

NIGHT MISSION



MG

NIGHT MISSION

Book Written By

Craig G. Miller

Program Written By

Mike S. McCue
&
Craig G. Miller

Copyright 1985 by Millers Graphics

The Information contained in this book is subject to change without notice.

Millers Graphics shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material or product described by this manual.

This book contains information protected by copyright. All rights are reserved. No part of this manual may be photocopied or reproduced in any form without prior written consent from Millers Graphics.

Night Mission Software and Associated Book
(Printed in the United States of America)

Copyright 1985

by

Millers Graphics
1475 W. Cypress Ave.
San Dimas, CA 91773

Dedicated to the
memory of
Patrick McCue

TABLE OF CONTENTS

Loading Night Mission into Your System	1
The Game Documentation	4
The Power of AND	8
The Characters	15
The Variables	20
The Program Flow	21
10 - 210 Start & Title Screen	23
220 - 290 Restart & Game Over	33
300 - 400 Rockets or Planes Screen	35
410 - 560 Enemy chopper Screen	41
570 - 730 Ship Screen	47
740 - 1120 Tank Screen	53
1130 - 1170 Laser Fire	67
Call Peek and Call Load	68
APPENDIX - Program Listings	
A - Cassette Joystick	71
B - Disk 'LOAD'er	74
C - Disk Keyboard	75
D - Disk Joystick	78
E - 8 x 8 Character Diagrams	81
F - 16 x 16 Character Diagrams	83

LOADING NIGHT MISSION INTO YOUR SYSTEM

The cassette that comes with the book contains 5 programs on it. Two on the Cassette side for cassette based systems and three on the Diskette side for disk based systems. The following pages will guide you through the steps necessary to load the proper version in your system.

CASSETTE BASED - Joystick version

1. The first program on the Cassette side of your cassette is the Joystick version of Night Mission. Load the cassette so that the Cassette side is ready to be played, type in **RUN "CS1"** and press **ENTER**.
2. Follow the instructions on your screen for loading the program.
3. The documentation for the game is in the next section of this book.
4. It is recommended that you run this program a few times to get familiar with it before you read through the "PROGRAM FLOW" section.

CASSETTE BASED - Keyboard version

1. The Keyboard version of Night Mission is the second program on the Cassette side of your cassette. You must advance the tape past the first program, which is the Joystick version, to the start (long continues tone) of the Keyboard version. You may have to unplug the **PAUSE** and **EARPHONE** jacks from your cassette before you press **PLAY** in order to allow you to advance the tape and hear the tones.
2. After you have found the beginning of the Keyboard version (second program) on the Cassette side plug your **PAUSE** and **EARPHONE** jacks back into your recorder. Now type in **RUN "CS1"** and press **ENTER**.
3. **DO NOT REWIND THE CASSETTE** as it says on the screen just press **ENTER**. Now you can follow the rest of the instructions on the screen for loading the program.
4. Now would be a good time to record a backup copy of this program on a blank cassette. By doing this you will not have to find the beginning of the keyboard version each time you load it.
5. The documentation for the game is in the next section of this book.
6. It is recommended that you run this program a few times to get familiar with it before you read through the "PROGRAM FLOW" section.

DISKETTE BASED - Loader, Joystick & Keyboard versions_____

1. Install a blank, freshly initialized diskette into disk drive 1 in your system and then select EXTENDED BASIC.
2. The Disk based versions of Night Mission are on the Diskette side of the cassette tape. This program was too large to load into a Disk based system without Memory Expansion, so we divided it into sections to allow it to fit into a 16K system. Even if you have Memory Expansion the program would be in a INT/VAR254 file type instead of a PROGRAM type and it would take longer to load. The first program on the Diskette side is the Loader program. This program initializes the characters and screen, displays the title screen and generates a menu. The menu allows you to select the Joystick or Keyboard version and then loads and runs that version. Load the cassette so that the Diskette side is ready to be played, type in **OLD CS1** and then press ENTER.
3. Follow the instructions on your screen for loading the program.
4. After the program has been loaded, type in **SAVE DSK1.LOAD** and press ENTER. You have now saved the first part of the disk based program. By saving this part of the program under the name of LOAD it will automatically be loaded and run whenever this diskette is placed into drive 1 and you select EXTENDED BASIC.
5. Now we are ready to load the Joystick version of Night Mission. The disk based version differs from the cassette based version in that the character definitions have been removed from the main body of the program and placed into the LOAD program. By doing this we were able to free up a lot of RAM and keep the file in a PROGRAM format, which allows it to be run without Expansion Memory. Type in **OLD CS1** and press ENTER.
6. DO NOT REWIND THE CASSETTE as it says on the screen just press ENTER. Now you can follow the rest of the instructions on the screen for loading the second part of the disk based program.
7. After this portion of the program has been loaded type in **SAVE DSK1.NMJOY** and press ENTER. It is important that you save this portion under the name of NMJOY because the LOAD program will search for a program named NMJOY to load and run from drive 1 when you select the Joystick version from the menu.

DISKETTE BASED - Continued

8. Now once again type in **OLD CS1** and press ENTER to load the Keyboard version. DO NOT REWIND THE CASSETTE as it says on your screen just press ENTER. Now you can follow the rest of the instructions for loading the third and final part of the disk based program.
9. After this portion of the program has been loaded type in **SAVE DSK1.NMKEY** and press ENTER. It is important that you save this portion under the name of NMKEY because the LOAD program will search for a program named NMKEY to load and run from drive 1 when you select the Keyboard version from the menu.
10. When everything is saved to disk you can type in RUN "DSK1.LOAD" to start up the Night Mission Game. You could also type in BYE and then select EXTENDED BASIC from the menu and Night Mission will automatically load and run.
11. The documentation for the game is in the next section of this book.
12. It is recommended that you run this program a few times to get familiar with it before you read through the "PROGRAM FLOW" section.

THE GAME DOCUMENTATION

YOUR MISSION:

To rescue as many men as possible from the hostile enemy territory and bring them safely to the ship waiting offshore.

THE HAZARDS:

This is a covert action. You will be working entirely on your own. If you should get caught we will disallow any knowledge of your actions. The territory is fully protected by enemy tanks, rockets, jet planes and choppers. Your chopper can only hold 5 men at a time so you will have to make a number of trips to the ship. The first trip will be fairly easy but on the following trips they will be aware of your presence and they will try harder to stop you. Good Luck.

TANK SCREEN:

This screen has an enemy tank rolling along the bottom of the screen just above the closed missile silos. Your helicopter will come into play from the left hand edge of the screen, above the hills, on a starry moonlit night.

When you land on the ground a man will come running out from the left hand edge of the screen toward the helicopter. When he reaches your chopper he will climb aboard and your chopper will lift off. After you have picked up your fifth man you will move on to the next screen.

Watch out for the enemy's tank since it can shoot down your chopper, but you can also shoot back to protect yourself and the man on the ground. When your chopper has landed on the ground don't let it get too close or it will destroy the chopper and the man.

On this screen, the keyboard version uses the arrows keys (ESDX) to move your chopper in the appropriate direction and the V key to fire your lasers while you are in the air. The E or up key will also lift your chopper off the ground after it has landed.

In the joystick version all four directions are active on the number one joystick and they will move your chopper in the appropriate direction. The diagonals are also active but they react as a left or right motion. The fire button will fire your lasers while you are in the air.

LEVELS:

The order in which the other screens come up is determined by the number of successful trips you have made. Each time you successfully deliver five men to the ship the level of difficulty will increase. The difficulty is increased by adding more screens for you to go through before reaching the ship and by increasing the speed of the tank, planes, and rockets. The order of screens on each level is as follows:

- 1st Level** - Tank screen - Ship screen
- 2nd Level** - Tank screen - Enemy rockets or planes (randomly chosen) - Ship screen
- 3rd and up** - Tank screen - Enemy rockets or planes - Enemy choppers - Ship screen

SHIP SCREEN:

On this screen there is our offshore ship waiting for you to deliver the men in your chopper. The ship will be moving at a random speed from left to right. Your job is to GENTLY land your chopper on the landing platform on the rear of the ship. If you land too hard or land on any other part of the ship it will sink and all hands will be lost and so will all the points you have accumulated for the men you previously delivered. If you touch the water, your helicopter will sink and you will not receive any points for a safe delivery. Unlike the First screen in which you have one forward and one reverse speed for your helicopter, on this screen you have three forward and three reverse speeds.

(TIP: lower your chopper until it is about level with the landing platform and then ease into the ship from the rear.)

In the keyboard version the arrow keys (ESDX) are active. E will make you fly up and X will fly you down. S and D will change your forward or reverse velocity.

In the joystick version all four directions are active. The diagonals are also active but they react as a left or right joystick motion.

ENEMY PLANES:

On this screen a number of enemy planes will come on screen from the right hand edge and travel from right to left at various speeds. Your mission is to maneuver around the planes and fly your chopper from the left hand edge of the screen to the right hand edge. If you make it safely there you will move on to the next screen.

In the keyboard version the arrow keys (ESDX) are active. E will fly you upwards and X will fly you down. D will increase your forward motion slightly and S will bring you to a stop.

In the joystick version up and down will fly you in the appropriate direction. Right will increase your forward motion slightly and left will bring you to a stop.

ENEMY ROCKETS:

On this screen the rocket silos will open up and the enemy will launch a continuous barrage of missiles into the air. Your mission is to safely fly your chopper through this barrage from the right hand edge of the screen to the left hand edge. If your mission is a success you will move on to the next screen.

The keyboard and joysticks react the same as they do on the Enemy planes screen.

ENEMY CHOPPERS:

On this screen your chopper will appear from the top left corner of the screen and will move in a downward motion. The enemy chopper will appear from either the top or bottom right hand corner of the screen and will move to line up with you. Your mission is to destroy an unknown number (3-9) of the enemy choppers before they destroy you. In this screen your chopper can move up, down or stay stationary. You are not allowed to move forward or backwards.

In the keyboard version the arrow keys (ESDX) are active and the V key will allow you to shoot at the enemy. The E key will fly you up and the X key will fly you down. The S and D keys will stop your motion.

In the joystick version all four directions are active. Pushing the stick up or down will fly you in the appropriate direction. Pushing the stick right or left will stop your motion. The fire button will allow you to shoot at the enemy chopper.

POINTS:

MEN - Each man you safely deliver to the ship will add 500 points to your score. Total of 2500 points each time you land on the ship.

TANKS - The points for destroying the enemy tanks is determined by your height above the ground. You will receive 500 points for each tank you destroy from the top of the screen and the points will diminish the closer your chopper is to the ground.

ENEMY - You will receive 250 points for each enemy chopper you destroy.
CHOPPERS

BONUS (Free Plays):

For every 10,000 points you rack up you will receive one extra helicopter which will be awarded to you after you have lost your first five choppers.

SCREEN SCORE DISPLAY:

A BBB CCCCCCCCC DDD

A = Men in your helicopter.
B = Total men safely delivered to the ship.
C = Total points accumulated.
D = Remaining helicopters. (5 to start)

Good Luck. If you find the game to be too hard you could use the information contained in "The Program Flow" to help you make it easier.

THE POWER OF AND

One of the most powerful and versatile functions in Extended Basic and many other languages is the logical expression of AND. Unfortunately it has also been one of the least explained functions. In the TI Extended Basic manual they devote all of two and a half confusing pages to the logical expressions of AND, OR, XOR and NOT. This is also true for most of the other computer's basic manuals.

So a few years back we decided to find out all we could on this mysterious unexplained function. After much research and a number of books on Boolean Algebra and Boolean Logic we ended up more confused than when we began since most of the material did not deal with the subject on a computer level.

Well it was now time to tackle this subject on a purely experimental basis with Extended Basic. After many weeks of testing, trying, discovering and failing the answers finally became understandable and explainable in plain english. So now we would like to share our understanding of the use of AND on direct numbers with you.

You are all probably familiar with the use of AND with two relational expressions such as: IF A=1 AND B=2 THEN..... so we will not discuss this use. We will instead discuss the many uses of AND on direct numbers such as: C=C+1 AND 7 or C=C AND 32767 or IF C AND 1 THEN..... Used in this form AND works fast and usually reduces the amount of code, or bytes, in your program.

To start with lets look at a few of the possible uses of AND in your programs:

1. Very good for auto-reset counters that never increment beyond a certain value.
2. It can be used to easily determine if a number is Odd or Even.
3. Excellent for small pseudo random numbers when sprites are in motion and your program uses CALL POSITION.
4. Very good for conserving on the total number of variables in your program when you are using the variables as flags or condition indicators.
5. It can be used to easily round off a floating point number into an integer.
6. Easily converts Lower case to Upper case or visa versa.

Lets look at this function on a Binary level and then we will discuss the rules and some examples for its use. When you use AND on direct numbers you are actually comparing bits at the binary level. If a certain bit is on in B and the same bit is on in C then, when you AND these two (PRINT B AND C) that bit will be on in the result. If a certain bit is on in B but off in C it will be off in the result. So the number one basic rule is:

ON Bits that match in the two numbers will be ON in the result.
 and
 OFF Bits in either of the two numbers will be off in the result.

It might help to think of the use of AND as a filter that only allows ON bits that match up to pass through it for the result.

		binary	
Examples:	4 AND 7	4 = 00000100	
		7 = 00000111	
	4 AND 7 Result	= 00000100	or 4
	9 AND 7	9 = 00001001	
		7 = 00000111	
	9 AND 7 Result	= 00000001	or 1

From the above example we can see that when a number is ANDED to 7 the result can never be greater than 7 since the higher value bits are off (xxxxx111). This is true for any possible ANDED value within the valid range. On the TI in Extended Basic the Valid range is -32768 through 32767. Lets take a look at the value of each bit in a 16 bit 2's compliment binary number.

Bit No.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	sgn	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

These values can also be thought of as Binary Powers. If you raise 2 to the bit number the result is the Value displayed above.

Examples: $2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^{12} = 4096$ etc.

so if bits 0,1 & 2 are on (00000111) then the decimal number is: 1+2+4 or 7

If bits 0,2,3 & 5 are on (00101101) then it equals: 1+4+8+32 or 45

When Binary numbers are used in 2's compliment form like in Extended Basic the highest order bit, or in this case bit 15, is used as the SIGN bit. This bit tells the computer that the number is a positive value when it is off or that it is a negative value when it is on. We will talk more about 2's compliment Binary numbers a little later. So with these items in mind lets look a little deeper into this subject.

As we saw earlier the decimal value of 7 is 00000111 in binary and this will mask out any values higher than 7. This is also true for any other value in the valid range but some value serve this purpose better than others. We have found the values that best serve this purpose are the ones that are 1 less than a Binary Power.

Examples: 8-1 = 7 16 - 1 = 15 4096 - 1 = 4095 etc.

7 = 0000000000000111 15 = 0000000000001111
4095 = 0000111111111111

As you can see when you subtract 1 from a Binary power ALL of the lower bits are turned ON. This allows any value up to this number to pass through the filter but never a larger number. With this in mind lets look at a few examples:

1. Very good for auto-reset counters that never increment beyond a certain value.

Many times in an Extended Basic program you may need a counter that counts up to a certain value and resets itself. In normal Extended Basic code this may look like this:

```
B=B+1 :: IF B=8 THEN B=0
```

By using AND with a number that is 1 less than a Binary Power we can reduce the code and speed up the program by using:

```
B=B+1 AND 7
```

Since 7 is one less than a Binary Power and since we are using AND in this statement it will only allow the counter to count to 7. When B+1 becomes 8 the AND function will automatically reset B back to 0 because higher values are filtered or masked out. This type of counter can be used with any Binary Power minus 1 (ie: 1,3,7,15,31 16383 or 32767) and it will never count past your AND value.

2. It can be used to easily determine if a number is Odd or Even.

Using this same principal we can use AND to determine if a number is Odd or Even. An Odd number will ALWAYS have Binary bit 0 set and an Even number will never have bit 0 set since the bits represent powers of 2. With this in mind we could replace the following Extended Basic code of:

```
IF B/2-INT(B/2) THEN ..... the number is odd
```

with

```
IF B AND 1 THEN ..... the number is odd
```

Note: The statement IF B AND 1 returns the same true false condition as IF B AND 1<>0. This is also the same for IF C or IF C-1 etc. This type of statement is TRUE whenever the result of the test or variable IS NOT ZERO.

3. Excellent for small pseudo random numbers when sprites are in motion and your program uses CALL POSITION.

Whenever you have sprites in motion and your program uses CALL POSITION you can use AND to rapidly generate small pseudo random numbers especially if the player has control over the sprites movement. Since sprites are only allowed to have row and column positions from 1 through 256 you can easily AND their position with any value less than 255 to obtain a pseudo random number. Example: CALL POSITION(#1,X,Y):: IF Y AND 1 THEN By trying different AND values you can get a variety of random numbers or decisions from a sprites position. Here is a little Extended Basic program that displays the different possible combinations and also shows you some of the value patterns generated by AND:

```
100 CALL CLEAR :: INPUT "Start at what AND value ":V ::  
INPUT "Loop from zero through ":L  
  
110 FOR V=V TO 255 :: PRINT  
: : " V=";V :: FOR I=0 TO L  
:: PRINT USING "### AND ###  
= ###":I,V,I AND V  
  
120 CALL KEY(0,K,S):: IF S THEN GOSUB 140  
  
130 NEXT I :: NEXT V :: END  
  
140 CALL KEY(0,K,S):: IF S=1 THEN RETURN ELSE 140
```

After running this program for a little while you should notice that Even numbers and Odd numbers that are 1 less than a Binary Power generate nice regular patterns. Other Odd numbers also have set patterns within a group of numbers but they are not as regular and they do not automatically increase as the value increases.

So by using a value of 1 for AND you can get a random 1 or 0 for any sprite position. If you use a value of 3 you can get random values of 0,1,2 or 3. A value of 6 will return random numbers of 0,2,4 or 6 for any sprite position. The value of 8 returns 0 when the sprite is located in an Odd numbered Graphics column (1-32) and 8 when the sprite is in an Even numbered Graphics column. This method of generating small pseudo random numbers in Extended Basic is much faster than the RND function.

Note: After playing with AND in the above program change the two references to AND in line 110 to XOR and then change them to OR to see how these logical expressions work with direct numbers.

-
4. Very good for conserving on the total number of variables in your program when you are using the variables as flags or condition indicators.

When you use variables as Flags or Condition indicators in your program you can easily replace up to 15 variables with just one Flag variable. A Flag is a fancy term for a variable or bit that tells the program a certain condition exists or doesn't exist (on or off). For example, this condition could be anything from OUTPUT TO PRINTER to the fact that a certain For-Next Loop was just executed so the program needs to do something else before the loop is executed again. Lets also say that you are using a flag variable to tell your program that the user has a Color or B&W monitor. So, for this example lets say that normally you would set B=1 to send the Output to the printer, you would set C=1 when something else must be executed before the loop is executed again and you would set D=1 for a Color monitor. The parts of your program that may use these Flag variables may look something like this:

```
160 IF D=1 THEN CALL COLOR(x,x,x) ELSE CALL COLOR(y,y,y)
.
.
230 IF B=1 THEN PRINT #3:A$,B$
240 PRINT A$,BS
.
.
400 IF C=1 THEN RETURN
410 FOR I=1 TO 20 :: ..... :: NEXT I :: RETURN
```

As you can see we are using 3 variables as flags in this program example. By using AND, OR, XOR and NOT you can reduce this to one variable and just test to see if certain bits are On or Off for your flags. This method also allows you to test multiple conditions or flag bits in one operation.

First lets map out the values of those bits again:

Bit No.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	sgn	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

You can turn ON any of the 15 bits (0-14) with the OR function without worrying about the status of the other bits. Example: F=F OR 4. This turns on bit number 2 without affecting the other bits.

You can test any bit with the AND function. Example: IF F AND 4 THEN ... bit number 2 is on.

You can turn OFF a bit with the AND and NOT functions. Example F=F AND NOT 4 (which is the same as F=F AND -5) This turns off bit number 2 without affecting the other bits.

And, you can turn ON a bit that is OFF and turn OFF a bit that is ON (toggle the bit to its opposite state without knowing the state) with the XOR function. Example: F=F XOR 4. This will turn OFF bit 2 if it is ON and it will turn ON bit 2 if it is OFF.

With this in mind lets put it to work in the previous example program but first we will map out our Flag variable F.

OUTPUT = Set equals Output to printer
 NOLOOP = Set equals Don't execute loop
 COLOR = Set equals Color Monitor

Bit no.	15 --- 3	2	1	0
Value	-----	4	2	1
Condition	not used	COLOR	NOLOOP	OUTPUT
Old variable		D	C	B

And now lets apply them to the previous program example but now we only need one flag variable.

```

160 IF F AND 4 THEN CALL COLOR(x,x,x) ELSE CALL COLOR(y,y,y)
.
.
230 IF F AND 1 THEN PRINT #3:A$,B$
240 PRINT A$,B$
.
.
400 IF F AND 2 THEN RETURN
410 FOR I=1 TO 20 :: ..... :: NEXT I :: RETURN

```

In your program the IF F AND 4 statement uses the same number of bytes as IF F=1. However, since we have eliminated 2 other variables (flags C & D) and replaced B with F we have reduced the running size of the program. Also, the fewer the variables in your program the faster it runs! If your program had a few more flag variables in it they too could be eliminated by assigning their flag condition to one of the unused bits in F.

When you want to test for more than one condition at the same time simply add up the values for the bits you want to test and perform your AND and compare the result to the total value. Example:

```

IF F AND 6=6 THEN .... both the COLOR and NOLOP bits are set ....
ELSE .... one or none of the bits are set.

```

This type of multiple testing replaces the normal code of:
 IF D=1 AND C=1 THEN both conditions are set.

To replace the normal code of IF D=1 OR C=1 add up the bits again but do not compare the result of AND to the total. Example:

```

IF F AND 6 THEN .... both or either bit could be set ....
ELSE .... neither bit is set.

```

-
5. It can be used to easily round off a floating point number into an integer.

Since the logical expressions of AND, OR XOR and NOT work on binary integers you can use AND to round a floating point number into an integer. Normally your code to perform this may look like:

```
B=INT(B+.5)
```

By using AND you can save 3 bytes and slightly speed up your program by replacing the above code with:

```
B=B AND-1
```

This forces the computer do the rounding internally for you so it can operate on an integer. When you use this method your floating point value must be in the range of -32768.5000 through 32767.4999.

We used the AND value of -1 because -1 is the same as NOT 0 or ALL BITS ON. With all the bits turned on, your AND or filter allows all bits to pass through it and just return the result of the computer's internal rounding.

You can carry this one step further and use it to round for dollars and cents or any number of decimal places you want. For example, the normal code used to round to 2 decimal places of:

```
B=INT(B*100+.5)/100
```

Could be replaced with:

```
B=(B*100 AND-1)/100
```

But this makes the valid range limit -327.6850 through 327.6749 since we are multiplying by 100 before the AND is executed. Also, this method only saves 1 byte and with all the other floating point operations in this statement the time saved is very slight. So, we recommend that you only use this method of rounding for obtaining integers and not for rounding to decimal places.

