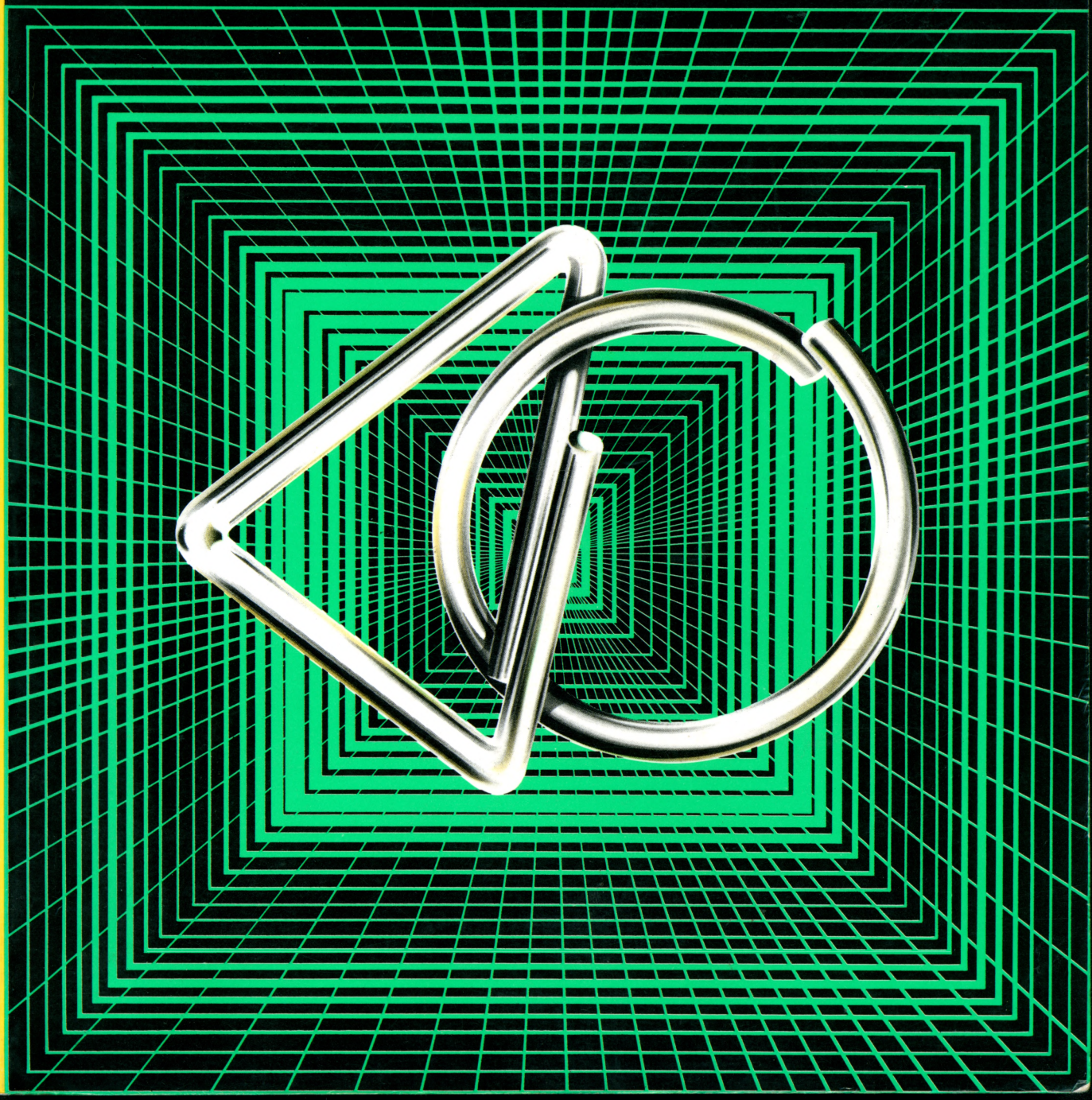
Creative Games and Programs

For the SORD M5



Authors: Susumu Arai
Hisaya Takahashi

Adapted from "Omoshiro Creative"
with the permission of GAKKEN Co., Ltd.

FIRST EDITION

Printed in Japan

CONTENTS

M5 GAMES AND PROGRAMS

You can run a computer!

LET'S PROGRAM THE M5

What is a Program?

A signal that directs the computer to perform a task is called a "command." A set of commands is required to instruct a computer to process data. The order of commands determines the order of computer processing. Creating sets of instructions for the computer is called "Programming."

What is a Programming Language?

A programming language is a written system of symbols, understood by the human being and the computer. Since it was once a problem that computer or 'machine' language and human language are so different, languages or "interpreters," which translate human language into computer language, were created.

The M5 is capable of handling 4 kinds of programming languages; BASIC-I, BASIC-G, BASIC-F, and FALC, and the ROM cartridge of each language has an interpreter. These languages, except for FALC, first need programming according to their respective syntaxes (rules for programming). Then just input the program and the M5 will start processing. If FALC is used, programming is unnecessary. All you do to operate the M5 is to input short commands to meet your needs. Languages like FALC are called "no-programming languages."

1. How to Input a Program

The simplest way to input a program is to type it on the keyboard. However, it is easy to make mistakes in the process of keyboard operation (input error). If a program with an error is input, a computer will respond with "Err 00 in XX" to indicate that the command makes no sense. In this case, the error should be corrected. An error in a program is called a "bug," and eliminating a bug is called "debugging."

2. Input with a cassette tape

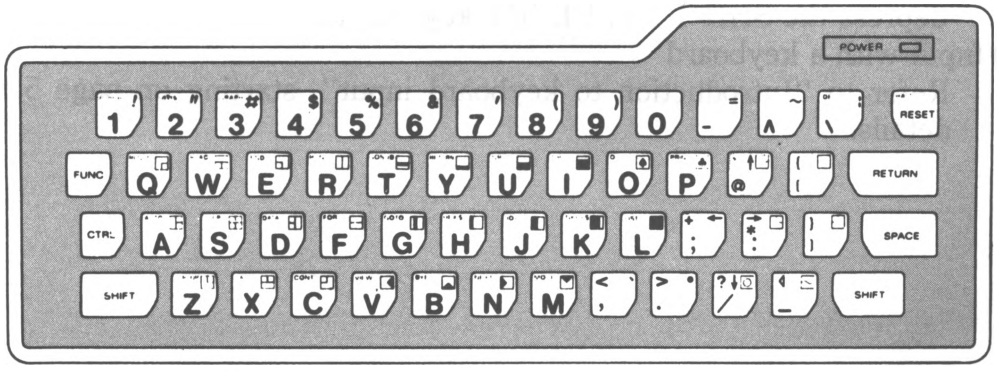
Game programs, which are recorded on cassette tapes, are generally available on the market. These programs can be input to the computer automatically by playing the tape after connecting the tape player with the main storage unit of the computer. (This kind of automatic input is called "loading" or "reading.") When a tape like this is used, input with a keyboard is unnecessary, and, therefore, bugs caused by typing mistakes can be eliminated. However, a computer may be unable to read a program if the tape recorder is not in good condition.

3. Input with a game cartrigde

A game cartridge has the same shape as a ROM cartridge of BASIC or FALC. A program can be processed as soon as the cartridge is set in the slot.

What Kind of Programs to Input?

Programming is not necessary for input with a cassette tape or game cartridge, but it is necessary otherwise.



You can either 1) develop original programs, or 2) type in program listings.

1) Original programming

This is writing your own sequence of commands in BASIC or (BASIC-I, BASIC-G, or BASIC-F). Expect a period of testing and debugging; however, original programming is the best way to learn BASIC languages.

2) Typing in listings

Written programs found in books and magazines are called listings. A program can be input by typing in its listing. Debugging is still necessary, beause there will likely be a few typing errors. Once a program is input and recorded on a cassette tape, the program can be processed just by loading the tape. Storing a program via keyboard on a cassette tape is called "saving."

How to Operate the M5

1) Activating a ROM (game) cartridge
   1) Firmly insert a cartridge with the power off.
   2) Turn the power on to start processing in BASIC or to play a game.
   *Note: Keep the power off whenever a cartridge is inserted or removed.
2) Activating a cassette tape recorder
   1) Connect a cassette tape recorder to the M5 main storage unit.
   2) Insert a BASIC-I cartridge in the slot and turn the power on. The machine signals that it is ready to process in BASIC by displaying "Ready" on the screen.
   3) Play the cassette tape (the tape should be rewound in advance).
   4) When the cursor (a square which blinks on and off under the "Ready" signal) shows the letter A, enter "OLD" then press RETURN, so that the tape (automatically) starts program loading. (When the cursor contains the letter K, depress the FUNC and 1 keys together and change the letter to A.) When "Ready" appears on the screen, which indicates the completion of loading, enter "RUN" then press RETURN to start processing. When you wish to stop processing, depress the SHIFT and RESET keys simultaneously.
3) Input with a keyboard

   Refer to "Introduction to keyboard input", starting on page 5 for details.

INTRODUCTION TO THE KEYBOARD

1 LET'S TRY KEYBOARD INPUT

This book contains 28 program listings starting on page 15. Any of these programs can be input via the keyboard.

Let's Input a Calculation Program

We will try an easy listing as follows, to get used to inputting with a keyboard.

Program List

```
10  REM CALCULATION
20  PRINT  "■■"
30  INPUT A
40  INPUT B
50  C=A+B
60  PRINT  "☆☆ ANSWER ☆☆"; C
70  END
```

If the cursor contains the letter A, you are ready to input. If not, depress the FUNC and 1 keys together. Let's try the first line. The screen displays the following.

```
10  rem CALCULATION
■
```

Go on to the second line. The screen display is as follows.

```
10  rem CALCULATION
20  print  "■■"
■
```

Run the third line.

Next, the fourth line.

Next, the fifth line.

The sixth line.

(If you made errors, you'll read how to the correct them in the following section.)

Finally, the last line. Now, the input of this listing is completed. The screen should display the following.

```
10 rem calculation
20 print "■■"
30 input a
40 input b
50 c=a+b
60 print "☆☆ ANSWER ☆☆"; C
70 end
```

The numbers at the beginning of each line are called line numbers and indicate the order of processing the program.

Now, we check whether all the lines are correctly input in the M5 by using the LIST command. Make sure the cursor contains the letter A and enter LIST, then hit RETURN. A list copied in capital letters from the original list is displayed on the screen. Make sure that both lists have exactly the same content. To make the program run, after the "Ready" sign flashes on the screen, enter RUN and RETURN. The M5 requests the input of a number by displaying a question mark (?) on the screen. For example, enter the number 4 by pushing the 4 key, then the RETURN key. When a question mark again appears on the screen, enter the number 5 by depressing the 5 key, then the RETURN key. Then, the following is displayed on the screen.

```
? 4
? 5
☆☆ ANSWER ☆☆9
```

This indicates that the M5 processed the calculation 4+5=9. Next, we deal with how to correct input errors of a program.

How to Correct a List

1) Shifting the cursor
   When the list was input, it might have been noticed that all the letters to be input are written in the cursor. If a letter must be corrected, the cursor should be shifted to its position. The directional (→ ← ↑ ↓) keys along with the CTRL key are used for shifting the cursor.
   CTRL + → One letter space shift to the right
   CTRL + ← One letter space shift to the left
   CTRL + ↑ One letter space shift upward
   CTRL + ↓ One letter space shift downward
2) When an incorrect letter is input
   Shift the cursor to the position of the letter to be corrected. Enter the correct letter over the incorrect one.
3) When an unnecessary letter is input
   Shift the cursor to the position of the letter to be erased.

   Depress the CTRL + DEL keys.
4) When an additional letter is necessary
   Shift the cursor to the position of the letter to be added. Depress the CTRL and P keys and add the desired letter. Note that the letter inside the cursor changes from A to a when the CTRL + P keys are pressed. As long as the lower case a is inside the cursor, letters can be added consecutively. After letter addition is completed, depress the CTRL + 0 keys, and the letter circumscribed by the cursor returns to A.

Corrections can be made as explained above only before the RETURN key is pushed. If an error is discovered after pushing the RETURN, reinput the whole line and push RETURN again. Then, insure that the error is properly corrected by listing the program. Lastly, here are some additional points to remember.

Points to Remember when Inputting

1) If the left edge of a program is not on the screen, depress the CTRL + T keys to obtain the whole program on the screen.
2) Make sure to press the RETURN key after inputting each line. Otherwise, the input will not be entered.
3) To stop the processing of a program, depress the SHIFT + RESET keys.

2 KEY MODES

Look at key A on the keyboard. Note that four different letters or symbols are on the key. If you push key A when the letter L is inside the cursor on the screen, only the lower case (a) will appear on the screen. How can the other letters or symbols be displayed on the screen?

1) Alphabet Position (FUNC + 1)
2) Shift Lock Position (FUNC + 2)
3) Graphic Position (FUNC + 3)
4) Function Key

Thus, when the FUNC key plus a number is entered, the mode of the screen changes. Even though the same key is depressed, the letter or symbol differs according to the position.

Using the M5 as a Calculator

The M5 can be used as a calculator with the PRINT command.

Example PRINT 5+3

When this is input, the result of the calculation is displayed on the screen as follows.

```
Print 5+3
8

Ready
```

The primary use of the PRINT command is to directly display letters on the screen.

Example PRINT "HELLO"

When this is input, the following is displayed on the screen.

```
Print "HELLO"
HELLO

Ready
```

Painting the Screen

To paint the whole screen, first depress the CTRL + the r keys to obtain the GII mode (Graphic Mode II).

10 PRINT "☆"

Suppose we wish to paint the screen dark blue. In the GII mode, the screen is divided into three sections labeled 4, 5, and 6. Assign the STCHR command to the character code & 80 (■).

20 STCHR "4444444444444444 " TO & 80, 6

30 STCHR "4444444444444444 " TO & 80, 5

40 STCHR "4444444444444444 " TO & 80, 4

Fill the whole screen with the character code & 80.

50 CLS & 80

The entire screen should be dark blue. When you wish to change the color, find the code number of the color desired from chart 1: Color Code. Enter a number to replace the number 4. The color of the screen will change.

Use a Sprite to make a UFO

Now, we'll add another program to the background we just made. First, we specify the size of the sprite (an animation pattern). We will use a square of 16 dots per side. (Refer to "Make Your Own Animated Cartoons" on page 132 for more details.)

60 MAG 1

Assign a UFO pattern to figure code 64.

70 STCHR "182442427e555566" TO 64

Assign the figure code to the sprite number.

80 SCOD 0, 64

Color the sprite, e.g:, deep red (see the color code chart).

90 SCOL 0, 6

Input the speed of the sprite.

100 INPUT

The loop to move the sprite.

110 FOR I = 0 TO 100

Fix the number zero position of the sprite.

120 LOC 0 TO I, 80

The loop to change the speed of the sprite.

130 FOR J = 0 TO S ; NEXT

The NEXT command of the loop to move the sprite.

140 NEXT

This will cause the program loop to run repeatedly.

150 GOTO 10

Input line numbers 10 through 150 and RUN.
Input a number when the speed is asked by the computer.
The smaller the number, the faster the UFO moves.
Depress the RETURN key, and the UFO moves from the left to the right. When it reaches the right edge of the screen, the computer asks for the speed again. the UFO moves by repeating this procedure. Depress the SHIFT + RESET keys to stop processing.

Chart 1: Color Code

Color Code	0	1	2	3	4	5	6	7
Color	trans-parent	black	green	pale green	dark blue	pale blue	deep red	cyan
Color Code	8	9	A	B	C	D	E	F
Color	red	pale red	deep yellow	pale yellow	deep green	magenta	grey	white

3 STORING AND RETRIEVING WITH A CASSETTE TAPE

A program listing input with a keyboard can be used repeatedly as long as the power is on. Once the power has been turned off, however, the listing is cleared. Therefore, in order to store the listing, it must be transferred. We will now discuss how to store ("SAVE") a program on a cassette tape using a cassette tape recorder.

Let's Save a Program

We will save the calculation program on a cassette tape.

First, connect the M5 to a cassette tape recorder with a signal cable for a cassette tape. Plug the white jack in the ERR, the red jack in the recording plug, and the black jack in the remote plug (if there is one).

Give a program or file name to the program (each program needs a name in order to be saved). We will name the program "Addition." A maximum of nine letters is permitted.

Make sure that the "Ready" and the letter A inside the cursor are on the screen. Enter the following.

S A V E [SPACE] [SHIFT + 2] [FUNC + 2] ADDITION [FUNC + 1] [SHIFT + 2]

Set the tape recorder for recording. If the tape recorder has a remote terminal, depress the RETURN key to start recording. If it has no remote terminal, depress the RETURN key after it starts recording.

Wait a few moments until the following appears on the screen.

```
save "ADDITION"
Ready
A
```

The "Ready" sign indicates that the "Addition" program has been saved. If the tape recorder has no remote terminal, stop recording, for it will

not stop automatically. Also remember that a maximum of nine letters can be used for a program name. If a tape counter is attached to the tape recorder, you could record the tape count and the program name together, as is done for music tapes.

Check After Saving

After you have completed the SAVE procedure, confirm that the desired program has, in fact, been stored by entering the VERIFY command before turning the power off. Rewind the saved tape and enter the following.

V E R I F Y [SPACE] [SHIFT + 2] [FUNC + 2] ADDITION [FUNC + 1] [SHIFT + 2]

Depress the RETURN key and play the tape. The following appears on the screen.

```
verify "ADDITION"
ADDITION *
```

An asterisk (*) beside the name of the program indicates that the computer is checking whether the program has been properly saved on the tape. When the computer completes its check, "Ready" appears on the screen. If the tape recorder has no remote terminal, you have to stop it because it will not stop automatically.

If there is an error in saving, an error warning like "ERR 18 in 0" will be displayed on the screen. When this happens, you have to resave from the beginning. Causes of errors include: 1) the recording volume is too low to VERIFY, 2) the cable is not connected, 3) the head of the tape recorder is dirty, etc.

An Example of the Use of the VERIFY command

When you wish to find out which program is recorded on which part of a tape, use the VERIFY command. First, check for unused programs by inputting VERIFY H. Then the programs stored on the tape are displayed on the screen. The part of a program to be saved can be found without clearing the whole program. This program can be retrieved at any time since it has been checked with the VERIFY command. You can turn the power off because the program is already saved on the tape. If you want to be especially careful, save the program on another tape as a backup. This way, even when the first tape is not available, the backup can be used to process the program. Next, let's process a program saved on a cassette tape.

Retrieve a program from tape (OLD)

The command to retrieve a program saved on a tape is "OLD." We will process the "Addition" program saved earlier with the OLD command.

First, rewind the tape to obtain the part containing the "Addition" program. Input "OLD Addition" by pressing [SHIFT 2] RETURN. Start playing the tape. Then the following is displayed on the screen:

```
OLD "ADDITION"
ADDITION*
```

This indicates that the computer found the program called "Addition." The appearance of an asterisk (*) means that the computer is reading the program. The "Ready" sign is displayed on the screen when the "OLD" command has finished processing. Now RUN the program.

OLD without a File Name

Enter OLD, [RETURN], and the computer reads the first listing found. Press the SHIFT + RESET keys to stop processing.

GAME PROGRAMS

HERE WE HAVE 6 GAME LISTINGS WRITTEN IN BASIC-I. ANY OF THESE GAMES OFFERS VIVID IMAGES WITH THE M5 SPRITE FUNCTION.

- Golf
- Boxing
- Ski Jump
- Antartica Invader
- Space Bars
- Puzzle Square

GOLF GAME

This game simulates golf played on a fairway with greens, roughs, banks, and even ponds.

Content:

RUN, and input the number of holes. The screen color changes to green, and the fairway, green, pond, bank, person, and flag will appear on the screen. The width of the fairway and the distance to the green differ each time, as determined by random numbers.

Next, enter the direction and strength of a shot (3 grades are available). The ball is hit. The distance the ball flies differs depending on the course. If the ball goes in the pond, 1 shot will be added as a penalty, and the ball Will come out at a random point as far away from the hole as the pond is. PAR (the average number of shots to the hole) differs depending on the course. The hole number, PAR, total score (SCORE), and score per hole (score) are displayed on the upper part of the screen. Let's try to get a hole-in-one!

Directions:

RUN. The computer first asks "How many holes?" Input any number from 1 to 32767. The course is now drawn. Fix the direction of the shot with the arrow keys (upward, downward, left, and right) on the keyboard. Pressing a key moves the cross in the direction of the arrow. The line between the ball and the center of the cross shows the direction the ball will fly. There is a limit on the horizontal movement of the cross. The limit is 3 spaces to the right from the original position of the ball. If you try to move the ball beyond the limit, the ball will be automatically returned to the original position. It is advisable to press the right arrow key for the first shot.

Now that the direction is fixed, let's select strength.
Level 1: Distance from the ball to the center of the cross
Level 2: Distance twice as long as 1

Level 3: Distance three times as long as 1
Pick one of these three and press the key. The ball flies as directed.

When the ball stops, remember its position and the hole. If the ball comes near the hole, it could be hidden behind the person. Press any key to move the person and cross to the position of the ball. Keep it up until you put the ball in the hole. You cannot go on to the next hole until you finish this one.

After each hole, its score appears in the "scoreboard" on the screen. Press any key to go on to the next hole. After the new hole is set, the total score (the sum of the scores of the holes played so far) appears in the "SCORE" on the screen. The lower the total score is, the better you are playing.

The knack of playing well is to avoid the pond, the bank, and (if possible) the rough. Even when the ball goes in the pond, you could still keep PAR if it happens only once. As for putting, experience is important. Practice repeatedly. When the ball comes near the hole, all you do is just to depress the 3 key until it goes in the hole.

When you finish all the holes, the total score is displayed in the "scoreboard" and the game ends. If you wish to resume the game, type RUN again.

Input:

This program consists of two parts: program I, a sprite-setting program, and program II, the main program. Input as follows:

1. Key in program I, RUN, and save it on tape.
2. Key in program II, and save it on the same tape.
   (The mode is GII should not be changed. If it is changed to another mode, like the text mode, the set sprite will be cleared.)

Program:

This program uses 15 sprites—3 for the person to swing the club, 2 each for the pond and bank, 4 for the green, and 1 each for the ball, flag, hole, and cross. The program incorporates a FOR loop that handles a subroutine to hit the ball, a routine to score a completed hole, and a routine to draw a border between the fairway and rough. The distance from each bank or pond to the green is fixed.

The green may sometimes be invisible momentarily, because 5 or more sprites cannot be displayed on the same line.

LIST OF VARIABLES:

A1:	PAR (number of shots)
A2:	Total score
A3, A4, E4, E5, F1	
Q, W, LI, D:	Course setting assistance

B1, E1, E2
E3, F2, F3,
H3, O: Ball hitting assistance
B2: Strength
B3: Sprite number of the person
G1, G2: Ball position
G3, G4: Cross position
B4, B5, P, J: FOR-NEXT
C, R: Flag position
CI, RI: Green position
V$: Cross shifting input

PROGRAM MAP:

30~560: Main routine
40~290: Course setting
350~500: Ball hitting
590~650: Routine for drawing a border between the rough and fairway
660~860: Routine for making the person swing after a direction and strength are input

PROGRAM LIST

PROGRAM I

```
10rem GOLF SPRITE
20 Print "■■■"
30 view 1,0,30,23
40 mag 2
50 for I=0 to 67
60 read A$
70 stchr A$ to 40+I
80 scod 15+I/4,40+I/4*4
90 next I
100 scol 16,8:scol 17,4
110 scol 18,4:scol 19,4
120 scol 20,15:scol 21,1
130 scol 22,2:scol 23,2
140 scol 24,2:scol 25,2
150 scol 26,7:scol 27,7
160 scol 28,9:scol 29,9
170 scol 30,9:scol 31,9
180 scol 15,13
190 for I=31 to 15 step-1
200 loc I to(31-I)*7,(31-I)*7
210 next
220 stchr "ffff7faf05000000" to 255,1
230 stchr "ffff7faf05000000" to 255,2
240 stchr "ffff7faf05000000" to 254,2
250 stchr "ffff7faf05000000" to 254,3
260 stchr "c3c3c3c3c3c3c3c3" to 255,4
270 stchr "c3c3c3c3c3c3c3c3" to 255,5
280 stchr "3c3c3c3c3c3c3c3c" to 254,5
290 stchr "3c3c3c3c3c3c3c3c" to 254,6
300 for I=4 to 6
```

```
310 stchr "3333333333333333" to 253,I
320 stchr "cccccccccccccccc" to 252,I
330 next I
340 data "0001010101010007c"
350 data "7c00010101010101"
360 data "0080808080800003e"
370 data "3e00808080808000"
380 data "0203030303030302"
390 data "0202020202020202"
400 data "000080c0e0c08000"
410 data "0000000000000000"
420 data "0103030107070f07"
430 data "0707070f0d091931"
440 data "80c0c080c0e0e0c0"
450 data"c0c0c0e060403018"
460 data"050b13313f1f0b03"
470 data"0707070e0c081830"
480 data"80c0c080c0c0c080"
490 data"8080c0e060203018"
500 data"0103030103030303"
510 data"0707070e0c081830"
520 data"a0d0d888f0e0c0c0"
530 data"c0c0c0e060203018"
540 data"40e0400000000000"
550 data"0000000000000000"
560 data"0000000000000000"
570 data"0000000000000000"
580 data"60f0f06000000000"
590 data"0000000000000000"
600 data"0000000000000000"
610 data"0000000000000000"
620 data"0000000000030303"
630 data"0f0f1f3f3f3f1f1f"
640 data"0000033fffffffff"
650 data"ffffffffffffffff"
660 data"1f0f0f07070f1f3f"
670 data"3f3f1f1f03010000"
680 data"ffffffffffffffff"
690 data"fffffffffffe3000"
700 data"387effffffffffff"
710 data"ffffffffffffffff"
720 data"0000e0f8fcfefefc"
730 data"fcfcf8f0f0f0f0f8"
740 data"ffffffffffffffff"
750 data"fffee0c080000000"
760 data"fcfefefeffffef8"
770 data"c000000000000000"
780 data"00031f1f3f3f3f3f"
790 data"3f3f1f1f3f3f7f1f"
800 data"00e0fcfefeffffff"
810 data"ffffffffefefcf8"
820 data"0f1f3f3fffffffff"
830 data"ffffffefececfcfc"
840 data"c0f0f8fcfcfcfefe"
850 data"fffffffcc0000000"
860 data"00393f7f3f7fffff"
870 data"7f7f7f3f3f3f1f08"
880 data"30f8fefefeffffff"
890 data"fefeffffffefc3000"
900 data"3071fbff7f7fffff"
910 data"ffffff7f3f1f0e08"
920 data"c0e0f8fcfffffefe"
930 data"fefcfcfcf8f07838"
940 data"0c0c0e3f7f8f7fff"
950 data"ffffff7f7f7f3f07"
```

```
960 data"0e1f3ffffffefcfc"
970 data"fefffffe0c08000"
980 data"01037f3f3f7f3f1f"
990 data"3f7f7fff7fff7f3f"
1000 data"b0f0f8fcfefffefc"
1010 data"f8f0e0c080c080c0"
1020 data"0101010101010101"
1030 data"0101010101010101"
1040 data"0000000000000000"
1050 data"0000000000000000"
1060 for I=0 to 31:loc I to 0,300:next:end
1070 for I=0 to 35
1080 stchr "0000000000000000" to 100+I
1090 scod I/4,100+I/4*4
1100 next
```

PROGRAM II

```
10 Print "[redacted]":input "How many Holes";F5
20 for P=1 to F5:cls 252
30 let Q=5:let W=255:let E4=6:let E5=1:let F1=10:gosub 580
40 let Q=18:let W=254:let E4=12:let E5=1:let F1=19:gosub 580
50 for I=1 to 30:let L1=0
60 for J=&3800+I to &3AE0+I step &20
70 let D=vPeek(J)
80 if D=254 then let L1=0:let J=&3AE0+I:goto 110
90 if L1=1 then vPoke J,253
100 if D=255 and L1=0 then let L1=1:goto 110
110 next:next:randomize
120 let C=rnd(10)*7+130
130 let A3=(C+8)/8
140 let C1=rnd(5)*3+C-16
150 let R=rnd(7)*8+64
160 let A4=(R+16)/8
170 if vPeek(&3821+A4*32+A3)=252 then goto 120
180 let R1=rnd(5)*3+R-6
190 loc 16 to C,R
200 loc 22 to C1,R1
210 loc 23 to C1,R1+15
220 loc 24 to C1+15,R1
230 loc 25 to C1+15,R1+15
240 loc 21 to C+6,R+15
250 loc 26 to C1-40,R1-15
260 loc 27 to C1-25,R1-15
270 loc 28 to C1-35,R1+30
280 loc 29 to C1-20,R1+30
290 let G3=0:let G4=74:let G1=6:let G2=92:let U=0:let H3=1
300 let A1=(C+81)/50
310 Print cursor(1,1);"hole ";P;cursor(1,2)"Par ";A1
320 Print cursor(14,2);"score ";A2
330 loc 20 to G1,G2
340 gosub 650
350 let U=U+1:Print cursor(14,2);"score ";U-A1
360 let F2=F2+8:let F3=F3+8
370 let O=F2-G1
380 let B1=1:if O<0 then let B1=-1
390 let E2=sgn(F3-G2)
400 for A5=G1 to G1+O*B2/H3 step B1
410 let E1=F3-G2
420 if O=0 then let E3=E1 else let E3=E1/O*sgn(O)
430 let G2=G2+E2*rnd(1)+E3+rnd(B2)-rnd(B2)
440 loc 20 to A5,G2:next
450 if abs(A5-C1+20)<15 and abs(G2-R1-38)<8 then let H3=10:P
rint "[redacted]":goto 480 else let H3=1
```

```
460 if abs(A5-C1+25)<15 and abs(G2-R1+8)<8 then let U=U+1:Pr
int "■":let G2=G2+15+rnd(20):goto 480
470 if vPeek(&3821+(G2/8-1)*32+A5/8)=252 then Print "■":let
H3=2 else let H3=1
480 let G1=A5:let G3=G1-8:let G4=G2-16
490 if abs(C+7-G1)<4 and abs(R+17-G2)<4 then goto 530
500 if inkey$="" then goto 500
510 loc B3 to 0,-90
520 goto 330
530 loc 20 to 0,-90:Print "■":loc B3 to 0,=-90
540 if inkey$="" then goto 540
550 let A2=U-A1+A2:next P
560 Print cursor(1,2);"game over ";cursor(20,2);A2
570 end
580 let S=0
590 let Q=1-rnd(2)+Q
600 Print cursor(S,Q);chr$(W):let S=S+1
610 if S>29 then return
620 if Q<E4 then let Q=Q+E5:goto 600
630 if Q>F1 then let Q=Q-E5:goto 600
640 goto 590
650 loc 17 to G3,G4
660 let F2=G1+16:let F3=G2-8
670 if abs(F2-G1)>25 then let F2=G1
680 let V$=inkey$:loc 15 to F2,F3
690 if V$="@" then let F3=F3-3
700 if V$="/" then let F3=F3+3
710 if V$=";" then let F2=F2-3
720 if V$=":" then let F2=F2+3
730 if ascii(V$)<49 or ascii(V$)>51 then goto 670
740 loc 15 to 0,-90
750 let B2=val(V$):let B3=17
760 if G3<C then restore 830 else restore 840
770 for B4=0 to 2
780 for B5=1 to 300:next
790 loc B3 to 0,300
800 read B3
810 loc B3 to G3,G4
820 next B4
830 data 18,17,19
840 data 19,17,18
850 return
```

BOXING GAME

Content:

The goal is to win the match by punching the opponent and reducing his stamina to zero within 5 rounds. A punch goes either to the face by bringing up an arm, or to the body by bringing it down. Every time you punch, you lose 1 unit of stamina. When your punch hits the opponent, he loses 5 units of stamina. However, when you punch him while he is guarding, his stamina does not decrease, and yours does. 100 stamina points are added to each boxer in every interval. This is a 5-round match. If both of them still have stamina at the end of the 5th round, the winner is decided by comparing their remaining stamina points.

Directions:

Set the program and RUN. Input the name of the boxer on the right (who is moved by the right joypad), within 7 letters. Input the name of the boxer on the left in the same manner. The game starts, and the names and stamina points of each player appear on the upper part of the screen.

