0244 69 04            ADC #4        divide m-1 by 4
0246 E9 04     SUB    SBC #4
0248 F0 0B            BEQ DUMP
024A C9 04            CMP #4
024C B0 F8            BCS SUB       keep dividing
024E AE 46 17         LDX 1746      random, timer#2
0251 E4 ED            CPX IQ        KIM smart enough?
0253 B0 02            BCS COMP      Yes
0255 A9 01     DUMP   LDA #1        No
```

```
0257 85 FA     COMP   STA POINTL   Record the move
0259 38               SEC
025A A5 F9            LDA INH
025C E5 FA            SBC POINTL   Subtract KIM move
025E 85 F9            STA INH        from total
0260 D0 A6            BNE PLAY
0262 A2 5A            LDA #$5A     Player wins:
0264 A0 FE            LDY #$FE       SAFE
0266 46 ED            LSR IQ       get smart
0268 10 07            BPL SHOW
026A A2 DE     DEAD   LDX #$DE     KIM wins:
026C A0 AD            LDY #$AD       DEAD
026E 38               SEC
026F 26 ED            ROL IQ       get dumb
0271 86 FB     SHOW   STX POINTH
0273 84 FA            STY POINTL
0275 20 1F 1F  LOK    JSR SCANDS
0278 D0 8A            BNE NEW      new game if key
027A F0 F9            BEQ LOK
027C           end
```

****** HEX DUMP - BLACK MATCH ******

```
0200- A9 21 85 ED A9 21 85 F9 A9 00 85 FB 20 1F 1F 20
0210- 6A 1F C9 04 10 F2 C9 00 F0 EE 85 FE A9 00 85 FA
0220- F8 38 A5 F9 E5 FE 30 E0 85 F9 A9 08 85 EE A9 FF
0230- 8D 07 17 20 1F 1F 2C 07 17 10 F8 C6 EE D0 EF 18
0240- A5 F9 F0 26 69 04 E9 04 F0 0B C9 04 B0 F8 AE 46
0250- 17 E4 ED E0 02 A9 01 85 FA 38 A5 F9 E5 FA 85 F9
0260- D0 A6 A2 5A A0 FE 46 ED 10 07 A2 DE A0 AD 38 26
0270- ED 86 FB 84 FA 20 1F 1F D0 8A F0 F9
```

CARD DEALER

BY DAN LEWART

DESCRIPTION -

THIS PROGRAM WILL DEAL A FULL DECK OF 52 CARDS. THE VALUE AND SUIT OF THE CARDS APPEARS IN THE RIGHT TWO DIGITS OF THE DISPLAY. PRESS ANY KEY TO GET ANOTHER CARD. EACH WILL APPEAR ONLY ONCE. WHEN ALL CARDS HAVE BEEN DEALT, THE PROGRAM MUST BE RESTARTED AT 0000.

```
0000   A2 06      INIT      LDX #$06       CLEAR DISPLAY
0002   A0 00                LDY #$00       (8C-91)=0
0004   94 8B      INIT 1    STY 008B,X
0006   CA                   DEX
0007   D0 FB                BNE INIT 1
0009   D8                   CLD
000A   A2 34                LDX #$34       FILL DECK
000C   86 92                STX 0092       STORE CARDS LEFT (52)
000E   C8                   INY            (93-C6)=1
000F   94 92      INIT 2    STY 0092,X
0011   CA                   DEX
0012   D0 FB                BNE INIT 2
0014   A5 92      NEWCRD    LDA 0092       DECK FINISHED?
0016   D0 03                BNE RANDOM
0018   4C 4F 1C             JMP START      YES, STOP
001B   AD 04 17   RANDOM    LDA 1704       GET RANDOM # (1-FF)
001E   D0 0B                BNE FASTER
0020   AD 44 17             LDA 1744
0023   D0 06                BNE FASTER
0025   A5 92                LDA 0092       BOTH CLOCKS OUT OF RANGE
0027   4A                   LSR            # APPROX. MIDDECK
0028   18                   CLC
0029   69 01                ADC #$01
002B   C5 92      FASTER    CMP 0092       GET NUMBER 1-34
002D   90 07                BCC FIND
002F   F0 05                BEQ FIND
0031   E5 92                SBC 0092
0033   4C 2B 00             JMP FASTER
0036   A2 33      FIND      LDX #$33       FIND THE CARD
0038   38         FIND 1    SEC            KEEP SUBTRACTING CARD
0039   F5 93                SBC 0093,X     CARD=0 MEANS PICKED
003B   F0 03                BEQ UPDATE     CARD=1 MEANS IN DECK
003D   CA                   DEX            X=CARD POSITION
003E   10 F8                BPL FIND 1
0040   95 93      UPDATE    STA 0093,X     CARD=0
0042   C6 92                DEC 0092       1 LESS CARD LEFT
0044   8A                   TXA            GET FIRST 6 BITS OF X
0045   4A                   LSR            Y=(0-C)
0046   4A                   LSR
0047   A8                   TAY
```

```
0048  B9 7B 00          LDA 007B,Y     GET VALUE FROM VALTBL
004B  85 90             STA 0090       STORE AS 5TH DISPLAY DIGIT
004D  8A                TXA            GET LAST 2 BITS OF X
004E  29 03             AND #$03       Y=(0-3)
0050  A8                TAY
0051  B9 88 00            LDA 0088,Y    GET SUIT FROM SUITBL
0054  85 91             STA 0091       STORE AS 6TH DISP. DIGIT
0056  20 62 00  K DOWN   JSR DISP       DISPLAY (8C-91)
0059  D0 FB             BNE K DOWN     UNTIL KEY UP
005B  20 62 00  K UP     JSR DISP       DISPLAY (8C-91)
005E  D0 B4              BNE NEWCRD     UNTIL KEY DOWN
0060  F0 F9             BEQ K UP
0062  A9 7F      DISP   LDA #$7F       SEGMENTS TO OUTPUT
0064  8D 41 17           STA 1741
0067  A0 00             LDY #$00       INITIALIZE
0069  A2 08             LDX #$08
006B  B9 8C 00  DISP 1   LDA 008C,Y     GET CHARACTER
006E  84 FC              STY OOFC
0070  20 4E 1F           JSR 1F4E       DISPLAY CHARACTER
0073  C8                 INY            NEXT CHARACTER
0074  C0 06             CPY #$06
0076  90 F3             BCC DISP 1
0078  4C 3D 1F           JMP 1F3D       DONE, KEY DOWN?

                  ******** TABLES ********
007B  77          VALTBL   "A"
007C  5B                   "2"
007D  4F                   "3"
007E  66                   "4"
007F  6D                   "5"
0080  7D                   "6"
0081  07                   "7"
0082  7F                   "8"
0083  6F                   "9"
0084  78                   "T"
0085  1E                   "J"
0086  67                   "Q"
0087  70                   "K"
0088  6D          SUITBL   "S"
0089  76                   "H"
008A  5E                   "D"
008B  39                   "C"
```

****** HEX DUMP - CARD DEALER ******

```
0000 A2 06 A0 00 94 8B CA D0 FB D8 A2 34 86 92 C8 94
0010 92 CA D0 FB A5 92 D0 03 4C 4F 1C AD 04 17 D0 0B
0020 AD 44 17 D0 06 A5 92 4A 18 69 01 C5 92 90 07 F0
0030 05 E5 92 4C 2B 00 A2 33 38 F5 93 F0 03 CA 10 F8
0040 95 93 C6 92 8A 4A 4A A8 B9 7B 00 85 90 8A 29 03
0050 A8 B9 88 00 85 91 20 62 00 D0 FB 20 62 00 D0 B4
0060 F0 F9 A9 7F 8D 41 17 A0 00 A2 08 B9 8C 00 84 FC
0070 20 4E 1F C8 C0 06 90 F3 4C 3D 1F 77 5B 4F 66 6D
0080 7D 07 7F 6F 78 1E 67 70 6D 76 5E 39
```

CHESS CLOCK

BY CASS LEWART

DESCRIPTION -

THE PROGRAM STARTS AT LOCATION 0200. TWO INDEPENDENT CLOCKS ARE OPERATED BY THE TWO PLAYERS BY DEPRESSING KEYS 1 OR 2 RESPECTIVELY. THE RIGHT TWO DIGITS SHOW THE MOVE NUMBER, THE LEFT FOUR DIGITS SHOW MINUTES AND SECONDS. MAXIMUM TIME IS 99 MINUTES 59 SEC. THE CLOCK PROGRAM CAN BE FINELY TUNED BY CHANGING THE VALUE OF WORD 027F, INCREASE BY 1 SLOWS THE CLOCK BY APPROXIMATELY 6 SEC/24 HOURS AND VICE VERSA.

```
0200  A9 00           LDA #$00       ZERO ALL OF PAGE ZERO
0202  AA              TAX
0203  9D 00 00  ZERO  STA 0000,X
0206  E8              INX
0207  D0 FA           BNE ZERO
0209  20 1F 1F  DISP  JSR SCANDS      DISPLAY ZEROS
020C  20 6A 1F        JSR GETKEY     KEY PRESSED?
020F  C9 02           CMP #$02       KEY # 2?
0211  D0 F6           BNE DISP       NO, WAIT TILL 2 DOWN
0213  A9 01     LOOP  LDA #$01       FLAG TO 1
0215  85 D4           STA 00D4       (CLOCK #1 TO RUN)
0217  20 60 02        JSR TIME       GET CLOCK RUNNING
021A  20 31 02        JSR SAVE       SAVE TIME ON DISPLAY
021D  A9 02           LDA #$02       FLAG TO 2
021F  85 D4           STA 00D4       (CLOCK #2 TO RUN)
0221  20 60 02        JSR TIME       GET OTHER CLOCK RUNNING
0224  18              CLC            ...INCREMENT MOVE
0225  A5 F9           LDA 00F9       NUMBER...
0227  69 01           ADC #$01
0229  85 F9           STA 00F9
022B  20 31 02        JSR SAVE       SAVE CLOCK 2 TIME
022E  4C 13 02        JMP LOOP       BACK TO CLOCK # 1
          ***** SAVE TIME INDICATED SUBROUTINE *****
0231  A9 02     SAVE  LDA #$02       CLOCK # 2?
0233  C5 D4           CMP 00D4
0235  D0 11           BNE CLK1       NO, STORE FOR CLOCK # 1
0237  A5 FB           LDA 00FB       ... STORE VALUES FOR
0239  85 D2           STA 00D2       CLOCK # 2 IN 00D2
023B  A5 FA           LDA 00FA       AND 00D3 ...
023D  85 D3           STA 00D3
023F  A5 D0           LDA 00D0       ... LOAD DISPLAY WITH
0241  85 FB           STA 00FB       VALUES FOR CLOCK # 1 ...
0243  A5 D1           LDA 00D1
0245  85 FA           STA 00FA
0247  60              RTS
0248  A5 FB     CLK1  LDA 00FB       ... STORE VALUES FOR
024A  85 D0           STA 00D0       CLOCK # 1 IN 00D0
024C  A5 FA           LDA 00FA       AND 00D1 ...
024E  85 D1           STA 00D1
0250  A5 D2           LDA 00D2       ... LOAD DISPLAY WITH
0252  85 FB           STA 00FB       VALUES FOR CLOCK # 2 ...
0254  A5 D3           LDA 00D3
0256  85 FA           STA 00FA
0258  60              RTS
```

```
                    ***** CLOCK ADVANCE SUBROUTINE *****
0260  F8          TIME    SED             SET DECIMAL MODE
0261  A9 04               LDA #$04        TIME MULTIPLIER TO 4
0263  85 D5               STA 00D5
0265  A9 F0        LOAD    LDA #$F0        SET TIMER
0267  8D 07 17             STA 1707
026A  20 1F 1F    LITE    JSR SCANDS      DISPLAY CLOCK
026D  20 6A 1F             JSR GETKEY      GET KEYBOARD ENTRY
0270  C5 D4               CMP 00D4        EQUAL TO FLAG?
0272  D0 01               BNE WAIT        NO,TIME OUT THEN UPDATE
0274  60                  RTS             YES, RETURN FROM SUBR.
0275  2C 07 17  WAIT      BIT 1707        TIME DONE?
0278  10 F0               BPL LITE        NOT YET
027A  C6 D5               DEC 00D5        DECREMENT TIME MULT.
027C  D0 E7               BNE LOAD        NOT ZERO, RESET TIMER
027E  A9 BF               LDA #$BF        LAST LITTLE BIT OF TIME
0280  8D 06 17            STA 1706        INTO TIMER
0283  2C 07 17  TINY      BIT 1707        DONE?
0286  10 FB               BPL TINY        NO
0288  18                  CLC             ..ONE SECOND ADDED
0289  A5 FA               LDA 00FA        TO CLOCK..
028B  69 01               ADC #$01
028D  85 FA               STA 00FA        (CENTER TWO DIGITS)
028F  C9 60               CMP #$60        A MINUTE UP?
0291  D0 05               BNE NOMN        NOT YET
0293  38                  SEC             YES, SEC. TO ZERO
0294  A9 00               LDA #$00
0296  85 FA               STA 00FA
0298  A5 FB      NOMN      LDA 00FB        ... MINUTES INCREMENTED
029A  69 00               ADC #$00        IF CARRY SET ...
029C  85 FB               STA 00FB
029E  4C 60 02            JMP TIME        LOOP
```

***** HEX DUMP - CHESS CLOCK *****

```
0200-   A9 00 AA 9D 00 00 E8 D0 FA 20 1F 1F 20 6A 1F C9
0210-   02 D0 F6 A9 01 85 D4 20 60 02 20 31 02 A9 02 85
0220-   D4 20 60 02 18 A5 F9 69 01 85 F9 20 31 02 4C 13
0230-   02 A9 02 C5 D4 D0 11 A5 FB 85 D2 A5 FA 85 D3 A5
0240-   D0 85 FB A5 D1 85 FA 60 A5 FE 85 D0 A5 FA 85 D1
0250-   A5 D2 85 FB A5 D3 85 FA 60
0260-   F8 A9 04 85 D5 A9 F0 8D 07 17 20 1F 1F 20 6A 1F
0270-   C5 D4 D0 01 60 2C 07 17 10 F0 C6 D5 D0 E7 A9 BF
0280-   8D 06 17 2C 07 17 10 FB 18 A5 FA 69 01 85 FA C9
0290-   60 D0 05 38 A9 00 85 FA A5 FB 69 00 85 FB 4C 60
02A0-   02
```

CLOCK

- Charles Parsons

This clock routine uses KIM's built in interval timer with the interrupt option. It works by loading $F4 into the timer (/1024) each time the Non-Maskable Interrupt (NMI) occurs. This theoretically produce a time of 249,856 microseconds or just under ¼ second. The adjustment to ¼ second is done with the timer (/1) in the interrupt routine. A fine adjustment of the clock can be made by modifying the value in location $0366. Only two subroutines will be documented here (ESCAPE TO KIM & HOUR CHIME) but many more can be added by simply replacing the NOP codes starting at $03DE with jumps to your own subroutines. For instance, a home control system could be set up using the clock program.

The escape to KIM allows KIM to run without stopping the clock. This means that you can run other programs simultaneously with the clock program unless your program also needs to use the NMI (such as single step operation) or if there could be a timing problem (such as with the audio tape operation). Pressing the KIM GO button will get you out of the KIM loop.

To start the clock:

1. Connect PB7 (A-15) to NMI (E-6).
2. Initialize NMI pointer (17FA, 17FB) with 60 and 03.
3. Set up the time and AM-PM counter locations in page zero.
4. Go to address $03C0 and press GO.

To get back into the clock display mode if the clock is running - start at location $03C9.

NOTE: These routines are not listed in any particular order so be watchful of the addresses when you load them.

PAGE ZERO LOCATIONS

0070	NOTE	Sets frequency of note
0080	QSEC	¼ second counter
0081	SEC	second counter
0082	MIN	minute counter
0083	HR	hour counter
0084	DAY	day counter for AM-PM

INTERRUPT ROUTINE

This routine uses the NMI to update a clock in zero page locations. Since the crystal may be slightly off one MHz a fine adjustment is located at 0366. NMI pointers must be set to the start of this program.

```
0360  48          PHA             save A
0361  8A          TXA
0362  48          PHA             save X
0363  98          TYA
0364  48          PHA             save Y
0365  A983        LDA #$83        fine adjust timing
0367  8D0417      STA TIME4
036A  2C0717  TM  BIT TIME7       test timer
036D  10FB        BPL TM          loop until time out
036F  E680        INC QSEC        count ¼ seconds
0371  A904        LDA #$04        do four times before
0373  C580        CMP QSEC        updating seconds
0375  D038        BNE RTN
0377  A900        LDA #$00        reset ¼ second counter
0379  8580        STA QSEC
037B  18          CLC
037C  F8          SED             advance clock in decimal
037D  A581        LDA SEC
037F  6901        ADC #$01        advance seconds
0381  8581        STA SEC
0383  C960        CMP #$60        until 60 seconds
0385  D028        BNE RTN
0387  A900        LDA #$00        then start again
0389  8581        STA SEC
038B  A582        LDA MIN
038D  18          CLC
038E  6901        ADC #$01        and advance minutes
0390  8582        STA MIN
0392  C960        CMP #$60        until 60 minutes
0394  D019        BNE RTN
0396  A900        LDA #$00        then start again
0398  8582        STA MIN
039A  A583        LDA HR          and advance hours
039C  18          CLC
039D  6901        ADC #$01
039F  8583        STA HR
03A1  C912        CMP #$12        until 12 hours
03A3  D002        BNE TH
03A5  E684        INC DAY         advance ½ day
03A7  C913    TH  CMP #$13        if 13 hours
03A9  D004        BNE RTN         start again with one
03AB  A901        LDA #$01
03AD  8583        STA HR
03AF  D8      RTN CLD             go back to hex mode
03B0  A9F4        LDA #$F4        start timer with interrupt
03B2  8D0F17      STA TIMEF       in 249,856 microseconds
```

```
03B5  68          PLA
03B6  A8          TAY              restore Y
03B7  68          PLA
03B8  AA          TAX              restore X
03B9  68          PLA              restore A
03BA  40          RTI              return from interrupt
```

ESCAPE TO KIM IF 1 ON KIM IS PRESSED

This is a subroutine which will return to the KIM monitor routine without stopping the real time clock. It is done by pressing 1 on the KIM keyboard.

```
0300  206A1F  KIM   JSR GETKEY      go back to KIM if
0303  C901          CMP #$01        KIM keyboard is one
0305  D00D          BNE ENDR
0307  201F1F        JSR SCANDS      delay to make sure
030A  206A1F        JSR GETKEY
030D  C901          CMP #$01
030F  D003          BNE ENDR
0311  4C051C        JMP SAVE1
0314  60      ENDR  RTS
```

TWO TONE SOUND TO INDICATE HOURS

This is a subroutine which when added to the clock display routine will use the real time clock data to produce one sound per hour on the hour. The output is a speaker circuit as shown on Pg. 57 of the KIM-1 Manual. It is hooked to PB0 rather than PA0. The specific notes can be changed by altering 0330 and 033C.

```
0320  A582    BEEP  LDA MIN        on the hour?
0322  D029          BNE END         if not return
0324  A581          LDA SEC         execute until SEC = HR
0326  38            SEC
0327  E583          SBC HR
0329  1024          BPL END
032B  A580    AGAIN LDA QSEC        first ¼ second?
032D  D006          BNE ONE
032F  A91E          LDA #$1E        set high note
0331  8570          STA NOTE
0333  D00A          BNE GO          sound note for ¼ second
0335  A901    ONE   LDA #$01        second ¼ second?
0337  C580          CMP QSEC
0339  D014          BNE END
033B  A928          LDA #$28        set low note
033D  8570          STA NOTE
033F  A901    GO    LDA #$01        set I/O ports
0341  8D0317        STA PBDD
0344  EE0217        INC PBD         toggle speaker
0347  A570          LDA NOTE
0349  AA            TAX             set delay
034A  CA      DEC   DEX
034B  10FD          BPL DEC
034D  30DC          BMI AGAIN       keep sounding
034F  60      END   RTS
```

DISPLAY CLOCK ON KIM-1 READOUT

```
03C0  A900          LDA #$00          reset ¼ second counter
03C2  8580          STA QSEC
03C4  A9F4          LDA #$F4          start timer with interrupt
03C6  8D0F17        STA TIMEF
03C9  A581    DSP   LDA SEC           start here if clock is running
03CB  85F9          STA INH           display clock on KIM
03CD  A582          LDA MIN
03CF  85FA          STA POINTL
03D1  A583          LDA HR
03D3  85FB          STA POINTH
03D5  201F1F        JSR SCANDS
03D8  200003        JSR KIM           escape to KIM
03DB  202003        JSR BEEP          sound on the hour
03DE  EAEAEA
03E1  EAEAEA
03E4  EAEAEA
03E7  EAEAEA
03EA  EAEAEA
03ED  EAEAEA
03F0  EAEAEA
03F3  EAEAEA
03F6  EAEAEA
03F9  EAEAEA
03FC  4CC903        JMP DSP
```

***** Hex Dump - Clock *****

```
0300-   20 6A 1F C9 01 D0 0D 20 1F 1F 20 6A 1F C9 01 D0
0310-   03 4C 05 1C 60
0320-   A5 82 D0 29 A5 81 38 E5 83 10 24 A5 80 D0 06 A9
0330-   1E 85 70 D0 0A A9 01 C5 80 D0 14 A9 28 85 70 A9
0340-   01 8D 03 17 EE 02 17 A5 70 AA CA 10 FD 30 DC 60

0360-   48 8A 48 98 48 A9 83 8D 04 17 20 C0 17 10 FB E6
0370-   80 A9 04 C5 80 D0 38 A9 00 85 80 18 F8 A5 81 69
0380-   01 85 81 C9 60 D0 28 A9 00 85 81 A5 82 18 69 01
0390-   85 82 C9 60 D0 19 A9 00 85 82 A5 83 18 69 01 85
03A0-   83 C9 12 D0 02 E6 84 C9 13 D0 04 A9 01 85 83 D8
03B0-   A9 F4 8D 0F 17 68 A8 68 AA 68 40
03C0-   A9 00 85 80 A9 F4 8D 0F 17 A5 81 85 F9 A5 82 85
03D0-   FA A5 83 85 FB 20 1F 1F 20 00 03 20 20 03 EA EA
03E0-   EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA
03F0-   EA EA EA EA EA EA EA EA EA EA EA 4C C9 03
```

CODE TEST

BY STAN OCKERS

DESCRIPTION -

THIS PROGRAM REQUIRES THAT A SPEAKER BE HOOKED TO PA0 AS IN FIGURE 5.1 OF THE KIM MANUAL. WHEN STARTED AT 0200, THE PROGRAM WILL SEND 5 LETTER CODE GROUPS, (INTERNATIONAL MORSE), OVER THE SPEAKER. THE CODE GROUPS WILL CONSIST OF RANDOM CHARACTERS INCLUDING A-Z, 0-9, A PERIOD, COMMA, QUESTION MARK AND EQUAL SIGN. AFTER THIS TRANSMISSION, YOUR RECEPTION CAN BE CHECKED BECAUSE THE GROUPS SENT WILL BE SHOWN ON THE DISPLAY. PRESSING ANY KEY WILL CAUSE THE NEXT GROUP TO BE DISPLAYED. LIMITATIONS IMPOSED BY THE 7 SEGMENT DISPLAYS MAKE SOME CHARACTERS PRETTY STRANGE AND THERE IS SOME REDUNDANCY; BUT BY SLOWING THE TRANSMISSION YOU SHOULD BE ABLE TO FIGURE OUT WHAT EACH CHARACTER IS.

```
0200  A2 0C          LDX #$0C      ... INITIALIZATION ...
0202  BD DF 02 INIT  LDA 02DF,X    .. 12 VALUES ARE LOADED
0205  95 E2          STA 00E2,X    FROM 00E2 ON UP ..
0207  CA             DEX
0208  10 F8          BPL INIT
020A  A2 04    GRUP  LDX #$04      (SPACE LENGTH)
020C  20 A0 02       JSR SPACE     SPACE FOR ANOTHER GROUP
020F  A9 06          LDA #$06      GROUP SIZE, 5 CHAR.
0211  85 E0          STA 00E0
0213  C6 E0    CHAR  DEC 00E0      NEXT CHAR. IN GROUP
0215  F0 F3          BEQ GRUP      FINISHED, GET NEW GROUP
0217  A2 03          LDX #$03      (SPACE LENGTH)
0219  20 A0 02       JSR SPACE     SPACE BETWEEN CHAR.
021C  20 CB 02 NUMB  JSR RAND      GET A RANDOM #
021F  29 3F          AND #$3F      MAKE SURE POSITIVE
0221  C9 28          CMP #$28      LESS THAN 41 (DECIMAL)?
0223  10 F7          BPL NUMB      NO, GET ANOTHER
0225  AA             TAX           USE AS INDEX
0226  BD 13 03       LDA 0313,X    GET DISPLAY CONVERSION
0229  A4 E2          LDY 00E2      CHAR. INDEX IN Y
022B  99 3B 03       STA 033B,Y    STORE CONVERSION
022E  E6 E2          INC 00E2      INDEX UP ONE
0230  A5 E2          LDA 00E2      LAST CHARACTER?
0232  C9 1A          CMP #$1A
0234  F0 20          BEQ DEBO      YES, GO READOUT
0236  BD EB 02       LDA 02EB,X    GET CODE CHARACTER
0239  85 DF          STA 00DF      TEMPORARY STORE
023B  06 DF    BITS  ASL 00DF      SHIFT
023D  F0 D4          BEQ CHAR      EMPTY, GET NEXT CHAR.
023F  B0 0D          BCS DASH      IF CARRY SET, SEND DASH
0241  A2 01          LDX #$01      ..ELSE SEND DOT
0243  20 82 02       JSR MARK
0246  A2 01    SPAC  LDX #$01      THEN SPACE
```

```
0248  20 A0 02          JSR SPACE
024B  18                CLC
024C  90 ED             BCC BITS        UNCOND. JUMP
024E  A2 03     DASH    LDX #$03        (DASH LENGTH)
0250  20 82 02          JSR MARK        SEND A DASH
0253  18                CLC
0254  90 F0             BCC SPAC        UNCOND. JUMP
0256  20 8E 1E  DEBO    JSR INIT1             ..DEBOUNCE KEY..
0259  20 B1 02          JSR DISP
025C  D0 F8             BNE DEBO        WAIT FOR KEY RELEASE
025E  20 B1 02  WAIT    JSR DISP
0261  F0 FB             BEQ WAIT        WAIT FOR KEY DOWN
0263  18                CLC
0264  A5 E4             LDA 00E4        ..UPDATE POINTER TO
0266  69 05             ADC #$05        POINT AT NEXT GROUP..
0268  85 E4             STA 00E4
026A  A0 04             LDY #$04        ..LOAD WINDOWS 00E8-
026C  B1 E4     WIND    LDA (00E4),Y    00EC WITH CONVERSIONS
026E  99 E8 00          STA 00E8,Y      FOR DISPLAY..
0271  88                DEY
0272  10 F8             BPL WIND
0274  C6 E3             DEC 00E3        LAST GROUP?
0276  D0 DE             BNE DEBO        NO, GET ANOTHER
0278  A9 36             LDA #$36        REINITILIZE POINTER
027A  85 E4             STA 00E4        TO RUN THRU GROUPS AGAIN
027C  A9 05             LDA #$05
027E  85 E3             STA 00E3
0280  D0 D4             BNE DEBO        UNCOND. JUMP
          ***** MARK SUBROUTINE *****
0282  86 DD             STX 00DD        TEMP. STORE
0284  A5 E6     TIMM    LDA 00E6        SPEED BYTE
0286  8D 07 17          STA 1707        START TIMER
0289  A9 01             LDA #$01        PA0 TO OUTPUT
028B  8D 01 17          STA 1701
028E  EE 00 17  TOGG    INC 1700        TOGGLE PA0
0291  A6 E7             LDX 00E7        DETERMINE FREQ.
0293  CA        FREQ    DEX
0294  D0 FD             BNE FREQ
0296  2C 07 17          BIT 1707      TIME UP?
0299  10 F3             BPL TOGG       NO
029B  C6 DD             DEC 00DD       DETERMINE MARK LENGTH
029D  D0 E5             BNE TIMM
029F  60                RTS

          ****** SPACE SUBROUTINE ******
02A0  86 DD     DISP    STX 00DD        TEMP. STORE
02A2  A5 E6     TIMS    LDA 00E6        SPEED BYTE
02A4  8D 07 17          STA 1707        START TIMER
02A7  2C 07 17  HOLD    BIT 1707        DONE?
02AA  10 FB             BPL HOLD        NO
02AC  C6 DD             DEC 00DD        FULL TIME UP?
02AE  D0 F2             BNE TIMS        NO
02B0  60                RTS
```

```
          ***** Display Subroutine *******
02B1 A9 7F      DISP    LDA #$7F    change segments..
02B3 8D 41 17           STA PADD    ..to outputs
02B6 A0 00              LDY #0      init. recall index
02B8 A2 09              LDX #9      init. digit number
02BA B9 E8 00   SIX     LDA 00E8,Y  get character
02BD 84 FC              STY YSAV    save Y
02BF 20 4E 1F           JSR DISPL   display character
02C2 C8                 INY         set up for next char
02C3 C0 06              CPY #6      6 chars displayed?
02C5 90 F3              BCC SIX     no, do more
02C7 20 3D 1F           JSR KEYTS   key down?
02CA 60                 RTS
          ***** Random Number Subroutine ******
02CB 38 D8      RAND    SEC CLD
02CD A5 D1              LDA RND+1   from Kim User Notes
02CF 65 D4              ADC RND+3   vol 1, #1
02D1 65 D5              ADC RND+4   (J. Butterfield)
02D3 85 D0              STA RND
02D5 A2 04              LDX #4
02D7 B5 D0      ROLL    LDA RND,X
02D9 95 D1              STA RND+1,X
02DB CA                 DEX
02DC 10 F9              BPL ROLL
02DE 60                 RTS
          ***** Initialization Values *******
02DF 00 05 36 03 33 66 C0 C0 C0 C0 C0 00
          ***** Morse Code Characters *******
02EB 60 88 A8 90 40 28 D0 08 20 78 B0 48 E0 A0 F0 68
02FB D8 50 10 C0 30 18 70 98 B8 C8 FC 7C 3C 1C 0C 04
030B 84 C4 E4 F4 56 CE 32 8C
          ***** Display Characters **********
0313 F7 FC B9 DE F9 F1 BD F6 84 9E F0 B8 B7 D4 DC F3
0323 E7 D0 ED F8 BE EA 9C 94 EE C9 BF 86 DB CF E6 ED
0333 FD 87 FF EF 90 84 D3 C8
       *** Characters sent stored in 033B - 03FF ***
```

CHANGES: The program sends and displays 5 groups of 5 characters each. This may be changed, although you may need to do some debugging along the way. Important parameters are:

--0233 contains characters-to-be-sent, plus one;
--02E0 contains groups-to-be-displayed-after-transmission;
--02E3 contains speed-of-transmission; hex 33 gives about 16 groups/min, hex 66 gives 8 words/min
--02E4 varies the tone
--02E1/02E2 points at the block of characters to be sent;
--0222 controls the character set; 1A for letters only.

See Byte magazine, October 1976, page 36, for details of morse character storage.

CRAPS

BY JIM BUTTERFIELD

DESCRIPTION -

SET ADDRESS 0200, THEN HOLD "GO" DOWN .. YOU'LL SEE:

- 2 DICE "ROLLING" ON THE LEFT
- $10 BALANCE ON THE RIGHT

LET "GO" GO ... THE DICE WILL STOP ROLLING, AND YOU'LL GET:

- A WIN ON A TOTAL OF 7 OR 11; YOU'LL SEE YOUR DOLLAR BALANCE RISE; OR
- A LOSS ON TOTALS OF 2,3, OR 12; YOUR DOLLAR BALANCE WILL DROP; OR
- A "POINT" - THE CENTER SEGMENTS WILL LIGHT WITH THE ROLL AND YOU MUST TRY TO ROLL THIS TOTAL AGAIN BEFORE YOU ROLL 7 -

PUSH THE "GO" BUTTON ONLY ON THE FIRST ROLL. FOR SUBSEQUENT ROLLS, PUSH ANOTHER BUTTON.