6. Easily converts Lower case to Upper case or visa versa.

To convert lower case or uppercase to uppercase value just AND it with 95.

Example: 97 AND 95 returns 65 65 AND 95 also returns 65.
 a A A A

To convert uppercase or lower case to lower case value just OR it with 32.

Example: 65 OR 32 returns 97 97 OR 32 also returns 97.
 A a a a

To exchange case, upper to lower & lower to upper, just XOR it with 32.

Example: 65 XOR 32 returns 97 97 XOR 32 returns 65.
 A a a A




One practical application for this is with CALL KEY statements. Example:






```
CALL KEY(0,K,S):: IF K AND 95=89 THEN .... Y or y was pressed.
```







However, on the 99/4A, TI gave us a CALL KEY that only returns uppercase values so, CALL KEY(3,K,S):: IF K=89 THEN works the same as the above example and K will equal 89 whether Y or y was pressed.











We hope that the previous pages have helped shed some light on the potential uses for AND, OR, XOR and NOT. With a little experimentation in your programs you may find many more areas that these logical expressions can be used to help speed up your programs and save a few bytes of code.



THE CHARACTERS



Character Set 1	Character Colors				
32	Space character	Tank	R&P	Ship	EC
33	 Solid block for foreground	11,1	11,1	5,1	1,1
34	 Top of the water				
35	 Closed and Open Missile silo				
36	36-39 Tank fire - sprite	Sprite Colors			
37				
38	9	--	--	--
39				



Character Set 2	Character Colors				
40	 Solid block for hills	Tank	R&P	Ship	EC
41	(not used)				
42	(not used)				
43	(not used)	11,1	11,1	1,1	1,1
44	 Hill				
45	 Hill				
46	 Hill				
47	 Hill				



Character Set 3	Character Colors				
48	 Chars 48 thru 57 are	Tank	R&P	Ship	EC
49	redefined as slanted	15,1	15,1	15,1	15,1
50	numerals.				
51					
52					
53					
54					
55					

Character Set 4			Character Colors			
56		8	Tank	R&P	Ship	EC
57		9	15,1	15,1	15,1	15,1
58		(not used)				
59		(not used)				
60		60-63 Man running, Planes & Rockets, Ship and Enemy Chopper sprites	Sprite Colors			
61			2	rnd	16	6
62				3-10		
63						
60			Planes screen			
61						
62						
63						
60			Rockets screen			
61						
62						
63						
60			Ship screen			
61						
62						
63						
60			Enemy Chopper screen			
61						
62						
63						
Character Set 5			Sprite Colors			
64		64-67 Man running, Ship & Enemy Chopper fire sprites	Tank	R&P	Ship	EC
65			2	--	16	7 or 4
66						
67						
64			Ship screen			
65						
66						
67						
64			Enemy Chopper screen			
65						
66						
67						

Character Set 5 Continued		Sprite Colors				
68		68-71 Man running and Ship sprites	Tank	R&P	Ship	EC
69		2	--	16	--
70					
71					
68						
69						
70		Ship screen				
71						

Character Set 6		Sprite Colors				
72		72-75 Man running sprite	Tank	R&P	Ship	EC
73		2	--	--	--
74					
75					
76		76-79 Man running sprite	Sprite Colors			
77		2	--	--	--
78					
79					

Character Set 7		Sprite Colors				
80		80-83 Man shot or climbing into chopper - sprite	Tank	R&P	Ship	EC
81		2	--	--	--
82					
83					
84		84-87 First pattern tank or chopper explosion - sprite	Sprite Colors			
85		Tank = 2	--	11	
86		Our Chopper = 16	or rnd 3-10		
87					

Character Set 8		Sprite Colors				
88		88-91 Second pattern tank or chopper explosion - sprite	Tank	R&P	Ship	EC
89		see char 84	--	11	15
90					
91					
92		92-95 Third pattern tank or chopper explosion - sprite	Sprite Colors			
93		see char 84	--	11	15
94					
95					

Character Set 9

96 . 96-99 Fourth pattern
97 . . tank or chopper
98 . . explosion - sprite
99

100 . 100-103 Fifth pattern
101 . . tank or chopper
102 . . explosion - sprite
103 . . and the Star char


Sprite Colors


Tank R&P Ship EC
see char 84 -- 11
15

Sprite Colors

see char 84 -- 11
(Star char - 15,1 all scrns)

Character Set 10

104  104-107 Backwards flight
105 chopper - sprite
106 and the letters
107 PLAY

108  108-111 Level flight
109 chopper - sprite
110 and the letters
111 GIN


Character Colors


Tank R&P Ship EC
End of Game - Letters
twinkle - 7,9,16,10 on 1

Sprite Colors

13 13 13 13

Character Set 11

112  112-115 Forward flight
113 chopper - sprite
114
115

116  116-119 Chopper on the ground
117 or landed on the ship
118 sprite
119


Sprite Colors


Tank R&P Ship EC
13 13 13 13

Sprite Colors

13 13 13 13

Character Set 12

120  120-123 Moon
121
122
123

124  124-127 First pattern
125 man waiving - sprite
126
127


Character Colors

Tank R&P Ship EC
15,1 15,1 15,1 15,1


Sprite Colors

2 -- -- --

Character Set 13


128  128-131 Second pattern
129 man waiving - sprite
130
131


Sprite Colors
Tank R&P Ship EC
2 -- -- --

132  M (CTRL D) —MG—


Character Colors
12,1 12,1 12,1 1,1

133  G (CTRL E)


134  Line — (CTRL F)

135  Line ____ (CTRL G)

Character Set 14

136  136-139 Laser fire - sprite
137
138
139

Sprite Colors
Tank R&P Ship EC
7 -- -- --

140  140-143 Tank - sprite
141
142
143

Sprite Colors
2 -- -- --

THE VARIABLES

Permanent String Variables

A\$ = Lower line of the hills & foreground with missile silos
B\$ = Image for the DISPLAY AT USING for Score
W\$ = Water screen for the ship
E\$ = Eight zeros for CALL CHAR statements
T\$ = Twelve zeros for CALL CHAR statements
S\$ = Sixteen zeros for CALL CHAR statements

Permanent Numeric Variables

B = Value for bonus (incremented by 10,000's)
SC = Total current Score
HP = Number for men in your chopper (incremented by 1's)
SP = Total men delivered to the ship (incremented by 5's)
Z = Remaining number of your choppers (decremented by 1's)
K = Current level you have reached (incremented by 1's)
V = Current column velocity of your chopper

Temporary Numeric Variables

X = Carries your choppers dot row position.
Y = Carries your choppers dot column position. It is also used in conjunction with the AND function to generate random numbers. ie: $SGN(Y-YY)*((Y \text{ AND } 6)+4+K)$. It is also used in FOR NEXT loops as the control variable.
Y1 = Carries the dot column position of the man running to your chopper. It is also used to carry the random number of enemy choppers for that screen.
YY = Carries the tanks dot column position. It is also used for the sound value on the rockets and planes screen and as the dot row of the enemy chopper.
T = The all around catch all. FOR NEXT loops. CALL COINC flag. Chopper just coming into play flag. The Key variable from CALL KEY or CALL JOYST etc.

NIGHT MISSION - THE PROGRAM FLOW

There are many ways to program the computer to do essentially the same task. Some ways are faster and/or they use up less RAM than others. What we hope to accomplish is to show you some of the tricks and methods we have found that work quite well for reducing program space and improving the execution speed of your programs.

The Night Mission program took approximately seven months to write with its' many revisions and improvements. Our main objective with this program was to write a multiple screen game that would be fun for all ages to play. It would also have to incorporate extensive use of sprites, animation, graphics, sound effects and of course lots of color and action. Our challenge was to do it all in Extended BASIC, without the use of expansion memory, and to maintain a fast response time to joystick and keyboard inputs. Well we are proud to say that once again the TI 99/4(A) computer has shown us how powerful it can be with the right program structure and proper use of its' built in subprograms! So now lets see how 'THE HOME COMPUTER' reacts to the various program statements and how it all came together to form an Award Winning Program.

The following pages contain the documentation for the Night Mission keyboard version. On the Left hand page you will find the program listings for the lines that are documented on the right hand page. Some of the lines have a label in front of them to indicate that there is a GOTO or GOSUB instruction somewhere else in the program that references that line.

The documentation was set up to match each program line so that each new paragraph starts a new program line. Also all IFs and ELSE IFs have been indented to make it easier to follow the program flow and, remember IFs without an ELSE always have an automatic ELSE (or continuation) on the next program line.

After you have completed the documentation try changing different things in the program to see what affect it has. When you are able to make successful changes you know that you fully understand that portion of the program. This is how we first learned to program since books on programming TI Extended Basic were nonexistent at the time.

We hope that you find this documentation simple to follow and complete enough to allow you to use different routines and subroutines from Night Mission in your own programs. And now - On With The Program.

NIGHT MISSION LINE NUMBER MAP

10- 210 Initialization and title screen.
220- 290 Restart, Game Over & Bonus play.
300- 400 Rockets and Enemy Planes initialization and game loop.
410- 560 Enemy Choppers initialization and game loop.
570- 730 Ship screen initialization and game loop.
740- 800 Tank screen initialization.
810-1020 Tank screen routines.
1030-1120 Tank screen game loop.
1130-1170 Laser fire routine.



```
START      10 CALL CLEAR :: CALL MAGNIF
           Y(3):: CALL SCREEN(2):: GOTO
           30 :: A$,B$,W$,B :: CALL KE
           Y :: CALL SOUND :: CALL PEEK
           :: CALL HCHAR :: CALL VCHAR
```

```
20 X,Y,Y1,YY,K,V,Z,HP,SP,SC
   :: CALL POSITION :: CALL PAT
   TERN :: CALL SPRITE :: CALL
   DELSPRITE :: CALL COINC :: C
   ALL MOTION :: CALL LOCATE
```

```
START1     30 E$="00000000" :: T$=E$&"0
           000" :: S$=E$&E$ :: CALL CHA
           R(132,"6152524CCC00000080808
           0FF809C84FC"&E$&"FF"&T$&"FF"
           ):: FOR T=1 TO 8 :: CALL COL
           OR(T,15,1):: NEXT T :: !@P-
```

```
40 CALL CHAR(47,"3C4299A1A19
           9423C",33,RPT$("FF",8)&"0000
           002011B3FFFFFFFFFFFF817E7E81
           ",112,"A00802103979FDF1F070
           1080701"&E$&"80D084C1F0FCF2E
           2E2F2FC9CFA1C")
```

N I G H T M I S S I O N



MG

BY

MIKE MC CUE & CRAIG MILLER

© 1985

MILLERS GRAPHICS
1475 W CYPRESS AVE
SAN DIMAS CA 91773

-
- START** Clear the screen, set the sprite magnification to 3 (four
10 regular sized characters per sprite), change the screen color
to black, and GOTO START1 (jump over the following items that
were placed here for a rapid Pre-Scan).
- 20 These items are also here for a rapid Pre-Scan.
- START1** Setup E,T & S strings as eight, twelve and sixteen zeros for
30 the CALL CHARs to follow, define chars 132 (ctrl d), 133 (ctrl
e), 134 (ctrl f) and 135 (ctrl g) as the MG logo and
accompanying lines, set the upper case character colors to grey
on transparent and turn off the Pre-Scan (10P-).
- 40 Define char 47 as the Copyright symbol for the Title screen,
char 33 as a solid foreground character for all screens but the
Enemy Chopper screen, char 34 as the top of the water for the
Ship screen, char 35 as the closed missile silos for the Tank
screen, and chars 112, 113, 114 and 115 for our forward flying
chopper sprite.

MG

TITLE_SCRN

```
50 DISPLAY AT(5,3):"N I G H  
T M I S S I O N": : :RPT$(  
"?",13)&"??"&RPT$("?",13)::  
CALL COLOR(13,12,1):: GOSUB  
240 :: CALL SPRITE(#1,112,1  
3,46,25,0,11)
```

```
60 DISPLAY AT(12,14):"BY": :  
" MIKE MC CUE & CRAIG MILLER  
": : : :TAB(12);"/ 1983": :"  
MILLERS GRAPHICS": "  
1475 W CYPRESS AVE": "  
SAN DIMAS CA 91773"
```

```
70 CALL CHAR(136,"9048241209  
0402010"&E$&S$&"080402090482  
41209",96,"10000040"&E$&"80"  
&E$&"2"&S$&"104000001000008"  
,100,"0000008"&S$&"000004"&S  
$&"2"&T$&"04")
```

```
80 CALL CHAR(120,"030E3F3C7F  
F7FFFFFF6F7F7F33360F03E0583C  
F8F8E8F0B0F0F0F07078E81CFC",  
124,"0008080907"&RPT$("01",1  
1)&"00C0C0C0E0E0E0E0E0C04040  
404040E0")
```

```
90 A$=RPT$(" ",22):: CALL CH  
AR(128,"0202040503"&RPT$("01  
",11)&"00C0C0C0E0E0E0E0E0C04  
040404040E0",140,T$&"0103073  
F6AAAAA7F"&S$&"80C0E0FCAA9A  
AFC")
```

```
100 A$=A$&"/ e e /,  
e .(- e .((-  
e /..(((, e e .(((  
- .((((((- ./ .((((  
(, e.(((((((((- .((-,(((((  
((-/.(((((((((((,"
```



```
110 GOSUB 240 :: B$=RPT$("!"  
,28)&RPT$("!!#",9)&RPT$("!"  
29):: W$=RPT$("""",28)&RPT$("  
!"",84)
```

```
120 CALL CHAR(92,"0008000020  
0000004000000001"&E$&"80"&E$&  
"208000002001",116,"00005501  
0F1E183020301E1F0F1320300000  
5580F078180C040C78F8F0C8040C  
")
```



```

CLEAR_TITLE 130 CALL CLEAR :: FOR T=1 TO
             13 :: CALL COLOR(T,1,1):: N
             EXT T :: FOR X=1 TO 30 :: RA
             NDOMIZE :: CALL PEEK(-31808,
             T,Y):: CALL HCHAR(T/18+1,Y*.
             12+1,101):: NEXT X

             140 GOSUB 240 :: CALL CHAR(8
             8,"000004001000000020000008"
             &E$&"001"&E$&"410000004001",
             56,"1F1122223E4444F81F21213E
             02040408")

             150 CALL CHAR(48,"1F21214242
             8484F801010202040408081F0101
             023C20407E3F0103021C0408F8",
             52,"1111223E020404081F202040
             7C0408F8102020407E82827C3F01
             020408102040")

             160 CALL CHAR(80,E$&"0003040
             1010101010202040600000018F8E
             0F8C0C0C070101018",84,"00000
             00200080000100002"&T$&"00200
             00008200000082")

             170 CALL CHAR(72,"0000010107
             05030101010102020203C0C0C0
             E0E0F8C0C0C02020101020406"):
             : CALL HCHAR(20,1,34,32)

             180 CALL CHAR(36,"140A200D1A
             21641A241A0814080800080"&S$,
             76,"000001030303030101010100
             00070400C0C0C0E0E0E0F0C0C040
             40C0C0A080C"):: CALL HCHAR(2
             1,1,33,128)

             190 DISPLAY AT(10,1):"e h i
             j k e j l j m n e k n e" :: CA
             LL CHAR(40,RPT$("FF",8),44,"
             00008082C2C7F7FF8080C0E0E4FC
             FEFF0103232B7F7FFFFFF"&E$&"08
             28A9FD")

```

CLEAR_TITLE

130 Clear off the Title screen, turn off the colors for all the character sets that make up the screen display, and generate 30 randomly placed stars on the screen using the double random number generator from the Smart Programming Guide for Sprites. These stars will end up between rows 1 thru 15 and columns 1 thru 32.

140 GOSUB CHOP_SOUND (keep the chopper sound going), define chars 88 thru 91 as the second pattern for the blow up routines, and redefine chars 56 and 57, the 8 and 9 characters, as slanted numerals.

150 Redefine chars 48 thru 51 and chars 52 thru 55, the 0 thru 7 characters, as slanted numerals.

160 Define chars 80 thru 83 as the man when he is climbing into the chopper or when he is shot by the tank, and chars 84 thru 87 as the first pattern for the blow up routines.

170 Define chars 72 thru 75 as the forth pattern for the running man, and place the top of the water character across the screen at row 20 to fill the edges of the screen.

180 Define chars 36 thru 39 as the sprite character that the tank fires at our chopper, chars 76 thru 79 as the fifth pattern for the running man, and fill the foreground portion of the screen with the solid block character. This HCHAR and the one in the previous line are used to fill the edges of the screen that are not written to with a DISPLAY or PRINT statement.

190 Place the characters that make up the words "PLAY AGAIN Y N" on the 10 row of the screen. There are also a few stars (e) displayed with these characters. The characters are not seen because their color is turned off. These characters are defined as the backwards and level flying choppers until the end of the game. When the game ends they are redefined into the letters and their color is turned on. Define char 40 as a solid block for the hills and chars 44 thru 47 as the top of the hills for the tank screen.

```
200 DISPLAY AT(2,1):RPT$("?"  
,13)&"??"&RPT$("?",13):: DIS  
PLAY AT(3,24):"xz" :: DISPLA  
Y AT(4,24):"y{" :: DISPLAY A  
T(15,1):A$:B$ :: A$=SEG$(A$,  
141,28)&B$
```

```
210 GOSUB 240 :: B$="e # ###  
# e##### e### e" :: CAL  
L DELSPRITE(ALL):: CALL COLO  
R(3,15,1,4,15,1,9,11,1,2,11,  
1,1,11,1,12,15,1,13,12,1)
```



```

RESTART      220 CALL COLOR(10,1,1):: CAL
              L CHAR(108,E$&"55000061E1FFF
              FFF00000003"&E$&"5540EOF8E4E
              2E1F1FF7C45FE")

              230 K=1 :: Z=5 :: HP,SP,SC,B
              =0 :: CALL CHAR(104,E$&"0104
              10400103CFFFFFF786001020820C0
              707CE2E1E1F3FEFCC54658E")::
              GOTO 740

CHOP_SOUND   240 CALL SOUND(-4250,-4,1,11
              0,30,110,30,200,30):: RETURN

GAME_OVER    250 CALL KEY(3,T,Y):: Z=INT(
              (SC-B)/10000):: FOR T=1 TO Z
              :: CALL SOUND(200,770,4,777
              ,6):: DISPLAY AT(1,24):USING
              "###":T :: NEXT T

              260 IF Z THEN B=B+Z*10000 ::
              GOTO 740 ELSE CALL CHAR(108
              ,"FF81BFAOAFB981FFFF81E71818
              E781FFE7B5B5BDBDADADE7")

              270 CALL CHAR(104,"FF81BD81B
              FAOAOEEOAOAOAOA0BF81FFFF81B
              DBD81BDA5E7E7A5BD81E7181818"
              ):: CALL SPRITE(#1,112,13,87
              ,1,0,12)

GETKEY       280 CALL COLOR(10,7,1,10,9,1
              ,10,16,1,10,6,1):: CALL KEY(
              0,T,T)

              290 IF T=89 THEN 220 ELSE IF
              T=78 THEN CALL DELSPRITE(AL
              L):: CALL VCHAR(1,1,32,768):
              : END ELSE 280

```

RESTART Turn off the colors for the PLAY AGAIN letters, in case the
 220 program came here from a game over condition, and define chars
 108 thru 111 as the level flying chopper (the game_over routine
 defines these characters as the letters GIN for the PLAY AGAIN
 Y N message at the end of the game).

230 Set the level (K) to 1, set the number of our choppers
 remaining (Z) to 5, clear out the number of men in the chopper
 (HP) - the number of men delivered to the ship (SP) - the score
 (SC) and the bonus points (B), and define the chars 104 thru
 107 as the backwards flight chopper (the game_over routine
 defines these characters as the letters PLAY for the PLAY AGAIN
 Y N message at the end of the game). GOTO TANK_INIT (set up the
 tank screen).

CHOP_SOUND This subroutine is used to keep the chopper sound going during
 240 initialization.

GAME_OVER Initialize keyboard for caps only, calculate the number of
 250 bonus choppers that should be awarded, if the current score
 minus the previous bonus is less than 10000 this calculation
 will set Z equal to zero and the following FOR NEXT loop will
 not execute, if Z is greater than zero (1,2,3 etc.) the loop
 will execute that many times and display the adding of choppers
 in the scoring portion of the screen with a bell sound each
 time it increments the display.

260 IF there are bonus choppers (Z<>0) THEN add to the variable
 (B) that keeps track of the bonuses awarded so far, and
 GOTO TANK_INIT (set up the tank screen since the game
 isn't over yet).
 ELSE redefine the characters used for the GIN letters in the
 PLAY AGAIN Y N message.

270 Redefine the characters used for the PLAY letters in the PLAY
 AGAIN Y N message, bring out our forward flight chopper and
 make it fly across the screen.

GETKEY Turn on the colors for the letters and make them twinkle, and
 280 scan for a key press.

290 IF a YES key is pressed THEN GOTO RESTART (start the game
 over).
 ELSE IF a NO key is pressed THEN delete all the sprites and
 wipe off the screen and END the program.
 ELSE since no key was pressed GOTO GETKEY (twinkle the
 letters and scan for a key).