Arms can be brought up and down with a joypad, and punches are released by depressing the attack switches. When a boxer's arm is up, it guards his face, and it can also punch his opponent's face. When the arm is down, it guards the body, and it can also punch the opponent's body. Timely guarding is effective in reducing the opponent's stamina.

When the game is over, the computer displays the winner's name and asks if you wish to have another game. If you do, depress the Y key; if not, the N key.

Input:

This program consists of program I, a sprite definition program, and program II, the main program. Input as follows:

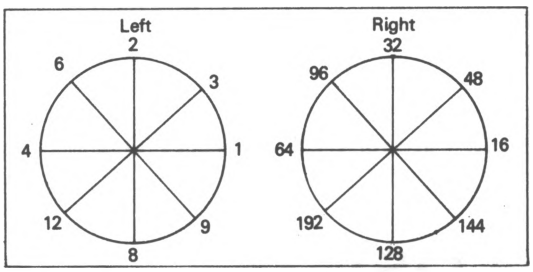
1. Input program I and save it on a tape.
2. Input program II, and save it on the tape.
3. VERIFY the tape, OLD program I, and wait for a minute.

4. Find "OLD main program" on the screen and OLD program II with the GII mode. (Remember that changing the mode will clear the sprite data!)
5. RUN, and the game starts.

Program:

BASIC-I uses the INP command to activate the joypads. Diagram 1 indicates the figures when each joypad is pressed, and the number pressed is picked. When both joypads are simultaneously depressed, their total number will be picked. This way, the computer is capable of simultaneously controlling the positions of both boxers' arms.

Diagram 1: Figures when a joypad is depressed



The attack switches, which are picked by INKEY$, for the right player are keys 1 and 2, and for the left player keys 5 and 6. Avoid uninterrupted are depression of the attack switches.

Unfortunately, gloves and trunks are not colored, for 5 or more sprites cannot be on the same line.

PROGRAM MAP: PROGRAM I

10: REM sentence
20: Screen initialization
30~60: Sprite definition
70~100: Sprite color definition
140~270: Set character data sentence

PROGRAM MAP: PROGRAM II

10~20: REM sentence
30~50: Initialization
60~70: Name input
110~240: Main routine
260: Arm up, the left boxer
270: Arm down, the left boxer
280: Arm up, the right boxer
290: Arm down, the right boxer
300~330: Left punch
340~370: Right punch

380～390: Processing when a punch is made
400～450: Processing to finish the game
460～520: Sound-making
530～550: Processing the rounds
560～570: Positioning the boxers

LIST OF VARIABLES: PROGRAM II

XR
YR: Co-ordinates of the right boxer
XL
YL: Co-ordinates of the left boxer
SR: Stamina points of the right boxer
SL: Stamina points of the left boxer
PR: Right arm position
PL: Left arm position
RN $: Right name
LN $: Left name
R: Number of the round
U: Arm changing
P $: Punches
WN $: Winner's name
Other: Starting with I, J are for the FOR-NEXT loop

PROGRAM LIST

PROGRAM I

```
10rem BOXING SPRITE DATA
20 cls:Print "■":view 1,0,30,23
30 for I=168 to 223
40 read S$
50 stchr S$ to I
60 next I
70 for I=5 to 18
80 if I=5 or I=8 or I=9 or I=10 then let CL=7 else if I=6 or
 I=7 or I=11 or I=12 then let CL=5 else let CL=9
90 scod I,168+(I-5)*4:scol I,CL
100 next I
110 mag 3
120 Print cursor(1,10);"OLD THE MAIN PROGRAM"
130 end
140 data"00003078fcfc7838","1c0e0e0707030100","0000000000000
000","000003Ofbffcf0c0"
150 data"0000000000000000","0000c0f0fd3f0f03","00000c1e3f3f1
e1c","387070e0e0c08000"
160 data"000000c0e0f0f070","7070707f7f7f0000","0000000000000
000","000c1effffff1e0c"
170 data "0000000000000000","003078ffffff7830","00000003070f
0f0e","0e0e0efefefe0000"
180 data "0000000000000000","0c1e3f3f1e0c0000","00001c1c1c1c
1c1c","1c1cfcfc00000000"
190 data "0103070703010101","0101010101010101","c0e0f0f0e0c0
c0c0","c0c0c0c0c0ffffff"
200 data "0000383838383838","38383f3f00000000","000000000000
0000","3078fcfc78300000"
210 data "03070f0f07030303","030303030fffffff","80c0e0e0c080
```

```
8080","8080808080808080"
220 data "030f1f2125207010","100c03010307070f0f","c0f0f8fcfcfc
fc78","3830c0c0e0f0f8f8"
230 data "0f0f0f0f0f0f0f0f","0f0f0f0f0f0f0f07","f8f8f8f8f8f8
f8f8","f8f8f8f8f8f8f8f0"
240 data "07070e0e1c397939","3838383838387878","707070e0c0c0
c0e0","f0783c1e0f07060c"
250 data "030f1f3f3f3f3f1f","1c0c0303070f1f1f","c0f0f884a404
0e08","0830c080c0e0f0f0"
260 data "1f1f1f1f1f1f1f1f","1f1f1f1f1f1f1f0f","f0f0f0f0f0f0
f0f0","f0f0f0f0f0f0f0e0"
270 data "0e0e0e0703030307","0f1e3c78f0e06030","e0e07070389c
9e9c","1c1c1c1c1c1c1e1f"
```

PROGRAM II

```
10rem BOXING MAIN PROGRAM
20 cls:print "■":view 1,0,30,23
30 let XR=115:let YR=40:let PR=1:let SR=500
40 let XL=72:let YL=40:let PL=1:let SL=500
50 print cursor(7,10);:input "NAME LIGHT";RN$
60 print cursor(7,12);:input "NAME LEFT";LN$
70 print "■"
80 mag 3
90 gosub $SP
100 for R=1 to 5
110 for IT=0 to 150
120 if SL<0 or SR<0 then gosub $SD:goto $ED
130 print cursor(2,0);LN$;SL
140 print cursor(14,0);RN$;SR
150 let U=inp(&37)
160 if U=2 then gosub $A else if U=8 then gosub $B else if U
=32 then gosub $C else if U=128 then gosub $D
170 if U=34 then gosub $A:gosub $C else if U=130 then gosub
$A:gosub $D else if U=40 then gosub $B:gosub $D
180 if U=136 then gosub $B:gosub $D
190 let P$=inkey$
200 if P$="1" or P$="2" then gosub $E else if P$="5" or P$="
6" then gosub $F
210 next IT
220 if R<5 then gosub $RD
230 next R
240 goto $ED
250$A:loc 11 to XL,250:loc 12 to 87,47:let PL=0:let IT=IT+1:
return
260$B:loc 12 to XL,250:loc 11 to 80,65:let PL=1:let IT=IT+1:
return
270$C loc 9 to XR,250:loc 10 to 100,47:let PR=0:let IT=IT+1:
return
280$D:loc 10 to XR,250:loc 9 to 107,65:let PR=1:let IT=IT+1:
return
290$E:if PL=0 then loc 12 to XL,250:loc 6 to 90,48 else loc
11 to XL,250:loc 7 to 92,62
300 for I=0 to 500:next I:loc 6 to XL,250:loc 7 to XL,250:if
 PL=0 then gosub $A else gosub $B
310 if(PL=0 and PR=1)or(PL=1 and PR=0)then gosub $LP else le
t SL=SL-1
320 return
330$F:if PR=0 then loc 10 to XR,250:loc 5 to 97,48 else loc
9 to XR,250:loc 8 to 95,62
340 for I=0 to 500:next I:loc 5 to XR,250:loc 8 to XR,250:if
 PR=0 then gosub $C else gosub $D
350 if(PL=0 and PR=1)or(PL=1 and PR=0)then gosub $RP else le
t SR=SR-1
```

```
360 return
370$LP:let SL=SL-1:let SR=SR-5:return
380$RP:let SR=SR-1:let SL=SL-5:return
390$ED:gosub $SD:for I=5 to 18:loc I to 250,250:next:Print "
■"
400 if SR=SL then goto 420 else if SR<0 then let WN$=LN$ els
e let WN$=RN$
410 Print cursor(10,10);"WINNER ";WN$:goto 430
420 Print cursor(10,10);"DRAW"
430 Print cursor(7,15);"PLAY AGAIN ? (y/n)"
440 let Z$=inkey$:if Z$<>"y" and Z$<>"n" then goto 440 else
if Z$="y" then goto 20 else end
450$SD:for I1=0 to 5
460 out &20,&C9:out &20,&50
470 out &20,&D0
480 for I2=1 to 500:next
490 out &20,&DF
500 next
510 return
520$RD:gosub $SD:for I=5 to 18:loc I to 250,250:next
530 Print cursor(10,10);"ROUND ";R+1:for J=0 to 5000:next J:
cls:gosub $SP:let SL=SL+100:let SR=SR+100
540 return
550$SP:loc 13 to XR,YR:loc 14 to XR,YR+32:loc 15 to XR,YR+64
:gosub $A
560 loc 16 to XL,YL:loc 17 to XL,YL+32:loc 18 to XL,YL+64:go
sub $C:return
```

SKI JUMP GAME

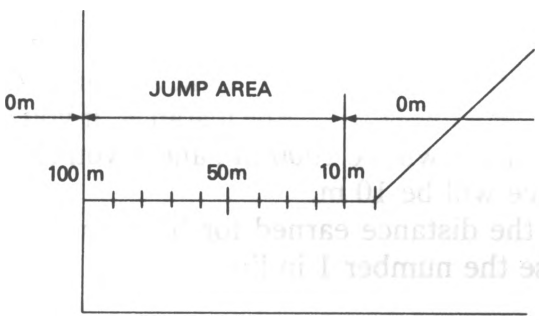
An olympic ski jump event played in your own living room

Content:

A jumper starts gliding down a sloping ramp. When he gets to the level of the ramp, make him jump. After he jumps, winds start to disturb him. Control his balance with keys, and keep him up as long as possible. Remember that if you don't make him jump he will crash to the ground.

Chart 1 Timing of Jump and Flying Distance

Jumping Area



Directions:

When the program runs, the skiers are introduced into the lower part of the screen with music. The music stops and the skiers emerge in the upper part of the screen. Now they are ready to jump.

After the go-signals (a short series of beeps and a green light), a skier starts slowly gliding down the ramp to the level part of the ramp. When you think he should jump, press J. If you press the J key too late, he will fall to the ground with a record of zero m. and the game will be over.

The maximum distance obtained by this initial jump is 100 m. The initial distance is determined according to the point where the J key is

depressed. The longer you wait, the longer the jump will be. But if you wait too long the skier will fall.

Control his balance against winds until his altitude becomes zero (a red line on the right side of the screen indicates altitude). Winds strike the skier from the right and from the left. Press the → key to counter wind in the ← direction, and the ← key against wind in the → direction. The better balanced a jumper is, the longer his flying distane. Gain as much distance as possible by pressing the ↑ key (continuously, if you wish) while there are no winds, and increase your score. A maximum of 50 points can be added by this balance control. The total distance, the sum of the distances earned by the initial jump and by balance control, will be in the upper left of the screen. The maximum total distance is 150 m. When the altitude becomes zero, the computer displays the highest score and asks if you desire to "Replay?". Depress the Y key to replay and the N key to end the game.

Program:

This program consists of two parts, one directing jumping and one for balancing. The program can be varied as follows.

1) To increase the speed of gliding down the runway, change the number 50 in line number 160, "FOR J=1 TO 50" to a smaller number. To decrease the speed, make the number bigger than 50.
2) To shorten the time a jumper is in the jumping area (which makes jumping more challenging), change the number 40 in line number 180, "FOR J=1 TO 40", to a smaller number. To lengthen the time, which makes jumping easier, make the number larger than 40.
3) To change the maximum flying distance, change the number 10 in line number 170, "P=(90-I)*10". For instance, if you change it to 100, the maximum distance will be 1000 m., and if you change it to 1, the maximum distance will be 10 m.
4) To increase the distance earned for balancing (1 m. per key depression) increase the number 1 in line number 130, "LEP P=P+1".

LIST OF VARIABLES

S1, S2, S3: variable to make sound
M: position when jumpers ski horizontally from left to right
P: score
Q: random numbers
MU: balance
Y: snow movement
HI: high score

PROGRAM MAP

10: initial positioning
20~200: jumping

210～350: balancing
360～370: music data

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23
20 for I=4 to 6:stchr "7777777777777777" to 151,I:next I:cls
 151
30 let A$="a7a7a7a7a7a7a7a7":for I=4 to 6:stchr A$ to 128,I:
stchr A$ to 236,I:next I
40 for I=1 to 13:let A$="▄▄▄▄":for J=1 to I:let A$=A$+"■":ne
xt J:Print cursor(25-I,5+I);A$:next I
50 for I=1 to 4:Print cursor(10,18+I);rPt$(19,"■"):next I
60 stchr "060608103b0418ff" to 140:stchr "6060100083b2018ff"
to 141:stchr "92d47830204080000" to 142
70 mag 0:for I=1 to 3:scod I,139+I:scol I,1:next I:scod 5,22
5:restore 360:let M=0
80 read S1,S2,S3:if S1=0 then goto 110
90 out &20,&80+S2:out &20,S1:out &20,&93
100 for J=1 to S3:let M=M+1:loc 1 to M,180:next J:out &20,&9
F:goto 80
110 for I=255 to 200 steP-1:loc 2 to I,40:next I:for I=1 to
1500:next I
120 scod 4,225:loc 4 to 210,70:for I=1 to 3:scol 4,8:Print "
■";:for J=1 to 1000:next J:scol 4,1:for J=1 to 500:next J,I
130 scol 4,2:loc 4 to 210,80:out &20,&80:out &20,&04:out &20
,&90:for I=1 to 1000:next I:out &20,&9F
140 loc 2 to 300,-50:loc 4 to 300,-50:out &20,&93:scol 4,14:
scol 5,14:for I=1 to 105:loc 3 to 200-I,40+I
150 out &20,&80:out &20,(105-I)/2:for J=1 to 50:next J,I:out
 &20,&9F:loc 3 to 100,250
160 let P=-1:for I=90 to 80 steP-1:loc 2 to I,143:if P=-1 an
d inP(&33)=64 then let P=(90-I)*10
170 for J=1 to 40:next J,I:if P=-1 then let P=0
180 Print cursor(3,2);P;"m":out &20,&93
190 if P=0 then for I=0 to 40:loc 2 to 80,143+I:out &20,&80:
out &20,I+20:next I:out &20,&9F:goto 330
200 for I=1 to 100:loc 2 to 80-I,143-I:out &20,&80:out &20,(
100-I)/2:next I
210 let A$="17171717171717":for I=4 to 6:stchr A$ to 45,I:
stchr A$ to 60,I:stchr A$ to 62,I
220 stchr "8080808080808080" to 128,I:for J=236 to 239:stchr
 "17171717171717" to J,I:next J,I
230 cls 151:for I=1 to 3:Print cursor(13-I,19+I):"◢ ◤":Print
 cursor(16+I,19+I);"◥ ◣":next I
240 for I=2 to 21:Print cursor(27,I);"■":next I:randomize:fo
r M=1 to 10
250 let Q=rnd(2):if Q=0 then let MU=4:Print cursor(11,12):rP
t$(10,"■")
260 if Q=1 then let MU=32:Print cursor(11,12);"■■■■":cursor(
17,12);"--->"
270 if Q=2 then let MU=64:Print cursor(17,12);"■■■■":cursor(
11,12);"<---"
280 for MM=1 to 5:for I=0 to 15:let Y=Y+2:if Y=30 then let Y
=0 else loc 4 to 80-Y,160+Y:loc 5 to 167+Y,160+Y
290 out &20,&80+I:out &20,(M-1)*5+MM:next I
300 if inP(&36)=MU then let P=P+1
310 Print cursor(3,2);P;"m":next MM:for I=2 to 1+M*2:Print c
ursor(27,I);" ":next I,M
320 out &20,&9F:if P>HI then let HI=P
```

```
330 Print cursor(3,16);"HIGH SCORE";HI;"m";cursor(3,18);"REP
LAY ? (y/n)"
340 let A$=inkey$:if A$<>"y" and A$<>"n" then goto 340
350 if A$="y" then loc 2 to 100,200:loc 4 to 100,200:loc 5 t
o 200,200:goto 20 else end
360 data &8,&f,50,&7,&f,25,&8,&f,50,&7,&f,25,&8,&f,100,&a,&a
,100
370 data &a,&a,50,&a,&0,25,&a,&a,50,&a,&0,25,&a,&a,100,&d,&6
,100,0,0,0
```

ANTARCTICA INVASION

Help Antarctica wildlife fight back.
Be a true animal lover.

Content:

A white penguin attacks an invader with blue ice cubes (he moves the ice cubes by blocking them with his body). The invader does not, of course, just wait to be attacked. He is always waiting for a chance to attack the penguin. The game is over when either one defeats the other.

Directions:

RUN the program, and 25 blue ice cubes in a frame of 12 cubes in width and 11 cubes in length appear. An invader sneaks around the ice cubes. Command keys are as follows.

↑ Upward	→ To the right
↓ Downward	← To the left

Pressing any of these directional keys produces a white penguin on the screen which moves in the direction indicated by the depressed key. Move the penguin into an ice cube so that the ice cube is projected toward the invader in the direction of the penguin's movement. A propelled ice cube continues movement until it contacts another ice cube or any side of the frame. The game is over either when the invader is killed with the ice cube attack, or when the penguin is attacked by the invader. The score in the upper right of the screen is calculated by subtracting the total seconds spent on the fight from 100. After the game ends, depress the CTRL+Z keys, then the CTRL+T keys. RUN again to play another game.

Program:

Since it is possible to simultaneously assign 32 sprites, theoretically, sprites can be used for all 25 ice cubes. In practice, however, a maximum of only four sprites can be on the same line. Another consideration is that if a penguin or an invader passes near four sprites in a row, it will be invisible. To avoid this inconvenience, all the ice cubes are set characters written

with the PRINT command, and a sprite is used only for their movement.

LIST OF VARIABLES

XX, YY:	penguin's position
D:	code to fix the penguin's direction
P5, P6:	data of an input key
BX, BY:	invader's position
XO, YO:	range of each sprite's shift
SI, S:	check if the penguin or an ice cube is along the edge of the frame
X8, Y8:	position of sprite
XA, YA:	penguin & ice when moving
R, XC, YC:	invader's direction
TT:	score

PROGRAM MAP

10~50:	screen production
60~110:	key input
120~150:	penguin's movement
160~220:	ice cube's movement
230:	start and stop music
240~270:	subroutine to set up the screen
280~310:	subroutine to shift the sprites of the penguin and ice cubes
320~360:	subroutine to shift the sprite of the invader

PROGRAM LIST

```
20 print "■■■■":for I=0 to 132:vpoke I,0:next:let T=time
30 stchr "18343e3c7c3c382c" to 160:stchr "182c7c3c3e3c1c34"
to 161:scol 0,15:stchr "ffabd5abd5abd5ff" to 162:scol 1,5
40 stchr "24183c5aff182442" to 163:scod 2,163:scol 2,6:view
1,0,30,23:out 32,&B0
50 mag 1:let XX=96:let YY=80:let D=160:scod 0,D:scod 1,162:g
osub 240
60 let P5=inp(&35):let P6=inp(&36):if P5=2 or abs(XX-BX)<12
and abs(YY-BY)<12 then goto 230
70 gosub 340:if P5<>32 and P6<>4 and P6<>32 and P6<>64 then
goto 60
80 if P6=64 then let X0=2:let Y0=0:if E=161 then loc 0 to 30
0,300:let D=160
90 if P5=32 then let Y0=2:let X0=0
100 if P6=32 then let X0=-2:let Y0=0:if D=160 then loc D to
300,300:let D=160
110 if P6=4 then let Y0=-2:let X0=0
120 let S=0:let XA=XX:let YA=YY:gosub 280:let XX=XA:let YY=Y
A:let S1=S
130 scod 0,D:loc 0 to XX,YY:if abs(XX-BX)<12 and abs(YY-BY)<
12 then goto 230 else let X8=XX+8+8*X0/2
140 let Y8=YY+8+8*Y0/2:if XX=176 and P6=64 or YY>158 and P5=
32 then goto 60
150 if vpeek(Y8/16*12+X8/16)=0 then goto 60 else vpoke Y8/16
*12+X8/16,0
```

```
160 loc 1 to X8/16*16,Y8/16*16:Print cursor(X8/16*2,Y8/16*2)
;" ↓←← "
170 let S=0:let XA=X8:let YA=Y8:let X0=X0*4:let Y0=Y0*4:gosu
b 280:let X8=XA:let Y8=YA
180 let X0=X0/4:let Y0=Y0/4
190 if S=1 and S1=0 or P6=32 and XA=0 or P5=32 and YA=160 or
 P6=64 and XA=176 or P6=4 and YA=0 then goto 220
200 if vPeek((Y8+4*Y0)/16*12+(X8+4*X0)/16)<>0 then goto 220
210 if S=1 then loc 1 to X8,Y8:goto 170 else loc 1 to X8-8,Y
8-8:goto 170
220 loc 1 to X8,Y8:vPoke Y0/16*12+X8/16,1:Print cursor(X8/16
*2,Y8/16*2);" ▼↓←←▲▶":loc 1 to 300,300:goto 60
230 for I=1 to 9:for M=40 to 90:out 32,&A0+I:out 32,M:next:n
ext:out 32,&BF:Print "game over■" end
240 stchr "ffffccccf3f3cccc" to 152,1:stchr "ffffcfcf3333cfc
f" to 153,1:stchr "f3f3ccccf3f3ffff" to 154,1
250 stchr "3333cfcf3333ffff" to 155,1:vPoke &3B96,&50:for I=
1 to 25
260 let X=rnd(11):let Y=rnd(10):if vPeek(X+12*Y)then goto 26
0
270 vPoke X+12*Y,1:Print cursor(X*2,Y*2);" ▼←←↓▲▶":next:retu
rn
280 gosub 340:if abs(XA-BX)>12 or abs(YA-BY)>12 then goto 30
0 else if abs(X0+Y0)<>2 then Print cursor(24,0);TT+C+C/2
290 goto 230
300 let C=C+1:let XA=XA+X0:let YA=YA+Y0:if P6=4 or P5=32 the
n goto 320
310 if XA<0 then let S=1:let XA=0:goto 330 else if XA>175 th
en let S=1:let XA=176:goto 330
320 if YA<0 then let S=A:let YA=0 else if YA>159 then let S=
1:let YA=160
330 return
340 let R=rnd(3):if R=0 then let XC=0:let XC=1 else if R=1 t
hen let YC=1:let XC=0
350 if R=2 then let YC=0:let XC=-1 else if R=3 then let YC=-
1:let XC=0
360 let BX=BX+8*XC:let BY=BY+8*YC:if BX<1 then let BX=0 else
 if BX>175 then let BX=176
370 if BY<1 then let BY=0 else if BY>159 then let BY=160
380 loc 2 to BX,BY:let TT=time-T:Print cursor(24,0);TT+C:out
 32,&A0+rnd(9):out 32,rnd(8):return
```

SPACE BASES GAME

Content:

The macrocosm (the whole screen) is divided into microcosms of 7 units × 7 units. Each microcosm is divided into equal 9 × 9 coordinates, on which the spaceship, stars, invaders, and bases are distributed.

Although the initial supply of energy and missiles is fixed, the spaceship can gain energy and ten more missiles at each base. Time, however, decreases each time a command is processed. You win the game when all the invaders are defeated. You lose the game when either energy or time reaches zero and at least one invader is still alive.

Directions:

RUN and short-range sensor shows the spaceship in a microcosm in the upper left of the screen. Amounts of energy (ENERGY) and time (TIME), the number of missiles (MISSILES) and invaders (INVADERS), the position of the spaceship (POSITION) in the microcosm are all displayed in the upper right of the screen. Each character in the short-range sensor indicates the following.

E: Earth Thunder (your ship) *:Star
I: Invader B: Base

The keys for various commands, e.g., shifting the spaceship or shooting a missile, are as follows.

key 1 Drive: To move the spaceship within a microcosm and to stop by a base.

key 2 Warp: To shift the spaceship from one microcosm to another. If an invader is in the microcosm into which the spaceship is shifted, it will attack the ship according to a fixed probability. Energy of the ship decreases with the attack.

key 3 Long-range sensor: To check the microcosm containing the spaceship and the surrounding 3 × 3 microcosms. When the 3 key is depressed 3-digit numbers appear in the microcosm containing the spaceship and in each of the

eight surrounding microcosms. The first digit of each 3-digit number indicates the number of bases, the second, the number of invaders, and the the third, the number of stars in the respective microcosm. Depress the SPACE key to return the original image.

key 4 Missiles: To attack invaders in the same microcosm. These commands are used to fight with invaders. Commands 1, 2, and 4 are directed to specific locations within a microcosm by the coordinates. Reprepare the program by reading program I to replay the game.

Input:

This game incorporates two programs: program I, the data-making program, and program II, the main program.

1. INPUT program I and SAVE it on a tape.
2. INPUT program II and SAVE it on the same tape program I is on.
3. RESET and READ program I.
4. RUN program I and wait for a while.
5. Find "Ready" on the screen and READ program II with OLD.
6. RUN, then the game is ready to start.

Program:

A data-making program and a main program are used. The data-making program sets the initial conditions of the invaders, stars, and bases, or processes initialization for the long-range sensor on the back screen.

Since the game is made in GI mode, the area of numbers 0 to 2000 of VRAM is vacant, and this vacant area is used for data. The main program receives and processes commands and processes the long-range sensor on the back screen.

Chart 1 indicates the amounts of energy and time lost by processing each command.

CHART 1:

Command 1	Energy:	$(\text{X distance})^2+(\text{Y distance})^2$
	Time:	2 units
Command 2	Energy:	$\{(\text{X distance})^2+(\text{Y distance})^2\}\times 2$
	Time:	3 units
Command 3	Energy:	200 units
	Time:	3 units
Command 4	Energy:	$\{(\text{X distance})^2+(\text{Y distance})^2\}\div$
	Time:	2 units

LIST OF VARIABLES: PROGRAM I

R1: Invader distribution

R2: Base distribution
B: Base distribution for short-range sensor
K: Invader distribution for short-range sensor
S: Star distribution for short-range sensor
M
N: Intermediary variables for subroutine
I, J: Intermediary variables
AD: Data writing variables
X: X co-ordinate for short-range sensor
Y: Y co-ordinate for short-range sensor
D: Address

PROGRAM MAP: PROGRAM I

10~50: Initial setting
60~80: Initial setting for a sensor
90~130: Distribution of bases, invaders, and stars
140~180: Writing of initial image and long-range sensor
190~230: Subroutine

LIST OF VARIABLES: PROGRAM II

K: Total number of invaders
EN: Energy
TI: Time
MI: Missiles
QX, QY: Co-ordinates of a macrocosm
SX, SY: Co-ordinates of a microcosm
R: Initial address for a screen
D: Data area pointer
C $
C
Z: Variables for checking
X, Y: Co-ordinate input
CH: Reading the ASCII code
A, B: Checking long-range sensor
A, D: Short-range sensor address
B, D: Cosmos Address
I, J: Intermediary variables

PROGRAM MAP: PROGRAM II

10~90: Initial value and image setting
100~120: Generating short-range sensor
130~140: Command processing
150~190: Impulse drive processing
200~230: Warp processing
240~320: Long-range sensor processing
330~390: Missile processing
400: Processing when the spaceship wins
410: Processing when the spaceship loses
420: END processing
430~450: Subroutine for inputting co-ordinates
460: Timer routine

PROGRAM LIST

PROGRAM I

```
10 dim R1(48),R2(48)
20 randomize:print "█████ Working..."
30 for I=0 to 48:vpoke &0100+I,0:next
40 for I=0 to 146:vpoke I,0:next
50 for I=&1000 to &1F80:vpoke I,0:next
60 let P=rnd(15)+15:for I=0 to P:let A=rnd(48):let R1(A)=R1(
A)+1:next
70 let P=rnd(2)+2:for I=0 to P:let A=rnd(48):let R2(A)=R2(A)
+1:next
80 for I=0 to 144 step 3:vpoke I,R2(I/3):vpoke I+1,R1(I/3):v
poke I+2,rnd(5)+4:next
90 for I=0 to 144 step 3:let B=vpeek(I):let K=vpeek(I+1):let
 S=vpeek(I+2):let AD=&1000+I/3*81
100 let M=B:let N=66:gosub 190
110 let M=K:let N=73:gosub 190
120 let M=S:let N=42:gosub 190
130 next I
140 print "█"
150 print cursor(7,0);"Long Range Sensor"
160 for I=0 to 6:print cursor(I*4+4,2);I:next
170 for I=0 to 6:print cursor(1,I*2+4);I;:for J=0 to 6:print
cursor(J*4+4,I*2+4);"***":next:next
180 print "██":end
190 if M=0 then return
200 for J=1 to M
210 let X=rnd(8):let Y=rnd(8):let D=AD+X+Y*9
220 if vpeek(D)<>0 then goto 210
230 vpoke D,N:next:return
```

PROGRAM II

```
10 view:let K=0:for I=1 to 145 step 3:let K=K+vpeek(I):next:
randomize
20 let R=&3863:let EN=3000:let TI=100:let QX=rnd(6):let QY=r
nd(6):let MI=10
30 print "████";cursor(3,1);"012345678█→→ ──────────── "
40 for I=0 to 8:print cursor(0,I+3);I;"←██→→→→|":next:print
cursor(2,12);" └────────────┘ "
50 print cursor(17,3);"ENERGY :";cursor(19,5);"TIME :";curso
r(16,7);"MISSILE :"
60 print cursor(16,9);"INVADER :";cursor(15,11);"POSITION :"
70 let D=&1000+QX*81+QY*567
80 let SX=rnd(8):let SY=rnd(8):let AD=D+SX+SY*9:if vpeek(AD)
<>0 then goto 80
90 for I=0 to 8:for J=0 to 8:vpoke R+I*32+J,vpeek(D+I*9+J):n
ext:next:vpoke R+SX+SY*32,69
100 view:if(EN<0)+(TI<0)+(MI<0)then goto 400
110 if K=0 then goto 410
120 print cursor(25,3);"█";EN;cursor(25,5);"█";TI;cursor(25,
7);"█";MI;cursor(25,9);"█";K;cursor(25,11);QX;"-";QY;
130 view 2,14,31,23:print "█COMMAND :";:input,C$:let C=val(C
$):if(C<1)+(C>4)then goto 130
140 if C=4 then goto 330 else if C=3 then goto 240 else if C
=2 then goto 200
150 print "Drive from (";SX;",";SY ")"
160 let Z=8:gosub 430:let EN=EN-((X-SX)^2+(Y-SY)^2):let TI=T
I-2
170 let CH=vpeek(R+X+Y*32):if(CH=42)+(CH=75)then print "█Can
't go":goto 160
180 if CH=66 then let EN=3000:let MI=MI+10
```