```
0200 D8        START  CLD
0201 20 40 1F         JSR KEYIN
0204 20 6A 1F         JSR GETKEY
0207 C5 40            CMP LAST
0209 F0 79            BEQ LIGHT   same key as before?
020B 85 40            STA LAST
020D 49 15            EOR #$15    no-key test
020F 85 41            STA FLAG    into flag
0211 C9 06            CMP #6      GO key?
0213 D0 05            BNE NOGO    nope..
0215 A9 10            LDA #$10    yes, $10
0217 20 A9 02         JSR DOBUX   put in window
021A AD 04 17 NOGO    LDA TIMER   random value
021D A2 C0            LDX #$C0    divide by 6
021F 86 4E            STX DIVR
0221 A2 05            LDX #5
0223 C5 4E     RNDLP  CMP DIVR    divide..
0225 90 02            BCC RNDOV     ..a..
0227 E5 4E            SBC DIVR      ..digit
0229 46 4E     RNDOV  LSR DIVR
022B CA               DEX
022C 10 F5            BPL RNDLP
022E AA               TAX           die 0-5
022F E8               INX           die 1-6
0230 BD E7 1F         LDA TABLE,X   segment
0233 A4 41            LDY FLAG    which die?
0235 F0 06            BEQ PLAY    second?
0237 86 42            STX DIE     first, save it..
0239 85 43            STA WINDX   ..& segment
023B D0 47            BNE LIGHT   unconditional
023D 85 47     PLAY   STA WINDOW+1  show die..
023F A5 43            LDA WINDX     ..and other
0241 85 46            STA WINDOW          one
0243 A5 44            LDA BUX     out of dough?
```

```
0245 F0 3D          BEQ LIGHT ..no bread
0247 8A 18          TXA CLC
0249 65 42          ADC DIE  add other die
024B C5 45          CMP POINT get the point?
024D F0 28          BEQ WIN   ..yup
024F A6 45          LDX POINT  point=zero...
0251 F0 12          BEQ FIRST ..first roll
0253 C9 07          CMP #7   seven you lose
0255 D0 2D          BNE LIGHT  ..nope
0257 A5 44   LOSE   LDA BUX
0259 F0 05          BEQ LOSX  nough dough?
025B 18 F8          CLC SED   decimal add..
025D E9 00          SBC #0    neg one
025F D8             CLD
0260 20 A9 02       JSR DOBUX put in window
0263 D0 1F          BNE LIGHT unconditional
0265 A6 46   FIRST  LDX WINDOW  copy point
0267 86 48          STX WINDOW+2
0269 A6 47          LDX WINDOW+1
026B 86 49          STX WINDOW+3
026D 85 45          STA POINT
026F AA             TAX           point value
0270 BD C6 02       LDA TAB-2,X   'win' table
0273 F0 0F          BEQ LIGHT     ..says point
0275 30 E0          BMI LOSE     ..says craps
0277 A5 44   WIN    LDA BUX      ..says win
0279 C9 99          CMP #$99  maximum bucks?
027B F0 04          BEQ WINX  yes, skip add
027D F8             SED       decimally add..
027E 69 01          ADC #1    ..one
0280 D8             CLD
0281 20 A9 02 WINX  JSR DOBUX make segments
0284 A5 41   LIGHT  LDA FLAG  still rolling?
0286 F0 04          BEQ NOINC   ..nope;
0288 E6 46          INC WINDOW  ..yup, so..
028A E6 47          INC WINDOW+1 ..roll em!
028C A9 7F   NOINC  LDA #$7F
028E 8D 41 17       STA PADD
0291 A0 13          LDY #$13
0293 A2 05          LDX #5
0295 B5 46   LITE   LDA WINDOW,X
0297 8D 40 17       STA SAD
029A 8C 42 17       STY SBD
029D E6 4F   PAWS   INC PAUSE
029F D0 FC          BNE PAWS
02A1 88 88          DEY DEY
02A3 CA             DEX
02A4 10 EF          BPL LITE
02A6 4C 00 02       JMP START

02A9 85 44   DOBUX  STA BUX
02AB A0 00          LDY #0
02AD 84 45          STY POINT clear point
02AF 84 48          STY WINDOW+2  ..and..
```

```
02B1 84 49          STY WINDOW+3  display
02B3 A8 4A          TAY LSRA
02B5 4A 4A 4A       LSRA LSRA LSRA
02B8 AA             TAX
02B9 BD E7 1F       LDA TABLE,X
02BC 85 4A          STA WINDOW+4
02BE 98             TYA
02BF 29 0F          AND #$0F
02C1 AA             TAX
02C2 BD E7 1F       LDA TABLE,X
02C5 85 4B          STA WINDOW+5
02C7 60             RTS
02C8 FF FF 00 00 00 01 00 00 00 01 FF   (TAB)
```

****** HEX DUMP - CRAPS ******

```
0200-  D8 20 40 1F 20 6A [illegible] C5 40 F0 79 85 40 49 15 85
0210-  41 C9 06 D0 05 A9 [illegible] 20 A9 02 AD 04 17 A2 C0 86
0220-  4E A2 05 C5 4E 90 02 E5 4E 46 4E CA 10 F5 AA E8
0230-  BD E7 1F A4 41 F0 06 86 42 85 43 D0 47 85 47 A5
0240-  43 85 46 A5 44 F0 3D 8A 18 65 42 C5 45 F0 28 A6
0250-  45 F0 12 C9 07 D0 2D A5 44 F0 05 18 F8 E9 00 D8
0260-  20 A9 02 D0 1F A6 46 86 48 A6 47 86 49 85 45 AA
0270-  BD C6 02 F0 0F 30 E0 A5 44 C9 99 F0 04 F8 69 01
0280-  D8 20 A9 02 A5 41 F0 04 E6 46 E6 47 A9 7F 8D 41
0290-  17 A0 13 A2 05 B5 46 8D 40 17 8C 42 17 E6 4F D0
02A0-  FC 88 88 CA 10 EF 4C 00 02 85 44 A0 00 84 45 84
02B0-  48 84 49 A8 4A 4A 4A 4A AA BD E7 1F 85 4A 98 29
02C0-  0F AA BD E7 1F 85 4B 60 FF FF 00 00 00 01 00 00
02D0-  00 01 FF
```

Coding notes: CRAPS is a highly top-down program. The program always flows from START to LIGHT and back again with few breaks in sequence. The dice are randomized from TIMER (1704) and RNDLP contains a small division routine, dividing by 6; the remainder, randomly 0 to 5, gives the roll of one die. On the first roll of a run, we use the table at 02 C8 to analyze the total: in this table, FF means you lose and 01 means you win. FLAG is zero if you're not pushing any button. Segments for the display are stored in table WINDOW, 0046 to 004B.

DUEL

BY STAN OCKERS

DESCRIPTION -

THIS IS A GAME FOR TWO PLAYERS. WHEN THE PROGRAM IS STARTED AT 0200, EACH PLAYER IS GIVEN TEN POINTS AS INDICATED ON OPPOSITE SIDES OF THE DISPLAY. THE CENTER DIGITS WILL BE BLANK. AFTER A RANDOM DELAY, THE CENTER DIGITS WILL LIGHT. THE FIRST PLAYER TO PRESS HIS KEY WILL INCREASE HIS SCORE BY ONE AND DECREASE HIS OPPONENT'S BY ONE. THE CENTER DIGITS WILL THEN BLANK FOR ANOTHER RANDOM DELAY. IF A PLAYER PRESSES HIS KEY WHILE THE CENTER DIGITS ARE BLANK, HIS SCORE WILL BE DECREASED BY ONE. WHEN ONE PLAYER REACHES ZERO THE GAME IS OVER AND MUST BE RESTARTED AT 0200. THE PLAYER TO THE LEFT USES KEY ZERO AND THE ONE ON THE RIGHT USES KEY SEVEN.

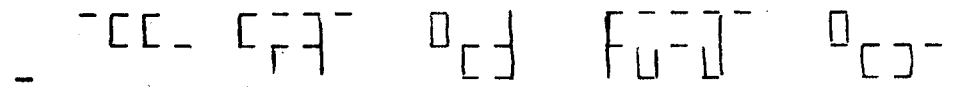
```
0200   A9 10          LDA #$10      INITIALIZE DIGITS
0202   85 F9          STA 00F9
0204   85 FB          STA 00FB
0206   AD 44 17 RAND  LDA 1744      GET "RANDOM" #
0209   29 1F          AND #$1F      NOT TOO BIG
020B   09 01          ORA #$01      NOT TOO SMALL
020D   85 EE          STA 00EE      PUT IN DECREMENT LOC.
020F   A9 00          LDA #$00      BLANK CENTER DIGITS
0211   85 FA          STA 00FA
0213   20 71 02 DISP  JSR LITE      DISPLAY DIGITS
0216   AD 07 17       LDA 1707      TIME UP?
0219   F0 0D          BEQ MORE      NO
021B   A9 FF          LDA #$FF
021D   8D 07 17       STA 1707      START TIMER
0220   C6 EE          DEC 00EE      FULL TIME UP?
0222   10 04          BPL MORE      NO, SKIP
0224   A9 36          LDA #$36      YES, CHANGE ..
0226   85 FA          STA 00FA      CENTER DIGITS
0228   D8       MORE  CLD           CLEAR FOR KEYBOARD
0229   20 40 1F       JSR KEYIN     INIT. KEYBOARD
022C   20 6A 1F       JSR GET KEY   KEY DEPRESSED?
022F   C9 15          CMP #$15      VALID KEY?
0231   10 E0          BPL DISP      NO
0233   C9 07          CMP #$07      RIGHT KEY?
0235   F0 0E          BEQ RITE      YES
0237   C9 00          CMP #$00      LEFT KEY?
0239   F0 02          BEQ LEFT      YES
023B   D0 D6          BNE DISP      NOT A 0 OR A 7
023D   A2 02     LEFT  LDX #$02      INDEX FOR LEFT
023F   A5 EE          LDA 00EE      TIME UP?
0241   10 14          BPL LOS1      NO DECREASE LEFT ONE
0243   30 06          BMI ADD1      YES, INCREASE LEFT
0245   A2 00     RITE  LDX #$00      INDEX FOR RIGHT
0247   A5 EE          LDA 00EE      CHECK TIME
0249   10 0C          BPL LOS1      NOPE, NOT YET
```

```
024B  F8          ADD1    SED
024C  18                  CLC             INCREASE SCORE ..
024D  B5 F9               LDA 00F9,X      BY ONE
024F  69 01               ADC #$01
0251  95 F9               STA 00F9,X
0253  8A                  TXA             INDEX TO OTHER ..
0254  49 02               EOR #$02        SIDE
0256  AA                  TAX
0257  F8          LOS1    SED             DECREASE SCORE ..
0258  38                  SEC             BY ONE
0259  B5 F9               LDA 00F9,X
025B  E9 01               SBC #$01
025D  95 F9               STA 00F9,X
025F  F0 0A               BEQ FIN         GO TO FIN IF ZERO
0261  20 71 02    WAIT    JSR LITE        WAIT FOR SWITCH ..
0264  20 40 1F            JSR KEYIN       TO BE RELEASED
0267  D0 F8               BNE WAIT
0269  F0 9B               BEQ RAND        THEN START NEW DELAY
026B  20 71 02    FIN     JSR LITE        FINISHED LOOP
026E  B8                  CLV
026F  50 FA               BVC FIN         UNCOND. JUMP
                ****** DISPLAY SUBROUTINE ******
0271  A9 7F       LITE    LDA #$7F
0273  8D 41 17            STA SADD
0276  A2 09               LDX #$09        INIT. DIGIT #
0278  A5 FB               LDA 00FB
027A  20 8B 02            JSR 2HEX
027D  A5 FA               LDA 00FA        GET CENTER DIGITS
027F  20 4E 1F            JSR CONVX       CONVERT NONHEX CHAR.
0282  20 4E 1F            JSR CONVX       TWO OF THEM
0285  A5 F9               LDA 00F9
0287  20 8B 02            JSR 2HEX
028A  60                  RTS
      ****** HEX CHARACTER CONVERSION SUBROUTINE ******
028B  A8          2HEX    TAY
028C  4A                  LSR A           SUBROUTINE TO CONVERT
028D  4A                  LSR A           ONE WORD TO 2 HEX
028E  4A                  LSR A           CHARACTERS
028F  4A                  LSR A
0290  F0 0A               BEQ ZBLK
0292  20 48 1F            JSR CONVD
0295  98          2NDC    TYA             SECOND CHARACTER
0296  29 0F               AND #$0F
0298  20 48 1F            JSR CONVD
029B  60                  RTS
029C  A9 80       ZBLK    LDA #$80        BLANK LEADING ZEROS
029E  84 FC               STY 00FC
02A0  20 4E 1F            JSR CONVX       CONVERT NONHEX CHAR.
02A3  B8                  CLV
02A4  50 EF               BVC 2NDC        UNCOND. JUMP
```

FARMER BROWN

by Jim Butterfield

You are farmer Brown. You are growing a beautiful crop of corn But the following animals try to come and steal your corn:



Ant Bird Cow Dog Elephant Fox

As soon as you see one of these animals coming for your corn, you can scare it away by calling its name. Press the button with the first letter of the animal's name. So you would press A to shoo away an ant, B to shoo away a bird, and so on.

If you press the right button, the animal will go back. If you press the wrong button, it will think you mean somebody else and keep coming for your corn. And when all your corn is gone, KIM will show 000 and the game is over.

The animal won't "shoo" unless it has completely entered the display. Speed of the animals can be adjusted by changing the contents of location 026A.

```
0200 A2 0D    START  LDX #$13
0202 86 6E           STX CORN      bushels of corn to start
0204 A9 00           LDA #0        clear the window
0206 95 60    SLOOP  STA WINDOW,X
0208 CA              DEX
0209 10 FB           BPL SLOOP
020B A2 0B    TEST   LDX #11       is window empty?
020D B5 60    TLOOP  LDA WINDOW,X
020F D0 3B           BNE CONTIN       no, keep going
0211 CA              DEX
0212 10 F9           BPL TLOOP
0214 E6 6D           INC GOT       yes. make new animal
0216 A5 6C           LDA FLAG
0218 F0 09           BEQ MORE      did last animal get in?
021A C6 6D           DEC GOT
021C C6 6E           DEC CORN      take away some corn
021E D0 03           BNE MORE      any left?
0220 4C 25 19        JMP DONE      no, end of game
0223 AD 04 17 MORE   LDA TIMER     random value..
0226 4A 4A 4A        LSRA LSRA LSRA    ..to generate..
0229 4A 4A           LSRA LSRA          ..new random animal
022B C9 06           CMP #6        6 types of animal
022D 90 02           BCC MAKE
022F 29 03           AND #$03
0231 18       MAKE   CLC
0232 AA              TAX           animal type to X
0233 69 0A           ADC #$0A      key type A to F
```

```
0235 85 6F          STA KEY
0237 BD A4 02       LDA INDEX,X  animal 'picture' address
023A 85 70          STA POINL       to indirect pointer
023C A9 02          LDA #2
023E 85 71          STA POINH
0240 A0 05          LDY #5      six locations to move
0242 B1 70    ALOOP LDA (POINL),Y    from 'picture'
0244 99 66 00       STA WINGS,Y       ..to 'wings'
0247 88             DEY
0248 10 F8          BPL ALOOP
024A 84 6C          STY FLAG    flag FF = animal coming
024C A2 05    CONTIN LDX #5       test:
024E B5 66    CLOOP LDA WINGS,X    is animal out of 'wings'?
0250 D0 13          BNE NOKEY      no, ignore keyboard
0252 CA             DEX
0253 10 F9          BPL CLOOP
0255 20 40 1F       JSR KEYIN
0258 20 6A 1F       JSR GETKEY
025B C5 6F          CMP KEY          right anim al named?
025D D0 06          BNE NOKEY        no, ignore key
025F A5 6C          LDA FLAG
0261 10 02          BPL NOKEY*    animal retreating?
0263 E6 6C          INC FLAG      make animal retreat
0265 C6 72    NOKEY DEC DELAY     wait a while..
0267 D0 1E          BNE NOMOVE    before moving animal
0269 A9 20          LDA #$20      speed control value
026B 85 72          STA DELAY
026D A5 6C          LDA FLAG      move animal - which way?
026F 30 0D          BMI COMING     ..left
0271 A2 0A          LDX #10        ..right
0273 B5 5A    RLOOP LDA WINDOW-6,X
0275 95 5B          STA WINDOW-5,X
0277 CA             DEX
0278 D0 F9          BNE RLOOP
027A 86 5A          STX WINDOW-6    clear extreme left
027C F0 09          BEQ NOMOVE      unconditional branch
027E A2 F0    COMING LDX #$F0       -16
0280 B5 6C    CMLOOP LDA WINDOW+12,X
0282 95 6B          STA WINDOW+11,X
0284 E8             INX
0285 30 F9          BMI CMLOOP
0287 A9 7F    NOMOVE LDA #$7F       light KIM display
0289 8D 41 17       STA PADD
028C A0 13          LDY #$13
028E A2 05          LDX #5        six display digits
0290 B5 60    LITE  LDA WINDOW,X
0292 8D 40 17       STA SAD
0295 8C 42 17       STY SBD
0298 E6 73    LITEX INC WAIT
029A D0 FC          BNE LITEX
029C 88 88 CA       DEY DEY DEX
029F 10 EF          BPL LITE
02A1 4C 0B 02       JMP TEST
            ; index and animal 'pictures' in hexadecimal form
02A4  AA B0 B6 BC C2 C8  08 00 00 00 00 00  01 61 61 40 00 00
02B6  61 51 47 01 00 00  63 58 4E 00 00 00  71 1D 41 1F 01 00
02C8  63 58 4C 40 00 00
```

FARMER BROWN....

Exercises:

1. You can see that each animal occupies 6 memory locations, starting at 02AA (the Ant) - and the last location must always be zero. Can you make up your own animals? The letters may not fit exactly, but you can always invent names or use odd ones (you could make an Aardvark, a Burfle, a Cobra, and so on).

2. The game might be more fun if the animals went faster after a while, so that sooner or later they would just zip by. The location that controls speed is at address 026A; the lower the number, the faster the animals will go. So if you could arrange to have the program decrease this number automatically once in a while, you'd get a nice speed-up feature.

3. You can't "shoo" the animal until it's completely entered the display; but you can still catch it after it's partly left. The game would be harder - and maybe more fun - if you could only shoo it while it was completely in the display. Hint - testing location 005F (WINDOW-1) would tell you if an animal was on its way out.

4. You'd have a "Target Practice" game if you made the animal disappear (instead of backing up) when you pressed the right button. With a little planning, you'll find that this is quite easy to do.

****** HEX DUMP - FARMER BROWN ******

```
0200-   A2 0D 86 6E A9 00 95 60 CA 10 FB A2 0B B5 60 D0
0210-   3B CA 10 F9 E6 6D A5 6C F0 09 C6 6D C6 6E D0 03
0220-   4C 25 19 AD 04 17 4A 4A 4A 4A 4A C9 06 90 02 29
0230-   03 18 AA 69 0A 85 6F BD A4 02 85 70 A9 02 85 71
0240-   A0 05 B1 70 99 66 00 88 10 F8 84 6C A2 05 B5 66
0250-   D0 13 CA 10 F9 20 40 1F 20 6A 1F C5 6F D0 06 A5
0260-   6C 10 02 E6 6C C6 72 D0 1E A9 20 85 72 A5 6C 30
0270-   0D A2 0A B5 5A 95 5B CA D0 F9 86 5A F0 09 A2 F0
0280-   B5 6C 95 6B E8 30 F9 A9 7F 8D 41 17 A0 13 A2 05
0290-   B5 60 8D 40 17 8C 42 17 E6 73 D0 FC 88 88 CA 10
02A0-   EF 4C 0B 02 AA B0 B6 BC C2 C8 08 00 00 00 00 00
02B0-   01 61 61 40 00 00 61 51 47 01 00 00 63 58 4E 00
02C0-   00 00 71 1D 41 1F 01 00 63 58 4C 40 00 00
```

HI-LO

BY JIM BUTTERFIELD

DESCRIPTION -

AN EASY GAME FOR ONE OR MORE PLAYERS. KIM CHOOSES A SECRET NUMBER FROM 01 TO 98. AT THE START, THE FIRST FOUR DIGITS SHOW THE HIGH AND LOW BOUNDS OF THE NUMBER - 99 HIGH AND 00 LOW. AS GUESSES ARE ENTERED - ENTER THE GUESS AND PRESS A FOR ATTEMPT - THE BOUNDS CHANGE AS YOU ARE NARROWING DOWN THE POSSIBILITIES. FOR EXAMPLE, GUESS 32 AND THE DISPLAY MIGHT CHANGE TO 32 00, MEANING THAT THE COMPUTER'S SECRET NUMBER IS BETWEEN THESE VALUES. AFTER EACH LEGAL GUESS, THE COMPUTER SHOWS THE NUMBER OF ATTEMPTS MADE SO FAR.

ONE PLAYER GAME: TRY TO GET THE MYSTERY NUMBER IN SIX ATTEMPTS.

MULTI PLAYER GAME: EACH PLAYER TRIES TO AVOID GUESSING THE MYSTERY NUMBER - THE CORRECT GUESSER LOSES AND IS "OUT".

```
0200 F8          START  SED
0201 A5 E0       TOP    LDA RND  generate random #
0203 38                 SEC      01 to 98
0204 69 00              ADC #0
0206 A2 01              LDX #1   overflow at 99
0208 C9 99              CMP #$99
020A D0 01              BNE OVR0
020C 8A                 TXA
020D 85 E0       OVR0   STA RND
020F 20 40 1F           JSR KEYIN
0212 D0 ED              BNE TOP
0214 D8                 CLD          initialize:
0215 A9 99              LDA #$99           hi
0217 85 FB              STA POINTH
0219 A9 00              LDA #0
021B 85 FA              STA POINTL      and lo
021D A2 A0       RSET   LDX #$A0 guess counter
021F 86 F9       NSET   STX INH
0221 86 E1              STX NGUESS
0223 20 1F 1F    GUESS  JSR SCANDS light display
0226 20 6A 1F           JSR GETKEY test key
0229 C9 13              CMP #$13   go key?
022B F0 D3              BEQ START
022D C5 E2              CMP LAST
022F F0 F2              BEQ GUESS  same key?
0231 85 E2              STA LAST
```

```
0233 C9 0A            CMP #$0A     'A' key?
0235 F0 10            BEQ EVAL     yes, evaluate guess
0237 B0 EA            BCS GUESS    no key?
0239 0A               ASL A        roll character
023A 0A               ASL A        ..into..
023B 0A               ASL A        position..
023C 0A               ASL A
023D A2 03            LDX #3
023F 0A         LOOP  ASL A        ..then
0240 26 F9            ROL INH      ..into
0242 CA               DEX          ..display
0243 10 FA            BPL LOOP
0245 30 DC            BMI GUESS
0247 A5 F9      EVAL  LDA INH      guess lower..
0249 C5 E0            CMP RND      ..than number?
024B 90 06            BCC OVR1     yes, skip
024D C5 FB            CMP POINTH   no, check hi
024F B0 D2            BCS GUESS    out of range?
0251 85 FB            STA POINTH
0253 A6 E0      OVR1  LDX RND      number lower..
0255 E4 F9            CPX INH      ..than guess?
0257 90 08            BCC OVR2     yes, skip
0259 A6 FA            LDX POINTL   no,check lo
025B E4 F9            CPX INH
025D B0 C4            BCS GUESS    out of range?
025F 85 FA            STA POINTL
0261 A6 E1      OVR2  LDX NGUESS   'guess' number
0263 E8               INX             ..plus 1
0264 E0 AA            CPX #$AA     past limit?
0266 F0 B5            BEQ RSET     yes, reset
0268 D0 B5            BNE NSET
```

xxxxxx HEX DUMP - HI LO xxxxxx

```
0200 F8 A5 E0 38 69 00 A2 01 C9 99 D0 01 8A 85 E0 20
0210 40 1F D0 ED D8 A9 99 85 FB A9 00 85 FA A2 A0 86
0220 F9 86 E1 20 1F 1F 20 6A 1F C9 13 F0 D3 C5 E2 F0
0230 F2 85 E2 C9 0A F0 10 B0 EA 0A 0A 0A 0A A2 03 0A
0240 26 F9 CA 10 FA 30 DC A5 F9 C5 E0 90 06 C5 FB B0
0250 D2 85 FB A6 E0 E4 F9 90 08 A6 FA E4 F9 B0 C4 85
0260 FA A6 E1 E8 E0 AA F0 B5 D0 B5
```

HORSERACE

BY CHUCK EATON

DESCRIPTION -

THIS IS AN EIGHT LAP HORSE RACE AND YOU CAN BE THE JOCKEY AND WHIP YOUR HORSE TO GO FASTER. WARNING ... WHIP THE HORSE TOO MUCH AND HE PROBABLY POOPS OUT. THE PROGRAM STARTS AT 0200.

HORSE	TRACK	WHIPPING BUTTON
PRINCE CHARMING	TOP	PC
COLORADO COWBOY	MIDDLE	C
IRISH RAIR	BOTTOM	4

```
0200  D8              CLD          ...INITIALIZATION...
0201  A2 13           LDX #$13
0203  BD D9 02  INIT  LDA 02D9,X  HORSES TO STARTING GATE
0206  95 7C           STA 007C,X
0208  CA              DEX
0209  10 F8           BPL INIT
020B  A9 7F      DISP LDA #$7F     ...LIGHT DISPLAY...
020D  8D 41 17        STA 1741
0210  A0 00           LDY #$00
0212  A2 09           LDX #$09
0214  B9 7C 00  LITE  LDA 007C,Y
0217  84 FC           STY 00FC
0219  20 4E 1F        JSR 1F4E    OUTPUT DIGIT
021C  C8              INY
021D  C0 06           CPY #$06    SIX DIGITS DISPLAYED?
021F  90 F3           BCC LITE    NOT YET
0221  20 3D 1F        JSR 1F3D    TURN OFF DIGITS
0224  A5 8F           LDA LAP CNT.FINISHED TOTAL LAPS?
0226  30 E3           BMI DISP    YES, FREEZE DISPLAY
0228  A2 03           LDX #$03
022A  CA        NEXT  DEX         NEXT HORSE
022B  30 DE           BMI DISP    FINISHED 3 HORSES
022D  D6 86           DEC 0086,X  DEC. CNT., HORSE X
022F  D0 F9           BNE NEXT    NOT ZERO, NEXT HORSE
0231  86 99           STX 0099    SAVE HORE INDEX
0233  A4 99           LDY 0099    AND PUT IN Y AS INDEX
0235  B6 83           LDX 0083,Y  DIGIT POS. OF HORSE IN X
0237  B9 ED 02        LDA 02ED,Y  MASK TO REMOVE HORSE
023A  35 7C           AND 007C,X  GET RID OF HORSE
023C  95 7C           STA 007C,X  RETURN REMAINING HORSES
023E  E8              INX         GO TO NEXT DIGIT RIGHT
023F  96 83           STX 0083,Y  UPDATE HORSE DIGIT POS.
0241  B9 ED 02        LDA 02ED,Y  GET MASK
0244  49 FF           EOR #$FF    CHANGE TO AN INSERT MASK
0246  15 7C           ORA 007C,X  PUT HORSE IN NEXT
0248  95 7C           STA 007C,X  DIGIT RIGHT
024A  E0 05           CPX #$05    REACHED RIGHT SIDE?
024C  30 2B            BMI POOP    NOT YET
024E  D0 06           BNE NLAP    OFF RIGHT SIDE, CHANGE LAP
0250  A5 8F           LDA 008F    CHECK LAP COUNTER
0252  F0 1B           BEQ LAST    IF ZERO, LAST LAP
0254  D0 23           BNE POOP
```

```
0256  A2 02      NLAP    LDX #$02     ...CHANGE TO A NEW LAP
0258  38         DOWN    SEC          SHIFT ALL HORSE DIGIT
0259  B5 83              LDA 0083,X   POSITIONS SIX PLACES
025B  E9 06              SBC #$06     DOWN...
025D  95 83              STA 0083,X
025F  CA                 DEX
0260  10 F6              BPL DOWN
0262  A2 06              LDX #$06
0264  B5 7C      STOR    LDA 007C,X   ...ALSO SHIFT DIGIT
0266  95 76              STA 0076,X   CONTENTS INTO STORAGE
0268  A9 80              LDA #$80     AREA AND CLEAR DISPLAY
026A  95 7C              STA 007C,X   AREA...
026C  CA                 DEX
026D  D0 F5              BNE STOR
026F  C6 8F      LAST    DEC 008F     DEC. LAP COUNTER
0271  D0 06              BNE POOP     NOT LAST LAP, CONTINUE
0273  A5 81              LDA 0081     LAST LAP, PUT FINISH
0275  09 06              ORA #$06     LINE IN LAST DIGIT
0277  85 81              STA 0081
0279  B9 89 00   POOP    LDA 0089,Y   HORSE Y POOP FLAG
027C  F0 0A              BEQ NOPO     HORSE NOT POOPED
027E  20 C5 02           JSR RAND     ...POOPED, BUT MAY
0281  29 3C              AND #$3C     BECOME UNPOOPED DEPENDING
0283  D0 1A              BNE FAST     ON RANDOM NUMBER
0285  99 89 00           STA 0089,Y
0288  20 C5 02   NOPO    JSR RAND     ...NOT POOPED, BUT MAY
028B  29 38              AND #$38     BECOME POOPED DEPENDING
028D  85 9A              STA 009A     ON RANDOM NUMBER...
028F  B9 8C 00           LDA 008C,Y
0292  30 0B              BMI FAST
0294  29 38              AND #$38
0296  C5 9A              CMP 009A
0298  B0 05              BCS FAST
029A  A9 FF              LDA #$FF     IF POOPED, SET POOP
029C  99 89 00           STA 0089,Y   FLAG TO "FF"
029F  20 3D 1F   FAST    JSR KEYIN        GET KEY FROM KEYBOARD
02A2  A0 FF              LDY #$FF       INIT. Y TO MAX
02A4  A6 99              LDX 0099       HORSE INDEX IN X
02A6  3D F0 02           AND 02F0,X     MASK (IS HORSE WHIPPED?)
02A9  F0 01              BEQ SKIP       NO, NOT BEING WHIPPED
02AB  88                 DEY            WHIPPED, Y MADE SMALLER
02AC  98         SKIP    TYA            ..CHANGE SIGN IF POOPED
02AD  55 89              EOR 0089,X     EXC. OR WITH 00 OR FF
02AF  85 9A              STA 009A       SAVE SPEED UPDATE
02B1  20 C5 02           JSR RAND       GET A RANDOM NUMBER
02B4  38                 SEC
02B5  29 01              AND #$01       ..LOWEST BIT OF #
02B7  65 9A              ADC 009A       COMBINE WHIP UPDATE,
02B9  18                 CLC            RAND # (0 OR 1) & CARRY
02BA  A6 99              LDX 0099       HORSE INDEX IN X
02BC  75 8C              ADC 008C,X     HORSES SPEED ADDED IN
02BE  95 8C              STA 008C,X     SAVE NEW SPEED
02C0  95 86              STA 0086,X     ALSO IN WINDOW COUNTER
02C2  4C 2A 02           JMP NEXT       LOOP
```

```
          ***** RANDOM NUMBER SUBROUTINE *****
02C5  38        RAND    SEC
02C6  A5 92             LDA 0092        FROM J. BUTTERFIELD
02C8  65 95             ADC 0095        KIM USER NOTES * 1
02CA  65 96             ADC 0096        PAGE 4
02CC  85 91             STA 0091
02CE  A2 04             LDX #$04
02D0  B5 91     MOVE    LDA 0091,X
02D2  95 92             STA 0092,X
02D4  CA                DEX
02D5  10 F9             BPL MOVE
02D7  60                RTS
```

****** TABLES - HORSERACE ******

```
02D8-  00/80/80/80/80/80/80/80
02E0-  FF/FF/FF/80/80/80/00/00/00/80/80/80/08/FE/BF/F7
02F0-  01/02/04
```

****** HEX DUMP - HORSERACE ******

```
0200 D8 A2 13 BD D9 02 95 7C CA 10 F8 A9 7F 8D 41 17
0210 A0 00 A2 09 B9 7C 00 84 FC 20 4E 1F C8 C0 06 90
0220 F3 20 3D 1F A5 8F 30 E3 A2 03 CA 30 DE D6 86 D0
0230 F9 86 99 A4 99 B6 83 B9 ED 02 35 7C 95 7C E8 96
0240 83 B9 ED 02 49 FF 15 7C 95 7C E0 05 30 2B D0 06
0250 A5 8F F0 1B D0 23 A2 02 38 B5 83 E9 06 95 83 CA
0260 10 F6 A2 06 B5 7C 95 76 A9 80 95 7C CA D0 F5 C6
0270 8F D0 06 A5 81 09 06 85 81 B9 89 00 F0 0A 20 C5
0280 02 29 3C D0 1A 99 89 00 20 C5 02 29 38 85 9A B9
0290 8C 00 30 0B 29 38 C5 9A B0 05 A9 FF 99 89 00 20
02A0 3D 1F A0 FF A6 99 3D F0 02 F0 01 88 98 55 89 85
02B0 9A 20 C5 02 38 29 01 65 9A 18 A6 99 75 8C 95 8C
02C0 95 86 4C 2A 02 38 A5 92 65 95 65 96 85 91 A2 04
02D0 B5 91 95 92 CA 10 F9 60 00 80 80 80 80 80 80 80
02E0 FF FF FF 80 80 80 00 00 00 80 80 80 08 FE BF F7
02F0 01 02 04
```

KEY TRAIN

By Jim Butterfield

Ever wish you could touch-type your KIM keypad like some people can type? It's not hard; all you need is practice. And what better teacher to drill you on key entry than the KIM system itself?

Load this fully relocatable program anywhere. Start it up, and the display will show a random hexadecimal digit, from 0 to F. Hit the corresponding key, and the display will blank, and then present you with another random digit. Hit the wrong key, and nothing will happen.

The educational principle involved is called positive reinforcement. That is, you're rewarded for doing the right thing, and ignored if you do it wrong. A few minutes of practice a day, and you'll become a speed demon on the keyboard!

```
0000 20 40 1F START  JSR KEYIN
0003 D0 FB           BNE START   key still depressed - blank
0005 AD 04 17        LDA TIMER   random value
0008 4A 4A           LSRA LSRA   wipe high order bits
000A 4A 4A           LSRA LSRA
000C 85 FF           STA TEMP    save the digit
000E 0A 0A           ASLA ASLA   move back left
0010 0A 0A           ASLA ASLA
0012 05 FF           ORA TEMP    repeat the digit
0014 85 F9           STA INH     put..
0016 85 FA           STA POINTL    ..into..
0018 85 FB           STA POINTH          ..display
001A 20 1F 1F LIGHT  JSR SCANDS  light display
001D 20 6A 1F        JSR GETKEY  test keys
0020 C5 FF           CMP TEMP    right key?
0022 F0 DC           BEQ START   yes, blank & rpeat
0024 D0 F4           BNE LIGHT
```

The random number used in this program is taken from the KIM timer. This timer runs continuously and might be anywhere between 00 and FF at the instant we push the button. We use the four left hand (high order) bits of the timer to produce the next digit.

Be sure that KIM is not in decimal mode when you run this program - set address 00F1 to 00 before starting. If you forget, you might find that the alphabetic keys (A to F) don't work right.

Exercises: can you make the program clear decimal mode automatically? How about a counter to record the number of correct keystrokes you have made? That way, you could time yourself to see how many keys you can get right in 60 seconds. The count could be shown in the two right hand digits of the display. Do you think it should be in decimal or hexadecimal?

KIM NIM

BY JIM BUTTERFIELD

Here's a jumbo NIM that's good for all skill levels. Why? Because KIM matches wits with you - literally. Play a duffer's game and KIM will make lots of errors, too. Start winning a few - and KIM will move up to the master player level.

Hit GO and several digits on the KIM display will light. Each lit digit represents a pile of objects you can pick from. Decide which pile you want, and enter its identity: A for the left-hand pile through to F for the right-hand pile. The pile you have selected will start to flash on and off. Now enter the number of items you want to take from that pile.

KIM will take its turn the same way - you'll see the pile selected begin to flash, and then some items will be taken away. After the computer moves, it's your turn again.

The winner is the player who takes the last object. When this happens, KIM will identify the winner. A new game can be started at any time by hitting GO.

```
0200 20 40 1F START  JSR KEYIN   directional regs
0203 20 6A 1F        JSR GETKEY
0206 C9 13           CMP #$13    GO key?
0208 D0 3A           BNE NOGO    nope, skip
020A AD 04 17        LDA TIMER   get random nbr
020D A2 02           LDX #2      split into 3
020F A8       SPLIT  TAY         save A
0210 29 07           AND #7      extract 3 bits
0212 F0 03           BEQ ZINCH   unless zero..
0214 18              CLC         ..add two
0215 69 02           ADC #2
0217 95 04    ZINCH  STA VALUE,X  store pile val
0219 98              TYA           bring back rand
021A 4A 4A 4A        LSRA LSRA LSRA
021D CA              DEX
021E 10 EF           BPL SPLIT
0220 20 40 1F STALL  JSR KEYIN   wait for..
0223 D0 FB           BNE STALL   ..key release
0225 AD 04 17        LDA TIMER   new random nbr
0228 A2 02           LDX #2      split 3 ways
022A A8       SPLAT  TAY               again
022B 29 07           AND #7      3 bits
022D 95 07           STA VALUE+3,X
022F 98              TYA
0230 4A 4A 4A        LSRA LSRA LSRA
0233 CA              DEX
0234 10 F4           BPL SPLAT
0236 85 01           STA PILE    pile zero
0238 85 02           STA MOVE    it's your move
023A A2 06           LDX #6      for each pile..
```

```
023C B5 03     DRESS  LDA VALUE-1,X   ..change to
023E 20 2D 03         JSR SEG        ..segments
0241 CA               DEX
0242 D0 F8            BNE DRESS
0244 A6 02     NOGO   LDX MOVE       whose move?
0246 D0 3D            BNE NOKEY      computer's, skip
0248 C9 10            CMP #$10       hex digit keyed?
024A B0 39            BCS NOKEY      no, skip
024C C9 00            CMP #0         zero key?
024E F0 35            BEQ NOKEY      yes, skip
0250 C9 0A            CMP #$0A       alphabetic?
0252 90 12            BCC NUM        no, numeric
0254 38               SEC            change A-F...
0255 E9 09            SBC #9         ..to 1-6
0257 A6 01            LDX PILE       pile already..
0259 D0 2A            BNE NOKEY      ..selected?
025B AA               TAX
025C B5 0A            LDA FLASHR,X
025E F0 25            BEQ NOKEY      nothing in pile?
0260 86 01            STX PILE       OK, mark pile
0262 85 0A            STA FLASHR     store flash code
0264 B0 1F            BCS NOKEY      unconditional
0266 A6 01     NUM    LDX PILE
0268 F0 1B            BEQ NOKEY      no pile selected
026A 85 03            STA TEMP       save number
026C B5 03            LDA VALUE-1,X  pile value
026E C5 03            CMP TEMP       pile big enough?
0270 90 13            BCC NOKEY      nope
0272 E5 03            SBC TEMP       yes, take out
0274 20 2D 03         JSR SEG        compute segments
0277 E6 02            INC MOVE       computer's move
0279 20 16 03         JSR SURVEY     end of game?
027C D0 07            BNE NOKEY      no, keep going
027E 20 05 03         JSR MESSAG     yes, show messg
0281 85 0B            STA WINDOW     ''I LOSE''
0283 46 00            LSR IQ         get smart!
                ; all routines join here - display
0285 A6 01     NOKEY  LDX PILE
0287 A5 0A            LDA FLASHR     flash pile
0289 55 0A            EOR FLASHR,X
028B 95 0A            STA FLASHR,X
028D A9 7F            LDA #$7F
028F 8D 41 17         STA PADD
0292 A0 13     LIGHT  LDY #13
0294 A2 05            LDX #5
0296 B5 0B     LITE   LDA WINDOW,X
0298 8D 40 17         STA SAD
029B 8C 42 17         STY SBD
029E E6 11     LITEX  INC CUE
02A0 D0 FC            BNE LITEX
02A2 88 88            DEY DEY
02A4 CA               DEX
02A5 10 EF            BPL LITE
02A7 E6 12            INC WAIT
02A9 D0 E7            BNE LIGHT
```

```
02AB A9 F8          LDA #$F8
02AD 85 12          STA WAIT
02AF A6 02          LDX MOVE    whose move?
02B1 F0 4E          BEQ EXIT    not computer's
02B3 CA             DEX         first step?
02B4 D0 2B          BNE TRY     no, skip stratgy
02B6 A9 00          LDA #0
02B8 A2 05          LDX #5      merge all piles..
02BA 55 04    MERGE EOR VALUE,X  ..by EOR-ing them
02BC CA             DEX
02BD 10 FB          BPL MERGE
02BF 85 0A          STA FLASHR   save EOR product
02C1 A2 06          LDX #6      re-examine piles
02C3 B5 03    LOOP  LDA VALUE-1,X
02C5 45 0A          EOR FLASHR
02C7 D5 03          CMP VALUE-1,X
02C9 90 05          BCC FOUND
02CB CA             DEX
02CC D0 F5          BNE LOOP
02CE F0 0B          BEQ MOVE
02D0 A4 00    FOUND LDY IQ   IQ high enuff?
02D2 CC 04 17       CPY TIMER  ..randomly..
02D5 B0 04          BCS MOVE   no, move dumb
02D7 85 03          STA TEMP   amount
02D9 86 01          STX PILE   pile number
02DB A6 01    MOVE  LDX PILE
02DD B5 0A          LDA FLASHR,X  flash mask
02DF 85 0A          STA FLASHR    Flash...
02E1 E6 02    TRY   INC MOVE   but don*t make
02E3 A5 02          LDA MOVE   ..the move till..
02E5 C9 10          CMP #$10   ..time has passed
02E7 90 18          BCC EXIT
02E9 A6 01          LDX PILE   time to move!
02EB A5 03          LDA TEMP
02ED 20 2D 03       JSR SEG    make move
02F0 20 16 03       JSR SURVEY  end of game?
02F3 D0 06          BNE KEEP   nope, keep goin
02F5 20 05 03       JSR MESSAG  'U LOSE'
02F8 38             SEC        dummy up..
02F9 26 00          ROL IQ     ..the computer
02FB A9 00    KEEP  LDA #0
02FD 85 02          STA MOVE   it's your move
02FF 85 01          STA PILE   un-flash
0301 D8             CLD
0302 4C 00 02       JMP START

0305 A9 00    MESSAG LDA #0
0307 85 02          STA MOVE   end of play
0309 85 01          STA PILE   no flashing
030B A2 06          LDX #6     move 7 digits
030D BD 3B 03 MLOOP LDA DATA,X   pick em up..
0310 95 0A          STA FLASHR,X ..put em down
0312 CA             DEX
0313 10 F8          BPL MLOOP
0315 60             RTS
```

```
0316 A9 00     SURVEY LDA #0
0318 85 0A            STA FLASHR un-flash
031A A2 06            LDX #6      for all piles..
031C D5 03     REVUE  CMP VALUE-1,X
031E B0 06            BCS SMALL
0320 B5 03            LDA VALUE-1,X
0322 85 03            STA TEMP
0324 86 01            STX PILE
0326 CA        SMALL  DEX
0327 D0 F3            BNE REVUE
0329 C6 03            DEC TEMP
032B A8               TAY          test A
032C 60               RTS

032D 95 03     SEG    STA VALUE-1,X   store value
032F F0 04            BEQ NIL         blank digit
0331 A8               TAY
0332 B9 E7 1F         LDA TABLE,Y
0335 95 0A     NIL    STA FLASHR,X  segments to wndw
0337 A9 00            LDA #0
0339 60               RTS

033A FF 06 BE 00 B8 BF ED F9    (DATA)
```

0342

****** HEX DUMP - KIM NIM ******

```
0200 20 40 1F 20 6A 1F C9 13 D0 3A AD 04 17 A2 02 A8
0210 29 07 F0 03 18 69 02 95 04 98 4A 4A 4A CA 10 EF
0220 20 40 1F D0 FB AD 04 17 A2 02 A8 29 07 95 07 98
0230 4A 4A 4A CA 10 F4 85 01 85 02 A2 06 B5 03 20 2D
0240 03 CA D0 F8 A6 02 D0 3D C9 10 B0 39 C9 00 F0 35
0250 C9 0A 90 12 38 E9 09 A6 01 D0 2A AA B5 0A F0 25
0260 86 01 85 0A B0 1F A6 01 F0 1B 85 03 B5 03 C5 03
0270 90 13 E5 03 20 2D 03 E6 02 20 16 03 D0 07 20 05
0280 03 85 0B 46 00 A6 01 A5 0A 55 0A 95 0A A9 7F 8D
0290 41 17 A0 13 A2 05 B5 0B 8D 40 17 8C 42 17 E6 11
02A0 D0 FC 88 88 CA 10 EF E6 12 D0 E7 A9 F8 85 12 A6
02B0 02 F0 4E CA D0 2B A9 00 A2 05 55 04 CA 10 FB 85
02C0 0A A2 06 B5 03 45 0A D5 03 90 05 CA D0 F5 F0 0B
02D0 A4 00 CC 04 17 B0 04 85 03 86 01 A6 01 B5 0A 85
02E0 0A E6 02 A5 02 C9 10 90 18 A6 01 A5 03 20 2D 03
02F0 20 16 03 D0 06 20 05 03 38 26 00 A9 00 85 02 85
0300 01 D8 4C 00 02 A9 00 85 02 85 01 A2 06 BD 3B 03
0310 95 0A CA 10 F8 60 A9 00 85 0A A2 06 D5 03 B0 06
0320 B5 03 85 03 86 01 CA D0 F3 C6 03 A8 60 95 03 F0
0330 04 A8 B9 E7 1F 95 0A A9 00 60 FF 06 BE 00 B8 BF
0340 ED F9
```

KIM-TAC-TOE

BY LEW EDWARDS

DIRECTIONS -

PLAY BEGINS WITH KIM MAKING THE FIRST PLAY WHEN "GO" IS PRESSED. THE SECOND THROUGH FOURTH DIGITS OF THE DISPLAY HOLD THE PATTERN WITH SQUARES NUMBERED AS:

```
7 8 9
4 5 6
1 2 3
```

YOUR ENTRY WILL BE IMMEDIATE BUT KIM'S ACTION WILL BE DELAYED. YOUR PLAYS LIGHT STEADILY WHILE KIM'S FLICKER. A WINNING ROW BLINKS AND A DRAW BLINKS EVERYTHING. ON COMPLETION OF A GAME, THE "GO" KEY WILL START A NEW GAME. IF YOU PREFER TO PLAY FIRST, PRESS THE "+" KEY INSTEAD. THE KIM HAS AN I.Q. LEVEL THAT CAN BE CHANGED BY PRESSING "PC" AT GAMES END. YOU WILL SEE "ODDS" AND KIM'S I.Q. DISPLAYED. THE I.Q. IS INITIALLY SET TO 75%, (0C). CHANGE IT TO WHAT YOU WISH AND THEN PRESS "DA" TO RETURN TO THE DONE LOOP AND START A NEW GAME IN THE NORMAL MANNER. THE I.Q. IS ADJUSTED UPWARD EACH TIME THE PLAYER WINS AND DOWNWARD EACH TIME KIM WINS. THE PROGRAM STARTS AT 0100.