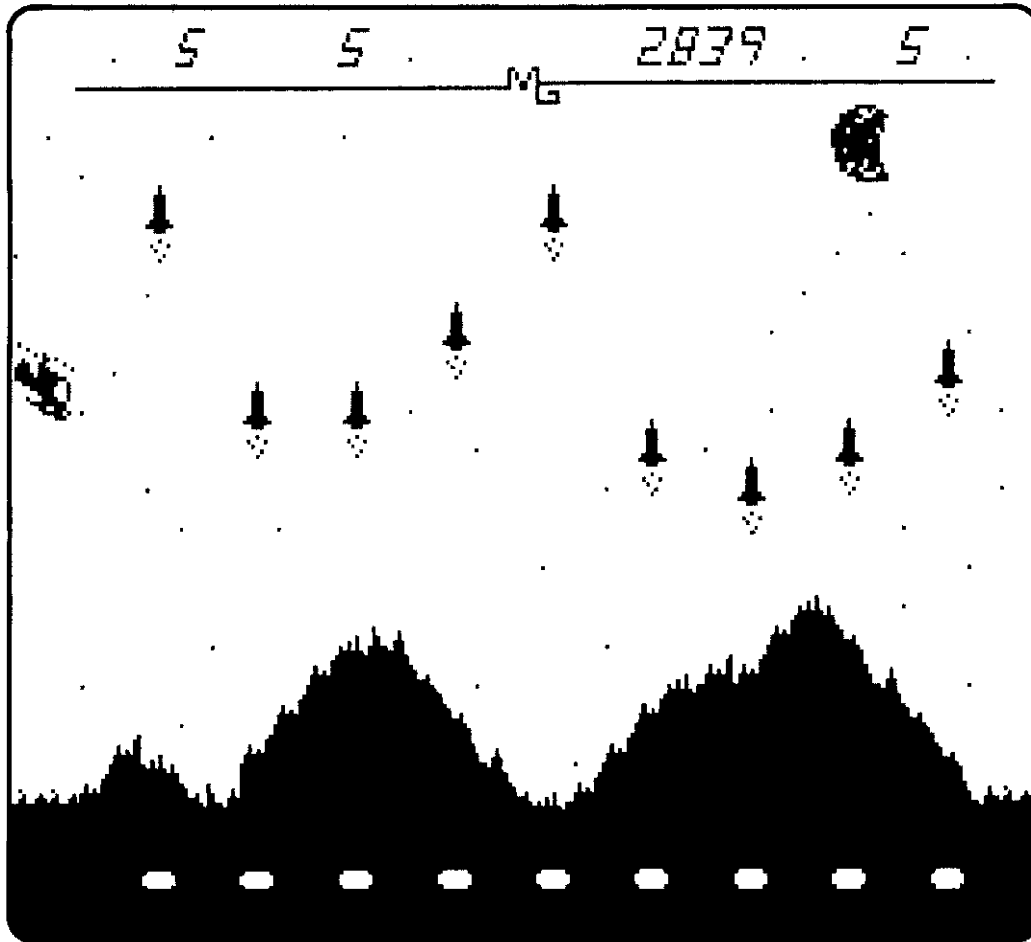
R&P_INIT

```
300 V=8 :: IF K<2 THEN 570 E
LSE IF Y AND 1 THEN 340 ELSE
  CALL CHAR(60,"08081C1C1C1C1
C3E7F1C0008221004080"&S$)
```

```
310 YY=600 :: CALL DELSPRITE
(ALL):: FOR T=2 TO 5 :: CALL
LOCATE(#T,1,T*17,#T+4,177,T
*17):: NEXT T
```

```
320 CALL SOUND(-350,-7,6,110
,5):: CALL CHAR(35,"FFFFFFF
81000081"):: CALL SOUND(4250
,-8,4,110,27,115,28,YY,30)
```

```
330 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
  CALL SPRITE(#T,60,(X AND 7)
+3,177,T*24-208,-X/8-3-K,0):
: NEXT T :: GOTO 360
```



```

R&P_INIT      Set V equal to 8 for use with the blow_up_chop subroutine, in
300           case we get hit, or in case the level is less than 2.
              IF the level is less than 2 THEN GOTO SHIP_INIT (set up the
                ship screen).
              ELSE IF when our chopper's last position was checked and it
                was on an odd dot column ( (Y AND 1)<>0 ) then GOTO
                PLANE_INIT (bring out enemy planes).
                ELSE Define the rocket characters.

310           Set YY equal to 600 for the rocket sound, delete all the
              sprites, place four invisible sprites at the top and bottom of
              the screen (this makes the rockets disappear under the scoring
              and reappear at the silos).

320           Generate the silo opening sound, define the silo character as
              open and generate the rocket sound.

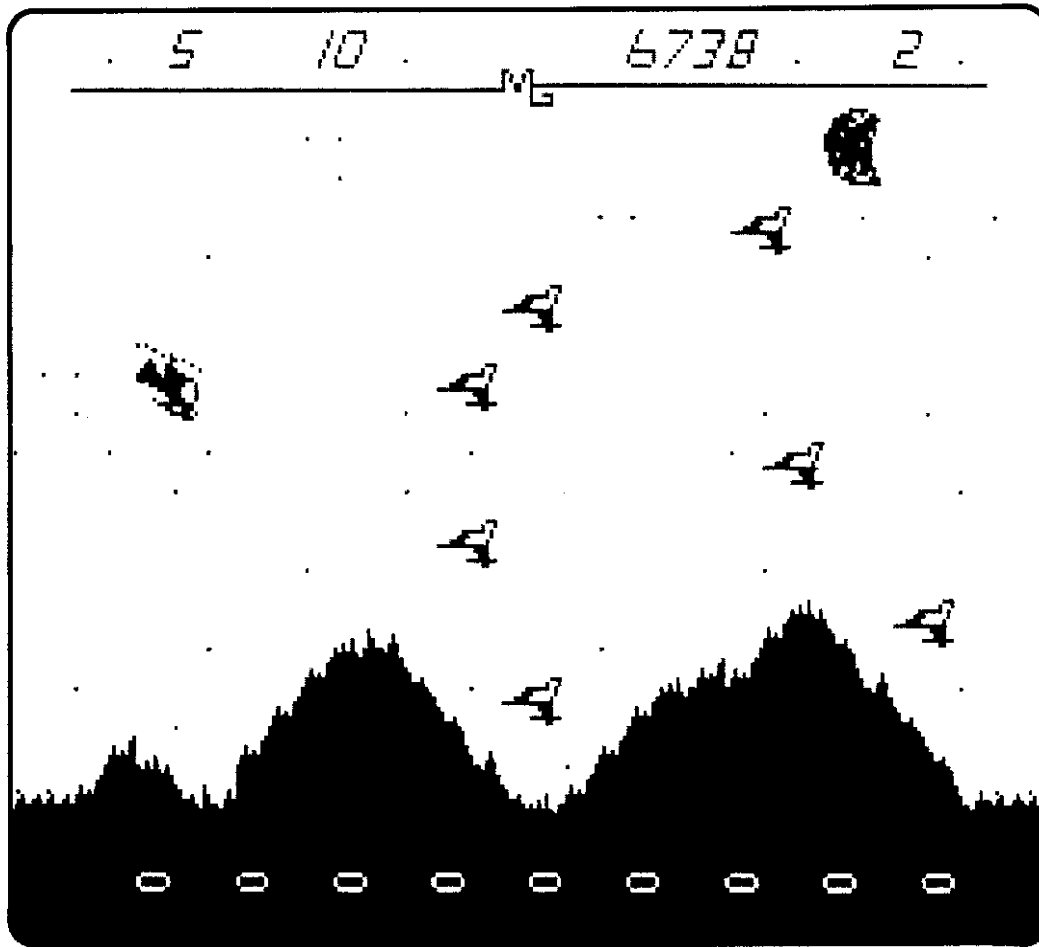
330           Bring out the 9 enemy rockets with random colors and at random
              speeds using the level (K) to increase the speeds at higher
              levels, and GOTO OUR_CHOP_OUT (bring out our chopper).

```

```
PLANE_INIT      340 CALL CHAR(60,T$&"01030F7
                FO"&E$&T$&"709FA8204FC3C1CFF
                OC"):: YY=1600 :: CALL SOUND
                (-4250,-8,6,110,27,115,28,YY
                ,30):: CALL DELSPRITE(ALL)

                350 FOR T=10 TO 18 :: RANDOM
                IZE :: CALL PEEK(-31880,X):·
                CALL SPRITE(#T,60,(X AND 7)
                +3,T*16-120,256,0,-X/8-3-K):
                : NEXT T

OUR_CHOP_OUT    360 CALL SPRITE(#1,112,13,72
                ,1,0,3)
```



PLANE_INIT Define the plane character, set YY equal to 1600 for the plane
 340 sound, generate the plane sound, and delete all the sprites.

350 Bring out the 9 enemy planes with random colors and at random
 speeds using the level (K) to increase the speeds at higher
 levels.

OUR_CHOP_OUT Bring out our forward flight chopper on the left hand edge of
 360 the screen flying slowly to the right.

```
R&P_LOOP      370 CALL SOUND(-999,-8,6,110
                ,27,115,28,YY,30):: CALL COI
                NC(ALL,T):: CALL POSITION(#1
                ,X,Y):: IF T OR X>161 THEN G
                OSUB 870 :: GOTO 740

                380 IF Y>224 THEN 410 ELSE C
                ALL KEY(1,T,T):: IF T<0 THEN
                CALL MOTION(#1,0,2):: GOTO
                370

                390 IF T=0 THEN T=8 ELSE IF
                T=5 THEN T=8*(X>35)ELSE IF T
                =3 THEN CALL MOTION(#1,0,4):
                : GOTO 370 ELSE CALL MOTION(
                #1,0,0):: GOTO 370

END_R&P_LOOP   400 CALL MOTION(#1,T,2):: GO
                TO 370
```

```

R&P_LOOP      Generate rocket or plane sound according to value of YY, check
370           coincidence between any sprites, and get the position of our
              chopper.
              IF there was a coincidence or our chopper has flown into the
                ground THEN GOSUB CRASH_CHOP (blow it up), GOTO TANK_INIT
                (go back to the tank screen).

380           IF the chopper has made it to the right hand edge of the
              screen THEN GOTO EC_INIT (move on to the next screen)
              ELSE scan for a key press.
                IF no key was pressed THEN slow down the chopper and
                  GOTO R&P_LOOP (start the loop over again).

390           IF the Down arrow key was pressed THEN set T equal to 8 for
              downward motion and continue with end_r&p_loop.
              ELSE IF the Up arrow key was pressed THEN set T equal to -8
                if our chopper is is on a dot row greater that 35
                else set T equal to 0 and continue with end_r&p_loop
              ELSE IF the Right arrow key was pressed THEN put our
                chopper into a faster forward motion, and GOTO
                R&P_LOOP (start the loop over again).
                ELSE it must have been some other key that was
                  pressed so bring our chopper to a stop, and GOTO
                  R&P_LOOP (start the loop over again).

END_R&P_LOOP  Put our chopper into motion according to the value of T (8,-8
400           or 0), and GOTO R&P_LOOP (start the loop over again).

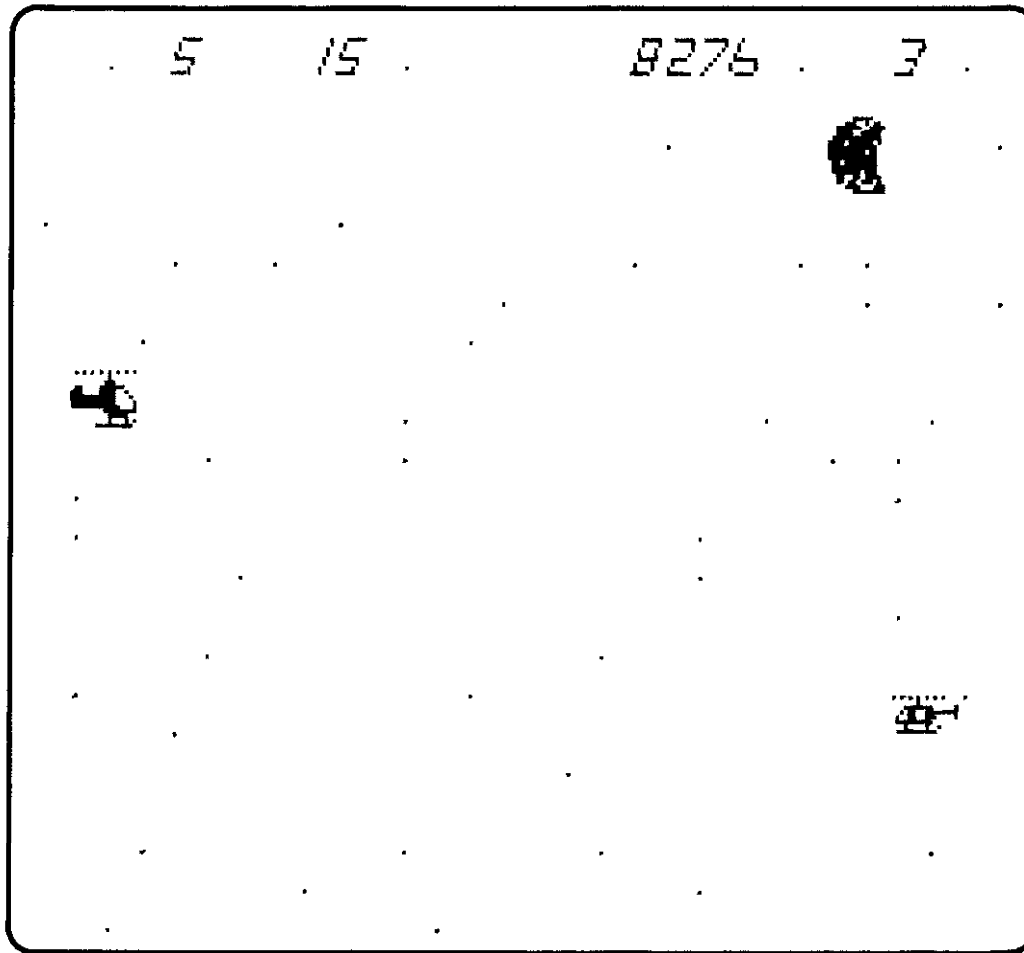
```

EC_INIT

```
410 CALL DELSPRITE(#1):: IF
K<3 THEN 570 ELSE Y1=(Y AND
6)-4 :: CALL DELSPRITE(ALL):
: CALL COLOR(2,1,1,1,1,1,13,
1,1)
```

```
420 DISPLAY AT(21,1):" e
      e     e     e
e           e     e
      e"
```

```
430 CALL CHAR(60,T$&"AA021F2
C4C7F107F"&S$&"A800C1FFE1C09
0E",64,S$&"00009292"&T$&S$&"
4949"):: CALL SPRITE(#1,108,
13,40,31,8,0)
```



EC_INIT
410

Delete our chopper
IF the level is less than 3 THEN GOTO SHIP_INIT (set up the ship screen).
ELSE set Y1 equal to a random number of enemy choppers based on the last dot column of our chopper, delete all sprites, and turn off the colors for the hills and the lines with the MG logo.

(Note: since Y, which comes from the call position statement, can only have a value between 1 and 256, the Y AND 6 function can only return 0,2,4 or 6 for any value of Y. By subtracting 4 from 0,2,4 or 6, Y1 can only have a value of -4,-2,0 or 2. Each time our chopper shoots down an enemy chopper we add 1 to Y1 and then test it to see if it is less than 5. If it is less than 5 there are more enemy choppers. So the least number of enemy choppers that must be shot down before you can go on to the next screen is 3 and the maximum number is 9).

420

Display some more stars (e) at the bottom of the screen.

430

Define the enemy chopper character and the sprite that it fires at us and we fire at them, and bring out our level flight chopper at the top left hand edge of the screen moving downward.

MG

```
EC_OUT      440 CALL SPRITE(#2,60,6,242,
            216,(Y AND 1)*80-40,0)

EC_LOOP     450 CALL SOUND(-4250,-4,1,20
            0,30,200,30,200,30):: CALL P
            OSITION(#1,X,Y,#2,YY,Y)

            460 IF X>180 THEN CALL LOCAT
            E(#1,1-(V<0)*180,31)ELSE IF
            ABS(X-YY)<7 THEN 490 ELSE CA
            LL MOTION(#2,SGN(X-YY)*((YY
            AND 11)+9),0)

            470 CALL KEY(1,T,Y):: IF T=1
            3 THEN 510 ELSE IF T=5 THEN
            V=-8 ELSE IF T=0 THEN V=8 EL
            SE IF Y THEN V=0

END_EC_LOOP 480 CALL MOTION(#1,V,0):: GO
            TO 450
```

```

EC_OUT      Bring out the enemy chopper, randomly from the top or bottom of
440         the screen based on our chopper's previous position, or the
           missile's position, that we fired at the last enemy chopper,
           dot column position (odd or even).

EC_LOOP     Generate chopper sound, get our chopper's and enemy chopper's
450         positions (we only need their dot row position since they are
           not allowed to change column positions).

460         IF our chopper is moving off the screen THEN wrap it around
           back onto the screen (determined by the motion direction
           according to the value of V).
           ELSE IF the enemy chopper is lined up with us THEN GOTO
           EC_FIRES (enemy shoots us down).
           ELSE make the enemy chopper move at a random speed to
           align itself with us.

470         Scan for a key press.
           IF the fire key was pressed THEN GOTO WE_FIRE (shoot at the
           enemy chopper).
           ELSE IF the Up arrow key was pressed then set V equal to -8
           (upward motion) and continue with end_ec_loop.
           ELSE IF the Down arrow key was pressed the set V equal
           to 8 (downward motion) and continue with
           end_ec_loop.
           ELSE IF any other key was pressed (Y<>0) then set V
           equal to 0 (no motion) and continue with
           end_ec_loop.

           (Note: if no key was pressed the above IFs would not be
           executed so the value of V would remain unchanged and the
           program flow would drop thru to end_ec_loop)

END_EC_LOOP Put chopper into an upward or downward motion according to the
480         value of V and GOTO EC_LOOP (start the loop over again).

```



```

EC_FIRES      490 CALL SPRITE(#3,64,7,YY,2
              09,V+X-YY,-127):: CALL SOUND
              (-900,-8,1,110,30,110,30,999
              9,30):: T=0

WHERE         500 CALL POSITION(#3,Y,Y)::
              IF Y>50 THEN 500 ELSE CALL D
              ELSPRITE(#3):: CALL COLOR(#1
              ,11):: GOSUB 930 :: CALL DEL
              SPRITE(#2):: GOTO 740

WE_FIRE      510 CALL SOUND(-900,-8,0,110
              ,30,110,30,300,30):: CALL PO
              SITION(#1,X,Y):: CALL SPRITE
              (#3,64,4,X,36,0,127)

WHERE1       520 CALL POSITION(#3,X,Y)::
              IF Y<192 THEN 520 ELSE CALL
              DELSPRITE(#3):: CALL COINC(#
              2,X,220,7,T):: IF T THEN CAL
              L SOUND(-1,-4,9)ELSE 450

BLOW_UP_EC   530 CALL COLOR(#2,15):: FOR
              T=0 TO 2 :: CALL PATTERN(#2,
              88+T*4):: FOR X=5 TO 7 :: CA
              LL SOUND(100,-X,T*10):: NEXT
              X :: NEXT T

              540 CALL DELSPRITE(#2):: Y1=
              Y1+1 :: SC=SC+250 :: DISPLAY
              AT(1,12)SIZE(10):USING "###
              #####":SC :: IF Y1<5 THEN
              440

              550 CALL POSITION(#1,X,Y)::
              IF X>185 THEN CALL LOCATE(#1
              ,35,Y)

              560 CALL PATTERN(#1,112):: C
              ALL MOTION(#1,0,20):: CALL S
              OUND(-4250,-4,1,200,30,200,3
              0,200,30)

```

EC_FIRES Shoot enemy's missile at us, according to the values obtained
490 from the last position check, generate missile fire sound, and
 set T equal to zero. T is used as a flag in the BLOW_UP_CHOP
 subroutine to indicate a coincidence between the crashing
 chopper and the tank on the Tank screen.

WHERE Check the missile's position.
500 IF it has not reached us yet GOTO WHERE (keep checking).
 ELSE delete the missile, change our choppers color to light
 yellow for a better blow up effect, GOSUB BLOW_UP_CHOP
 (blow up our chopper), delete the enemy chopper and GOTO
 TANK_INIT (go back to the Tank screen).

WE_FIRE Generate missile fire sound, get our chopper's position and
510 shoot our missile at the enemy from our current position.

WHERE1 Check the missile's position.
520 IF it has not reached them yet GOTO WHERE1 (keep checking).
 ELSE Delete the missile and check for coincidence between our
 missile's last position and the enemy's chopper.
 IF we hit them, turn off the missile sound, since the
 next sound statements have a positive duration, and
 continue with BLOW_UP_EC (blow up the enemy chopper).
 ELSE GOTO EC_LOOP (start the loop over again).

BLOW_UP_EC Since we hit them change their color to grey for a better blow
530 up effect, and start the blow up routine. Change their pattern
 to 88,92 & 96 while the blow up sound is being generated.

540 Delete the blow up pattern, add 1 to the enemy choppers
 destroyed counter (Y1), add 250 points to our score (SC), and
 display the new score at the top of the screen.
 IF the EC counter is less than five (there are more enemy
 choppers to shoot down before we can move on so) THEN GOTO
 EC_OUT (bring out the next enemy chopper).

550 Check our chopper's position.
 IF it is too low or off the screen THEN locate it at the
 top of the screen at its current dot column.

560 Change its pattern to forward flight, set it in forward motion,
 generate the chopper sound and continue execution with
 SHIP_INIT (set up the Ship screen).

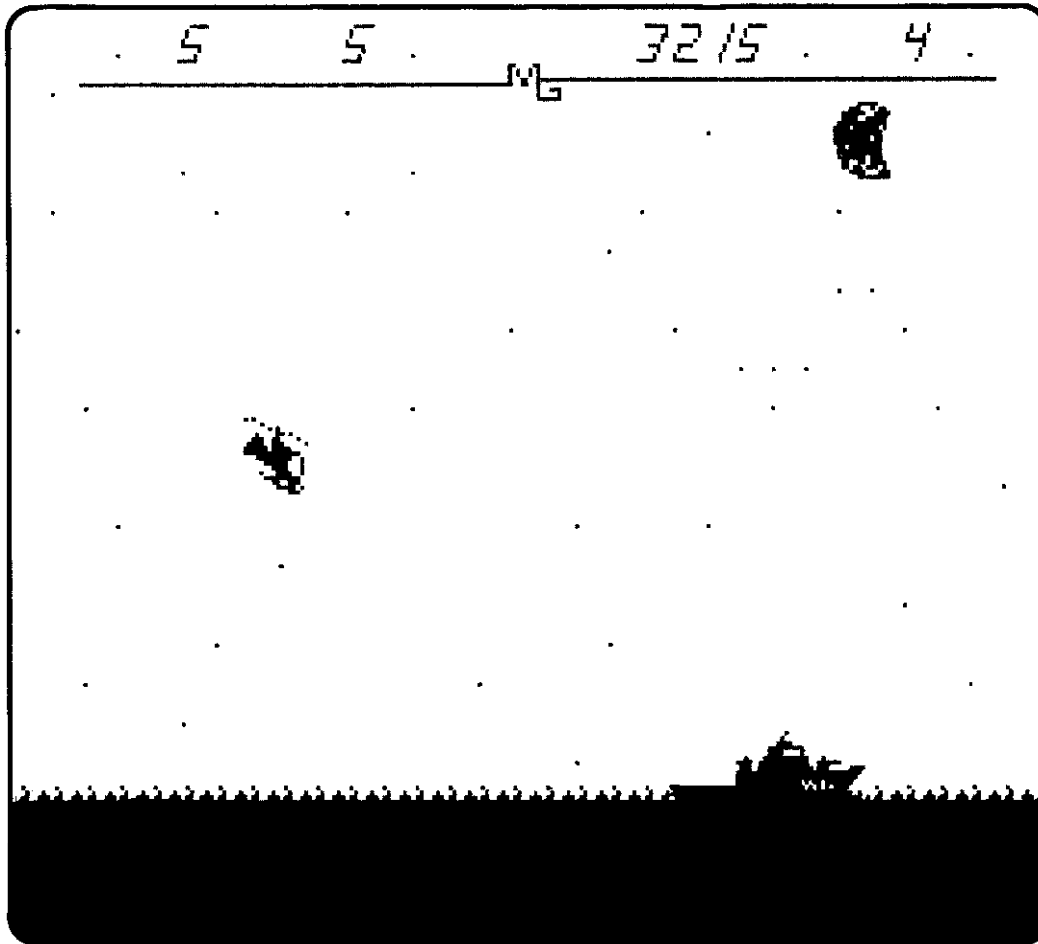
SHIP_INIT

```
570 CALL CHAR(68,S$&"000000F
F7F1FFEAB"&S$&"000000FFFFFFFA
A6D",64,E$&"0021F373FFFFFFFFF
FFFF972A08103CFFFOFFFFFFFFF
FFFFFFFF5EAA")
```

```
580 Y1=(Y AND 5)+6 :: GOSUB
800 :: CALL CHAR(60,"0000008
080C4CFEEEEFFF5BAAFFF722B"&T
$&"CO0FFEF8F0E2C67B3D")::
CALL COLOR(1,1,1,2,1,1)
```

```
590 DISPLAY AT(20,1):W$ :: C
ALL COLOR(1,5,1,7,12,1):: FO
R T=1 TO 4 :: CALL LOCATE(#T
,161,1):: NEXT T :: CALL SPR
ITE(#9,112,13,15,1,20,35)
```

```
600 CALL SPRITE(#10,68,15,14
3,1,#11,64,15,143,17,#12,60,
15,143,33):: CALL MOTION(#10
,0,Y1,#11,0,Y1,#12,0,Y1)
```



SHIP_INIT Define the back portion (char 68) and middle portion (char 64) of the ship.