```
190 vPoke R+SX+SY*32,0:vPoke R+X+Y*32,69:let SX=X:let SY=Y:g
oto 100
200 Print "WarP from (";QX;",";QY ")"
210 let Z=6:gosub 430:let EN=EN-((X-QX)^2+(Y-QY)^2)*2:let TI
=TI-3
220 let QX=X:let QY=Y:if rnd(9)>3 then goto 70
230 let C=rnd(20)+10:Print "■Invader Attack --";C:let EN=EN-
C:gosub 460:goto 70
240 Print "■":for I=0 to 2:for J=0 to 2:let A=QX-1+I:let B=Q
Y-1+J
250 if(A<0)+(A>6)+(B<0)+(B>6)then goto 270
260 vPoke &0100+A+B*7,1
270 next:next
280 for I=0 to 6:for J=0 to 6:if vPeek(&0100+J+I*7)=0 then g
oto 300
290 for L=0 to 2:Print cursor(J*4+4+L,I*2+4);num$(vPeek(J*3+
I*21+L));:next
300 next:next:Print cursor(8,19);"POSITION :";QX;"-";QY
310 let TI=TI-3:let EN=EN-200:Print cursor(13,22);"Hit SPace
 Key";
320 if Peek(&702B)<>7 then goto 320 else Print "■":goto 100
330 Print "MISSILE":let Z=8:gosub 430:let AD=R+X+Y*32:let CH
=vPeek(AD)
340 if CH=0 then Print "MIssed !!":goto 390
350 if CH=83 then goto 400
360 if CH=42 then Print "Star";:let Z=2 else if CH=66 then P
rint "base";:let Z=0 else Print "Invader";:let Z=1
370 Print " Destroyed !!"
380 vPoke AD,0:vPoke D+X+Y*9,0:let K=K-1:let BD=QX*3+QY*21+Z
:vPoke BD,vPeek(BD)-1
390 let MI=MI-1:let TI=TI-2:let EN=EN-((X-SX)^2+(Y-SY)^2)/2:
gosub 460:goto 100
400 Print "■→→You are defeated !!":goto 420
410 Print "■→→You are winner !!"
420 Print "■■■■ Game Over !!":end
430 Print "X=";:inPut,X:if(X<0)+(X>Z)then goto 430
440 Print "Y=";:inPut,Y:if(Y<0)+(Y>Z)then goto 440
450 return
460 for I=0 to 8000:next:return
```

PUZZLE SQUARE

Directions:

Type RUN. The computer displays 15 numbered pieces and asks the number of times to arrange the pieces as "Difficulty (minimum of 1)?" on the screen. The larger the input number is, the more difficult rearranging the pieces in order becomes. You might input a number between 30 and 50 for a start. After a number is input, the computer moves the pieces as often as the number input. The game starts when the computer finishes mixing them up.

The keys to move the pieces are as follows:

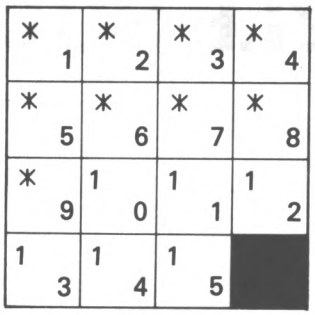
1. To move a piece above the blank downward ... ↓ key
2. To move a piece below the blank upward ... ↑ key
3. To move a piece from the left of the blank to the right ... → key
4. To move a piece from the right of the blank to the left ... ← key

The pieces can be moved only within the square frame. The computer counts the number of times a piece is moved, and displays it in the upper left of the screen. This number should be smaller than the number of times the computer initially arranged the pieces because the way the computer mixes up the pieces is quite inefficient. Encouraging, isn't it?

If the pieces are already arranged in proper order at the beginning of the game, move a key a couple of times, e.g., exchange piece 15 and the blank, then exchange their positions again, to finish the game.

If you cannot complete the puzzle, you can ask the computer for help. Press the H key, and "GIVE UP" will be displayed on the screen. Then press any key to move each remaining piece to the right position until the puzzle is completed.

Diagram 1: Right Position of the Pieces



However, if the sum of the numbers of moves by you and the computer is 750 or more, the computer will be unable to finish the puzzle and the game ends.

After the game is over, the computer asks if you would like to "REPLAY (y/n)?" on the screen. Depress the N key to quit, and the Y key to replay the game. Those of you who completed the puzzle with the computer's help will depress the Y key to try again, won't you?

Program:

This program is written in the GII mode and is divided into four parts — inputting, mixing up, shifting, and giving up. The computer reads the data of line numbers 0 through 9, and the "*" character from VRAM. 15 sprites are used to assign the 15 pieces, which are displayed on the screen with MAG3.

Pieces are shifted by producing random numbers from 0 to 3, one corresponding to each piece. This way, when a piece should be moved beyond the frame, the computer reproduces a random number.

The answer to help you when you GIVE UP is given by putting the data of your own moves so far together with the computer's shuffling at the beginning together in VRAM, and by reprocessing it in reverse. If the sum of the numbers shifted by you and by the computer is 750 or more, VRAM space will be insufficient, and the program cannot respond to the GIVE UP command. Shifting pieces are processed by line numbers 410 to 430, a subroutine which counts the number of times shifted (pp).

When you wish to change the color of a piece, change the number after "SCOL, I" in lines number 70 and 80 to the number of the desired color. To change the color of a background or piece, change the number in quotation marks " " of STCHR RPT$ (16 "7") to the number of the desired color.

LIST OF VARIABLES:

D:	Number displayed
C $:	Data of numbers

DI: Difficulty
V: Number to fix direction of movement
BX, BY: Spot of the blank
CX, CY: Spot of the numbered piece to be moved
A $: Direction of movement
P: Number of times shifted by a player
PP: Number of times shifted by the player and the computer

PROGRAM MAP:

10: Initial setting
20~110: Screen setting
120~130: Difficulty input
140~200: Mixing up
210~310: Shifting
320~400: GIVE UP
410~430: Subroutine to shift pieces

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23:dim D(4,4)
20 for I=124 to 183:stchr "0000000000000000" to I:next
30 let C$="":for I=0 to 7:let C$=C$+right$(hex$(vPeek(&2000+
42*8+I)),2):next:for I=124 to 156 step 4:stchr C$ to I:next
40 let C$="":for I=0 to 7:let C$=C$+right$(hex$(vPeek(&2000+
49*8+I)),2):next:for I=160 to 180 step 4:stchr C$ to I:next
50 for I=1 to 15:let C$="":for J=0 to 7:let C$=C$+right$(hex
$(vPeek(&2000+8*(48+(I mod 10))+J)),2):next
60 stchr C$ to 123+I*4:next
70 for I=1 to 15:scod I,120+4*I:next:mag 3:for I=1 to 4:scol
 I,10:next
80 for I=5 to 8:scol I,6:next:for I=9 to 12:scol I,4:next:fo
r I=13 to 15:scol I,12:next
90 for I=4 to 6:stchr rPt$(16,"7")to 128,I:stchr rPt$(16,"1"
)to 150,I:next
100 cls 128:for I=3 to 21:if I=7 or I=12 or I=17 then next e
lse for J=0 to 3:Print cursor(6+J*5,I);"■■■■":next:next
110 for I=1 to 4:for J=1 to 4:loc(I-1)*4+J to 6*8+(J-1)*5*8,
3*8+(I-1)*5*8:next:next
120 Print cursor(1,1);rPt$(30,"■");cursor(1,1);"DIFFICULTY (
 1 ABOVE ) ";:inPut DI:if DI<1 then Print "■":goto 120
130 Print cursor(1,1);rPt$(30,"■");cursor(15,1);DI
140 for I=1 to 4:for J=1 to 4:let D(I,J)=(I-1)*4+J:next:next
:let D(4,4)=0
150 let BY=4:let BX=4:randomize:for I=1 to DI
160 let V=rnd(3):if V=0 and BY<>1 then let CY=BY-1:let CX=BX
:gosub 410:goto 200
170 if V=1 and BX<>4 then let CY=BY:let CX=BX+1:gosub 410:go
to 200
180 if V=2 and BY<>4 then let CY=BY+1:let CX=BX:gosub 410:go
to 200
190 if V=3 and BX<>1 then let CY=BY:let CX=BX-1:gosub 410:go
to 200 else goto 160
200 next
210 let A$=inkey$:if A$="" then goto 230
220 if A$="h" then goto 320
230 if A$="/" and CY<>1 then let V=0:let CY=BY-1:let CX=BX:g
osub 410:goto 270
```

```
240 if A$=";" and CX<>4 then let V=1:let CY=BY:let CX=BX+1:g
osub 410:goto 270
250 if A$="@" and CY<>4 then let V=2:let CY=BY+1:let CX=BX:g
osub 410:goto 270
260 if A$=":" and CX<>1 then let V=3:let CY=BY:let CX=BX-1:g
osub 410:goto 270 else goto 210
270 let P=P+1:Print cursor(1,1);P;"TIME":for I=1 to 4:for J=
1 to 4:if D(I,J)<>(I-1)*4+J then goto 210
280 if I=4 and J=3 then goto 290 else next:next
290 Print "[CLR]";:for I=1 to 1000:next:Print "[CLR]";
300 Print cursor(1,22);"REPLAY (y/n) ?":let Z$=inkey$:if Z$<
>"y" and Z$<>"n" then goto 300
310 if Z$="y" then let P=0:let PP=0:goto 100 else end
320 Print cursor(1,1);"GIVE UP";rpt$(23,"■")
330 if PP>750 then Print "[CLR]":goto 300
340 for I=PP to 1 step-1:let V=vpeek(15360+I)
350 if inkey$="" then goto 350
360 if V=0 then let CY=BY+1:let CX=BX:gosub 410:goto 400
370 if V=1 then let CY=BY:let CX=BX-1:gosub 410:goto 400
380 if V=2 then let CY=BY-1:let CX=BX:gosub 410:goto 400
390 if V=3 then let CY=BY:let CX=BX+1:gosub 410:goto 400
400 next:goto 300
410 let PP=PP+1:if PP>750 then goto 420 else vpoke 15360+PP,
V
420 loc D(CY,CX)to 6*8+(BX-1)*5*8,3*8+(BY-1)*5*8
430 let D(BY,BX)=D(CY,CX):let D(CY,CX)=0:let BY=CY:let BX=CX
:return
```

DEMONSTRATION PROGRAMS

HOURGLASS WITH A CHIME
LORELEI
DIGITAL CLOCK

HOURGLASS WITH A CHIME

Content:

Type RUN. An hourglass appears on the screen. Once you set the time sand begins to flow through the channel. When all the sand has finished flowing the chime rings. (Unlike an ordinary hourglass, you don't have to keep watching it to know the time.)

Directions:

Input the program and type RUN. The program draws an hourglass and asks "How many minutes?" on the screen. Input a number of minutes desired (206 maximum), and the sand begins to flow through the channel. When the flowing ends, the chime rings.

Input another number of minutes desired to reset, and the number 0 to stop. If you wish to stop processing while the program is still on, press the E key, then input the number 0. If you use the SHIFT + RESET keys instead, "Ready" will appear on the screen. In this case, try to RUN again first. If this does not work, input "VIEW 0, 1, 30, 23" and press the RETURN key. When "Ready" appears on the screen, process RUN 1250, and the screen will be cleared.

If a number more than 206 is input for the number of minutes, the program might end too quickly or errors might occur. If the program stops due to an error, RUN immediately to recondition the program.

Program: This is how the program works.

This program incorporates 8 sprites; 5 to draw the hourglass and 3 to express the sand flowing. The frame of the hourglass and the sand in the upper part of the hourglass are made with characters. The sand flowing is expressed as follows. To show the decrease of sand in the upper part, a black sprite is moved down little by little over the light blue sand. To show the increase of sand in the lower part, a light blue sprite is moved up little by little. To express the movement of sand flowing, the center part of the sprite used in the lower part of the hourglass is cut off with a VIEW com-

mand and slowed down.

The OUT command rings the chime.

LIST OF VARIABLES:

A $:	Data reading when the sprites are set
I, J, Q:	FOR-NEXT
S1:	Data reading when the chime rings
S2:	Same as the above
T:	Time input
T1:	Time

PROGRAM MAP:

30~40:	Sprite pattern setting
50~70:	Sprite setting
90~140:	Sprite color setting
150~220:	Sprite position setting
230~300:	Character color setting
150~480:	Hourglass drawing
500~640:	Sand flowing
660~970:	Sprite pattern data
980~1050:	Time input subroutine
1060~1150:	Sound making part
1180~1240:	Sound data
1250~1300:	Screen and sound clearing

PROGRAM LIST

```
10 print "■":cls
20 view 1,0,30,23
30 restore
40 for I=40 to 71:read A$:stchr A$ to I:next
50 for I=0 to 7
60 scod I,40+4*I
70 next I
80 mag 3
90 scol 0,14
100 scol 1,14
110 scol 2,14
120 scol 3,1
130 scol 4,8:scol 5,8
140 scol 6,1:scol 7,7
150 for I=0 to 2
160 loc I to 104,72+32*I
170 next I
180 loc 3 to 104,104
190 loc 4 to 104,40
200 loc 5 to 104,168
210 loc 6 to 104,88
220 loc 7 to 104,136
230 for I=4 to 6
240 for J=252 to 255
250 stchr "8080808080808080" to J,I
260 next J
```

```
270 stchr "8080808080808080" to 128,I
280 stchr "8080808080808080" to 250,I
290 stchr "7777777777777777" to 177,I
300 next I
310 Print cursor(9,6);"▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀"
320 Print cursor(9,7);"▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀"
330 Print cursor(9,8);"▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀"
340 Print cursor(10,9);"▼              ▼"
350 Print cursor(10,10);"◥              ◤"
360 Print cursor(10,11);"◢   ●●●●     ◣"
370 Print cursor(10,12);"◤   ●●●●     ◥"
380 Print cursor(10,13);"◣   ●●●●     ◢"
390 Print cursor(10,14);"◥   ●●●●     ◤"
400 Print cursor(10,15);"◢              ◣"
410 Print cursor(10,16);"◤              ◥"
420 Print cursor(10,17);"◣              ◢"
430 Print cursor(10,18);"◥              ◤"
440 Print cursor(10,19);"◢              ◣"
450 Print cursor(10,20);"▼              ▼"
460 Print cursor(9,21);"▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
470 Print cursor(9,22);"▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
480 Print cursor(9,23);"▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄";
490 gosub 980
500 for I=0 to 32
510 loc 6 to 104,56+I
520 loc 7 to 104,168-I
530 for J=1 to 200:next J
540 view 15,15,17,19:for Q=1 to T1
550 Print cursor(1,Q mod 4);"."
560 Print cursor(rnd(5),1);"▪";
570 Print cursor(rnd(1),rnd(2));" "
580 if inkey$="e" then goto 20
590 Print "■":next Q
600 view 1,0,30,23
610 next I
620 view 15,15,17,19
630 for I=1 to 3:Print "■":next I
640 view 1,0,30,23
650 goto 1060
660 data "8080808080808080"
670 data "8080808080808080"
680 data "0101010101010101"
690 data "0101010101010101"
700 data "8080402010080402"
710 data "0204081020408080"
720 data "0101020408102040"
730 data "4020100804020101"
740 data "8080808080808080"
750 data "8080808080808080"
760 data "0101010101010101"
770 data "0101010101010101"
780 data "000080c0e0f0f8fc"
790 data "fcfefcfefcf8fcf8"
800 data "000001030703f1f3f"
810 data "7f7f3f7f3f1f3f1f"
820 data "0000000000000000"
830 data "ffffffffffffffff"
840 data "0000000000000000"
850 data "ffffffffffffffff"
860 data "ffffffffffffffff"
870 data "ffffffff00000000"
880 data "ffffffffffffffff"
890 data "ffffffff00000000"
900 data "ffffffffffffffff"
910 data "fffffffffff1f0701"
```

```
920 data "ffffffffffffffff"
930 data "fffffffffffce080"
940 data "0000030f3fffffff"
950 data "ffffffffffffffff"
960 data "0000c0f0fcffffff"
970 data "ffffffffffffffff"
980rem INPUT
990 view 1,1,30,1
1000 input "HOW MANY MINUTES?";T
1010 if T=<0 then goto 1250
1020 view 1,0,30,23
1030 Print cursor(1,1);"■■"
1040 let T1=65*T+T*T*2
1050 return
1060rem CHIME
1070 restore 1180
1080 read S1,S2
1090 if S2=0 then goto 490
1100 out 32,&80+S1
1110 out 32,S2
1120 for I=0 to 15
1130 for J=1 to 100:next J
1140 if S1=0 then goto 1160
1150 out 32,&90+I
1160 next I
1170 goto 1080
1180 data &6,&d,&a,&a,&e,&b
1190 data &d,&11,0,1,&6,&d,&e,&b
1200 data &a,&a,&6,&d,0,1
1210 data &a,&a,&6,&d,&e,&b
1220 data &d,&11,0,1,&d,&11
1230 data &e,&b,&a,&a,&6,&d
1240 data  0,0
1250 for I=0 to 7
1260 loc I to 100,1000
1270 next I
1280 view 1,0,30,23:cls
1290 out 32,&9F
1300 end
```

MUSIC BOX TUNE : LORELEI

The essential qualities of sound are not lost with the M5's music creation program.

Content:

This program automatically plays the tune Lorelei. It performs a duet using the M5 sound generator.

Directions:

Type in the program as it is; when you type RUN, the music is played automatically. When you want to stop it, execute RUN 360.

Look at "Enjoy Computerized Music" on page 137 for explanation of this program.

PROGRAM LIST

```
10rem lorelei
20 clear 256,&7DFF
30 Print "■when the music stoPs....run360 "
40 for I=0 to 110
50 read A:Poke &7E00+I,A:next
60 restore 220:for I=0 to 114
70 read A:Poke &7EA0+I,A:next
80 restore 330:for I=0 to 4
90 read A:Poke &7064+I,A:next
100 restore 340:for I=0 to 4
110 read A:Poke &7072+I,A:next
120 data &57,&41,&3f,&70,&36
130 data &22,&88,8,&88,12,&8a,4,&60,8,8,&23,1,&22,12,10
140 data &88,24,&86,16,6,&85,16,5,3,1,3,&85,32,0
150 data 8,&88,12,&8a,4,8,&23,1,&22,12,10
160 data &88,24,&86,16,6,&85,16,5,8,6,3,&81,32
170 data 0,5,&83,12,&85,4,3,8,3,3,&8c,24,&8a,16,10
180 data &88,16,8,7,8,10,&88,32,0,8
190 data &88,12,&8a,4,8,&23,1,&22,12,10,&88,16
200 data &23,5,&83,16,3,&81,16,1,&22,12,10
210 data 12,&23,&81,32,0
220 data &57,&41,&70,&36,&3f,&60,8
```

```
230 data &22,8,&85,12,&86,4,5,10,8,6
240 data &85,24,&83,16,&83,8,&81,16,1,&21,12,&22,1,&21,12
250 data &22,&81,24,1,0,8
260 data &85,12,&86,4,5,10,8,6
270 data &85,24,&83,16,3,&81,16,1,&21,12,12,12
280 data &22,&81,24,1,0,1
290 data &21,&8c,12,&8c,4,12,12,12,12,&22,&83,24,&81,16,1
300 data &21,&8c,16,12,&22,&83,16,1,&21,&8c,24,12,0
310 data &22,8,&85,12,&86,4,5,10,8,6
320 data &85,24,&86,16,6,&85,16,5,&88,16,6,&85,24,5,0
330 data &00,&7e,&f0,111,0
340 data &a0,&7e,&f0,115,0
350 end
360 Poke &7067,0:Poke &7068,0
370 Poke &7075,0:Poke &7076,0
```

KEYING IN LONG PROGRAMS AND DEBUGGING

• When keying in is interrupted

It is ideal to be able to key in a single program at one time; however, there are those inevitable times when you will be interrupted.

At such times do not be reluctant to stop; the part of the program which had already been keyed in has been saved on cassette tape. When you have time to resume keying in the program, load that part which had been saved onto cassette, list and read it. All you have to do now is to continue keying in.

When you have finished keying in the whole program, the first part which you had saved is now no longer needed, so the now completed program can be saved on the cassette tape by recording over it. You only need to save the finished program.

• Debugging long programs

No matter how carefully you key in a program there are bound to be errors. Programs with errors in them cannot be run on the M5. Parts of programs with errors in them are referred to as being "buggy." Removing these errors is called "debugging".

The longer the program the greater the likelihood of bugs. Ideally, after the program has been keyed in and saved on cassette tape, it should be given a test run. Some errors may cause the program to be erased. It's a good idea to get in the habit of patiently saving programs.

DIGITAL CLOCK

This program transforms the TV screen into a beautiful color digital clock. It also can be used as an alarm clock.

Directions:

Enter RUN and "What time is it now (hhmmss)?" will appear. Either the present time or any time you wish can be put in. The order of hours, minutes, and seconds must be followed. Each of the times must be entered in double-digit form. Also, PM times must be entered as 13:00 to 23:00. EXAMPLE; 2:09.06PM ... 140906 [RETURN]

If the hour entered is greater than 23 and the minute and second greater than 59, the buzzer will sound and they must be corrected.

After the time has been entered, a large "hour: minute", and below that to the right in a smaller size "second", will be displayed. Also, the alarm can be set to sound for any minute.

When you wish to set the time for the alarm, first push the "a" key. "ALARM" will be displayed in the upper left portion of the screen. To set the time for the alarm, enter in the hour and minute in that order using double-digits for each. (This time cannot be indicated in seconds.) For example, if you wish to have the alarm go off at 6:30 AM, enter 0630. "06:30" will be displayed to the right of the previously indicated "ALARM". The alarm time will then have been set.

It you make a mistake in entering the time or if you wish to change the alarm time, push the "a" key again and the time can be reset.

This alarm will continue to sound until the "s" key is pushed.

When entering the alarm time make sure that no characters other than numbers are used. To stop this program, push the "SHIFT + RETURN" keys.

Program:

This program was created in the GII mode.

Within the program the hours, minutes, and seconds are handled separately. If the hours are converted to minutes and seconds this will create an overflow.

This clock has the same precision as the clock within the M5, and each second is managed by the line numbers 80—280. At line number 280 if the indicated time is different from line number 80 in the inner clock, the time will be considered to have elapsed one second and will be returned to line number 80. Therefore, the time indicated will always be as accurate as the time of the clock within the computer.

In the span of one second (line number 80 — line number 280) "calculation of the time", "indication of the time", "sounding the time", "sounding the alarm", and "entering the alarm time" are managed in that order.

If this clock is actually used as an alarm clock, the sounding of the time each minute will probably become bothersome, so when entering the program eliminate line numbers 150 and 160. This will stop the time from being sounded.

If you wish the time to be sounded each hour rather than each minute, change S+57 OR S+58 OR S+59 at line number 150 to M+59 AND (S+57 OR S+58 OR S+59) and change S+0 at line number 160 to M+0 AND S+0.

If you wish to change the characters (hours, minutes) used to indicate the time, change line numbers 320—410. Line number 320 indicates "0", line number 330 indicates "1", and so on, with line number 410 indicating "9".

[LIST OF VARIABLE NAMES]

N$:	Present time
HH, MM, SS:	breakdown of present time
T:	Present time converted to second unit (only minutes and seconds)
T1:	Time of the clock within the computer immediately before the clock begins to operate
MU:	Variable which determines whether one second of time of the clock has elapsed or not
T2:	Indicated time (only the minutes and seconds parts are converted into units of seconds)
H, M, S:	Breakdown of indicated time
C:	Number to be indicated
X:	Place of indication (X coordinate)
AH:	Hour set for the alarm
AM:	Minute set for the alarm
ZZ:	Variable used to check the alarm time
Y$:	Variable for setting the alarm time
B, BB:	Variables for indicating the alarm time

A$, CA$: Variables for reading data statements
Z: Variable which determines whether the alarm time has been set or not

[PROGRAM MAP]

10: Initial setting
20~70: Entering present time
80~160: Indicating the time
170~280: Alarm
290~300: Subroutine for writing the numbers
310~420: Data for numbers

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23:dim D(4)
20 for I=4 to 6:stchr "4747474747474747" to 146,I:stchr "a0a
0a0a0a0a0a0a0" to 128,I:next I:scod 1,225:scod 2,225:let AH=
-1
30 cls:Print cursor(1,10);"WHAT TIME IS IT (hhmmss) ";:inPut
 N$:if len(N$)<>6 then Print "■":9oto 30
40 let HH=val(left$(N$,2)):let MM=val(mid$(N$,3,2)):let SS=v
al(ri9ht$(N$,2))
50 if HH>23 or MM>59 or SS>59 then Print "■":9oto 30
60 cls 146:for I=1 to 11:Print cursor(2,5+I);:for J=1 to 27:
Print " ";:next J,I
70 loc 1 to 120,72:loc 2 to 120,104:let T=MM*60+SS:let T1=ti
me
80 let MU=time:let T2=MU-T1-(MU-T1)/3600*3600+T
90 let H=T2/3600:let M=(T2-H*3600)/60:let S=T2-H*3600-M*60
100 if M=0 and S=0 and MM<>0 and SS<>0 then let HH=HH+1
110 if HH=24 then let HH=0
120 let C=HH/10:if C=0 then let C=C+10
130 let X=3:9osub 290:let C=HH-HH/10*10:let X=9:9osub 290
140 let C=M/10:let X=17:9osub 290:let C=M-M/10*10:let X=23:9
osub 290:Print cursor(25,19);S
150 if S=57 or S=58 or S=59 then Print "■"
160 if S=0 then out &20,&90:out &20,&80:out &20,&04:for I=1
to 1000:next I:out &20,&9F
170 if HH=AH and M=AM and S=0 then let ZZ=1
180 if ZZ=1 then Print "■":for I=1 to 700:next I:Print "■"
190 let V$=inke9$
200 if V$="a" then Print cursor(2,3):rPt$(11,"▬");cursor(2,3
);"ARARM":let Z=1:let B=0:let BB=7
210 if V$="s" then Print cursor(2,3):rPt$(11,"▬");let ZZ=0
220 if V$="" or Z<>1 or ascii(V$)<48 or ascii(V$)>57 then 9o
to 270
230 let B=B+1:if B=5 then 9oto 270
240 if B=3 then Print cursor(10,3);":";let BB=8
250 Print cursor(BB+B,3);V$:let D(B)=val(V$)
260 if B=4 then let AH=D(1)*10+D(2):let AM=D(3)*10+D(4):let
Z=0
270 scol 1,6:scol 2,6
280 if time=MU then 9oto 280 else scol 1,0:scol 2,0:9oto 80
290 restore 310:for I=1 to C*7+1:read A$:next I
300 for I=1 to 7:read CA$:Print cursor(X,7+I);CA$:next I:ret
urn
```

```
310 data""
320 data" ■■■■ ","■    ■","■    ■","■    ■","■    ■","■    ■","■    ■","■■■■
■ "
330 data"  ■   ","■■■   ","   ■   ","   ■   ","   ■   ","   ■   ","   ■   ","■■■
■ "
340 data" ■■■■ ","■    ■","     ■","    ■ ","   ■   ","   ■   ","■■■■
■■"
350 data"■■■■■■■","    ■ ","   ■   ","    ■ ","     ■","■    ■","■■■■
■ "
360 data"    ■ ","   ■■■ ","  ■ ■  ","■   ■  ","■■■■■■■","    ■ ","
■ "
370 data"■■■■■■■","■      ","■■■■■  ","     ■","     ■","■    ■","■■■■
■ "
380 data"  ■■■ ","  ■    ","■      ","■■■■■  ","■    ■","■    ■","■■■■
■ "
390 data"■■■■■■■","     ■","    ■  ","   ■   ","   ■   ","   ■   ","   ■
  "
400 data" ■■■■ ","■    ■","■    ■","  ■■■■ ","■    ■","■    ■","■■■■
■ "
410 data" ■■■■ ","■    ■","■    ■","■■■■■■","     ■","    ■ ","■■■■
  "
420 data"      ","      ","      ","      ","      ","      ","
  "
```

COMPUTER FORTUNETELLING

WHAT SORT OF LUCK WILL YOU HAVE TODAY?
- Astrology

PREDICTIONS BY THE GREAT ARCANA
- Tarot Fortunetelling

ASTROLOGY

In astrology, the position of the stars are determined for each person when they are born, and that person's fate is guided by their stellar positions.

Do you know your horoscope? If you don't, you can find it by running this program. It will provide you with the special characteristics of your horoscope and also tell you what sort of luck is in store for you.

Content:

In astrology the daily fortunes are divided into the 9 following types.

★ Days of changeable luck — Caution is needed because there are many possibilities of adverse events. If one becomes overconfident, trouble can easily occur in matters related to work or the affairs of the heart. There is a desire to begin new projects, but those other than long-range ones are likely to fail.

★ Days of dissatisfaction — Tendency to lose one's confidence. Desires are unattainable because of too much impatience. Dissatisfaction and discontent easily occur; however, friends come to your aid. It's a good idea to postpone purchases of non-essential items. Negotiations are prone to failure.

★ Fortuitous days for negotiations — These are days when your ideas are recognized. However, it is easy to become overconfident, and dissatisfaction may arise. These are days when expenses for entertainment are quite large. Trouble also can easily occur in regard to the affections.

★ Days of fortune through endeavor — These are days when you are able to regain self-composure and you are brought closer to your goals. Results are commensurate with the effort mode. The morning is not very lucky, however, from noon until night things improve. Good days for shopping.

★ Days of fortune through acquisition — A very lucky time when everything seems to go your way. Social relationships and desires are fulfilled. Very propitious time for shopping, a good day for lotteries.

★ Days of conflict — Lucky feelings are strong, however, these days are filled with busy activity. Personal confrontations increase and relationships with superiors can become stormy. Domestic discord also is likely to occur and caution must be exercised to avoid fines resulting from traffic accidents or violations. However, these are good days for taking examinations or for contests.
★ Days of peace — The feeling of luck in the morning is not very strong; however, this improves in the afternoon and into the night. Supporters are met by chance. Objects which you thought you had lost are again found. There is some loss in positivity; however, you are about 90% satisfied with everything.
★ Days of prosperity — Days of good fortune. Very lucky in contests and unexpected financial luck. Everything goes as you like. But this day is the apex, after it everything goes downhill so it is best to go to bed early. Because you are prone to be proud and arrogant at these times, caution must be exercised so that adversity does not follow.
★ Days of adversity — These are days of culminated pessimistic misfortune. Whatever is attempted seems to be forestalled and ends in bone-breaking loss. These are days of strong independent feelings when illnesses become even worse. With adverse affects on the emotions, caution must be exercised against theft and losing things.

Directions:

Type RUN. You are first asked for your date of birth. This must be entered in the order of year, month and day.

Next, the day to be read is asked and the fortune for any date between the years 1800 and 2100 can be entered. The fortune is immediately indicated while in the background the 12 planets of the zodiac flow in order throughout your horoscope. The tune "Homeward" is played.

The sprite and the music are in a endless loop so in order to stop it hit the SHIFT + RESET keys. To stop the sound execute RUN 270. The sprite at the top of the screen will disappear and the sound will cease when the music ends.

Release STCHR by entering the CTRL + T keys (or CTRL + S) when switching to another program first push the CTRL + T keys or the CTRL + S keys, and this will release the STCHR.

Program:

The program is divided into two parts, Program I (program for entering the sprite) and Program II (main program). Make certain you do not make any mistakes when putting in the information.

1) Enter Program I, SAVE it on tape, then RUN it.
2) Enter Program II and SAVE it on tape.

Because this program runs on GII mode, push the CTRL + 0 keys when

it is on GI mode in order to switch over to GII mode. When RUN is entered the program will begin.

Program:

STCHR is used to produce various colors for 38 dots used to write the first point (intended to represent a planet) on the result indication screen.

The planets, houses and moon use sprites.

Separate sprites are set up for the houses and those dots used between the sections which make the lines between the stars and the parts of stars. Only the parts of the stars should flicker.

The astrological section is relatively simple, comprised of only the 9 divisions of the fortunes for every day and the quality of the 12 signs of the zodiac.

[LIST OF VARIABLE]

ZA: Date of birth
DA: Date to be read
O, PO: Variables to determine the planets
R$: Fortune of the day selected to be read
BA$: Planets
G$, J$: Quality of Planets

[PROGRAM MAP]

10~50: Fortune determination
60~200: Result indication screen construction
210~250: Calendar determination
260: Sprite and sound control
270~340: Data relating to fortunetelling
350: Data for sound

[Sprite]

10~30: Creation of sprite
40: Creation of sound
50~350: Data for sprite
360~390: Data for sound

PROGRAM LIST

PROGRAM I

```
10 clear 256,&7F7F:Print "■"
20 for I=0 to 123:read A$:if A$="" then let A$=rPt$(16,"0")
30 stchr A$ to I:next
40 for I=&7F80 to &7FFE:read A:Poke I,A:next
50 data 20702000000000000,0000000000020702,,000040E040000000
60 data 0000001008040201,,,8000000000800000
```