```
0100  4C 10 03            JMP STIQ        JUMP TO START LOCATION
0103  EA EA EA            NOP'S
        ****** SUBROUTINE "LOAD BLINK" ******
0106  A9 20               LDA #$20        BLINK FLAG
0108  15 BF               ORA SQST,X      ADD IT TO THE..
010A  95 BF               STA SQST,X      INDEXED BYTE
010C  60                  RTS
010D  EA EA               NOP'S
          ****** TABLE - SEGMENTS ZZ****
010F  08/08/08/40/40/40/01/01/01
              ******* TABLE - ROWS *******
0118  01/04/07/01/02/03/01/03
0120  02/05/08/04/05/06/05/05
0128  03/06/09/07/08/09/09/07
            **** SUBROUTINE "GET PLAY" ****
0130  85 D9     GPLA    STA TEMP        SAVE THE ACCUMULATOR
0132  A2 09             LDX #$09        FOR TESTING
0134  A5 D9     GPLP    LDA TEMP        GET IT BACK
0136  35 DB             AND PS,X        MASK THE STATUS BYTE
0138  24 D9             BIT TEMP        CHECK FOR BIT ON
013A  D0 03             BNE OUT         GOT IT - DONE
013C  CA                DEX
013D  D0 F5             BNE GPLP        NOPE - KEEP TRYING
013F  60        OUT     RTS             SQUARE VALUE IN X
        0 = NO MATCH
        ****** SUBROUTINE "TEST AND INCREMENT" ******
0140  B5 BF             LDA SS,X
0142  D0 02             BNE OUT         COUNT OPEN SQUARES
0144  F6 DB             INC PS,X        ONLY
0146  60        OUT     RTS
```

```
                        ****** SUBROUTINE "UPDATE" ******
0147  95 BF      UPDA     STA SS,X        FLAG THE SQUARE
0149  A0 08               LDY #$08
014B  A9 00      UPLP     LDA #$00        CLEAR THE REGISTER
014D  99 C8 00            STA RS,Y
0150  BE 17 01            LDX SQ1,Y       THEN LOAD
0153  20 8A 03            JSR RSADD       CURRENT STATUS
0156  BE 1F 01            LDX SQ2,Y       VALUES
0159  20 8A 03            JSR RSADD
015C  BE 27 01            LDX SQ3,Y
015F  20 8A 03            JSR RSADD
0162  88                  DEY
0163  D0 E6               BNE UPLP        LOOP TILL DONE
0165  60                  RTS

0200   A9 00     NEW      LDA #$00
0202   A2 1D               LDX #$1D        CLEAR REGISTERS
0204   95 B4     INLP     STA 00B4,X
0206   CA                 DEX
0207   D0 FB               BNE INLP
0209   A9 05               LDA #$05        INITALIZE ORDER OF..
020B   85 BB               STA 00BB        NON-CALCULATED PLAYS
020D   A0 04               LDY #$04        CENTER - FIXED ORDER
020F   20 F2 03  ELP1      JSR RPLA
0212   A2 04               LDX #$04
0214   D5 BB     ELP2     CMP REVN,X
0216   F0 F7               BEQ ELP1
0218   CA                 DEX
0219   D0 F9               BNE ELP2
021B   99 BB 00            STA REVN,Y      SIDES IN RANDOM ORDER
021E   88                 DEY
021F   D0 EE               BNE ELP1
0221   E6 B6               INC ODEV
0223   A0 04               LDY #$04
0225   20 F2 03  OLP1      JSR RPLA
0228   A2 05               LDX #$05
022A   D5 B6     OLP2     CMP RODD,X
022C   F0 F7               BEQ OLP1
022E   CA                 DEX
022F   D0 F9               BNE OLP2
0231   99 B6 00            STA RODD,Y      CORNERS-IN RANDOM ORDER
0234   88                 DEY
0235   D0 EE               BNE OLP1
0237   A9 03     PVAL     LDA #$03
0239   A0 08     TEST     LDY #$08        TEST FOR 3 IN A ROW
023B   D9 C8 00  WNLP     CMP ROWS,Y      03=PLAYER WIN/0C=KIM WIN
023E   F0 05               BEQ WIN         GAME WON-BLINK THE ROW
0240   88                 DEY
0241   D0 F8               BNE WNLP        NOT YET-CK NEXT ROW
0243   F0 15               BEQ DRAW        NO WINNER-CK FOR DRAW
0245  BE 17 01   WIN      LDX SQ1,Y
0248  20 06 01            JSR BLNK        BLINK #1
024B  BE 1F 01            LDX SQ2,Y
024E  20 06 01            JSR BLNK        BLINK #2
```

```
0251  BE 27 01          LDX SQ3,Y
0254  20 06 01          JSR BLNK        BLINK #3
0257  4C FE 02          JMP MTST        CHECK THE WINNER
025A  A2 09       DRAW  LDX #$09
025C  A9 C0       OPEN  LDA #$C0        OPEN SQUARE?
025E  35 BF             AND DSPL,X
0260   F0 0E              BEQ TURN       YES - CONTINUE GAME
0262   CA                 DEX            NO - CK NEXT SQUARE
0263   D0 F7              BNE OPEN       ALL DONE?
0265   A2 09              LDX #$09
0267  20 06 01  NXBL      JSR BLNK       NO OPEN SQUARES
026A  CA                 DEX            IT'S A DRAW
026B  D0 FA              BNE NXBL       BLINK 'EM ALL
026D  4C 15 03           JMP DONE       GAME'S OVER
0270  E6 B5       TURN    INC PLA4       COUNT THE PLAYS
0272  A5 DB              LDA MODE       WHO'S TURN?
0274  D0 17              BNE WAIT       KIM'S
0276  20 A6 03  KEY       JSR KEYS       PLAYER'S
0279  F0 FB              BEQ KEY        GET A KEY
027B  C9 0A              CMP #$0A       OVER 9?
027D  B0 F7              BCS KEY        GET ANOTHER
027F  AA                 TAX            USE IT AS AN INDEX
0280  B4 BF              LDY DSPL,X     SEE IF SQUARE'S OPEN
0282  D0 F2              BNE KEY        NO, TRY AGAIN
0284  A9 40              LDA #$40       YES, MARK IT FOR..
0286  20 47 01           JSR UPDATE     PLAYER
0289  E6 DB              INC MODE       KIM'S NEXT
028B  D0 AA              BNE PVAL       BUT FIRST CK FOR WIN
028D  20 4C 03  WAIT      JSR DISPLAY    HOLD KIM BACK
0290  E6 D1              INC LPCNT      A LITTLE
0292  D0 F9              BNE WAIT       UPDATE AND..
0294  A9 08              LDA #$08       THEN CHECK THE..
0296  20 C8 03           JSR PSLD       BOARD
0299  A9 02              LDA #$02
029B  20 C8 03           JSR PSLD
029E  A9 04              LDA #$04
02A0  20 C8 03           JSR PSLD
02A3  A9 01              LDA #$01
02A5  20 C8 03           JSR PSLD
02A8  A9 C0              LDA #$C0       WINNING PLAY FOR KIM
02AA  20 30 01           JSR GETPLA
02AD  D0 43              BNE PLAY       YES - MAKE IT
02AF  A9 30              LDA #$30       2 IN A ROW FOR..
02B1  20 30 01           JSR GETPLA     PLAYER
02B4  D0 3C              BNE PLAY       YES - BLOCK IT
02B6  A9 08              LDA #$08       POSSIBLE SQUEEZE
02B8  20 30 01           JSR GETPLA     PLAY FOR KIM
02BB  D0 35              BNE PLAY       YES - DO IT
02BD  20 B3 03  IPLA      JSR RAND       HOW MUCH SMARTS?
02C0  29 0F              AND #$0F       NEEDED?
02C2  C5 D2              CMP IQ         KIM'S I.Q.
02C4  B0 1F              BCS DUMB       TOO LOW - BAD MOVES
02C6  A4 B5              LDY PLAC       SMART
```

```
02C8  C0 01        CPY #$01      1ST PLAY?
02CA  D0 04        BNE FOUR      NO
02CC  29 01        AND #$01      YES
02CE  D0 17        BNE TPLA      1/2 TIME PLAY A CORNER
02D0  C0 04  FOUR  CPY #$04      4TH PLAY?
02D2  D0 06        BNE SPLA      NO, SKIP
02D4  24 C4        BIT SQST+5    YES, CK WHO HAS CENTER
02D6  30 0D        BMI DUMB      KIM - PLAY A SIDE
02D8  70 07        BVS PLAC      PLAYER-PLAY A CORNER
02DA  A9 02  SPLA  LDA #$02      CAN PLAYER MAKE A..
02DC  20 30 01     JSR GETPLA    SQUEEZE PLAY?
02DF  D0 11        BNE PLAY      YES - BLOCK IT
02E1  A0 05  PLAC  LDY #$05
02E3  D0 02        BNE TPLA      START WITH THE CENTER
02E5  A0 09  DUMB  LDY #$09      START WITH THE SIDES
02E7  B6 B6  TPLA  LDX RPLA,Y    USE THE RANDOM PLAY
02E9  B5 BF        LDA DISP,X    TABLE - OPEN SQUARE?
02EB  F0 05        BEQ PLAY      FOUND ONE - PLAY IT
02ED  88           DEY           NO, TRY NEXT ONE
02EE  D0 F7        BNE RPLA      NOT YET
02F0  F0 F3        BEQ DUMB      START OVER
02F2  A9 80  PLAY  LDA #$80      MARK THE..
02F4  20 47 01     JSR UPDATE    SQUARE FOR KIM
02F7  C6 DB        DEC MODE      PLAYER'S TURN NEXT
02F9  A9 0C        LDA #$0C      FIRST, DID KIM WIN?
02FB  4C 39 02     JMP TEST
02FE  A5 DB  MTST  LDA MODE      WHO WON?
0300  D0 04        BNE IQUP      PLAYER, UP KIM'S I.Q.
0302  C6 D2  IQDN  DEC IQ        KIM'S TOO SMART
0304  10 0F        BPL DONE      LOWER THE I.Q.
0306  E6 D2  IQUP  INC IQ        NOT BELOW ZERO
0308  A9 10        LDA #$10      NOT OVER 10 HEX
030A  C5 D2        CMP IQ
030C  90 F4        BCC IQDN
030E  B0 05        BCS DONE
0310  A9 0C  STIQ  LDA #$0C      START WITH 75%
0312  85 D2  IQST  STA IQ        I.Q.
0314  D8           CLD
0315  20 A6 03 DONE JSR KEYS      DISPLAY RESULTS-GET KEY
0318  A0 01        LDY #$01      START WITH KIM
031A  C9 13        CMP #$13      IF "GO" KEY PRESSED
031C  F0 28        BEQ SEMO
031E  88           DEY           START WITH PLAYER..
031F  C9 12        CMP #$12      IF "+" KEY PRESSED
0321  F0 23        BEQ SEMO
0323  C9 14        CMP #$14      "PC" PRESSED - SKIP
0325  D0 EE        BNE DONE      NO KEY - LOOP
0327  A9 0D  CHIQ  LDA #$0D
0329  85 FB        STA POINTH    SHOW "ODDS"
032B  A9 D5        LDA #$D5
032D  85 FA        STA POINTL
032F  A5 D2        LDA IQ        AND I.Q.
0331  85 F9        STA INH
0333  20 1F 1F     JSR SCANDS    ON DISPLAY
0336  20 40 1F     JSR KEYPR
0339  20 6A 1F     JSR GETKEY
```

```
033C   C9 11            CMP #$11        "DA" KEY PRESSED
033E   F0 D5            BEQ DONE        RETURN TO "DONE" LOOP
0340   B0 E5            BCS CHIQ        KEEP TRYING IF OVER "AD"
0342   85 D2            STA IQ          UNER 11(HEX), CHANGE
0344   90 E1            BCC CHIQ        IQ TO KEY #, NO KEY AGAIN
0346   84 DB     SEMO   STY MODE        SET STARTING PLAY
0348   4C 00 02         JMP NEW         ANOTHER GAME
034B   EA               NOP
                  ****** SUBROUTINE "DISPLAY" ******
034C   A9 7F     DISPLAY LDA #$7F
034E   8D 41 17         STA PADD        OPEN DISPLAY CHANELS
0351   E6 DA            INC RATE
0353   A0 00            LDY #$00
0355   A2 0B     DIGX   LDX #$0B        INDEX DIGIT
0357   B9 C0 00  SEGY   LDA SQST,Y      GET CONTROL BYTE
035A   85 FC            STA SAVE        SAVE IT
035C   F0 14            BEQ OFF         OPEN SQUARE
035E   29 20            AND #$20        BLINK FLAG
0360   F0 04            BEQ FLIC        NOT ON - SKIP BLINK
0362   24 DA            BIT RATE
0364   70 0C            BVS OFF         ALTERNATE ON-OFF
0366   A5 FC     FLIC   LDA SAVE
0368   29 40            AND #$40        STEADY FLAG
036A   D0 0A            BNE ON          ON - SKIP FLICKER
036C   A5 DA            LDA RATE
036E   29 08            AND #$08        FLICKER RATE
0370   F0 04            BEQ ON          ON
0372 A9 00       OFF    LDA #$00        OFF
0374   F0 03            BEQ DIGT
0376   B9 0F 01  ON     LDA SEGS,Y
0379   84 FC     DIGT   STY SAVE        SAVE FROM LOSS IN SUBR.
037B   20 4E 1F         JSR CONVD+6     DISPLAY A SEGMENT
037E   C8               INY
037F   C0 09            CPY #$09        LAST SQUARE
0381   F0 06            BEQ LAST        YES - DONE
0383   E0 11            CPX #$11        NO, LAST DIGIT?
0385   F0 CE            BNE DIGX        YES - REPEAT DIGITS
0387   D0 CE            BNE SEGY        NO - NEXT DIGIT
0389   60        LAST   RTS
                  ****** SUBROUTINE "RS ADD" ******
038A   B5 BF     RSA    LDA SQST,X
038C   85 D9            STA TEMP
038E   24 D9            BIT TEMP        WHO'S SQUARE?
0390   30 06            BMI KIM         KIM'S
0392   70 08            BVS PLYR        PLAYER'S
0394   A9 00     OPEN   LDA #$00        OPEN SQUARE VALUE
0396   F0 06            BEQ ADD
0398   A9 04     KIM    LDA #$04        KIM VALUE
039A   D0 02            BNE ADD
039C   A9 01     PLYR   LDA #$01        PLAYER VALUE
039E   18        ADD    CLC
039F   79 C8 00         ADC RS,Y        ADD TO ROW STATUS
03A2   99 C8 00         STA RS,Y        BYTE
03A5   60               RTS
```

```
                    ****** SUBROUTINE "KEYS" ******
03A6   20 4C 03  BACK      JSR DISPLAY     DISPLAY LOOP
03A9   20 40 1F            JSR ANYK        UNLESS
03AC   F0 F8               BEQ BACK        A KEY IS PRESSED
03AE   20 6A 1F            JSR KEYS        THEN GET A NUMBER
03B1   AA                  TAX             RECOVER THE FLAGS
03B2   60                  RTS
                  ****** SUBROUTINE "RANDOM" ******
03B3   D8                  CLD
03B4   38                  SEC             GENERATES A..
03B5   A9 D4               LDA R+1         RANDOM NUMBER
03B7   65 D7               ADC R+4         (THANKS TO J. BUTTERFIELD)
03B9   65 D8               ADC R+5
03BB   85 D3               STA R
03BD   A2 04               LDX #$04
03BF   B5 D3     ROLL      LDA R,X
03C1   95 D4               STA R+1,X
03C3   CA                  DEX
03C4   10 F9               BPL ROLL
03C6   60                  RTS
03C7   EA                  NOP

                  ****   SUBROUTINE "PS LOAD"   ****
03C8   85 D9      PSL      STA TEMP
03CA   A2 09               LDX #$09
03CC   16 DB      XLP      ASL PS,X        SHIFT PREVIOUS DATA
03CE   16 DB               ASL PS,X        OUT OF THE WAY
03D0   CA                  DEX
03D1   D0 F9               BNE XLP
03D3   A0 08               LDY #$08
03D5   A5 D9      YLP      LDA TEMP
03D7   D9 C8 00            CMP RS,Y        COUNT THE TIMES AN OPEN..
03DA   D0 12               BNE NOCT        SQUARE FITS THE..
03DC   BE 17 01            LDX SQ1,Y       TEST PARAMETER
03DF   20 40 01             JSR T+1
03E2   BE 1F 01            LDX SQ2,Y
03E5   20 40 01            JSR T+1
03E8   BE 27 01            LDX SQ3,Y
03EB   20 40 01            JSR T+1
03EE   88         NOCT     DEY
03EF   D0 E4               BNE YLP
03F1   60                  RTS
                ****** SUBROUTINE "RANDOM PLAYS" ******
03F2   20 B3 03  RPLA      JSR RAND        GET RANDOM NUMBER
03F5   29 0E               AND #$0E        0 - E (EVEN)
03F7   05 B6               ORA 0DEV        MAKE IT ODD IF 01
03F9   F0 F7               BEQ RPLA        NO ZEROS
03FB   C9 0A               CMP #$0A
03FD   B0 F3               BCS RPLA        LOOP TILL DONE
03FF   60                  RTS
```

****** HEX DUMP - KIM TAC TOE ******

```
0100 4C 10 03 EA EA EA A9 20 15 BF 95 BF 60 EA EA 08
0110 08 08 40 40 40 01 01 01 01 04 07 01 02 03 01 03
0120 02 05 08 04 05 06 05 05 03 06 09 07 08 09 09 07
0130 85 D9 A2 09 A5 D9 35 DB 24 D9 D0 03 CA D0 F5 60
0140 B5 BF D0 02 F6 DB 60 95 BF A0 08 A9 00 99 C8 00
0150 BE 17 01 20 8A 03 BE 1F 01 20 8A 03 BE 27 01 20
0160 8A 03 88 D0 E6 60
0200 A9 00 A2 1D 95 B4 CA D0 FB A9 05 85 BB A0 04 20
0210 F2 03 A2 04 D5 BB F0 F7 CA D0 F9 99 BB 00 88 D0
0220 EE E6 B6 A0 04 20 F2 03 A2 05 D5 B6 F0 F7 CA D0
0230 F9 99 B6 00 88 D0 EE A9 03 A0 08 D9 C8 00 F0 05
0240 88 D0 F8 F0 15 BE 17 01 20 06 01 BE 1F 01 20 06
0250 01 BE 27 01 20 06 01 4C FE 02 A2 09 A9 C0 35 BF
0260 F0 0E CA D0 F7 A2 09 20 06 01 CA D0 FA 4C 15 03
0270 E6 B5 A5 DB D0 17 20 A6 03 F0 FB C9 0A B0 F7 AA
0280 B4 BF D0 F2 A9 40 20 47 01 E6 DB D0 AA 20 4C 03
0290 E6 D1 D0 F9 A9 08 20 C8 03 A9 02 20 C8 03 A9 04
02A0 20 C8 03 A9 01 20 C8 03 A9 C0 20 30 01 D0 43 A9
02B0 30 20 30 01 D0 3C A9 08 20 30 01 D0 35 20 B3 03
02C0 29 0F C5 D2 B0 1F A4 B5 C0 01 D0 04 29 01 D0 17
02D0 C0 04 D0 06 24 C4 30 0D 70 07 A9 02 20 30 01 D0
02E0 11 A0 05 D0 02 A0 09 B6 B6 B5 BF F0 05 88 D0 F7
02F0 F0 F3 A9 80 20 47 01 C6 DB A9 0C 4C 39 02 A5 DB
0300 D0 04 C6 D2 10 0F E6 D2 A9 10 C5 D2 90 F4 B0 05
0310 A9 0C 85 D2 D8 20 A6 03 A0 01 C9 13 F0 28 88 C9
0320 12 F0 23 C9 14 D0 EE A9 0D 85 FB A9 D5 85 FA A5
0330 D2 85 F9 20 1F 1F 20 40 1F 20 6A 1F C9 11 F0 D5
0340 B0 E5 85 D2 90 E1 84 DB 4C 00 02 EA A9 7F 8D 41
0350 17 E6 DA A0 00 A2 0B B9 C0 00 85 FC F0 14 29 20
0360 F0 04 24 DA 70 0C A5 FC 29 40 D0 0A A5 DA 29 08
0370 F0 04 A9 00 F0 03 B9 0F 01 84 FC 20 4E 1F C8 C0
0380 09 F0 06 E0 11 F0 CE D0 CE 60 B5 BF 85 D9 24 D9
0390 30 06 70 08 A9 00 F0 06 A9 04 D0 02 A9 01 18 79
03A0 C8 00 99 C8 00 60 20 4C 03 20 40 1F F0 F8 20 6A
03B0 1F AA 60 D8 38 A9 D4 65 D7 65 D8 85 D3 A2 04 B5
03C0 D3 95 D4 CA 10 F9 60 EA 85 D9 A2 09 16 DB 16 DB
03D0 CA D0 F9 A0 08 A5 D9 D9 C8 00 D0 12 BE 17 01 20
03E0 40 01 BE 1F 01 20 40 01 BE 27 01 20 40 01 88 D0
03F0 E4 60 20 B3 03 29 0E 05 B6 F0 F7 C9 0A B0 F3 60
```

****** ZERO PAGE USAGE ******

00B6	ODD/EVEN MODIFIER
00C0-C8	PRESTORED RANDOM PLAYS
00C9-D0	ROWS STATUS
00D1	DELAY TIMER
00D2	I.Q.
00D3-D8	RANDOM NUMBER REGISTERS
00D9	TEMPORARY STORAGE
00DA	FLICKER / BLINK RATE
00DB	PLAY MODE
00DC-E4	PLAY STATUS
00FC	SAVE

LUNAR LANDER

Jim Butterfield

Description -

The program starts at 0200. When started, you will find yourself at 4500 feet and falling. The thrust on your machine is set to low; so you'll pick up speed due to the force of gravity.

You can look at your fuel at any time by pressing the "F" button. Your fuel (initially 800 pounds) will be shown in the first four digits of the KIM display.

The last two digits of the KIM display always show your rate of descent or ascent. "A" restores altitude.

Set your thrust by pressing buttons 1 through 9. Warning: button 0 turns your motor off, and it will not reignite! A thrust of 1, minimum, burns very little fuel; but gravity will be pulling your craft down faster and faster. A thrust of 9, maximum, overcomes gravity and reduces your rate of descent very sharply. A thrust of 5 exactly counterbalances gravity; you will continue to descend (or ascend) at a constant rate. If you run out of fuel, your thrust controls will become inoperative.

A safe landing is considered to be one where you land at a descent rate of 5 or less. After you land, your thrust controls will be inoperative, since the motor is automatically turned off; but you can still press "F" to look at your fuel. Pressing "GO" starts a new flight.

Suggestions for a safe flight:

(1) Conserve fuel at the beginning by pressing 1. You will begin to pick up speed downwards.

(2) When your rate of descent gets up to the 90's, you're falling fast enough. Press 5 to steady the rate.

(3) When your altitude reaches about 1500 feet, you'll need to slow down. Press 9 and slow down fast.

(4) When your rate of descent has dropped to 15 to 20, steady the craft by pressing 5 or 6. Now you're on your own.

```
                    ;  main routine - initialization
0200 A2 0D      GO      LDX #13   fourteen bytes
0202 BD CC 02   LP1     LDA INIT,X
0205 95 D5              STA ALT,X
0207 CA                 DEX
0208 10 F8              BPL LP1
                    ;  update height & velocity
020A A2 05      CALC    LDX #5
020C A0 01      RECAL   LDY #1
020E F8                 SED
020F 18                 CLC
```

```
0210 B5 D5    DIGIT  LDA ALT,X
0212 75 D7           ADC ALT+2,X  add each digit
0214 95 D5           STA ALT,X
0216 CA              DEX
0217 88              DEY
0218 10 F6           BPL DIGIT     next digit
021A B5 D3           LDA ALT+3,X  hi-order .. zero..
021C 10 02           BPL INCR           .. or ..
021E A9 99           LDA #$99
0220 75 D5    INCR   ADC ALT,X
0222 95 D5           STA ALT,X
0224 CA              DEX
0225 10 E5           BPL RECAL  do next addition
0227 A5 D5           LDA ALT
0229 10 0D           BPL UP    still flying?
022B A9 00           LDA #0     nope, turn off
022D 85 E2           STA DOWN
022F A2 02           LDX #2
0231 95 D5    DD     STA ALT,X
0233 95 DB           STA TH2,X
0235 CA              DEX
0236 10 F9           BPL DD
0238 38       UP     SEC    update fuel
0239 A5 E0           LDA FUEL+2
023B E5 DD           SBC THRUST
023D 85 E0           STA FUEL+2
023F A2 01           LDX #1     two more digits to go
0241 B5 DE    LP2    LDA FUEL,X
0243 E9 00           SBC #0
0245 95 DE           STA FUEL,X
0247 CA              DEX
0248 10 F7           BPL LP2
024A B0 0C           BCS TANK  still got fuel?
024C A9 00           LDA #0     nope, kill motor
024E A2 03           LDX #3
0250 95 DD    LP3    STA THRUST,X
0252 CA              DEX
0253 10 FB           BPL LP3
                ; show alt, fuel, or messages
0255 20 BD 02        JSR THRSET
0258 A5 DE    TANK   LDA FUEL   fuel into regstrs
025A A6 DF           LDX FUEL+1
025C 09 F0           ORA #$F0    plus F flag
025E A4 E1           LDY MODE
0260 F0 20           BEQ ST
0262 F0 9C    GOLINK BEQ GO
0264 F0 A4    CLINK  BEQ CALC
0266 A2 FE           LDX #$FE
0268 A0 5A           LDY #$5A
026A 18              CLC
026B A5 D9           LDA VEL+1
026D 69 05           ADC #5
026F A5 D8           LDA VEL
0271 69 00           ADC #0
```

```
0273 B0 04            BCS GOOD
0275 A2 AD            LDX #$AD
0277 A0 DE            LDY #$DE
0279 98       GOOD    TYA
027A A4 E2            LDY DOWN
027C F0 04            BEQ ST
027E A5 D5            LDA ALT
0280 A6 D6            LDX ALT+1
0282 85 FB    ST      STA POINTH
0284 86 FA            STX POINTL
                      ; show rate of ascent/descnt as absolute
0286 A5 D9            LDA VEL+1
0288 A6 D8            LDX VEL     up or down?
028A 10 05            BPL FLY       ..up, we're OK
028C 38               SEC
028D A9 00            LDA #0
028F E5 D9            SBC VEL+1
0291 85 F9    FLY     STA INH
0293 A9 02            LDA #2      loop twice thru display
0295 85 E3            STA DECK
0297 D8       FLITE   CLD       display & key test
0298 20 1F 1F         JSR SCANDS  light 'em up
029B 20 6A 1F         JSR GETKEY  check keys
029E C9 13            CMP #$13    GO key?
02A0 F0 C0            BEQ GOLINK    ..yes
02A2 B0 03            BCS NOKEY   ..if no key
02A4 20 AD 02         JSR DOKEY
02A7 C6 E3    NOKEY   DEC DECK
02A9 D0 ED            BNE FLITE
02AB F0 B7            BEQ CLINK   to CALC
                      ; subroutine to test keys
02AD C9 0A    DOKEY   CMP #$0A    test numeric
02AF 90 05            BCC NUMBER
02B1 49 0F            EOR #$0F  Fuel F gives 0 flag
02B3 85 E1            STA MODE
02B5 60       RETRN   RTS
02B6 AA       NUMBER  TAX
02B7 A5 DD            LDA THRUST    test; is motor off?
02B9 F0 FA            BEQ RETRN   yes, ignore key
02BB 86 DD            STX THRUST  no, set thrust
02BD                  ;calculate accel as thrust minus 5
02BD A5 DD    THRSET  LDA THRUST
02BF 38               SEC
02C0 F8               SED
02C1 E9 05            SBC #5
02C3 85 DC            STA TH2+1
02C5 A9 00            LDA #0
02C7 E9 00            SBC #0
02C9 85 DB            STA TH2
02CB 60               RTS
                      ; initial values
02CC 45 01 00 INIT    .BYTE $45,1,0   altitude
02CF 99 81 00         .BYTE $99,$81,0 rate of ascent
```

```
02D2 99 97          .BYTE $99,$97    acceleration
02D4 02             .BYTE 2          thrust
02D5 08 00 00       .BYTE 8,0,0      fuel
02D8 01             .BYTE 1          display mode
02D9 01             .BYTE 1          in flight/landed
                ; end

   00D5          ALT     *=*+3
   00D8          VEL     *=*+3
   00DB          TH2     *=*+2
   00DD          THRUST  *=*+1
   00DE          FUEL    *=*+3
   00E1          MODE    *=*+1
   00E2          DOWN    *=*+1
   00E3          DECK    *=*+1
                 ;  linkages to KIM monitor
                 SCANDS =$1F1F
                 GETKEY =$1F6A
                 POINTH =$FB
                 POINTL =$FA
                 INH    =$F9
```

******* Hex Dump - Lunar Lander *******

```
0200 A2 0D BD CC 02 95 D5 CA 10 F8 A2 05 A0 01 F8 18
0210 B5 D5 75 D7 95 D5 CA 88 10 F6 B5 D8 10 02 A9 99
0220 75 D5 95 D5 CA 10 E5 A5 D5 10 0D A9 00 85 E2 A2
0230 02 95 D5 95 DB CA 10 F9 38 A5 E0 E5 DD 85 E0 A2
0240 01 B5 DE E9 00 95 DE CA 10 F7 B0 0C A9 00 A2 03
0250 95 DD CA 10 FB 20 BD 02 A5 DE A6 DF 09 F0 A4 E1
0260 F0 20 F0 9C F0 A4 A2 FE A0 5A 18 A5 D9 69 05 A5
0270 D8 69 00 B0 04 A2 AD A0 DE 98 A4 E2 F0 04 A5 D5
0280 A6 D6 85 FB 86 FA A5 D9 A6 D8 10 05 38 A9 00 E5
0290 D9 85 F9 A9 02 85 E3 D8 20 1F 1F 20 6A 1F C9 13
02A0 F0 C0 B0 03 20 AD 02 C6 E3 D0 ED F0 B7 C9 0A 90
02B0 05 49 0F 85 E1 60 AA A5 DD F0 FA 86 DD A5 DD 38
02C0 F8 E9 05 85 DC A9 00 E9 00 85 DB 60 45 01 00 99
02D0 81 00 99 97 02 08 00 00 01 01
```

ACKNOWLEDGEMENTS: Ted Beach suggested the addition of the F flag when displaying fuel. Chuck Eaton spotted the cause of an erratic bug in the original keyboard input subroutine. Thanks to both.

MULTI-MAZE

BY JIM BUTTERFIELD

Description: Find your way out of the maze. You are the flashing light in the centre of the display. As you move up (key 9), down (1), left (4) or right (6), KIM will keep you in the central display; you'll see the walls of the maze moving by as you travel. Like walking through a real maze, you'll only see a small part of the maze as you pass through. If you can get out, you'll find yourself in a large open area; that means you've won. Press GO at any time for a new maze. Program starts at address 0200.

Listing:

```
0200 E6 D0      START  INC RND        random seed
0202 20 40 1F          JSR KEYIN
0205 D0 F9             BNE START
0207 A2 07             LDX #7         patch the maze
0209 26 D0      LP1    ROL RND        in 8 places
020B 90 17             BCC NXUP
020D BC 08 03          LDY PLACE,X
0210 BD 10 03          LDA POINT1,X
0213 59 DE 02          EOR MAZE,Y
0216 99 DE 02          STA MAZE,Y
0219 C8                INY
021A C8                INY
021B BD 18 03          LDA POINT2,X
021E 59 DE 02          EOR MAZE,Y
0221 99 DE 02          STA MAZE,Y
0224 CA         NXUP   DEX
0225 10 E2             BPL LP1
0227 A2 02             LDX #2
0229 D8                CLD
022A 30 D4      SLINK  BMI START
022C BD DB 02   SETUP  LDA INIT,X
022F 95 D2             STA MZPT,X
0231 CA                DEX            3 values from INIT
0232 10 F8             BPL SETUP
                ;  pick out specific part of maze
0234 A0 0B      MAP    LDY #11
0236 B1 D2      GETMOR LDA (MZPT),Y   6 rows x 2
0238 99 D8 00          STA WORK,Y
023B 88                DEY
023C 10 F8             BPL GETMOR
                ; shift for vertical position
023E A2 0A             LDX #10        for each of 6 rows
0240 A4 D4      NXDIG  LDY POSIT      ..shift Y positions
0242 A9 FF             LDA #$FF       filling with 'walls'
0244 38         REROL  SEC            ...on both sides
0245 36 D9             ROL WORK+1,X
0247 36 D8             ROL WORK,X     roll 'em
0249 2A                ROL A
024A 88                DEY
024B D0 F7             BNE REROL
```