570

580 Calculate a random value for the ship's speed based on the last value of Y (dot column tracker), GOSUB INVIS_SUB (delete all sprites and put invisible sprites at the bottom of the screen), define rest of the ship, and turn off the hill colors (in case we haven't just come from the Enemy Chopper screen.)

590 Display the top of the water at row 20, turn on the water and the MG logo colors, put 4 more invisible sprites just below the top of the water for the sinking effect, and bring out our chopper.

600 Bring out the 3 sprites that make up the ship and put them all into motion at the same time so the ship doesn't break up.

```

SHIP_LOOP      610 CALL SOUND(-4250,-4,1,20
                0,30,200,30,200,30):: CALL P
                OSITION(#9,X,Y):: IF X<35 TH
                EN CALL MOTION(#9,0,V):: CAL
                L LOCATE(#9,35,Y)ELSE IF X>1
                40 THEN 690

                620 CALL COINC(#9,#11,16,T):
                : IF T AND X>130 THEN 680 EL
                SE CALL COINC(#9,#10,9,T)::
                IF T THEN 710

                630 CALL KEY(1,T,T):: IF T<0
                THEN CALL PATTERN(#9,SGN(V)
                *4+108):: CALL MOTION(#9,3*(
                X>35),V):: GOTO 610

                640 IF T=3 THEN CALL PATTERN
                (#9,112):: V=V-4*(V<12):: GO
                TO 670 ELSE IF T=2 THEN CALL
                PATTERN(#9,104):: V=V+4*(V>
                -12):: GOTO 670

                650 IF T=0 THEN CALL PATTERN
                (#9,108):: CALL MOTION(#9,8,
                V):: GOTO 610

                660 IF T=5 THEN CALL PATTERN
                (#9,108):: CALL POSITION(#9,
                X,Y):: IF X<36 THEN CALL LOC
                ATE(#9,35,Y)ELSE CALL MOTION
                (#9,-8,V):: GOTO 610

END_SHIP_LOOP  670 CALL MOTION(#9,0,V):: GO
                TO 610

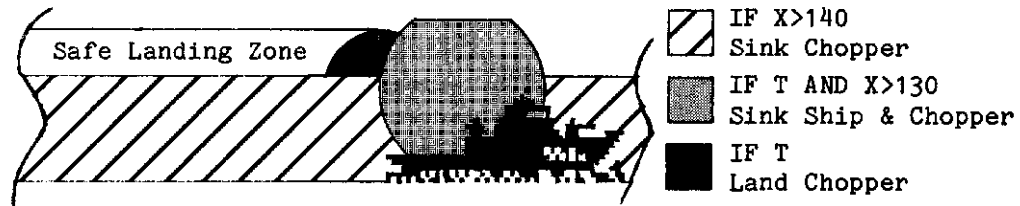
```

```

SHIP_LOOP      Generate chopper sound, get chopper's position,
610           IF  it is too high THEN adjust its motion to level flight and
              bring it down to dot row 35.
              ELSE IF  it is too low THEN GOTO SINK_CHOP (Sink the chopper).

620           Check coincidence between our chopper and the sprite in the
              middle of the ship (see drawing).
              IF  there is a coincidence GOTO SINK_SHIP (crash the chopper,
              sink the ship and sink the chopper).
              ELSE check for proper coincidence between our chopper and the
              back of the ship (see drawing).
              IF  there is a coincidence THEN GOTO LAND_CHOP (land the
              chopper on the back of the ship).

```



```

630           Scan for a key press.
              IF  no key was pressed THEN change pattern to forward or
              backward motion chopper according to the value of V,
              adjust motion to make our chopper fly slowly up and GOTO
              SHIP_LOOP (start the loop over again).

640           IF  the Right arrow key was pressed THEN change the choppers
              pattern to forward flight, add 4 to V if V is less than
              12, and GOTO END_SHIP_LOOP (change the chopper's motion).
              ELSE IF  the Left arrow key was pressed THEN change the
              chopper's pattern to backward flight, subtract 4 from
              V if V is greater than -12, and GOTO END_SHIP_LOOP
              (set the chopper into the new motion).

650           IF  the Down arrow key was pressed THEN change the chopper's
              pattern to level flight, set the chopper into a diagonal
              downward motion along with the forward or backward motion
              according to the value of V, and GOTO SHIP_LOOP (start the
              loop over again).

660           IF  The Up arrow key was pressed THEN change the chopper's
              pattern to level flight, and get its current position.
              IF  it is too high THEN bring it back down to dot row 35
              at its current dot column, and continue with
              end_ship_loop.
              ELSE set the chopper into an upward motion along with the
              forward or backward motion according to the value of
              V and GOTO SHIP_LOOP (start the loop over again).
              (Note: If a key was pressed that wasn't one of the 4 arrow keys
              the program would not execute any of the previous IF statements
              or change the value of V. It would just drop through and
              execute end_ship_loop.)

```

```

END_SHIP_LOOP Set the chopper into forward or backward motion according to
670           the value of V, and GOTO SHIP_LOOP (start the loop over again).

```

```

SINK_SHIP      680 CALL MOTION(#9,0,0,#10,2
,4,#11,2,4,#12,2,4):: SC=SC-
SP*500 :: SP=0

SINK_CHOP     690 CALL MOTION(#9,3,0):: CA
LL SOUND(-2450,-8,6,110,30,1
10,30,9999,30):: CALL PATTEN
N(#9,104):: CALL SOUND(1,-4,
9):: HP=0 :: Z=Z-1

700 CALL DELSPRITE(#9):: IF
SP THEN 740 ELSE CALL DELSPR
ITE(#10,#11,#12):: GOTO 740

LAND_CHOP    710 CALL POSITION(#10,X,Y)::
CALL SPRITE(#9,116,13,136,Y
,0,Y1):: FOR T=306 TO 122 ST
EP -6 :: CALL SOUND(-200,-4,
1,T,27,T,30,T,30):: NEXT T

720 FOR T=130 TO 306 STEP 8
:: CALL SOUND(-4250,-4,1,T,3
0,T,30,T,27):: NEXT T :: CAL
L MOTION(#9,-10,0):: K=K+1 :
: SC=SC+500*HP

730 FOR T=1 TO 300 :: NEXT T
:: SP=SP+HP :: HP=0

```

SINK_SHIP Stop our choppers motion and set the ship into a slow diagonal
680 downward motion to sink it, subtract all the points for the
 total number of men that have been delivered to the ship, and
 reset the variable for the total men delivered to the ship (SP)
 to zero.

SINK_CHOP Set our chopper into a slow downward motion, generate the
690 sizzling sound, change our choppers pattern to the backward
 flight chopper to give it the effect of sinking tail first,
 turn off the sizzling sound, reset the variable for the number
 of men in the chopper (HP) to zero, and subtract 1 from the
 number of our choppers remaining (Z).

700 Delete our chopper off the screen.
 IF there are men on the ship (SP<>0) (we didn't sink the
 ship, however, it is possible that we never delivered any
 men to the ship, we just sank our chopper) THEN GOTO
 TANK_INIT (set up the tank screen).
 ELSE Delete the ship, because odds are that it is sinking and
 we don't want it to wrap around and come floating out of
 the top of the screen, and then GOTO TANK_INIT (set up the
 Tank screen).

LAND_CHOP Get the back of the ship's position, set our chopper down there
710 in the landed pattern using the same motion (velocity) as the
 ship, and generate the sound of the chopper engines slowing
 down (for a slight time delay to give your joystick or keyboard
 hand a rest).

720 Rest time is over, so rev up the chopper engines, make our
 chopper lift off the ship, add 1 to the level (K), and add 500
 points for each man in the chopper to the score.
 (Note: by using the HP variable here you can easily change the
 total number of men required to be picked up on the tank screen
 without worrying about adjustments to the scoring).

730 Generate a slight time delay to allow the chopper to get higher
 in the air before going back to the Tank screen, add the number
 of men in the chopper (HP) to the total men delivered to the
 ship (SP), and reset the number of men in the chopper (HP) to
 zero, and continue with tank_init.

TANK_INIT

740 DISPLAY AT(1,2):USING B\$
:HP,SP,SC,Z :: IF Z THEN CAL
L DELSPRITE(ALL)ELSE 250

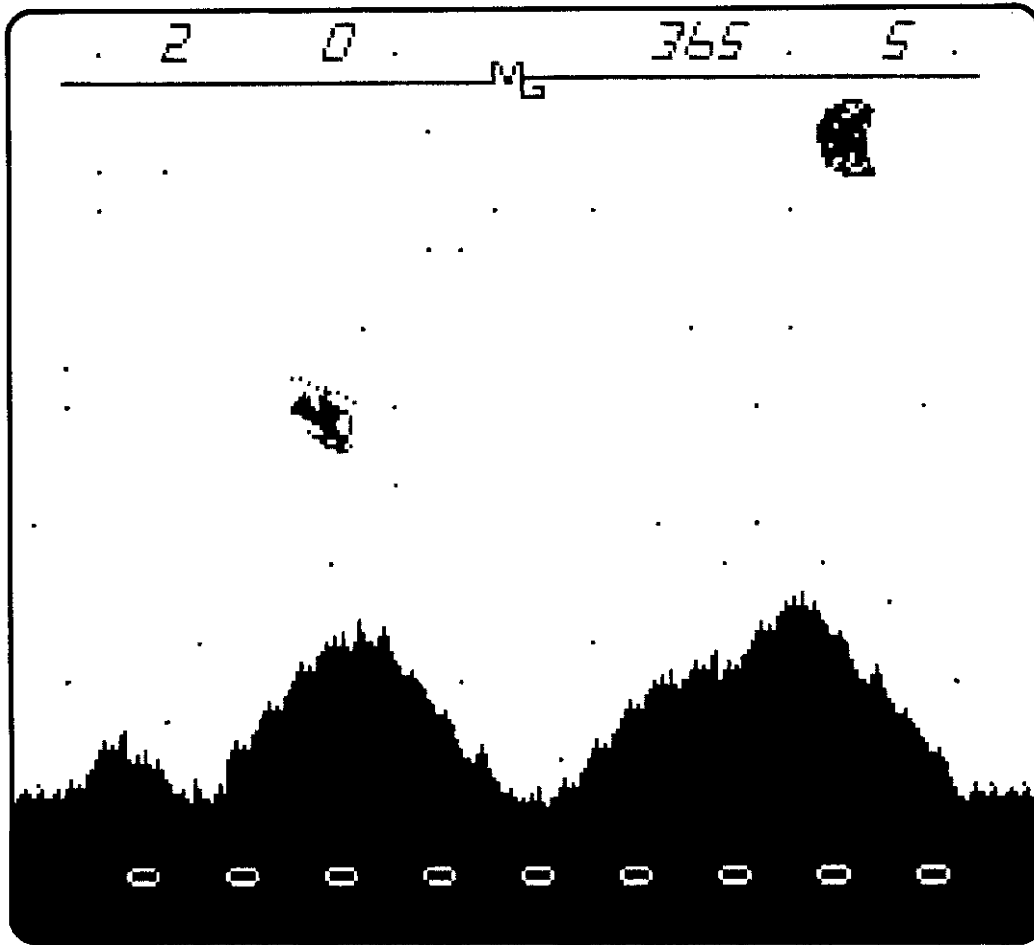
750 CALL CHAR(60,"000001010F
090909010101013E200000C0C4C4
FCC0C0C0C0C02010080810203")

760 CALL CHAR(64,"0101031F13
1313030303033F20000000808888
88F8808080808000008040203")

770 CALL CHAR(68,"0000010107
0909050101010106181000C0C0C2
F4C8C0C0C0C02010080810203",3
5,"FFFFFFFF817E7E81")

780 DISPLAY AT(20,1):A\$:: C
ALL COLOR(2,11,1,1,11,1,13,1
2,1):: GOSUB 800

790 CALL SPRITE(#3,140,2,161
,256,0,-24):: FOR T=9 TO 20
:: CALL SPRITE(#T,136,7,200,
1):: NEXT T :: GOTO 1030



```

TANK_INIT      Display the scoring on row 1.
740             IF there are any of our choppers left (Z<>0) THEN delete all
                sprites.
                ELSE GOTO GAME_OVER (end of game).

750             Define the first pattern of the running man sprite (char 60).

760             Define the second pattern of the running man sprite (char 64).

770             Define the third pattern of the running man sprite (char 68)
                and redefine the missile silo character as closed, in case it
                was opened.

780             Display the Tank screen foreground with the missile silos in
                case we just came from the Ship or Enemy Chopper screens, turn
                on the colors for the hills and the MG logo, and GOSUB
                INVIS_SUB (delete all sprites and place 4 invisible sprites at
                the bottom of the screen).

790             Bring out the tank from the right hand edge of the screen at a
                rapid motion so that it will be near the middle of the screen
                when our chopper comes out, set up the sprites used for the
                laser fire but place them off the visible portion of the
                screen, and GOTO CHOP_OUT (bring out our chopper).

```

```
INVIS_SUB      800 CALL DELSPRITE(ALL):: FOR T=5 TO 8 :: CALL LOCATE(#T,177,T*17):: NEXT T :: RETURN

TANK_SHOOTS   810 IF T<>12 THEN CALL POSITION(#1,X,Y,#3,YY,YY):: IF ABS(Y-YY)<80 THEN CALL SPRITE(#4,36,9,157,YY,X-147,2*V+Y-YY)ELSE 1060 ELSE 1060

               820 CALL SOUND(-150,-8,3,110,30,110,30,5010,30):: CALL SOUND(300,-8,1,128,30,128,30,1100,30):: CALL DELSPRITE(#4):: GOSUB 870 :: GOTO 1030

MACHINE_GUN   830 CALL MOTION(#3,0,V/4,#2,0,0):: CALL PATTERN(#2,80):: CALL SOUND(-1,-4,9):: IF T=12 THEN CALL MOTION(#1,0,V)

               840 FOR T=1 TO 9 :: CALL SOUND(50,-6,1):: NEXT T :: CALL DELSPRITE(#2):: Y1=0 :: RETURN
```

INVIS_SUB This subroutine deletes all sprites, sets up 4 invisible
800 sprites at the bottom of the screen, and returns.

TANK_SHOOTS IF our chopper is not just coming out (T<>12, the just coming
810 out flag) THEN get our choppers current position and the
 tanks current position, we only need the tanks dot column
 since its dot row is not allowed to change.
 IF they are still in alignment and our chopper has not
 flown, wrapped, around to the other side of the
 screen (ABS(Y-YY)<80) THEN make the tank shoot a
 sprite at the chopper from the tanks current position
 to the choppers current position and continue with
 the next line.
 ELSE the chopper must have flown around to the other side
 of the screen so just GOTO TANK_LOOP1 (go back into
 the tank_loop).
 ELSE our chopper is just coming out so give it a chance and
 GOTO TANK_LOOP1 (go back into the tank_loop).

820 Generate tank shoots sound which we will use in conjunction
 with the next positive duration sound statement as a time
 delay, generate the hit sound, delete the sprite that was shot
 at the chopper, GOSUB CRASH_CHOP (crash or blow up the chopper)
 and GOTO OUR_CHOP_OUT (to start the tank loop over).

MACHINE_GUN This subroutine, which can be branched to from the tank_loop or
830 the land_it sections of the program, will change the tanks
 motion and move it at 1/4 the speed of our choppers last
 velocity (this usually makes the tank move away from the
 chopper when it has landed on the ground so that the
 blow_up_chop routine doesn't cause the chopper to hit the tank
 and blow it up too). Also, in case the man is running, stop his
 motion, change the man's pattern in case he is on the screen,
 to the bent over figure, stop any previous sound statements
 that may be on since the next sound statements have a positive
 duration.
 IF the chopper is just coming out (T=12, the just coming out
 flag) THEN change its motion to level flight so that it
 doesn't fly through the ground while the tank machine guns
 the man.

840 Generate the machine gun sound, delete the man off the screen,
 set the mans dot column variable (Y1) to zero, this tells the
 tank_loop that the man is no longer on the screen and it tells
 the land_it loop to bring out another man, and then return to
 the next statement after the GOSUB MACHINE_GUN statement.

```
BLOW_UP_TANK 850 CALL COLOR(#3,2):: FOR T
              =0 TO 16 STEP 4 :: CALL SOUN
              D(-999,-8,T,120,27,127,28,10
              00,30):: CALL PATTERN(#3,84+
              T):: NEXT T

              860 SC=605-3*X+SC :: CALL DE
              LSPRITE(#3):: CALL SPRITE(#3
              ,140,2,161,256,0,-24):: RETU
              RN

CRASH_CHOP    870 T=0 :: CALL SOUND(-1,-4,
              9):: IF Y AND 1 THEN CALL PA
              TTERN(#1,112):: CALL MOTION(
              #1,9,V)ELSE CALL COLOR(#1,16
              ):: GOTO 930

NEW_COLOR     880 CALL COLOR(#1,RND*7+3)::
              FOR T=1 TO 26 STEP 5 :: CAL
              L SOUND(T*40+200,-8,T,110,30
              ,110,30,1100-T,30)

              890 CALL POSITION(#1,X,Y)::
              IF X>155 THEN CALL MOTION(#1
              ,0,0):: GOTO 910 ELSE IF X A
              ND 2 THEN 880

              900 NEXT T :: GOTO 880
```

BLOW_UP_TANK This subroutine will change the tanks color back to black since
850 the laser_fire routine changes it to white, start the blow up
routine by generating the blow up sound while changing the tank
pattern into expanding fragments.

860 Calculate and add a number of points to our score (the higher
the chopper is in the air the greater the points, X is the
choppers dot row position, so if the chopper is low, dot row
160, this will add 125 points and if the chopper is at the top
of the screen, dot row 35 this will add 500 points to our
score), delete the blown up tank sprite, bring out a new tank
from the right hand edge of the screen, and return.

CRASH_CHOP This Subroutine Has 2 entry points. The First one randomly
870 decides to either crash the chopper into the ground or to just
blow it up in the air. The second entry point at BLOW_UP_CHOP
blows up the chopper wherever it is. So, by letting CRASH_CHOP
fall through to BLOW_UP_CHOP we can crash the chopper into the
ground and then blow it up. First we clear the COINC flag (T=0)
in case we hop right into BLOW_UP_CHOP. Next we turn off any
previous sounds that may be on since the next sound statements
have a positive duration.
IF Our choppers last checked position was on an odd dot
column (Y AND 1) THEN we will get ready to crash it into
the ground by changing its pattern to the forward flight
chopper, and setting it into a diagonally downward motion
according to the value of V.
ELSE We will get ready to blow it up in the sky by changing its
color to white and then we will GOTO BLOW_UP_CHOP.

NEW_COLOR This is part of the loop that crashes the chopper into the
880 ground. First change its color to a randomly selected color
between 3 and 10. Next we start up a For-Next loop that may or
may not be completed but it is used to generate a sound like
the choppers engines are in trouble.

890 Now we check our choppers position.
IF we are close to the ground (X>155) THEN stop our choppers
motion and GOTO CHK_COINC (this leaves the For-Next loop
and the crash loop).
ELSE IF our choppers last dot row position has the binary bit
for the value of 2 turned on, then goto NEW_COLOR.
This will not allow the entire For-Next loop to
execute so the sound will not have a regular pattern.
(the X AND 2 is a simple random test since the
chopper is constantly changing its position)

900 Since the X AND 2 test returned 0 or false then execute NEXT T.
When the For-Next loop is finished GOTO NEW_COLOR and start the
loop over again. The only way out of the loop is when the
choppers dot row is greater than 155.

```

CHK_COINC 910 CALL COINC(#1,#2,16,T)::
             IF T THEN CALL DELSPRITE(#2
             )

             920 CALL COINC(#1,#3,17,T)::
                 CALL COLOR(#1,2):: IF T THE
                 N CALL SOUND(-300,-8,1,110,3
                 0,110,30,3000,30):: CALL COL
                 OR(#3,7)

BLOW_UP_CHOP 930 FOR Y=0 TO 16 STEP 4 ::
                 CALL SOUND(-999,-8,Y,120,27,
                 127,28,1100,30):: CALL PATTE
                 RN(#1,84+Y):: NEXT Y

                 940 CALL DELSPRITE(#1):: Z=Z
                 -1 :: HP=0 :: IF T THEN GOSU
                 B 850 :: RETURN ELSE RETURN

LAND_IT 950 CALL MOTION(#1,0,V):: CA
             LL COINC(#1,#2,12,YY):: IF Y
             Y THEN CALL SOUND(-500,-8,1,
             110,30,110,30,840,28):: CALL
             DELSPRITE(#2)

             960 CALL SOUND(-4250,-4,1,14
             0,30,140,30,140,30):: CALL P
             OSITION(#1,X,Y,#2,Y1,Y1):: C
             ALL SPRITE(#1,116,13,160,Y,0
             ,0)

MAN_OUT 970 CALL MOTION(#2,0,4*SGN(Y
             -Y1)):: IF Y1=0 THEN CALL SP
             RITE(#2,76,2,163,256)

```

CHK_COINC When the program flow comes here our chopper is near the ground
910 (X>155) so check to see if it has crashed on top of the man.
IF it has crashed on top of the man THEN delete him off the
screen.

920 Next check to see if our chopper has crashed on top of the
tank and change our choppers color to black since it was
changed to random colors in the crash loop.
IF it has crashed on top of the tank THEN generate a noise
that sounds like the choppers blades hitting the tank and
change the tank's color to red.

BLOW_UP_CHOP Now we will execute a For-Next loop to generate the blow up
930 sound while it changes the blow up pattern for our chopper.

940 Next we delete our blown up chopper off the screen, subtract 1
from the number of our choppers remaining (Z=Z-1) and clear out
the variable that contains the number of men currently in our
chopper (HP=0).
IF our chopper crashed into the tank (T<>0) THEN GOSUB
BLOW_UP_TANK and RETURN.
ELSE just RETURN to the statement after gosub crash_chop or
blow_up_chop.