```
70 data 040E257020000103,0100020000010000,0000000000002080,
2000001000022702
80 data 0000010000080400,,0000008040000000,0000D00008000000
90 data 0200103810000000,0008022000000104,081D880000000000,
0020000000080CA80
100 data 0005000000001008,0000100804040000,0000200080404020,
0000001010000004
110 data 0001000000000000,0100001038100000,80C0800000000080,
c080000000020702
120 data ,0000040000000000,0000000080800000,0000100800000000
130 data 0000000000001000,0840E04000000000,0400012000207020,
0004001038100000
140 data 0000000000000720,0010000000000000,000A100000000000,
0800080000000000
150 data 0000000040E0400,0000207020100080,10000A0702800000,
1038100000000000
160 data 0000000000000010,1010000000400000,0000000018204020,
170 data 2070200000000000,0000000000081C08,0000000000040E0,
4000000000000000
180 data 0000000804020100,0000010204000000,,0080000000000000
190 data ,0020702140801002,0401103812004000,400080C080008000
200 data ,0000000000006904,0208020200400040,
210 data 0000000000207220,0080002000000000,8000000088002000,
0000207020000800
220 data 0000000000010000,4000400000000000,0000800000500000,
2000000000100000
230 data 00000000000040E0,4000100000000000,0000000002070204,
240 data ,002F000000000000,,F800000000000000
250 data 0000010010044E04,0000000000000080,0080C08010381000,
0001000000000000
260 data 0000000002200000,4040808080800000,0000000000000004,
0200000000000000
270 data 080040E042070200,,,0000000040E0401
280 data 0010000018000000,,0000000000000080,4020100800000000
290 data 04060e0e1c3c78f0,,,
300 data 00010307 0f1f3f7f,,80c0e0f0f8fcfeff,
310 data 1f1f1c1c1f1c1c1f,1f1f000000000000,fcfc9c9cfc9c9cfc,
fcfc000000000000
320 data ,,0000000000000000c,0c0c0c0c04000000
330 data ,0000000030 70c080,,
340 data ,0000010300000000,,0000c08000000000
350 data ,,,0000070e00000000
360 data &70,&40,&44,34,96,12,5,&88,4,&88,16,5,&83,4,&81,16
,3,&85,4,8,&85,4,&83,32,5,&88,4,&88,16,5
370 data &83,4,&81,16,&83,8,&85,8,3,&81,4,&81,32,10,35,&81,
4,&81,16,96,8,34,12,8,&8a,16,10,35,1,34,12,8,&8a,32
380 data &8a,12,35,&81,4,&81,16,34,12,8,&8a,16,10,35,1,34,1
2,8,&8a,32,96,12,5,&88,4,&88,16,5,&83,4,&81,16,3
390 data &85,4,8,&85,4,&83,16,&84,16,5,&88,4,&88,16,35,1,&8
3,4,&85,16,3,&81,4,&83,8,34,10,35,&81,32
```

PROGRAM II

```
10 cls:view 1,0,31,23
20 Print cursor(5,5):"BIRTHDAY?":gosub 210:let ZA=DA:let PO=
MO*100+ZA
30 cls:Print cursor(5,5):"DATE?":gosub 210:restore
35 let VV=abs(abs((DA-ZA)<0)*9-(abs(DA-ZA)mod 9)):if VV=9 th
en let VV=0
40 for Q=0 to VV:read R$:next:restore 300
50 read Q,SU,BA$,G$,J$:if PO>Q then goto 50
60 cls:for J=123 to 160:randomize:let Y=rnd(22)
70 stchr "0000001818000000" to J,Y/8+1
```

```
80 stchr rPt$(8,num$(rnd(9))+"0")to J,Y/8+4:Print cursor(rnd
(31),Y);chr$(J);:next
90 Print cursor(7,5);"TODAY IS";cursor(7,7);"< ";R$;"    >";
cursor(7,11);BA$;" NATURE";
100 Print cursor(7,13);"< ";G$;" >";cursor(7,15);"< ";J$;" >
";
110 restore 370:for I=&7064 to &7068:read A:Poke I,A:next
120 mag 3:scod 0,SU:scod 1,SU+4:scod 6,112
130 scod 2,100:scod 3,104:scod 4,108:scod 5,96:scol 2,8:scol
 3,4:scol 4,&07:scol 5,10
140 for Y=150 to 10 steP-1:loc 5 to 10,Y:next
150 for X=-32 to 180:loc 2 to X,156:loc 3 to X,172:loc 4 to
X,140:next:let K=0:let E=0
160 let Y=-10:scol 0,15:scol 1,15:for X=-32 to 130:loc 0 to
X,Y:loc 1 to X,Y
170 scol 5,E+8:scod 6,(56+2*E)*2:scol 6,15:loc 6 to 206,120:
for C=0 to 3:scol 0,rnd(15)
180 for T=0 to 50:next:next:let E=(E+1)mod 3
190 for T=0 to 50:next:let Y=X+X/2:next:let SU=(SU+8)mod 96
200 scod 0,SU:scod 1,SU+4:goto 160
210 Print cursor(8,7);"YEAR   ";:inPut,YE:if YE<1800 or YE>
2100 then goto 210
220 Print cursor(8,9);"MONTH   ";:inPut,MO:if MO<1 or MO>12
 then goto 220
230 Print cursor(8,11);"DAY   ";:inPut,DA
240 if DA<1 or DA>30+(MO-(MO>7))mod 2 then goto 230 else if
MO-2 then return
250 if DA>28-(YE=2000 or((YE mod 100)<>0 and(YE mod 4)=0))th
en goto 230 else return
260 for I=0 to 6:loc I to 0,192:next:Poke &7067,126:end
270 data DAY OF CHANGEABLE LUCK,DAY OF DISSATISFUCTION,FORTU
ITOUS DAY FOR NEGOTIATION,DAY OF ADVERSITY
280 data DAY OF FORTUNE THROUGH ENDEAVOR,DAY OF FORTUNE THRO
UGH ACQUISITION,DAY OF CONFLICT
290 data DAY OF PEACE,DAY OF PROSPERITY
300 data 120,72,CAPRICORN,PATIENT,SELF-SENTERED,219,80,AQUAR
IUS,SOLITUDE,STUBBORN
310 data 320,88,PISCES,OPEN MINDED,FICKLE,420,0,ARIES,BOASTF
UL,PASSIONATE
320 data 521,8,TAURUS,TOLERANT,TENACITY,621,16,GEMENI,DEFT,S
ENSITIVE
330 data 723,24,CANCER,SENSIBLE,ROMANTIC,823,32,LEO,INDEPEND
ENT,LONELYU
340 data 923,40,VIRGO,PRECISE,ALOOf,1023,48,LIBRIA,JUST,FLAT
TERING
350 data 1122,56,SCORPIO,PROUD,PERMISSIVE,1222,64,SACITIARIU
S,TACENTED,HONEST
360 data 1231,72,CAPRICORN.PATIENT,SELF-CENTERED
370 data &80,&7f,127,200,0
```

TAROT FORTUNETELLING

Predictions by the great arcana

Are you familiar with the tarot cards? They are similar to playing cards and are used for games and reading the future. This is a fortunetelling program designed to use the tarot cards.

Content:

There are a total of 78 cards in a deck of tarot cards. 22 of these are called the Great Arcana while the remaining 56 are called the Lesser Arcana.

The Great Arcana are 22 picture cards beginning with the Magician at 1 and ending with the Fool at 0.

The computer shuffles these cards and deals out 7 cards to be read. The future can be determined by the positioning of the cards.

(If you are interested in learning how to read these cards in more detail, refer to a book on tarot card reading.)

The following is an explanation of each of the 22 cards which make up the Great Arcana.

1 THE MAGICIAN

This is the very first tarot card and represents the beginning of life.

2 THE HIGH PRIESTESS

The Priestess is the only card in the Great Arcana holding writing materials. This card concerns studies and specialized education.

3 THE EMPRESS

This card concerns marriage and indicates a happy marriage.

4 THE EMPEROR

This card represents masculinity and indicates managerial capabilities; it signifies an advanced sense of business. This is the type who sets a goal and then moves directly forward to it.

5 THE HEIROPHANT

This is the teacher or leader whom we have been influenced by and from whom we have received power. It is important to listen to the advice of others.

6 THE LOVERS

This is the card for lovers and means that two people will always move together in the same direction. When doing something, it is best to do it with two or more people.

7 THE CHARIOT

This card indicates strong will. No matter what happens your ideas remain unchanged and you move forward. This card also indicates travel, and as such, represents life as being a kind of long journey.

8 STRENGTH

This card controls human instincts, and indicates ideal human behaviour. It is the form of facing up to the challenges of life.

9 THE HERMIT

This represents people who move ahead steadily at their own pace and are aware of their own abilities.

10 WHEEL OF FORTUNE

This represents the passage of all time throughout the universe. This time is repeated regularly. It is the card of time.

11 JUSTICE

This card can judge matters equitably. It represents moral people with a sense of fairness. It is also related to the courts; bad people will receive judgment.

12 THE HANGED MAN

Life has its difficult moments. Depending on the person they can last for either a long or short time. As a person, one must move ahead and overcome these difficult moments. This card signifies problems and trials.

13 DEATH

This card has two meanings: all things coming to an end or rebirth. This card represents efforts made to start all over again from the beginning in regards to all things.

14 TEMPERANCE

This card signifies the balance achieved between spiritual and material matters. In other words, it is the card which governs regulation.

15 THE DEVIL

Life is full of many kinds of temptation. This card signifies redemption through hard work if one has succumbed to temptation. Through hard work, opportunities arise.

16 THE TOWER

It is the tower struck by lightning. It forewarns the possibility of sudden accident. Illnesses take a turn for the worse. The overall implications of the card are adverse.

17 THE STAR

This card represents human aspirations. It signifies people who have a talent for plans and ideas, and people who are free; also bright futures. It is an ideal condition.

18 THE MOON

This is the card of moods. It shows unstable conditions with rises and falls like that of the moon. However, if given time, good results can be expected.

19 THE SUN

This is a healthy card representative of the good which brings good to all things under the sun. It gives strength to both the spiritual and material aspects of life. This card promises a happy marriage.

20 JUDGMENT

This card signifies revival and rebirth. People suffering from illness improve. Matters take a turn for the better.

21 THE WORLD

All things enjoy good fortune. Since this is the zenith, all that is left is the descent. It new goals are not found, there will be no more forward progress.

0 THE FOOL

This indicates a new start. Starting from 0 there is some instability, however, with a new attitude, the desire for progress should make itself manifest.

If the card is shown reversed, take the opposite meaning of the explanation for the number of that tarot card.

In addition, for each of the 7 tarot cards, the meaning is suggested by its number.

(1) The Past (causes for what lies ahead)
(2) The Present (current situation)
(3) The Future (predictions for the future)
(4) Countermeasures (what will this measure bring?)
(5) Surrounding Conditions
(6) Desires (what the person hopes for)
(7) Ultimate Forecast (final results)

Directions:

After you have entered the program exactly as it is listed, RUN it.

The screen will then turn green and the following will appear: "* * * * Tarot Fortunetelling * * * *", "The cards are now being shuffled. Hit any key to stop the shuffle."

Hit any key on the keyboard. Tarot cards will then appear from 1 to 7.

Look at these cards and read what your future holds for you.

Because it takes awhile for the cards to be shuffled, wait about 30 seconds after you hit the RUN key before hitting another key to stop the shuffle. If you don't, the cards won't get shuffled well.

The cards will be shuffled again after they have been displayed.

When you wish to end this program, push the SHIFT + RESET keys.

Program:

This program does not use the sprite or graphics functions. For shuffling, the RND function is used to change the order of the cards.

The shuffled numbers are read from data statements and displayed on the screen. The reverse decision also uses the RND function. Shuffled numbers look for the 7th, 8th, 9th, 16th, 17th, 18th, and 19th numbers.

For those interested in making a better tarot fortunetelling program, the sprite capability could be used to indicate the pictures of the 22 tarot cards.

[LIST OF VARIABLE]

C (I):	Tarot data numbers
R (I):	Shuffled numbers
Y (I):	Numbers to be shuffled
K:	Numbers pulled from the loop
Q:	Shuffle variable
S:	Shuffle variable
Z$:	INKEY Y$ variable
G:	Tarot data
U:	Numbers to be shuffled
D:	Shuffled numbers
A$:	Names of the tarot card data
U:	FOR-NEXT
I:	FOR-NEXT
J:	FOR-NEXT

[PROGRAM MAP]

10~20:	Initial setting
30:	Subroutine (background color)
40~110:	Shuffle
120~150:	Stop shuffle
160:	Subroutine (background color)
170~260:	Display of shuffled tarot cards
270~480:	Tarot card data
490~510:	Background color
520:	Data to be shuffled
530:	Tarot card numbers

PROGAM LIST

```
10 Print "■■■"
20 dim C(22),R(22),Y(7)
30 gosub 490
40 Print cursor(5,3);"****  TAROT ****";
50 restore 530
60 for I=1 to 22:read T:let C(I)=T:next
70 for I=1 to 7 step 2:let R(I)=1:let R(I+1)=2:next
80 for K=1 to 21
```

```
90 if rnd(1)=1 then let Q=C(K):let C(K)=C(K+1):let C(k+1)=0
100 for H=1 to 6
110 if rnd(1)=1 then let S=R(H):let R(H)=R(H+1):let R(H+1)=S
120 let Z$=inkey$
130 if Z$<>"" then goto 160
140 Print cursor(3,22);"SHUFFLE.  PRESS ANY KEY TO STOP";
150 next:next:goto 80
160 gosub 490
170 restore 520:for I=1 to 7:read G:let Y(I)=G:next:restore
270
180 for I=1 to 7
190 let U=Y(I)
200 let D=C(U)
210 for J=0 to D
220 read A$:next J:if R(I)=2 then Print cursor(4,I*3-1);"REV
ERSE"
230 Print cursor(10,I*3-1);A$;C(U)
240 Print cursor(0,I*3-1);I;
250 restore 270
260 next I:goto 80
270 data "THE FOOL"
280 data "THE MAGICIAN"
290 data "THE HIGH PRIESTESS"
300 data "THE EMPRESS"
310 data "THE EMPEROR"
320 data "THE HIEROPHANT"
330 data "THE LOVERS"
340 data "THE CHARIOT"
350 data "STRENGTH"
360 data "THE HERMIT"
370 data "WHELL OF FORTURN"
380 data "JUSTICE"
390 data "THE HANGED MAN"
400 data "DEATH"
410 data "TEMPERANCE"
420 data "THE DEVIL"
430 data "THE TOWER"
440 data "THE STAR"
450 data "THE MOON"
460 data "THE SUN"
470 data "JUDGEMENT"
480 data "THE WORLD"
490 cls 128:view 1,0,31,23
500 for I=4 to 6:stchr "3333333333333333" to 128,I:next
510 return
520 data 7,8,9,16,17,18,19
530 data 16,1,2,9,3,15,0,11,4,8,14,21,6,19,10,7,18,16,13,20,
5,12
```

RAISE YOUR GRADES SOFTWARE SERIES

THERE ARE NO STUDIES WITH WHICH THE M5 CAN'T HELP YOU. JUST KEY IN THE QUESTIONS AND ANSWERS AND THE M5 WILL BECOME YOUR TEACHER. YOUR GRADES WILL DEFINITELY IMPROVE. AND AS A BONUS, THE M5 WILL EVEN PROVIDE YOU WITH YOUR OWN SCHEDULE ORGANIZER AND SHOW YOU HOW TO MAKE A GRADE CHART.

MEMORIZATION IS EASY WITH PERSONAL COMPUTER TESTS
- Learning A Foreign Language

YOU CAN USE ARITHMETIC SYMBOLS
- Mathematical Exercises

PLAN YOUR DAY
- Schedule Organizer

LEARNING A FOREIGN LANGUAGE

Raise your grades Software

If you're studying a foreign language, here's something that'll really be useful. Using this program is like playing a game, and what's more, you'll be remembering those words and idioms with no effort at all. If you don't believe this, try for yourself!

Content:

First, enter the foreign words and their equivalents and then SAVE them on tape to make your vocabulary data. The computer randomly selects numbers from this data and uses it to give tests. These tests can either be from English to Japanese or Japanese to English or any other language.

After each test your score is displayed and if you wish, you can take another test. The order of the questions is changed each time.

Now why don't you try actually running the program.

Directions:

When you RUN the program the menu is displayed. Because the vocabulary (words and idioms) data must be entered, press key 1. When you do, you will be asked "How much DATA?" and you must then enter the number (up to 200). Next you will be asked "WORD?" and you will be expected to enter those words you wish to remember (up to 17 characters). After that you will be asked "ENGLISH?" and you should then enter the equivalents (up to 17 characters) of the previously entered WORD (A mode switching device is incorporated in this program so when entering the equivalents just put them in that mode). Repeat this procedure for all the data that needs to be entered.

After this input has been completed "TAPE RECORDER → RECORD" will be displayed. Put a blank tape in your tape recorder and after you have put it on RECORD, push the proper key. Your data will be recorded on the tape.

Next you will be asked "TEST?", and if you want to then take a test

push either key 3 or 4. If you don't want to take a test push key n and you'll be returned to the menu.

Try to add more data. After you have returned to the menu push key 2. This will again cause "TAPE RECORDER → RECORD" to appear. After you push the appropriate key, rewind the tape which the data has just been recorded onto and then play it. When you wish to add data it is first necessary to read the previously entered data (OLD).

When the OLD data has ended and when you enter new data you will be asked "How much DATA?" and you can follow the previous procedure.

When you enter additional data, the numbers of the data indicated above are all numbers so the amount of data entered into memory is indicated as "*** n+* ***".

All the data has been put into memory and it's almost time for a test. Return to the menu and push key 3 if you want to do a Foreign language to English or key 4 if you want to do English to a Foreign language.

First, the data needs to be read so you will be asked "Number of Questions?"; key in the number of questions you want on your test. It's impossible for more questions to be made than you have pieces of data stored in memory.

After you have entered the number of questions music will play and the questions will be displayed. If the answer you key in is correct you'll hear a chime; however, if your answer is incorrect, "WRONG!!!" will be displayed to the accompaniment of a buzzer and the correct response will be then be displayed.

After the test, your score out of a possible 100 points will be displayed. If you want to take another test, push either key 3 or 4. If you don't want another test, press key n. You will then be returned to the menu.

Since only 200 items can be entered, be certain not to exceed that amount. If you should put in more data than this the buzzer will sound and data entry will have to be repeated.

This program can be used for foreign idioms as well as foreign words. But in this case you must be careful to remember that the computer also reads the blanks. For example if "at once" is the answer to a question which appears and you enter "atonce", your answer will be considered to be a mistake.

With English translations when there is more than one possible translation, it goes without saying that only the equivalent you previously had entered will be considered to be correct. Also remember to be careful to distinguish between the upper and lower cases of the alphabet.

Now all that's left to do is to master the vocabulary and idioms through frequent tests.

Program:

This program is divided into four parts: menu, OLD data, SAVE data, and tests. One file can only hold a maximum of 200 pieces of data in memory. You can increase the amount of your data by increasing the number of your data files.

This program can also be used to study historical dates by replacing the "WORD" with years and the "ENGLISH" data with what occurred at that time. In that case "WORD" should be changed to "Year" and "ENGLISH" to "Event" in line numbers 40, 110, 150, 180, 240, and 550.

[LIST OF VARIABLE]

A$:	Menu numbers
N:	Amount of data in memory
NN:	Amount of data to be memorized
E$:	Word
Y$:	English
QN:	Number of questions
S1, S2:	Variables to produce sound
Q:	Numbers of data selected for questions
C$:	Variable used to alternate between word and English
AN$:	Answers
P:	Number of correct responses

[PROGRAM MAP]

10:	Initial setting
20~60:	Menu
70~110:	OLD data
120~240:	SAVE data
250~570:	Tests
580:	Music data

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23
20 cls:Print cursor(8,4);"*** MENU ***"
30 Print cursor(5,7);"1 INPUT DATA";cursor(5,9);"2 ADD DATA"
40 Print cursor(5,11);"3 TEST ( WORD -> ENGLISH )";cursor(5,
13);"4 TEST ( ENGLISH -> WORD )"
50 Print cursor(5,16);"WHICH ONE?"
60 let A$=inkey$:if A$<>"1" and A$<>"2" and A$<>"3" and A$<>
"4" then goto 60
70 if A$="1" then goto 120
80 cls:Print cursor(1,10);"HIT ANY KEY !"
90 Print cursor(1,12);"and TAPE RECORDER -> PLAY"
100 if inkey$="" then goto 100
110 old "WORD":if A$<>"2" then goto 250 else let N=vPeek(0)
120 cls:Print cursor(0,10);tab(30)
130 Print cursor(1,10);"HOW MUCH DATA ";:inPut NN
140 if N+NN>200 or NN<1 then Print "■":goto 120
150 for I=1 to NN:cls:Print cursor(10,4);"***";N+I;"***";cur
sor(8,6);"< MAX 17 CHARACTERS >";cursor(1,11);"WORD  ";:inPu
t E$
```

```
160 for L=1 to 18-len(E$):let E$=E$+" ":next
170 for J=1 to 18:vPoke(N+I-1)*40+J,ascii(mid$(E$,J,1)):next
180 Poke &701A,5 or Peek(&701A)and &F8:Print cursor(1,14);"E
NGLISH"::inPut Y$:Poke &701A,4 or Peek(&701A)and &F8
190 for L=1 to 18-len(Y$):let Y$=Y$+" ":next
200 for J=1 to 18:vPoke(N+I-1)*40+20+J,ascii(mid$(Y$,J,1)):n
ext:next:vPoke 0,N+NN
210 cls:Print cursor(1,10);"TAPE RECORDER -> RECORD"
220 Print cursor(1,12);"and HIT ANY KEY !"
230 if inkey$="" then 9oto 230
240 save "WORD",0,(N+NN)*40,1:cls:9oto 550
250 cls:Print cursor(1,10);"NUMBER OF QUESTION ";:inPut QN
260 if QN>vPeek(0)or QN<1 then Print "■":9oto 250
270 cls:Print cursor(1,10);"QUESTION START !!!":let P=0:rest
ore 580
280 read S1,S2:if S1=0 then 9oto 300
290 out &20,&80+S2:out &20,S1:out &20,&93:for I=1 to 1000:ne
xt:out &20,&9F:9oto 280
300 if A$="3" then Poke &701A,5 or Peek(&701A)and &F8
310 randomize:for I=1 to QN:cls
320 Print cursor(5,5);"*** QUESTION";I;"***"
330 if I=vPeek(0)then let Q=1:9oto 350
340 let Q=rnd(vPeek(0)-1-(I-1))+1
350 let E$="":let Y$="":for J=1 to 18:let E$=E$+chr$(vPeek(4
0*(Q-1)+J)):next
360 for J=1 to 18:let Y$=Y$+chr$(vPeek(40*(Q-1)+20+J)):next
370 if A$="4" then let C$=E$:let E$=Y$:let Y$=C$
380 Print cursor(1,10);"QUESTION";E$
390 Print cursor(1,12);"ANSWER";:inPut AN$
400 for J=1 to 18-len(AN$):let AN$=AN$+" ":next
410 cls:if Y$<>AN$ then 9oto 440
420 out &20,&8F:out &20,&07:out &20,&93:for L=1 to 1000:next
:out &20,&9F
430 for M=3 to 10:out &20,&80:out &20,&0A:out &20,&90+M:for
L=1 to 250:next:next:out &20,&9F:let P=P+1:9oto 470
440 Print "■";cursor(1,10);"WRONG !!!":Print cursor(1,12);"C
ORRECT";Y$
450 Print cursor(1,15);"HIT ANY KEY !"
460 if inkey$="" then 9oto 460
470 if A$="4" then let C$=E$:let E$=Y$:let Y$=C$
480 for J=1 to 18
490 vPoke 40*(Q-1)+J,vPeek(40*(vPeek(0)-1-(I-1))+J)
500 vPoke 40*(vPeek(0)-1-(I-1))+J,ascii(mid$(E$,J,1))
510 vPoke 40*(Q-1)+20+J,vPeek(40*(vPeek(0)-1-(I-1))+20+J)
520 vPoke 40*(vPeek(0)-1-(I-1))+20+J,ascii(mid$(Y$,J,1)):nex
t:next
530 Poke &701A,4 or Peek(&701A)and &F8
540 cls:Print cursor(1,10);"TOTAL POINTS";P*100/QN;"POINTS"
550 Print cursor(1,14);"TEST ? n..NO , 3..WORD->ENGLISH";cur
sor(16,16);"4..ENGLISH->WORD"
560 let A$=inkey$:if A$<>"n" and A$<>"3" and A$<>"4" then 9o
to 560
570 if A$="n" then 9oto 20 else 9oto 250:end
580 data &d,&6,&d,&6,&8,&f,&8,&f,&7,&f,&7,&f,&8,&f,0,0
```

MATHEMATICAL EXERCISES

Arithmetic calculation is easy on computers, but because most of them do not have special symbols for this, input of numerical formula can be rather troublesome. But with the M5 on GII mode, numerical formula can be easily entered.

If you practice this program enough times, math tests will be easy to pass.

Content:

After you have entered the mathematical problems and their answers as data, the computer randomly selects and puts them out as test questions.

If you save the data on cassette tape it can be used as often as you like and you can also add to it later.

After each test your score is displayed and you can immediately take a new test. (Questions being randomly selected by the computer.)

Because the arithmetic notation is set in this program, the entry of mathematical formula is made easier. Take full advantage of it.

Directions:

When you RUN the program the menu is displayed. When you are entering data push key 1, when you are adding additional data push key 2, and when you want to take a test push key 3.

Pushing key 1 allows you to enter data. First you will be asked "Data Amount?" and you then should key in the number of items of data you intend to enter (up to 50 items). Then you will be asked "Question?". You must then enter the questions (mathematical formula). That is followed by the prompt "Answers?" and the answers to your problems must then be entered. Be careful to remember that you may use only up to 17 characters for either the question or the answer.

Repeat the above procedure for the amount of your set data.

If you switch to graphic mode the following changes are made:

P → π	R → $\sqrt{\ }$	I → ∫
M → μ	M+SHIFT → ∞	
S → σ	S+SHIFT → Σ.	

After all of your data has been entered, the instruction "TAPE RECORDER → RECORD" will be indicated. Load your tape recorder with a blank tape and after it has been set to record, push any key. This will save your data on tape.

You will then be asked "TEST?" and if you wish to take a test at that time press the Y key; if you don't, press the N key. Pressing the N key returns you to the menu.

When you wish to add more data, press key 2 when the menu is being shown. Pushing key 2 will cause "TAPE RECORDER → RECORD" to be indicated. After you push any key, set the tape which has your data recorded on it and replay the tape on your tape recorder. Before you can add new data it is necessary to first read the data (to OLD) which has already been recorded. After the data has been OLDed, the indication "Data Amount?" will appear on the screen just as it did when you entered data the first time. This time enter the number for the additional data you wish to enter. Next add the new data just as you did before. The difference is that numbers of the data entered at the top of the screen become through numbers.

The amount of data you can add is the difference between 50 and the amount already entered.

Pushing key 3 of the menu allows you to take a test. When you push key 3 and after the data has been OLD, you will be asked "How many questions?" and you should then enter the number of test questions you want. You of course cannot take a test with more questions than you have stored in data, and if you should enter a larger number, the buzzer will sound and you will have to put in a new entry.

After you have entered the number of questions you want on your test, the indication "QUESTION START!!!" will appear and music will be played. After the music stops the question number and the question will appear. Enter the answer to the question. If your answer is correct, the chime will ring and the next question will appear. If your answer is incorrect, the buzzer will sound and "WRONG!!!" will appear followed by the correct answer. Push any key and the next question will appear.

After you have answered all the questions of the test, your score will be shown. Below that "TEST?" will be indicated. If you want to take another test push Y, if not and you wish to stop, push N. The N key returns you to the menu.

Program:

This program was made in the GII mode so it could use arithmetic symbols. For that reason, only 50 items can be stored, which is somewhat less than that for vocabulary exercises and historical dates exercises.

The program is divided into 5 parts: designated characters for arithmetic notation, menu, data OLD, data SAVE, and tests.

Although only 50 items can be SAVED in a file, by using several different file names, much more data can be saved. For example, it's good idea if you store Section 1 of your memorized data as S1 and Section 2 of the same as S2. Change data after "STCHR" in lines 20—80 using the data for your own arithmetic symbols.