```
                    ; calculate segments
024D 29 07          NXDIG   AND #7
024F A8                     TAY
0250 B9 C6 02               LDA TAB1,Y    3 bits to segment
0253 95 D8                  STA WORK,X      ..stored
0255 CA                     DEX
0256 CA                     DEX
0257 10 E7                  BPL NXDIG
                    ; test flasher
0259 C6 D5          LIGHT   DEC PLUG    time out?
025B 10 0A                  BPL MUG     ..no
025D A9 05                  LDA #5      ..yes, reset
025F 85 D5                  STA PLUG
0261 A5 DE                  LDA WORK+6    ..and..
0263 49 40                  EOR #$40       ..flip..
0265 85 DE                  STA WORK+6      ..flasher
                    ; light display
0267 A9 7F          MUG     LDA #$7F    open the gate
0269 8D 41 17               STA SADD
026C A0 09                  LDY #$09
026E A2 0A                  LDX #10
0270 B5 D8          SHOW    LDA WORK,X  tiptoe thru..
0272 8D 40 17               STA SAD      ..the segments
0275 8C 42 17               STY SBD
0278 C6 D6          ST1     DEC STALL    ..pausing
027A D0 FC                  BNE ST1
027C C8                     INY
027D C8                     INY
027E CA                     DEX
027F CA                     DEX
0280 10 EE                  BPL SHOW
                    ; test new key depression
0282 20 40 1F               JSR KEYIN   set dir reg
0285 20 6A 1F               JSR GETKEY
0288 C5 D7                  CMP SOK     same as last?
028A F0 CD                  BEQ LIGHT
028C 85 D7                  STA SOK
                    ; test which key
028E A2 04                  LDX #4   5 items in table
0290 DD CE 02       SCAN    CMP TAB2,X
0293 F0 05                  BEQ FOUND
0295 CA                     DEX
0296 10 F8                  BPL SCAN
0298 30 BC                  BMI LIGHT
029A CA             FOUND   DEX
029B 30 8D                  BMI SLINK  go key?
029D BC D3 02               LDY TAB3,X
02A0 B9 D8 00               LDA WORK,Y
02A3 3D D7 02               AND TAB4,X
02A6 D0 B1                  BNE LIGHT
                    ; move
02A8 CA                     DEX
02A9 10 04                  BPL NOTUP
02AB C6 D4                  DEC POSIT  upward move
02AD D0 85          MLINK   BNE MAP    l.o.n.g  branch
```

```
02AF D0 04      NOTUP   BNE SIDEWY
02B1 E6 D4              INC POSIT   downward move
02B3 D0 F8              BNE MLINK
02B5 CA         SIDEWY  DEX
02B6 D0 06              BNE LEFT
02B8 C6 D2      RIGHT   DEC MZPT    right move
02BA C6 D2              DEC MZPT
02BC D0 EF              BNE MLINK
02BE E6 D2      LEFT    INC MZPT    left move
02C0 E6 D2              INC MZPT
02C2 D0 E9              BNE MLINK
02C4 F0 F2              BEQ RIGHT
                ; tables follow in Hex format
02C6 TAB1    00 08 40 48 01 09 41 49
02CE TAB2    13 09 01 06 04
02D3 TAB3    06 06 04 08
02D7 TAB4    01 08 40 40
02DB INIT    DA 02 08
02DE MAZE    FF FF 04 00 F5 7F 15 00 41 FE 5F 04 51 7D 5D 04
             51 B6 54 14 F7 D5 04 54 7F 5E 01 00 FD FF 00 00
             00 00 00 00 00 00 00 00 00 00
0308 PLACE   05 0B 10 10 14 18 17 10
0310 POINT1  01 04 80 10 80 02 40 40
0318 POINT2  02 02 40 01 10 04 80 10
                ; end of program
```

***** Hex Dump - Multimaze *****

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200 E6 D0 20 40 1F D0 F9 A2 07 26 D0 90 17 BC 08 03
0210 BD 10 03 59 DE 02 99 DE 02 C8 C8 BD 18 03 59 DE
0220 02 99 DE 02 CA 10 E2 A2 02 D8 30 D4 BD DB 02 95
0230 D2 CA 10 F8 A0 0B B1 D2 99 D8 00 88 10 F8 A2 0A
0240 A4 D4 A9 FF 38 36 D9 36 D8 2A 88 D0 F7 29 07 A8
0250 B9 C6 02 95 D8 CA CA 10 E7 C6 D5 10 0A A9 05 85
0260 D5 A5 DE 49 40 85 DE A9 7F 8D 41 17 A0 09 A2 0A
0270 B5 D8 8D 40 17 8C 42 17 C6 D6 D0 FC C8 C8 CA CA
0280 10 EE 20 40 1F 20 6A 1F C5 D7 F0 CD 85 D7 A2 04
0290 DD CE 02 F0 05 CA 10 F8 30 BC CA 30 8D BC D3 02
02A0 B9 D8 00 30 D7 02 D0 B1 CA 10 04 C6 D4 D0 85 D0
02B0 04 E6 D4 D0 F8 CA D0 06 C6 D2 C6 D2 D0 EF E6 D2
02C0 E6 D2 D0 E9 F0 F2 00 08 40 48 01 09 41 49 13 09
02D0 01 06 04 06 06 04 08 01 08 40 40 DA 02 08 FF FF
02E0 04 00 F5 7F 15 00 41 FE 5F 04 51 7D 5D 04 51 B6
02F0 54 14 F7 D5 04 54 7F 5E 01 00 FD FF 00 00 00 00

0300 00 00 00 00 00 00 00 00 05 0B 10 10 14 18 17 10
0310 01 04 80 10 80 02 40 40 02 02 40 01 10 04 80 10
```

MUSIC BOX

JIM BUTTERFIELD

DESCRIPTION

THIS PROGRAM PLAYS ONE OR SEVERAL TUNES VIA THE "AUDIO OUT" INTERFACE OF KIM-1. USE THE SAME CONNECTION AS THAT FOR RECORDING ON CASSETTE TAPE. IF YOUR TAPE RECORDER HAS A "MONITOR" FEATURE, YOU CAN LISTEN TO THE TUNE AS WELL AS RECORD IT. ALTERNATIVELY, AN AMPLIFIER WILL PLAY THE SIGNAL THROUGH A SPEAKER.

HOW TO RUN

LOAD THE PROGRAM. LOAD THE TUNE(S) EITHER FROM CASSETTE TAPE, PAPER TAPE, OR KEYBOARD ENTRY. BE SURE TO STORE THE VALUE FA AT THE END OF EACH TUNE, AND BEHIND THE LAST TUNE, STORE: FF 00.
STARTING ADDRESS FOR THE PROGRAM IS 200. ENTER AD 0 2 0 0 GO

HOW TO WRITE YOUR OWN TUNE(S)

EACH NOTE GOES INTO A BYTE OF STORAGE, STARTING AT LOCATION 0000 OF MEMORY. EACH TUNE SHOULD END WITH THE VALUE FA WHICH STOPS THE PROGRAM UNTIL GO IS PRESSED.

SPECIAL CODES ARE INCORPORATED IN THE PROGRAM TO ALLOW CERTAIN EFFECTS - ADJUSTMENT OF SPEED, TONE, ETC. THE CODES ARE FOLLOWED BY A VALUE WHICH SETS THE PARTICULAR EFFECT. CODES ARE LISTED BELOW.

CODE	EFFECT	INITIALLY	EXAMPLES
FB	SETS SPEED OF TUNE	$30	18 IS QUICK; 60 IS SLOW
FC	SETS LENGTH OF "LONG" NOTES	02	2 MEANS, "LONG NOTE LASTS TWICE AS LONG AS SHORT"
FD	SETS OCTAVE (PITCH)	01	2 IS BASS; 4 IS DEEP BASS.
FE	SETS INSTRUMENT	$FF	FF IS PIANO; 00 IS CLARINET.
FF	SETS ADDRESS FOR TUNE	00	00 WILL TAKE YOU BACK TO FIRST TUNE; LIKE A "JUMP".

FOR EXAMPLE, AT ANY TIME DURING A TUNE, YOU MAY INSERT THE SEQUENCE FB 18 AND THE TUNE WILL THEN BEGIN TO PLAY AT FAST SPEED. INSERTING FF 45 WILL CAUSE A SWITCH TO THE TUNE AT ADDRESS 45. THE INITIAL VALUES SHOWN CAN BE RESET AT ANY TIME BY STARTING AT ADDRESS 200.

NO TUNE SHOULD EXTEND BEYOND ADDRESS DF, SINCE PROGRAM VALUES ARE STORED AT E0 AND UP.

THE PROGRAM CAN BE EASILY CONVERTED TO A SUBROUTINE (BY REPLACING THE BRK INSTRUCTION WITH A RTS). THIS ALLOWS THE PROGRAMMER TO PLAY VARIOUS "PHRASES" OF MUSIC TO PRODUCE QUITE COMPLEX TUNES.

THE LOWEST NOTE YOU CAN PLAY IS A BELOW MIDDLE C. FOR EACH NOTE, YOU CAN SELECT WHETHER IT IS PLAYED AS A LONG NOTE OR A SHORT NOTE (NORMALLY, A LONG NOTE WILL LAST TWICE AS LONG AS A SHORT NOTE).

SOME OF THE NOTES ARE AS FOLLOWS:

	NOTE	SHORT	LONG
	A	75	F5
	A#	6E	EE
	B	68	E8
MIDDLE	C	62	E2
	C#	5C	DC
	D	56	D6
	D#	52	D2
	E	4D	CD
	F	48	C8
	F#	44	C4
	G	40	C0
	G#	3C	BC
	A	39	B9
	A#	35	B5
	B	32	B2
HIGH	C	2F	AF
	C#	2C	AC
	D	29	A9
	E	24	A4
	F	22	A2
	G	1E	9E
PAUSE		00	80

```
        ; INITIALIZE - RESET WORK PARAMETERS

0200  A2 05     START     LDX #$05
0202  BD 86 02  LP1       LDA INIT,X
0205  95 E0               STA WORK,X
0207  CA                  DEX
0208  10 F8               BPL LP1

        ; MAIN ROUTINE HERE - WORK NOT RESET

020A  A9 BF     GO        LDA #$BF
020C  8D 43 17            STA PBDD        OPEN OUTPUT CHANNEL
020F  A0 00               LDY #$00
0211  B1 E4               LDA (WORK+4),Y GET NEXT NOTE
0213  E6 E4               INC WORK+4
0215  C9 FA               CMP #$FA        TEST FOR HALT
0217  D0 04               BNE NEXT
0219  00                 BRK             (OR RTS IF USED AS SUBR.)
021A  EA                  NOP
021B  F0 ED               BEQ GO          RESUME WHEN GO PRESSED
021D  90 0B     NEXT      BCC NOTE        IS IT A NOTE?
021F  E9 FB               SBC #$FB        IF NOT, DECODE INSTR.
0221  AA                  TAX             AND PUT INTO X
```

```
0222 B1 E4          LDA (WORK+4),Y  get parameter
0224 E6 E4          INC WORK+4
0226 95 E0          STA WORK,X      store in work table
0228 B0 E0          BCS GO          unconditional branch

                ; set up for timing note

022A A6 E0    NOTE  LDX WORK        timing
022C 86 E7          STX LIMIT+1
022E A6 E1          LDX WORK+1      long note factor
0230 A8             TAY             test accumulator
0231 30 02          BMI OVER        long note?
0233 A2 01          LDX #1          nope, set short note
0235 86 E6    OVER  STX LIMIT       store length factor
0237 29 7F          AND #$7F        remove short/long flag
0239 85 E9          STA VAL2
023B F0 02          BEQ HUSH        is it a pause?
023D 85 EA          STA VAL1        no, set pitch
023F A5 E9    HUSH  LDA VAL2        get timing and..
0241 25 E3          AND WORK+3      bypass if muted
0243 F0 04          BEQ ON
0245 E6 EA          INC VAL1        else fade the
0247 C6 E9          DEC VAL2              note
0249 A6 E9    ON    LDX VAL2
024B A9 A7          LDA #$A7        bit 7 on
024D 20 5D 02       JSR SOUND       delay half cycle
0250 30 B8          BMI GO
0252 A6 EA          LDX VAL1
0254 A9 27          LDA #$27        bit 7 off
0256 20 5D 02       JSR SOUND       delay the other half
0259 30 AF          BMI GO          end of note?
025B 10 E2          BPL HUSH        no, more cycles

                ; subroutine to send a bit

025D A4 E2    SOUND LDY WORK+2      octave flag
025F 84 EB          STY TIMER
0261 86 EC          STX XSAV        bit timing
0263 E0 00    SLOOP CPX #0          end of timing?
0265 D0 08          BNE CONT        no, continue
0267 A6 EC          LDX XSAV        restore timing
0269 C6 EB          DEC TIMER       in case of..
026B D0 F6          BNE SLOOP       ..another octave
026D F0 16          BEQ SEX         else exit
026F 8D 42 17 CONT  STA SBD
0272 CA             DEX
0273 C6 E8          DEC LIMIT+2
0275 D0 EC          BNE SLOOP
0277 C6 E7          DEC LIMIT+1
0279 D0 E8          BNE SLOOP
027B A4 E0          LDY WORK
027D 84 E7          STY LIMIT+1
027F C6 E6          DEC LIMIT
```

```
0281  D0 E0              BNE SLOOP
0283  A9 FF              LDA #$FF
0285  60          SEX    RTS
                  ; INITIAL CONSTANTS
0286  30 02 01    INIT   .BYTE $30,2,1,$FF,0,0
      FF 00 00
```

SAMPLE MUSIC FOR MUSIC BOX PROGRAM

```
0000 FB 18 FE FF 44 51 E6 E6 66 5A 51 4C C4 C4 C4 D1
0010 BD BD BD 00 44 BD 00 44 3D 36 33 2D A8 80 80 33
0020 44 B3 80 80 44 51 C4 80 80 5A 51 E6 80 80 FA

0020                                              FE
0030 00 FB 28 5A 5A 51 48 5A 48 D1 5A 5A 51 48 DA E0
0040 5A 5A 51 48 44 48 51 5A 60 79 6C 60 DA DA FA

0040                                                 FE
0050 FF 5A 5A 5A 5A 5A 5A 66 72 79 E6 E6 80 00 56 56
0060 56 56 56 56 5A 66 F2 80 80 4C 4B 4C 4C 4C 4C 56
0070 5A 56 4C 00 C4 44 4C 56 5A 5A 56 5A 66 56 5A 66
0080 F2 80 FE 00 00 72 5A CC 72 5A CC 72 5A CC 80 B8
0090 80 4C 56 5A 56 5A E6 F2 80 FA FF 00
```

NOTE THAT TUNES 1 AND 2 SET BOTHTHE SPEED AND THE INSTRUMENT. TUNE 3 CONTINUES AT THE SAME SPEED AS THE PREVIOUS ONE; BUT THE INSTRUMENT IS CHANGED DURING THE TUNE.

THE PROGRAM CAN BE CHANGED TO USE THE SPEAKER SHOWN IN FIGURE 5.1 OF THE KIM MANUAL AS FOLLOWS:

BYTE	INITIALLY	CHANGE TO
020D	43	01
024C	A7	FF
0255	27	00
0270	42	00

***** Extra Datafile for Music Box *****

```
0000-   FE 00 56 52 4D AF 4D AF 4D FC 06 AF FC 02 FE FF
0010-   2F 29 26 24 2F 29 A4 32 A9 FC 06 AF FC 02 FE 00
0020-   56 52 4D AF 4D AF 4D FC 06 AF FC 02 FE FF 39 40
0030-   44 39 2F A4 29 2F 39 A9 80 80 FE 00 56 52 4D AF
0040-   4D AF 4D FC 06 AF FC 02 FE FF 2F 29 26 24 2F 29
0050-   A4 32 A9 AF 80 80 2F 29 24 2F 29 A4 2F 29 2F 24
0060-   2F 29 A4 2F 29 2F 24 2F 29 A4 32 A9 AF 80 80 FA
0070-   FF 00
```

Note: be sure to set the break vector 17FE,FF (00,1C)

PING PONG

JIM BUTTERFIELD

Play against the computer, or change the program for a two-player game. On each shot, you choose between four plays: Spin, Lob, Block, or Slam. If you're playing the left side of the court, use the left-hand buttons (0, 4, 8 and C). See the diagram at right.

C	slam	F
8	block	B
4	lob	7
0	spin	3

Each shot has its own strengths and weaknesses: for example, a Slam is a powerful shot, but it's also likely to be "fluffed". Strategy is not trivial - your chances of success on any play depend not only on your choice of shot, but on what shots have gone before. You'll have to learn the combinations the hard way.

You'll see the net in the middle of the court. Don't try to play the ball until it is on your side of the net, or you'll lose the point. Each type of shot has a distinctive appearance, which you'll learn to recognize. They are similar to the key positions: a Spin lights the bottom segment, a Lob lights the middle segment, a Block lights the upper segment, and the mighty Slam shot lights all three segments and travels faster.

The original version of the game was published for the HP-67 calculator in "65 Notes", V4N2P5. Authorship was not given.

At first, the shots will come too fast for you to cope with. There are two ways to solve this. The easy way is the "freeze" the ball by holding down any unused key, like AD or 7: play will be suspended until you figure out what you want to do next. The harder way, but not too hard, is just to slow down the ball by changing the program: locations 0331 to 0334 contain the speeds for each type of shot. Increase these values and the ball will slow down, e.g., 40 40 40 28 will halve the speed.

For a two-player game, where KIM does not play the right side, change location 032C to 01. To have KIM play the left side, change location 032B to 00.
KIM plays a strong game, but CAN BE BEATEN!

```
0200 20 40 1F START  JSR KEYIN    directional registrs
0203 20 6A 1F        JSR GETKEY         input key
0206 C9 13           CMP #$13     GO key?
0208 D0 0A           BNE NOGO     nope, skip
                ; GO key - set up game here
020A A2 08           LDX #8       get 9 ..
020C BD 24 03 SETUP  LDA INIT,X         ..inital valus
020F 95 80           STA SPEED,X        to zero page
0211 CA              DEX
0212 10 F8           BPL SETUP
                ; test legal keys (0,3,4,7,8,B,C,F)
0214 C9 10     NOGO  CMP #$10     key 0 to F?
0216 B0 22           BCS NOKEY    no, skip
0218 AA              TAX          save key in X
0219 29 03           AND #3       test column
021B F0 04           BEQ KEY      col 0 (0,4,8,C)?
021D C9 03           CMP #3       col 3 (3,7,B,F)?
021F D0 19           BNE NOKEY    neither - skip
0221 45 85     KEY   EOR PLACE    check vs ball postn
0223 A8              TAY
0224 29 04           AND #4       ball off screen?
0226 D0 12           BNE NOKEY
0228 8A              TXA          restore key
0229 45 84           EOR DIRECT   ball going away?
022B 29 02           AND #2
022D F0 0B           BEQ NOKEY    yes, ignore key
022F 98              TYA          ball position
0230 29 02           AND #2       wrong side of net?
0232 D0 69           BNE POINT    yes, lose!
                ; legal play found here
0234 8A              TXA          restore key
0235 4A 4A           LSRA LSRA    type (0=Spin, etc)
0237 20 B1 02        JSR SHOT     make shot
                ; key rtns complete - play ball
023A 20 40 1F NOKEY  JSR KEYIN    if key still prest..
023D D0 27           BNE FREEZE         freeze ball
023F C6 83           DEC PAUSE
0241 10 23           BPL FREEZE   wait til timeout
0243 A5 80           LDA SPEED
0245 85 83           STA PAUSE
0247 18              CLC
0248 A5 85           LDA PLACE    move..
024A 65 84           ADC DIRECT         ..ball
024C 85 85           STA PLACE
024E 29 04           AND #4       ball still..
0250 F0 14           BEQ FREEZE         in court?
                ; ball outside - KIM to play?
0252 A5 85           LDA PLACE
0254 30 04           BMI TESTL    ball on left
0256 A5 88           LDA PRITE    KIM plays right?
0258 10 02           BPL SKPT     unconditional
025A A5 87     TESTL LDA PLEFT    KIM plays left?
025C D0 3F     SKPT  BNE POINT    no, lose point
```

```
                    ; KIM plays either side here
025E A6 82          LDX LOG     log determines..
0260 BD 39 03       LDA PLAY,X  ..KIM's play
0263 20 B1 02       JSR SHOT    make the shot
0266 A9 7F    FREEZE LDA #$7F
0268 8D 41 17       STA PADD    open registers
                    ; light display here
026B A0 13          LDY #$13
026D A2 01          LDX #1
026F 86 89          STX DIGIT   count score digts
0271 A5 86          LDA SCORE
0273 4A 4A          LSRA LSRA   shift & store..
0275 4A 4A          LSRA LSRA   ..left player score
0277 85 8A          STA ARG
0279 A5 86          LDA SCORE
027B 29 0F          AND #$0F    ..right player score
027D AA             TAX
027E BD E7 1F HOOP  LDA TABLE,X
0281 20 A4 02       JSR SHOW
0284 A6 8A          LDA ARG
0286 C6 89          DEC DIGIT
0288 10 F4          BPL HOOP
028A A2 03          LDX #3
028C BD 2D 03 VUE   LDA PIX,X
028F E4 85          CPX PLACE
0291 D0 02          BNE NOPIX
0293 05 81          ORA SPOT    show the ball
0295 20 A4 02 NOPIX JSR SHOW
0298 CA             DEX
0299 10 F1          BPL VUE
029B 30 03          BMI SLINK
                    ; lose! score & reverse board
029D 20 E9 02 POINT JSR SKORE
02A0 D8       SLINK CLD
02A1 4C 00 02       JMP START   return to main loop
                    ; display subroutine
02A4 8D 40 17 SHOW  STA SAD
02A7 8C 42 17       STY SBD
02AA C6 8B    STALL DEC MOD
02AC D0 FC          BNE STALL
02AE 88 88          DEY DEY
02B0 60             RTS
02B1 A8       SHOT  TAY         save shot in Y
02B2 A6 82          LDX LOG     old log in X
02B4 06 82          ASL LOG
02B6 06 82          ASL LOG
02B8 05 82          ORA LOG
02BA 29 0F          AND #$F     update log book
02BC 85 82          STA LOG     ..last two shots
02BE 38             SEC
02BF A5 80          LDA SPEED
02C1 E5 83          SBC PAUSE   invert timing
02C3 85 83          STA PAUSE
```

```
                    ; set speed & display segment(s)
02C5 B9 31 03           LDA SPD,Y
02C8 85 80              STA SPEED
02CA B9 35 03           LDA SEG,Y
02CD 85 81              STA SPOT
                    ; test play success - random
02CF BD 49 03           LDA CHANCE,X  odds from log bk
02D2 88         GIT     DEY
02D3 30 04              BMI GET
02D5 4A 4A              LSRA LSRA
02D7 10 F9              BPL GIT       unconditional
02D9 29 03      GET     AND #3        odds 0 to 3..
02DB 0A                 ASL A         now 0 to 6
02DC 85 8C              STA TEMP
02DE AD 04 17           LDA TIMER     random number
02E1 29 07              AND #7        now 0 to 7
02E3 C5 8C              CMP TEMP
02E5 F0 33              BEQ REVRS     success?
02E7 90 31              BCC REVRS     success?
                    ; lose a point & position to serve
02E9 A2 04      SKORE   LDX #4        position ball R
02EB A5 84              LDA DIRECT
02ED 0A 0A              ASLA ASLA
02EF 0A 0A              ASLA ASLA
02F1 10 04              BPL OVER
02F3 A2 FF              LDX #$FF      position ball L
02F5 A9 01              LDA #1
02F7 86 85      OVER    STX PLACE
02F9 18                 CLC
02FA 65 86              ADC SCORE
02FC 85 86              STA SCORE
02FE A0 00              LDY #0    end game, kill ball
0300 AA         TLP     TAX
0301 29 0F              AND #$F       get one score
0303 C9 0B              CMP #$11      11 points?
0305 D0 02              BNE SKI
0307 84 84              STY DIRECT    kill ball
0309 8A         SKI     TXA
030A 4A 4A              LSRA LSRA
030C 4A 4A              LSRA LSRA
030E D0 F0              BNE TLP
                    ; set serve - speed, spot, log, pause
0310 A2 03              LDX #3
0312 BD 24 03 SRV       LDA INIT,X
0315 95 80              STA SPEED,X
0317 CA                 DEX
0318 10 F8              BPL SERVE
                    ; reverse ball direction
031A A5 84      REVRS   LDA DIRECT
031C 18                 CLC
031D 49 FF              EOR #$FF
031F 69 01              ADC #1
0321 85 84              STA DIRECT
0323 60                 RTS
```

```
                ; tables - in Hexadecimal format
0324 INIT       30 08 00 80 01 FF 00 01 00
032D PIX        00 06 30 00
0331 SPD        20 20 20 14
0335 SEG        08 40 01 49
0339 PLAY       02 02 01 02 01 03 01 02 03 03 00 02 00 00 02 02
0349 CHANCE     78 B5 9E 76 6E A1 AE 75 AA EB 8F 75 5B 56 7A 35
0359 end
```

Zero Page:

80: SPEED - speed ball travels
81: SPOT - segment(s) ball lights
82: LOG - record of recent plays
83: PAUSE - delay before ball moves
84: DIRECT - direction of ball
85: PLACE - position of ball
86: SCORE
87: PLEFT - 0 for KIM to play left
88: PRITE - 0 for KIM to play right

***** Hex Dump - Ping Pong *****

```
0200 20 40 1F 20 6A 1F C9 13 D0 0A A2 08 BD 24 03 95
0210 80 CA 10 F8 C9 10 B0 22 AA 29 03 F0 04 C9 03 D0
0220 19 45 85 A8 29 04 D0 12 8A 45 84 29 02 F0 0B 98
0230 29 02 D0 69 8A 4A 4A 20 B1 02 20 40 1F D0 27 C6
0240 83 10 23 A5 80 85 83 18 A5 85 65 84 85 85 29 04
0250 F0 14 A5 85 30 04 A5 88 10 02 A5 87 D0 3F A6 82
0260 BD 39 03 20 B1 02 A9 7F 8D 41 17 A0 13 A2 01 86
0270 89 A5 86 4A 4A 4A 4A 85 8A A5 86 29 0F AA BD E7
0280 1F 20 A4 02 A6 8A C6 89 10 F4 A2 03 BD 2D 03 E4
0290 85 D0 02 05 81 20 A4 02 CA 10 F1 30 03 20 E9 02
02A0 D8 4C 00 02 8D 40 17 8C 42 17 C6 8B D0 FC 88 88
02B0 60 A8 A6 82 06 82 06 82 05 82 29 0F 85 82 38 A5
02C0 80 E5 83 85 83 B9 31 03 85 80 B9 35 03 85 81 BD
02D0 49 03 88 30 04 4A 4A 10 F9 29 03 0A 85 8C AD 04
02E0 17 29 07 C5 8C F0 33 90 31 A2 04 A5 84 0A 0A 0A
02F0 0A 10 04 A2 FF A9 01 86 85 18 65 86 85 86 A0 00

0300 AA 29 0F C9 0B D0 02 84 84 8A 4A 4A 4A 4A D0 F0
0310 A2 03 BD 24 03 95 80 CA 10 F8 A5 84 18 49 FF 69
0320 01 85 84 60 30 C0 00 80 01 FF 00 01 00 00 06 30
0330 00 20 20 20 14 08 40 01 49 02 02 01 02 01 03 01
0340 02 03 03 00 02 00 00 02 02 78 B5 9E 76 6E A1 AE
0350 75 AA EB 8F 75 5B 56 7A 35
```

QUICK

By Peter Jennings
Modified by Jim Rutterfield

Description -

Here's a program to test your speed of reaction. Press "GO" and the display will blank for a random period of time. When it lights, hit any numbered button. The number on the display will tell you how quick you were; the smaller the number, the faster your reaction time. You may play repeatedly, just press "GO" each time you want a new test.

```
0300  A5 F9     START    LDA INH    RANDOMIZE DELAY
0302  2A                ROL A      ..BY MULTIPLYING
0303  65 F9             ADC INH       BY 3 AND
0305  29 7F             AND #$7F       MASKING
0307  85 FB             STA POINTH WORK IN DISPLAY AREA
0309  20 40 1F  ZIP      JSR KEYIN  IF YOU CHEAT BY KEYING...
030C  D0 FB              BNE ZIP    PROGRAM WAITS YOU OUT
030E  E6 FA              INC POINTL
0310  D0 F7              BNE ZIP    COUNT DOWN FOR
0312  E6 FB              INC POINTH  RANDOM DELAY
0314  D0 F3              BNE ZIP
0316  85 F9             STA INH    SET TO ZERO
0318  A2 FD     RUN      LDX #$FD   NEGATIVE THREE
031A  F8                 SED        COUNT IN DECIMAL
031B  38                 SEC        ADD VALUE 1
031C  B5 FC     DIGIT    LDA POINTH+1,X
031E  69 00              ADC #$00   ADD IT IN
0320  95 FC              STA POINTH+1,X
0322  E8                 INX        MOVE ON TO NEXT DIGITS
0323  D0 F7              BNE DIGIT
0325  D8                 CLD
0326  20 1F 1F           JSR SCANDS LIGHT UP COUNT
0329  F0 ED              BEQ RUN   AND KEEP COUNTING
032B  20 1F 1F  STAND    JSR SCANDS
032E  20 6A 1F           JSR GETKEY
0331  C9 13              CMP #$13   GO KEY DEPRESSED?
0333  D0 F6              BNE STAND  NOPE, HOLD IT
0335  F0 C9              BEQ START  YUP, START OVER
```

***** Hex Dump - Quick *****

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0300-    A5 F9 2A 65 F9 29 7F 85 FB 20 40 1F D0 FB E6 FA
0310-    D0 F7 E6 FB D0 F3 85 F9 A2 FD F8 38 B5 FC 69 00
0320-    95 FC E8 D0 F7 D8 20 1F 1F F0 ED 20 1F 1F 20 6A
0330-    1F C9 13 D0 F6 F0 C9
```

REVERSE

By Jim Butterfield

Start at 0200 - the display will show a combination of 6 letters such as CDBAEF. Hit a number from 2 to six to 'flip' letters. For example, if you hit 2 with the previous example, the first two letters will flip over to give DCBAEF. Now if you hit 4, you'll get the winning combination - ABCDEF - and the display will signal your win with a line of dashes.

The computer won't limit your number of flips - but try to get a win in 6 moves or less. By the way, the computer forbids doing the same flip twice in succession - so you can't back up a move.

```
0200 E6 16    START  INC RND+4     randomize
0202 20 40 1F        JSR KEYIN          **Game by Bob Albrecht -
0205 D0 F9           BNE START            People's Computer Co **
0207 D8              CLD
0208 A2 05           LDX #5
020A A9 00           LDA #0
020C 86 10           STX POINTR
020E 95 18    ZLOOP  STA WINDOW,X   set window  to zeros
0210 CA              DEX
0211 10 FB           BPL ZLOOP
0213 38       RAND   SEC
0214 A5 13           LDA RND+1     hash in new random number
0216 65 16           ADC RND+4
0218 65 17           ADC RND+5
021A 85 12           STA RND
021C A2 04           LDX #4
021E B5 12    RLP    LDA RND,X     move random string down one
0220 95 13           STA RND+1,X
0222 CA              DEX
0223 10 F9           BPL RLP
0225 A0 C0           LDY #$C0      divide random # by 6
0227 84 11           STY MOD
0229 A0 06           LDY #6
022B C5 11    SET    CMP MOD
022D 90 02           BCC PASS
022F E5 11           SBC MOD
0231 46 11    PASS   LSR MOD
0233 88              DEY
0234 D0 F5           BNE SET
0236 AA              TAX
0237 A4 10           LDY POINTR
0239 B9 F1 1F        LDA TABLE+10,Y   digits A to F
023C CA       TOP    DEX
023D 10 02           BPL TRY          find an empty window
023F A2 05           LDX #5
0241 B4 18    TRY    LDY WINDOW,X
0243 D0 F7           BNE TOP
0245 95 18           STA WINDOW,X       and put the digit in
0247 C6 10           DEC POINTR
0249 10 C8           BPL RAND
```

```
024B F0 B3     SLINK  BEQ START      link to start
024D A2 05     WTEST  LDX #5         test
024F B5 18     TEST2  LDA WINDOW,X       win
0251 DD A6 02         CMP WINNER,X          condition
0254 D0 0C            BNE PLAY
0256 CA               DEX
0257 10 F6            BPL TEST2
0259 A2 05            LDX #5
025B A9 40            LDA #$40       set
025D 95 18     SET    STA WINDOW,X      to
025F CA               DEX                "------"
0260 10 FB            BPL SET
0262 A9 7F     PLAY   LDA #$7F       directional
0264 8D 41 17         STA SADD                  registers
0267 A0 09            LDY #$09
0269 A2 FA            LDX #$FA       negative 5
026B B5 1E     SHOW   LDA WINDOW,X   light
026D 8D 40 17         STA SAD            display
0270 8C 42 17         STY SBD
0273 C6 11     ST1    DEC MOD
0275 D0 FC            BNE ST1
0277 C8               INY
0278 C8               INY
0279 E8               INX
027A 30 EF            BPL SHOW
027C 20 40 1F         JSR KEYIN
027F 20 6A 1F         JSR GETKEY
0282 C9 13            CMP #$13       GO key?
0284 F0 C5            BEQ SLINK        yes, restart
0286 C9 07            CMP #7         Keys 0 to 6?
0288 B0 C3            BCS WTEST        no, test win
028A AA               TAX            Keys 1 to 6?
028B F0 D5            BEQ PLAY         no, exit
028D CA               DEX            Keys 2 to 6 (=1 to 5)?
028E F0 D2            BEQ PLAY         no, exit
0290 E4 10            CPX POINTR     Same key as before?
0292 F0 CE            BEQ PLAY         yes, ignore
0294 86 10            STX POINTR       no, we've got a live one
0296 B5 18     TOP1   LDA WINDOW,X
0298 48               PHA              roll 'em out...
0299 CA               DEX
029A 10 FA            BPL TOP1
029C A6 10            LDX POINTR
029E 68        TOP2   PLA            roll 'em back in
029F 95 18            STA WINDOW,X
02A1 CA               DEX
02A2 10 FA            BPL TOP2
02A4 30 BC            BMI PLAY
02A6 F7 FC B9 WINNER  .BYTE $F7,$FC,$B9,$DE,$F9,$F1
02A9 DE F9 F1
                ;  end
```

TEASER

By Lew Edwards

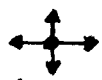
Description -

This program is an adaptation of the "Shooting Stars" game utilizing the keyboard and display of the KIM-1. Originally published in the Sept. '74 issue of PCC, a version also appeared in the May '76 issue of Byte magazine.

The starfield is displayed on the horizontal segments of the second through fourth digits of the display. The segments represent stars when lit and are numbered as follows:

7 8 9
4 5 6
1 2 3

Shooting a star creates a hole where the star was. The resulting "explosion" changes the condition of certain adjacent stars or holes, (stars to holes, or holes to stars) according to the following:

Center (5)  , Sides (2,8)  or (4,6) 

Corners (1)  , (3)  , (7)  , (9) 

The game starts with a star in position 5; the rest are holes. The object of the game is to reverse the initial condition, making 5 a hole and all the rest stars. Eleven moves are the minimum number.

Should you attempt to "shoot" a hole, the first digit displays a "H" until a star key is pressed. This digit also displays a valid number selection. A count of valid moves is given at the right of the display. A win gives a "F" in the first digit. All holes is a losing situation, ("L" in the first digit). You may start over at any time by pressing the "Go" button. The program starts at 0200.