LAND_IT This routine, which makes our chopper land on the ground, is
950 branched to from the TANK_LOOP when our chopper's dot row is
greater than 151. First we stop its downward motion but we keep
its forward motion going so it can slide in for a landing. Then
we check to see if we landed the chopper on top of the man.
IF we have landed on top of the man (YY<>0) THEN generate a
sound and delete the man off the screen.

960 Next generate a slightly different sound for the chopper while
its on the ground, check the position of our chopper and the
man so we can set the man into motion toward our chopper and
then set our chopper on the ground in the landed pattern with
all velocities at 0 or bring it to a stop.

MAN_OUT Now put the man into motion toward our chopper. Note: Since we
970 used CALL DELSPRITE(#2) instead of CALL DELSPRITE(ALL) the
sprite is not actually deleted it is just moved off the screen.
IF the man is not on the screen (Y1=0 when the sprite is
deleted and its position is checked) THEN bring him out on
the extreme right hand edge of the screen. The previous
CALL MOTION will put him into motion towards the chopper
so he will smoothly come out on the left hand edge of the
screen.

```
KEY_LOOP  980 CALL KEY(1,T,T):: IF T=5
          THEN 1020 ELSE YY=YY-4 :: I
          F YY<60 THEN YY=76

          990 CALL PATTERN(#2,YY):: CA
          LL COINC(#1,#3,32,T):: IF T
          THEN GOSUB 830 :: GOSUB 870
          :: GOTO 1030

          1000 CALL COINC(#2,#3,24,T):
          : IF T THEN GOSUB 830 :: GOT
          O 970 ELSE CALL COINC(#1,#2,
          11,T):: IF T THEN CALL PATTE
          RN(#2,80)ELSE 980

          1010 CALL SOUND(-200,220,7,2
          23,8,226,9):: HP=HP+1 :: DIS
          PLAY AT(1,3)SIZE(2):HP :: CA
          LL DELSPRITE(#2)

TAKE_OFF  1020 CALL SOUND(-4000,-4,1,1
          10,30,110,30,320,30):: CALL
          MOTION(#1,-17,V/2,#2,0,0)::
          CALL PATTERN(#2,128):: IF HP
          =5 THEN 300 ELSE 1050
```

```

KEY_LOOP  Scan the left hand keyboard, we don't care about the status so
980        we can reuse the T variable here for faster operation.
          IF the up arrow key was pressed THEN goto TAKE_OFF.
          ELSE subtract 4 from the pattern variable for our animated
            running man.
            IF the pattern variable is less than 60 we are at the
              end of the running man patterns so THEN reset it back
                to 76.

990        Change the running man's pattern and then check to see if the
          tank is within 32 pixels of our chopper.
          IF it is (T<>0) THEN GOSUB MACHINE_GUN to shoot the tank and
            man, GOSUB CRASH_CHOP to blow up our chopper and GOTO
              OUR_CHOP_OUT to start the Tank Loop all over again.

1000       Since the tank was not close to our chopper check to see if
          just the man is close to the tank.
          IF he is THEN GOSUB MACHINE_GUN to shoot the man and GOTO
            MAN_OUT to bring out another man and continue with the
              key_loop.
          ELSE Since the man was not close to the tank check to see if he
            is close to our chopper so he can climb aboard.
            IF he is then change his pattern and make him bend down
              to climb in and continue with the next line.
            ELSE since he is not close enough to our chopper yet GOTO
              KEY_LOOP and start this loop over again.

1010       Since he was close enough to climb aboard generate the beep
          sound to indicate that he made it, add 1 to the variable that
            contains the total number of men in our chopper, display the
              total number of men in our chopper at the top of the screen and
                delete the man of the screen.

TAKE_OFF  Generate the take off sound for our chopper, put our chopper
1020      into a diagonally upwards motion according to the value of V
          and stop the running man and change his pattern to the waving
            man in case we came to this part of the routine directly from
              key_loop and left him on the ground.
          IF we have 5 men in the chopper THEN goto R&P_INIT to check
            the level and see which screen we should goto next.
          ELSE go back to the TANK_LOOP to continue playing on the Tank
            Screen since we don't have enough men in the chopper yet.

```

```
OUR_CHOP_OUT  1030 IF Z THEN CALL SPRITE(#
               1,112,13,20,1,20,35):: T,V=1
               2

TANK_SCORE    1040 DISPLAY AT(1,2):USING B
               $:HP,SP,SC,Z :: IF Z=0 THEN
               250

TANK_LOOP     1050 CALL SOUND(-999,-4,1,11
               0,30,110,30,200,30):: CALL P
               OSITION(#1,X,Y,#2,Y1,Y1,#3,Y
               Y,YY):: IF ABS(YY-Y)<5 THEN
               810 ELSE IF Y1 THEN CALL PAT
               TERN(#2,124)

TANK_LOOP1    1060 CALL MOTION(#3,0,SGN(Y-
               YY)*((Y AND 6)+4+K)):: IF X<
               35 THEN CALL MOTION(#1,0,V):
               : CALL LOCATE(#1,35,Y)ELSE I
               F X>151 THEN 950

               1070 IF Y1 THEN CALL PATTERN
               (#2,128):: IF ABS(YY-Y1)<26
               THEN GOSUB 830
```

```

OUR_CHOP_OUT  IF  there are any choppers left (Z<>0) THEN bring out our
1030          chopper from the upper left hand corner of the screen
              moving diagonally down and forward, initialize the
              chopper's velocity variable (V) and set the chopper just
              coming out flag (T) with the same value.

TANKSCORE     Display the scoring on row 1.
1040          IF  there aren't any choppers left (Z=0) THEN GOTO GAME_OVER
              (end of game).

TANK_LOOP     Generate chopper sound, get the chopper's (sprite #1), the
1050          running man's (sprite #2) and the tank's (sprite #3) positions
              (Note: we need the dot row and dot column for our chopper but
              we only need the dot column for the man and the tank since they
              are always on the same dot row).
              IF  the tank is lined up within 4 dot column pixels of the
              flying chopper (ABS(Y1-Y)<5) THEN GOTO TANK_SHOOTS (tank
              shoots at the flying chopper).
              ELSE IF  the man is on the screen (Y1<>0) THEN change his
              pattern to make his arm wave.

TANK_LOOP1    Make the tank move towards the chopper (SGN(Y-YY) at a random
1060          speed based on our choppers last dot column position ((Y AND
              6)+4) adding some extra speed for the current level (+K).
              IF  our chopper is too high on the screen (X<35) THEN change
              its motion into level flight in the current direction it
              is heading (V), and relocate it back down to dot row 35 at
              its current dot column position.
              ELSE IF  our chopper is near the ground (X>151) THEN GOTO
              LAND_IT (land it on the ground).

1070          IF  the man is on the screen (Y1<>0) THEN change his pattern
              to make his arm wave and,
              IF  the tank is within 25 dot column pixels of him
              (ABS(YY-Y1)<26 THEN GOSUB MACHINE_GUN (make the tank
              shoot the man).

```

```
1080 CALL KEY(1,T,T):: IF T=
13 THEN 1130 ELSE IF T<0 THE
N CALL PATTERN(#1,V/3+108)::
CALL MOTION(#1,3*(X>35),V):
: GOTO 1050
```

```
1090 IF T=3 THEN CALL PATT
ER N(#1,112):: V=12 :: GOTO 112
0 ELSE IF T=2 THEN CALL PATT
ER N(#1,104):: V=-12 :: GOTO
1120
```

```
1100 IF T=0 THEN CALL PATT
ER N(#1,108):: CALL MOTION(#1,8
,V):: GOTO 1050
```

```
1110 IF T=5 THEN CALL PATT
ER N(#1,108):: CALL POSITION(#1
,X,Y):: IF X<36 THEN CALL LO
CATE(#1,35,Y)ELSE CALL MOTIO
N(#1,-12,V):: GOTO 1050
```

```
END_TANK_LOOP 1120 CALL MOTION(#1,0,V):: G
OTO 1050
```

```

1080 Scan the left hand side of the keyboard for a key press (we
      don't need the status).
      IF the fire key was pressed (T=13) THEN GOTO LASER_FIRE
         (shoot at the tank).
      ELSE IF no key was pressed (T<0 or T=-1) THEN set the
         choppers pattern according to the velocity (V), the
         pattern equals 104 (backwards) when V=-12 and it
         equals 112 (forwards) when V=12, set the chopper into
         a slightly upward diagonal motion (forward or
         backward according to V) if it is not above or at dot
         row 35, and then GOTO TANK_LOOP (start the over loop
         again).

1090 IF the Right arrow key was pressed THEN change the choppers
      pattern to forward flight (char 112), set the velocity
      variable (V) equal to 12, and GOTO END_TANK_LOOP (adjust
      our choppers motion).
      ELSE IF the Left arrow key was pressed THEN change the
         choppers pattern to backward flight (char 104), set
         the velocity variable (V) equal to -12, and GOTO
         END_TANK_LOOP (adjust choppers motion).

1100 IF the Down arrow key was pressed THEN change the choppers
      pattern to level flight, set it into a diagonally downward
      motion (forward or backward according to V), and GOTO
      TANK_LOOP.

1110 IF the Up arrow key was pressed THEN change the choppers
      pattern to level flight, check its current position and,
      IF it is too high (X<36) THEN locate it back down to dot
      row 35, if it was on dot row 35 this locate statement
      won't have any visual effect on the screen, and
      continue with end_tank_loop.
      ELSE set it into a diagonally upward motion (forward or
         backward according to the value of V), and GOTO
         TANK_LOOP (start the loop over again).

```

(Note: if an invalid key was pressed then none of the above IF statements would be executed and the value for V will remain unchanged and the program will drop through to end_tank_loop)

```

END_TANK_LOOP Adjust our chopper's motion according to the value of V and
1120 GOTO TANK_LOOP (start the loop over again).

```

LASER_FIRE

1130 CALL SOUND(-999,-8,3,128,30,128,30,999,30):: CALL MOTION(#1,0,V):: CALL POSITION(#1,X,Y):: Y=V/2+Y :: IF Y<1 THEN Y=1

1140 FOR T=X+16 TO 175 STEP 13 :: Y=Y+13 :: IF Y>255 THEN Y=1

1150 CALL LOCATE(#T/13+7,T,Y):: NEXT T :: IF Y1 THEN CALL COINC(#2,160,Y,18,T):: IF T THEN CALL DELSPRITE(#2)

1160 CALL SOUND(-999,-8,3,128,30,128,30,500,30):: CALL COINC(ALL,T):: CALL DELSPRITE(#9,#10,#11,#12,#13,#14,#15,#16,#17,#18,#19,#20)

1170 IF T THEN CALL COLOR(#3,16):: GOSUB 850 :: GOTO 1040 ELSE 1050

LASER_FIRE

- 1130 Generate the Laser Fire sound, change our chopper's motion to level flight V and check its position. Next adjust the dot column variable slightly to compensate for our chopper's level flight motion. This allows the laser to always start underneath the chopper no matter which direction it is moving in.
IF after the adjustment, we are left with a value less than 1, because our chopper is moving backwards and on the left hand edge of the screen, THEN change it to 1.
- 1140 Start the laser fire For-Next loop 16 pixels lower than our choppers position. This also determines how many laser sprites to bring out. The closer our chopper is to the ground the fewer the sprites we need to display. This loop increments in units of 13 to compensate for the double line laser pattern. If you look in the appendix for the definition of character number 136, laser fire, you will notice that only 13 pixel rows were used. If you use a single line laser you can use all 16 pixel rows without worrying about sprite pixel overlap. The end of the loop at 175 assures that the laser will be displayed all the way down to the 4 invisible sprites at the bottom of the screen, which have a lower number than the laser sprites. This automatically ends the laser at the same position on the ground no matter where our chopper is in the air since any part of the laser that resides in the same dot rows will not be seen. On each loop we also add 13 to Y (the dot column) to make the laser come out diagonally.
IF Y is greater than 255 we need to wrap the laser around to the other side of the screen.
- 1150 Bring out a laser fire sprite. At the end of the TANK_INIT routine we placed sprite numbers 9 through 20 off the bottom of the screen for the laser fire. This was done because a CALL LOCATE works MUCH faster than a CALL SPRITE, so now with them defined and off the screen we can just use a CALL LOCATE to display them. Keep looping until the For-Next loop is done.
IF our man is on the screen (Y1<>0), THEN check the coincidence between him and the last laser sprites position from the For-Next loop (Y).
IF we hit him THEN delete him off the screen. Note: You could make the game much harder here by clearing out the total score if you shoot one of our men or you could penalize the player by blowing up the chopper or.....? its up to you.
- 1160 Generate the end of the laser fire sound and check for any coincidence (we already checked to see if we hit the man so if there is a coincidence it must be the tank) and delete the laser fire off the screen. By using CALL DELSPRITE(#x) instead of (ALL) we will not actually delete the laser sprites we will just move them off the visible portion of the screen. This method works faster than a For-Next loop to delete the sprites.
- 1170 IF we hit the tank THEN change its color to white, GOSUB BLOW_UP_TANK and GOTO TANK_SCORE to display the score and start the tank_loop over again.
ELSE since we didn't hit the tank there's no need to display the score so just goto TANK_LOOP to start it over again.

MORE CALL PEEKs & CALL LOADs

Listed below are some new CALL PEEKs and CALL LOADs. Along with these you will also find the CALL PEEKs and CALL LOADs that were in the Smart Programming Guide for Sprites and the Smart Programmer newsletter.

CALL PEEK (Extended Basic)

CALL PEEK(-28672,A)::IF A=0 OR A=127 The speech synthesizer is NOT attached

RANDOMIZE :: CALL PEEK(-31880,A) Random Integers 0-99

RANDOMIZE :: CALL PEEK(-31808,A,B) Double Random Integers 0-255

CALL PEEK(-31879) VDP Interrupt Timer

CALL PEEK(-31878) Highest # Sprite in Auto-Motion

CALL PEEK(-31877) VDP Status Register

CALL PEEK(8198,A,B)::IF A/B=2 THEN CALL INIT has been executed
or IF A*256+B=43605
or IF A=170 AND B=85

CALL PEEK(8194,A,B,C,D)::(C-A)*256+D-B = Free Space in Low Memory after CALL
INIT or CALL LOAD("DSKx.xxxxxx")

CALL PEEK(-31974,A,B) :: A*256+B-2487 = Running free space in VDP Ram. Note:
FOR - NEXT LOOPS, GOSUBs etc. use
running space, garbage collection
recovers it. This PEEK will not
ALWAYS return EXACT amount of free
VDP Space unless Garbage collection
has JUST been accomplished. (SIZE
performs garbage collection before
reporting STACK Free Space)

CALL PEEK(-31936,A,B) :: A*256+B-2487 = Exact amount of Free Stack space
while the program is running. Does
not count the garbage collection
area as used.

CALL PEEK(-31866,A,B) :: A*256+B-41023 = Free Program space in High Memory

CALL PEEK(-31952,A,B) :: A*256+B = Start of Line number Table - Without
Mem-Expansion this points into VDP
Ram. With Mem-Expansion this points
into High Mem-Expansion.

CALL PEEK(-31950,A,B) :: A*256+B = End of Line Number Table - points to
the last byte of the line number
table.

CALL PEEK(-31954,A,B) :: A*256+B = The memory address of the pointer to
the current line being executed.

CALL PEEK (Extended Basic) Continued

CALL PEEK(-31954,A,B) ::

CALL PEEK(A*256+B-65536,C,D):: C*256+D = Start address of current program
line being executed.

CALL PEEK(-31954,A,B) ::

CALL PEEK(A*256+B-65538,C,D):: C*256+D = Current line number being executed.

CALL PEEK(-31952,A) :: IF A=55 THEN No Memory Expansion

CALL LOAD (Extended Basic)

CALL LOAD(-31962,0,32) Execute Power Up Routine - Go To Title Screen
does not close open files.

CALL LOAD(-31962,33,111) Hop directly into TI Basic

CALL LOAD(-31962,99,114) Restart Extended Basic - try to reload DSK1.LOAD

CALL LOAD(-31962,101,190) Execute LIST command - from command mode only

CALL LOAD(-31962,100,155) Execute RUN command

CALL LOAD(-31962,100,124) Execute NEW command

CALL LOAD(-31962,100,126) Execute CONTINUE command - from command mode only

CALL LOAD(-31962,100,128) Another LIST command - from command mode only

CALL LOAD(-31962,100,130) Execute BYE command (closes all open files)

CALL LOAD(-31962,100,132) Execute default NUM command - when running
program ends. Line 100 contains garbage so just
place a REM there.

CALL LOAD(-31962,100,136) Execute default RESEQUENCE command

CALL LOAD(-31962,160,000) Generates colorful Title Screen

CALL LOAD(-31962,160,04) Execute RUN without Pre-Scan (Faster than having
a RUN command in your program to restart it.)

CALL LOAD(-31806,128) Disables Auto Sprite motion, Auto Sound and the
QUIT Key

CALL LOAD(-31806,64) Disables Auto Sprite motion - brings ALL moving
Sprites to an immediate stop.

CALL LOAD(-31878,0) Brings ALL moving Sprites to an immediate stop -
placing a value in here between 1 and 28 allows
only the sprite numbers that are equal to or less
than that number to be in auto motion.



CALL LOAD (Extended Basic) Continued

CALL LOAD(-31806,32) Disables Auto Sound processing - leaves the sound on forever.

CALL LOAD(-31806,16) Disables the QUIT key

CALL LOAD(-31806,0) Enables Auto Sprite motion, Auto sound processing and the QUIT key.

CALL LOAD(-31744,x,x,x,x) Sound chip location, different values turn on different sounds.

CALL LOAD(-27648,x,x,x) Speech chip location

CALL LOAD(-31868,0,0)::
RUN "DSKx.xxxx" Turns OFF Memory Expansion

CALL LOAD(-31868,255,231)::
RUN "DSKx.xxxx" Turns ON Memory Expansion

APPENDIX - A

CASSETTE JOYSTICK VERSION

```
10 CALL CLEAR :: CALL MAGNIF
Y(3):: CALL SCREEN(2):: GOTO
30 :: A$,B$,W$,B :: CALL KE
Y :: CALL JOYST :: CALL SOUN
D :: CALL PEEK :: CALL HCHAR
:: CALL VCHAR
```

```
20 X,Y,Y1,YY,K,V,Z,HP,SP,SC
:: CALL POSITION :: CALL PAT
TERN :: CALL SPRITE :: CALL
DELSPRITE :: CALL COINC :: C
ALL MOTION :: CALL LOCATE
```

```
30 E$="00000000" :: T$=E$&"0
000" :: S$=E$&E$ :: CALL CHA
R(132,"6152524CCCC0000080808
OFF809C84FC"&E$&"FF"&T$&"FF"
):: FOR T=1 TO 8 :: CALL COL
OR(T,15,1):: NEXT T :: !@P-
```

```
35 FOR T=0 TO 27 STEP 3 :: D
ISPLAY AT(12,2):"RELEASE THE
ALPHA LOCK KEY" :: CALL SOU
ND(-140,550,T,557,T):: DISPL
AY AT(12,1):: NEXT T
```

```
40 CALL CHAR(47,"3C4299A1A19
9423C",33,RPT$("FF",8)&"0000
002011B3FFFFFFFFFFFF817E7E81
",112,"A00802103979FDF1F070
1080701"&E$&"80D084C1F0FCF2E
2E2F2FC9CFA1C")
```

```
50 DISPLAY AT(5,3):"N I G H
T M I S S I O N": : :RPT$(
" ",13)&" " &RPT$(" ",13)::
CALL COLOR(13,12,1):: GOSUB
240 :: CALL SPRITE(#1,112,1
3,46,25,0,11)
```

```
60 DISPLAY AT(12,14):"BY": :
" MIKE MC CUE & CRAIG MILLER
": : :TAB(12),"/ 1983": : "
MILLERS GRAPHICS": "
1475 W CYPRESS AVE": "
SAN DIMAS CA 91773"
```

```
70 CALL CHAR(136,"9048241209
0402010"&E$&S$&"080402090482
41209",96,"10000040"&E$&"80"
&E$&"2"&S$&"104000001000008"
,100,"0000008"&S$&"000004"&S
$&"2"&T$&"04")
```

```
80 CALL CHAR(120,"030E3F3C7F
F7FFFFFF6F7F7F33360F03E0583C
F8F8E8F0B0F0F0F07078E81CFC",
124,"0008080907"&RPT$("01",1
1)&"00C0C0C0E0E0E0E0C04040
404040E0")
```

```
90 A$=RPT$(" ",22):: CALL CH
AR(128,"0202040503"&RPT$("01
",11)&"00C0C0C0E0E0E0E0C04
040404040E0",140,T$&"0103073
F6AAAAA7F"&S$&"80C0E0FCAAA9A
AFC")
```

```
100 A$=A$&"/ e e e /,
e e .(- e e .((-
e /..((( e e .(((
- .((((((- / .((((
{, e.((((((- .((-,((((
((-/.(((((((((((,"
```

```
110 GOSUB 240 :: B$=RPT$("1",
,28)&RPT$("11",9)&RPT$("1",
29):: W$=RPT$(" ",28)&RPT$("
1",84)
```

```
120 CALL CHAR(92,"0008000020
00000400000001"&E$&"80"&E$&
"208000002001",116,"00005501
0F1E183020301E1F0F1320300000
5580F078180C040C78F8F0C8040C
")
```

```
130 CALL CLEAR :: FOR T=1 TO
13 :: CALL COLOR(T,1,1):: N
EXT T :: FOR X=1 TO 30 :: RA
NDOMIZE :: CALL PEEK(-31808,
T,Y):: CALL HCHAR(T/18+1,Y*.
12+1,101):: NEXT X
```

```
140 GOSUB 240 :: CALL CHAR(8
8,"000004001000000020000008"
&E$&"001"&E$&"410000004001",
56,"1F112223E44444F81F21213E
02040408")
```

```
150 CALL CHAR(48,"1F21214242
8484F801010202040408081F0101
023C20407E3F0103021C0408F8",
52,"1111223E020404081F202040
7C0408F8102020407E82827C3F01
020408102040")
```

```
160 CALL CHAR(80,E$&"0003040
1010101010202040600000018F8E
0F8C0C0C070101018",84,"00000
00200080000100002"&T$&"00200
00082000000082")
```

```
170 CALL CHAR(72,"0000010107
05030101010101020203C0C0C0
E0E0F8C0C0C02020101020406"):
: CALL HCHAR(20,1,34,32)
```

```
180 CALL CHAR(36,"140A200D1A
21641A241A0814080800080"&S$,
76,"000001030303030101010100
00070400C0C0C0E0E0E0F0C0C040
40C0C0A080C"): : CALL HCHAR(2
1,1,33,128)
```

```
190 DISPLAY AT(10,1):"e h i
j k e j l j m n e k n e" :: CA
LL CHAR(40,RPT$("FF",8),44,"
00008082C2C7F7FF8080C0E0E4FC
FEFF0103232B7F7FF7FF"&E$&"08
28A9FD")
```

```
200 DISPLAY AT(2,1):RPT$(" "
,13)&" " &RPT$(" ",13):: DIS
PLAY AT(3,24):"xz" :: DISPLA
Y AT(4,24):"y{" :: DISPLAY A
T(15,1):A$:B$ :: A$=SEG$(A$,
141,28)&B$
```

```
210 GOSUB 240 :: B$="e # ###
# e###e###e### e### e" :: CAL
L DELSPRITE(ALL):: CALL COLO
R(3,15,1,4,15,1,9,11,1,2,11,
1,1,11,1,12,15,1,13,12,1)
```

```
220 CALL COLOR(10,1,1):: CAL
L CHAR(108,E$&"55000061E1FFF
FFF00000003"&E$&"5540E0F8E4E
2E1F1FF7C45FE")
```

```
230 K=1 :: Z=5 :: HP,SP,SC,B
=0 :: CALL CHAR(104,E$&"0104
10400103CFFFFFF786001020820C0
707CE2E1E1F3FEFCC54658E"): :
GOTO 740
```

```
240 CALL SOUND(-4250,-4,1,11
0,30,110,30,200,30):: RETURN
```

```
250 CALL KEY(3,T,Y):: Z=INT(
(SC-B)/10000):: FOR T=1 TO Z
:: CALL SOUND(200,770,4,777
,6):: DISPLAY AT(1,24):USING
"###":T :: NEXT T
```

```
260 IF Z THEN B=B+Z*10000 ::
GOTO 740 ELSE CALL CHAR(108
,"FF81BFA0AFB981FFFF81E71818
E781FFE7B5B5BDBDADADE7")
```

```
270 CALL CHAR(104,"FF81BD81B
FA0A0E0E0A0A0A0A0B81FFFF81B
D81BD81BD81E7A5BD81E7181818"
):: CALL SPRITE(#1,112,13,87
,1,0,12)
```

```
280 CALL COLOR(10,7,1,10,9,1
,10,16,1,10,6,1):: CALL KEY(
1,T,Y1):: CALL KEY(0,T,T)
```

```
290 IF T=89 OR Y1 THEN 220 E
LSE IF T=78 THEN CALL DELSPR
ITE(ALL):: CALL VCHAR(1,1,32
,768):: END ELSE 280
```

```
300 V=8 :: IF K<2 THEN 570 E
LSE IF Y AND 1 THEN 340 ELSE
CALL CHAR(60,"08081C1C1C1C1C
C3E7F1C0008221004080"&S$)
```

```
310 YY=600 :: CALL DELSPRITE
(ALL):: FOR T=2 TO 5 :: CALL
LOCATE(#T,1,T*17,#T+4,177,T
*17):: NEXT T
```

```
320 CALL SOUND(-350,-7,6,110
,5):: CALL CHAR(35,"FFFFFFF
81000081"): : CALL SOUND(4250
,-8,4,110,27,115,28,YY,30)
```

```
330 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,177,T*24-208,-X/8-3-K,0):
: NEXT T :: GOTO 360
```

```
340 CALL CHAR(60,T$&"01030F7
F0"&E$&T$&"709FA8204FC3C1CFF
0C"): : YY=1600 :: CALL SOUND
(-4250,-8,6,110,27,115,28,YY
,30):: CALL DELSPRITE(ALL)
```

```
350 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,T*16-120,256,0,-X/8-3-K):
: NEXT T
```

```
360 CALL SPRITE(#1,112,13,72
,1,0,3)
```

```
370 CALL SOUND(-999,-8,6,110
,27,115,28,YY,30):: CALL COI
NC(ALL,T):: CALL POSITION(#1
,X,Y):: IF T OR X>161 THEN G
OSUB 870 :: GOTO 740
```

```
380 IF Y>224 THEN 410 ELSE C
ALL JOYST(1,T,Y1):: IF Y1 TH
EN CALL PATTERN(#1,108):: CA
LL MOTION(#1,(X-2*Y1>30)*2*Y
1,2):: GOTO 370
```

```
390 CALL PATTERN(#1,112):: C
ALL MOTION(#1,0,T/2+2):: GOT
O 370
```