[LIST OF VARIABLE]

A$:	Menu numbers
N:	Amount of memorized data
NN:	Amount of data to be memorized
E$:	Questions
Y$:	Correct response
QN:	Number of questions
S1, S2:	Sound producing variable
Q	Numbers of data selected for questions
AN$:	Answers
P:	Number of correct responses

[PROGRAM MAP]

10:	Initial setting
20~80:	Arithmetic symbol character setting
90~130:	Menu
140~180:	Data OLD
190~310:	Data SAVE
320~600:	Tests
610:	Music data

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23
20 for I=1 to 3:stchr "00007c2828284c00" to 131,I
30 stchr "1c10101050301000" to 135,I
40 stchr "0000484874404000" to 153,I
50 stchr "0004385050502000" to 139,I
60 stchr "7c44201020447c00" to 235,I
70 stchr "1810101010103000" to 147,I
80 stchr "0000006c92926c00" to 251,I:next
90 cls:Print cursor(8,4);"*** MENU ***"
100 Print cursor(5,7);"1 INPUT DATA";cursor(5,9);"2 ADD DATA
"
110 Print cursor(5,11);"TEST 3" READYRINT cursor(5,14);"WHIC
H DO YOU CHOOSE?"
120 Print cursor(5,14);"WHICH DO YOU CHOOSE?"
130 let A$=inkey$:if A$<>"1" and A$<>"2" and A$<>"3" then 9o
to 175
```

```
140 if A$="1" then goto 255
150 cls:Print cursor(1,10);"HIT ANY KEY !"
160 Print cursor(1,12);"and TAPE RECORDER -> PLAY"
170 if inkey$="" then goto 220
180 old "MATH":if A$<>"2" then goto 412 else let N=vPeek(0)
190 cls:Print cursor(0,10);tab(30)
200 Print cursor(1,10);"DATA ITEMS ";:inPut NN
210 if N+NN>50 or NN<1 then Print "■":goto 255
220 for I=1 to NN:cls:Print cursor(10,4);"***";N+I;"***";cur
sor(8,6);"< UP TO 17 CHARACTERS >";cursor(1,11);"QUESTION";
225 inPut E$
230 for L=1 to 18-len(E$):let E$=E$+" ":next
240 for J=1 to 18:vPoke &1800+(N+I-1)*40+J,ascii(mid$(E$,J,1
)):next
250 Print cursor(1,14);"ANSWER   ";:inPut Y$
260 for L=1 to 18-len(Y$):let Y$=Y$+" ":next
270 for J=1 to 18:vPoke &1800+(N+I-1)*40+20+J,ascii(mid$(Y$,
J,1)):next:next:vPoke &1800,N+NN
280 cls:Print cursor(1,10);"TAPE RECORDER -> RECORD"
290 Print cursor(1,12);"and HIT ANY KEY !"
300 if inkey$="" then goto 400
310 save "MATH",&1800,&1800+(N+NN)*40,1:cls:goto 750
320 cls:Print cursor(1,10);"NUMBER OF QUESTIONS ";:inPut QN
330 if QN>vPeek(&1800)or QN<1 then Print "■":goto 412
340 cls:Print cursor(1,10);"QUESTION START !!!":let P=0:rest
ore 1000
350 read S1,S2:if S1=0 then goto 550
360 out &20,&80+S2:out &20,S1:out &20,&93:for I=1 to 1000:ne
xt:out &20,&9F:goto 450
370 randomize:for I=1 to QN:cls
380 Print cursor(5,5);"*** QUESTION";I;"***"
390 if I=vPeek(&1800)then let Q=1:goto 590
400 let Q=rnd(vPeek(&1800)-1-(I-1))+1
410 let E$="":let Y$="":for J=1 to 18:let E$=E$+chr$(vPeek(&
1800+40*(Q-1)+J)):next
420 for J=1 to 18:let Y$=Y$+chr$(vPeek(&1800+40*(Q-1)+20+J))
:next LISTRINT cursor(1,10);"QUESTION":E$
430 Print cursor(1,10);"QUESTION";E$
440 Print cursor(1,12);"ANSWER";:inPut AN$
450 for J=1 to 18-len(AN$):let AN$=AN$+" ":next
460 cls:if Y$<>AN$ then goto 670
470 out &20,&8F:out &20,&07:out &20,&93:for L=1 to 1000:next
:out &20,&9F
480 for M=3 to 10:out &20,&80:out &20,&0A:out &20,&90+M:for
L=1 to 250:next:next:out &20,&9F:let P=P+1:goto 700
490 Print "■";cursor(1,10);"WRONG !!!":Print cursor(1,12);"C
ORRECT";Y$
500 Print cursor(1,15);"HIT ANY KEY !"
510 if inkey$="" then goto 690
520 for J=1 to 18
530 vPoke &1800+40*(Q-1)+J,vPeek(&1800+40*(vPeek(&1800)-1-(I
-1))+J)
540 vPoke &1800+40*(vPeek(&1800)-1-(I-1))+J,ascii(mid$(E$,J,
1))
550 vPoke &1800+40*(Q-1)+20+J,vPeek(&1800+40*(vPeek(&1800)-1
-(I-1))+20+J)
560 vPoke &1800+40*(vPeek(&1800)-1-(I-1))+20+J,ascii(mid$(Y$
,J,1)):next:next
570 cls:Print cursor(1,10);"POINT";P*100/QN;"POINTS"
580 Print cursor(1,14);"TEST ? (y/n)"
590 let A$=inkey$:if A$<>"y" and A$<>"n" then goto 765
600 if A$="n" then goto 110 else goto 412:end
610 data &d,&6,&d,&6,&8,&f,&8,&f,&7,&f,&7,&f,&8,&f,0,0
```

SCHEDULE ORGANIZER

How many hours a day do you sleep? What!? 10 hours!? Don't you think that is a bit much?

You study 30 minutes and watch TV for 3 hours!?... Sounds like you need to reorganize your daily schedule.

And that's how this program got its start. It gives you a clear picture of your schedule for one day. You have to execute it. You have to decide yourself whether to stick to it or not!!

Content:

Enter your schedule for a day and this program will divide it into belt-shaped colored sections for easy reading at a glance.

And regarding certain things you have to do, it will also show you plans which require several hours in a day. Put in the current time (or the time you wish to know the schedule for) and the program will tell you what you should be doing at that time.

Directions:

Try to design a schedule like the one shown in Diagram 1.

RUN the program and you will be asked, "Number of items (up to 9)?". You should then enter the number of things you have to do. For now, enter 7 and call them "Sleep", "Breakfast", " School", "Free", "Dinner", "Study", and "TV". If you enter a number less than 1 or greater than 9 the buzzer will sound and you will have to re-enter your information.

Next a colored box will appear and you should enter the names of the previously mentioned schedule items one at a time. The names of these can only have up to 7 characters in them. If you put in more than 7, the buzzer will sound and you will have to do it again.

After you have entered these items the color and the graph for these items will be set up in the top half of the screen. Below that "00:00 — 07:00?" will be indicated and you should select from the above what it is you must do then and enter the number shown to the left of the color. If you

enter a number that is not shown above, the buzzer will sound and you will have to repeat this procedure.

TIME	ITEMS
24 TO 7	SLEEP
7 TO 8	BREAKFAST
8 TO 16	SCHOOL
16 TO 18	FREE
18 TO 19	DINNER
19 TO 21	STUDY
21 TO 22	T.V.
22 TO 24	STUDY

In our case we want to enter 1 for "Sleep". Enter the other items in the same way for the rest of the 24-hour period. In this program the afternoon hours are referred to as 13:00 to 23:00. Midnight is indicated as 00:00.

After you have entered all that you must do for a 24-hour period, your schedule will be shown on the lower half of the screen. If you wish to end at this point press the E key.

If you press the F key instead of the E key you will again be asked "Number of items?" and if you enter the number of that item, the number of hours in one day which it requires will be indicated. For example, if you wish to know how many hours you have allotted to sleep, enter the number for sleep, 1, and "Sleep 7 hours" will be shown.

If you should push the N key, you will be asked "TIME NOW?". When you enter the time, you will be told what you have scheduled for that time. For example, if you enter "20", "It's time to study" will be indicated.

If the present time is 20:30, drop the minutes and only enter the hour "20" and everything between 20:00 and 21:00 will be shown.

Pushing any key will return you to the beginning.

If you push the C key, you can change your schedule. First you will be asked "TIME TO CHANGE?". Put in the beginning hour of the time period you wish to change. For example, you want to change the period from 19:00 to 20:00 in your schedule, type in "19".

After you have finished making changes, you will again be asked "Item?" and you should then enter in the number of the new item you wish to do at that time. If you should again enter a number that is not indicated in the list at the top of the screen, the buzzer will sound and you will have to repeat the procedure. This is true also if you enter a number less than 0 and greater than 23.

After you have finished entering the numbers of the items, the bottom half of the screen will again change to shown you your changed schedule.

See if you can do this yourself and change the schedule so that the period from 22:00 to 23:00 becomes free.

First press the C key, then enter "22" for the time you wish to have changed and enter "4" which is the number above listed for "Free". Your schedule will then be changed to read "Free" for that period from 22:00 to 23:00 rather than "Study".

This function can be repeated as often as you like and this will make such things as deciding your schedule changes while looking at the time easy.

Program:

This program was made in the GII mode and is divided into three major parts: entries, setting up the schedule form, and schedule changes.

Because the keying in is a little troublesome, you can improve it so that it will indicate color, too. In this way if you want to select a color, you can use 9 different colors when you indicate "9" for the number of items. And as for colors which are not used in the entry of the item names, if you enter any letter, you can enter the item in any color you like.

[LIST OF VARIABLE]

N:	Number of items
M$ (arrangement):	Item name
S (arrangement):	Item number
A$:	Character entries for divisions
K:	Number of item for time to be checked
F:	Time
H:	Present time
C:	Time to be changed
CK:	Item to be changed

[PROGRAM MAP]

10:	Initial setting
20~100:	Entries
110~120:	Setting up schedule diagram
130~310:	Changes

PROGRAM LIST

```
10 Print "■■■":view 1,0,30,23
20 cls:Print cursor(1,10):"number of items (uP to 9) ";:inPu
t N:if N<1 or N>9 then Print "■":goto 20
30 dim M$(N):cls:for I=1 to N
40 for J=4 to 6:stchr rPt$(16,right$(hex$(I+I),1))to 240+I,J
:next
50 Print cursor(1,2*I):tab(30)
60 Print cursor(3,2*I):rPt$(3,chr$(240+I));" -- ";:inPut M$(
I):if len(M$(I))>7 then Print "■":goto 50 else next
70 cls:let X=1:let Y=0:for I=1 to N:if I>5 then let X=16:let
 Y=5
```

```
80 Print cursor(X,(I-Y)*2):num$(I):rPt$(2,chr$(240+I));" - "
:M$(I):next
90 view 1,12,30,23:dim S(23):for I=0 to 23
100 cls:Print cursor(2,5):I:"time -":I+1:"time ":inPut S(I)
:if S(I)<1 or S(I)>N then Print "■":9oto 100 else next
110 cls:for I=0 to 4:Print cursor(1+I*6,2):I*6:next
120 for I=0 to 23:for J=1 to 6:Print cursor(3+I,2+J)CHR$(240
+S(I)):next:next
130 Print cursor(0,10):tab(30):cursor(1,10):"f..time n..now
c..chan9e e..end"
140 let A$=inkey$:if A$<>"f" and A$<>"n" and A$<>"c" and A$<
>"e" then 9oto 140
150 if A$="e" then end
160 if A$<>"f" then 9oto 220
170 Print cursor(0,10):tab(30):cursor(1,10):"items":inPut K
:if K<1 or K>N then Print "■":9oto 170
180 let F=0:for I=0 to 23:if S(I)=K then let F=F+1
190 next
200 Print cursor(0,10):tab(30):cursor(1,10):M$(K),F:"time"
210 if inkey$="" then 9oto 210 else 9oto 130
220 if A$<>"n" then 9oto 260
230 Print cursor(0,10):tab(30):cursor(1,10):"time now ":inP
ut H:if H<0 or H>23 then Print "■":9oto 230
240 Print cursor(0,10):tab(30):cursor(1,10):M$(S(H));"time"
250 if inkey$="" then 9oto 250 else 9oto 130
260 Print cursor(0,10):tab(30):cursor(1,10):"time to chan9e"
:inPut C
270 if C<0 or C>23 then Print "■":then 9oto 260
280 Print cursor(0,10):tab(30):cursor(1,10):"items":inPut C
K
290 if CK<1 or CK>N then Print "■":9oto 280
300 let S(C)=CK:9oto 110
310 end
```

PERSONAL COMPUTER TERMINOLOGY(1)

ADDRESS

This is divided into various kinds such as absolute address, relative address, key address, etc. This indicates the place where information is stored in memory or the place of a forwarding address.

ALL-IN-ONE TYPE

A computer in which all the devices necessary to the personal computer (display, external memory device, CPU, and keyboard, etc.) have been built as a single unit.

ASCII CODE

This is the abbreviation for the American Standard Code for Information Interchange.

CHARACTER DISPLAY DEVICE

Among CRT displays, this one is especially used for character reproduction. A 9 to 15-inch screen can display anywhere from 250 to 2000 characters.

CAD/CAM

(Computer Aided Design/Computer Aided Manufacture)

Designed to increase the level of efficiency in design and manufacture by the use of computer; used in high level calculations and diagram production.

Today almost all design of LSI and airplanes are done this way. The development of this is closely connected to computer graphics.

COMMAND

What the human tells the computer to do. For example, list the program or read the program and shortened commands. It is also used for instructions and commands from the central processing unit in regard to input and output devices.

DISAPPEARING LINE ELIMINATION

In three-dimensional graphics, this prevents lines from appearing in solid forms which intrinsically are not visible when running in the opposite direction. This is a particularly important function for displaying solid forms in vector scan graphics.

ERROR MESSAGE

This message is sent from the computer when it cannot correctly deal with data either due to a program or operational error.

HIDDEN FRAME ELIMINATION

This function prevents the backside of solid objects from appearing in such forms as computer animation. Although at a glance it appears simple, it requires a very complex procedure.

IC (Integrated Circuit)

A circuit composed of several hundred semiconductor elements integrated on the surface of a small silicon chip.

IMAGE PROCESSING

Information entered and displayed on the screen which is handled by the computer as digital signals and then transformed into a new data image or else discernable data.

INTERFACE

An exchange of data signals occurs between devices and this circuit controls the electric signals of each device.

LSI

This stands for Large Scale Integration. Among circuits such as diode, transistor, condensor, and resistors, which are integrated on silicon surfaces, integration performed on a large scale (1000 — 10,000 elements) is called LSI. And on a scale larger than that it is called VLSI.

RESOLUTION

This scale measures to what degree of precision the lines of the picture image are defined and reproduced.

RGB

Ths system by which the screen image is made from the three colors red, green, and blue.

RS-232C

This is a standard port for serial transmission established by the American Electronics Industry. It is used by almost every model of personal computer.

SHADOW DISPLAY

Depending on the shadowing given to figures, a sense of form or being is produced. This is one important screen image process together with the use of disappearing lines.

THREE-DIMENSIONAL GRAPHICS

This allows display of three-dimensional objects on a two-dimensional screen.

The object is displayed in lines and in faded form. It is important that the point from where the object is viewed by movable.

It is mainly used in simulation, animation, and CAD.

WINDOW

This function determines if the display areas of the CRT screen are proportionately distributed when graphics are displayed. A section of the large imaginary coordinate system appears as through it is viewed through a window on the CRT, thus window.

XY PLOTTER

Figure drawing device. If you move the pen in the XY direction or in the Y direction, depending on if you move the paper in the X direction, you can

freely draw lines.

The pen uses ball point pens, etc., and many use a variety of coloring devices.

HOUSEHOLD SOFTWARE SERIES

THE M5 IS ALSO A RELIABLE FRIEND IN THE HOUSE. HOUSEHOLD EXPENSES AND NUTRITIONAL PLANNING CAN BE MANAGED, AND IT IS ALSO HELPFUL IN DIET PLANNING. IF THE DATA FOR EACH IS SAVED, IT BECOMES A USEFUL FAMILY RECORD.

RESOURCEFULNESS MAKES A HAPPY HOME
- Household Budget Program

NUTRITIONAL MANAGEMENT FOR THE FAMILY
- Calorie Calculator

WEIGHT CHANGES DISPLAYED GRAPHICALLY
- Diet Planner

HOUSEHOLD BUDGET PROGRAM

Prices continuously are rising but incomes never do. This is a major headache for all. But those who are "with it" are different. They use the M5 in managing their family budgets.

Content:

Divide expenses into 11 categories and enter them as Food Expenses, Rent, Clothing, Utilities, School Expenses, Medical, Entertainment, Leisure, News, Allowances, Public. When income is then entered, the remainder or deficit from the above items will be shown. It will also calculate total income and total expenses to date.

The data for income, expenses, and balance can be saved on cassette tape. By entering new data everyday and reading it and also by preserving it on tape, your cassette tape becomes an indispensable record of household expenses.

Directions:

Run the program and the menu will be displayed. Select 2, "Today's Input", for the first day.

When you select 2, you will be asked to enter data for each of the 11 items, "Food", "Housing", "Expenses", "Balance", and "Today's Date", etc. First, enter the date, year, month, day. After you have done so you will be asked "Is this correct? (y/n)" and if it is, hit the Y key, if not, hit the N key. If you hit the N key you then have to put in the data for the date one more time.

If you push the Y key, you then will need to enter the amount spent on each of the items (up to seven columns), and after they all have been entered (including "Income"), "Expenses" and "Balance" will be displayed. At the same time "Hit Any Key!!" will also be shown; doing so returns you to the menu.

In the event that you mistakenly enter the number for "Today's Input", push CTRL + DEL before you hit the RETURN key. However, if you have

already hit the RETURN key before you realize your mistake, even though it is troublesome, enter all of your data to the end and after you have returned to the menu, select 2 again, enter the same date, and enter all of the data correctly one more time. If you should mistakenly enter the date, all the data entered will be for that day so be very careful when entering the date.

When you select 3, "Total", at the menu, the names of the various items will be shown, and if values have been entered for them that day, the total up to that day for all the items will be totalled and displayed. If the values for that day have not been entered, then the total up through the previous day will be calculated and shown. The balance is not totalled, though. After it has all been shown, "Hit Any Key!!" will be shown and if you hit a key, you will return to the menu.

If you select 4, "End", at the menu, you will be asked "Is the cassette tape ready?" and the computer will wait for your response. At this stage, set your cassette recorder to the record position. Then, hit any key and the RETURN key and your data will be recorded. After it has been recorded, the program will end and "Ready" will appear on the screen.

For a day other than the first day, first select 1, "Data Input".

When you enter 1, "CASSETTE READY?" will be displayed and will wait for your answer. At this point, load the tape with the previous day's data recorded on it, rewind it to the beginning, and set your cassette recorder in the PLAY position. Then hit any key and the RETURN key and your data will automatically be entered.

When you are entering data from cassette, verify whether or not "Household Data" is indicated on the screen.

If, after a little while it isn't indicated, rewind your tape some and play it back again.

After your data has been entered from the tape, the menu will automatically appear. Repeat the same procedure you used for the first day executing steps 2 — 4 of the menu in their proper order.

If you stop the program while it is running by pressing SHIFT+RESET, data entered after typing the RETURN key will be verified. However, if you wish to save it on tape, execute RUN 10. The menu will appear again. This time, only execute RUN. If you change modes, be careful because the data that had already been stored will be erased. If the data is erased, you must then enter all of it from the beginning.

Program:

There really aren't very many steps, and there is very little left over memory. Enter your data in the GI mode writing it a column at a time in the middle of a blank VRAM &0000—&0BFF. Calculations are also produced 1 column at a time. If this is not done, financial calculations with BASIC-I would probably be impossible.

[LIST OF VARIABLE]

A$: For input, for including data
B: Menu number
C: Year
C1: Month
D: Day
E$
F$: Calculation supplement
H
H1
I
J: FOR-NEXT
K$
L$
M$: Calculation supplement
M1$
P$
Q: FOR-NEXT
S$: Calculation supplement
S1
S2: Output supplement
W$: Calculation supplement

[PROGRAM MAP]

10: Memory clear
20~80: MENU
90~110: Output for various items
120~280: "Today's Input"
290~390: "Total"
400~410: "Data Input" and "END"
420~550: Calculation of balance
560~620: Output subroutine

PROGRAM LIST

```
10 for I=0 to &0BFF:vPoke I,0:next
20 view:Print "■■■■":view 8,2,28,23
30 else Print "■"
40 Print "*** MENU ***":Print "1.INPUT DATA":Print "2.INPUT
DAY"
50 Print "3.TOTAL":Print "4.END"
```

```
60 Print:Print "PICK ONE";
70 let A$=inkey$:if A$<"1" or A$>"4" then goto 70
80 let B=val(A$):if B=4 or B=1 then goto 400
90 cls:view 5,2,30,23:restore:for I=1 to 16:read A$:Print A$
:next:Print
100 data "** BUDGET **","","FOOD EXPENCE","RENT","CLOTHING",
"UTILITY","EDUCATION","MEDICAL"
110 data "ENTERTAINMENT","LEISURE","PHONE BILL","ALLOWANCES"
,"DUES","INCOME","TOTAL EXPENCE","ENTERTAINMENT"
120 if B=3 then goto 290
130 view 3,19,30,20:Print "■";:input "TODAYS DATA",C,C1,D:Pr
int "YEAR";C;" MON.";C1;" DAY";D
140 if D>31 or D<1 then goto 130
150 Print "IS THIS CORRECT?";
160 let A$=inkey$:if A$="n" then Print "■":goto 130
170 if A$<>"y" then goto 160 else Print "■"
180 for I=0 to 11:view 18,4+I,30,4+I:let H=0:let L$="":let M
$=""
190 Print:input " ";E$:Print E$;
200 if len(E$)>7 then goto 190
210 let E$=right$(rPt$(7,"0")+E$,7):for J=1 to 7:let F$=mid$
(E$,J,1)
220 if F$<"0" or F$>"9" then goto 190
230 vPoke 91*D+I*7+J-1,val(F$):next:next:view 1,0,30,23
240 for J=6 to 0 steP-1:for I=0 to 10
250 let H=vPeek(91*D+I*7+J)+H:next:let K$=num$(H):vPoke 91*D
+12*7+J,H mod 10
260 let H=H/10:let L$=right$(K$,1)+L$:next:Print cursor(17,1
6);:let S$=L$:gosub 560
270 gosub 420:Print cursor(17,17);:let S$=P$:gosub 560
280 Print cursor(3,21);"Hit Any Key !!":goto 390
290 for I=0 to 10:let H=0:let M$=""
300 for J=6 to 0 steP-1:for Q=1 to 31
310 let H=H+vPeek(91*Q+7*I+J)
320 next:let M$=num$(H mod 10)+M$:let H=H/10:next
330 Print cursor(13,2+I);:let S$=M$:gosub 560
340 next:gosub 420
350 Print cursor(13,13);:let S$=M$:gosub 560
360 Print cursor(13,14);:let S$=L$:gosub 560
370 Print cursor(13,15);:let S$=P$:gosub 560
380 Print cursor(0,17);"Hit Any Key !!"
390 if inkey$<>"" then goto 20 else goto 390
400 cls:Print "CASSETTE READY?":input A$
410 if B=4 then save "HOUSE HOLD DATA",&00,&0BFF,1:end else
old "HOUSE HOLD DATA":goto 20
420 let H=0:let H1=0:let M$="":let L$="":let P$=""
430 for J=6 to 0 steP-1:for I=1 to 31
440 let H=H+vPeek(I*91+77+J)
450 let H1=H1+vPeek(I*91+84+J):next
460 let M$=num$(H mod 10)+M$:let H=H/10:let L$=num$(H1 mod 1
0)+L$:let H1=H1/10
470 next:let W$=L$:let R$=M$
480 if M$<L$ then let W$=M$:let R$=L$:let M1$="-" else let M
1$=""
490 let H=0:let H1=0
500 for I=7 to 1 steP-1
510 let H=val(mid$(R$,I,1))-val(mid$(W$,I,1))-H1
520 if H<0 then let H=H+10:let H1=1 else let H1=0
530 let P$=num$(H)+P$:next
540 if M1$="-" then let P$=M1$+right$(P$,6)
550 return
560rem PRINT
570 let S1=val(left$(S$,4)):let S2=val(right$(S$,3))
580 if S1=0 and S2=0 then Print "0":return
```

```
590 if S1<>0 and S2=0 then Print num$(S1);right$(S$,3):retur
n
600 if S1=0 and S2<>0 then Print num$(S2):return
610 Print num$(S1);num$(S2)
620 return
```

CALORIE CALCULATOR

Nutritional management for the family

Is Pop getting a pot belly? Better cut back on fats..., Growing Bobby needs more protein and calcium..., Grandma needs low-protein and low-calorie food... These are some of the many problems facing homemakers everywhere.

Now all of you out there who have been worrying about these things can relax. Here's a strong friend you can rely on to solve all of your problems.

And of course, it's very effective for dieting.

Content:

For data, enter the names of food and their calorie amounts per 100 grams, yours (and your family's) sex, weight, age, and how hard each has to work. When you enter the amount of food consumed for a certain day, this program will display and compare the amount of calories consumed and the amount needed (the amount of calories that person should have).

This program will only work for those people between the ages of 6 and 89 and with weights under 100 kilograms.

Directions:

First enter program "VPOKE1" exactly as it is listed, and RUN it. There is no need to SAVE this program.

Next, enter program "VPOKE2" exactly as it is listed and if you RUN it after you have SAVEd it, you will be asked if you are going to enter data for the first time or are adding more data.

If you are adding data for the first time, you will first be asked the names of the different kind of food. Enter the kinds of food you regularly eat; you may use between 2 and 10 characters for each. You will next be asked the calories for each type of food you entered, so enter their caloric values per 100 grams. This can be found by referring to "Lists of Food" put out by various publishing companies.

It is possible to enter up to 999 cal., however decimals cannot be used. Also, if you make a mistake in entering the data, you must re-enter it.

When you are finished entering the data, enter an asterisk (*).

If you wish to make alteratioins on the data entered, hit the Y key, if not, hit the N key. If you enter Y, you will again be asked for the names of the food and the calories. The N key will cause the next food item and its calories to be displayed.

After you have finished correcting your data, SAVE it. Set your cassette recorder to the record position, then enter the RETURN key. At this time the file name is "DATA". After the data has been SAVEd and verified, execute RUN 400.

You next must enter the program list for "Calories Calculation". SAVE, and RUN it.

First you enter a name, and then a "1" or a "2" for sex. After that you will be asked for the weight. Do not enter a figure more than 99 kg. Then enter the age between 6 and 89. Then you will be asked how hard the person works. Enter a number from 1 to 4.

"Hit Return Key" will next be displayed. After you push the RETURN key, enter the food. Enter the name of the food used in the program and the intake measured in grams (keep this under 1 kg.). If you enter the name of a food that has not been entered into your data, the buzzer will sound and you will have to re-enter an appropriate name. After you have entered the nemes, enter an asterisk (*). The amount needed and the amount consumed of the person will then be displayed.

If you wish to stop data entry at this point press the E key. If you wish to enter more data, press the M key, and if you wish to start from the beginning again, the N key.

This is the procedure for running this program for the first time. After you have gone to GI mode by pressing CTRL + the S key, OLD the "DATA" and "Calorie Calculation" program. The following procedure is the same as that for the first time.

When you wish to increase your data, first OLD the "VPOKE2" program, RUN it, select "2", prepare the "DATA" cassette tape in your tape recorder, and after you have put it in the record position, hit the RETURN key to OLD the data. After the data has been OLDed, you can add food data because the main points are the same as for the first time.

Program:

"VPOKE1" is a program that has basic metabolism rates, according to age and sex, entered on VRAM. Because these standard values are in decimals, they have been divided into double integers.

"VPOKE2" is a program with food data entered on VRAM.

"Calorie Calculator" is a program which uses the data of the above two and calculates calories. Because you cannot use decimal figures ıs necessary because of the necessity of decimals in the calculation of calories.

[LIST OF VARIABLE]

NA$:	Name
SE:	Sex
KG:	Weight
YE:	Age
M:	VRAM address for standard metabolism rates
NO1, NO0:	Standard metabolism basic values
WO:	Intensity of work
WO1, WO0:	Fixed number for work differentials
EA1:	Integer part of calorie intake needed
EA0:	Decimal part of calorie intake needed
W2:	Integer part of calories consumed
W1:	Decimal part of calories consumed
F1$:	Name of food
FZ:	Gram units of F1$
FOOD$:	Names of food in data
CL:	Calories per 100 grams of FOOD$
FZ2:	Number of hundred units of FZ
FZ1:	Number of + units of FZ
FZ0:	Number of − units of FZ
1:	VRAM address

[PROGRAM MAP OF CALCALC]

10～150:	Personal information
160～190:	Calculation of calories required
200～330:	Input of food
340～380:	Calculation of calories consumed
390～430:	Display of results
440:	Work differentation data

[PROGRAM MAP OF VPOKE1]

10～50:	Enter VRAM
60～160:	Data on Basic Metabolism Values

[PROGRAM OF VPOKE2]

10～100:	Reading in of data
110～230:	Entry of food data
240～390:	Correction of food data
400～430:	Data save

PROGRAM LIST

CALORIE CALCULATION PROGRAM

```
10rem Calculation
20 Print "■■":view 1,0,30,23
30 Print cursor(3,3);"YOUR NAME ";:inPut NA$
40 Print cursor(3,5);"SEX?";cursor(4,7);"MALE---(1),FEMALE--
-(2)■";:inPut SE
```

```
50 if SE<1 or SE>2 then goto 40
60 Print cursor(3,9);"WEIGHT? (kg)■";:inPut KG
70 if KG<1 or KG>99 then goto 60
80 Print cursor(3,11);"AGE?■";:inPut YE:if YE<6 or YE>89 the
n goto 80
90 if YE<=19 then let M=2*(YE-6)else let M=2*(YE/10+12)
100 if SE=2 then let M=M+1
110 let M=2*M:let NO1=vPeek(M):let NO0=vPeek(M+1)
120 Print cursor(3,13);"HOW HARD DID YOU WORK?";cursor(6,15)
;"Light---------(1)"
130 Print cursor(6,17);"Normal--------(2)";cursor(6,19);"Mid
dle Heavy--(3)"
140 Print cursor(6,21);"Heavy---------(4)■";:inPut WO:if WO<
1 or WO>4 then goto 140
150 restore:for I=1 to WO:read WO1,WO0:next
160 let EA0=NO0*KG mod 100:let EA1=NO1*KG+NO0*KG/100:let EA2
=EA1/100:let EA1=EA1 mod 100
170 let EB0=EA1*WO0+EA0*WO1+EA0*WO0/100:let EB1=EA2*WO1*100+
EA1*WO1+EA2*WO0
180 let EB1=EB1+EB0/100:let EB0=EB0 mod 100:let EA0=((EB1 mo
d 9)*100+EB0)/9:let EA1=EB1/9
190 let EA1=EA1*10+EA0/10:let EA0=EA0 mod 10
200 cls:Print cursor(3,9);"INPUT FOOD";cursor(7,20);"Hit Ret
urn Key !!"
210 if inkey$<>chr$(13)then goto 210
220 let W2=0:let W1=0
230 let H=8:Print "■↓YOU SPENT THIS MUCH ENERGY";EA1;"←";EA0
;"←Cal←←←←←."
240 Print cursor(1,6);"****** FOOD ******"
250 Print cursor(10,22);"[NOW";W2;"←";W1;"Cal]←←←←←←←.";curs
or(1,4);"■"
260 let F1$="":Print cursor(2,3);"FOOD TYPE IS ■";:inPut F1$
:let I=100
270 if F1$="*" then goto 390
280 let F1$=left$(F1$+"          ",10)
290 let FOOD$="":for J=1 to 10:let FOOD$=FOOD$+chr$(vPeek(I)
):let I=I+1:next
300 let CL=0:let CL=vPeek(I)*100+vPeek(I+1):let I=I+2
310 if F1$=FOOD$ then Print cursor(5,4);"X(9)      X=";:inPut
 FZ:goto 330
320 if vPeek(I)=0 then Print "■":goto 260 else goto 290
330 if FZ<1 or FZ>999 then Print cursor(1,4);"■■■":goto 260
340 let FZ2=FZ/100*CL:let FZ1=(FZ mod 100)/10*CL:let FZ0=(FZ
 mod 100)mod 10)*CL
350 let FY1=(FZ1 mod 10)+(FZ0 mod 100)/10:let FZ2=FZ2+FZ1/10
+FZ0/100+FY1/10:let FZ1=FY1 mod 10
360 Print cursor(1,H);F1$;FZ;"9";cursor(19,H);FZ2;"←" FZ1;"C
al←←←←←←."
370 let W1=W1+FZ1:let W2=W2+FZ2+W1/10:let W1=W1 mod 10
380 if H/18 then goto 230 else let H=H+2:goto 250
390 cls:Print cursor(7,5);NA$;" RESULT OF 3";cursor(3,9);"AM
 UNT YOU TAKE        AMOUNT YOU NEED" READY
400 Print cursor(3,11);EA1;"←";EA0;"Cal←←←←←←.";cursor(17,11
);W2;"←";W1;"Cal←←←←←←."
410 Print cursor(15,19);"NEXT------(n)";cursor(15,21);"END--
------(e)"
420 Print cursor(15,17);"NEW MENU--(m)"
430 if inkey$="e" then end else if inkey$="n" then goto 20 e
lse if inkey$="m" then goto 220 else goto 430
440 data 13,23,15,00,17,85,20,80
```

V POKE 1

```
10 Print "■":let I=0
20 read NO1,NO0
30 vPoke I,NO1:vPoke I+1,NO0
40 let I=I+2:if I=84 then end else 9oto 20
50 data 4,86,4,61,4,56,4,29
60 data 4,26,4,02,3,97,3,76
70 data 3,73,3,56,3,53,3,33
80 data 3,33,3,12,3,13,2,93
90 data 2,95,2,76,2,81,2,59
100 data 2,72,2,48,2,64,2,42
110 data 2,59,2,40,2,54,2,38
120 data 2,43,2,34,2,31,2,20
130 data 2,27,2,12,2,25,2,09
140 data 2,21,2,09,2,16,2,11
150 data 2,11,2,13
```

V POKE 2

```
10rem CALCULATION(VPOKE2)
20 Print "■■"
30 let P=100
40 Print cursor(5,7);"1.FIRST INPUT";cursor(5,10);"2.INCREAS
E INPUT   ■";:inPut K
50 if K=1 then 9oto 100 else if K<>2 then 9oto 40
60 cls:Print cursor(4,6);"< PLEASE WAIT";cursor(4,9);"IF CAS
SETTE IS READY"
70 Print cursor(4,12);"PRESS RETURN KEY"
80 if inkey$<>chr$(13)then 9oto 80 else old "DATA"
90 if vPeek(P)<>0 then let P=P+12:9oto 90
100 let I=P
110 cls:Print cursor(5,7);"FOOD TYPE=■";:inPut FOOD$
120 if FOOD$="*" then vPoke I,0:9oto 240
130 let M=len(FOOD$):if M<2 or M>10 then Print "■":9oto 110
140 let FOOD$=left$(FOOD$+"          ",10)
150 for J=1 to 10
160 let A$=mid$(FOOD$,J,1)
170 vPoke I,ascii(A$):let I=I+1
180 next J
190 Print cursor(7,12);"Cal=■";:inPut CL
200 if CL<1 or CL>999 then Print "■":9oto 190
210 let CL1=CL/100:let CL0=CL mod 100
220 vPoke I,CL1:vPoke I+1,CL0
230 let I=I+2:9oto 110
240 cls:Print cursor(5,8);"PLEASE WAIT"
250 Print cursor(8,19);"Hit Return Key !!"
260 if inkey$<>chr$(13)then 9oto 260
270 let I=100
280 if vPeek(I)=0 then 9oto 400
290 let FOOD$="":for J=1 to 10
300 let FOOD$=FOOD$+chr$(vPeek(I)):let I=I+1:next
310 let CL=0:let CL=vPeek(I)*100+vPeek(I+1):let I=I+2
320 cls:Print cursor(5,8);FOOD$;CL;cursor(5,18);"MAKE CORREC
TION? (y/n)"
330 if inkey$<>"y" then if inkey$="n" then 9oto 280 else 9ot
o 330
340 cls:Print cursor(5,7);"FOOD TYPE=■";:inPut FOOD$
350 let M=len(FOOD$):if M<2 or M>10 then Print "■":9oto 340
360 let I=I-12:let FOOD$=left$(FOOD$+"          ",10):for J=1
to 10:let A$=mid$(FOOD$,J,1)
370 vPoke I,ascii(A$):let I=I+1:next
380 Print cursor(7,12);"Cal=■";:inPut CL:if CL<1 or CL>999 t
hen Print "■":9oto 380
```