```
0200  A9 00     BEGN    LDA #$00       ZERO REGISTERS D0-DA
0202  A2 10             LDX #$10
0204  95 CF     CLOP    STA 00CF,X
0206  CA                DEX
0207  D0 FB             BNE CLOP
0209  A9 40             LDA #$40       ...INITIALIZE DISPLAY...
020B  85 D4             STA 00DR
020D  A9 10             LDA #$10       INIT. STARFIELD
020F  85 DE             STA 00DE       REGISTERS
0211  4A                LSR
0212  85 DF             STA 00DF
0214  20 DD 02  MLOP    JSR DISP       ...DISPLAY...
0217  A6 D3             LDX 00D3       MODE?
0219  D0 50             BNE DELA       MODE=1, DELAY AND UPDATE
021B  20 40 1F          JSR 1F40       MODE=0, GET KEY
021E  F0 F4             BEQ MLOP       NO KEY, RETURN
0220  20 40 1F          JSR 1F40       KEY STILL PRESSED?
0223  F0 EF             BEQ MLOP       NO, RETURN
0225  20 6A 1F          JSR GETKEY     YES, GET KEY
0228  C9 13             CMP #$13       "GO" KEY?
022A  F0 D4             BEQ BEGN       YES, START AGAIN
022C  C9 0A             CMP #$0A       OVER 9?
022E  10 E4             BPL MLOP       YES, TRY AGAIN
0230  A8                TAY            USE AS INDEX
```

```
0231  F0 E1          BEQ MLOP       0? - NOT VALID
0233  85 D1          STA 00D1       1-9 STORE IT
0235  20 F4 02       JSR SEG        CONVERT TO SEGMENTS
0238  85 D0          STA 00D0       DISPLAY - LEFT DIGIT
023A  B9 CA 02       LDA 02CA,Y     GET STAR TEST BIT
023D  C0 06          CMP #$06       TEST KEY #
023F  30 06          BMI SKIP       1-5, SKIP
0241  24 DF          BIT 00DF       6-9, TEST HI FIELD
0243  D0 0C          BNE STAR       IT'S A STAR
0245  F0 04          BEQ HOLE       IT'S A HOLE

0247  24 DE    SKIP  BIT 00DE       1 TO 5, TEST LO FIELD
0249  D0 06          BNE STAR       IT'S A STAR
024B  A9 76    HOLE  LDA #$76       IT'S A HOLE LOAD "H"
024D  85 D0          STA 00D0       DISPLAY-LEFT DIGIT
024F  D0 C3          BNE MLOP       UNCOND. JUMP
0251  F8      STAR  SED            UPDATE COUNT
0252  38            SEC
0253  A9 00          LDA #$00
0255  65 D5          ADC 00D5       BY ADDING ONE
0257  85 D5          STA 00D5       STORE IT
0259  D8            CLD
025A  20 F4 02       JSR SEG        UNPACK, CONVERT
025D  85 DA          STA 00DA       TO SEGMENTS AND
025F  A5 D5          LDA 00D5       DISPLAY IN DIGITS
0261  20 F0 02       JSR LEFT       5 AND 6...
0264  85 D8          STA 00D8
0266  E6 D3          INC 00D3       SET MODE TO 1
0268  4C 14 02       JMP MLOP       MAIN LOOP AGAIN
026B  A0 00    DELA  LDY #$00       MODE = 1
026D  20 DD 02       JSR DISP       DELAY ABOUT .8 SEC
0270  88             DEY            WHILE DISPLAYING
0271  D0 FA          BNE DELA
0273  A6 D1           LDX 00D1       KEY # AS INDEX
0275  BD D3 02       LDA 02D3,X     GET SHOT PATTERN
0278  A8            TAY            SAVE IN Y REGISTER
0279  E0 06          CPX #$06       KEY # OVER 5?
027B  30 08          BMI LOWF       NO, GO TO LOW FIELD
027D  45 DF          EOR 00DF       UPDATE HI FIELD, 6-9
027F  85 DF          STA 00DF
0281  98             TYA            RECALL PATTERN, 6-9
0282  A0 00          LDY #$00       NO SHOT 3RD TIME
0284  0A             ASL A          ALIGN WITH LO FIELD
0285  45 DE    LOWF  EOR 00DE       UPDATE LO FIELD
0287  85 DE          STA 00DE
0289  98             TYA            RECALL PATTERN, 1-5
028A  4A             LSR A          ALIGN WITH HI FIELD
028B  45 DF          EOR 00DF       UPDATE HI FIELD, 1-5
028D  85 DF          STA 00DF       (BLANK SHOT IF 6-9)
028F  0A             ASL A          SHIFT 9 TO CARRY
0290  A5 DE          LDA 00DE       GET REST OF FIELD
0292  A2 06          LDX #$06       ...STAR DISPLAY...
```

```
0294  2A        DLOP    ROL             ALIGN WITH DISPLAY
0295  48                PHA             SAVE IT FOR NEXT TIME
0296  29 49             AND #$49        MASK TO HORIZ. SEGS
0298  95 D0             STA 00D0,X      INTO DISPLAY WINDOW
029A  68                PLA             RECALL FIELD
029B  CA                DEX             SHIFT TO NEXT
029C  CA                DEX             DISPLAY DIGIT
029D  D0 F5             BNE DLOP        REPEAT TILL DONE
029F  2A                ROL             BIT FOR 5 TO CARRY
02A0  B0 0E             BCS MODE        5 IS STAR, CONTINUE
02A2  F0 08             BEQ LOSE        5 IS HOLE, ALL HOLES
02A4  C9 FF             CMP #$FF        ALL THE REST STARS?
02A6  D0 08             BNE MODE        NO
02A8  A9 71             LDA #$71        YES, LOAD "F"
02AA  D0 08             BNE FRST        AND SKIP
02AC  A9 38       LOSE  LDA #$38        LOAD "L", (LOSE)
02AE  D0 04             BNE FRST        AND SKIP
02B0  C6 D3       MODE  DEC 00D3        SET MODE TO 0
02B2  A9 00             LDA #$00        BLANK FIRST DIGIT
02B4  85 D0       FRST  STA 00D0        FILL FIRST DIGIT
02B6  D0 03             BNE NONE        END OF GAME
02B8  4C 14 02          JMP MLOP        MAIN LOOP AGAIN
02BB  20 DD 02    DONE  JSR DISP        DISPLAY UNTIL
02BE  20 40 1F          JSR 1F40        "GO" KEY IS
02C1  20 6A 1F          JSR GETKEY      PUSHED
02C4  C9 13             CMP #$13
02C6  D0 F3             BNE DONE
02C8  4C 00 02          JMP BEGN        START A NEW GAME
02CB  01 02 04 08 10 10 20 40 80 1B 07 36 49 BA 92 6C
02DB  E0 D8

                  **** DISPLAY SUBROUTINE ****
02DD  A9 7F       DISP  LDA #$7F        TURN ON DISPLAY
02DF  8D 41 17          STA 1741
02E2  A2 09             LDX #$09
02E4  B5 C7       MORE  LDA 00C7,X      PUT IN SEGMENTS
02E6  84 FC             STY 00FC        SAVE Y
02E8  20 4E 1F          JSR 1F4E        DISPLAY THEM
02EB  E0 15             CPX #$15        DONE? 6 TIMES
02ED  D0 F5             BNE MORE        NO, LOOP
02EF  60                RTS             YES, RETURN

            **** HEX CONVERSION SUBROUTINE ****
02F0  4A          LEFT  LSR A
02F1  4A                LSR A
02F2  4A                LSR A
02F3  4A                LSR A
02F4  29 0F       SEG   AND #$0F        MASK TO 4 BITS
02F6  A8                TAY             USE AS INDEX
02F7  B9 E7 1F          LDA 1FE7,Y      CONVERT TO SEGMENTS
02FA  60                RTS             RETURN
```

TIMER

By Joel Swank

Description -

TIMER turns KIM into a digital stopwatch showing up to 99 minutes and 59.99 seconds. It is designed to be accurate to 50 microseconds per second. The interval timer is used to count 9984 cycles and the instructions between the time out and the reset of the timer make up the other 16 cycles in .01 seconds. The keyboard is used to control the routine as follows: Stop (0), Go (1), Return to KIM (4), Reset (2).

```
0200  A9 00     BEGN    LDA #$00
0202  85 F9             STA INH       ZERO DISPLAY
0204  85 FA             STA POINTL
0206  85 FB             STA POINTH
0208  20 1F 1F  HOLD    JSR SCANDS    LIGHT DISPLAY
020B  20 6A 1F           JSR GETKEY
020E  C9 04              CMP #$04      KEY 4?
0210  D0 03              BNE CONT
0212  4C 64 1C           JMP 1C64      RETURN TO KIM
0215  C9 02      CONT    CMP #$02      KEY 2?
0217  F0 E7              BEQ BEGN      BACK TO ZERO
0219  C9 01              CMP #$01      KEY 1?
021B  D0 EB              BNE HOLD
021D  A9 9C              LDA #$9C
021F  8D 06 17           STA 1706      SET TIMER
0222  20 1F 1F  DISP    JSR SCANDS    DISPLAY VALUE
0225  AD 07 17  CLCK    LDA 1707      CHECK TIMER
0228  F0 FB              BEQ CLCK
022A  8D 00 1C           STA ROM       DELAY 4 MICROSEC.
022D  A9 9C              LDA #$9C      SET TIMER
022F  8D 06 17           STA 1706
0232  18                 CLC
0233  F8                 SED           SET FLAGS
0234  A5 F9              LDA INH
0236  69 01              ADC #$01      INC. 100THS
0238  85 F9              STA INH
023A  A5 FA              LDA POINTL
023C  69 00              ADC #$00      INC. SECONDS
023E  85 FA              STA POINTL
0240  C9 60              CMP #$60      STOP AT 60
0242  D0 0B              BNE CKEY
0244  A9 00              LDA #$00
0246  85 FA              STA POINTL    ZERO SECONDS
0248  A5 FB              LDA POINTH
024A  18                 CLC
024B  69 01              ADC #$01      INC. MINUTES
024D  85 FB              STA POINTH
024F  D8         CKEY    CLD
0250  20 6A 1F           JSR GETKEY    READ KEYBOARD
0253  C9 00              CMP #$00      KEY 0?
0255  D0 CB              BNE DISP
0257  F0 AF              BEQ HOLD      STOP
```

WUMPUS

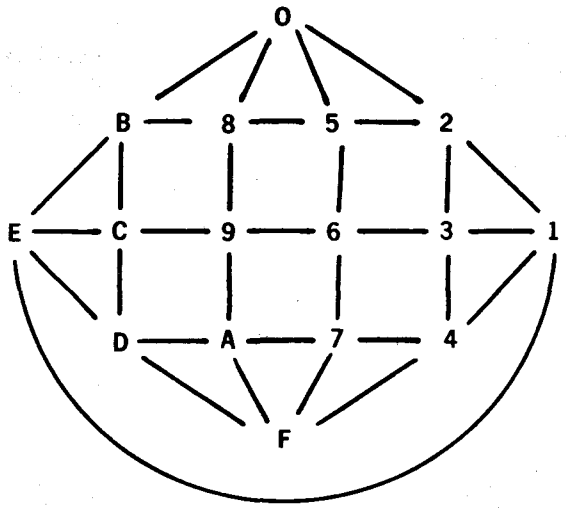
By Stan Ockers

Description -

Wumpus lives in a cave of 16 rooms (labeled 0-F). Each room has four tunnels leading to other rooms (see the figure). When the program is started at 0305, you and Wumpus are placed at random in the rooms. Also placed at random are two bottomless pits (they don't bother Wumpus, he has sucker-type feet) and two rooms with Superbats (also no trouble to Wumpus, he's too heavy). If you enter a bat's room you are picked up and flown at random to another room. You will be warned when bats, pits or Wumpus are nearby. If you enter the room with Wumpus, he wakes and either moves to an adjacent room or just eats you up (you lose). In order to capture Wumpus, you have three cans of "mood change" gas. When thrown into a room containing Wumpus, the gas causes him to turn from a vicious snarling beast into a meek and loveable creature. He will even come out and give you a hug. Beware though, once you toss a can of gas in the room, it is contaminated and you cannot enter or the gas will turn you into a beast (you lose).

If you lose and want everything to stay the same for another try, start at 0316. The byte at 0229 controls the speed of the display. Once you get used to the characters, you can speed things up by putting in a lower number. The message normally given tells you what room you are in and what the choices are for the next room. In order to fire the mood gas, press PC (pitch can?), when the rooms to be selected are displayed. Then indicate the room into which you want to pitch the can. It takes a fresh can to get Wumpus (he may move into a room already gassed) and he will hear you and change rooms whenever a can is tossed (unless you get him). If Wumpus moves into a room with a pit or Superbats, he'll be hidden - you won't be told WUMPUS CLOSE. Either guess, or pitch a can to make him move. Good hunting.

The program is adapted from a game by Gregory Yob which appears in The Best of Creative Computing.



```
0305  A9 FF          LDA #$FF       ...INITIALIZATION...
0307  A2 0E          LDX #$0E       ..CLEAN OUT ROOMS..
0309  95 C1   INIT  STA 00C1,X     INIT. TO FF
030B  CA             DEX            FINISHED?
030C  10 FB          BPL INIT       NO
030E  A9 03          LDA #$03       GIVE THREE CANS OF GAS
0310  85 E0          STA 00E0
0312  A0 05          LDY #$05       ...RANDOMIZE...
0314  10 02          BPL GETN       YOU,WUMPUS,PITS AND BATS
0316  A0 00          LDY #$00       (ONLY YOU ENTRY)
0318  A2 05   GETN  LDX #$05
031A  20 72 02       JSR RAND
031D  29 0F          AND #$0F
031F  D5 CA     CKNO CMP 00CA,X     ..MAKING SURE ALL
0321  F0 F5          BEQ GETN       ARE DIFFERENT..
0323  CA             DEX
0324  10 F9          BPL CKNO
0326  99 CA 00       STA 00CA,Y     STORE IN 00CA-00CF
0329  88             DEY
032A  10 EC          BPL GETN
032C  20 B2 02  ADJR  JSR NXTR       SET UP ADJACENT ROOM LIST
032F  A0 03          LDY #$03       HAZARDS IN ADJ. ROOMS?
0331  84 E1          STY 00E1
0333  B9 C6 00  NXTR  LDA 00C6,Y
0336  20 8F 02       JSR COMP       COMPARE EACH TO HAZARDS
0339  8A             TXA            (X CONTAINS MATCH INFO.)
033A  30 17          BMI NOMA       NO MATCH, NO HAZARDS
033C  E0 03          CPX #$03       BATS?
033E  30 04          BMI SKP1       NO
0340  A9 19          LDA #$19       (BATS NEARBY MESSAGE)
0342  10 0A          BPL MESS
0344  E0 01   SKP1  CPX #$01       PIT?
0346  30 04          BMI SKP2       NO
0348  A9 0E          LDA #$0E       (PIT CLOSE MESSAGE)
034A  10 02          BPL MESS
034C  A9 00   SKP2  LDA #$00       MUST BE WUMPUS
034E  A0 01   MESS  LDY #$01       (PAGE ONE)
0350  20 00 02       JSR SCAN       DISPLAY HAZARD MESSAGE
0353  C6 E1     NOMA DEC 00E1       TRY NEXT ADJ. ROOM
0355  A4 E1          LDY 00E1       FINISHED?
0357  10 DA          BPL NXTR       NO
0359  A4 CA          LDY 00CA       ..LOAD AND DISPLAY -
035B  B9 E7 1F       LDA 1FE7,Y     "YOU ARE IN ... TUNNELS
035E  85 0C          STA 000C       LEAD TO ...." MESSAGE..
0260  A2 03          LDX #$03       (FOUR NEXT ROOMS)
0362  B4 C6     XRO  LDY 00C6,X
0364  B9 E7 1F       LDA 1FE7,Y     CONVERSION
0367  95 20          STA 0020,X     PUT IN MESSAGE
0369  CA             DEX            FINISHED?
036A  10 F6          BPL XRO        NO
036C  A0 00   ROOM  LDY #$00       LOCATION AND..
036E  98             TYA            PAGE OF MESSAGE
036F  20 00 02       JSR SCAN       DISPLAY MESSAGE
```

```
0372  20 58 02          JSR DEBO      DEBOUNCE KEY
0375  C9 14             CMP #$14      PC PUSHED?
0377  F0 48             BEQ ROOM      YES
0379  20 C5 02          JSR VALID     AN ADJACENT ROOM?
037C  85 CA             STA 00CA      UPDATE YOUR ROOM
037E  8A                TXA
037F  30 EB             BMI ROOMS     IF X=FF, NOT VALID ROOM
0381  A5 CA             LDA 00CA      CHECK FOR GAS IN ROOM
0383  A2 04             LDX #$04      5 POSSIBLE (EXPANSION)
0385  D5 C1       NXTG  CMP 00C1,X
0387  F0 33             BEQ GASM      GASSED!!
0389  CA                DEX           ALL CHECKED?
038A  10 F9             BPL NXTG      NO
038C  20 8F 02          JSR COMP      CHECK YOUR NEW
038F  8A                TXA           ROOM FOR HAZARDS..
0390  30 9A              BMI ADJR      NO MATCH, NO HAZARDS
0392  E0 03             CPX #$03
0394  10 17             BPL BATM      BATS
0396  E0 01             CPX #$01
0398  10 1D             BPL PITM      PIT!!!
039A  A0 00             LDY #$00
039C  A9 26             LDA #$26      MUST HAVE BUMPED WUMPUS
039E  20 00 02          JSR SCAN      DISPLAY MESSAGE
03A1  20 99 02          JSR MOVE      ..SEE IF HE MOVES..
03A4  C5 CA             CMP 00CA      STILL IN YOUR ROOM?
03A6  D0 84             BNE ADJR      NO, YOU'RE O.K.
03A8  A9 26             LDA #$26      HE GOT YOU!
03AA  4C CF 02          JMP LOSE
03AD  A0 01       BATM  LDY #$01      BAT MESSAGE
03AF  A9 3D             LDA #$3D
03B1  20 00 02          JSR SCAN
03B4  4C 16 03          JMP CHNG      CHANGE YOUR ROOM
03B7  A9 4F       PITM  LDA #$4F      FELL IN PIT!
03B9  4C CF 02          JSR LOSE
03BC  A9 65       GASM  LDA #$65      GAS IN ROOM!
03BE  4C CF 02          JMP LOSE
03C1  A0 00       ROOM  LDY #$00      PITCH CAN AND SEE..
03C3  A9 B7             LDA #$B7      IF YOU GET HIM
03C5  20 00 02          JSR SCAN      ROOM?
03C8  20 58 02          JSR DEBO
03CB  20 C5 02          JSR VALID     VALID ROOM?
03CE  85 D1             STA 00D1
03D0  8A                TXA
03D1  30 EE             BMI ROOM      IF X=FF, NOT VALID
03D3  A5 D1             LDA 00D1
03D5  A6 E0             LDX 00E0      CANS OF GAS LEFT
03D7  95 C0             STA 00C0,X    ..IS WUMPUS IN
03D9  C5 CB             CMP 00CB      ROOM GASSED?
03DB  F0 15             BEQ WIN       YES, YOU GOT HIM
03DD  C6 E0             DEC 00E0      DECREASE CAN COUNT
03DF  F0 1A             BEQ OUT       GAS IS GONE
03E1  A6 CB             LDX 00CB      ..MOVE WUMPUS TO AN
03E3  20 B4 02          JSR NEXT      ADJACENT ROOM (FOR HIM)
03E6  20 A5 02          JSR MOVE
```

```
03E9  C5 CA              CMP 00CA      DID HE MOVE INTO YOUR ROOM?
03EB  F0 BB              BEQ 03A8      YES
03ED  4C DE 02           JMP 02DE      DISPLAY CANS LEFT MESSAGE
03F2  A0 01              LDY #$01      GREAT& ETC. MESSAGE
03F4  A9 80              LDA #$80
03F6  20 00 02           JSR SCAN
03F9  F0 F7              BEQ WIN       REPEAT
03FB  A9 73      OUT     LDA #$73      OUT OF GAS!
03FD  4C CF 02           JMP LOSE


0200  84 DE              STY 00DE      TRANSFER POINTER HIGH
0202  85 DD              STA 00DD      TRANSFER POINTER LOW
0204  A9 07              LDA #$07      INIT. SCAN FORWARD
0206  85 DF              STA 00DF
0208  A0 05              LDY #$05      INIT Y
020A  A2 05      CONT    LDX #$05      INIT X
020C  B1 DD      CHAR    LDA (00DD),Y  GET CHARACTER
020E  C9 00              CMP #$00      LAST CHARACTER?
0210  D0 01              BNE MORE      IF NOT, CONTINUE
0212  60                RTS
0213  95 E8              STA 00E8,X    STORE IT
0215  88        MORE    DEY           SET UP NEXT CHARACTER
0216  CA                DEX           SET UP NEXT STORE LOC.
0217  10 F3              BPL CHAR      LOOP IF NOT 6TH CHAR.
0219  D8                CLD           BINARY MODE
021A  18                CLC           PREPARE TO ADD
021B  98                TYA           GET CHAR. POINTER
021C  65 DF              ADC 00DF      UPDATE FOR 6 NEW CHAR.
021E  85 DC              STA 00DC      SAVE NEW POINTER
0220  20 28 02           JSR 0228      DELAY-DISPLAY
0223  A4 DC              LDY 00DC      RESTORE POINTER
0225  4C 0A 02           JMP CONT      CONTINUE REST OF MESSAGE
          ***** DELAY DISPLAY SUBROUTINE *****
0228  A2 0A              LDX #$0A      SET RATE
022A  86 DB              STX 00DB      PUT IN DECR. LOC.
022C  A9 52      TIME    LDA #$52      LOAD TIMER
022E  8D 07 17           STA 1707      START TIMER
0231  20 3E 02           JSR DISP      JUMP TO DISPLAY SUBR.
0234  2C 07 17           BIT 1707      TIMER DONE?
0237  10 F8              BPL LITE      IF NOT, LOOP
0239  C6 DB              DEC 00DB      DECREMENT TIMER
023B  D0 EF              BNE TIME      NOT FINISHED
023D  60                 RTS           GET 6 NEW CHAR.
      ***** BASIC DISPLAY SUBROUTINE *****
023E  A9 7F              LDA #$7F      CHANGE SEGMENTS..
0240  8D 41 17           STA PADD      TO OUTPUT
0243  A0 00              LDY #$00      INIT. RECALL INDEX
0245  A2 09              LDX #$09      INIT. DIGIT NUMBER
0247  B9 E8 00  SIX      LDA 00E8,Y    GET CHARACTER
024A  84 FC              STY 00FC      SAVE Y
024C  20 4E 1F           JSR 1F4E      DISPLAY CHARACTER
```

```
024F  C8                      INY             SET UP FOR NEXT CHAR.
0250  C0 06                   CPY #$06        6 CHAR. DISPLAYED?
0252  90 F3                   BCC SIX         NO
0254  20 3D 1F                JSR 1F3D        KEY DOWN?
0257  60                      RTS             EXIT
                    ***** DEBOUNCE SUBROUTINE *****
0258  20 8C 1E  DEBO          JSR INIT1
025B  20 3E 02               JSR DISP        WAIT FOR PREVIOUS KEY
025E  D0 F8                  BNE DEBO        TO BE RELEASED
0260  20 3E 02  SHOW          JSR DISP        WAIT FOR NEW KEY TO
0263  F0 FB                  BEQ SHOW        BE DEPRESSED
0265  20 3E 02               JSR DISP        CHECK AGAIN AFTER
0268  F0 F6                  BEQ SHOW        SLIGHT DELAY
026A  20 6A 1F               JSR GETKEY      GET A KEY
026D  C9 15                  CMP #$15        A VALID KEY?
026F  10 E7                  BPL DEBO        NO
0271  60                      RTS
              ***** RANDOM NUMBER SUBROUTINE *****
0272  8A          RAND       TXA             SAVE X REGISTER
0273  48                     PHA
0274  D8                     CLD             RANDOM # ROUTINE FROM
0275  38                     SEC             J. BUTTERFIELD, KIM
0276  A5 41                  LDA 0041        USER NOTES #1 PAGE 4
0278  65 44                  ADC 0044
027A  65 45                  ADC 0045
027C  85 40                  STA 0040
027E  A2 04                  LDX #$04
0280  B5 40       NXTN       LDA 0040,X
0282  95 41                  STA 0041,X
0284  CA                     DEX
0285  10 F9                  BPL NXTN
0287  85 C0                  STA 00C0
0289  68                     PLA             RETURN X REGISTER'
028A  AA                     TAX
028B  A5 C0                  LDA 00C0
028D  60                     RTS
                  ***** COMPARE SUBROUTINE *****
028F  A2 04       COMP       LDX #$04        COMPARE ROOM IN ACC.
0291  D5 CB       HAZD       CMP 00CB,X      WITH EACH HAZARD.
0293  F0 03                  BEQ OUT
0295  CA                     DEX
0296  10 F9                  BPL HAZD        X ON EXIT SHOWS MATCH
0298  60          OUT        RTS
                ***** MOVE WUMPUS SUBROUTINE *****
0299  20 72 02    MOVE       JSR RAND        GET A RANDOM #
029C  29 0F                  AND #$0F        STRIP TO HEX DIGIT
029E  C9 04                  CMP #$04        CHANGE ROOMS 75%
02A0  30 0D                  BMI NOCH        OF THE TIME
02A2  20 B2 02               JSR NEXT        GET ADJ. ROOMS (TO WUMPUS)
02A5  AD 06 17               LDA 1706        GET RANDOM #, 0-3
02A8  29 03                  AND #$03
02AA  AA                     TAX             USE AS INDEX
02AB  B5 C6                  LDA 00C6,X      GET AN ADJ. ROOM
02AD  85 CB                  STA 00CB        PUT WUMPUS IN IT
```

```
02AF  A5 CB     NOCH    LDA 00CB        WUMPUS ROOM IN ACC.
02B1  60                RTS
        ***** LOAD NEXT ROOMS SUBROUTINE *****
02B2  A6 CA             LDX 00CA        YOUR ROOM AS INDEX
02B4  B5 50             LDA 0050,X      ... NEXT ROOMS ARE LOADED
02B6  85 C6             STA 00C6        INTO 00C6-00C9 FROM
02B8  B5 60             LDA 0060,X      TABLES ...
02BA  85 C7             STA 00C7
02BC  B5 70             LDA 0070,X
02BE  85 C8             STA 00C8
02C0  B5 80             LDA 0080,X
02C2  85 C9             STA 00C9
02C4  60                RTS
              ***** CHECK VALID SUBROUTINE *****
02C5  A2 03     VALID   LDX #$03        ... CHECK IF ACC.
02C7  D5 C6             CMP 00C6,X      MATCHS 00C6-00C9 ...
02C9  F0 03             BEQ YVAL        YES, VALID ROOM
02CB  CA                DEX
02CC  10 F9             BPL NXTV
02CE  60        YVAL    RTS
            ***** LOSE SUBROUTINE *****
02CF  A0 01     LOSE    LDY #$01        ...DISPLAY REASON LOST,
02D1  20 00 02          JSR SCAN        THEN "YOU LOSE" ...
02D4  A0 00             LDY #$00
02D6  A9 AC             LDA #$AC
02D8  20 00 02          JSR SCAN
02DB  4C D4 02          JMP REPT
          ***** GAS LEFT MESSAGE *****
02DE  A4 E0             LDY 00E0        GET CANS LEFT
02E0  B9 E7 1F          LDA 1FE7,Y      GET CONVERSION
02E3  85 9F             STA 09F         STORE IN MESSAGE
02E5  A0 00             LDY #$00        (PAGE ZERO)
02E7  A9 90             LDA #$90        DISPLAY CANS OF GAS
02E9  20 00 02          JSR SCAN        LEFT MESSAGE
02EC  4C 2C 03          JMP ADJR
```

***** Messages *****

```
0000 80 EE DC BE 80 F7 D0 F9 80 84 D4 80 EF 80 C0 80
0010 F8 BE D4 D4 F9 B8 ED 80 B8 F9 F7 DE 80 F8 DC 80
0020 FD FF F7 B9 80 00 80 DC DC F3 ED 80 C0 80 FC BE
0030 B7 F3 F9 DE 80 F7 80 9C BE B7 F3 BE ED 80 80 00
```

***** Next Room List *****

```
0050 02 02 00 01 01 00 03 04 00 06 07 00 09 0A 01 04
0060 05 03 01 02 03 02 05 06 05 08 09 08 0B 0C 0B 07
0070 08 04 03 04 07 06 07 0A 09 0A 0F 0C 0D 0E 0C 0A
0080 0B 0E 05 06 0F 08 09 0F 0B 0C 0D 0E 0E 0F 0D 0D
```

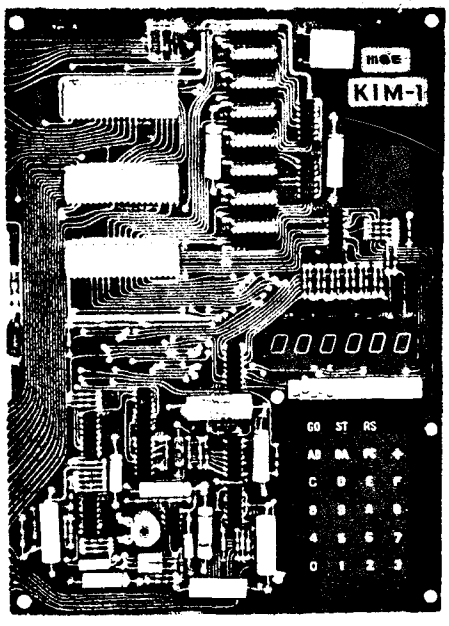
***** Messages *****

```
0090 80 B7 84 ED ED F9 DE 80 C0 80 DC D4 B8 EE 80 DB
00A0 80 B9 F7 D4 ED 80 B8 F9 F1 F8 80 00 80 EE DC BE
00B0 80 B8 DC ED F9 80 00 80 D0 DC DC B7 D3 80 00 03

0100 80 9C BE B7 F3 BE ED 80 B9 B8 DC ED F9 00 80 F3
0110 84 F8 80 B9 B8 DC ED F9 00 80 FC F7 F8 ED 80 B9
0120 B8 DC ED F9 80 00 80 F6 F7 80 F6 F7 80 9C BE B7
0130 F3 BE ED 80 BD DC F8 80 EE DC BE 80 00 80 ED BE
0140 F3 F9 D0 FC F7 F8 80 ED D4 F7 F8 B9 F6 80 00 80
0150 EE EE 84 84 F9 F9 F9 80 F1 F9 B8 B8 80 84 D4 80
0160 F3 84 F8 80 00 80 BD F7 ED 80 84 D4 80 D0 DC DC
0170 B7 80 00 80 DC BE F8 80 DC F1 80 BD F7 ED 80 00
0180 80 80 80 80 80 BD D0 F9 F7 F8 C0 80 EE DC BE 80
0190 BD F9 F8 80 F7 80 F6 BE BD 80 F1 D0 DC B7 80 9C
01A0 BE B7 F3 BE ED 80 00
```

***** Hex Dump - Main Program *****
Wumpus

```
0200 84 DE 85 DD A9 07 85 DF A0 05 A2 05 B1 DD C9 00
0210 D0 01 60 95 E8 88 CA 10 F3 D8 18 98 65 DF 85 DC
0220 20 28 02 A4 DC 4C 0A 02 A2 03 86 DB A9 52 8D 07
0230 17 20 3E 02 2C 07 17 10 F8 C6 DB D0 EF 60 A9 7F
0240 8D 41 17 A0 00 A2 09 B9 E8 00 84 FC 20 4E 1F C8
0250 C0 06 90 F3 20 3D 1F 60 20 8C 1E 20 3E 02 D0 F8
0260 20 3E 02 F0 FB 20 3E 02 F0 F6 20 6A 1F C9 15 10
0270 E7 60 8A 48 D8 38 A5 41 65 44 65 45 85 40 A2 04
0280 B5 40 95 41 CA 10 F9 85 C0 68 AA A5 C0 60 60 A2
0290 04 D5 CB F0 03 CA 10 F9 60 20 72 02 29 0F C9 04
02A0 30 0D 20 B2 02 AD 06 17 29 03 AA B5 C6 85 CB A5
02B0 CB 60 A6 CA B5 50 85 C6 B5 60 85 C7 B5 70 85 C8
02C0 B5 80 85 C9 60 A2 03 D5 C6 F0 03 CA 10 F9 60 A0
02D0 01 20 00 02 A0 00 A9 AC 20 00 02 4C D4 02 A4 E0
02E0 B9 E7 1F 85 0F A0 00 A9 90 20 00 02 4C 2C 03 F6
02F0 BE BD 80 F1 D0 DC B7 80 9C BE B7 F3 BE ED 80 00

0300 EA EA EA EA EA A9 FF A2 0E 95 C1 CA 10 FB A9 03
0310 85 E0 A0 05 10 02 A0 00 A2 05 20 72 02 29 0F D5
0320 CA F0 F5 CA 10 F9 99 CA 00 88 10 EC 20 B2 02 A0
0330 03 84 E1 B9 C6 00 20 8F 02 8A 30 17 E0 03 30 04
0340 A9 19 10 0A E0 01 30 04 A9 0E 10 02 A9 00 A0 01
0350 20 00 02 C6 E1 A4 E1 10 DA A4 CA B9 E7 1F 85 0C
0360 A2 03 B4 C6 B9 E7 1F 95 20 CA 10 F6 A0 00 98 20
0370 00 02 20 58 02 C9 14 F0 48 20 C5 02 85 CA 8A 30
0380 EB A5 CA A2 04 D5 C1 F0 33 CA 10 F9 20 8F 02 8A
0390 30 9A E0 03 10 17 E0 01 10 1D A0 00 A9 26 20 00
03A0 02 20 99 02 C5 CA D0 84 A9 26 4C CF 02 A0 01 A9
03B0 3D 20 00 02 4C 16 03 A9 4F 4C CF 02 A9 65 4C CF
03C0 02 A0 00 A9 B7 20 00 02 20 58 02 20 C5 02 85 D1
03D0 8A 30 EE A5 D1 A6 E0 95 C0 C5 CB F0 15 C6 E0 F0
03E0 1A A6 CB 20 B4 02 20 A5 02 C5 CA F0 BB 4C DE 02
03F0 EA EA A0 01 A9 80 20 00 02 F0 F7 A9 73 20 CF 02
```

DIAGNOSTIC AND UTILITY PROGRAMS



BRANCH

BY JIM BUTTERFIELD

Load this fully relocatable program anywhere. Once it starts, key in the last two digits of a branch instruction address; then the last two digits of the address to which you are branching; and read off the relative branch address.

For example, to calculate the branch to ADDR near the end of this program: hit 26 (from 0026); 20 (to 0020) and read F8 on the two right hand digits of the display.

The program must be stopped with the RS key.

```
0000 D8       START CLD
0001 18             CLC
0002 A5 FA          LDA POINTL
0004 E5 FB          SBC POINTH
0006 85 F9          STA INH
0008 C6 F9          DEC INH
000A 20 1F 1F       JSR SCANDS
000D 20 6A 1F       JSR GETKEY
0010 C5 F3          CMP LAST
0012 F0 EC          BEQ START
0014 85 F3          STA LAST
0016 C9 10          CMP #$10
0018 B0 E6          BCS START
001A 0A             ASL A
001B 0A             ASL A
001C 0A             ASL A
001D 0A             ASL A
001E A2 04          LDX #4
0020 0A       ADDR  ASL A
0021 26 FA          ROL POINTL
0023 26 FB          ROL POINTH
0025 CA             DEX
0026 D0 F8          BNE ADDR
0028 F0 D6          BEQ START
```

Keep in mind that the maximum "reach" of a branch instruction is 127 locations forward (7F) or 128 locations backward (80). If you want a forward branch, check that the calculated branch is in the range 01 to 7F. Similarly, be sure that a backward branch produces a value from 80 to FE. In either case, a value outside these limits means that your desired branch is out of reach.

BROWSE

Jim Butterfield

Load BROWSE anywhere in memory - it's fully relocatable - start it up, and presto! It doesn't seem to do anything.

BROWSE is a mini-Monitor that performs most of the functions of the regular KIM monitor; but you'll find it handy for entering and proof-reading programs. Most of the keys work the same as usual; but PC, +, and DA are slightly different.

When you hit + you go to the next address as usual .. but then you keep on going! Great for proofreading a program you've just entered. It lets you browse through memory.

Hit PC and the program steps backwards, so you can look at a value you've just passed. All other keys instantly freeze the browsing process; you can hit AD or DA to stop on a given address, or just enter a new address if you wish.

Key DA operates a little differently from the regular KIM function. To enter data, first set up the address before the one you want to change. As you enter the data, BROWSE will automatically step forward to the next address - and then the next one, and so on. You never need to hit the + key during entry; and the display will show the last value you have entered.

```
0110 D8       START  CLD          clear decimal mode
0111 A9 13           LDA #$13     GO key image
0113 85 FE           STA CHAR
0115 A9 00           LDA #0       value zero..
0117 85 FA           STA POINTL   ..to address pointer
0119 85 FB           STA POINTH
011B C6 F3    LOOP   DEC WAIT     main program loop
011D D0 0E           BNE LP1      pause 1 second
011F A5 FD           LDA TMPX     up or down?
0121 F0 0A           BEQ LP1      neither
0123 10 69           BPL UP
0125 A5 FA           LDA POINTL   down, decrement
0127 D0 02           BNE DOWN     next page?
0129 C6 FB           DEC POINTH
012B C6 FA    DOWN   DEC POINTL
012D 20 19 1F LP1    JSR SCAND    light display
0130 20 6A 1F        JSR GETKEY   check keys
0133 C5 FE           CMP CHAR     same key as last time?
0135 F0 E4           BEQ LOOP
0137 85 FE           STA CHAR     note new key input
0139 C9 15           CMP #$15     no key?
013B F0 DE           BEQ LOOP     yes, skip
013D A2 00           LDX #0
013F 86 FD           STX TMPX     clear up/down flag
```

```
0141 C9 10          CMP #$10    numeric?
0143 90 1C          BCC NUM     yes, branch
0145 86 F4          STX DIGIT
0147 C9 11          CMP #$11    DA?
0149 F0 01          BEQ OVER    yes, leave X=0
014B E8             INX         no, set X=1
014C 86 FF    OVER  STX MODE    0 or 1 into MODE
014E C9 12          CMP #$12    +?
0150 D0 02          BNE PASS    no, skip
0152 E6 FD          INC TMPX    yes, set browse
0154 C9 14    PASS  CMP #$14    PC?
0156 D0 02          BNE PASS2   no, skip
0158 C6 FD          DEC TMPX    yes, down-browse
015A C9 13    PASS2 CMP #$13    GO?
015C D0 CF          BNE LP1     no, loop
015E 4C C8 1D       JMP GOEXEC  start program
                ; numeric (hex) entry comes here
0161 0A 0A    NUM   ASLA ASLA   position digit
0163 0A 0A          ASLA ASLA   to left
0165 85 FC          STA TEMP
0167 A2 04          LDX #4      4 bits to move
0169 A4 FF          LDY MODE    AD or DA?
016B D0 17          BNE ADDR    branch if AD mode
016D C6 F4          DEC DIGIT   time to step?
016F 10 07          BPL SAME    no, skip
0171 20 63 1F       JSR INCPT   yes, step
0174 E6 F4          INC DIGIT   ..and restore
0176 E6 F4          INC DIGIT         ..digit count
0178 B1 FA    SAME  LDA (POINTL),Y   get data
017A 06 FC    DADA  ASL TEMP        move a bit..
017C 2A             ROL A           ..into data
017D 91 FA          STA (POINTL),Y
017F CA             DEX
0180 D0 F8          BNE DADA        last bit?
0182 F0 A9          BEQ LP1         yes, exit
0184 0A       ADDR  ASL A         move bits
0185 26 FA          ROL POINTL       into address
0187 26 FB          ROL POINTH
0189 CA             DEX
018A D0 F8          BNE ADDR
018C F0 9F          BEQ LP1
                ; increment address for browsing
018E 20 63 1F UP    JSR INCPT
0191 AA             TAX
0192 10 99          BPL LP1
0194          end
```

0H0 — 6197

DIRECTORY

Jim Butterfield

Ever thought about the best way to organize your programs on tape? I used to call the first program on each tape number 01, the next 02, etc. Mostly I was afraid of forgetting the ID number and having trouble reading it in. Program DIRECTORY (below) fixes up that part of the problem and liberates you to choose a better numbering scheme.

You've got 254 program IDs to choose from ... enough for most program libraries with some to spare.

So every program and data file would carry a unique number ... and if you've forgotten what's on a given tape, just run DIRECTORY and get all the IDs.

Another thing that's handy to know is the starting address (SA) of a program, expecially if you want to copy it to another tape. (Ending addresses are easy ... just load the program, then look at the contents of 17ED and 17EE). Well, DIRECTORY shows starting addresses, too.

The program is fully relocatable, so put it anywhere convenient. Start at the first instruction (0000 in the listing). Incidentally, 0001 to 001D of this program are functionally identical to the KIM monitor 188C to 18C1.

After you start the program, start your audio tape input. When DIRECTORY finds a program, it will display the Start Address (first four digits) and the Program ID. Hit any key and it will scan for the next program.