CASSETTE JOYSTICK VERSION Continued

```

410 CALL DELSPRITE(#1):: IF
K<3 THEN 570 ELSE Y1=(Y AND
6)-4 :: CALL DELSPRITE(ALL):
: CALL COLOR(2,1,1,1,1,13,
1,1)

420 DISPLAY AT(21,1):" e
e e e e e
e
e"

430 CALL CHAR(60,T$&"AA021F2
C4C7F107F"&S$&"A800C1FFE1C09
0E",64,S$&"00009292"&T$&S$&"
4949"):: CALL SPRITE(#1,108,
13,40,31,8,0)

440 CALL SPRITE(#2,60,6,242,
216,(Y AND 1)*80-40,0)

450 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#1,X,Y,#2,YY,Y)

460 IF X>180 THEN CALL LOCAT
E(#1,1-(V<0)*180,31)ELSE IF
ABS(X-YY)<7 THEN 490 ELSE CA
LL MOTION(#2,SGN(X-YY)*((YY
AND 11)+9),0)

470 CALL KEY(1,T,Y):: IF Y T
HEN 510 ELSE CALL JOYST(1,Y,
T):: IF T THEN V=-2*T ELSE I
F Y THEN V=0 ELSE 450

480 CALL MOTION(#1,V,0):: GO
TO 450

490 CALL SPRITE(#3,64,7,YY,2
09,V+X-YY,-127):: CALL SOUND
(-900,-8,1,110,30,110,30,999
9,30):: T=0

500 CALL POSITION(#3,Y,Y)::
IF Y>50 THEN 500 ELSE CALL D
ELSPRITE(#3):: CALL COLOR(#1
,11):: GOSUB 930 :: CALL DEL
SPRITE(#2):: GOTO 740

510 CALL SOUND(-900,-8,0,110
,30,110,30,300,30):: CALL PO
SITION(#1,X,Y):: CALL SPRITE
(#3,64,4,X,36,0,127)

520 CALL POSITION(#3,X,Y)::
IF Y<192 THEN 520 ELSE CALL
DELSPRITE(#3):: CALL COINC(#
2,X,220,7,T):: IF T THEN CAL
L SOUND(-1,-4,9)ELSE 450

530 CALL COLOR(#2,15):: FOR
T=0 TO 2 :: CALL PATTERN(#2,
88+T*4):: FOR X=5 TO 7 :: CA
LL SOUND(100,-X,T*10):: NEXT
X :: NEXT T

540 CALL DELSPRITE(#2):: Y1=
Y1+1 :: SC=SC+250 :: DISPLAY
AT(1,12)SIZE(10):USING "###
#####":SC :: IF Y1<5 THEN
440

550 CALL POSITION(#1,X,Y)::
IF X>185 THEN CALL LOCATE(#1
,35,Y)

560 CALL PATTERN(#1,112):: C
ALL MOTION(#1,0,20):: CALL S
OUND(-4250,-4,1,200,30,200,3
0,200,30)

570 CALL CHAR(68,S$&"000000F
F7F1FFFEAB"&S$&"000000FFFFFFFA
A6D",64,E$&"0021F373FFFFFFF
FFFF972A08103CFFFOFFFFFFF
FFFFFFF5EAA")

580 Y1=(Y AND 5)+6 :: GOSUB
800 :: CALL CHAR(60,"0000008
080C4CFREFFFF5BAAFFFF722B"&T
$&"C00FFFEFCF8F0E2C67B3D")::
CALL COLOR(1,1,1,2,1,1)

590 DISPLAY AT(20,1):W$ :: C
ALL COLOR(1,5,1,7,12,1):: FO
R T=1 TO 4 :: CALL LOCATE(#T
,161,1):: NEXT T :: CALL SPR
ITE(#9,112,13,15,1,20,35)

600 CALL SPRITE(#10,68,15,14
3,1,#11,64,15,143,17,#12,60,
15,143,33):: CALL MOTION(#10
,0,Y1,#11,0,Y1,#12,0,Y1)

610 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#9,X,Y):: IF X<35 TH
EN CALL MOTION(#9,0,V):: CAL
L LOCATE(#9,35,Y)ELSE IF X>1
40 THEN 690

620 CALL COINC(#9,#11,16,T)::
IF T AND X>130 THEN 680 EL
SE CALL COINC(#9,#10,9,T)::
IF T THEN 710

630 CALL JOYST(1,Y,T):: IF Y
THEN CALL PATTERN(#9,Y+108)
:: V=V-Y*(ABS(Y+V)<16):: CAL
L MOTION(#9,0,V):: GOTO 610

640 IF T THEN CALL PATTERN(#
9,108):: CALL MOTION(#9,2*T*
(X-T>35),V):: GOTO 610

650 CALL PATTERN(#9,SGN(V)*4
+108):: CALL MOTION(#9,(X>35
)*3,V):: GOTO 610

680 CALL MOTION(#9,0,0,#10,2
,4,#11,2,4,#12,2,4):: SC=SC-
SP*500 :: SP=0

690 CALL MOTION(#9,3,0):: CA
LL SOUND(-2450,-8,6,110,30,1
10,30,9999,30):: CALL PATER
N(#9,104):: CALL SOUND(1,-4,
9):: HP=0 :: Z=Z-1

700 CALL DELSPRITE(#9):: IF
SP THEN 740 ELSE CALL DELSPR
ITE(#10,#11,#12):: GOTO 740

710 CALL POSITION(#10,X,Y)::
CALL SPRITE(#9,116,13,136,Y
,0,Y1):: FOR T=306 TO 122 ST
EP -6 :: CALL SOUND(-200,-4,
1,T,27,T,30,T,30):: NEXT T

720 FOR T=130 TO 306 STEP 8
:: CALL SOUND(-4250,-4,1,T,3
0,T,30,T,27):: NEXT T :: CAL
L MOTION(#9,-10,0):: K=K+1 ::
SC=SC+500*HP

730 FOR T=1 TO 300 :: NEXT T
:: SP=SP+HP :: HP=0

740 DISPLAY AT(1,2):USING B$
:HP,SP,SC,Z :: IF Z THEN CAL
L DELSPRITE(ALL)ELSE 250 111

750 CALL CHAR(60,"000001010F
090909010101013E200000C0C4C4
FCC0C0C0C0C0C02010080810203")

760 CALL CHAR(64,"0101031F13
1313030303033F20000000808888
88F8808080808000008040203")

770 CALL CHAR(68,"0000010107
0909050101010106181000C0C0C2
F4C8C0C0C0C0C02010080810203",3
5,"FFFFFFF817E7E81")

780 DISPLAY AT(20,1):A$ :: C
ALL COLOR(2,11,1,1,11,1,13,1
2,1):: GOSUB 800

790 CALL SPRITE(#3,140,2,161
,256,0,-24):: FOR T=9 TO 20
:: CALL SPRITE(#T,136,7,200,
1):: NEXT T :: GOTO 1030

800 CALL DELSPRITE(ALL):: FO
R T=5 TO 8 :: CALL LOCATE(#T
,177,T*17):: NEXT T :: RETUR
N

810 IF T<>12 THEN CALL POSIT
ION(#1,X,Y,#3,YY,YY):: IF AB
S(Y-YY)<80 THEN CALL SPRITE(
#4,36,9,157,YY,X-147,2*V+Y-Y
Y)ELSE 1060 ELSE 1060

820 CALL SOUND(-150,-8,3,110
,30,110,30,5010,30):: CALL S
OUND(300,-8,1,128,30,128,30,
1100,30):: CALL DELSPRITE(#4
):: GOSUB 870 :: GOTO 1030

830 CALL MOTION(#3,0,V/4,#2,
0,0):: CALL PATTERN(#2,80)::
CALL SOUND(-1,-4,9):: IF T=
12 THEN CALL MOTION(#1,0,V)

840 FOR T=1 TO 9 :: CALL SOU
ND(50,-6,1):: NEXT T :: CALL
DELSPRITE(#2):: Y1=0 :: RET
URN

850 CALL COLOR(#3,2):: FOR T
=0 TO 16 STEP 4 :: CALL SOUN
D(-999,-8,T,120,27,127,28,10
00,30):: CALL PATTERN(#3,84+
T):: NEXT T

860 SC=605-3*X+SC :: CALL DE
LSPRITE(#3):: CALL SPRITE(#3
,140,2,161,256,0,-24):: RETU
RN

870 T=0 :: CALL SOUND(-1,-4,
9):: IF Y AND 1 THEN CALL PA
TTERN(#1,112):: CALL MOTION(
#1,9,V)ELSE CALL COLOR(#1,16
):: GOTO 930

880 CALL COLOR(#1,RND*7+3)::
FOR T=1 TO 26 STEP 5 :: CAL
L SOUND(T*40+200,-8,T,110,30
,110,30,1100-T,30)

890 CALL POSITION(#1,X,Y)::
IF X>155 THEN CALL MOTION(#1
,0,0):: GOTO 910 ELSE IF X A
ND 2 THEN 880

```



CASSETTE JOYSTICK VERSION Continued

```

900 NEXT T :: GOTO 880
910 CALL COINC(#1,#2,16,T)::
  IF T THEN CALL DELSPRITE(#2
)
920 CALL COINC(#1,#3,17,T)::
  CALL COLOR(#1,2):: IF T THE
N CALL SOUND(-300,-8,1,110,3
0,110,30,3000,30):: CALL COL
OR(#3,7)
930 FOR Y=0 TO 16 STEP 4 ::
CALL SOUND(-999,-8,Y,120,27,
127,28,1100,30):: CALL PATTE
RN(#1,84+Y):: NEXT Y
940 CALL DELSPRITE(#1):: Z=Z
-1 :: HP=0 :: IF T THEN GOSU
B 850 :: RETURN ELSE RETURN
950 CALL MOTION(#1,0,V):: CA
LL COINC(#1,#2,12,YY):: IF Y
Y THEN CALL SOUND(-500,-8,1,
110,30,110,30,840,28):: CALL
DELSPRITE(#2)
960 CALL SOUND(-4250,-4,1,14
0,30,140,30,140,30):: CALL P
OSITION(#1,X,Y,#2,Y1,Y1):: C
ALL SPRITE(#1,116,13,160,Y,0
,0)
970 CALL MOTION(#2,0,4*SGN(Y
-Y1)):: IF Y1=0 THEN CALL SP
RITE(#2,76,2,163,256)
980 CALL JOYST(1,T,Y1):: IF
Y1>0 THEN 1020 ELSE YY=YY-4
:: IF YY<60 THEN YY=76
990 CALL PATTERN(#2,YY):: CA
LL COINC(#1,#3,32,T):: IF T
THEN GOSUB 830 :: GOSUB 870
:: GOTO 1030
1000 CALL COINC(#2,#3,24,T)::
: IF T THEN GOSUB 830 :: GOT
O 970 ELSE CALL COINC(#1,#2,
11,T):: IF T THEN CALL PATTE
RN(#2,80)ELSE 980
1010 CALL SOUND(-200,220,7,2
23,8,226,9):: HP=HP+1 :: DIS
PLAY AT(1,3)SIZE(2)::HP :: CA
LL DELSPRITE(#2)
1020 CALL SOUND(-4000,-4,1,1
10,30,110,30,320,30):: CALL
MOTION(#1,-17,V/2,#2,0,0)::
CALL PATTERN(#2,128):: IF HP
=5 THEN 300 ELSE 1050
1030 IF Z THEN CALL SPRITE(#
1,112,13,20,1,20,35):: T,V=1
2
1040 DISPLAY AT(1,2):USING B
$:HP,SP,SC,Z :: IF Z=0 THEN
250
1050 CALL SOUND(-999,-4,1,11
0,30,110,30,200,30):: CALL P
OSITION(#1,X,Y,#2,Y1,Y1,#3,Y
Y,YY):: IF ABS(YY-Y)<5 THEN
810 ELSE IF Y1 THEN CALL PAT
TERN(#2,124)
1060 CALL MOTION(#3,0,SGN(Y-
YY)*((Y AND 6)+4+K)):: IF X<
35 THEN CALL MOTION(#1,0,V)::
: CALL LOCATE(#1,35,Y)ELSE I
F X>151 THEN 950
1070 IF Y1 THEN CALL PATTERN
(#2,128):: IF ABS(YY-Y1)<26
THEN GOSUB 830
1080 CALL KEY(1,YY,T):: IF T
THEN 1130 ELSE CALL JOYST(1
,T,YY):: IF T OR YY THEN CAL
L PATTERN(#1,108+T)ELSE CALL
PATTERN(#1,V/3+108):: CALL
MOTION(#1,3*(X>35),V):: GOTO
1050
1090 IF T THEN V=T*3 :: CALL
MOTION(#1,0,V):: GOTO 1050
ELSE CALL MOTION(#1,YY*2*(X-
YY>31),V):: GOTO 1050
1130 CALL SOUND(-999,-8,3,12
8,30,128,30,999,30):: CALL M
OTION(#1,0,V):: CALL POSITIO
N(#1,X,Y):: Y=V/2+Y :: IF Y<
1 THEN Y=1
1140 FOR T=X+16 TO 175 STEP
13 :: Y=Y+13 :: IF Y>255 THE
N Y=1
1150 CALL LOCATE(#T/13+7,T,Y
):: NEXT T :: IF Y1 THEN CAL
L COINC(#2,160,Y,18,T):: IF
T THEN CALL DELSPRITE(#2)
1160 CALL SOUND(-999,-8,3,12
8,30,128,30,500,30):: CALL C
OINC(ALL,T):: CALL DELSPRITE
(#9,#10,#11,#12,#13,#14,#15,
#16,#17,#18,#19,#20)
1170 IF T THEN CALL COLOR(#3
,16):: GOSUB 850 :: GOTO 104
0 ELSE 1050

```

APPENDIX - B

DISK LOADER

```
10 CALL CLEAR :: CALL MAGNIF
Y(3):: CALL SCREEN(2):: GOTO
30 :: YY,X,Y
```

```
20 A$ :: CALL KEY :: CALL PE
EK :: CALL HCHAR :: CALL SPR
ITE :: CALL SOUND :: CALL DE
LSprite
```

```
30 E$="00000000" :: T$=E$&"0
000" :: S$=E$&E$ :: CALL CHA
R(132,"6152524CCCC0000080808
OFF809C84FC"&E$&"FF"&T$&"FF"
):: FOR T=1 TO 8 :: CALL COL
OR(T,15,1):: NEXT T :: I@P-
```

```
40 CALL CHAR(47,"3C4299A1A19
9423C",33,RPT$("FF",8)&"0000
002011B3FFFFFFFFFFFF817E7E81
",112,"A00802103979FDF1F070
1080701"&E$&"80D084C1F0FCF2E
2E2F2FC9CFA1C")
```

```
50 DISPLAY AT(5,3):"N I G H
T M I S S I O N": : :RPT$(
" ",13)&" "&RPT$(" ",13)::
CALL COLOR(13,12,1):: GOSUB
290 :: CALL SPRITE(#1,112,1
3,46,25,0,11)
```

```
60 DISPLAY AT(14,1):" PRESS
TO USE": : : " 1 THE NU
MBER 1 JOYSTICK": : : " 2 T
HE ARROW KEYS (ESDX)": : "
AND V TO FIRE"
```

```
70 X=X+1 :: CALL KEY(0,YY,T)
:: IF T=0 THEN IF X<90 THEN
70 ELSE GOSUB 290 :: GOTO 70
```

```
80 IF YY<49 OR YY>50 THEN DI
SPLAY AT(14,1):: CALL SOUND(
-100,200,4,208,6):: X=90 ::
GOTO 60 ELSE IF YY=49 THEN 1
00
```

```
90 FOR X=0 TO 27 STEP 3 :: C
ALL SOUND(-140,550,X,557,X):
: DISPLAY AT(14,1): : : :
: : : : : DISPLAY AT(16,2
):"RELEASE THE ALPHA LOCK KE
Y" :: NEXT X
```

```
100 GOSUB 290 :: DISPLAY AT(
12,14):"BY": : " MIKE MC CUE
& CRAIG MILLER": : : :TAB(12
);"/ 1985": : " MILLERS
GRAPHICS": " 1475 W CYPRE
SS AVE": " SAN DIMAS CA 9
1773"
```

```
110 CALL CHAR(136,"904824120
90402010"&E$&S$&"08040209048
241209",96,"10000040"&E$&"80
"&E$&"2"&S$&"104000001000008
",100,"0000008"&S$&"000004"&
S$&"2"&T$&"04")
```

```
120 CALL CHAR(120,"030E3F3C7
FF7FFFFFF6F7F7F33360F03E0583
CF8F8E8F0B0F0F0F07078E81CFC"
,124,"0008080907"&RPT$("01",
11)&"00C0C0C0E0E0E0E0E0C0404
0404040E0")
```

```
130 CALL CHAR(128,"020204050
3"&RPT$("01",11)&"00C0C0C0E0
E0E0E0E0C04040404040E0",140,
T$&"0103073F6AAAAA7F"&S$&"80
C0E0FCAA9AAFC")
```

```
140 CALL CHAR(92,"0008000020
000000400000001"&E$&"80"&E$&
"208000002001",116,"00005501
0F1E183020301E1F0F1320300000
5580F078180C040C78F8F0C8040C
")
```

```
150 A$="e e e
e e e
e "
```

```
160 CALL CLEAR :: FOR T=1 TO
8 :: CALL COLOR(T,1,1):: NE
XT T :: GOSUB 290
```

```
170 DISPLAY AT(3,24):"xz" ::
DISPLAY AT(4,24):"y{" :: CA
LL COLOR(12,9,1,13,1,1):: CA
LL HCHAR(20,1,34,32)
```

```
180 CALL HCHAR(21,1,33,128):
: FOR X=1 TO 30 :: RANDOMIZE
: : CALL PEEK(-31808,T,Y)::
IF T>26 AND T<64 AND Y>204 A
ND Y<221 THEN T=T+40
```

```
190 CALL HCHAR(T/18+1,Y#.12+
1,101):: NEXT X
```

```
200 DISPLAY AT(10,1):"e h i
j k e j l j m n e k n e"
```

```
210 DISPLAY AT(2,1):RPT$(" "
,13)&" "&RPT$(" ",13):: DIS
PLAY AT(15,1):A$&A$&A$ :: CA
LL COLOR(12,15,1,9,11,1)
```

```
220 GOSUB 290 :: CALL CHAR(8
8,"000004001000000020000008"
&E$&"001"&E$&"410000004001",
56,"1F1122223E4444F81F21213E
02040408")
```

```
230 CALL CHAR(48,"1F21214242
8484F801010202040408081F0101
023C20407E3F0103021C0408F8",
52,"1111223E020404081F202040
7C0408F8102020407E82827C3F01
020408102040")
```

```
240 CALL CHAR(80,E$&"0003040
1010101010202040600000018F8E
0F8C0C0C070101018",84,"00000
00200080000100002"&T$&"00200
00008200000082")
```

```
250 CALL CHAR(72,"0000010107
0503010101010102020203C0C0C0
E0E0F8C0C0C02020101020406")
```

```
260 CALL CHAR(36,"140A200D1A
21641A241A0814080800080"&S$,
76,"000001030303030101010100
00070400C0C0C0E0E0E0F0C0C040
40C0C0A080C")
```

```
270 CALL CHAR(40,RPT$("FF",8
),44,"000808082C2C7F7FF8080C0
E0E4FCFEFF0103232B7F7FFFFF"&
E$&"0828A9FD")
```

```
280 GOSUB 290 :: CALL DELSPR
ITE(ALL):: IF YY=49 THEN RUN
"DSK1.NMJOY" ELSE RUN "DSK1
.NMKEY"
```

```
290 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: X=0 ::
RETURN
```