```
390 vPoke I,CL/100:vPoke I+1,I mod 100:let I=I+2:9oto 280
400 cls:Print cursor(3,6);"< NOW, SAVE THE DATA":cursor(3,9)
;"IF CASSETTE IS READY"
410 Print cursor(3,12);"PRESS  RETURN KEY"
420 if inkey$<>chr$(13)then 9oto 420
430 save "DATA",0,I,1:end
```

DIET PLANNER

Weight changes displayed graphically

Jumping rope, jogging, jazz dancing — everybody is really working hard to control their weight, aren't they? Two or three times a day they get on the scales and stare at the needle. Is your weight about what it should be?

Here's where this program comes in, it allows you to easily understand weight changes by seeing them on a graph.

Content:

When you put in your weight for a given day, it will be combined with the date of 10 previous recordings and displayed in a line graph.

When the amount of data exceeds 10, the earliest data entry will be eliminated. Data for up to 5 people can be entered at one time which means the whole family can use this program.

Directions:

Enter the program exactly as it is listed. To begin enter "RUN 660".

When "READY" appears, enter "RUN". At this point you will be asked, "CHOOSE one?" and the numbers 1 to 5 will appear with "....."to their sides. This indicates that no data whatsoever has been entered. Enter the numbers from 1 to 5. When you do, you will be asked each time for names; each name may use up to five characters.

The next step is to enter the data for weights. The maximum limit is 100 kg. After entry has been completed, the message "Any Corrections?" will appear at the bottom of the screen. If you wish to make corrections, press key Y and this will allow you to re-enter the data for weights. If, you press the N key, a graph of the weights will be drawn.

After the graph has been drawn, hit the RETURN key. This will enable data for individuals to be deleted. After the deletions have been made, the number section will return to its original state. As long as you do not enter "6", input of data can be repeated. If "6" is entered, the program returns to the beginning and you will again be asked for which person data should be entered. To end the program press SHIFT + RESET.

After you have set your tape recorder with a blank tape and put it in the record position, enter SAVE, "DATA", 0, 74, 1 RETURN. "READY" will appear and your data will have been SAVEd.

After the initial setting has been established, OLD the program data with the OLD command after you have used CTRL + T to enter the text mode. Then enter "RUN" + RETURN normally. It is not necessary to repeat the "RUN 660" RETURN procedure executed at the beginning.

Program:

VRAM is used to save the data so it is not possible to add color.

In saving the data for each person, 5 bytes from the head are used for the name and the remaining 10 bytes are used to record the weight.

In the initial setting the name and the weight are written with characters 165, 226 of ASCII code.

Those for the name are entered in the VRAM one character at a time after they have been changed to ASCII code, however, the figures for the wieght, that is 1 to 100, are as numerical data in the VRAM.

Because characters are difficult to read in the text mode, characters are shown with a space between them.

[LIST OF VARIABLE]

NA$:	Name
NN$:	Name (Entered for the first time)
DJ:	VRAM address (Final)
E:	VRAM address to enter data for this time
KG:	Weight
TN$:	Name (For the graph)
AL:	Maximum value for graph
GQ:	Weight distribution per 1 memory after returning to AL
X:	X coordinate for writing graph
H:	Y coordinate for writing graph

[PROGRAM MAP]

10~60:	Selection of individual data
70~140:	Input of names
150~180:	Checks VRAM address for final data
190~270:	Input of weights
280~530:	Setting up graph
540~600:	Data erasure
610~660:	Subroutine to read in name data
670~730:	Initial setting of data

PROGRAM LIST

```
10 Print "■■■":view 1,0,39,23
12 cls:Print cursor(10,4);"CHOOSE ONE":let PK=11:gosub 445
30 Print cursor(12,17);"■";:inPut,DJ
40 if DJ<1 or DJ>5 then goto 3050 let DJ=(DJ-1)*15
50 let DJ=(DJ-1)*15
60 if vPeek(DJ)<>165 then goto 10
70 cls:Print cursor(9,5);"<  NE INPUT  >";cursor(10,8);"NAME
?  (UP TO 6 CHARACTERS)"
80 Print cursor(10,12);"■";:INPT,NN$
85 if len(NN$)>5 then goto 80
90 let NN$=left$(NN$+"     ",5)
95 for K=1 to 5
100 let KK$=mid$(NN$,K,1)
103 let F=ascii(KK$):vPoke K+DJ-1,F:next K
110 for E=DJ+5 to DJ+13
115 if vPeek(DJ+14)<>226 then let H=vPeek(E+1):vPoke E,H
120 if vPeek(E)=226 then goto 140
130 next E
140 cls:Print cursor(8,6);"WEIGHT?";cursor(10,8);"( UP TO 10
0kg )"
160 Print cursor(12,11);"■";:inPut,KG
170 if KG>100 or KG<1 then goto 160 else vPoke E,KG
180 Print cursor(8,18);"ANY CORRECTIONS?";cursor(10,20);"YES
 ...(y) , NO ...(n)"
190 if inkey$="y" then goto 140
200 if inkey$<>"n" then goto 190
223 let TN$="":for J=DJ to DJ+4
225 let TN$=TN$+chr$(vPeek(J))
227 next J
330 cls:Print cursor(6,0);"* ";TN$;" GRAPH *"
331 let AL=0:for J=DJ+5 to E
332 if AL<vPeek(J)then let AL=vPeek(J)
333 next J
334 if AL<25 then let GQ=1:let AL=25:goto 340
335 if(AL mod 25)=0 then let GQ=AL/25 else let GQ=AL/25+1:le
t AL=GQ*25
340 let I=1:let H=2
350 for J=DJ+5 to E
360 Print cursor(0,H);I
370 let X=4:let M=vPeek(J)/GQ
380 for N=1 to M
390 Print cursor(X,H);"■"
400 let X=X+1:next N
401 let AM=vPeek(J)mod GQ
402 if AM=0 then goto 410
403 if GQ<>2 then goto 405
404 Print cursor(X,H);"▌"
405 if GQ<>3 then goto 407
406 if AM=1 then Print cursor(X,H);"▌" else Print cursor(X,H
);"▌"
407 if GQ<>4 then goto 410
408 if AM=1 then Print cursor(X,H);"▌":goto 410
409 if AM=2 then Print cursor(X,H);"▌" else Print cursor(X,H
);"■"
410 Print cursor(30,H);vPeek(J);"(kg)"
415 let H=H+2:let I=I+1:next J
420 Print cursor(3,H);"▌═══════════╪═══════════▐"
425 Print cursor(3,H+1);"0";cursor(15,H+1);AL/2;cursor(27,H+
1);AL;
```

```
550 Print cursor(10,6);"YES---"  cursor(11,16); "■NO--- 6
 "::inPut,UH:if UH<1 or UH>6 then 9oto 550
560 if UH=6 then 9oto 20
570 let PH=(UH-1)*15:for J=PH to PH+4
580 vPoke J,165:next J
590 for J=PH+5 to PH+14:vPoke J,226:next J:9oto 540
600 for B=0 to 4:let H=B*15
610 for W=H to H+4
620 let NA$=NA$+chr$(vPeek(W))
630 next W
640 Print cursor(PK,2*B+6);B+1;" -  ";NA$
650 let NA$="":next B:return
660 Print "■■"
670 for I=0 to 4
680 for J=5 to 14
690 vPoke I*15+J,226
700 next J
710 for J=0 to 4
720 vPoke I*15+J,165
730 next J,I
```

PERSONAL COMPUTER TERMINOLOGY (2)

CENTRAL PROCESSING UNIT (CPU)

As the central component of a computer system, it reads, interprets, and executes commands. It is made up of a memory unit, operation unit, and control unit.

DIGITAL

Indicates numerical values and amounts.

DIGITIZER

Device for entering coordinates. According to where the cursor, the device which reads position, is located on the digitizer board, coordinates can then be entered onto the board.

There are also models where the board is transparent and light appears to be transmitted from the rear.

DISPLAY PAGE

Refers to the page being displayed at the moment. Can also be used for multi-page display.

HEADECIMAL

Numbers with a 16 base. Represented by the numbers 0 — 9 and the letters A — F.

JOYSTICK

A small stick several centimeters long which can be freely tilted in any direction. Depending on which way it is tilted, direction and degree can be entered. It gets its name because it resembles the control stick in airplanes.

It is used to move the indexes at the top of the screen and for moving the point of vision of objects shown at the top of the screen.

LIGHT PEN

Divice used for indicating coordinates on the screen depending upon where it is touched on the CRT screen. It receives its name because of the way it appears to receive light from the CRT.

PALETTE FUNCTION

Changes the palette number in response to the color code; this function instantaneously switches responses from color code to color.

PARALLEL INTERFACE

This is a circuit used to transfer n signals using a number of control wires when parallel data of n bit (for example, one word for the computer) is transferred.

RAM

A memory device with random access capability. RAM sands for Random Access Memory.

RASTER-SCAN GRAPHIC

Like a television receiver, an electric beam scans the screen surface and displays forms as a collection of points. This graphic form is noted for its diverse abilities such as being able to mold surfaces, diverse colors, halftones, and partial elimination.

ROM

Memory for reading out statements. In its normal use, it is a memory which cannot be written. Generally, fixed data, programs, etc. are written in.

ROM is an abbreviation of Read Only Memory.

SCREEN EDITOR

Displays the texts of source programs, etc. on the CRT screen, moves the cursor, and enables the text shown on the screen to be corrected.

It is a type editor for editing texts.

SIGGRAPH

An American organization established to research computer graphics and social enlightenment. It is composed of people from diverse backgrounds.

VIDEO MEMORY

Called refresh memory. Special memory for showing on the display; sends data at set intervals (refresh action).

VEIWPORT

This function prescribes the area within the CRT screen for graphic display. Depending on how it is used with the window function, enlargement or miniaturization of figures can easily be performed.

WORLD COORDINATES

An imaginary screen larger than the actual screen. Determines the area of display using such commands as the window command and displays on the actual screen.

CONVENIENT SOFTWARE SECTION

THE M5 IS GREAT FOR SUCH APPLICATIONS AS DATA ARRANGEMENT, CALCULATIONS OR MAKING TABLES AND GRAPHS. HERE WE'LL BE INTRODUCING SEVERAL SAMPLE PROGRAMS. TRY IT YOURSELF: YOU'LL FIND THE M5 OFFERS MANY WAYS TO ENJOY LIFE WITH COMPUTERS.

A USEFUL WAY TO MAKE SALES FIGURES UNDERSTANDABLE
- Bar graphs

SHOW AGE GROUPINGS FOR MALES AND FEMALES
- Population profiles

THE KEYSTONE OF A FILING SYSTEM
- Name & address records

KEEP THOSE LONG DISTANCE BILLS TO A MINIMUM
- Telephone rate table

MAKES IT EASY TO FIND THE WINNER
- Golf competition scale

BAR GRAPHS

Make sales figures clear at a glance

Contents:

With your entry of the title of the graph, data figures, data readout and numerical data, this program will produce a bar graph.

Depending on what you want, you can produce simple bar graphs, combined bar graphs, double-sided graphs, itemized graphs, percentage graphs and others. Here we explain the program format for the easiest type of graph, the simple bar graph.

Directions:

For example, suppose you want to make a bar graph based on the data given in table 1.

When typing in the RUN command, you will see the prompt "GRAPH TITLE (UP TO FIVE CHARACTERS)?" Type in "SALES" and press the RETURN key. Now you will enter up to five characters, but remember that ", 0 and blank spaces also count as one character, and that you cannot enter quotation marks. (This is the same when entering the data readout which appears afterwards.)

After selecting the title for the graph the prompt No. OF DATA ITEMS (UP TO 18) will appear. Here you type in the number of sales offices, which is 10. As per the prompt, the maximum you can enter on this program is 18.

Next, the prompt INSERT DATA READOUT (UP TO 5 CHARACTERS) will appear. Remembering the prompt when you typed in the title, proceed to type in the readout for each category.

Let's type in the names of sales offices "A" through "J". Next, the prompt DATA INPUT and a display will appear. Look over at table 1 to see the sales figure for "A", which is 893. Type this in, and move on to B, C, and so on, until you have entered all the figures.

Since this program does not have a data checking function, you will not be able to correct any input errors. Therefore be careful that you are typing in the correct figures.

After all data has been entered, the prompt GRAPH MAX? will appear. Here you enter the highest possible figure on the scale, which will be 900. At this time, when you enter the highest value on the scale from the data (i.e., 893) which is less than that maximum, this figure will be readjusted.

When the maximum scale is entered, the bar graph will be produced.

The Program:

In this program, the bar graph segments are produced by characters.

From 0 to the maximum figure of 20 characters, each item of data will be calculated and made into a number of characters. When the results are determined, the calculations will be lined up in the form of bars filled in with characters. The characters will be produced so that when a fraction is one, each character will be displayed as pseudo-characters in 1/8 segments. This will then result in more recognizable differences being produced in the displays.

TABLE 1: SALES PERFORMANCE TABLE

A SALES OFFICE	893
B SALES OFFICE	592
C SALES OFFICE	289
D SALES OFFICE	722
E SALES OFFICE	138
F SALES OFFICE	333
G SALES OFFICE	851
H SALES OFFICE	475
I SALES OFFICE	660
J SALES OFFICE	198

Finally, if the largest number entered on the bar graph scale is odd, any remainder in the center number will be rounded off by one-half that figure. This is an adjustment feature used only in the BASIC-I program, so you won't have to bother with this operation yourself.

[LIST OF VARIABLES]

GT$: TITLE OF GRAPH
N: NUMBER OF DATA
D (ROW): DATA
T$ (ROW): DATA READOUT
X, Y: SET COORDINATES WHEN ENTERING DATA
GW: LONGEST STRING OF CHARACTERS ON GRAPH
GH: MAXIMUM FIGURE ON GRAPH SCALE
G$: SINGLE CHARACTER BAR GRAPH
G1$: BAR GRAPH IN WHICH 18 GD IS SMALLER THAN OR EQUAL TO GD WHICH IS SMALLER THAN 19 ??
GD: NUMBER OF CHARACTERS IN EACH DATUM IN RELATION TO GW

GM: NUMBER OF 1/8 CHARACTERS OF FRACTIONS

[PROGRAM MAP]
10~40: SET INITIAL PERIOD
50~360: ENTER DATA
370~620: PRODUCE GRAPH
630~780: SUBROUTINE TO PRODUCE GRAPH CHARACTERS

PROGRAM LIST

```
10rem BAR GRAPH
20 Print "■"
30 view 1,0,31,23
40 cls
50 input "TITLE OF GRAPH(UP TO 5 CHARACTERS)";GT$:if len(GT$
)>5 then goto 40
60 cls:input "NUMBER OF ITEMS (UP TO 18 CHARACTERS)";N
70 if N>18 or N<1 then goto 60
80 dim D(N),T$(N)
90 for I=1 to N step 10
100 cls:let X=3:let Y=1
110 Print " INPUT DATA TITLE (UP TO 5 CHARACTERS)"
120 let JJ=I+9
130 if JJ>N then let JJ=N
140 for J=I to JJ
150 let Y=Y+2
160 Print cursor(0,Y);tab(30)
170 Print cursor(X,Y)
180 Print J;:input T$(J)
190 if len(T$(J))>5 then goto 160
200 next J
210 next I
220 let MAX=0
230 for I=1 to N step 10
240 cls:let X=3:let Y=1
250 Print "  INPUT DATA "
260 let JJ=I+9
270 if JJ>N then let JJ=N
280 for J=I to JJ
290 let Y=Y+2
300 Print cursor(0,Y);tab(30)
310 Print cursor(X,Y);T$(J)
320 Print cursor(X+12,Y);:input D(J)
330 if D(J)<0 or D(J)>999 then goto 300
340 if D(J)>=MAX then let MAX=D(J)
350 next J
360 next I
370rem ****** GRAPH ******
380 gosub 630
390 gosub 740
400 let GW=20
410 cls:input "  GRAPH MAX";GH
420 if GH<MAX then goto 410
430 cls
440 Print cursor(1,0);GT$
450 Print cursor(7,0);"0"
460 Print cursor(15,0);GH/2
470 Print cursor(25,0);GH
480 Print cursor(7,1);"━━━━━━━━━━━━━━━━━━━━━━"
490 for I=1 to N
```

```
500 let G$="":let G1$=""
510 if D(I)=0 then goto 590
520 let GD=D(I)*GW/GH:if GD=0 then goto 570
530 for J=1 to GD
540 let G$=G$+chr$(248)
550 if len(G$)=>18 and len(G$)<19 then let G1$=G$:let G$=""
560 next J
570 let GM=(D(I)*(GW*10/GH)-GD*10)*8/10
580 if GM>0 and GM<8 then let G$=G$+chr$(240+GM)
590 Print cursor(1,I+1);T$(I)
600 Print cursor(7,I+1);G1$+G$;D(I)
610 next I
620 end
630 for I=1 to 3
640 stchr "0080808080808000" to 241,I
650 stchr "00c0c0c0c0c0c000" to 242,I
660 stchr "00e0e0e0e0e0e000" to 243,I
670 stchr "00f0f0f0f0f0f000" to 244,I
680 stchr "00f8f8f8f8f8f800" to 245,I
690 stchr "00fcfcfcfcfcfc00" to 246,I
700 stchr "00fefefefefefe00" to 247,I
710 stchr "00ffffffffffff00" to 248,I
720 next I
730 return
740 for I=4 to 6
750 for J=1 to 8
760 stchr rPt$(8,"80")to 240+J,I
770 next J,I
780 return
```

POPULATION PROFILES

Show age groupings for males and females

This type of graph is of course by no means limited only to population statistics — it's useful for processing many types of data.

Contents:

This is a program for a two-sided graph in which readouts and data entered will appear in bar graph form from both left and right sides.

Directions:

Using the age data for males and females given in Table 1, lit's make a population profile graph.

Starting the program using th RUN command, you first see the prompt GRAPH TITLE MAX 5 CHARACTERS. Here, enter the title, "POP". Remember that ", 0, and blank spaces also count as characters, and that quotation marks cannot be used.

After this step is complete, next the prompt NO. OF DATA MAX 12 will appear. Enter 9 age groupings from Table 1.

Next, the prompt RIGHT TITLE UP TO MAX 5? will appear. Enter "FML" (female). The prompt for the right side will then appear, and here you should enter "MALE" (male). Remember that the five characters also include", 0 and blanks, and that quotation marks cannot be used.

With this done, the prompt DATA READOUT MAX 5 will appear, and when the readout for 1 comes to the screen, enter 0~9. Do this the same way for steps 2 through 9. After the readout input has been completed, you will see the display ENTER FML RIGHT, followed by 0~9?. Here you enter 91 according to the table. Continue with 2 through 9 according to the figures in the table. After this is complete, the display ENTER MALE LEFT will appear. Enter this information from the table as you have just done for the female side.

Remember that this program does not incorporate any data checking function, so that once a wrong figure has been entered by pressing

RETURN it cannot be revised.

After data for right and left sides has been entered, the prompt MAX FOR RIGHT GRAPH ? will appear. Here, enter the highest number on the scale as 100. Use the same figure 100 for the left side. These will be re-entered when the highest figures in your data goes below 100.

Once the maximum values have been entered for right and left sides, the population profile graph will be displayed.

Program:

IN UNIT 10,000 PEOPLE

MALE	AGE	FEMALE
95	0—9	91
88	10—19	84
85	20—29	84
100	30—39	100
82	40—49	82
60	50—59	68
37	60—69	47
22	70—79	29
59	80—	10

This population graph program produced the left and right sides of its bars by the SET CHARACTER command.

From 0 to the largest scale figure of 8, the amount of characters to fill in the bar are calculated and the corresponding amount of (box) figures are aligned horizontally.

If the maximum figure entered in the scale is an odd number, any remainder will be rounded off to 1/2 of the value of the central figure. This is a built-in function used only in M5 BASIC-I.

[LIST OF VARIABLES]

GT$: TITLE OF GRAPH
N: NUMBER OF DATA
T$ (ROW): (0) — RIGHT TITLE (1) — LEFT TITLE
M$ (ROW): DATA READOUT
MX1: MAXIMUM VALUE OF RIGHT DATA
MX2: MAXIMUM VALUE OF LEFT DATA
MX: SUBROUTINE OF MAXIMUM DATA VALUE
GW: NUMBER OF CHARACTERS IN LONGEST GRAPH
GH1: MAXIMUM SCALE OF RIGHT GRAPH
GH2: MAXIMUM SCALE OF LEFT GRAPH
DX: NUMBER OF CHARACTERS FOR EACH DATA IN RELATION TO GW
XO: X COORDINATES FOR DATA DISPLAY
X, Y: SET COORDINATES WHEN ENTERING DATA

[PROGRAM MAP]

10~20:	INITIAL SETTING
30~160:	ENTER DATA
170~340:	PRODUCE GRAPH
350~410:	DATA INPUT SUBROUTINE
420~440:	GRAPH CHARACTER PRODUCTION SUBROUTINE

PROGRAM LIST

```
10rem POPULATION PROFILE
20 print "■■":view 1,0,31,23
30 input "■GRAPH TITLE MAX 5 CHARACTERS";GT$:if len(GT$)>5 t
hen goto 30
40 input "■NO.OF DATA MAX 12";N:if N>12 or N<1 then goto 40
50 dim D(1,N),T$(1),M$(N)
60 input "■ RIGHT TITLE UP TO 5";T$(0):if len(T$(0))>5 then
goto 60
70 print cursor(0,3);:input "■LEFT TITLE UP TO 5";T$(1)
80 if len(T$(1))>5 then goto 70
90 for J=1 to N
100 if J mod 10=1 then print "■ DATA READOUT MAX 5":let X=3:
let Y=1
110 let Y=Y+2
120 print cursor(X,Y);"■";
130 print J;:input M$(J)
140 if len(M$(J))>5 then goto 120
150 next J:let MX1,MX2,MX=0
160 let A$="■ RIGHT ":let I=0:gosub 350:let MX1=MX:let MX=0:
let A$="■LEFT":let I=1:gosub 350:let MX2=MX
170rem ****** GRAPH *****
180 gosub 420:let GW=8
190 input "■ MAX FOR RIGHT GRAPH";GH1
200 if GH1<MX1 then goto 190
210 print cursor(0,3);:input "■MAX FOR LEFT GRAPG";GH2
220 if GH2<MX2 then goto 210
230 cls:print cursor(5,0);T$(1);cursor(13,0);GT$;cursor(21,0
);T$(0)
240 print cursor(2,1);GH2;cursor(6,1);GH2/2;cursor(11,1);"0"
;cursor(19,1);"0";cursor(22,1);GH1/2;cursor(26,1);GH1
250 print cursor(4,2);"________          ________"
260 let K=0:let GH=GH1:for J=1 to N*2
270 if J>N then let K=1:let GH=GH2:print cursor(13,2+J-N*K);
M$(J-N*K)
280 let DX=GW*D(K,J-N*K)/GH
290 let X0=19+DX-(9+2*DX+len(num$(D(K,J-N*K))))*K
300 if DX<1 then let X0=18-(7+len(num$(D(K,J-N*K))))*K:goto
320
310 print cursor(19-(7+DX)*K,2+J-N*K);rpt$(DX,chr$(248))
320 print cursor(X0,2+J-N*K);D(K,J-N*K)
330 next J
340 end
350 for J=1 to N:if J mod 10=1 then print A$;T$(I);" IN A DA
TA":let X=3:let Y=1
360 let Y=Y+2
370 print cursor(X,Y);"■";M$(J)
380 print cursor(X+8,Y);:input D(I,J)
390 if D(I,J)<0 or D(I,J)>999 then goto 370
400 if D(I,J)>=MX then let MX=D(I,J)
410 next J:return
```

```
420 for I=1 to 3:stchr "00ffffffffffff00" to 248,I:next I
430 for J=4 to 6:stchr "8080808080808080" to 248,J:next J
440 return
```

NAME & ADDRESS RECORD

The keystone of your filing system

Contents:

This program can store a maximum of 128 names (each with up to 16 characters), their addresses (each with up to either 16 or 32 characters), and their telephone numbers. This will allow you to call them for display on the screen; make new additions or deletions; and store the contents on a cassette.

Correction or deletion of any entered data is performed on the display.

Besides being able to read all data from the name list, the program has a search facility which will let you locate any name by entering its initial letters.

Directions:

When you enter the Run command according to the listed program, the prompt: IS THERE ANY TAPE DATA? will appear. Here, for your first entry, type N. Then the new register display will appear.

On the display you will see. New entry up to 16 characters — 1. name. First enter the name, followed by the address and telephone number. The first item can accept a maximum of 16 characters, including ", 0 or blank spaces. Also when inserting names or addresses, be sure and be careful of the following.

For the address, a maximum of 16 characters can be entered. Should this number be exceeded, after pressing the RETURN key the prompt ADDITIONAL? will appear. By pressing the Y key, the remainder of the address can be entered. If the first 16 characters are sufficient, then press the N key. Finally, enter the telephone number. The postal zip code is part of the address.

After entering the telephone number, the prompt, IS THIS CORRECT? will appear. If correct, type Y, and if you wish to make any revisions, type N. When you press N and the prompt NO? appears, you can revise any of the data by typing the number of the item. When you type the number, the cursor will move automatically, permitting you to insert the new data.

If you typed the Y key, the prompt CONTINUE ENTRIES ? will appear. If you want to enter new information, type Y, and if you are finished type N. If you type N at this point the menu will be displayed and you can perform the 4 (DATA SAVE) operation.

By pressing 4, OK ? will appear in the left corner of the display. Insert a cassette in your cassette recorder, and after having pressed the record button, press the required keys and hit RETURN. After a brief pause the Ready display will appear, indicating that storage of the data on the tape has been completed.

If you're doing more than two SAVEs, after the RUN command when the IS THERE TAPE DATA ? question appears, hit the Y key. Then press the playback button on the tape recorder with the cassette containing the data to OLD the data. When the OLD has been completed, the menu will be shown on the display. By selecting the output from 1, the question will appear. If you want to see the entire list of names, type Y. Type N if you want access to only a specific name.

By pressing, Y the data will appear in the order, 1. Name; 2. Address (1), 3. Address (2); 4. Telephone number. Here you may chose from four operations: H — (CHANGE); C — (DELETE); N — E (NEXT); or E — (END). If you type H, you can either enter new data or revise existing data. By typing C, the display will stop with the next person's name, and at that time it will automatically delete whatever data about that person you request it to. Typing E returns you to the menu.

When the prompt EVERYONE ? appears, and you press the N key, the display will clear, and the prompt Key Word ? will appear. Here, type in the first letter of the name of the person whose information you wish to access. If no name is listed with that letter a buzzer will sound and you may enter a new letter.

If more than one person is listed whose names begin with that letter, all of them will be called to the screen one after the next. When all the names have appeared, NOW COMPLETED will appear on the display. By entering the appropriate key, you can return to the menu. If you pressed 2 (to add new data), be sure to save it by pressing 4.

You can exit from the program by typing 3.

Program:

In this program, addresses and other data are entered directly into the VRAM by the VPOKE command, and when required are read out by the VPOKE command.

After the data for a certain person has been deleted, the name in the list which followed it is not moved forward, but left as a blank segment. (The memory is not equipped with a routine to move it forward.) Therefore when a blank segment is encountered upon a request for a readout of the entire name list, the program moves on to the next listing.

It is therefore necessary to enter the beginning of the name. When entering a new listing after having deleted a previous one, since the blank after deletion is buried in the record order, the output order will never be the same as the order in which entries were made.

When entering characters for new listings or for revisions, since the blank space is also considered a character, if you enter a blank in the address, etc. it will facilitate the readout.

[LIST OF VARIABLES]

D$(No.):	(0) ... Name (Last, First), (1) ... Address (1)
	(2) ... Address (2), (3) ... Telephone number
A:	Selection number from menu
N:	Number of names in address listing
S:	Number of for revisions

[PROGRAM MAP]

10~30:	TAPE DATA input
40~70:	Menu
80~200:	New listings
210~330:	Screen display
340~390:	Changes, revisions
400:	Store data

PROGRAM LIST

```
10 Print "↓■2.NEW RECORD";"■■3.END"
15rem ADDRESS RECORD
20 inPut "■■IS THERE ANY TAPE DATA?";Z$:if Z$<>"y" and Z$<>"
n" then 9oto 20
30 dim D$(3):Print "■■":view 1,0,31,23:if Z$="n" then vPoke
0,0:let C=1:9oto 90 else old "add"
40 Print "■■←● ADDRESS  BOOK ●";"↓■■←INPUT PROCESSED NUMBERS
";"↓↓■■1.OUTPUT";"■■→→DELETE";"■■→→CHANGE"
50 Print "↓■2.NEW RECORD";"■■3.END";"■■4.DATA SAVE"
60 Print cursor(5,16);"■WHICH ONE";:inPut A:if A<1 or A>4 th
en 9oto 60 else let N=vPeek(0):let C=0
70 if A=4 then 9oto 400 else if A=3 then end else if A=1 the
n 9oto 210
80 let SN=0:for I=1 to N*64-63 step 64:let SN=SN+1:if chr$(v
Peek(I))=" " then let N=SN-1:9oto 90 else next:let C=1
90 let D$(0)="":let D$(1)="":let D$(2)="":let D$(3)=""
100 Print "■   NEW RECORD";"■■→→MAX 16 ";rPt$(16,"_")
110 Print cursor(1,4);"■1.NAME    ";:inPut D$(0):if len(D$(0)
)>16 then 9oto 110
120 Print cursor(1,6);"■2.ADDRESS";:INPUTREADY):if len(D$(1)
)>16 then 9oto 120
130 Print cursor(1,8);"■";:inPut "3.ADDITIONAL ADDRESS ";Z$:
if Z$<>"y" and Z$<>"n" then 9oto 130 else if Z$="n" then 9ot
o 150
140 Print cursor(9,8);"■";:inPut D$(2):if len(D$(2))>16 then
 9oto 140
150 Print cursor(1,10);"■4.TEL    ";:inPut D$(3):if len(D$(3)
)>16 then 9oto 150
```