```
0000  D8          GO    CLD
0001  A9 07             LDA #$07      Directional reg
0003  8D 42 17          STA SBD
0006  20 41 1A    SYN   JSR  RDBIT    Scan thru bits...
0009  46 F9             LSR INH       ..shifting new bit
000B  05 F9             ORA INH       ..into left of
000D  85 F9             STA INH       ..byte INH
000F  C9 16       TST   CMP #$16      SYNC character?
0011  D0 F3             BNE SYN       no, back to bits
0013  20 24 1A          JSR RDCHT     get a character
0016  C6 F9             DEC INH       count 22 SYNC's
0018  10 F5             BPL TST
001A  C9 2A             CMP #$2A      then test astk
001C  D0 F1             BNE TST       ..or SYNC
001E  A2 FD             LDX #$FD      if asterisk,
0020  20 F3 19    RD    JSR RDBYT     stack 3 bytes
0023  95 FC             STA POINTH+1,X      into display
0025  E8                INX                 area
0026  30 F8             BMI RD
0028  20 1F 1F    SHOW  JSR SCANDS    ...and shine
002B  D0 D3             BNE GO        until keyed
002D  F0 F9             BEQ SHOW      at's all folks
```

HYPERTAPE

by Jim Butterfield

How long does it take you to load a full 1K of KIM-1 memory? Over two minutes? And if you're going for memory expansion, how long will it take you to load your 8K? Twenty minutes?

Hold onto your hats. Program HYPERTAPE! will write fully compatible tapes in a fraction of the time. You can load a full 1K in 21 seconds.

Fully compatible means this: once you've written a tape using HYPERTAPE! you can read it back in using the normal KIM-1 program (starting at 1873 as usual). And the utilities and daagnostic programs work on this super-compressed data (e.g., DIRECTORY and VUTAPE).

You'll need some memory space for the program, of course. If you have memory expansion, there'll be no problem finding space, of course. But if you're on the basic KIM-1, as I am, you'll have to "squeeze in" HYPERTAPE! along with the programs you're dumping to tape. I try to leave page 1 alone usually (the stack can overwrite your program due to bugs); so I stage HYPERTAPE! in that area. For the convenience of relocation, the listing underlines those addresses that will need changing. There are also four values needed in page zero which you may change to any convenient location.

For those interested in the theory of the thing, I should mention: HYPERTAPE! is not the limit. If you wished to abandon KIM-1 monitor compatibility, you could continue to speed up tape by a factor of 4 or 5 times more. Can you imagine reading 1K in four seconds? For the moment, however, HYPERTAPE! is plenty fast for me.

```
                        ;this program also included in Super-dupe
0100 A9 AD      DUMP    LDA #$AD
0102 8D EC 17           STA VEB
0105 20 32 19           JSR INTVEB    set up sub
0108 A9 27              LDA #$27
010A 85 F5              STA GANG      flag for SBD
010C A9 BF              LDA #$BF
010E 8D 43 17           STA PBDD
0111 A2 64              LDX #$64
0113 A9 16              LDA #$16
0115 20 61 01           JSR HIC
0118 A9 2A              LDA #$2A
011A 20 88 01           JSR OUTCHT
011D AD F9 17           LDA ID
0120 20 70 01           JSR OUTBT
0123 AD F5 17           LDA SAL
```

```
0126 20 6D 01          JSR OUTBTC
0129 AD F6 17          LDA SAH
012C 20 6D 01          JSR OUTBTC
012F 20 EC 17  DUMPT4  JSR VEB
0132 20 6D 01          JSR OUTBTC
0135 20 EA 19          JSR INCVEB
0138 AD ED 17          LDA VEB+1
013B CD F7 17          CMP EAL
013E AD EE 17          LDA VEB+2
0141 ED F8 17          SBC EAH
0144 90 E9             BCC DUMPT4
0146 A9 2F             LDA #$2F
0148 20 88 01          JSR OUTCHT
014B AD E7 17          LDA CHKL
014E 20 70 01          JSR OUTBT
0151 AD E8 17          LDA CHKH
0154 20 70 01  EXIT    JSR OUTBT
0157 A2 02             LDX #$02
0159 A9 04             LDA #$04
015B 20 61 01          JSR HIC
015E 4C 5C 18          JMP DISPZ
                       ;subroutines
0161 86 F1     HIC     STX TIC
0163 48        HIC1    PHA
0164 20 88 01          JSR OUTCHT
0167 68                PLA
0168 C6 F1             DEC TIC
016A D0 F7             BNE HIC1
016C 60                RTS
016D 20 4C 19  OUTBTC  JSR CHKT
0170 48        OUTBT   PHA
0171 4A                LSR A
0172 4A                LSR A
0173 4A                LSR A
0174 4A                LSR A
0175 20 7D 01          JSR HEXOUT
0178 68                PLA
0179 20 7D 01          JSR HEXOUT
017C 60                RTS
                       ;
017D 29 0F     HEXOUT  AND #$0F
017F C9 0A             CMP #$0A
0181 18                CLC
0182 30 02             BMI HEX1
0184 69 07             ADC #$07
0186 69 30     HEX1    ADC #$30
0188 A0 07     OUTCHT  LDY #$07
018A 84 F2             STY COUNT
018C A0 02     TRY     LDY #$02
018E 84 F3             STY TRIB
0190 BE BE 01  ZON     LDX NPUL,Y
0193 48                PHA
```

```
0194 2C 47 17   ZON1    BIT CLKRDI
0197 10 FB              BPL ZON1
0199 B9 BF 01           LDA TIMG,Y
019C 8D 44 17           STA CLK1T
019F A5 F5              LDA GANG
01A1 49 80              EOR #$80
01A3 8D 42 17           STA SBD
01A6 85 F5              STA GANG
01A8 CA                 DEX
01A9 D0 E9              BNE ZON1
01AB 68                 PLA
01AC C6 F3              DEC TRIB
01AE F0 05              BEQ SETZ
01B0 30 07              BMI ROUT
01B2 4A                 LSR A
01B3 90 DB              BCC ZON
01B5 A0 00      SETZ    LDY #0
01B7 F0 D7              BEQ ZON
01B9 C6 F2      ROUT    DEC COUNT
01BB 10 CF              BPL TRY
01BD 60                 RTS
                ;frequency/density controls
01BE 02         NPUL    .BYTE $02
01BF C3 03 7E   TIMG    .BYTE $C3,$03,$7E
```

***** Hex Dump - Hypertape *****

```
0100-   A9 AD 8D EC 17 20 32 19 A9 27 85 F5 A9 BF 8D 43
0110-   17 A2 64 A9 16 20 61 01 A9 2A 20 88 01 AD F9 17
0120-   20 70 01 AD F5 17 20 6D 01 AD F6 17 20 6D 01 20
0130-   EC 17 20 6D 01 20 EA 19 AD ED 17 CD F7 17 AD EE
0140-   17 ED F8 17 90 E9 A9 2F 20 88 01 AD E7 17 20 70
0150-   01 AD E8 17 20 70 01 A2 02 A9 04 20 61 01 4C 5C
0160-   18 86 F1 48 20 88 01 68 C6 F1 D0 F7 60 20 4C 19
0170-   48 4A 4A 4A 4A 20 7D 01 68 20 7D 01 60 29 0F C9
0180-   0A 18 30 02 69 07 69 30 A0 07 84 F2 A0 02 84 F3
0190-   BE BE 01 48 2C 47 17 10 FB B9 BF 01 8D 44 17 A5
01A0-   F5 49 80 8D 42 17 85 F5 CA D0 E9 68 C6 F3 F0 05
01B0-   30 07 4A 90 DB A0 00 F0 D7 C6 F2 10 CF 60 02 C3
01C0-   03 7E
```

Thanks go to Julien Dubé for his help in staging early versions of HYPERTAPE!

MEMORY TEST

Jim Butterfield

Testing RAM isn't just a question of storing a value and then checking it. It's important to test for interference between locations. Such tests often involve writing to one location and then checking all other locations to see they haven't been disturbed; this can be time consuming.

This program checks memory thoroughly and runs exceptionally fast. It is adapted from an algorithm by Knaizuk and Hartmann published in 'IEEE Transactions on Computers', April 1977.

The program first puts value FF in every location under test. Then it puts 00 in every third location, after which it tests all locations for correctness. The test is repeated twice more with the positions of the 00's changed each time. Finally, the whole thing is repeated with the FF and 00 values interchanged.

To run: Set the addresses of the first and last memory pages you wish to test into locations 0000 and 0001 respectively. Start the program at address 0002; it will halt with a memory address on the display. If no faults were found, the address will be one location past the last address tested. If a fault is found, its address will be displayed.

Example: To test 0100 to 02FF (pages 01 and 02) in KIM: Set 0000 to 01, 0001 to 02, start program at 0002. If memory is good, see 0300 (=02FF + 1). Now if you try testing 0100 to 16FF (0000=01,0001=16) the program will halt at the first bad location - this will be 0400 if you haven't added memory.

```
0000 xx       BEGIN  xx       starting page for test
0001 xx       END    xx       ending page for test
0002 A9 00    START  LDA #0       zero pointers
0004 A8              TAY          for low-order
0005 85 FA           STA POINTL   addresses;
0007 85 70    BIGLP  STA FLAG      =00 first pass, =FF second pass
0009 A2 02           LDX #2
000B 86 72           STX MOD      set 3 tests each pass
000D A5 00    PASS   LDA BEGIN    set pointer to..
000F 85 FB           STA POINTH   ..start of test area
0011 A6 01           LDX END
0013 A5 70           LDA FLAG
0015 49 FF           EOR #$FF     reverse FLAG
0017 85 71           STA FLIP     ..=FF first pass, =00 second pass
0019 91 FA    CLEAR  STA (POINTL),Y    write above FLIP value..
001B C8              INY                 ..into all locations
001C D0 FB           BNE CLEAR
001E E6 FB           INC POINTH
0020 E4 FB           CPX POINTH
0022 B0 F5           BCS CLEAR
```

```
                ; FLIP value in all locations - now change 1 in 3
0024 A6 72          LDX MOD
0026 A5 00          LDA BEGIN     set pointer..
0028 85 FB          STA POINTH     ..back to start
002A A5 70    FILL  LDA FLAG      change value
002C CA       TOP   DEX
002D 10 04          BPL SKIP      skip 2 out of 3
002F A2 02          LDX #2        restore 3-counter
0031 91 FA          STA (POINTL),Y   change 1 out of 3
0033 C8       SKIP  INY
0034 D0 F6          BNE TOP
0036 E6 FB          INC POINTH    new page
0038 A5 01          LDA END       have we passed..
003A C5 FB          CMP POINTH    ..end of test area?
003C B0 EC          BCS FILL      nope, keep going
                ; memory set up - now test it
003E A5 00          LDA BEGIN     set pointer..
0040 85 FB          STA POINTH    ..back to start
0042 A6 72          LDX MOD       set up 3-counter
0044 A5 71    POP   LDA FLIP      test for FLIP value..
0046 CA             DEX            ..2 out of 3 times..
0047 10 04          BPL SLIP             - or -
0049 A2 02          LDX #2           1 out of 3..
004B A5 70          LDA FLAG      test for FLAG value;
004D D1 FA    SLIP  CMP (POINTL),Y    here's the test...
004F D0 15          BNE OUT          branch if failed
0051 C8             INY
0052 D0 F0          BNE POP
0054 E6 FB          INC POINTH
0056 A5 01          LDA END
0058 C5 FB          CMP POINTH
005A B0 E8          BCS POP
                ; above test OK - change & repeat
005C C6 72          DEC MOD      change 1/3 position
005E 10 AD          BPL PASS       ..& do next third
0060 A5 70          LDA FLAG     invert..
0062 49 FF          EOR #$FF       ..flag for pass two
0064 30 A1          BMI BIGLP
0066 84 FA    OUT   STY POINTL    put low order adds to display
0068 4C 4F 1C       JMP START        ...and exit to KIM
                                                        006B
```

***** Hex Dump - Memory Test *****

```
0000 00 00 A9 00 A8 85 FA 85 70 A2 02 86 72 A5 00 85
0010 FB A6 01 A5 70 49 FF 85 71 91 FA C8 D0 FB E6 FB
0020 E4 FB B0 F5 A6 72 A5 00 85 FB A5 70 CA 10 04 A2
0030 02 91 FA C8 D0 F6 E6 FB A5 01 C5 FB B0 EC A5 00
0040 85 FB A6 72 A5 71 CA 10 04 A2 02 A5 70 D1 FA D0
0050 15 C8 D0 F0 E6 FB A5 01 C5 FB B0 E8 C6 72 10 AD
0060 A5 70 49 FF 30 A1 84 FA 4C 4F 1C
```

MINI DIS

By Dan Lewart

One day I was single-stepping through a program and not being too alert, I kept going after the program ended. Then I noticed I was going through instructions not in any OP-code table. What was being executed? With a little luck I found that many nonexistent codes would duplicate others with only one bit changed. I haven't looked into it very deeply, but here are two examples: 17 is the same as 16 (ASL-Z, PAGE) and FF is the same as FE (INC ABS,X).

By single-stepping I could determine the number of bytes in all instructions. This worked for all instructions except for 02,12,22,32,42,52,62,72,92,B2,D2 and F2, which blank the display. After filling in the Bytes per Instruction table many patterns became obvious. For example, the op-code ending with digits 8 and A could be summarized as having a bit pattern of xxxx10x0, where "x" means don't care. This covers all possibilities and when a number of this form is ANDed with 00001101 (mask all the x bits) the result will be 00001000. By doing this for all 0 (illegal), 1 and 3 byte instructions and having the 2 byte instructions "whatever's left over" I had the basis of my semi-disassembler. The only odd byte length is that of 20 (JSR) which "should" be only 1 byte long.

Though this is not a full disassembler, it has helped me to write several programs, including itself. To relocate the program change locations 374-6, 379-B and 38E-390 to jump to the appropriate locations. If you have a program in page 1 or don't want to write on the stack, change 397 and 39A to EA (NOP).

To run the program, store 00 in 17FA and 03 in 17FB. Go to the beginning of your program and press "ST". You will then see the first instruction displayed. If it is illegal, the location and opcode will flash on and off. In that case, press "RS". To display the next instruction press "F" To display the current address and opcode press "0", at any time. To backstep press "8" When you have backstepped to the beginning of your program, or changed locations 397 and 39A, pressing "B" acts like "PC".

```
0300  D8        START    SED
0301  A2 FF               LDX #$FF        INITIALIZE STACK
0303  9A                 TXS             POINTER
0304  A0 00      INIT     LDY #$00        (E6-EE)=0
0306  A2 09               LDX #$09
0308  94 E5      INIT1    STY 00E5,X
030A  CA                 DEX
030B  D0 FB              BNE INIT1
030D  E8                 INX             X=1
```

```
030E  B1 FA     LENGTH    LDA (POINTL),Y GET OPCODE, FIND LENGTH
0310  C9 20               CMP#$20        ANALYZE BIT PATTERNS
0312  F0 3B               BEQ 3BYTE      %00100000 ; 3 BYTES
0314  29 9F               AND #$9F       "X" MEANS DON'T CARE
0316  F0 35               BEQ 1BYTE      %0XX00000 ; 1 BYTE (20)
0318  C9 92               CMP #$92
031A  F0 1A               BEQ FLASH      %1XX10010 ; ILLEGAL (B2,D2)
031C  A8                  TAY            STORE TEMPORARILY
031D  29 1D               AND #$1D
031F  C9 19               CMP #$19
0321  F0 2C               BEQ 3BYTE      %XXX110X1 ; 3 BYTES (59,B9)
0323  29 0D               AND 34 0D
0325  C9 08               CMP #$08
0327  F0 24               BEQ 1BYTE      %XXXXX0X0 ; 1 BYTE (D8,4A)
0329  29 0C               AND #$0C
032B  C9 0C               CMP #$0C
032D  F0 20               BEQ 3BYTE      %XXXX11XX  ; 3 BYTES (4C,EE)
032F  98                  TYA            RESTORE
0330  29 8F               AND #$8F
0332  C9 02               CMP #$02       %0XXX0010 ; ILLEGAL (22,52)
0334  D0 18               BNE 2BYTE      ALL LEFTOVERS ; 2 BYTES
0336  E6 EC     FLASH     INC 00EC       FLIP BIT 0
0338  A9 FF               LDA #$FF       LOOP FOR 1/4 SEC.
033A  8D 07 17            STA 1707
033D  A5 EC     FLASH1    LDA 00EC       BLINK ON OR OFF
033F  29 01               AND #$01
0341  F0 03               BEQ FLASH2     BIT 0=0 ; BLINK OFF
0343  20 19 1F            JSR SCAND      BIT 0=1 ; BLINK ON
0346  2C 07 17  FLASH2    BIT 1707
0349  30 EB               BMI FLASH
034B  10 F0               BPL FLASH1
034D  E8          1BYTE   INX
034E  E8          2BYTE   INX
034F  8A          3BYTE   TXA       CENTER CODE
0350  49 07               EOR #$07
0352  85 ED               STA 00ED
0354  A4 EE       CONVRT  LDY #$EE   LOOP FOR EACH BYTE
0356  B1 FA               LDA (POINTL),Y CONVERT AND STORE
0358  48                  PHA          IN E6 - EB
0359  4A 4A               LSR's
035B  4A 4A               LSR's
035D  A8                  TAY
035E  B9 E7 1F            LDA TABLE,Y
0361  95 E5               STA 00E5,X
0363  E8                  INX
0364  68                  PLA
0365  29 0F               AND #$0F
0367  A8                  TAY
0368  B9 E7 1F            LDA TABLE,Y
036B  95 E5               STA 00E5,X
036D  E8                  INX
036E  E6 EE               INC 00EE
0370  E4 ED               CPX 00ED
0372  90 E0               BCC CONVRT
0374  20 AF 03    K DOWN  JSR DISP       DISPLAY UNTIL ALL KEYS
0377  D0 FB               BNE K DOWN     ARE UP
0379  20 AF 03    K UP    JSR DISP       DISPLAY AND GET KEY
```

```
037C  20 6A 1F            JSR GETKEY
037F  C9 0B       B?      CMP #$0B      IS "B" PRESSED?
0381  D0 0E              BNE PLUS?     NO, BRANCH
0383  BA          BCKSTP  TSX
0384  E0 FF               CPX #$FF      IS STACK EMPTY?
0386  F0 20              BEQ WINDOW    YES, ACT LIKE "PC"
0388  68                  PLA           PULL FB AND FA
0389  85 FB               STA 00FB      DISPLAY WORD
038B  68                  PLA
038C  85 FA               STA 00FA
038E  4C 04 03    NEWORD  JMP INIT
0391  C9 0F       PLUS?   CMP #$0F      IS "F" PRESSED?
0393  D0 0F               BNE PC?       NO, BRANCH
0395  A5 FA       STEP    LDA 00FA      PUSH FA AND FB
0397  48                  PHA
0398  A5 FB               LDA 00FB
039A  48                  PHA
039B  20 63 1F    STEP 1  JSR INCPT     FIND NEW LOCATION
039E  C6 EE               DEC 00EE      DISPLAY WORD
03A0  F0 EC               BEQ NEWORD
03A2  D0 F7               BNE STEP 1
03A4  C9 14       PC?     CMP #$14      IS "PC" PRESSED?
03A6  D0 D1               BNE K UP      NO, GET KEY
03A8  20 19 1F    WINDOW  JSR SCAND     DISPLAY LOCATION
03AB  F0 CC               BEQ K UP      UNTIL KEY RELEASED
03AD  D0 F9               BNE WINDOW    THEN GET KEY
03AF  A9 7F       DISP    LDA #$7F      SEGMENTS TO OUTPUT
03B1  8D 41 17            STA PADD
03B4  A2 08               LDX #$08      INITIALIZE
03B6  A0 00               LDY #$00
03B8  84 FC       DISP 1  STY 00FC
03BA  B9 E6 00            LDA 00E6,Y    GET CHARACTER
03BD  20 4E 1F            JSR 1F4E      DISPLAY CHARACTER
03C0  C8                  INY           NEXT CHARACTER
03C1  C0 06               CPY #$06
03C3  90 F3               BCC DISP1
03C5  4C 3D 1F            JMP 1F3D      DONE, KEY DOWN?
```

***** HEX DUMP - MINI DIS *****

```
0300 D8 A2 FF 9A A0 00 A2 09 94 E5 CA D0 FB E8 B1 FA
0310 C9 20 F0 3B 29 9F F0 35 C9 92 F0 1A A8 29 1D C9
0320 19 F0 2C 29 0D C9 08 F0 24 29 0C C9 0C F0 20 98
0330 29 8F C9 02 D0 18 E6 EC A9 FF 8D 07 17 A5 EC 29
0340 01 F0 03 20 19 1F 2C 07 17 30 EB 10 F0 E8 E8 8A
0350 49 07 85 ED A4 EE B1 FA 48 4A 4A 4A 4A A8 B9 E7
0360 1F 95 E5 E8 68 29 0F A8 B9 E7 1F 95 E5 E8 E6 EE
0370 E4 ED 90 E0 20 AF 03 D0 FB 20 AF 03 20 6A 1F C9
0380 0B D0 0E BA E0 FF F0 20 68 85 FB 68 85 FA 4C 04
0390 03 C9 12 D0 0F A5 FA 48 A5 FB 48 20 63 1F C6 EE
03A0 F0 EC D0 F7 C9 14 D0 D1 20 19 1F F0 CC D0 F9 A9
03B0 7F 8D 41 17 A2 08 A0 00 84 FC B9 E6 00 20 4E 1F
03C0 C8 C0 06 90 F3 4C 3D 1F
```

MOVIT

By Lew Edwards

ANOTHER move program? This one moves anything anywhere! No limit to number of bytes, or locations in memory, or overlapping of source and destination. Use it to lift sections of code from other programs, close in or open up gaps for altering programs, moving programs to another location (use Butterfield's RELOCATE to take care of the branch and address correction). Locate it wherever you have the room.

Use is straight forward. Old start address goes in D0,1 ; old end address in D2,3; new start address in D4,5 before running the program which starts at 1780, or wherever you want to have it in your system. Program uses zero page locations D0 thru D9 to do the job.

```
1780  D8        START   CLD
1781  A0 FF             LDY #$FF        STORE TEST VALUE
1783  38                SEC
1784  A5 D2             LDA OEAL        HOW MANY BYTES?
1786  E5 D0             SBC OSAL        TO MOVE?
1788  85 D8             STA BCL
178A  A5 D3             LDA OEAH
178C  E5 D1             SBC OSAH
178E  85 D9             STA BCH
1790  18                CLC
1791  A5 D8             LDA BCL         ADD THE COUNT TO
1793  65 D4             ADC NSAL        THE NEW START TO
1795  85 D6             STA NEAL        GET A NEW END
1797  A5 D9             LDA BCH
1799  65 D5             ADC NSAH
179B  85 D7             STA NEAH
179D  E6 D8             INC BCL         ADJUST THE BYTE COUNT
179F  E6 D9             INC BCH         TO PERMIT ZERO TESTING
17A1  38                SEC
17A2  A5 D4             LDA NSAL        IF NEW LOCATION
17A4  E5 D0             SBC OSAL        HIGHER THAN OLD
17A6  A5 D5             LDA NSAH        CARRY FLAG IS SET
17A8  E5 D1             SBC OSAH
17AA  A2 00     LOOP    LDX #$00        HIGH POINTER INDEX
17AC  90 02             BCC MOVE
17AE  A2 02             LDX #$02        LOW POINTER INDEX
17B0  A1 D0     MOVE    LDA OSAL,X      MOVE OLD
17B2  81 D4             STA NSAL,X      TO NEW
17B4  90 14             BCC DOWN
17B6  C6 D2             DEC OEAL        ADJUST UP POINTER, (OLD)
17B8  98                TYA             BELOW ZERO?
17B9  45 D2             EOR OEAL
17BB  D0 02             BNE NO          NO, ENOUGH
```

```
17BD  C6 D3          DEC OEAH      YES, ADJUST THE HIGH BYTE
17BF  C6 D6   NOT    DEC NEAL      ADJUST THE OTHER ONE (NEW)
17C1  98             TYA
17C2  45 D6          EOR NEAL      NEED HIGH BYTE ADJUSTED?
17C4  D0 02          BNE NEIN      NO
17C6  C6 D7          DEC NEAH      YES, DO IT
17C8  B0 0C   NEIN   BCS COUNT
17CA  E6 D0   DOWN   INC OSAL      ADJUST "OLD" DOWN POINTER
17CC  D0 02          BNE NYET
17CE  E6 D1          INC OSAH      AND THE HIGH BYTE IF NEEDED
17D0  E6 D4   NYET   INC NSAL      AND THE "NEW" ONE
17D2  D0 02          BNE COUNT
17D4  E6 D5          INC NSAH
17D6  C6 D8   COUNT  DEC BCL       TICK OFF THE BYTES,
17D8  D0 02          BNE ONE       ENOUGH FINGERS?
17DA  C6 D9          DEC BCH       USE THE OTHER HAND
17DC  D0 CC   ONE    BNE LOOP      'TIL THEY'RE ALL DONE
17DE  00      DONE   BRK           & BACK TO MONITOR
```

P.S. Don't forget to set the IRQ vector for the break
(KIM - 1C00 at 17FE,FF)

***** Hex Dump - Movit *****

```
1780 D8 A0 FF 38 A5 D2 E5 D0 85 D8 A5 D3 E5 D1 85 D9
1790 18 A5 D8 65 D4 85 D6 A5 D9 65 D5 85 D7 E6 D8 E6
17A0 D9 38 A5 D4 E5 D0 A5 D5 E5 D1 A2 00 90 02 A2 02
17B0 A1 D0 81 D4 90 14 C6 D2 98 45 D2 D0 02 C6 D3 C6
17C0 D6 98 45 D6 D0 02 C6 D7 B0 0C E6 D0 D0 02 E6 D1
17D0 E6 D4 D0 02 E6 D5 C6 D8 D0 02 C6 D9 D0 CC 00
```

Addition: The last address filled can be displayed after the program is complete by adding the following code:

(1) 85 FA between instructions now at 1795 and 1797
(2) 85 FB between instructions now at 179B and 179D
(3) replace the break at the end with 4C 4F 1C

Use Movit to move itself to another location and then again to open up the necessary spaces!

PLL SET

Lewis Edwards, Jr.

Having trouble loading from tape, especially on "HYPERTAPE"? Suspect the PLL adjustment might be off, but were afraid to adjust it, or didn't have a meter or scope handy? Use this program and KIM's built in hardware to make the adjustment. Hold the tip of the plug you plug into the tape recorder's earphone jack to applications pin #14 and adjust the control for 0's or combinations of 7's and L's on the display. "L" means the PLL TEST line is low and "7" means it's high. The program generates a signal that alternates slightly below and slightly above theone generated by KIM at 1A6B. The regular tape input channel is utilized and decoded to control the display.

```
1780  A9 07       BEGN   LDA #07      Set the input
1782  8D 42 17           STA SBD
1785  A9 01              LDA #01      and output ports
1787  8D 01 17           STA PAO
178A  85 E1              STA E1       Initialize the toggle
178C  A9 7F              LDA #7F
178E  8D 41 17           STA PADD     Open display channels
1791  A2 09       MORE   LDX #09      Start with the first
1793  A0 07              LDY #07      digit  Light top & right
1795  2C 42 17           BIT SBD      if PLL output
1798  30 02              BMI SEGS     is high
179A  A0 38              LDY #38      otherwise left & bottom
179C  8C 40 17     SEGS   STY SAD      Turn on the segments
179F  8E 42 17           STX SBD      and the digit
17A2  2C 47 17     DELA   BIT CLKRDI   Half cycle done?
17A5  10 FB              BPL DELA     No, wait for time up
17A7  E6 E2              INC E2       Count the cycles
17A9  30 04              BMI LOTO     128 ½ cycles, send low tone
17AB  A9 91        HITO   LDA #91      128 ½ cycles, send hi tone
17AD  D0 03              BNE CLK1
17AF  A9 93        LOTO   LDA #93
17B1  EA                 NOP          Equalize the branches
17B2  8D 44 17     CLK1   STA CLK1T    Set the clock
17B5  A9 01              LDA #01
17B7  45 E1              EOR E1       Flip the toggle register
17B9  85 E1              STA E1
17BB  8D 00 17           STA PAO      Toggle the output port
17BE  E8                 INX
17BF  E8                 INX          Next display digit
17C0  E0 15              CPX #15      Last one?
17C2  D0 CF              BNE NEXT     No, do next
17C4  F0 CB              BEQ MORE     Yes, do more
```

```
1780 A9 07 8D 42 17 A9 01 8D 01 17 85 E1 A9 7F 8D 41
1790 17 A2 09 A0 07 2C 42 17 30 02 A0 38 8C 40 17 8E
17A0 42 17 2C 47 17 10 FB E6 E2 30 04 A9 91 D0 03 A9
17B0 93 EA 8D 44 17 A9 01 45 E1 85 E1 8D 00 17 E8 E8
17C0 E0 15 D0 CF F0 CB
```

RELOCATE

Jim Butterfield

Ever long for an assembler? Remember when you wrote that 300 byte program - and discovered that you'd forgotten one vital instruction in the middle? And to make room, you'd have to change all those branches, all those addresses... Or the program with that neat piece of coding in it, that you suddenly need to remove (say, to change it to a subroutine)...but if you do, you'll have to fill all that empty space with NOPs? It's enough to make a grown programmer cry...

Dry those tears. Program RELOCATE will fix up all those addresses and branches for you, whether you're opening out a program to fit in an extra instruction, closing up space you don't need, or just moving the whole thing someplace else.

RELOCATE doesn't move the data. It just fixes up the addresses before you make the move. It won't touch zero page addresses; you'll want them to stay the same. And be careful: it won't warn you if a branch instruction goes out of range.

You'll have to give RELOCATE a lot of information about your program:

(1) Where your program starts. This is the first instruction in your whole program (including the part that doesn't move). RELOCATE has to look through your whole program, instruction by instruction, correcting addresses and branches where necessary. Be sure your program is a continuous series of instructions (don't mix data in; RELOCATE will take a data value of 10 as a BPL instruction and try to correct the branch address), and place a dud instruction (FF) behind your last program instruction. This tells RELOCATE where to stop.

Place the program start address in locations EA and EB, low order first as usual. Don't forget the FF behind the last instruction; it doesn't matter if you temporarily wipe out a byte of data - you can always put it back later.

(2) Where relocation starts, this is the first address in your program that you want to move. If you're moving the whole program, it will be the same as the program start address, above. This address is called the boundary.

Place the boundary address in locations EC and ED, low order first.

(3) How far you will want to relocate information above the boundary. This value is called the increment. For example, if you want to open up three more locations in your program, the increment will be 0003. If you want to close up four addresses, the increment will be FFFC (effectively, a negative number).

Place the increment value in locations E8 and E9, low order first.

(4) A page limit, above which relocation should be disabled. For example, if you're working on a program in the 0200 to 03FF range, your program might also address a timer or I/O registers, and might call subroutines in the monitor. You don't want these addresses relocated, even though they are above the boundary! So your page limit would be 17, since these addresses are all over 1700.

On the other hand, if you have memory expansion and your program is at address 2000 and up, your page limit will need to be much higher. You'd normally set the page limit to FF, the highest page in memory.

Place the page limit in location E7.

Now you're ready to go. Set RELOCATE's start address, hit go - and ZAP!-your addresses are fixed up.

After the run, it's a good idea to check the address now in 00EA and 00EB - it should point at the FF at the end of your program, confirming that the run went OK.

Now you can move the program. If you have lots of memory to spare, you can write a general MOVE program and link it in to RELOCATE, so as to do the whole job in one shot.

But if, like me, you're memory-deprived, you'll likely want to run RELOCATE first, and then load in a little dustom-written program to do the actual moving. The program will vary depending on which way you want to move, how far, and how much memory is to be moved. In a pinch, you can use the FF option of the cassette input program to move your program.

Last note: the program terminates with a BRK instruction. Be sure your interrupt vector (at 17FE and 17FF) is set to KIM address 1C00 so that you get a valid "halt".

RELOCATE — Jim Butterfield

```
                        ; following addresses must be initialized
                        ; by user prior to run
00E7                    PAGLIM *=*+1        limit above which kill relocn
00E8                    ADJST  *=*+2        adjustment distance (signed)
00EA                    POINT  *=*+2        start of program
00EC                    BOUND  *=*+2        lower boundary for adjustment
                        ; main program starts here
0110 D8                 START  CLD
0111 A0 00                     LDY #0
0113 B1 EA                     LDA (POINT),Y      get op code
0115 A8                        TAY                +cache in Y
0116 A2 07                     LDX #7
0118 98                 LOOP   TYA                restore op code
0119 3D 8E 01                  AND TAB1-1,X       remove unwanted bits
011C 5D 95 01                  EOR TAB2-1,X       & test the rest
011F F0 03                     BEQ FOUND
```

```
0121 CA                     DEX             on to the next test
0122 D0 F4                  BNE LOOP        ...if any
0124 BC 9D 01     FOUND     LDY TAB3,X      length or flag
0127 30 0D                  BMI TRIP        triple length?
0129 F0 22                  BEQ BRAN        branch?
012B E6 EA        SKIP      INC POINT       mving right along..
012D D0 02                  BNE INEX        ..to next op code
012F E6 EB                  INC POINT+1
0131 88           INEX      DEY
0132 D0 F7                  BNE SKIP
0134 F0 DA                  BEQ START
                  ; length 3 or illegal
0136 C8           TRIP      INY
0137 30 D9                  BMI START+2     illegal/end to BRK halt
0139 C8                     INY             set Y to 1
013A B1 EA                  LDA (POINT),Y   lo-order operand
013C AA                     TAX             ...into X reg
013D C8                     INY             Y=2
013E B1 EA                  LDA (POINT),Y   hi-order operand
0140 20 79 01               JSR ADJUST      change address, maybe
0143 91 EA                  STA (POINT),Y   ...and put it back
0145 88                     DEY             Y=1
0146 8A                     TXA
0147 91 EA                  STA (POINT),Y   ...also hi-order
0149 A0 03                  LDY #3          Y=3
014B 10 DE                  BPL SKIP
                  ; branch: check "to" and "from" address
014D C8           BRAN      INY             Y=1
014E A6 EA                  LDX POINT       "from" addrs lo-order
0150 A5 EB                  LDA POINT+1     ...& hi-order
0152 20 79 01               JSR ADJUST      change, maybe
0155 86 E0                  STX ALOC        save lo-order only
0157 A2 FF                  LDX #$FF        flag for "back" branches
0159 B1 EA                  LDA (POINT),Y   get relative branch
015B 18                     CLC
015C 69 02                  ADC #2          adjust the offset
015E 30 01                  BMI OVER        backwards branch?
0160 E8                     INX             nope
0161 86 E3        OVER      STX LIMIT
0163 18                     CLC
0164 65 EA                  ADC POINT       calculate "to" lo-order
0166 AA                     TAX             ...and put in X
0167 A5 E3                  LDA LIMIT       00 or FF
0169 65 EB                  ADC POINT+1     "to" hi-order
016B 20 79 01               JSR ADJUST      change, maybe
016E CA                     DEX             readjust the offset
016F CA                     DEX
0170 8A                     TXA
0171 38                     SEC
0172 E5 E0                  SBC ALOC        recalculate relative branch
0174 91 EA                  STA (POINT),Y   and re-insert
0176 C8                     INY             Y=2
0177 10 B2                  BPL SKIP
```

```
                        ; examine address and adjust, maybe
0179 C5 E7      ADJUST  CMP PAGLIM
017B B0 11              BCS OUT             too high?
017D C5 ED              CMP BOUND+1
017F D0 02              BNE TES2            hi-order?
0181 E4 EC              CPX BOUND           lo-order?
0183 90 09      TES2    BCC OUT             too low?
0185 48                 PHA                 stack hi-order
0186 8A                 TXA
0187 18                 CLC
0188 65 E8              ADC ADJUST          adjust lo-order
018A AA                 TAX
018B 68                 PLA                 unstack hi-order
018C 65 E9              ADC ADJST+1         and adjust
018E 60         OUT     RTS
                        ; tables for op-code indentification
018F 0C 1F 0D   TAB1    .BYTE               $0C,$1F,$0D,$87,$1F,$FF,$03
0192 87 1F FF
0195 03
0196 0C 19 08   TAB2    .BYTE               $0C,$19,$08,$00,$10,$20,$03
0199 00 10 20
019C 03
019D 02 FF FF   TAB3    .BYTE           $02,$FF,$FF,$01,$01,$00,$FF,$FE
01A0 01 01 00
01A3 FF FE
                        ;       end
```

Credit for the concept of RELOCATE goes to Stan Ockers, who insisted that it was badly needed, and maintained despite my misgivings that it should be quite straightforward to program. He was right on both counts.

***** Hex Dump - Relocate *****

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0110-   D8 A0 00 B1 EA A8 A2 07 98 3D 8E 01 5D 95 01 F0
0120-   03 CA D0 F4 BC 9D 01 30 0D F0 22 E6 EA D0 02 E6
0130-   EB 88 D0 F7 F0 DA C8 30 D9 C8 B1 EA AA C8 B1 EA
0140-   20 79 01 91 EA 88 8A 91 EA A0 03 10 DE C8 A6 EA
0150-   A5 EB 20 79 01 86 E0 A2 FF B1 EA 18 69 02 30 01
0160-   E8 86 E3 18 65 EA AA A5 E3 65 EB 20 79 01 CA CA
0170-   8A 38 E5 E0 91 EA C8 10 B2 C5 E7 B0 11 C5 ED D0
0180-   02 E4 EC 90 09 48 8A 18 65 E8 AA 68 65 E9 60 0C
0190-   1F 0D 87 1F FF 03 0C 19 08 00 10 20 03 02 FF FF
01A0-   01 01 00 FF FE
```

USING PROGRAM RELOCATE - an example. Jim Butterfield

Program RELOCATE is important, and powerful. But it takes a little getting used to. Let's run through an example. Follow along on your KIM, if you like.