APPENDIX - C

DISKETTE KEYBOARD VERSION

```
10 CALL MAGNIFY(3):: CALL SC
REEN(2):: GOTO 30 :: CALL KE
Y :: CALL SOUND :: CALL PEEK
:: CALL HCHAR :: CALL VCHAR
```

```
20 X,Y,Y1,YY,V,T :: CALL POS
ITION :: CALL PATTERN :: CAL
L SPRITE :: CALL DELSPRITE :
: CALL COINC :: CALL MOTION
:: CALL LOCATE
```

```
30 E$="00000000" :: T$=E$&"0
000" :: S$=E$&E$
```

```
90 A$=RPT$( " ",22)
```

```
100 A$=A$&"/ e e /,
e e .(- e e .((-
e /..(((, e e .(((
- .(((((((- ./ .(((
(, e.((((((((- .((-,(((
((-/.(((((((((,")
```

```
110 GOSUB 240 :: B$=RPT$( "1"
,28)&RPT$( "1",9)&RPT$( "1"
,29):: W$=RPT$( " ",28)&RPT$(
"1",84)
```

```
200 DISPLAY AT(15,1):A$:B$ :
: A$=SEG$(A$,141,28)&B$
```

```
210 GOSUB 240 :: B$="e # ##
# e##### e### e" :: CAL
L DELSPRITE(ALL):: CALL COLO
R(3,15,1,4,15,1,9,11,1,2,11,
1,1,11,1,12,15,1,13,12,1)
```

```
220 CALL COLOR(10,1,1):: CAL
L CHAR(108,E$&"55000061E1FFF
FFF00000003"E$&"5540E0F8E4E
2E1F1FF7C45FE")
```

```
230 K=1 :: Z=5 :: HP,SP,SC,B
=0 :: CALL CHAR(104,E$&"0104
10400103CFFFFF786001020820C0
707CB2E1E1F3FEFCC54658E")::
GOTO 740 :: 16P-
```

```
240 CALL SOUND(-4250,-4,1,11
0,30,110,30,200,30):: RETURN
```

```
250 CALL KEY(3,T,Y):: Z=INT(
(SC-B)/10000):: FOR T=1 TO Z
:: CALL SOUND(200,770,4,777
,6):: DISPLAY AT(1,24):USING
"###":T :: NEXT T
```

```
260 IF Z THEN B=B+Z*10000 ::
GOTO 740 ELSE CALL CHAR(108
,"FF81BFA0AFB981FFFF81E71818
E781FFE7B5B5BDBDADADE7")
```

```
270 CALL CHAR(104,"FF81BD81B
FACA0E0E0A0A0A0A0CBF81FFFF81B
DBD81BDA5E7E7A5BD81E7181818"
):: CALL SPRITE(#1,112,13,87
,1,0,12)
```

```
280 CALL COLOR(10,7,1,10,9,1
,10,16,1,10,6,1):: CALL KEY(
0,T,T)
```

```
290 IF T=89 THEN 220 ELSE IF
T=78 THEN CALL DELSPRITE(ALL)::
CALL VCHAR(1,1,32,768)::
: END ELSE 280
```

```
300 V=8 :: IF K<2 THEN 570 E
LSE IF Y AND 1 THEN 340 ELSE
CALL CHAR(60,"08081C1C1C1C1
C3E7F1C0008221004080"&S$)
```

```
310 YY=600 :: CALL DELSPRITE
(ALL):: FOR T=2 TO 5 :: CALL
LOCATE(#T,1,T*17,#T+4,177,T
*17):: NEXT T
```

```
320 CALL SOUND(-350,-7,6,110
,5):: CALL CHAR(35,"FFFFFFF
81000081"):: CALL SOUND(4250
,-8,4,110,27,115,28,YY,30)
```

```
330 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,177,T*24-208,-X/8-3-K,0):
: NEXT T :: GOTO 360
```

```
340 CALL CHAR(60,T$&"01030F7
F0"&E$&T$&"709FA8204FC3C1CF
OC"):: YY=1600 :: CALL SOUND
(-4250,-8,6,110,27,115,28,YY
,30):: CALL DELSPRITE(ALL)
```

```
350 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,T*16-120,256,0,-X/8-3-K):
: NEXT T
```

```
360 CALL SPRITE(#1,112,13,72
,1,0,3)
```

```
370 CALL SOUND(-999,-8,6,110
,27,115,28,YY,30):: CALL COI
NC(ALL,T):: CALL POSITION(#1
,X,Y):: IF T OR X>161 THEN G
OSUB 870 :: GOTO 740
```

```
380 IF Y>224 THEN 410 ELSE C
ALL KEY(1,T,T):: IF T<0 THEN
CALL MOTION(#1,0,2):: GOTO
370
```

```
390 IF T=0 THEN T=8 ELSE IF
T=5 THEN T=8*(X>35)ELSE IF T
=3 THEN CALL MOTION(#1,0,4):
: GOTO 370 ELSE CALL MOTION(
#1,0,0):: GOTO 370
```

```
400 CALL MOTION(#1,T,2):: GO
TO 370
```

```
410 CALL DELSPRITE(#1):: IF
K<3 THEN 570 ELSE Y1=(Y AND
6)-4 :: CALL DELSPRITE(ALL):
: CALL COLOR(2,1,1,1,1,1,13,
1,1)
```

```
420 DISPLAY AT(21,1):" e
e e e e
e"
e e
```

```
430 CALL CHAR(60,T$&"AA021F2
C4C7F107F"&S$&"A800C1PFE1C09
0E",64,S$&"00009292"&T$&S$&"
4949"):: CALL SPRITE(#1,108,
13,40,31,8,0)
```

```
440 CALL SPRITE(#2,60,6,242,
216,(Y AND 1)*80-40,0)
```

```
450 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#1,X,Y,#2,YY,Y)
```

```
460 IF X>180 THEN CALL LOCAT
E(#1,1-(V<0)*180,31)ELSE IF
ABS(X-YY)<7 THEN 490 ELSE CA
LL MOTION(#2,SGN(X-YY)*((YY
AND 11)+9),0)
```

```
470 CALL KEY(1,T,Y):: IF T=1
3 THEN 510 ELSE IF T=5 THEN
V=-8 ELSE IF T=0 THEN V=8 EL
SE IF Y THEN V=0
```

```
480 CALL MOTION(#1,V,0):: GO
TO 450
```

```
490 CALL SPRITE(#3,64,7,YY,2
09,V+X-YY,-127):: CALL SOUND
(-900,-8,1,110,30,110,30,999
9,30):: T=0
```

```
500 CALL POSITION(#3,Y,Y)::
IF Y>50 THEN 500 ELSE CALL D
ELSPRITE(#3):: CALL COLOR(#1
,11):: GOSUB 930 :: CALL DEL
SPRITE(#2):: GOTO 740
```

```
510 CALL SOUND(-900,-8,0,110
,30,110,30,300,30):: CALL PO
SITION(#1,X,Y):: CALL SPRITE
(#3,64,4,X,36,0,127)
```

```
520 CALL POSITION(#3,X,Y)::
IF Y<192 THEN 520 ELSE CALL
DELSPRITE(#3):: CALL COINC(#
2,X,220,7,T):: IF T THEN CAL
L SOUND(-1,-4,9)ELSE 450
```

```
530 CALL COLOR(#2,15):: FOR
T=0 TO 2 :: CALL PATTERN(#2,
88+T*4):: FOR X=5 TO 7 :: CA
LL SOUND(100,-X,T*10):: NEXT
X :: NEXT T
```

```
540 CALL DELSPRITE(#2):: Y1=
Y1+1 :: SC=SC+250 :: DISPLAY
AT(1,12)SIZE(10):USING "###
######":SC :: IF Y1<5 THEN
440
```

```
550 CALL POSITION(#1,X,Y)::
IF X>185 THEN CALL LOCATE(#1
,35,Y)
```

```
560 CALL PATTERN(#1,112):: C
ALL MOTION(#1,0,20):: CALL S
OUND(-4250,-4,1,200,30,200,3
0,200,30)
```

```
570 CALL CHAR(68,S$&"000000F
F7F1FFEAB"&S$&"000000FFFFFFFA
A6D",64,E$&"0021F373FFFFFFF
FFFF972A08103CFFFOFFFFFFF
FFFFFFF5EAA")
```

```
580 Y1=(Y AND 5)+6 :: GOSUB
800 :: CALL CHAR(60,"0000008
080C4CFEFFFFFF5BAAFFFFFF722B"&T
$&"C00FEFECF8F0E2C67B3D")::
CALL COLOR(1,1,1,2,1,1)
```

```
590 DISPLAY AT(20,1):W$ :: C
ALL COLOR(1,5,1,7,12,1):: FO
R T=1 TO 4 :: CALL LOCATE(#T
,161,1):: NEXT T :: CALL SPR
ITE(#9,112,13,15,1,20,35)
```

```
600 CALL SPRITE(#10,68,15,14
3,1,#11,64,15,143,17,#12,60,
15,143,33):: CALL MOTION(#10
,0,Y1,#11,0,Y1,#12,0,Y1)
```



DISKETTE KEYBOARD VERSION Continued

610 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#9,X,Y):: IF X<35 TH
EN CALL MOTION(#9,0,V):: CAL
L LOCATE(#9,35,Y)ELSE IF X>1
40 THEN 690

620 CALL COINC(#9,#11,16,T)::
: IF T AND X>130 THEN 680 EL
SE CALL COINC(#9,#10,9,T)::
IF T THEN 710

630 CALL KEY(1,T,T):: IF T<0
THEN CALL PATTERN(#9,SGN(V)
4+108):: CALL MOTION(#9,3(
X>35),V):: GOTO 610

640 IF T=3 THEN CALL PATTERN
(#9,112):: V=V-4*(V<12):: GO
TO 670 ELSE IF T=2 THEN CALL
PATTERN(#9,104):: V=V+4*(V>
-12):: GOTO 670

650 IF T=0 THEN CALL PATTERN
(#9,108):: CALL MOTION(#9,8,
V):: GOTO 610

660 IF T=5 THEN CALL PATTERN
(#9,108):: CALL POSITION(#9,
X,Y):: IF X<36 THEN CALL LOC
ATE(#9,35,Y)ELSE CALL MOTION
(#9,-8,V):: GOTO 610

670 CALL MOTION(#9,0,V):: GO
TO 610

680 CALL MOTION(#9,0,0,#10,2
,4,#11,2,4,#12,2,4):: SC=SC-
SP*500 :: SP=0

690 CALL MOTION(#9,3,0):: CA
LL SOUND(-2450,-8,6,110,30,1
10,30,9999,30):: CALL PATER
N(#9,104):: CALL SOUND(1,-4,
9):: HP=0 :: Z=Z-1

700 CALL DELSPRITE(#9):: IF
SP THEN 740 ELSE CALL DELSPR
ITE(#10,#11,#12):: GOTO 740

710 CALL POSITION(#10,X,Y)::
CALL SPRITE(#9,116,13,136,Y
,0,Y1):: FOR T=306 TO 122 ST
EP -6 :: CALL SOUND(-200,-4,
1,T,27,T,30,T,30):: NEXT T

720 FOR T=130 TO 306 STEP 8
:: CALL SOUND(-4250,-4,1,T,3
0,T,30,T,27):: NEXT T :: CAL
L MOTION(#9,-10,0):: K=K+1 ::
: SC=SC+500*HP

730 FOR T=1 TO 300 :: NEXT T
:: SP=SP+HP :: HP=0

740 DISPLAY AT(1,2):USING B\$
:HP,SP,SC,Z :: IF Z THEN CAL
L DELSPRITE(ALL)ELSE 250 !!!

750 CALL CHAR(60,"000001010F
090909010101013E20000C0C4C4
FCC0C0C0C0C02010080810203")

760 CALL CHAR(64,"0101031F13
1313030303033F20000000808888
88F8808080808000008040203")

770 CALL CHAR(68,"0000010107
0909050101010106181000C0C0C2
F4C8C0C0C0C02010080810203",3
5,"FFFFFFFF817E7E81")

780 DISPLAY AT(20,1):A\$:: C
ALL COLOR(2,11,1,1,11,1,13,1
2,1):: GOSUB 800

790 CALL SPRITE(#3,140,2,161
,256,0,-24):: FOR T=9 TO 20
:: CALL SPRITE(#T,136,7,200,
1):: NEXT T :: GOTO 1030

800 CALL DELSPRITE(ALL):: FO
R T=5 TO 8 :: CALL LOCATE(#T
,177,T*17):: NEXT T :: RETUR
N

810 IF T<>12 THEN CALL POSIT
ION(#1,X,Y,#3,YY,YY):: IF AB
S(Y-YY)<80 THEN CALL SPRITE(
#4,36,9,157,YY,X-147,2*V+Y-Y
Y)ELSE 1060 ELSE 1060

820 CALL SOUND(-150,-8,3,110
,30,110,30,5010,30):: CALL S
OUND(300,-8,1,128,30,128,30,
1100,30):: CALL DELSPRITE(#4
):: GOSUB 870 :: GOTO 1030

830 CALL MOTION(#3,0,V/4,#2,
0,0):: CALL PATTERN(#2,80)::
CALL SOUND(-1,-4,9):: IF T=
12 THEN CALL MOTION(#1,0,V)

840 FOR T=1 TO 9 :: CALL SOU
ND(50,-6,1):: NEXT T :: CALL
DELSPRITE(#2):: Y1=0 :: RET
URN

850 CALL COLOR(#3,2):: FOR T
=0 TO 16 STEP 4 :: CALL SOUN
D(-999,-8,T,120,27,127,28,10
00,30):: CALL PATTERN(#3,84+
T):: NEXT T

860 SC=605-3*X+SC :: CALL DE
LSPRITE(#3):: CALL SPRITE(#3
,140,2,161,256,0,-24):: RETU
RN

870 T=0 :: CALL SOUND(-1,-4,
9):: IF Y AND 1 THEN CALL PA
TTERN(#1,112):: CALL MOTION(
#1,9,V)ELSE CALL COLOR(#1,16
):: GOTO 930

880 CALL COLOR(#1,RND*7+3)::
FOR T=1 TO 26 STEP 5 :: CAL
L SOUND(T*40+200,-8,T,110,30
,110,30,1100-T,30)

890 CALL POSITION(#1,X,Y)::
IF X>155 THEN CALL MOTION(#1
,0,0):: GOTO 910 ELSE IF X A
ND 2 THEN 880

900 NEXT T :: GOTO 880

910 CALL COINC(#1,#2,16,T)::
IF T THEN CALL DELSPRITE(#2
)

920 CALL COINC(#1,#3,17,T)::
CALL COLOR(#1,2):: IF T THE
N CALL SOUND(-300,-8,1,110,3
0,110,30,3000,30):: CALL COL
OR(#3,7)

930 FOR Y=0 TO 16 STEP 4 ::
CALL SOUND(-999,-8,Y,120,27,
127,28,1100,30):: CALL PATTE
RN(#1,84+Y):: NEXT Y

940 CALL DELSPRITE(#1):: Z=Z
-1 :: HP=0 :: IF T THEN GOSU
B 850 :: RETURN ELSE RETURN

950 CALL MOTION(#1,0,V):: CA
LL COINC(#1,#2,12,YY):: IF Y
Y THEN CALL SOUND(-500,-8,1,
110,30,110,30,840,28):: CALL
DELSPRITE(#2)

960 CALL SOUND(-4250,-4,1,14
0,30,140,30,140,30):: CALL P
OSITION(#1,X,Y,#2,Y1,Y1):: C
ALL SPRITE(#1,116,13,160,Y,0
,0)

970 CALL MOTION(#2,0,4*SGN(Y
-Y1):: IF Y1=0 THEN CALL SP
RITE(#2,76,2,163,256)

980 CALL KEY(1,T,T):: IF T=5
THEN 1020 ELSE YY=YY-4 :: I
F YY<60 THEN YY=76

990 CALL PATTERN(#2,YY):: CA
LL COINC(#1,#3,32,T):: IF T
THEN GOSUB 830 :: GOSUB 870
:: GOTO 1030

1000 CALL COINC(#2,#3,24,T)::
: IF T THEN GOSUB 830 :: GOT
O 970 ELSE CALL COINC(#1,#2,
11,T):: IF T THEN CALL PATTE
RN(#2,80)ELSE 980

1010 CALL SOUND(-200,220,7,2
23,8,226,9):: HP=HP+1 :: DIS
PLAY AT(1,3)SIZE(2):HP :: CA
LL DELSPRITE(#2)

1020 CALL SOUND(-4000,-4,1,1
10,30,110,30,320,30):: CALL
MOTION(#1,-17,V/2,#2,0,0)::
CALL PATTERN(#2,128):: IF HP
=5 THEN 300 ELSE 1050

1030 IF Z THEN CALL SPRITE(#
1,112,13,20,1,20,35):: T,V=1
2

1040 DISPLAY AT(1,2):USING B
\$:HP,SP,SC,Z :: IF Z=0 THEN
250

1050 CALL SOUND(-999,-4,1,11
0,30,110,30,200,30):: CALL P
OSITION(#1,X,Y,#2,Y1,Y1,#3,Y
Y,YY):: IF ABS(YY-Y)<5 THEN
810 ELSE IF Y1 THEN CALL PAT
TERN(#2,124)

1060 CALL MOTION(#3,0,SGN(Y-
YY)*((Y AND 6)+4+K):: IF X<
35 THEN CALL MOTION(#1,0,V)::
: CALL LOCATE(#1,35,Y)ELSE I
F X>151 THEN 950



DISKETTE KEYBOARD VERSION Continued

```
1070 IF Y1 THEN CALL PATTERN
(#2,128):: IF ABS(Y1-Y2)<26
THEN GOSUB 830
```

```
1080 CALL KEY(1,T,T):: IF T=
13 THEN 1130 ELSE IF T<0 THE
N CALL PATTERN(#1,V/3+108)::
CALL MOTION(#1,3*(X>35),V):
: GOTO 1050
```

```
1090 IF T=3 THEN CALL PATTEN
N(#1,112):: V=12 :: GOTO 112
0 ELSE IF T=2 THEN CALL PATT
ERN(#1,104):: V=-12 :: GOTO
1120
```

```
1100 IF T=0 THEN CALL PATTEN
N(#1,108):: CALL MOTION(#1,8
,V):: GOTO 1050
```

```
1110 IF T=5 THEN CALL PATTEN
N(#1,108):: CALL POSITION(#1
,X,Y):: IF X<36 THEN CALL LO
CATE(#1,35,Y)ELSE CALL MOTIO
N(#1,-12,V):: GOTO 1050
```

```
1120 CALL MOTION(#1,0,V):: G
OTO 1050
```

```
1130 CALL SOUND(-999,-8,3,12
8,30,128,30,999,30):: CALL M
OTION(#1,0,V):: CALL POSITIO
N(#1,X,Y):: Y=V/2+Y :: IF Y<
1 THEN Y=1
```

```
1140 FOR T=X+16 TO 175 STEP
13 :: Y=Y+13 :: IF Y>255 THE
N Y=1
```

```
1150 CALL LOCATE(#T/13+7,T,Y
):: NEXT T :: IF Y1 THEN CAL
L COINC(#2,160,Y,18,T):: IF
T THEN CALL DELSPRITE(#2)
```

```
1160 CALL SOUND(-999,-8,3,12
8,30,128,30,500,30):: CALL C
OINC(ALL,T):: CALL DELSPRITE
(#9,#10,#11,#12,#13,#14,#15,
#16,#17,#18,#19,#20)
```

```
1170 IF T THEN CALL COLOR(#3
,16):: GOSUB 850 :: GOTO 104
0 ELSE 1050
```