```
160 Print cursor(5,15);"■CORRECT?";:inPut Z$:if Z$<>"y" and
Z$<>"n" then 9oto 160 else if Z$="y" then 9oto 180
170 Print cursor(5,15);:inPut "■No.";S:if S<1 or S>4 then 9o
to 170 else Print cursor(9,(S+1)*2);:inPut "■";D$(S-1):9oto
160
180 9osub 370
190 Print cursor(7,19);"■CONTINUE ENTRIES?";:inPut Z$:if Z$<
>"y" and Z$<>"n" then 9oto 190
200 if Z$="y" then let N=N+C:9oto 80 else 9oto 40
210 Print cursor(8,18);"■EVERY ONE?";:inPut Z$:if Z$<>"y" an
d Z$<>"n" then 9oto 210
220 let SN=0:if Z$="n" then 9oto 250
230 for JJ=1 to N*64-63 steP 64:let SN=SN+1:if chr$(vPeek(JJ
))=" " then next else 9osub 290:let N=vPeek(0):next
240 9oto 280
250 inPut "■  Key Ward";Q$:let Q$=mid$(Q$,1,1):let SW=1:if Q
$="0" then 9oto 40
260 for JJ=1 to N*64-63 steP 64:let SN=SN+1:if Q$<>chr$(vPee
k(JJ))then next else let SW=0:9osub 290:let N=vPeek(0):next
270 if SW=1 then Print "■":9oto 250
280 Print cursor(5,17);"NOW COMPLETED";:inPut,Z$:9oto 40
290 for J=0 to 3:let D$(J)="":for K=0 to 15:let D$(J)=D$(J)+
chr$(vPeek(JJ+J*16+K)):next:next
300 Print "■■■■■■■■■→1.NAME      ";D$(0);"■■■→2.ADDRESS  ";D$(1)
310 Print "■■→3.■■→→→";D$(2);"■■■→4.TEL      ";D$(3);"■■■■■→→→H··C
HANGE  C··DELETE"
320 Print cursor(3,15);:inPut "■N··NEXT  E··END";Z$:if Z$<>"
h" and Z$<>"c" and Z$<>"n" and Z$<>"e" then 9oto 320
330 if Z$="e" then 9oto 40 else if Z$="n" then return
340 let N=SN-1:if Z$="c" then 9oto 360
350 Print "↓↓→→";:inPut "No.";S:if S<1 or S>4 then 9oto 350
else Print cursor(9,(S+1)*2);:inPut "■";D$(S-1):9osub 370:9o
to 300
360 let D$(0)="":let D$(1)="":let D$(2)="":let D$(3)=""
370 for I=0 to 3:let B=16-len(D$(I)):if B=0 then next else l
et D$(I)=D$(I)+rPt$(B," "):next
380 for M=0 to 3:let II=0:for I=N*64+1+M*16 to N*64+16+M*16:
let II=II+1
390 vPoke I,ascii(mid$(D$(M),II,1)):next:next:vPoke 0,vPeek(
0)+C:return
400 inPut "OK ";Z$:save "add",0,vPeek(0)*64,1:end
```

COMPUTING A GOLF COMPETITION

Contents:

1 Computing the gross results

By entering the total number of strokes, the program will organize the players beginning from the lowest number, and give a readout in order of performance. If two or more players have the same gross totals, they will both be given the same number in the rankings.

2 Computing the net results

By entering the handicaps to the number of total strokes, this will compile the net in order from lowest to highest and display the results. In the event of a tie, the player with the lower handicap will be ranked higher.

Directions:

By beginning the RUN command, the prompt will ask for the number of players. Here, enter the number of participants, between 2 and 10. If the number is larger than 10, you will have to repeat the operation. The next prompt will be for the total par of the course; enter the par at this time. These two operations completed "IS THIS CORRECT ?" (any key/n) will be displayed. If the entry is correct, type any key except N; if wrong, enter N. Typing N will let you make whatever corrections are needed.

After the number of participants and course par have been entered correctly, you can proceed with name, gross, and handicap. First, when the prompt "1 NAME" (first players name) appears, enter the first name up to a maximum of five characters. Next, enter the gross (number of total strokes) and handicap in that order. After this, the prompt 2 NAME (second players name) will appear and you can repeat the process until all participants have been entered.

With the above completed, the menu will be displayed, letting you select between 1: NET rankings; 2: GROSS rankings; 3: corrections; and 4: end. If you select 1 or 2, after the rankings are displayed, you will see the prompt "hit any key". Typing the appropriate key will return you to the menu.

By selecting 3 for corrections, "1 any key/n" (Change the gross or handicap for player no. 1 any key/n) will appear. If you do not need to revise any data, type the N key. Otherwise, type any other key.

If you type N, the data for the next person will be displayed. When the data is displayed for the player you want to revise, pressing the appropriate key will let you re-enter the data starting from his name. After the check for all players is complete, the menu will return.

By selecting 4, the display will be cleared and the program finished.

Program:

The maximum of participants that the program can process is 10. If you want to enter a larger number than 10, revise the program by downloading the data into space in the video memory (VRAM). Or, enter the program in BASIC-G.

In this program, there are two displays for the NET and GROSS rankings; in order to conserve memory, the same routine is used for NET ranking arrangement and GROSS ranking arrangement.

[LIST OF VARIABLES]

N:	Number of players
P:	Par
A $ (N):	Name
B (N):	Total number of strokes (gross)
C (N):	HDCP (handicap)
D (N):	GROSS-HDCP (net)
E:	Ranking

[PROGRAM MAP]

10~30:	Assign color, display
40~50:	Enter par, number of players
60:	Arranger language
70~100:	Data input routine
110~130:	Menu display
140~210:	Data sorter
220~290:	Display (net ranking, gross ranking)
300~330:	Revisions

PROGRAM LIST

```
10 Print "■■■■■"
20 let CO$="8080808080808080":stchr CO$ to 226,4:for I=4 to
6:stchr CO$ to 232,I:stchr CO$ to 136,I:next
30 Print cursor(5,0)"●●●●●●●●●●●●●●●●●●●●●●●●":cursor(10,1)"COMP
ETITION":cursor(5,2)"●●●●●●●●●●●●●●●●●●●●●●●●":view 1,3,31,23
40 Print "■":cursor(5,5)::input "PLAYERS BETWEEN 2-10";N:if
N<2 or N>10 then goto 40
50 Print cursor(5,6)::input "PAR";P:Print cursor(5,8);"IS TH
IS CORRECT? (any key/n)"::input X$:if X$="n" then goto 40
```

```
60 dim A$(N),B(N),C(N),D(N):let B(0)=999:let C(0)=999
70 let D(0)=999:for J=1 to N
80 cls:Print cursor(10,5);"PLAYER NO." J;cursor(10,6);:inPut
 "NAME";A$(J)
90 Print cursor(10,7);:inPut "GROSS";B(J):Print cursor(10,8)
;:inPut "HDCP";C(J)
100 let D(J)=B(J)-C(J):if FL=1 then 9oto 330 else next
110 cls:Print cursor(10,10);"1:NET RANKING";cursor(10,11);"2
:GROSS RANKING";cursor(10,12);"3:CORRECTIONS";
120 Print cursor(10,13);"4.end";cursor(16,13):inPut M:if M<1
 or M>4 then 9oto 110
130 cls:if M=1 then let &=1:9oto 140 else if M=3 then 9oto 3
00 else if M=2 then let &=2:9oto 140 else end
140 let E=1:for K=N-1 to 1 steP-1:for J=1 to N-1
150 if &=1 then if D(J)<D(J+1)then 9oto 210 else if D(J)=D(J
+1)and C(J)<=C(J+1)then 9oto 210
160 if &=2 then if B(J)<B(J+1)or B(J)=B(J+1)then 9oto 210
170 let X$=A$(J):let A$(J)=A$(J+1):let A$(J+1)=X$
180 let M=B(J):let B(J)=B(J+1):let B(J+1)=M:let M=C(J)
190 let C(J)=C(J+1):let C(J+1)=M:let M=D(J):let D(J)=D(J+1)
200 let D(J+1)=M
210 next:next
220 Print cursor(0,3);"──────────────────────────────────────";curso
r(0,4)"PAR";P;cursor(0,5);"NAME   GROSS  HDCP  NET RANKING"
230 Print cursor(0,6);"──────────────────────────────────────":for J
=1 to N
240 Print cursor(0,J+6);A$(J);cursor(8-len(num$(B(J))),J+6);
B(J);
250 Print cursor(9,J+6);"(";(B(J)-P);")";cursor(15,J+6);C(J)
260 if &=1 then if D(J-1)<>D(J)then let E=J else if C(J-1)<>
C(J)then let E=J
270 if &=2 then if B(J-1)<>B(J)then let E=J
280 Print cursor(19,J+6);D(J);cursor(26,J+6);E:next
290 Print cursor(0,6+J);"══════════════════════════════════════":inP
ut "hit any key";U$:9oto 110
300 for J=1 to N:cls:Print cursor(10,5);"PLAYER NO." J;curso
r(10,6);"name";A$(J)
310 Print cursor(10,7);"GROSS ";B(J);cursor(10,8);"HDCP";C(J
)
320 Print cursor(5,20);"CORRECTIONS any key/n";:inPut X$:if
X$<>"n" then:let FL=1:9oto 80
330 let FL=0:next:9oto 110
```

HELPFUL HINTS

It's helpful if you commit these terms to memory:

Debugging methods and error messages

Debugging method

The debug function discovers any "bugs" (errors in the program) so that you can correct them.

In a bug, there are two situations: ones in which no error message is given, such as those involving results of calculations or mistakes in programming and ones in which the error message is given.

Here we explain the debugging method.

(1) Debugging method for when error messages are displayed.

The display shows the error message below:

error message line number on which message appears

Here, refer to the error message chart, and check the contents of the message. Also refer to the list's line numbers and find the program bug according to that error message. Then use the screen editor to make corrections.

(2) Debugging method when error message is not displayed but results or processing is incorrect.

[1] Use the PRINT command to enter the name of the variable which you think is wrong into the program. Then list the program and compare this with the actual value.

[2] Stop the program directly while running. In this case type CTRL + RESET.

To rerun the program press the CONT RETURN keys.

Next, in the stopped mode, check the value of the variable with the direct order.

Since the PRINT variable will determine its value with RETURN, you can compare the value with the actual figure.

If the actual figure is wrong, you have found a bug. Correct it at this point.

ERROR MESSAGE CHART

Message	contents of error	reason for error
ERR 1	error in FOR-TO-NEXT commands	● no FOR command was given for the NEXT command
ERR 2	statement error	● non-existing command language was typed for command
ERR 3	subroutine error	● CLEAR command was used while inside subroutine ● skipped to subroutine due to GOTO command ● only RETURN command was provided without GOSUB
ERR 4	error in READ, DATA message	● insufficient data in DATA message ● no DATA message for READ command
ERR 5	irregular figure obtained	● the figure provided for that function is irregular
ERR 6	overflow	● results are too large or small for multiplication or power calculations
ERR 7	memory overflow	● too many variables used ● too many subroutines used
ERR 8	no line specified	● no line specified for GOTO or GOSUB commands
ERR 9	error in line variable	● mistake at time of line statement ?? ● figure in ( ) is negative number or 0
ERR 10	error in line variable	● Same variable used in two line statements
ERR 11	division by 0	● number being divided is divided by 0
ERR 12	direct command is inappropriate	● error inside direct command ● CONT command was made when continuing operation was impossible
ERR 13	data mismatch	● different types of data modes used together
ERR 14	stack overflow	● insufficient memory for character line
ERR 15	error in character line	● line too long ● during calculations line became too long, exceeding 19 characters
ERR 16	error in line variables	● non-stated line used
ERR 17	$ (label) defined twice	● same label used twice or more in jump destination
ERR 18	tape lead error	● programming of OLD, VERIFY, CHAIN commands not performed properly
ERR 19	wrong display mode	● mistake in selection of G1, G2, multicolor and text modes
ERR 20	sprite error	● MOVE command made for sprite coordinates which have already left the display
ERR 21	stack error	● wrong PUSH-POP response
ERR 22	error in REPEAT-UNTIL	● wrong REPEAT-UNTIL response
ERR 23	time out error	● input time ended for any INPUT commands
ERR 24	error in RESUME message	● though error has not occurred, the RESUME message was produced
ERR 25	INPUT message error	● RETURN key was pressed without entering data

MORE WAYS TO USE THE M5

- Creative language section

BASIC-I
BASIC-G
FALC

- Creative application section

CAR RACE GAME USING BASIC-G
DIGITAL-ANALOG CLOCK

- Advanced creative section

PRODUCE ANIMATION
ENJOY MUSIC
USE A MACHINE LANGUAGE MONITOR

BASIC-I

BASIC-I is one of the ROM cartridges provided with the M5 at the time of purchase. Since it is a beginner's language, BASIC-I is a bit inconvenient for producing graphics or music functions. For these and other more complex functions, we recommend use of the BASIC-G program.

In BASIC-I, only about 3 kilobytes of memory is available to the user, so it can only handle a program of 150 steps or less. When entering the listings given here, you will be able to save memory by entering the lines in the VRAM (video memory) or by saving data for producing sprites on the cassette tape.

For graphics, the sprite function is an important feature.

The sprite is an animated pattern by which an 8×8 or 16×16 dot image can be produced and assigned to 32 sprites. The sprites have a priority arrangement, and in overlapping whichever pattern has highest priority is displayed. By this function, it is easy to express the degree of distance or depth in the picture. Since the sprite can be made to move rapidly, it makes it possible to produce for better animation on a personal computer using BASIC, than in the past.

It is possible to produce the sprite in 16 different colors.

The display can be switched to four different modes, including GRAPHICS I (G I), GRAPHICS II (G II), Multicolor and Text.

Music can be produced directly by means of the OUT command on the computer's sound generator. Data can be entered in the the SML (SORD MUSIC LANGUAGE) Interpreter and the POKE message used to produce music.

In any case, since this is a beginner's program, it is necessary to take additional steps if expansion of functions is desired.

Display mode	Character pattern	Characters (horiz-vert)	Resolution	Sprite applicable
GI	8×8 dot	24×32	192×256	○
GII	8×8 dot	24×32	192×256	○
Multicolor	4×4 dot	24×32	48×64	○
Text	8×6 dot	24×40	192×256	×

BASIC-G

BASIC-G is BASIC-I with enhanced capability for graphics and music. The user memory of this program is approximately double that of BASIC-I. In order to make it able to incorporate nearly all commands, the processing speed is faster, and it can also produce sound and pictures, accepting input entered with keyboard or joypad.

It is very easy to produce simple games in BASIC-G, and you will also be able to produce graphics at a speed never before possible on personal computers.

The sprite can be used under BASIC-I by means of the FOR loop, but under BASIC-G with the MOVE command you can freely select the speed or direction. The joint command lets you combine sprites in order to produce a large moving picture pattern.

Graphics commands beside the sprite have been enhanced, including a CIRCLE command which includes circles, wedges and ovals; a DRAW command which will produce straight lines; and a PAINT command that will fill in colors. The BOX or BAR commands can be used to draw simple boxes or bar lines. The image data you have produced can be stored on cassette tape, and reused again whenever you like.

By connecting the optional printer unit, you'll have the capability to print out listed programs by bit image in reverse characters. You can also use the display copy command to produce two-sided image copies. For music, you can use the PLAY command to facilitate producing sound, and also change the length of the tones, raise or lower their volume, change the tempo, change envelopes, move modulation, rest, and harmonize. This lets you compose and perform your own tunes. This type of processing is done by interruptions, so it is also easy to produce sounds while drawing pictures, and so on.

In this way you can also put BASIC-G to use to produce animated cartoons using sprites, and accompany the cartoons with music.

Give it a try: your own computerized cartoons!

FALC

FALC is a SORD program language developed for use with the M5 by which data is entered as a table. Afterwards, commands permit many free forms of editing.

(1) THE DISPLAY IS THE BASIS — under FALC, the display is the basis, and data of up to 4,000 characters may be entered.

(2) SIZE OF THE DISPLAY IS FREE — The display may be produced in any range within 255 characters by 15 lines. According to the number of items or characters the memory may be freely varied. During or after data is entered on the screen the number of items may easily be changed, added or deleted.

(3) EASY CORRECTION, ADDITION OR DELETION OF DATA — whether data is characters, numbers or graphics, it may be easily revised, added or deleted.

(4) CALCULATION FUNCTION — if the data items on the screen are numerical they may be used in calculations both horizontally and vertically, and those figures entered in another item. The computer may be utilized as a calculator. Refer to Table 1 for the codes used to perform calculations.

(5) DATA CAN BE CONVERTED TO GRAPHICS — the numerical data produced by FALC can be easily converted to horizontal graphs. Up to five items may be produced in an aggregate graph.

(6) THE DISPLAY CAN BE SAVED on a cassette tape — in addition to the image being displayed at the moment, two pages of main memory and three pages of calculations may be saved. However, this data will all be erased when the power is turned off. Therefore under FALC each page on the display can be named and saved on a cassette tape. If this tape is given the OLD command, it can be recalled to the screen at any time.

(7) PRINT OUT THE DISPLAY — using the optional printer, the display can be printed out in hard copy as it appears.

With FALC, all the above commands can be processed with just 10 commands. (Table 2)

You can use FALC at home for a wide range of applications: telephone memos, calendar memos, TV program charts, scheduling activities, etc.

Table 1

Calculation codes

Code	Function
+	Addition
−	Subtraction
*	Multiplication
/	Division
<	Power (simple precision)
* *	Power (high precision)

FUNCTION CODES

Code	Function
ABS (X)	absolute value
SIN (X)	sine
COS (X)	cosine
TAN (X)	tangent
EXP (X)	exponent
LOG (X)	logarithm
LN (X)	natural
SQR (X)	square root
SGN (X)	sign
INT (X)	integer

NUMERICAL VALUE IS INSERTED INTO ( )

Table 2

Name of command	function
P (PUT)	Save in display memory Save from memory to cassette
G (GET)	Call from cassette to memory Call from memory to display
L (LIST)	Display copy on optional printer
W (WRITE)	Write on screen
CTRL+DEL+X (DELETE)	Delete one character at a time Delete one line of program
SORT	Sort program contents
S (SEARCH)	Conditional search of program contents
M (MOVE)	Transfer program to blocks
GR (GRAPH)	Make program contents into graph
N (NEW)	Change in display mode

BASIC-G PROGRAM : CAR RACING GAME

Creative applications

This is a car racing program written with BASIC-G. It will give faster operation and pattern changes that would a program under BASIC-I.

Contents:

This game requires the "driver" to race his car for over three minutes while avoiding three types of hazards. The pattern of the hazard course includes a zigzag course where the road widens and narrows; a course with land mines; and a course with dangeous competitor. These appear randomly. Before a hazard makes its appearance, a warning mark appears on the display. If the car collides with a guard rail, land mine or the competitor's car, it will "explode", signalling an end to the game.

Car operations are performed by a joypad. For users without a joypad, changing line numbers 25, 710 and 720 will make it possible to employ the keyboard. The joypad should be connected to the jack on the right.

Directions:

Car operation is performed either by the joypad or the keyboard. In the case of the joypad, movement in eight directions is possible: up and down, plus three each to the right and left. This facilitates left-right movement. If using the keyboard, the cursor keys ↑ ↓ ← → are used (actually these are @ / ; :.)

When the program is run, a green car drops down from above. This is your car. It will take off from the starting point on the display. When it takes off, the first hazard warning will appear, so use the joypad or cursor keys to maneuver the car and avoid whatever hazards you encounter. If you successfully avoid the hazard, keep on course until the next one is encountered.

The farther the car advances the higher the score will climb, but it will become progressively difficult to avoid the hazards. The level of the score can be indicated by three levels of tones.

If the car remains safe for three minutes, or if it explodes, the display

will change, and your driving time and score will appear. The highest score previously recorded will also be displayed. Then the message PLAY AGAIN ? (Y/N) will appear. If you want to play once again, type Y, if not, type N.

Program:

Since this game uses BASIC-G, there is sufficient memory remaining even with this program entered. You can use the extra memory to add sound effects, music, or hazard pattern subroutines of your own to modify the game.

With the joypad, left-right movement should be relatively easy, but for those who find it too difficult, the IF message on lines 700~720 allow you to modify movement to easier levels.

For those who find the road too narrow, raising the WDl figure on line number 620 will widen the road.

The number of competitors' cars can be increased by raising the figure of the TK2 on line number 850.

Since the decision for collisions with the competitor's car does not use an interrupt, there might be cases where even if there is a collision no explosion will occur.

Those players who desire a longer game time can extend the race by raising the figure of the ALARM on line number 70. (Example: 00 : 05 will extend the race to five minutes.)

[EXPLANATION OF LABELS]

$ZG: Produce zigzag road
$RT: Use when road turns right
$LT: Use when road turns left
$WD: Set road width
$LD: Detect road, car movement or contact with hazards
$TK: Used when competitor car appears
$ED: Processing at end of game
$BM: Processing of explosion
$JR: Used when land mine appears

[LIST OF VARIABLES]

RX:	Character coordinates for left side of guard rail
WD, WDl:	Road width
CX, CY:	Sprite coordinates for own car
CL:	Color or competitor car
COL, CO2:	Sprite collision flag
GX, GY:	Character coordinates for own car
JY:	Joypad direction
TKl:	Number of times competitor car appears
TK2:	Number of competitor cars
TK3:	Number of goal points for competitor car movements
CD:	Score

HS: High score
RD1: Jumping out of hazards
RD2: Length of zigzag road
P: Flags on right and left curves in road
Also, to make I and J begin, the for-next message

[PROGRAM MAP]

10: REM message
20～100: Initialize
110～180: Fixing car sprites
190～280: Fixing of warning patterns
290～380: Fixing of explosion patterns
390～490: Fixing of guard rail and land mines
500: Startup sound
520～540: Main routine
560～670: Make zigzag course
680～790: Road
800～880: Appearance of competitor car
890～930: End of game processing
940～1000: Explosions
1010～1030: Appearance of land mines

} subroutine

PROGRAM LIST

```
10' CAR RACE:BASIC-G
20 dim GR(3)
30 HS=0
40 Print "■■":view 1,0,31,23
50 alarm on:coinc on
60 time$="00:00:00":alarm$="00:03"
70 on alarm gosub $ED
80 let RX=7:let WD,WD1=10:let CX=104:let CY=152:CD=0
90 randomize:bcol 4
100 mag 2:!Poke &701A,Peek(&701A)and &EF
110 for I=0 to 20 step 4
120 stchr "000701031b1e1a02" to I+148
130 stchr "0303076f7f6f0504" to I+149
140 stchr "00e080c0d8785840" to I+150
150 stchr "c0c0e0f6fef6a020" to I+151
160 if I=0 then let CL=7 else let CL=rnd(12)+1
170 scod I/4+5,I/4+148:scol I/4+5,CL
180 next
190 stchr "000010f7ee07e0f" to 140
200 stchr "0100000031ffce0" to 141
210 stchr "073ff8c000000080" to 142
220 stchr "f07e077ef0800000" to 143
230 scod 4,140:scol 4,6
240 stchr "0000000000000000" to 144
250 stchr "0003033f7fff0000" to 145
260 stchr "0000000000000000" to 146
270 stchr "00c0c0fcfeff0000" to 147
280 scod 3,144:scol 3,1
290 stchr "08a4422291092503" to 200
```

```
300 stchr "a902420404480910" to 201
310 stchr "10410214295240c0" to 202
320 stchr "954040a422101008" to 203
330 scod 1,200:scol 1,7
340 stchr "984a884424124683" to 204
350 stchr "074a12a445880650" to 205
360 stchr "883291a224c8d2a0" to 206
370 stchr "a4504ca422942251" to 207
380 scod 2,204:scol 2,6
390 for I=1 to 3
400 stchr "ffffffffffffffff" to 179,I
410 stchr "eeeeeeeeeeeeeeee" to 179,I+3
420 stchr "ffffffffffffffff" to 180,I
430 stchr "6666666666666666" to 180,I+3
440 stchr "ffffffffffffffff" to 181,I
450 stchr "ffffffffffffffff" to 181,I+3
460 stchr "0000000000000000" to 182,I+3
470 stchr "183c66dbdb663c18" to 183,I
480 stchr "1e1e1e1e1e1e1e1e" to 183,I+3
490 next
500 for I=0 to 10:CY=I*16:gosub $LD:sg 3,,0:next:for I=0 to
1000:next:play "h3v15o5dddo6h8d.":for I=0 to 5000:next
510! ■■■■ main ■■■■
520 let RD1=rnd(8)+1
530 if RD1<4 then gosub $ZG else if RD1<7 then gosub $TK els
e gosub $JR
540 goto 520
550! ■■■■ sub ■■■■
560$ZG:loc 4 to 104,0:for J9=0 to 5:gosub $LD:next J9:loc 4
to 300,0
570 for I1=0 to 3:let RD2=rnd(9)+1:gosub $WD:for I2=0 to RD2
580 let R=rnd(10)+1:let P=rnd(1)+1:on P gosub $RT,$LT
590 next I2,I1:return
600$RT:for I3=1 to R:if RX>2 then let RX=RX-1:gosub $LD:next
 I3
610 return
620$LT:for I4=1 to R:if RX+WD+4<30 then let RX=RX+1:gosub $L
D:next I4
630 return
640$WD:WD1=rnd(6)+4
650 if WD=WD1 then goto $WD else if WD<WD1 then ST=1 else ST
=-1
660 for J=WD to WD1 step ST:if RX+J+4>30 then goto 670 else
WD=J:gosub $LD
670 next J:return
680$LD:CO1=coinc(5,6,8)
690 Print "■":Print cursor(RX,0);chr$(179);chr$(180);rPt$(WD
,"■");chr$(180);chr$(179)
700 Print "■":Print cursor(RX,0);chr$(179);chr$(181);rPt$(WD
,"■");chr$(181);chr$(179)
710 JY=Joy(1):if 2<=JY and JY<=4 then let CX=CX+8 else if 6<
=JY and JY<=8 then let CX=CX-8
720 if JY=1 and CY>35 then CY=CY-4 else if JY=5 and CY<145 t
hen CY=CY+8
730 loc 5 to CX,CY:sg 3,(CY/38)-1,15:CD=CD+5-CY/38
740 CO2=coinc(5,6,8):if CO1<>-1 or CO2<>-1 then goto $BM
750 GX=CX/8:GY=CY/8
760 GR(0)=rcrt(GX,GY):GR(1)=rcrt(GX+1,GY)
770 GR(2)=rcrt(GX,GY+1):GR(3)=rcrt(GX+1,GY+1)
780 for J1=0 to 3:if GR(J1)<>128 then goto $BM else next J1
790 return
800$TK:if RX+14>30 then return else if WD<>10 then WD1=10:go
sub 650
810 loc 6 to 104,0:for I9=0 to 5:gosub $LD:next I9:loc 6 to
300,0:TC=rnd(3)+1
```

```
820 TC=rnd(3)+1
830 for J5=0 to 5:gosub $LD:next J5
840 for J4=1 to TC:TK1=rnd(5)+1:TK2=rnd(4)+1:gosub $LD
850 for J2=1 to TK2:TK3=(RX+2+rnd(8))*8:loc J2+5 to TK3,-20:
move J2+5 to(RX+2+rnd(8))*8,200,rnd(1)+1
860 gosub $LD:next J2
870 for J3=1 to 10:gosub $LD:next J3,J4
880 return
890$ED:sg 3,,0:Print "■■":if HS<CD then HS=CD
900 Print cursor(12,5);"TIME ";time$
910 Print cursor(12,7);"YOUR SCORE ";CD:Print cursor(12,9);"
HIGH SCORE ";HS
920 Print cursor(12,12);"PLAY AGAIN ? (y/n)"
930 Z$=inkey$:if Z$<>"y" and Z$<>"n" then goto 930 else if Z
$="y" then goto 40 else if Z$="n" then end
940$BM
950 for K=3 to 0 step-1:sg 3,6,15:for K1=0 to 400:next:next
960 for K=3 to 0 step-1:sg 3,5,15:for K1=0 to 700:next:next
970 for K=15 to 0 step-1:sg 3,5,K:for K1=0 to 500:next:next
980 sg 3,3,0:for J6=0 to 10:loc 1 to CX,CY:for J7=0 to 200:n
ext J7:loc 1 to 200,200
990 loc 2 to CX,CY:for J7=0 to 200:next J7:loc 2 to 200,200:
next J6
1000 goto $ED
1010$JR:loc 3 to 104,0:for J8=0 to 5:gosub $LD:next J8:loc 3
 to 300,0:JR1=rnd(20)+10
1020 for I5=0 to JR1:for I6=0 to rnd(2)+1:gosub $LD:next I6:
Print cursor(rnd(WD-1)+2+RX,0);"*":next I5
1030 return
```

Changes for operating from keyboard instead of joypads

```
25 console 0,0
710 JY1=inp(&36):JY2=inp(&35):if JY1=64 then let CX=CX+8 els
e if JY1=32 then let CX=CX-8
720 if JY1=4 and CY>35 then CY=CY-4 else if JY2=32 and CY<14
5 then CY=CY+8
```

DIGITAL-ANALOG CLOCK

BASIC-G program :

This demonstration program is a digital-analog clock produced in BASIC-G. It also has an alarm.

Contents:

This program features a designed analog clock in the center of the display, and above it the present time and preset alarm time in digital readout.

There are two methods of shutting off the alarm buzzer: with the simple method the alarm will repeat itself five minutes after being shut off the first time. To halt the alarm completely it is necessary to employ a slightly complicated procedure — so in order to do it you have to be pretty much awake.

Directions:

When you RUN the program, the prompt PRESENT TIME HH : MM : SS will appear. Here you should enter the present 24-hour time in the order hours, minutes and seconds. For example, if the present time is 7:58:07 pm, you must enter 19 : 58 : 07. Don't forget to insert a colon between the numbers for hours, minutes, etc.

After the time has been set, the prompt SET ALARM ? will appear. If you want to set the alarm, type Y; if an alarm setting is not needed, type N. Here the display will go off for a moment and the clock will appear.

If you type Y, the prompt ALARM HH : MM will appear. Here, enter the time at which you want the alarm to sound. (Remember this is a 24-hour clock.) To enter 6:30 am for example, enter "06 : 30". It is not possible to set the alarm precision to seconds.

After setting the alarm, the display will go off for a moment, after which the clock will appear.

When reaching the preset alarm time the buzzer will sound. To turn it off, type C. The alarm will sound once again after five minutes. After that however, you must type the five letters C L E A R. (We made it a little more difficult to make sure you are fully awake.)

If you type C L E A R before the alarm sounds, press the SPACE bar. You can then change the preset alarm time. Even if you had typed N when setting the alarm in the beginning, you can hit the space bar and set the alarm.

Program:

The M5 incorporates a function known as INP which turns any key on the keyboard into a numeric figure. This is utilized to recognize when the five letters in CLEAR have been pressed. In the INP, C represents 4, L — 16, E = 4, A — 1, and R — 8. When you type them together, the value changes slightly to return to 41. As long as 41 is not entered, the alarm will not shut off, so you *must* type all five keys simultaneously to shut off the alarm.

It also uses the commands for graphics under BASIC-G and the ON ALARM or ON EVENT messages.

If you write on the display using graphics under BASIC-G, the display cannot produce circles or straight lines concurrently with characters. At that point in this program the output is produced in the form of numbers. These are different from ordinary numbers.

The display uses the ON ALARM message. When you enter the time in ALARM$ and TIME$ together, this orders the program to skip to GOSUB, and in this program this skipping causes the alarm to sound. This makes the alarm buzzer sound at the specified time.

The ON EVENT message is used after five minutes or when the alarm sounds even when the alarm is stoped by typing C.

Line number 600 EVENT, will, in accordance with 180, 18000 (18000=60=60=) skip to the ON __ Event message GOSUB five minutes later. At this point the alarm sounds once again, and when stopped by typing C or when returning to line number 660, this time it sounds again after three seconds. (180 = 60 =).

[LIST OF VARIABLES]

A$:	Present time
B$:	Alarm time
M1:	Angle of hands
XM, YM:	X, Y coordinates of hands
HI:	Angle of mark which displays time
XH, YH:	X, Y coordinates of mark which displays time
S:	Seconds taken from time $ (=A$)
S1:	Angle of mark which displays seconds
X, Y:	X, Y coordinates of second display mark
M:	Minutes taken from time $

[PROGRAM MAP]

10~80:	Enter present time, alarm time
90~450:	Produce display

460～520: Clock movement
530～650: Alarm subroutine
600～730: Repeat alarm
740～790: Subroutine for hand movement
800～850: Subroutine to remove long hand at 0 minutes
Subroutine to remove heart at 12 o'clock
860～870: Subroutine to set new alarm time

PROGRAM LIST