Suppose we'd like to change program LUNAR LANDER. When you run out of fuel on the lander, you get no special indication, except that you start falling very quickly. Let's say we want to make this minor change: if you run out of fuel, the display flips over to Fuel mode, so that the pilot will see immediately.

Digging through the program reveals two things: (i) you go to fuel mode by storing 00 into MODE (address E1); and, (ii) the out-of-fuel part of the program is located at 024C to 0257. So if we can insert a program to store zero in mode as part of our out-of-fuel, we should have accomplished our goal. Closer inspection reveals that we can accomplish this by inserting 85 E1 (STA MODE) right behind the LDA instruction at 024C.

Let's do it.

First, we must store value FF behind the last instruction of our program. So put FF into address 02CC. That wipes out the value 45, but we'll put it back later.

Now, we put our program start address (0200) into addresses EA and EB. Low order first, so 00 goes into address 00EA and 02 goes into 00EB.

Next, the part that we want to move. Since we want to insert a new instruction at address 024E, we must move the program up at this point to make space. In goes the address, low order first: 4E into address 00EC and 02 into address 00ED.

The page limit should be set to 17, since we don't want the addresses of the KIM subroutines to be changed (SCANDS, GETKEY, etc.). So put 17 into address 00E7.

Finally, how far do we want to move the program to make room? Two bytes, of course. Put 02 and 00 into addresses 00E8 and 00E9 respectively.

We're ready to go. Be sure your vectors have been set properly (at addresses 17FA to 17FF). Then set address 0110, the start address of RELOCATE, and press GO.

The display will stop showing 0114 EA, confirming that RELOCATE ran properly. Now check to see the whole program was properly converted by looking at the addresses 00EA-B. We put address 0200 there, remember? Now we'll see address 02CC stored there - the address of the value FF we stored to signal end of program.

Go back to 02CC, where we stored FF, and restore the original value of 45.

We've completed part I. The addresses have been corrected for the move. Let's go on to part II and actually move the program to make room.

My favorite method is to use a tiny program to do the move itself. For moving 1 to 256 bytes to a higher address, I use the program:

A2 nn BD xx xx 9D tt tt CA D0 F7 00.

In the above, nn is the number of bytes to be moved, and xxxx and tttt are the from and to addresses of the data, minus one. Since we want to move about 160 bytes from a block starting at 024E to a block starting at 0250, we code like this:

A2 A0 BD 4D 02 9D 4F 02 CA D0 F7 00.

This little program can be fitted in anywhere. Let's put it in memory starting at address 0040. The final byte, value 00, should end up in 004B. Now back to 0040, hit GO ... and your data/program is moved over. (The tiny program should stop showing address 004D).

There's nothing left to do but actually put the extra instruction (85 E1) into the program at 024E and 024F.

Now run the program. Try deliberately running out of fuel and see if the display flips over to fuel mode automatically when you run out.

If you have followed the above successfully with your KIM, it all seems very easy. It's hard to realize that program RELOCATE has done so much work. But if you check, you'll find the following addresses have been automatically changed:

0203 024B 0256/8 0263/5 0265/7 02A5/7

Do you think that you'd have caught every one of those addresses if you'd tried to do the job manually?

SORT

by Jim Pollock

This program will take any given block of data and arrange it in numerical sequence, whether the data is hex or BCD, or both. Since the program uses relative branch addressing, it can be located anywhere in memory without modification.

The instruction that determines whether data is arranged in ascending or descending order is 011F, (B0 - descending order, 90 - ascending order).

This is a bubble sort. The top item is compared with succeeding items and if a larger number is found, they are swapped. The larger item (now at the top) is then used for comparisons as the process continues through the list. After one complete pass, the largest number will have "bubbled" to the top. The whole process is repeated using the second item to start, then again starting with the third item. Eventually the whole list will be sorted in sequence.

```
          17F5        START LO
          17F6        START HIGH
          17F7        END LO
          17F8        END HI     (NOTE: ENDING ADDRESS IS ONE PAST LAST ITEM)

0200  AD F5 17  SORT        LDA 17F5        TRANSFER START POINTER
0203  85 E8                STA 00E8        TO ZERO PAGE
0205  85 EA                STA 00EA
0207  AD F6 17              LDA 17F6
020A  85 E9                STA 00E9
020C  85 EB                STA 00EB
020E  AD F7 17              LDA 17F7        TRANSFER END POINTER
0211  85 EC                STA 00EC
0213  AD F8 17              LDA 17F8
0216  85 ED                STA 00ED
0218  A2 00                LDX #$00        INDEX TO ZERO (STAYS THERE)
021A  D8                   CLD
021B  A1 E8        GET     LDA (00E8,X)    GET DATA INDIRECT 00E8
021D  C1 EA                CMP (00EA,X)    GREATER THAN INDIR. 00EA?
021F  B0 0C                BCS INCN        NO, INCR. POINTER 00EA
0221  A1 E8        SWAP    LDA (00E8,X)    SWAP DATA IN POINTER
0223  85 E7                STA 00E7        LOCATIONS
```

```
0225  A1 EA          LDA (00EA,X)
0227  81 E8          STA (00E8,X)
0229  A5 E7          LDA 00E7
022B  81 EA          STA (00EA,X)
022D  E6 EA    INCN  INC 00EA       SET UP NEXT COMPARISON
022F  D0 02          BNE LASTN      NO PAGE CHANGE
0231  E6 EB          INC 00EB       PAGE CHANGE
0233  A5 EA    LASTN LDA 00EA       CK FOR LAST ITEM IN PASS
0235  C5 EC          CMP 00EC
0237  D0 E2          BNE GET        NOT YET
0239  A5 ED          LDA 00ED       IS THIS LAST PASS/LOOP?
023B  C5 EB          CMP 00EB
023D  D0 DC          BNE GET        NO
023F  E6 E8          INC 00E8
0241  D0 02          BNE OVER       NO PAGE CHANGE
0243  E6 E9          INC 00E9       PAGE CHANGE
0245  A5 E8    OVER  LDA 00E8       INIT. VALUE FOR NEXT PASS
0247  85 EA          STA 00EA
0249  A5 E9          LDA 00E9
024B  85 EB          STA 00EB
024D  A5 EA          LDA 00EA       LAST ITEM IN LIST?
024F  C5 EC          CMP 00EC
0251  D0 C8          BNE GET        NO, NOT YET
0253  A5 E9          LDA 00E9
0255  85 EB          STA 00EB
0257  C5 ED          CMP 00ED       LAST PAGE?
0259  D0 C0          BNE GET        NO
025B  4C 4F 1C       JMP 1C4F       BACK TO KIM, DONE
```

***** Hex Dump - Sort *****

```
0200 AD F5 17 85 E8 85 EA AD F6 17 85 E9 85 EB AD F7
0210 17 85 EC AD F8 17 85 ED A2 00 D8 A1 E8 C1 EA B0
0220 0C A1 E8 85 E7 A1 EA 81 E8 A5 E7 81 EA E6 EA D0
0230 02 E6 EB A5 EA C5 EC D0 E2 A5 ED C5 EB D0 DC E6
0240 E8 D0 02 E6 E9 A5 E8 85 EA A5 E9 85 EB A5 EA C5
0250 EC D0 C8 A5 E9 85 EB C5 ED D0 C0 4C 4F 1C
```

SUPER-DUPE

by Jim Butterfield

SUPER-DUPE is handy: it lets you duplicate a complete tape containing many programs in jig time. SUPER-DUPE is versatile: it will write various tape densities, from regular to Hypertape. SUPER-DUPE is multi-purpose: if you don't want to duplicate programs, you can use it for cataloguing tapes, or for writing Hypertape.

The maximum size program that SUPER-DUPE can copy is dependent on the amount of memory of the KIM system. The basic 1K system can copy programs up to 512 bytes long.

For duplicating tape, it's useful to have two tape recorders: one for reading the old tape, one for writing the new. They are connected in the usual way, at TAPE IN and TAPE OUT. Pause controls are handy.

SUPER-DUPE starts at address 0000. Hit GO and start the input tape. When a program has been read from the input tape, the display will light, showing the start address of the program and its ID. If you don't want to copy this program, hit 0. Otherwise, stop the input tape; start the output tape (on RECORD); then hit 1 for Hypertape, 6 for regular tape, or any intermediate number. The output tape will be written; upon completion, the display will light showing 0000 A2. Stop the output tape. Now hit GO to copy the next program.

SUPER-DUPE contains a Hypertape writing program which can be used independently; this starts at address 0100.

Basically, SUPER-DUPE saves you the work of setting up the SA, EA, and ID for each program, and the trouble of arranging the Hypertape writer into a part of memory suitable for each program.

```
0000 A2 03      START  LDX #3
0002 B5 E2      LOOP   LDA POINT2,X
0004 95 E0             STA POINT,X
0006 CA                DEX
0007 10 F9             BPL LOOP
0009 A9 00             LDA #0
000B 85 F6             STA CHKSUM
000D 85 F7             STA CHKHI
000F D8                CLD
0010 A9 07             LDA #7
0012 8D 42 17          STA SBD
0015 20 41 1A   SYN    JSR RDBIT
0018 46 F9             LSR INH
001A 05 F9             ORA INH
```

```
001C 85 F9          STA INH
001E C9 16    TST   CMP #$16 sync?
0020 D0 F3          BNE SYN
0022 20 24 1A       JSR RDCHT
0025 C6 F9          DEC INH
0027 10 F5          BPL TST
0029 C9 2A          CMP #$2A
002B D0 F1          BNE TST
002D 20 F3 19       JSR RDBYT
0030 85 F9          STA INH
0032 A2 FE          LDX #$FE neg 2
0034 20 F3 19 ADDR  JSR RDBYT
0037 95 FC          STA POINTH+1,X
0039 20 91 1F       JSR CHK
003C E8             INX
003D 30 F5          BMI ADDR
003F A2 02    BYTE  LDX #2
0041 20 24 1A DUBL  JSR RDCHT
0044 C9 2F          CMP #$2F eot?
0046 F0 15          BEQ WIND
0048 20 00 1A       JSR PACKT
004B D0 1C          BNE ELNK error?
004D CA             DEX
004E D0 F1          BNE DUBL
0050 81 E0          STA (POINT,X)
0052 20 91 1F       JSR CHK
0055 E6 E0          INC POINT
0057 D0 02          BNE OVER
0059 E6 E1          INC POINT+1
005B D0 E2    OVER  BNE BYTE
005D 20 F3 19 WIND  JSR RDBYT
0060 C5 F7          CMP CHKHI
0062 D0 05          BNE ELNK error?
0064 20 F3 19       JSR RDBYT
0067 C5 F6          CMP CHKSUM
0069 D0 95    ELNK  BNE START    (or 65?)
006B 20 1F 1F FLSH  JSR SCANDS
006E F0 FB          BEQ FLSH display SA,ID
0070 20 6A 1F       JSR GETKEY
0073 85 F5          STA GANG
0075 0A             ASL A
0076 F0 88          BEQ START
0078 8D BE 01       STA NPUL
007B 65 F5          ADC GANG
007D 8D C0 01       STA TIMG+1
0080 A9 27          LDA #$27 register mask
0082 85 F5          STA GANG
0084 A9 BF          LDA #$BF
0086 8D 43 17       STA PBDD
0089 A2 64          LDX #$64
008B A9 16          LDA #$16 sync
```

```
008D A2 64          LDX #$64     send 100
008F A9 16          LDA #$16     sync
0091 20 61 01       JSR HIC
0094 A9 2A          LDA #$2A     start char
0096 20 88 01       JSR OUTCHT
0099 A5 F9          LDA INH      write ID
009B 20 70 01       JSR OUTBT
009E A5 FA          LDA POINTL     start adds
00A0 20 70 01       JSR OUTBT
00A3 A5 FB          LDA POINTH
00A5 20 70 01       JSR OUTBT
00A8 A0 00    DATA  LDY #0
00AA B1 E2          LDA (POINT2),Y
00AC 20 70 01       JSR OUTBT       write data
00AF E6 E2          INC POINT2
00B1 D0 02          BNE SAMP        next addrs
00B3 E6 E3          INC POINT2+1
00B5 A5 E2    SAMP  LDA POINT2
00B7 C5 E0          CMP POINT
00B9 A5 E3          LDA POINT2+1
00BB E5 E1          SBC POINT+1
00BD 90 E9          BCC DATA        more data?
00BF A9 2F          LDA #$2F        eot
00C1 20 88 01       JSR OUTCHT
00C4 A5 F7          LDA CHKHI       checksum
00C6 20 70 01       JSR OUTBT
00C9 A5 F6          LDA CHKSUM
00CB 4C 54 01       JMP EXIT

00D0 4C 29 19       JMP LOADT9      FFFF option

00E2 00 02 00 02    data area; set as desired
```

****** Hex Dump Super - Dupe ******

```
0000-   A2 03 B5 E2 95 E0 CA 10 F9 A9 00 85 F6 85 F7 D8
0010-   A9 07 8D 42 17 20 41 1A 46 F9 05 F9 85 F9 C9 16
0020-   D0 F3 20 24 1A C6 F9 10 F5 C9 2A D0 F1 20 F3 19
0030-   85 F9 A2 FE 20 F3 19 95 FC 20 91 1F E8 30 F5 A2
0040-   02 20 24 1A C9 2F F0 15 20 00 1A D0 1C CA D0 F1
0050-   81 E0 20 91 1F E6 E0 D0 02 E6 E1 D0 E2 20 F3 19
0060-   C5 F7 D0 05 20 F3 19 C5 F6 D0 95 20 1F 1F F0 FB
0070-   20 6A 1F C9 07 B0 F4 85 F5 0A F0 84 8D BE 01 65
0080-   F5 8D C0 01 A9 27 85 F5 A9 BF 8D 43 17 A2 64 A9
0090-   16 20 61 01 A9 2A 20 88 01 A5 F9 20 70 01 A5 FA
00A0-   20 70 01 A5 FB 20 70 01 A0 00 B1 E2 20 70 01 E6
00B0-   E2 D0 02 E6 E3 A5 E2 C5 E0 A5 E3 E5 E1 90 E9 A9
00C0-   2F 20 88 01 A5 F7 20 70 01 A5 F6 4C 54 01 FF EA
00D0-   4C 29 19
00E0-         00 02 00 02
```

REMEMBER: You must also include HYPERTAPE! (page 119).

VERIFY TAPE

James Van Ornum

Do you want to verify the cassette tape you just recorded before the information is lost? Then follow this simple procedure:

1. Manually verify that the starting address ($17F5, $17F6), the ending address ($17F7, $17F8) and the block identification ($17F9) locations are correct in memory.

2. Enter zeros ($00) into CHKL ($17E7) and CHKH ($17E8).

3. Enter the following routine:

```
17EC  CD  00  00   VEB      cmp  START
17EF  D0  03               bne  failed
17F1  4C  0F  19           jmp  LOAD12
17F4  4C  29  19   failed   jmp  LOADT9
```

4. Rewind the tape, enter address $188C, press GO and playback the tape. If the tape compares, the LEDs will come back on with address $0000. If there is a discrepancy between memory and the tape, the LEDs will come on with address $FFFF.

VU-TAPE

Jim Butterfield

Program VUTAPE lets you actually see the contents of a KIM format tape as it's going by. It shows the data going by very quickly, because of the tape speed..but you can at least "sense" the kind of material on the tape.

In case of tape troubles, this should give you a hint as to the area of your problem: nothing? noise? dropouts? And you can prepare a test tape (see below) to check out the tape quality and your recorder. The test tape will also help you establish the best settings for your volume and tone controls.

Perhaps VUTAPE's most useful function, though, is to give you a "feeling" for how data is stored on tape. You can actually watch the processor trying to synchronize into the bit stream. Once it's synched, you'll see the characters rolling off the tape...until an END or illegal character drops you back into the sync mode again. It's educational to watch. And since the program is fairly short, you should be able to trace out just how the processor tracks the input tape.

VUTAPE starts at location 0000 and is fully relocatable (so you can load it anyplace it fits).

KIM UTILITY: VUTAPE

```
0000  D8          START   CLD
0001  A9 7F               LDA #$7F
0003  8D 41 17            STA PADD      set display dir reg
0006  A9 13       SYN     LDA #$13      ..window 6 and tape in
0008  85 E0               STA POINT     and keep pointer
000A  8D 42 17            STA SBD
000D  20 41 1A            JSR RDBIT     get a bit and
0010  46 F9               LSR INH       ..slip it into
0012  05 F9               ORA INH       ..the right-hand
0014  85 F9               STA INH       ..side:
0016  8D 40 17            STA SAD       show bit flow on display
0019  C9 16       TST     CMP #$16      ..is it a SYNC?
001B  D0 E9               BNE SYN       nope, keep 'em rolling
001D  20 24 1A            JSR RDCHT     yup, start grabbing
0020  C9 2A               CMP #$2A      ..8 bits at a time and..
0022  D0 F5               BNE TST       ..if it's not an "*"..
0024  A9 00       STREAM  LDA #$00      ..then start showing
0026  8D E9 17            STA SAVX      ..characters 1 at a time
0029  20 24 1A            JSR RDCHT
002C  20 00 1A            JSR PACKT     ..converting to hexadec..
002F  D0 D5               BNE SYN       ..if legal
0031  A6 E0               LDX POINT
0033  E8                  INX
0034  E8                  INX           Move along to next..
0035  E0 15               CPX #$15      ..display position
0037  D0 02               BNE OVER      (If last digit,..
0039  A2 09               LDX #$09      ..reset to first)
003B  86 E0       OVER    STX POINT
003D  8E 42 17            STX SBD
0040  AA                  TAX           change character read
0041  BD E7 1F            LDA TABLE,X   ..to segments and..
0044  8D 40 17            STA SAD       send to the display
0047  D0 DB               BNE STREAM    unconditional jump
```

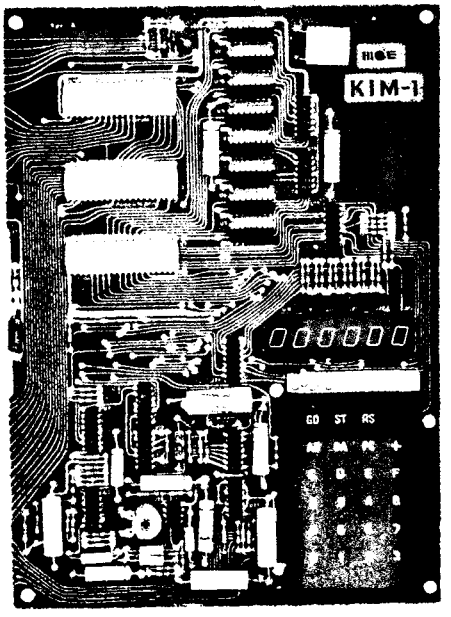
Checking Out Tapes/Recorders

Make a test tape containing an endless stream of SYNC characters with the following program:

```
0050  A0 BF       GO      LDY #$BF      directional..
0052  8C 43 17            STY PBOD      ...registers
0055  A9 16       LP      LDA #$16      SYNC
0057  20 7A 19            JST OUTCH     ...out to tape
005A  D0 F9               BNE LP
```

Now use the program VUTAPE. The display should show a steady synchronization pattern consisting of segments b,c, and e on the right hand LED. Try playing with your controls and see over what range the pattern stays locked in. The wider the range, the better your cassette/recorder.

EXPANDING YOUR KIM



EXPANDING YOUR KIM

Games and diversions using the keyboard and display are fine. Programming in assembly language can even be a lot of fun, once you get over the first few hurdles. But, sooner or later you are going to get the urge to have your KIM act like the "big machines". What do you have to add on? How much will it cost? How much trouble is it going to be? Let's look at a few of the options and you can decide for yourself.

Memory Expansion

If you only had more memory, you could do anything, right? Well, not exactly, but let's see what's involved in adding memory.

Computer buffs abreviate a thousand memory locations, more or less, with the letter K. Your KIM-1 has a 1K block of RAM and 2K of ROM. Provision is also built into the KIM-1 for easily adding an additional 4K of memory.

4K Expansion

If you want to add only 4K of memory, it's not especially difficult. An article in Kilobaud #4, (April '77), gives instructions for adding one of the lower priced 4K RAM kits. It is primarily a matter of connecting wires between the expansion connector on your KIM and the new board. Depending on the size of your present power supply, an additional supply may be required for the new board.

Further Expansion

Adding more than 4K of memory is a bit more difficult. Part of the problem has to do with address decoding. The expansion connector is essentially an extension of the main arteries of the computer, the address and data busses. These carry signals between the CPU and memory. The data bus carries information to or from a memory location specified by the address bus.

The "Central Processing Unit"' (CPU), on the KIM has the potential of addressing 64K however, so you can see that we have barely begun to scratch the surface.

Decoding

The complete address bus isn't available to each memory chip because there are just too many lines and not enough pins on the chips. Instead, there is some extra circuitry which looks

at the entire address bus and determines which block, (usually 1K blocks), of memory should be allowed to function. This is called decoding circuitry. Sub-addressing within blocks is handled by the lower address lines which are connected to all chips.

Decoding sufficient to select one of four 1K blocks already exists on the KIM and is brought out to the expansion connector. If you add more than 4K of memory, additional decoding will be required. Usually this is built into the memory board.

Buffering

If you start adding too many chips to the address and data busses, the extra circuits begin to "load down" the bus and cause it to not function properly. Additional boards are sometimes isolated from the main busses with circuits called "buffers" which prevent this from happening. Some memory boards have buffers built in.

Speed

Another problem you should be aware of has to do with how fast the CPU runs and how fast memory chips respond. Some CPU's have a wait state so that if the memory is a little slow in responding to entry or retrevial of information, the CPU can wait for it. The 6502 processor in KIM doesn't have this feature. This means that the memory used has to be fast enough to work with the processor. $\leq$ 450 ns

What Board?

We see then that memory expansion can get a little complicated. Further details are given in sections 3.2 and 6.1 of the Kim User's Manual. Perhaps the easiest way to get around these problems is to buy an assembled board made especially for the KIM. All decoding, buffering etc. should already have been taken care of in this case.

If you build from a kit, there are many solder connections that are very close to each other; it's easy to make mistakes. Kit or assembled board however, you should follow the instructions of someone who has already done it.

What does it cost?

Here's the good part! Memory prices have been dropping and are continuing to drop. Recently boards have been coming out using 4K memory chips which have more bits per chip than the older 1K RAM. This reduces the cost further, especially on boards having a lot of memory.

Any price quoted would soon be out of date and the price per byte depends heavily on the size of board you buy. A quick scan through a recent hobbyist publication should give you a rough idea of what to expect.

How Much Do You Need?

It depends primarily on what you want to do. Quite a bit can be done with just the 1K on the basic KIM-1. Even if you add a terminal, this 1K should be adequate for small games etc. written in assembly language. If you want to use a lot of text or go to a higher level language like Basic, you will have to expand. Exactly how much you need to expand depends on how elaborate your software is.

Motherboards

If you want to add more than just one board to the expansion connector of your KIM, you should start thinking in terms of a motherboard. A motherboard is a group of sockets connected in parallel. Buffering is also usually provided so the extra boards don't load the busses.

If you buy a motherboard specifically for the KIM-1, it will also have provision for letting KIM know when one of its boards is being addressed. This is so the decoding present on the KIM will be disengaged and not conflict with decoding on the expansion boards.

"Standard" Busses

The largest number of boards made for hobbyist use have a 100 pin configuration that plugs into the so-called "S-100" bus. MOS Technology also makes a motherboard for KIM with yet another bus. It should be possible to hook the KIM to motherboards made for other 8 bit machines too. One group is getting together an expansion board for KIM based on the standard 44 pin connector.

Once you decide on a particular motherboard, you are pretty much locked in to buying or building boards whose pins match those in the sockets of the motherboard.

"S-100" Bus

The S-100 bus derives from the Altair® motherboard. Presumably, any board which works in an Altair then should work in any other S-100 machine. Unfortunately, that has not always been the case. The S-100 bus is popular though and already a couple manufacturers have advertised S-100 motherboards meant to be attached to the KIM. Because of the competition, S-100 boards sometimes give a cost advantage. This is especially true in the case of memory boards where competition is fierce.

NOTE: Altair is a trademark of MITS, Inc.

A Caution

No matter what bus you decide on, you are going to need programs written for KIM to drive certain boards you might plug in. Unless there is a program for that particular board, written for KIM, you are in for a lot of work.

The Serial Port

It's not necessary that all expansion take place along the data and address busses of your KIM. There is another entrance/exit for information - the serial ports. The serial I/O, (Input and Output), ports also have the advantage that most of the required software already exists in the ROM of KIM. For example, to output a character, it is only necessary to put that character in the accumulator and jump to the subroutine OUTCH (1EA0). The character then comes spewing out the serial output port, bit by bit.

ASCII

The code that is used in this process is the "American Standard Code for Information Interchange", or ASCII for short. The hardware connection is also standardized and is made of two 20 milliamp current loops. The device to be connected to KIM should be set up for these standards. Connections are made as shown starting on page 17 of the Kim User's Manual.

The Teletype®

The serial ports were obviously set up with a particular machine in mind, the Teletype. The problem is that a new Teletype will cost over $1000 and used ones aren't much cheaper.

Baudot Machines

Older model Teletypes and some other makes of teleprinters go for $25 on up. The difference? These are Baudot machines. Where the modern Teletype uses a 8 bit (8 level) code to represent ASCII characters, the older machines use a 5 bit (5 level) code called Baudot. A good place to find out what is available etc. is a series of three articles appearing in the April, May and June '77 issues of Byte magazine.

Teleprinters are noisey, smelly and slow. What's more, the interface of a Baudot machine to your KIM is far from a trivial problem. Why then even bother with the teleprinter? One reason - it's great to have a hardcopy of your program, a piece of paper you can sit down and take a pencil to when something goes wrong.

Video Terminals

Also easily connected through the serial port are stand alone video terminals. These units contain a cathode ray T.V. tube,

Teletype is a trademark of Teletype Corp.

(CRT), keyboard and all necessary guts to display a large number of lines of characters on the screen at once. Common are 12 or 24 lines of 80 characters each. With 80 characters, a full 72 character Teletype line can be duplicated, making the unit indeed a "Glass Teletype".

Fewer Characters - Lower Price

The price of most video terminals is still up around $1000 even in kit form. One way to reduce the cost is to reduce the number of characters and display the results on an ordinary T.V. set. 16 lines of 32 or 64 characters are common.

This type of unit can be purchased as a video board alone or along with a keyboard in a nice case. If purchased seperately, you will also need a serial interface board.

Serial/Parallel Conversion

Remember that we had planned to use the serial I/O ports on KIM. The video board or the keyboard is more than likely hooked up to input or output in bytes, (parallel input or output). A whole byte appears on 8 seperate pins along with a timing pulse, called a strobe, on yet another pin. The strobe is used to indicate when data is valid. We have to convert this type of input or output to the sequential bit by bit information required by the serial port.

Luckily, there are chips designed especially to do this. They are called UART's and are found on serial interface boards. One such board was described in issue #1 of Kilobaud, (Jan. '77).

What to look for

Video boards vary considerably in the features they offer. The simplist boards begin writing characters in the upper left of the screen and continue on down the page. When the end of the last line is reached, they return to the upper left corner and start over. The only control you might have is a "home" signal which returns you to the starting point. Any carriage returns, linefeed etc. have to be taken care of by a program which is keeping track of exactly where you are.

A better scheme is to have a cursor which is usually a flashing or solid white square located where the next character will appear. In more advanced units, you can move this cursor around under software (or hardware) control. That way, it's easy to back up and go over any mistakes.

Another handy feature is scrolling. When you reach the end of the last line on the screen, it's a little confusing to have

the next line start at the top. Instead, some boards automatically push every line up to make room for the incoming line, (the top line goes off the screen).

Blank to end-of-line and blank to end-of-screen features are necessary to keep from having a lot of unwanted characters left on the screen. Be sure to check to find out exactly what features are included on the board you are buying. If you can, find someone who has a similar board up and running.

Back To The Busses

It's not manditory that a video board work off the serial ports. There are boards made to plug into most "standard" motherboards. These work off the data and address busses directly. In many cases, they include memory to hold the characters which looks just like any other memory to the processor. This has the advantage that any character can be changed instantaneously. A board like this is undoubtedly going to require software to keep things organized and you'll have to provide programs written especially for KIM.

Hardware vs Software

With the prices of memory continuing to drop, it's becoming cheaper to replace many hardware functions with software. In the case of video, you can use software not only to keep track of what characters go where; you can also use it to generate most of the display itself. This tends to reduce the cost considerably.

Using this fact, Don Lancaster describes a T.V. Typewritter addition to the KIM for $25-$35, (Kilobaud #6, June '77 or Popular Electronics, July '77 and August '77). But a word of caution. You'll have to "chop up" your KIM a bit to implement this- the project involves cutting a piece of KIM's printed circuit foil, plus wiring in a whole bunch of new wires. And while the changes don't affect KIM's operation, you have to recognize that memory expansion becomes a different ball game. Don uses the addresses from 2000 to EFFF, and that means that you can't just add on extra memory in those areas.

Dedicating the processor to running the display in this manner also means that it is going to have to "steal" time from this job to run your programs. This can slow things up a bit.

Keyboards

The keyboard also doesn't have to come into the serial port. Some video boards have a keyboard port built in. Another possibility is the parallel I/O ports on the KIM itself. Again, you'll have to provide the necessary software, but it would save you from having to buy a serial interface board.

If you are thinking of running both the keyboard and video board off the parallel ports of KIM, you should add up the total number of lines you need. By the time you include all necessary strobe lines, you will probably find you don't have enough ports available.

Hooking To Your T.V.

When you hook a video board to a T.V. set, make sure that the T.V. has a transformer which isolates the set from the A.C. line. 110 volts can ruin a lot of chips in a hurry!

There are two ways of putting the video signal in the T.V. If you want to go into the antenna terminals, you will need a board which generates a regular T.V. frequency signal with the video signal being imposed upon it. Kits are available for $10 - $15.

A method less susceptible to interference problems is to go directly into the video amplifier of the set. A T.V. repair shop should be able to handle this if you can't. About the simplest circuit was given in July '76 Byte, p. 38. Another appeared in Kilobaud #7, (July '77 p. 30). Kits are available to make this type of conversion also.

Video Monitors

A video monitor is like a T.V. set without the ability to pick up channels. It just takes a standard video signal (like the one coming from a video board) and puts it on the screen. Because they have a larger bandwidth than the normal T.V. set, they can display more information without the characters getting fuzzy.

Costs

At the present time, (Summer '77), you can expect to pay $150 - $250 for a video board, $50 - $150 for a keyboard and over $300 for the combination in a box along with a serial interface. Most of the serial interface is in the UART chip which sells for about $10. Kits may be available for about $25 - $50. Motherboards run $100 - $150 and a video monitor will cost around $150 - $200.

Graphics

If you want to use your KIM for simulating video games on a T.V., you should be thinking in terms of a graphics board. The graphics boards that are used with T.V. sets generate many tiny white rectangles, squares or dot patterns on the screen. these can be individually turned on or off at will. Some video boards meant to display characters also have limited graphics capability.

Printers

There are a number of printers on the market which use many small solenoids to form dot patterns through a typewritter ribbon onto paper. These dot patterns form characters faster than can be done with a typewritter or teleprinter. Some use adding machine paper and others, a standard size sheet. Prices run from $250 on up.

Also available are printers which use a specially sensitized paper and print using a thermal process.

Floppy Disks

Once you start reading in programs which require 4K or more of memory, you are going to find the cassette interface on your KIM a little slow. Even with Hypertape, it will take about 1 1/2 minutes to read in 4K.

There are faster tape units on the market, but the ultimate as far as the hobbist is now concerned is the "floppy". The floppy disk is like a flexable phonograph record coated with iron oxide as is used on tapes. A read/write head is moved radially outward from the center to read or write on different "tracks". The main advantage over tape is the speed at which any block of information can be located. The information is also put on very compactly and reading it back takes only a few seconds at most.

The mechanism to do all this is a precision piece of equipment and quite expensive. Prices are continuing to drop however as the demand becomes greater. The electronics necessary is also quite complex, but as with the UART, single chips are now being made which do most of the job.

Floppies are often used in pairs. One reason for this is to be able to back up what is stored on a disk. One disk is simply copied to another. Since each disk may store over 1/4 million bytes, you can see how time consuming this would be if you tried to read all information into memory and back out on another disk. Smaller versions of floppies using a 5" diskette (with less storage capacity) are also available at somewhat lower prices.

Again, you need not only the floppy drive and controller (electronics), but also the necessary software written for KIM. The operating system software that goes with floppies is quite complex. But then, it's also very powerful.

SOFTWARE TO EXPAND YOUR KIM

In addition to building extra devices onto your KIM system, like teletype, display, or more memory, you can increase the power of your system with special programs called software.

The name, software, is often misunderstood. Software, strictly speaking, refers to programs that help you do the job. They are helping programs, not doing programs. For example, if you write a program to play a game, that's not software - it's called an application program, for it actually does something. But the programs that help your game, such as the Monitor subroutines that you may call, are software. They don't do the job, but they sure help.

Most of the extra software that we'll talk about here will require extra memory to be fitted to your KIM system.

Assemblers

If you've tried writing a program, you may have noticed that converting your coding into KIM's machine language is quite a tedious job. For example, you may have written the command LDA TOTAL to load the accumulator with a zero page quantity that you have called TOTAL. Before you can enter the program, you must convert this to the 6502 code: A5 (for LDA from zero page), 63 (the zero page location you have chosen for TOTAL). Not too hard, perhaps; but you must look up the code and keep track of the addresses If your program contains dozens of instructions, this conversion - called hand assembly - can become quite a chore.

An assembler program will do the conversion for you, quickly, neatly, and without error. If you have a hard copy printing device, it will give you a complete printout (called a "listing") of your program.

A resident assembler works on program data held entirely within KIM's memory. It's very fast, but it does need lots of memory to hold all of your program information. Other assemblers work from data stored on magnetic tape or on floppy disk. They are slower, since the data must be copied into memory as it's needed, but allow your programs to be almost unlimited in size. A cross-assembler will assemble your KIM program on a completely different machine, such as a Digital Equipment Corporation PDP-11 or a commercial time-sharing processor. Because these other computers are not so limited in size compared to the KIM, they can be very powerful.

Dis-Assemblers

A disassembler works in reverse from an assembler. If you have a program in KIM machine language, the disassembler will print it out in the more easily readable assembly language . Very handy for investigating a working program, if you don't have the listing.

For example, if you have coding starting at address 020F that reads: CA 10 F8 AD 04 17 85 80 ... , the disassembler would print something like this:

```
020F  CA          DEX
0210  10 F8       BPL 020A
0212  AD 04 17    LDA 1704
0215  85 80       STA 0080  ...
```

As you can see, this is much more readable.

Interpreters (BASIC,FOCAL, etc.)

There are several "high level" languages that are much easier for writing programs than KIM (6502) machine language. With the proper software package, KIM can translate these high level instructions and perform the desired actions. The translation job takes time, so KIM will run many times slower than its normal "machine" speed. Programming convenience is so great, however, that most users don't mind the loss of speed.