```
10rem CLOCK-G
20 input "■■■PRESENT TIME HH:MM:SS ";A$
30 input "■■SET ALARM?";Z$
40 if Z$<>"y" and Z$<>"n" then goto 30
50 if Z$="n" then goto 80 else alarm on
60 input "↑ALARM          HH:MM     ";B$
70 alarm$=B$
80 time$=A$
90 print "■■■■■■■■":fcol 8:ginit 51,1
100 view 1,0,31,21
110 scod 15,225:scol 15,8
120 gmove 125,95
130 for J=1 to 3:for I=0 to 255:stchr rpt$(16,"0")to I,J:nex
t I,J
140 restore:for I=48 to 58:read C$:stchr C$ to I,1:next I
150 mag 0:for J=0 to 11:stchr "6CFEFFFEFE7C3810" to 36+J:sco
d J,36+J:scol J,13:next J
160 data"007e42420042427e"
170 data"0010101000101010"
180 data"007e02023c40407e"
190 data"007e02023c02027e"
200 data"004242423c020202"
210 data"007e40403c02027e"
220 data"007e40403c42427e"
230 data"007e4242000202 02"
240 data"007e42423c42427e"
250 data"007e42423c02027e"
260 data"0000101000101000"
270 Print cursor(0,0);"      "
280 Print cursor(24,7);"           "
290 Print cursor(3,8);" ";cursor(3,16);" "
300 if Z$="n" then goto 320
310 Print cursor(12,2);B$+":00"
320 circle 65,,30
330 gmove 88,0
340 draw 88,25:draw 168,25:draw 168,0
350 fcol 5:Paint 10,10,8
360 stchr rpt$(16,"0")to 32,2
370 stchr rpt$(16,"0")to 32,3
380 Print cursor(0,8);"1111";cursor(3,16);"v"
390 for M0=0 to val(mid$(time$,4,2))
400 M1=270+6*M0:XM=125+cos(35,M1):YM=95+sin(35,M1)
410 gmove 125,95:draw XM,YM:next M0
420 for H0=0 to val(left$(time$,2))mod 12
430 H1=270+30*H0:XH=125+cos(55,H1):YH=95+sin(55,H1)
440 loc@H0 to XH,YH:next H0
450 Print "■"
460 Print cursor(12,0);time$
```

Advanced creative level :

MAKE YOUR OWN ANIMATED CARTOONS

Use the 32 sprites featured in the M5 to make patterns.

With either 8×8 or 16×16 dot patterns, the sprites can be made to move rapidly on the display. When two sprites overlap, the M5 is designed to function so that the one with higher priority is displayed. This it makes it easy to express the image of depth.

Let's begin by using the sprite to make a character pattern.

Apple Character Pattern:

(1) Design of an apple inside the 8×8 measure. (Fig. 1)
(2) Producing binary data using 1 for filled-in areas, 0 for blank areas. (Fig. 2)
(3) Separating binary data into the upper (7~4) and lower (3~0) four sections. (Fig. 3)
(4) Refer to table 1 on how to convert binary data to 16-bit data. (Fig. 4)
(5) Align the 16-bit data for the apple character pattern horizontally starting from above. the data thus become "082A7F7F7F7F3E1C".
(6) Assign the character pattern to a figure code.
STCHR "082AF7F7F7F3EC1C" to figure code (0~255).
This is how the 8×8 dot character pattern is set.

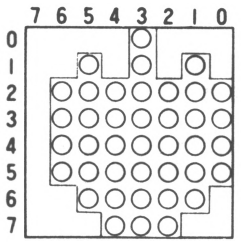


Fig. 1

Fig. 2

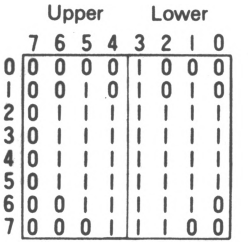


Fig. 3

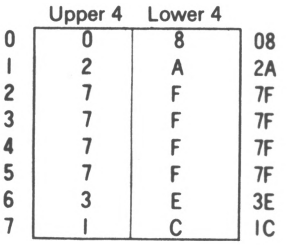


Fig. 4

Binary	Hex	Binary	Hex
0 0 0 0	0	1 0 0 0	8
0 0 0 1	1	1 0 0 1	9
0 0 1 0	2	1 0 1 0	A
0 0 1 1	3	1 0 1 1	B
0 1 0 0	4	1 1 0 0	C
0 1 0 1	5	1 1 0 1	D
0 1 1 0	6	1 1 1 0	E
0 1 1 1	7	1 1 1 1	F

Table 1 Binary ←→ Hex Conversion

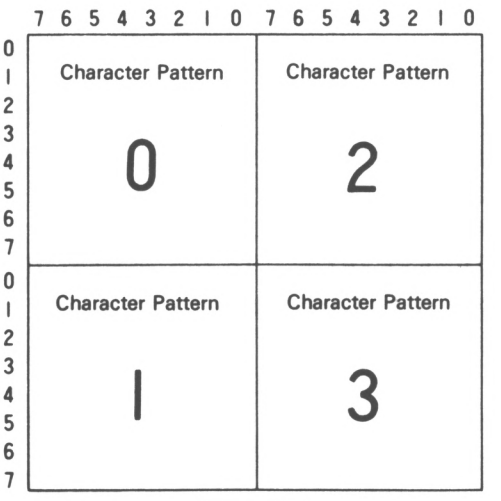


Fig. 5

16×16 Dot Sprite:

You can produce a 16×16 dot sprite by using four character patterns.

(1) Make the design for the 16×16 dot figure just as you did for the 8×8 dot one.
(2) Determine the binary and 16-bit data for each of the four patterns.
(3) Perform the following for character patterns 0×3, using the STCHR command to assign the figure code.

CHARACTER PATTERN 0
STCHR "16-bit data" TO FIGURE CODE (a multiple of 4)
CHARACTER PATTERN 1
STCHR "16-bit data" TO FIGURE CODE (a multiple of 4 + 1)
CHARACTER PATTERN 2
STCHR "16-bit data" TO FIGURE CODE (a multiple of 4 + 2)
CHARACTER PATTERN 3
STCHR "16-bit data" TO FIGURE CODE (a multiple of 4 + 3)

Setting the Sprite:

(1) Determine the size
MAG 0 (1 character pattern = 8×8 dots)
MAG 1 (1 character pattern = 16×16 dots) — twice the number of dots
MAG 2 (4 character pattern = 16×16 dots)
MAG 3 (4 character pattern = 32×32 dots) — twice the number of dots
(2) Use the SCOD command to assign a figure code to the sprite.
SCOD N,M
N (sprite number; 0~31)
M (figure code; for MAG 0, MAGI — 0~225, for MAG 2, MAG 3 — four times that figure.
(3) Use the SCOL command to color the sprite.
SCOL N, M
N (sprite number; 0~31)
M (color code; 0~15)
(4) Set the sprite position on the display by using the LOC command.
LOC N TO X, Y
N (sprite number; 0~31)
X (X coordinate; dot 0~225)
Y (Y coordinate; dot 0~191)

When moving the sprites under BASIC-I, the X, Y values by the LOC command are moved by using the FOR~NEXT message.

Demonstration Program:

Using the STCHR and sprite commands we've just covered, let's produce some animated computer cartoons.

These cartoons will give images such as a boat on the waves, rising and

falling and moving from left to right, while a sunrise takes place in the background. At the same time, you can make clouds move from right to left, intermittently obscuring the sun.

[PROGRAM MAP]

10～20: Set display mode, clear display, move to G-II mode, move to standard mode
30～60: Enter character pattern in figure codes 40～71
70～90: Color background character
100: Set characters on display
110: Set size of sprites (32×32 dots)
120～140: Set figure codes for sprites (0～7)
160～190: Set color for sprites (0～12)
200: Draw the sun
220～340: Move waves (240), boat (250～260), clouds (270～280), and sun (290～320)
350～380: Wave pattern
390～420: Mast pattern
430～460: Boat pattern
470～500: Cloud pattern
510～540: Cloud pattern
550～580: Sunshine pattern
590～620: Sunshine pattern
630～660: Sun pattern

PROGRAM LIST

```
10 Print "■■0":cls
20 view 1,0,30,23
30 for I=40 to 71
40 read A$
50 stchr A$ to I
60 next I
70 stchr "4444444444444444" to 255,6
80 stchr "7777777777777777" to 255,5
90 stchr "7777777777777777" to 255,4
100 cls 255
110 mag 3
120 for I=0 to 7
130 scod I,40+I*4
140 next I
150 let L=0
160 scol 0,4:scol 1,11
170 scol 3,14:scol 2,14
180 scol 5,8:scol 4,14
190 scol 6,9:scol 7,12
200 loc 5,to 20,30:loc 7 to 20,30
210 let J=6
220 for I=0 to 281
230 for K=1 to 420:next K
240 loc 0 to I+rnd(4)-3,128+rnd(4)
250 loc 1 to I,96
260 loc 2 to I,125
270 loc 3 to 192-I,30
```

```
280 loc 4 to 223-I,30
290 loc J to 1,500
300 let J=5+I mod 2
310 scol 7,11-I mod 4
320 scol J,8+I mod 4
330 loc J to 20,30
340 next I
350 data "048ec+++++++++++"
360 data "++++++++++++++++"
370 data "183c3e++++++++++"
380 data "++++++++++++++++"
390 data "000001030?0f1f3f"
400 data "7+++++++++0101?1"
410 data "0000000000000000"
420 data "0000000000000000"
430 data "++++7+3+1+000000"
440 data "0000000000000000"
450 data "fffefc+8+00000000"
460 data "0000000000000000"
470 data "0000030+1+3+7fff"
480 data "++++7+0+07070300"
490 data "00030+5+++++++++"
500 data "fffffffff9f0c000"
510 data "0081c7c+++++++++"
520 data "++++++++++++e?000"
530 data "00c0e0fd++++fefe"
540 data "++++++++fe7c3818"
550 data "0000111010008040"
560 data "2000000038000102"
570 data "4080001c00000004"
580 data "0201000808880000"
590 data "0002042420200000"
600 data "8040200001060000"
610 data "0030400000040201"
620 data "0004040420408000"
630 data "0000000003070f0f"
640 data "0f0f070300000000"
650 data "00000000c0e0f0f0"
660 data "f0f0e0c000000000"
```

ENJOY COMPUTERIZED MUSIC

There are two methods by which the M5 produces sounds: with the SML and with the OUT command. Here we'll give you explanations of both.

Method using SML:

SML is an abbreviation for Sord Music Language Interpreter. Working by breaking into the M5's A-80 CTC (Counter Timer Circuit), this is a subroutine in the monitor for playing music. This routine constantly interrupts the CTC channels at a tempo ranging from 1 second to 1 millisecond. The tempo counter lowers count, and each time it reaches zero it is read out. The called-out SML reduces the tone length counter, and when it reaches zero it reads out the next command from the buffer generator and plays it.

The above functions are performed for sound 1 through sound 3. By doing this, envelopes (methods of making sounds — see figure 1) which cannot be specified by means of the OUT command and tempo can easily be set. On the other hand, if the number of items of data are not precisely counted and correctly addressed, the tune will run out of control. In some cases the program you took a lot of trouble to write will be erased, so it is recommended that you exercise caution.

Here, let's review the methods of producing sound.

(1) SETTING THE OCTAVES

In the M5's sound generator, six octaves of musical scales can be produced. The data for the six octaves is as follows; from low to high, &20, &21, &22, &23, &24 and &25.

(2) SETTING MUSICAL SCALE AND LENGTH

The pitch within each octave is determined by referring to the information in table 1. Next, determine the length (4-minute note, 8-minute note, etc.) by referring to table 2.

Setting the musical pitch and length is done once for each note. However, with this method when there are many repeated notes of the

same length, a large portion of data is required. In this case, you should perform the following steps. When the same note repeats itself, just set the length of the first note. To set the length of the next note, it is only necessary to set the pitch. By doing this, the length of the note will be maintained continuously. To make the settings, write in the data which appears after &60 in table 2.

(3) SETTING OF ENVELOPE

There are 8 envelope forms, numbered 0 through 7. These are set by writing &40 ~ &47 respectively.

(4) SETTING OF VOLUME

There are 16 volume levels, from 0 ~ 15. A 0 setting is equivalent to no volume. These settings are made by writing from &30 ~ &3F respectively. (See table 3)

(5) SETTING OF TEMPO

The tempo in the data is set at quarter notes. For example, if you want to make quarter notes, set &70, 60

(6) SETTING THE HOLD TIME

The hold time refers to the time a note of specified length is maintained. Figure 2 shows the nine types — from 0 to 8 — which are set by &50 ~ &58.

Next we will give a detailed explanation of the tune in figure 3 (the opening notes of Lorelei).

Line numbers 100 ~ 190 are the data for the upper part. Line number 100 is the initial setting, entered in the order hold time, envelope and tempo. From the next line, the length and musical interval.

The octave is set by &22. The &88, 8 is "so" eighth note, &88j, 12 is a dotted eighth "so" note, &89, 4 is a sixteenth "la" note, and &88, 8 sets an eighth "so" note. The next, &60, 8, &23, 1 sets the octave for the &23 eighth "do" note. Next, &22, 12 sets the octave for &22 "ti". The length was set by the previously-entered &60, 8 for the eighth note, so that is the length for this note. The next 10 is the "la" note. The length is the same as above.

Note: the volume is only effective when the envelope setting is at 0.

Method using the OUT command:

The OUT command is a command to output the data in the output port.

In the OUT command three tones can be harmonized. That is, there are three sounds produced, which we will refer to as tone 1, tone 2 and tone 3.

To produce a sound, you must perform the OUT command three times. The first time the OUT command is used determines what number tone will be produced and writes in the lower 1-digit data (16-progression) for the step. The second time writes the upper 2-digit step data (16-progression). The third time writes the volume. For data on setting the step, refer to table 4.

For example, to produce tone 1,
1) OUT &20, &82
2) OUT &20, &OD
3) OUT &20, &90

When making tone 2, the &86 in tone one becomes &A6 and the &90 in tone 3 becomes &BO. When producing tone 3, the &86 in tone one becomes &C6 and the &90 in tone 3 becomes &DO.

In general, this may be expressed as:
1) OUT &20, A + B
2) OUT &20, C
3) OUT &20, D + E

A is the value which sets tones 1,2 and 3. For tone 1, this becomes &80, for tone 2, &AO, and for tone 3, &CO.

B is the value as shown in the lower two digits (16-progression) in the same data. D is the value which sets tones 1,2 and 3. For tone 1, this is &90, for tone 2, &BO and for tone 3, &DO.

E is the value which sets the volume. It can be set in 16 steps from &00 to &0F, with the maximum &00. At &0F, no sound is produced.

When 1, 2 determine the step and 3 the volume, the sound will be produced. However, it should be cautioned that this sound will continue until the volume is set to &0F. Therefore the OUT command makes it easy to play a tune.

MUSICAL SCALE	DO	DO ♯, RE ♭	RE	RE ♯, MI ♭	MI	FA
DATA	&81	&82	&83	&84	&85	&86

MUSICAL SCALE	FA ♯, SO ♭	SO	SO ♯, LA ♭	LA	LA ♯, TI ♭,	TI
DATA	&87	&88	&89	& 8 A	& 8 B	& 8 C

Table 1 MUSIC PITCH

NOTE	WHOLE NOTE	𝅗𝅥.	𝅗𝅥	♩.	♩	♪.
DATA	64	48	32	24	16	12

NOTE	♪	𝅘𝅥𝅯.	𝅘𝅥𝅯	𝅘𝅥𝅰.	𝅘𝅥𝅰
DATA	8	6	4	2	1

Table 2 LENGTH

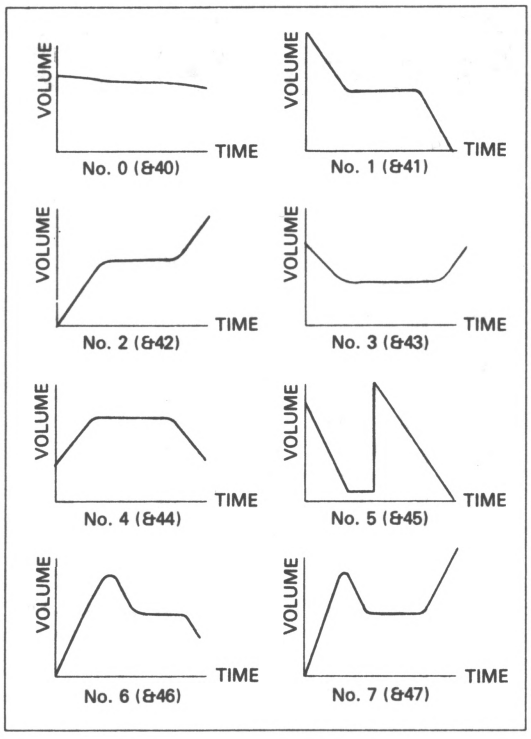


Fig. 1 ENVELOPE

No.	0	1	2	3	4	5	6	7
VOLUME (db)	0	−2	−4	−6	−7	−9	−11	−13
DATA	&30	&31	&32	&33	&34	&35	&36	&37

No.	8	9	10	11	12	13	14	15
VOLUME (db)	−12.5	−14.5	−16.5	−18.5	−19.5	−21.5	−23.5	OFF
DATA	&38	&39	& 3 A	& 3 B	& 3 C	& 3 D	& 3 E	& 3 F

Table 3 VOLUME (db = decibels)

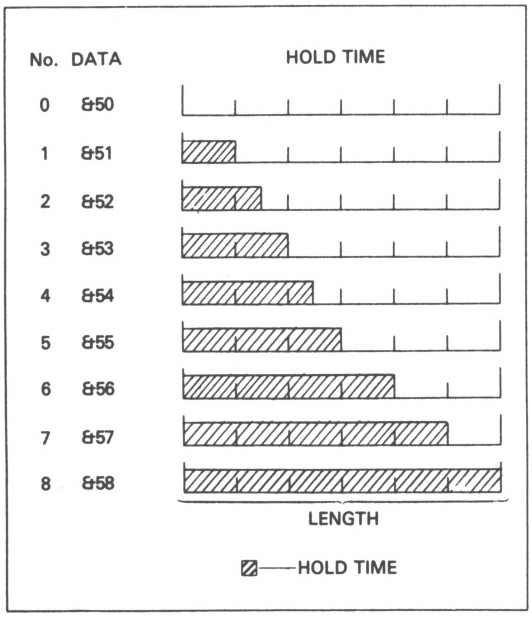


Fig. 2 HOLD TIME

Fig. 3 SCORE (Lorelei)

PITCH	DATA (Hex)	DATA (Decimal)	PITCH	DATA (Hex)	DATA (Decimal)	PITCH	DATA (Hex)	DATA (Decimal)
LA	3F9	1017	RE	17D	381	SO	08F	143
LA♯ , TI♭	3C0	960	RE♯ , MI♭	168	360	SO♯ , LA♭	087	135
TI	38A	906	MI	153	339	LA	07F	127
DO	357	855	FA	140	320	LA♯ , TI♭	078	120
DO♯ , RE♭	327	807	FA♯ , SO♭	12E	302	TI	071	113
RE	2FA	762	SO	11D	285	DO	06B	107
RE♯ , TI♭	2CF	719	SO♯ , LA♭	10D	269	DO♯ , RE♭	065	101
MI	2A7	679	LA	0FE	254	RE	05F	95
FA	281	641	LA♯ , TI♭	0F0	240	RE♯ , MI♭	0SA	90
FA♯ , SO♭	25D	605	TI	0E2	226	MI	055	85
SO	23B	571	DO	0D6	214	FA	050	80
SO♯ , LA♭	21B	539	DO♯ , RE♭	0CA	202	FA♯ , SO♭	04C	76
LA	1FC	508	RE	0BE	190	SO	047	71
LA♯ , TI♭	1E0	480	RE♯ , MI♭	0B4	180	SO♯ , LA♭	043	67
TI	1C5	453	MI	0AA	170	LA	040	64
DO	1AC	428	FA	0A0	160	LA♯ , TI♭	03C	60
DO♯ , RE♭	194	404	FA♯ , SO♭	097	151	TI	039	57

Table 4

PROGRAM LIST

```
10rem lorelei
20 clear 256,&7DFF
30 Print "■when the music stops....run360 "
40 for I=0 to 110
50 read A:Poke &7E00+I,A:next
60 restore 220:for I=0 to 114
70 read A:Poke &7EA0+I,A:next
80 restore 330:for I=0 to 4
90 read A:Poke &7064+I,A:next
100 restore 340:for I=0 to 4
110 read A:Poke &7072+I,A:next
120 data &57,&41,&3f,&70,&36
130 data &22,&88,8,&88,12,&8a,4,&60,8,8,&23,1,&22,12,10
140 data &88,24,&86,16,6,&85,16,5,3,1,3,&85,32,0
150 data 8,&88,12,&8a,4,8,&23,1,&22,12,10
160 data &88,24,&86,16,6,&85,16,5,8,6,3,&81,32
170 data 0,5,&83,12,&85,4,3,8,3,3,&8c,24,&8a,16,10
180 data &88,16,8,7,8,10,&88,32,0,8
190 data &88,12,&8a,4,8,&23,1,&22,12,10,&88,16
200 data &23,5,&83,16,3,&81,16,1,&22,12,10
210 data 12,&23,&81,32,0
220 data &57,&41,&70,&36,&3f,&60,8
230 data &22,8,&85,12,&86,4,5,10,8,6
240 data &85,24,&83,16,&83,8,&81,16,1,&21,12,&22,1,&21,12
250 data &22,&81,24,1,0,8
260 data &85,12,&86,4,5,10,8,6
270 data &85,24,&83,16,3,&81,16,1,&21,12,12,12
280 data &22,&81,24,1,0,1
290 data &21,&8c,12,&8c,4,12,12,12,12,&22,&83,24,&81,16,1
300 data &21,&8c,16,12,&22,&83,16,1,&21,&8c,24,12,0
310 data &22,8,&85,12,&86,4,5,10,8,6
320 data &85,24,&86,16,6,&85,16,5,&88,16,6,&85,24,5,0
330 data &00,&7e,&f0,111,0
340 data &a0,&7e,&f0,115,0
350 end
360 Poke &7067,0:Poke &7068,0
370 Poke &7075,0:Poke &7076,0
```

USING THE MACHINE LANGUAGE MONITOR

All computer operation is performed by what is called "machine language." From the largest mainframe to the smallest microcomputer, every computer performs its functions by machine language. Programs written in BASIC or FORTRAN are also converted to machine language.

What is a machine language monitor?

To understand the basics of a microcomputer, it is necessary to study machine language.

Machine language is directly received and processed in the computer's CPU (Central Processing Unit), and this determines the maximum performance of a computer in terms of its processing speed and access to memory. However, the method by which machine language operates the computer—the so-called binary system of ones and zeros—is difficult for people to understand. Therefore in all computers the machine language is written in assembly language — an easier-to-follow method closer to human language. The assembler writes directly in machine language (it never uses words), but in the case of a small personal computer such as the M5, the assembler cannot gain entry to the memory. Therefore assembly language cannot be entered in this computer. For this reason, programs written in assembly language must first be written out on paper (hand assembly) and then other computer assemblers used to convert to machine language.

Users who wish to attempt such operations must be sure that the assembler is the same as that for the Z-80 used in the M5's CPU (8080 can also be used).

Programs produced in this manner may be entered and used with the M5, but since they are using assembly language, which is more complicated than BASIC, it's no exaggeration to say that the first time, nothing will work right. Probably it will be necessary to remove bugs and repeat operations dozens, even hundreds of times.

At this time, although having a BASIC program using assembly language in the form of a subroutine will enable removal of bugs and opera-

tion, it will still be necessary first to run after changing the data messages and the READ message pointer. When running out of control, using the SAVE message also incorporated in BASIC is extremely troublesome. Another problem is that if the machine language program is halted midway, it is nearly impossible to be able to view the register or memory values.

The machine language monitor is provided to simplify such operations as those mentioned above, and to raise debugging capability. Other microcomputers selling for $1000~$1500 are equipped with a mechanical language monitor in their ROM or on the tape, but in the case of the M5, the ROM is filled with subroutines for assisting BASIC and FALC languages or other machine language programs, so there is no machine language monitor. Consequently, a simple machine language monitor has been written in BASIC.

The reason it was written in BASIC is that it is easy to enter and easy to convert.

Monitor functions:

A conventional monitor consists of memory readout, register readout, program operate/stop and program load/save.

This monitor is familiar with the M5 is interior, so the user can enter the functions he wants, though functions are held to a minimum. The memory is also more than sufficient for the monitor's requirements. The following gives the particulars for monitor use.

Using the monitor:

Enter and operate the program, and the symbol ? appears, signifying that it is awaiting a command. At this time, the display more on the screen must be multicolor. The input mode should be set to the overlay write mode.

From the command-ready status, you may enter any of the nine commands explained in the following paragraphs. If you type any other characters other than the nine commands, PRINT CHR$ is displayed. If a mode is to be changed during operation, you can make use of this. If something other than a command is entered while in the command-ready position, the monitor will keep and those characters will be displayed. It will then await the next command but no other operations may be performed.

Here are the contents of the nine commands:

1 D MEMORY DUMP

Press the D key, and the prompt START ADDRESS will appear. By entering a 4-digit 16-progression number, the contents of the memory will display eight characters at a time on line 1. By pressing the shift key at the

right of the keyboard, the dump will stop. When you want to halt dump, press the RETURN key.

2) P DUMP MEMORY (PRINTER)

This operates the printer in the same manner as the D command, above.

3) W WRITE TO MEMORY

This begins with the START ADDRESS prompt as for the D command. The address is displayed on the left side. Next the address contents are displayed, and the cursor flashes on and off at the top line. Here, enter a two-digit 16-progression number. After the data is written into the memory, the monitor will once again read the data from the memory and display it on the screen to verify the entry. If the present address is in a place without ROM or memory, a different value will be displayed after entry to indicate that it was not written in.

When entering this input, pressing the SPACE bar will not write the memory, and the next address memory will be entered.

During input, pressing the RETURN key will return you to the command-ready status.

4) L LOAD MEMORY DATA

Pressing the L key will prompt FILE NAME. Here you may enter up to nine characters. Then press RETURN. If you make any mistakes, use the DEL key. Afterwards, the prompt LOAD OR NOT ? will appear. To load, type Y. If loading is not desired, type any other character.

5) S SAVE MEMORY

After typing S the prompts START ADDRESS and END ADDRESS appear. Enter a 4-digit 16-progression figure. Pressing RETURN takes you back to the command-ready mode. Next, as for the previous L command, type in the file name, and decide whether or not to save. Save operation will then begin.

6) J JUMP CALL

After typing J, the START ADDRESS will appear as with previous commands, and the JUMP CALL selection is made. The monitor will recall the value of the previous jump call (only A, B, C, D, E, H, L flags) and will call the entered start address.

When returning to the monitor from the machine language program, type the RET command. The monitor will return to the command-ready mode after storing the registered value.

7) A REFER TO REGISTER

When A is entered, A, B, C, D, E, H and L flag registers will appear respectively, displaying the value of the previous jump call. Display then returns to the command-ready mode.

8) M MAIN MEMORY

After entering the M command, the commands D, P, L and S are all performed relative to the main memory.

9) V VIDEO MEMORY

After entering the V command, the commands D, P, L and S are performed relative to the video memory.

Precautions for monitor use:

- In the monitor, the RESET and HALF keys are not controlled. Pressing them may cause functions to stop.
- In the L and S commands, from the time Y is pressed until the load-save is completed, the RESET key cannot be used for any operation except stopping the monitor.
- When starting the monitor, the register value is not cleared.
- Only the A, B, C, D, E, H and L flags can be stored in the register.
- The user memory which can be employed to run the monitor in this form is from 7F00 to 7FCB.

Explanation of program:

First use CLEAR to open an area to house the machine language subroutine. Then use POKE to send the subroutine to the memory. There are two machine language subroutines: one calls a one-character subroutine with a flashing cursor in the M5's ROM; the other is for load-store when operating the jump call.

Next, in the command-ready mode, process jump is done for each command. However, the MV command will change only the variables and return to command-ready position.

Since other command processing is easily understood from looking at the program, it will not be explained here. However, in the D and P commands, since it is very complicated to write the same processing twice, use the variable P. Then PRINT # P, then CRT, then PRINT # 2 to print out on printer to enable processing in a signal operation.

[LIST OF VARIABLES]

S: Address value (Mainly start address)
D: Data and key code, yes/no subroutine flag I: Y
B: Memory flag 0: Main 1: Video
P: Output flag 0: Television 1: Printer
R: Input subroutine call number 1: Return key is pressed
D$: 16-progression 2-digit data code, file name
N: Input subroutine call number 0 File name/ 1 Start address/ 2 End address/ 3 6-progression, two-digit
E: Number counter for data written on the line, end address
I: Work
F: Flags in input subroutine 0 Character row/ 1 16-progression four-digit/ 2 16-character two-digit
NI: Number of input characters in input subroutine 2, 4, 9

[PROGRAM MAP]

10~20:	Initial setting, machine language subroutine light
30~70:	Command-ready, processing jump according to command input
60:	M, M command processing
80~110:	D, P command processing
120~170:	W command processing
180~200:	S command processing
210~220:	L command processing
230~250:	J command processing
260~270:	U command processing
280:	Start address input subroutine
290~370:	Input subroutine
380~390:	Yes/No input subroutine
400:	Single character input subroutine
410~420:	Memory lead and 16-progression 2-digit character row arrangement subroutine
430~450:	Machine language subroutine data
460:	Input subroutine data
7FOO~7FCB:	Free user area
7FCC~7FD2:	Single character input subroutine
7FD3~7FF6:	Jump call sub
7FF7~7FFF:	Register housing memory

PROGRAM LIST

```
10 clear 256,&7EFF
20 for S=&7FCC to &7FF6:read D:Poke S,D:next:let B=0
30 Print:Print
40 Print ">";:9osub 400:Print chr$(D):if D=100 then let P=0:
9oto 80 else if D=112 then let P=2:9oto 80
50 if D=119 then 9oto 120 else if D=115 then 9oto 180 else i
f D=108 then 9oto 210
60 if D=118 then let B=1:9oto 30 else if D=109 then let B=0:
9oto 30 else if D=106 then 9oto 230
70 if D=97 then 9oto 260 else Print "■";:9oto 40
80 9osub 280:if R then 9oto 30
90 Print#P,:Print#P,hex$(S);" ";:for I=1 to 8:9osub 410:Prin
t#P,D$;:let S=S+1:next
100 if inkey$=chr$(13)then 9oto 30 else if inP(&30)=8 then 9
oto 100
110 9oto 90
120 9osub 280:if R then 9oto 30
130 Print hex$(S);" ";:let E=8
140 9osub 410:Print D$;"←←";:let N=3:9osub 290
150 if R then 9oto 30 else if D>=0 then if B then vPoke S,D
else Poke S,D
160 9osub 410:Print rPt$((Peek(&70A7)mod 3)+1,"←");D$;
170 let S=S+1:let E=E-1:if E then 9oto 140 else Print:9oto 1
30
180 9osub 280:if R then 9oto 30 else let N=2:9osub 290:if R
then 9oto 30
190 let E=D:let N=0:9osub 290:if len(D$)then 9osub 380:if D
then save D$,S,D,B
200 9oto 30
```

```
210 let N=0:gosub 290:gosub 380:if D then if len(D$)then old
 D$ else old
220 goto 30
230 gosub 280:if R then goto 30
240 gosub 380:if D then Poke &7FE4,S mod 256:Poke &7FE5,S/25
6:call &7FD3
250 goto 30
260 Print "      A  F  B  C  D  E  H  LIX    ";
270 for S=&7FF7 to &7FFE:Print " ";right$(hex$(Peek(S)),2);:
next:goto 30
280 let N=1:gosub 290:let S=D:return
290 restore 460:for I=0 to N:read F,N1,D$:next:if F-2 then P
rint D$;" ";
300 let D$="":let R=0:let N=0
310 call &7FCC:let D=Peek(&7FFF):if D=13 then let R=1:Print:
return
320 if F=2 and D=32 then let D=-1:return
330 if D=8 and N then let N=N-1:Print "← ←";:if N then let D
$=left$(D$,N):goto 310 else goto 300
340 if F and val("&1"+chr$(D))=1 or D<32 then Print "■";:got
o 310
350 Print chr$(D);:let D$=D$+chr$(D):let N=N+1:if N<N1 then
goto 310
360 if F then let D=val("&"+D$)
370 if F-2 then Print:return else return
380 Print "ok ? ";:gosub 400:if D>32 then Print chr$(D);
390 if D=121 then let D=1:return else let D=0:return
400 call &7FCC:let D=Peek(&7FFF):return
410 if B then let D=vPeek(S)else let D=Peek(S)
420 let D$=" "+right$(hex$(D),2):return
430 data &cd,&45,&08,&32,&ff,&7f,&c9,&21,&f7,&7f,&06,&04,&56
,&23,&5e,&23
440 data &d5,&10,&f9,&e1,&d1,&c1,&f1,&cd,&2e,&00,&e5,&d5,&c5
,&f5,&21,&f7
450 data &7f,&06,&04,&d1,&72,&23,&73,&23,&10,&f9,&c9
460 data 0,9,file name      -,1,4,start address -,1,4,end    a
ddress -,2,2,
```

SORD
SORD COMPUTER CORPORATION
Kyobashi K-1 Bldg., 7-12 Yaesu 2-chome, Chuo-ku, Tokyo 104, Japan.

US$4.50