Interpreters can take up quite a bit of memory - anywhere from 2K to 16K locations - so you'll have to be fitted with the appropriate amount of memory expansion. If you hear of an 8K Basic interpreter, you'll know that means 8,000 locations for the program; and of course you'll need to provide extra memory to fit your own programs in.

A brief example will show how simple a language like BASIC can be for programming. To input a number from your keyboard, and type its square, you need only write:

```
50  INPUT A                receive value "a" from keyboard
60 LET B = A*A              "*" means multiplication
70 PRINT "THE SQUARE OF ";A;" IS ";B
80 STOP
```

See how easy it is? KIM must read each line, character by character, decide what it means: inputting, calculating, printing or whatever, and then perform that action. KIM works hard, but you don't.

Text Editors

It can be very handy to compose a number of lines of material such as a letter, a program, or general data; put it into your KIM system; save it permanently on tape or disk; and then later recall it and change, insert or delete information.

If you're writing a letter, you can correct mistakes and insert new thoughts as they occur to you, perhaps even generating several slightly different versions to mail to various people. If you have a program, you can correct bugs as you find them and insert new coding as needed Data files can be kept up to date.

Text Editors are very important with other software such as assemblers and interpreters; often, they are built in.

Mathematical Packages

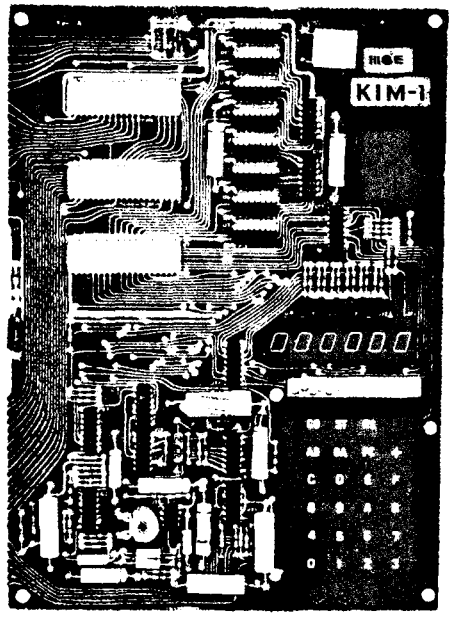
Each memory location in KIM can store a number from 0 to FF hexadecimal, or 0 to 255 decimal. Ther are no fractions, and you have to make special arrangement for signed (positive and negative) numbers. You can link memory locations together to hold larger numbers; but extremely large numbers and fractions call for special mathematical techniques to be used. In addition, KIM gives you only addition and subtraction; you have to work out multiplication and division for yourself, to say nothing of more complex functions like square roots and powers.

You can program all this yourself, if you have the time and the mathematical background. But if you really need to perform advanced math on your KIM, you'll be better off to obtain a pre-written mathematical package.

Floating-point on computers means about the same as the term "Scientific Notation" on calculators. It lets you use fractions and deal with very large and very small values . In addition, you'll often get extra functions - powers, roots, logarithms, and trigonometric functions such as sines and cosines.

Many mathematical functions are often included in large interpreters.

CONNECTING TO THE WORLD



KIM RUNS THE WORLD OR HOW TO CONNECT YOUR MICROPROCESSOR TO EXTERNAL DEVICES

By Cass Lewart

Introduction - Calculator versus Computer

Most of you are familiar with the ubiquitous pocket calculator. From the simple "four-banger" to the most sophisticated card-programmable, the sequence of operations is always the same. You enter numbers from either the keyboard or a program card, depress a few keys, the calculator "crunches" your input and out come the processed numbers on the display or printer.

Though a calculator will do a great job of processing numbers, just try to make it perform a simple trick of a different kind - e.g., ring a bell after completing the 150th iteration. No way! A calculator is a closed system. In general it is not possible to attach to it external devices not envisioned during the original design. A microprocessor such as KIM is quite different in this respect. In fact frequently its main functions are not to "crunch" numbers but to receive signals from various sensors such as photocells, thermostats, switches or pressure transducers, to do a small amount of processing of these inputs and then to control devices such as lights, motors, relays or even to play music.

In this chapter we will try to show you how easy it is for KIM to perform operations of the type described. KIM via its input/output ports can receive and transmit control signals. Its built-in precision quartz crystal controlled time reference and a built-in interval timer further simplify various controlling tasks.

KIM Ports - KIM Talks and Listens

KIM has four special memory locations which are used for input, output and various applications. Great things happen if you store numbers in these locations!

Location	
1700	Contents of Application Port A
1701	Data Direction of Port A
1702	Contents of Application Port B
1703	Data Direction of Port B

The data contents locations 1700 and 1702 store the data transmitted to or from KIM while the data direction locations 1701 and 1703 determine which port operates in the input and which in the output mode. These four special memory locations can be accessed by KIM programs in the same way as any other location. In addition the application port A in location 1700 and the application port B in location 1702 are also accessible on connector pins. They represent the physical interface of KIM. By monitoring the appropriate pins with a voltmeter one can detect the data stored in memory locations 1700 and 1702 when KIM is in the output mode. By setting the appropriate pins to ground or to V_{CC} (+5 Volts) one can feed data into KIM in the input mode.

As KIM is an 8-bit microprocessor, each of the two ports A and B actually consists of eight independent inputs or outputs. Each of the eight bit positions from 0 through 7 appears on a different connector pin and is a port in itself. The following are connector pin assignments for the A and B application ports. For example PA0 represents the 0-th or the least significant bit of port A and PA7 the 7-th or the most significant bit. Pin A-14 means Application connector (lower left), the 14-th pin counting from the top, on the upper side of the connector (the lower side of the connector is designated by letters instead of numbers).

Connector Pin Assignments

Port	Pin	Port	Pin
PA0	A-14	PB0	A-9
PA1	A-4	PB1	A-10

Port	Pin	Port	Pin
PA2	A-3	PB2	A-11
PA3	A-2	PB3	A-12
PA4	A-5	PB4	A-13
PA5	A-6	PB5	A-16
PA6	A-7	PB6	Not accessible
PA7	A-8	PB7	A-15

To assign any of the above connector pins to either input or output mode we have to store a "magic" number in location 1701 to control port A or in location 1703 to control port B. A "1" stored in a specific bit position makes the corresponding port into an output, a "0" into an input. For example, to assign PA7 to output and PA0 through PA6 to input requires storing 10000000 or 80_{hex} in location 1701. In the following example although we deal only with port A, all the remarks apply equally to the port B.

Example - Burglar Alarm

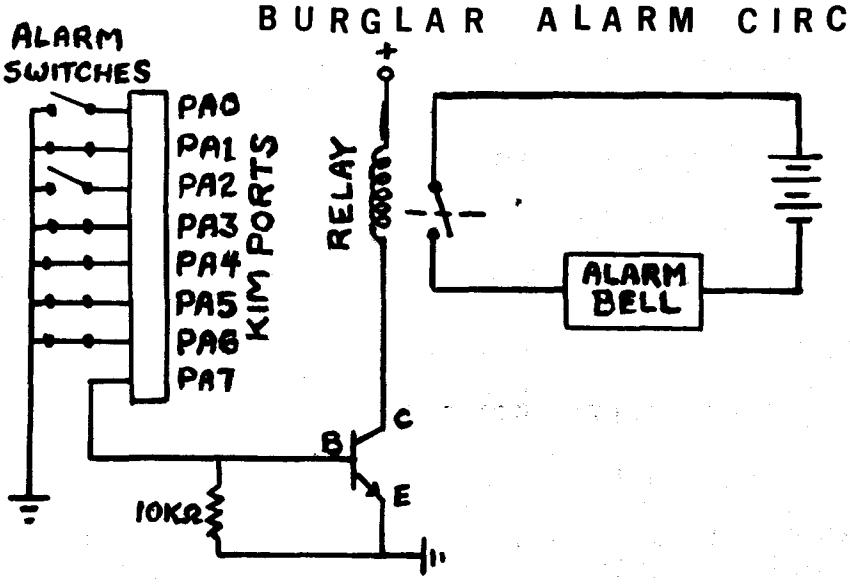
Let's suppose that we want to design a system under KIM control such that PA0 through PA6 are connected to seven normally closed burglar alarm switches while PA7 should control a warning bell. We want the bell to start ringing as soon as one of the contacts opens. The bell should keep ringing even if the contact closes again. We will first describe the software, or the programming part of the problem, and then will show you the actual circuit. We assume that by now you scanned through the KIM software chapters and are familiar with its basic instruction set.

Burglar Alarm Program

Loc	Code	Mnemonic	Comments
00	A9 80	LDA #80	Set PA0 through PA6 to
02	8D 01 17	STA 1701	input and PA7 to output
05	A9 00	LDA #00	Set output to 0
07	8D 00 17	STA 1700	Will affect PA7 only
0A	AD 00 17	LDA 1700	Read 1700 to find if PA0
0D	29 7F	AND #7F	through PA6 contain all
0F	C9 7F	CMP #7F	"1"s (closed switches)
11	F0 F7	BEQ 0A	All are closed, go to 0A
13	A9 80	LDA #80	At least one switch open,
15	8D 00 17	STA 1700	sound alarm
18	4C 13 00	JMP 0013	Stay in the loop

Now let's look at the simple circuit to operate our burglar alarm. We connect PA0 through PA6 pins directly to the switches. If a switch is closed then the voltage at that port is 0 Volts (ground); as soon as the switch opens, an internal resistor located on the KIM board "pulls" the port to the positive voltage V_{CC} of 5 Volts. All ports except PB7 are equipped with built-in resistors, called "pull-up" resistors connected to V_{CC}, which set voltage at a port to V_{CC} when the port is in the input mode and is not connected to ground. On the output port PA7 is connected to the base of an amplifying transistor which drives a relay to operate an alarm bell. The transistor is necessary because the maximum available current of each KIM port is only on the order of 1 mA. This current would not be sufficient to drive a relay directly.

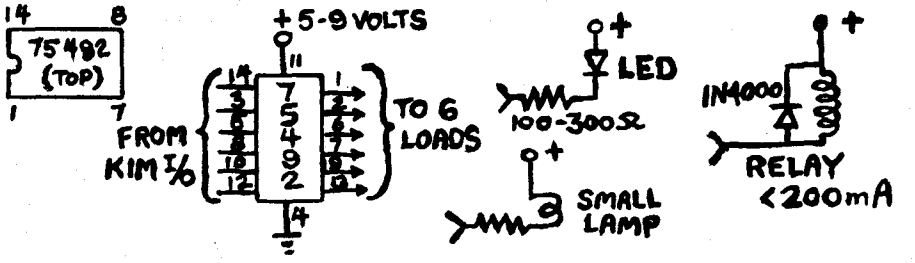
BURGLAR ALARM CIRCUIT



Multiple Drives

Now suppose you want KIM to drive several devices rather than a single one. For example you may want to connect a 3 x 3 matrix of LED lights to the A and B ports to play tic-tac-toe. The simplest way to do this is by using one of the inexpensive digit driving ICs, such as 75492 used in many calculator circuits. Each of these ICs will drive up to 6 lights, relays or what have you with the simple circuit shown below. The six IC outputs act as "sinks", which requires that you connect one side of your electric load to the positive battery voltage and the other side to one of the IC outputs. When the appropriate port is "on" current will flow through your load; when the port is "off", current will stop. The maximum current through each load is 200 mA.

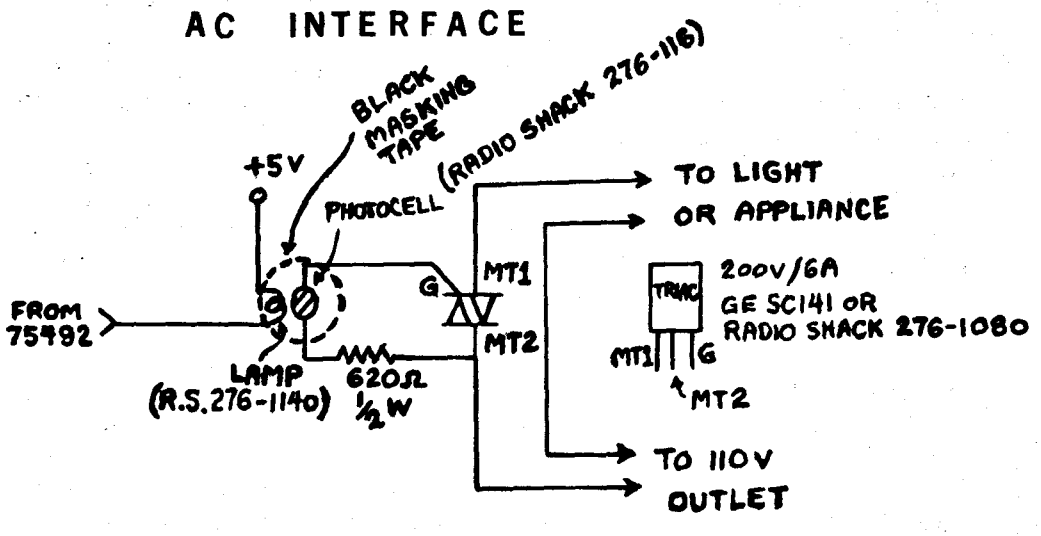
MULTIPLE KIM INTERFACE



AC Control

To go one step further we can show you how KIM can operate AC devices without relays. However we would like to caution you that the power line voltage of 110 Volts AC and the low voltages in your KIM do not mix easily. You may even achieve a non-voluntary beautiful pyrotechnic display. In other words, if you are not careful in your soldering techniques and like to leave a few wires dangling "just in case" we would recommend that you skip the following paragraph.

The circuit we show here electrically separates KIM from the power line by means of a lamp/photocell interface. The amplified voltage from one of the KIM ports turns on an incandescent lamp or an LED which lowers the resistance of a photocell which then turns on the electronic TRIAC switch. This simple and inexpensive circuit can easily control an AC lamp or appliance of up to 600 Watts.



KIM versus Hardwired Logic

We have showed you how KIM can control relays, lights and AC operated devices but these applications hardly tap KIM's capabilities. With the same methods you can also switch tracks on a model train layout, control traffic lights, and keep your fans and air conditioners going. The beauty of performing such tasks with a computer rather than with hardwired relay logic is that logical responses and changes in rules can easily be implemented by changing a few statements in your program. A redesign of a hardwired circuit on the other hand is always difficult, time consuming, frequently impossible without starting your design from scratch.

D/A and A/D Converters

So far we have discussed on/off type controls such as switches or relays which are either open or closed. However, there are many areas where a proportional control with "shades of gray" instead of black or white would be more desirable. For example if you are interested in electronic music you would like to shape the electric signals driving your amplifiers and speakers into sinusoids, triangles and seesaws to mimic various instruments. Though even with a simple on/off control you can create sounds, their acoustical range is very limited. If you connect an audio amplifier to one of the KIM ports and listen to the sound generated by the 5 Volt pulses of various length and at various repetition rates the sound will remind you only of a variety of buzz saws and not of musical instruments. The next step therefore is to develop a digital-to-analog (D/A) interface for your KIM. Such an interface will, for example, translate an 8-bit binary number on ports A0 through A7 into a voltage proportional to the numerical value stored in location 1700 (Port A). A number FF_{hex} stored in 1700 could then generate 2.0 Volts, while 20_{hex} stored in the same location would generate (32/255) x 2.0 = 0.25 Volts. Though we will not describe a D/A converter in detail, it can easily be built with either separate amplifiers or with specially designed ICs. An example of a relatively inexpensive converter is MC1408L by Motorola.

Similarly an analog-to-digital (A/D) converter interface can be used to turn KIM into a measuring instrument such as a digital voltmeter, thermometer or even a speech recognizer. Applications of a microprocessor equipped with D/A and A/D converters are limited only by your imagination and by your wallet.

Interval Timer

Many applications which interface KIM to the outside world benefit from the addition of a timer. For example, you may want the train in a model train layout to stop for exactly 45 seconds at a station under some conditions but for only 30 seconds under other conditions. For this and other purposes as well, KIM has a built-in interval timer which can be set to various multiples of its crystal controlled cycle time of 1 microsecond (10^{-6} sec.). By storing a number K between 1 and FF_{hex} in one of the special memory locations listed below we direct the timer to count a specific number of cycles. The special memory locations used by the interval timer and the longest count-down period are as follows:

Location	Timer Count (microseconds)	Max. Period (sec.) For K = FF_{hex}
1704	K x 1	0.000255
1705	K x 8	0.002
1706	K x 64	0.016
1707	K x 1024	0.26

Location 1707 is also used to sense that the timer has finished counting. By putting the interval timer inside a loop the timing can be lengthened to seconds, minutes and hours. The timer starts counting as soon as a number between 1 and FF_{hex} is stored in one of the above four locations by means of the STA (STore Accumulator in memory) instruction. When time runs out the BIT (test BITs in memory with accumulator) instruction returns a non-positive value from location 1707.

Timer Example

The following short program illustrates the use of the interval timer. The program will leave the loop after 5 x 64 = 320 microseconds count is detected by the BIT instruction. While the timer counts, other tasks can be performed by KIM.

Loc	Code	Mnemonic	Comments
00	A9 05	LDA #05	Start timer by storing
02	8D 06 17	STA 1706	5 in 1706
05	.....		Perform other tasks
	.....		
10	2C 07 17	BIT 1707	Check if timer finished?
13	10 F0	BPL 05	If still counting, go to 05
15	.....		Otherwise continue

How KIM Communicates with its own Keyboard and Display

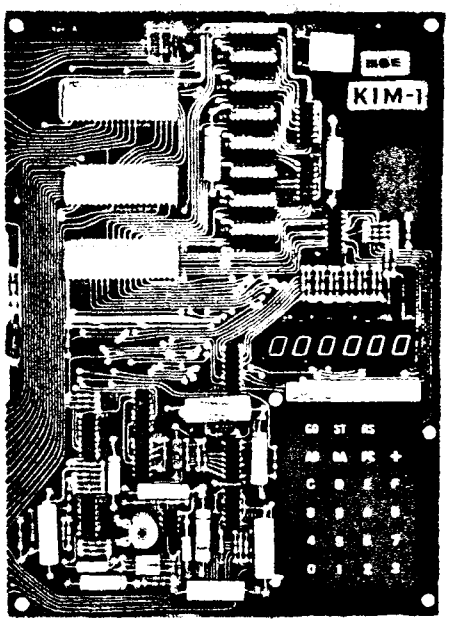
At first glance the KIM keyboard and the LED display seem to be a hardwired fixed part of the microprocessor and as difficult to access as if they would belong to a calculator. Fortunately it is not so. Both the keyboard and the display can be used quite differently from the way they are used by the KIM built-in operating system program. You can run the display and the keyboard under the control of your own programs to perform all kinds of tricks. For example, you can program the LEDs to display any pattern in any digit position which can be made with the seven LED segments. Similarly the keyboard can be used as input to various programs with individual keys performing functions unrelated to their numerical labels. For example, the "B" key in your program can

indicate a "Backward" command, while the "F" key can mean "Forward". Various game programs shown in other sections of this book are examples of such applications.

We have tried in this chapter to give you a feeling for what KIM can do in the way of control applications. We hope that by now you have gained some appreciation for KIMs potential.

****

POTPOURRI



GUIDELINES FOR WRITING KIM PROGRAMS

1. Use of Memory.

   --Wherever possible, place your programs in pages 2 and 3 -- addresses 0200 to 03FF. It's handy to keep page zero for variables - values that change during program run; and page one is best left alone because the program Stack uses it. The Stack, by the way, only uses a few locations - usually. But a small program error can sometimes make the stack run wild, which would destroy your page one data.

   --Your variables (changeable data) should be kept in page zero, in locations 0000 to 00EE. These addresses are easy to use, since you can use zero-page addressing modes which save you time and memory.

2. Program and constants.

   --Set up your programs in the following pattern: first, the main program (starting at address 0200 or higher); then your subroutines; and finally your data. Keep them all fairly close together, so that when you dump the whole thing to cassette tape it won't take extra time to write the 'blank spaces in between'.

3. Initial values.

   --Don't assume anything about the beginning values in your registers or in zero page. If you want to be out of decimal mode (and you usually do), make your first command a CLD (D8). If you want the accumulator to be zero, load it with LDA #$00 (A9 00). Every zero page variable that needs to start at a certain value should be set to that value by the program. For example, if you want address 0043 to start out with a value of 7, write LDA #$07, STA 0043 (A9 07 85 43).

4. General.

   --Make your subroutines simple, with clearly visible entry and return points. One of the stickiest problems to find is a subroutine that doesn't return via a RTS command, but instead jumps straight back to your main coding ... or a subroutine that you somehow get into without giving the vital JSR command.

   --Avoid super clever programming, such as having the program change itself. (It can work ... but if it misbehaves, you can have a bad time).

5. Remember: Computers are dumber than humans, but smarter than programmers.

LIGHTING THE KIM-1 DISPLAY — Jim Butterfield

A. SIX-DIGIT HEXADECIMAL.

The easiest way to display six digits of data is to use the KIM-1 Monitor subroutine SCAND.

Calling JSR SCAND (20 19 1F) will cause the first four digits to show the address stored in POINTL and POINTH (00FA and 00FB), while the last two digits of the display show the contents of that address.

If you look at the first three lines of subroutine SCAND (lines 1057 to 1059 on page 25 of the listing), you'll see how the program 'digs out' the contents of the address given by POINTL/POINTH and stores it in location INH (00F9). It's neat programming, and worth studying if you're not completely familiar with the 6502's indirect addressing operation.

Thus, if you skip these three lines, and call JSR SCANDS (20 1F 1F) you will be displaying, in hexidecimal, the contents of three locations: POINTH, POINTL, and INH This, of course, takes six digits.

To recap: SCAND will display four digits of address and two digits of contents. SCANDS will display six digits of data.

Important: in both cases, the display will be illuminated for only a few milliseconds. You must call the subroutine repeatedly in order to obtain a steady display.

B. DRIVING THE BITS OF THE DISPLAY DIRECTLY.

1. Store the value $7F into PADD (1741). This sets the directional registers.

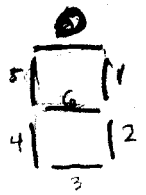
2. To select each digit of the display, you will want to store the following values in location SBD (1742):

Digit 1:	$09
Digit 2:	$0B
Digit 3:	$0D
Digit 4:	$0F
Digit 5:	$11
Digit 6:	$13

Note that this can easily be done in a loop, adding two to the value as you move to the next digit.

3. Now that you have selected a particular digit, light the segments you want by storing a 'segment control' byte into location SAD (1740). The segments will be lit by setting the appropriate bit to 1 in SAD according to the following table:

Bit:	7	6	5	4	3	2	1	0
	..	center	upper left	lower left	bottom	lower right	upper right	top
		"g"	"f"	"e"	"d"	"c"	"b"	"a"



For example, to generate a small letter 't', we would store $78 (center, upper left, lower left, bottom) into SAD.

4. Now that you have picked a digit and lit the appropriate segments, wait a while. Sit in a delay loop for about 1/2 millisecond before moving on to the next digit.

THE KIM-1 ALPHABET.

Some letters, like M and W, just won't go onto a 7-segment display. Some, like E, are only possible in capitals; others, like T, can only be done in lower case. So here's an alphabet of possibles:

A - $F7			
B - $FF	b - $FC		
C - $B9	c - $D8		
D - $BF	d - $DE		
E - $F9			
F - $F1	f - $F1		
G - $BD	g - $EF		
H - $F6	h - $F4	1 - $86	
I - $86	i - $84	2 - $DB	
J - $9E	j - $9E	3 - $CF	
L - $B8	l - $86	4 - $E6	
	n - $D4	5 - $ED	
O - $BF	o - $DC	6 - $FD	
P - $F3	p - $F3	7 - $87	
	r - $D0	8 - $FF	
S - $ED		9 - $EF	
	t - $F8	0 - $BF	
U - $BE	u - $9C	minus - $C0	
Y - $EE	y - $EE		

The following is reprinted from the KIM-1 User Manual with permission from MOS Technology.

Interval Timer

1. Capabilities

The KIM-1 Interval Timer allows the user to specify a preset count of up to 256_{10} and a clock divide rate of 1, 8, 64, or 1024 by writing to a memory location. As soon as the write occurs, counting at the specified rate begins. The timer counts down at the clock frequency divided by the divide rate. The current timer count may be read at any time. At the user's option, the timer may be programmed to generate an interrupt when the counter counts down past zero. When a count of zero is passed, the divide rate is automatically set to 1 and the counter continues to count down at the clock rate starting at a count of FF (-1 in two's complement arithmetic). This allows the user to determine how many clock cycles have passed since the timer reached a count of zero. Since the counter never stops, continued counting down will reach 00 again, then FF, and the count will continue.

2. Operation

a. Loading the timer

The divide rate and interrupt option enable/disable are programmed by decoding the least significant address bits. The starting count for the timer is determined by the value written to that address.

Writing to Address	Sets Divide Ratio To	Interrupt Capability Is
1704	1	Disabled
1705	8	Disabled
1706	64	Disabled
1707	1024	Disabled
170C	1	Enabled
170D	8	Enabled
170E	64	Enabled
170F	1024	Enabled

b. Determining the timer status

After timing has begun, reading address location 1707 will provide the timer status. If the counter has passed the count of zero, bit 7 will be set to 1, otherwise, bit 7 (and all other bits in location 1707) will be zero. This allows a program to "watch" location 1707 and determine when the timer has timed out.

c. Reading the count in the timer

If the timer has not counted past zero, reading location 1706 will provide the current timer count and disable the interrupt option; reading location 170E will provide the current timer count and enable the interrupt option. Thus the interrupt option can be changed while the timer is counting down.

If the timer has counted past zero, reading either memory location 1706 or 170E will restore the divide ratio to its previously programmed value, disable the interrupt option and leave the timer with its current count (not the count originally written to the timer). Because the timer never stops counting, the timer will continue to decrement, pass zero, set the divide rate to 1, and continue to count down at the clock frequency, unless new information is written to the timer.

d. Using the interrupt option

In order to use the interrupt option described above, line PB7 (application connector, pin 15) should be connected to either the IRQ (Expansion Connector, pin 4) or NMI (Expansion Connector, pin 6) pin depending on the desired interrupt function. PB7 should be programmed as in input line (its normal state after a RESET).

NOTE: If the programmer desires to use PB7 as a normal I/O line, the programmer is responsible for disabling the timer interrupt option (by writing or reading address 1706) so that it does not interfere with normal operation of PB7. Also, PB7 was designed to be wire-ORed with other possible interrupt sources; if this is not desired, a 5.1K resistor should be used as a pull-up from PB7 to +5v. (The pull-up should NOT be used if PB7 is connected to NMI or IRQ.)

****

IMPORTANT!!

The KIM Cassette Tape Interface

The KIM-1 USER GUIDE doesn't emphasize one vital instruction in telling you how to read and write tapes.

BEFORE READING OR WRITING MAGNETIC TAPE, BE SURE TO SET THE CONTENTS OF ADDRESS 00F1 TO VALUE 00.

This ensures that the computer is not in Decimal Mode. The key sequence is AD 0 0 F 1 DA 0 0 AD.

If you forget to do this, you're likely to have trouble with audio tape. You might write bad tape - which can never be read back in correctly; and you might find yourself unable to input properly from tape. Many of us have run into this problem, and have wasted countless hours trying different tapes and recorders or even investigating KIM's electronics.

You'll find KIM audio tape to be 100% reliable, even on inexpensive recorders, providing you follow this rule and always ensure that location 00F1 is set to zero.

NOTES ON A RANDOM NUMBER GENERATOR

Jim Butterfield

It's not my original idea - I picked up it from a technical journal many years ago. Wish I could remember the source, so I could credit it.

This program produces reasonably random numbers, and it won't "lock up" so that the same number starts coming out over and over again. The numbers are scattered over the entire range of hexadecimal 00 to FF. A Statistician would observe that the numbers aren't completely "unbiased", since a given series of numbers will tend to favor odd or even numbers slightly. But it's simple, and works well in many applications.

Here's how it works. Suppose the last five random numbers that we have produced were A, B, C, D and E. We'll make a new random number by calculating A + B + E + 1. (The one at the end is there so we don't get locked up on all zeros). When we add all these together, we may get a carry, but we just ignore it. That's all. The new "last five" will now be B, C, D, E and the new number. To keep everything straight, we move all these over one place, so that B goes where A used to be, and so on.

The program:

```
xxxx D8       RAND CLD  clear decimal if needed
xxxx 38            SEC  carry adds value 1
xxxx A5 13         LDA RND+1  last value (E)
xxxx 65 16         ADC RND+4  add B (+ carry)
xxxx 65 17         ADC RND+5  add C
xxxx 85 12         STA RND    new number
xxxx A2 04         LDX #4     move 5 numbers
xxxx B5 12    RPL  LDA RND,X
xxxx 95 13         STA RND+1,X  ..move over 1
xxxx CA            DEX
xxxx 10 F9         BPL RPL    all moved?
```

The new random number will be in A, and in RND, and in RND+1. Note that you must use six values in page zero to hold the random string ... I have used 0012 to 0017 in the above coding.

You often don't want a random number that goes all the way up to 255 (Hexadecimal FF). There are two ways of reducing this range. You can AND out the bits you don't want; for example, AND #$7 reduces the range to 0-7 only. Alternatively, you can write a small divide routine, and the remainder becomes your random number; examples of this can be seen in programs such as BAGELS.

The one publication that devotes all of its space to the KIM-1/6502 machines is:

KIM-1/6502 USER NOTES
109 Centre Ave.,
W. Norriton PA 19401

Six issues of this bimonthly newsletter costs U.S.$5.00 for North American subscribers and U.S.$10.00 for international subscribers.

Here's some pointers to other KIM-1/6502 articles-

BYTE-

November 1975 (p.56) - Son Of Motorola
- A description of the 6502 instruction set and comparison with the 6800.

May 1976 (p.8) - A Date With KIM
- An in depth description of KIM

August 1976 (p.44) - True Confessions: How I Relate To KIM
- How to; use cheap memories with KIM by stretching the clock; expand memory; implement interrupt prioritizing logic; simulate a HALT instruction.

March 1977 (p.36) - 6502 op code table

March 1977 (p.70) - Simplified Omega Receiver Details
- Using the 6502 for signal processing in a low cost navigation receiver (Mini-Omega).

April 1977 (p.8) - Kim Goes To The Moon
- A real-time lunar lander program for KIM

April 1977 (p.100) - Navigation With Mini-O
- Software details for a phase-tracking loop filter using Jolt or KIM.

June 1977 (p.18) - Designing Multichannel Analog Interfaces
- Hardware and 6502 software for an 8 channel analog I/O.

June 1977 (p.46) - Teaching KIM To Type
- Hardware and software for hooking KIM up to a Selectric.

June 1977 (p.76) - Come Fly With KIM
- Hardware and software for interfacing a Fly Paper Tape Reader to KIM.

July 1977 (p.126) - Giving KIM Some Fancy Jewels
- How to outboard KIM's seven-segment displays.

DR. DOBBS-

March 1976 (p.17) - 6502 Breakpoint Routine

August 1976 (p. 17) - 6502 Floating Point Routine

August 1976 (p.20) - Monitor For The 6502

August 1976 (p.21) - Lunar Lander For The 6502

September 1976 (p.22) - 6502 Disassembler

September 1976 (p.26) - A 6502 Number Game

September 1976 (p.33) - 6502 String Output Routine

November 1976 (p.50) 6502 String Output Routine

November 1976 (p.57) - 6502 Floating Point Errata

February 1977 (p.8) - More 6502 String Output Routine

INTERFACE AGE-

September 1976 (p.14) - A 6502 Disassembler

October 1976 (p.65) - Interfacing The Apple Computer
- How to: hook a SWTPPR-40 to the Apple 6502.

November 1976 (p.12) - Build A Simple A/D
- Hardware and 6502 software for simple joystick (or whatever) interface.

November 1976 (p.103) - Floating Point Routine For 6502

April 1977 (p.18) - "Mike"-A Computer Controlled Robot
- Hardware and 6502 software for a KIM controlled robot like vehicle.

KILOBAUD-

January 1977 (p.114) - A Teletype Alternative
- How to: Convert a parallel input TVT to serial operation; interface to KIM.

February 1977 (p.8) - Found: A Use For Your Computer

April 1977 (p.74) - KIM-1 Memory Expansion
- How to: Add an $89.95 4K Ram board to KIM.

May 1977 (p.98) - Adding "PLOP" To Your System
- A 6502 noisemaker for computer games.

June 1977 (p.50) - A TVT For Your KIM

NOTE: Kilobaud now has a monthly KIM column.

MICROTREK-

August 1976 (p.7) - KIM-1 Microcomputer Module
- A very in depth look inside KIM.

POPULAR ELECTRONICS-

July 1977 (p.47) - Build The TVT-6
- How to: KIM-1 TVT (same as Kilobaud #6).

73 MAGAZINE
January 1977 (p.100) - Bionic Brass Pounder
- How to: Turn KIM into a smart morse code keyboard.

****

6502 SOFTWARE SOURCES
(as of summer 1977)

ARESCO 314 Second Ave. Haddon Hts., New Jersey 08035	Focal, 2 1/2K assembler 6K assembler/text editor (send S.A.S.E. for info)
The Computerist P.O. Box 3 S. Chelmsford MA 01824	Please Package, Help, editor and mailing list packages (send S.A.S.E. for info)
Itty Bitty Computers P.O. Box 23189 San Jose, Calif. 95153	Tom Pittman's Tiny Basic (send S.A.S.E. for info)
MICROWARE 27 Firstbrooke Rd. Toronto, Ontario CANADA M4E 2L2	MICROCHESS, (Chess in 1k), assembler (send S.A.S.E. for info)
MICRO-SOFTWARE SPECIALISTS P.O. Box 3292 E. T. Station Commerce, Texas 75428	2K assembler/editor (send S.A.S.E. for info)
6502 Program Exchange 2920 Moana Lane Reno, Nevada 89509	Focal, Focal programs, Kim and TIM programs (send 50¢ for program list)
Pyramid Data Systems 6 Terrace Ave. New Egypt, New Jersey 08533	1K monitor system. (send S.A.S.E. for info)
Julien Dubé 3174 Rue Douai Ste-Foy, Quebec G1W 2X2 Canada	Baudot Monitor (send S.A.S.E.)

Jim Butterfield
14 Brooklyn Avenue
Toronto, Ontario, Canada
M4M 2X5

Charles Eaton
19606 Gary Avenue
Sunnyvale, California
94086

Lew Edwards
1451 Hamilton Ave.
Trenton 9, N.J. 08629

Peter Jennings
27 Firstbrooke Rd.
Toronto, Ontario Canada
M4E 2L2

Ron Kushnier
3108 Addison Ct.
Cornwells Hts., Penna.
19020

Cass Lewart or
Dan Lewart
12 Georjean Drive
Holmdel, N.J. 07733

Stan Ockers
R.R.#4, Box 209
Lockport, Ill.
60441

James Van Ornum
55 Cornell Drive
Hazlet, N.J.
07730

Charles Parsons
80 Longview Rd.
Monroe, Conn.
06468

Jim Pollock
6 Terrace Ave.
New Egypt, New Jersey
08533

Eric Rehnke
109 Centre Ave.
W. Norriton PA
19401

Joel Swank
#186
4655 S.W. 142nd
Beaverton, Ore. 97005

****

Here are the folks responsible. They eagerly await your praise, comments, criticism, indignation - whatever... Please do the courtesy of enclosing a self-addressed stamped (if possible) envelope (SASE) if you wish a reply.

5119-0
$8.95

THE FIRST BOOK OF KIM

JIM BUTTERFIELD, STAN OCKERS, and ERIC REHNKE

Here is a step-by-step guide that will take you through the fundamentals of writing KIM programs. This beginner's guide includes dozens of examples of programs that are run on a basic KIM-1 system. These programs include games and puzzles such as Blackjack, Chess Clock, Horserace, Lunar Lander, Music Box, and Ping Pong, which are fully described so that you can learn from the programming techniques illustrated as well as have fun playing the games.

The authors go into detail on how you can expand your KIM from the basic small-but-powerful KIM-1 system to a huge-and-super-powerful machine. They include diagnostic and utility programs to help you build extra devices onto your KIM system, such as teletype, display, or more memory. The book also covers the jargon of KIM programming and what's available in both hardware and software for the KIM microprocessor.

Other Books of Interest . . .

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT

TOD LOCFBOURROW

Use the KIM-1 microprocessor to build your own computer-controlled robot. Here are step-by-step directions for the construction of a robot with the complete control programs clearly written out. Photographs, diagrams, and tables direct you through the construction. #5681-8, paper.

BASIC BASIC: An Introduction to Computer Programming in BASIC Language, Second Edition
and
ADVANCED BASIC: Applications and Problems

BOTH by JAMES S. COAN

The complete picture of the BASIC language. One introduces the language through an integration of programming and the teaching of mathematics. The other offers advanced techniques and applications. Both begin with short, complete programs and progress to more sophisticated problems. Basic BASIC, #5106-9, paper, #5107-7, cloth; Advanced BASIC, #5855-1, paper, #5856-X, cloth.

HOME COMPUTER SYSTEMS HANDBOOK

SOL LIBES

An overview of the new world of home computing. Provides the basics of digital logic, number systems, computer hardware, and software to intelligently purchase, assemble, and interconnect components, and to program the microcomputer. #5678-8, paper.

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

ISBN 0-8104-5119-0