APPENDIX - D

DISKETTE JOYSTICK VERSION

```
10 CALL MAGNIFY(3):: CALL SC
REEN(2):: GOTO 30 :: CALL KE
Y :: CALL JOYST :: CALL SOUN
D :: CALL PEOK :: CALL HCHAR
:: CALL VCHAR
```

```
20 X,Y,Y1,YY,V,T :: CALL POS
ITION :: CALL PATTERN :: CAL
L SPRITE :: CALL DELSPRITE ::
CALL COINC :: CALL MOTION
:: CALL LOCATE
```

```
30 E$="00000000" :: T$=E$&"0
000" :: S$=E$&E$
```

```
90 A$=RPT$(" ",22)
```

```
100 A$=A$&"/ e e /,
e e .(- e e .((-
e e /..((( e e .(((
- .(((((- / .(((
(, e.(((((- .((-,((((
((-/.(((((-(((,(
```

```
110 GOSUB 240 :: B$=RPT$("1"
,28)&RPT$("1",9)&RPT$("1",
29):: W$=RPT$(" ",28)&RPT$("
1",84)
```

```
200 DISPLAY AT(15,1):A$:B$ :
: A$=SEG$(A$,141,28)&B$
```

```
210 B$="e # ### e#####
e### e" :: CALL COLOR(3,15,
1,4,15,1,9,11,1,2,11,1,1,11,
1,12,15,1,13,12,1)
```

```
220 CALL COLOR(10,1,1):: CAL
L CHAR(108,E$&"55000061E1FFF
FFF0000003"&E$&"5540E0F8E4E
2E1F1FF7C45FE")
```

```
230 K=1 :: Z=5 :: HP,SP,SC,B
=0 :: CALL CHAR(104,E$&"0104
10400103CFFFFFF786001020820C0
707CE2E1E1F3FEFCC54658E")::
GOTO 740 :: 1@P-
```

```
240 CALL SOUND(-4250,-4,1,11
0,30,110,30,200,30):: RETURN
```

```
250 CALL KEY(3,T,Y):: Z=INT(
(SC-B)/10000):: FOR T=1 TO Z
:: CALL SOUND(200,770,4,777
,6):: DISPLAY AT(1,24):USING
"###":T :: NEXT T
```

```
260 IF Z THEN B=B+Z*10000 ::
GOTO 740 ELSE CALL CHAR(108
,"FF81BFA0AFB981FFFF81E71818
E781FFE7B5B5BDBDADAE7")
```

```
270 CALL CHAR(104,"FF81BD81B
FA0A0E0E0A0A0A0A0BF81FFFF81B
DBD81BDA5E7E7A5BD81E7181818"
):: CALL SPRITE(#1,112,13,87
,1,0,12)
```

```
280 CALL COLOR(10,7,1,10,9,1
,10,16,1,10,6,1):: CALL KEY(
1,T,Y1):: CALL KEY(0,T,T)
```

```
290 IF T=89 OR Y1 THEN 220 E
LSE IF T=78 THEN CALL DELSPR
ITE(ALL):: CALL VCHAR(1,1,32
,768):: END ELSE 280
```

```
300 V=8 :: IF K<2 THEN 570 E
LSE IF Y AND 1 THEN 340 ELSE
CALL CHAR(60,"08081C1C1C1C1
C3E7F1C0008221004080"&S$)
```

```
310 YY=600 :: CALL DELSPRITE
(ALL):: FOR T=2 TO 5 :: CALL
LOCATE(#T,1,T*17,#T+4,177,T
*17):: NEXT T
```

```
320 CALL SOUND(-350,-7,6,110
,5):: CALL CHAR(35,"FFFFFFF
81000081"):: CALL SOUND(4250
,-8,4,110,27,115,28,YY,30)
```

```
330 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,177,T*24-208,-X/8-3-K,0):
: NEXT T :: GOTO 360
```

```
340 CALL CHAR(60,T$&"01030F7
FO"&E$&T$&"709FA8204FC3C1CFF
OC"):: YY=1600 :: CALL SOUND
(-4250,-8,6,110,27,115,28,YY
,30):: CALL DELSPRITE(ALL)
```

```
350 FOR T=10 TO 18 :: RANDOM
IZE :: CALL PEEK(-31880,X)::
CALL SPRITE(#T,60,(X AND 7)
+3,T*16-120,256,0,-X/8-3-K):
: NEXT T
```

```
360 CALL SPRITE(#1,112,13,72
,1,0,3)
```

```
370 CALL SOUND(-999,-8,6,110
,27,115,28,YY,30):: CALL COI
NC(ALL,T):: CALL POSITION(#1
,X,Y):: IF T OR X>161 THEN G
OSUB 870 :: GOTO 740
```

```
380 IF Y>224 THEN 410 ELSE C
ALL JOYST(1,T,Y1):: IF Y1 TH
EN CALL PATTERN(#1,108):: CAL
L MOTION(#1,(X-2*Y1>30)*2*Y
1,2):: GOTO 370
```

```
390 CALL PATTERN(#1,112):: C
ALL MOTION(#1,0,T/2+2):: GOT
O 370
```

```
410 CALL DELSPRITE(#1):: IF
K<3 THEN 570 ELSE Y1=(Y AND
6)-4 :: CALL DELSPRITE(ALL):
: CALL COLOR(2,1,1,1,1,1,13,
1,1)
```

```
420 DISPLAY AT(21,1):" e
e e e e
e"
e"
e e e e e
```

```
430 CALL CHAR(60,T$&"AA021F2
C4C7F107F"&S$&"A800C1FFE1C09
OE",64,S$&"00009292"&T$&S$&"
4949"):: CALL SPRITE(#1,108,
13,40,31,8,0)
```

```
440 CALL SPRITE(#2,60,6,242,
216,(Y AND 1)*80-40,0)
```

```
450 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#1,X,Y,#2,YY,Y)
```

```
460 IF X>180 THEN CALL LOCAT
E(#1,1-(V<0)*180,31)ELSE IF
ABS(X-YY)<7 THEN 490 ELSE CA
LL MOTION(#2,SGN(X-YY)*((YY
AND 11)+9),0)
```

```
470 CALL KEY(1,T,Y):: IF Y T
HEN 510 ELSE CALL JOYST(1,Y,
T):: IF T THEN V=-2*T ELSE I
F Y THEN V=0 ELSE 450
```

```
480 CALL MOTION(#1,V,0):: GO
TO 450
```

```
490 CALL SPRITE(#3,64,7,YY,2
09,V+X-YY,-127):: CALL SOUND
(-900,-8,1,110,30,110,30,999
9,30):: T=0
```

```
500 CALL POSITION(#3,Y,Y)::
IF Y>50 THEN 500 ELSE CALL D
ELSPRITE(#3):: CALL COLOR(#1
,11):: GOSUB 930 :: CALL DEL
SPRITE(#2):: GOTO 740
```

```
510 CALL SOUN(-900,-8,0,110
,30,110,30,300,30):: CALL PO
SITION(#1,X,Y):: CALL SPRITE
(#3,64,4,X,36,0,127)
```

```
520 CALL POSITION(#3,X,Y)::
IF Y<192 THEN 520 ELSE CALL
DELSPRITE(#3):: CALL COINC(#
2,X,220,7,T):: IF T THEN CAL
L SOUND(-1,-4,9)ELSE 450
```

```
530 CALL COLOR(#2,15):: FOR
T=0 TO 2 :: CALL PATTERN(#2,
88+T*4):: FOR X=5 TO 7 :: CA
LL SOUND(100,-X,T*10):: NEXT
X :: NEXT T
```

```
540 CALL DELSPRITE(#2):: Y1=
Y1+1 :: SC=SC+250 :: DISPLAY
AT(1,12)SIZE(10):USING "###
#####":SC :: IF Y1<5 THEN
440
```

```
550 CALL POSITION(#1,X,Y)::
IF X>185 THEN CALL LOCATE(#1
,35,Y)
```

```
560 CALL PATTERN(#1,112):: C
ALL MOTION(#1,0,20):: CALL S
OUND(-4250,-4,1,200,30,200,3
0,200,30)
```

```
570 CALL CHAR(68,S$&"000000F
F7F1FFEAB"&S$&"000000FFFFFFFA
A6D",64,E$&"0021F373FFFFFFF
FFFF972A08103CFF0FFFFFFF5EAA")
```

```
580 Y1=(Y AND 5)+6 :: GOSUB
800 :: CALL CHAR(60,"00000C8
080C4CFEEFFFF5BAAFF722B"&T
$&"CO0PFECF8F0E2C67B3D")::
CALL COLOR(1,1,1,2,1,1)
```

```
590 DISPLAY AT(20,1):W$ :: C
ALL COLOR(1,5,1,7,12,1):: FO
R T=1 TO 4 :: CALL LOCATE(#T
,161,1):: NEXT T :: CALL SPR
ITE(#9,112,13,15,1,20,35)
```

```
600 CALL SPRITE(#10,68,15,14
3,1,#11,64,15,143,17,#12,60,
15,143,33):: CALL MOTION(#10
,0,Y1,#11,0,Y1,#12,0,Y1)
```



DISKETTE JOYSTICK VERSION Continued

```

610 CALL SOUND(-4250,-4,1,20
0,30,200,30,200,30):: CALL P
OSITION(#9,X,Y):: IF X<35 TH
EN CALL MOTION(#9,0,V):: CAL
L LOCATE(#9,35,Y)ELSE IF X>1
40 THEN 690

620 CALL COINC(#9,#11,16,T)::
: IF T AND X>130 THEN 680 EL
SE CALL COINC(#9,#10,9,T)::
IF T THEN 710

630 CALL JOYST(1,Y,T):: IF Y
THEN CALL PATTERN(#9,Y+108)
:: V=V-Y*(ABS(Y+V)<16):: CAL
L MOTION(#9,0,V):: GOTO 610

640 IF T THEN CALL PATTERN(#
9,108):: CALL MOTION(#9,2*T*
(X-T>35),V):: GOTO 610

650 CALL PATTERN(#9,SGN(V)*4
+108):: CALL MOTION(#9,(X>35
)*3,V):: GOTO 610

680 CALL MOTION(#9,0,0,#10,2
,4,#11,2,4,#12,2,4):: SC=SC-
SP*500 :: SP=0

690 CALL MOTION(#9,3,0):: CA
LL SOUND(-2450,-8,6,110,30,1
10,30,9999,30):: CALL PATER
N(#9,104):: CALL SOUND(1,-4,
9):: HP=0 :: Z=Z-1

700 CALL DELSPRITE(#9):: IF
SP THEN 740 ELSE CALL DELSPR
ITE(#10,#11,#12):: GOTO 740

710 CALL POSITION(#10,X,Y)::
CALL SPRITE(#9,116,13,136,Y
,0,Y1):: FOR T=306 TO 122 ST
EP -6 :: CALL SOUND(-200,-4,
1,T,27,T,30,T,30):: NEXT T

720 FOR T=130 TO 306 STEP 8
:: CALL SOUND(-4250,-4,1,T,3
0,T,30,T,27):: NEXT T :: CAL
L MOTION(#9,-10,0):: K=K+1 ::
SC=SC+500*HP

730 FOR T=1 TO 300 :: NEXT T
:: SP=SP+HP :: HP=0

740 DISPLAY AT(1,2):USING B$
:HP,SP,SC,Z :: IF Z THEN CAL
L DELSPRITE(ALL)ELSE 250 !!!

750 CALL CHAR(60,"000001010F
090909010101013E200000C0C4C4
FCC0C0C0C0C02010080810203")

760 CALL CHAR(64,"0101031F13
1313030303033F20000000808888
88F8808080808000008040203")

770 CALL CHAR(68,"0000010107
09090501010106181000C0C0C2
F4C8C0C0C0C02010080810203",3
5,"FFFFFFFFF817E7E81")

780 DISPLAY AT(20,1):A$ :: C
ALL COLOR(2,11,1,1,11,1,13,1
2,1):: GOSUB 800

790 CALL SPRITE(#3,140,2,161
,256,0,-24):: FOR T=9 TO 20
:: CALL SPRITE(#T,136,7,200,
1):: NEXT T :: GOTO 1030

800 CALL DELSPRITE(ALL):: FO
R T=5 TO 8 :: CALL LOCATE(#T
,177,T*17):: NEXT T :: RETUR
N

810 IF T<>12 THEN CALL POSIT
ION(#1,X,Y,#3,YY,YY):: IF AB
S(Y-YY)<80 THEN CALL SPRITE(
#4,36,9,157,YY,X-147,2*V+Y-Y
)ELSE 1060 ELSE 1060

820 CALL SOUND(-150,-8,3,110
,30,110,30,5010,30):: CALL S
OUND(300,-8,1,128,30,128,30,
1100,30):: CALL DELSPRITE(#4
):: GOSUB 870 :: GOTO 1030

830 CALL MOTION(#3,0,V/4,#2,
0,0):: CALL PATTERN(#2,80)::
CALL SOUND(-1,-4,9):: IF T=
12 THEN CALL MOTION(#1,0,V)

840 FOR T=1 TO 9 :: CALL SOU
ND(50,-6,1):: NEXT T :: CALL
DELSPRITE(#2):: Y1=0 :: RET
URN

850 CALL COLOR(#3,2):: FOR T
=0 TO 16 STEP 4 :: CALL SOUN
D(-999,-8,T,120,27,127,28,10
00,30):: CALL PATTERN(#3,84+
T):: NEXT T

860 SC=605-3*X+SC :: CALL DE
LSPRITE(#3):: CALL SPRITE(#3
,140,2,161,256,0,-24):: RETU
RN

870 T=0 :: CALL SOUND(-1,-4,
9):: IF Y AND 1 THEN CALL PA
TTERN(#1,112):: CALL MOTION(
#1,9,V)ELSE CALL COLOR(#1,16
):: GOTO 930

880 CALL COLOR(#1,RND*7+3)::
FOR T=1 TO 26 STEP 5 :: CAL
L SOUND(T*40+200,-8,T,110,30
,110,30,1100-T,30)

890 CALL POSITION(#1,X,Y)::
IF X>155 THEN CALL MOTION(#1
,0,0):: GOTO 910 ELSE IF X A
ND 2 THEN 880

900 NEXT T :: GOTO 880

910 CALL COINC(#1,#2,16,T)::
IF T THEN CALL DELSPRITE(#2
)

920 CALL COINC(#1,#3,17,T)::
CALL COLOR(#1,2):: IF T THE
N CALL SOUND(-300,-8,1,110,3
0,110,30,3000,30):: CALL COL
OR(#3,7)

930 FOR Y=0 TO 16 STEP 4 ::
CALL SOUND(-999,-8,Y,120,27,
127,28,1100,30):: CALL PATTE
RN(#1,84+Y):: NEXT Y

940 CALL DELSPRITE(#1):: Z=Z
-1 :: HP=0 :: IF T THEN GOSU
B 850 :: RETURN ELSE RETURN

950 CALL MOTION(#1,0,V):: CA
LL COINC(#1,#2,12,YY):: IF Y
Y THEN CALL SOUND(-500,-8,1,
110,30,110,30,840,28):: CALL
DELSPRITE(#2)

960 CALL SOUND(-4250,-4,1,14
0,30,140,30,140,30):: CALL P
OSITION(#1,X,Y,#2,Y1,Y1):: C
ALL SPRITE(#1,116,13,160,Y,0
,0)

970 CALL MOTION(#2,0,4*SGN(Y
-Y1)):: IF Y1=0 THEN CALL SP
RITE(#2,76,2,163,256)

980 CALL JOYST(1,T,Y1):: IF
Y1>0 THEN 1020 ELSE YY=YY-4
:: IF YY<60 THEN YY=76

990 CALL PATTERN(#2,YY):: CA
LL COINC(#1,#3,32,T):: IF T
THEN GOSUB 830 :: GOSUB 870
:: GOTO 1030

1000 CALL COINC(#2,#3,24,T)::
: IF T THEN GOSUB 830 :: GOT
O 970 ELSE CALL COINC(#1,#2,
11,T):: IF T THEN CALL PATTE
RN(#2,80)ELSE 980

1010 CALL SOUND(-200,220,7,2
23,8,226,9):: HP=HP+1 :: DIS
PLAY AT(1,3)SIZE(2):HP :: CA
LL DELSPRITE(#2)

1020 CALL SOUND(-4000,-4,1,1
10,30,110,30,320,30):: CALL
MOTION(#1,-17,V/2,#2,0,0)::
CALL PATTERN(#2,128):: IF HP
=5 THEN 300 ELSE 1050

1030 IF Z THEN CALL SPRITE(#
1,112,13,20,1,20,35):: T,V=1
2

1040 DISPLAY AT(1,2):USING B
$:HP,SP,SC,Z :: IF Z=0 THEN
250

1050 CALL SOUND(-999,-4,1,11
0,30,110,30,200,30):: CALL P
OSITION(#1,X,Y,#2,Y1,#3,Y
Y,YY):: IF ABS(YY-Y)<5 THEN
810 ELSE IF Y1 THEN CALL PAT
TERN(#2,124)

1060 CALL MOTION(#3,0,SGN(Y-
YY))*(Y AND 6)+4+K):: IF X<
35 THEN CALL MOTION(#1,0,V)::
: CALL LOCATE(#1,35,Y)ELSE I
F X>151 THEN 950

1070 IF Y1 THEN CALL PATTERN
(#2,128):: IF ABS(YY-Y1)<26
THEN GOSUB 830

1080 CALL KEY(1,YY,T):: IF T
THEN 1130 ELSE CALL JOYST(1
,T,YY):: IF T OR YY THEN CAL
L PATTERN(#1,108+T)ELSE CALL
PATTERN(#1,V/3+108):: CALL
MOTION(#1,3*(X>35),V):: GOTO
1050

```



DISKETTE JOYSTICK VERSION Continued

```
1090 IF T THEN V=T*3 :: CALL
MOTION(#1,0,V):: GOTO 1050
ELSE CALL MOTION(#1,YY*2*(X-
YY>31),V):: GOTO 1050
```

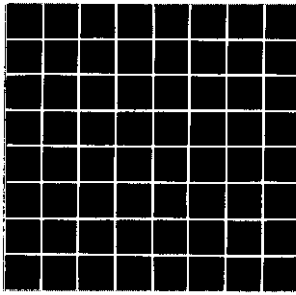
```
1130 CALL SOUND(-999,-8,3,12
8,30,128,30,999,30):: CALL M
OTION(#1,0,V):: CALL POSITIO
N(#1,X,Y):: Y=V/2+Y :: IF Y<
1 THEN Y=1
```

```
1140 FOR T=X+16 TO 175 STEP
13 :: Y=Y+13 :: IF Y>255 THE
N Y=1
```

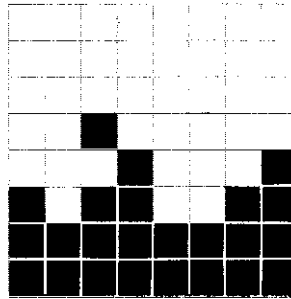
```
1150 CALL LOCATE(#T/13+7,T,Y
):: NEXT T :: IF Y1 THEN CAL
L COINC(#2,160,Y,18,T):: IF
T THEN CALL DELSPRITE(#2)
```

```
1160 CALL SOUND(-999,-8,3,12
8,30,128,30,500,30):: CALL C
OINC(ALL,T):: CALL DELSPRITE
(#9,#10,#11,#12,#13,#14,#15,
#16,#17,#18,#19,#20)
```

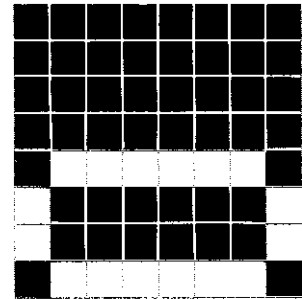
```
1170 IF T THEN CALL COLOR(#3
,16):: GOSUB 850 :: GOTO 104
0 ELSE 1050
```



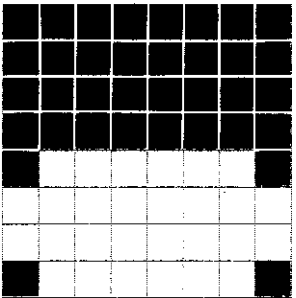
33 - FFFFFFFFFFFFFFFF



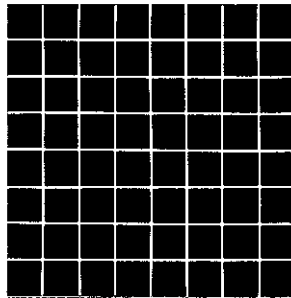
34 - 0000002011B3FFFF



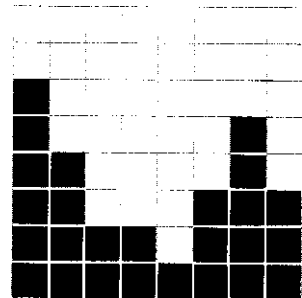
35 - FFFFFFFF817E7E81



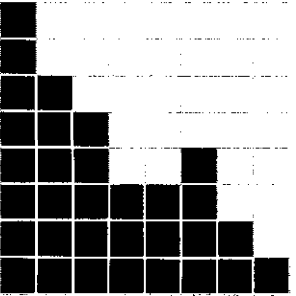
35 - FFFFFFFF81000081



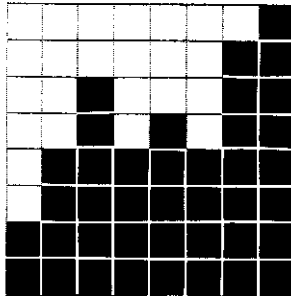
40 - FFFFFFFFFFFFFFFF



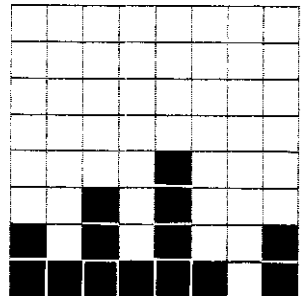
44 - 00008082C2C7F7FF



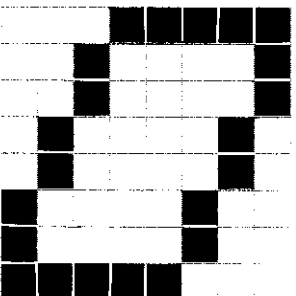
45 - 8080C0E0E4FCFEFF



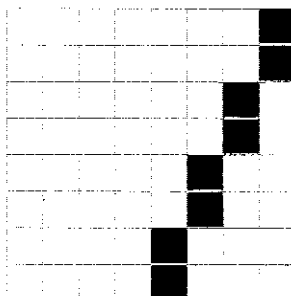
46 - 0103232B7F7FFFFFFF



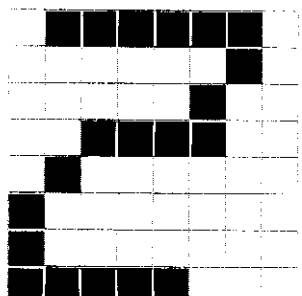
47 - 00000000828A9FD



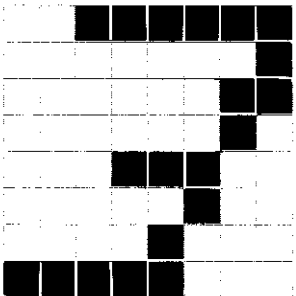
48 - 1F212142428484F8



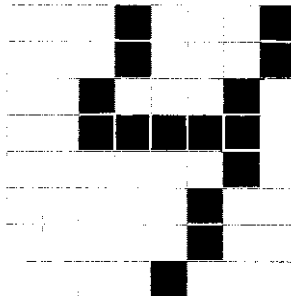
49 - 0101020204040808



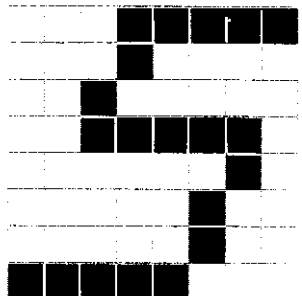
50 - 1F0101023C20407E



51 - 3F0103021C0408F8

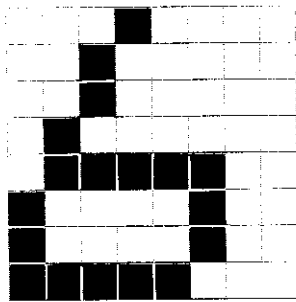


52 - 1111223E02040408

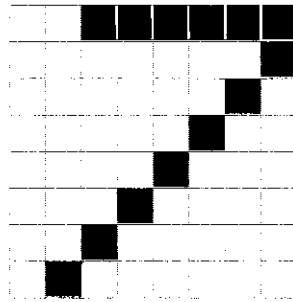


53 - 1F2020407C0408F8

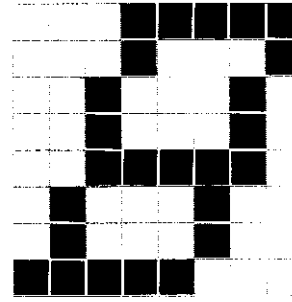




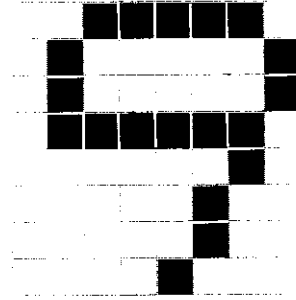
54 - 102020407E82827C



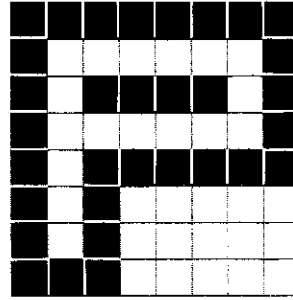
55 - 3F01020408102040



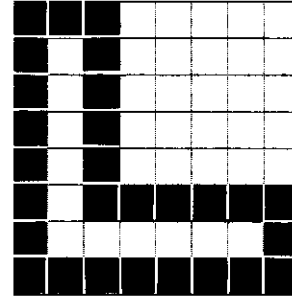
56 - 1F1122223E4444F8



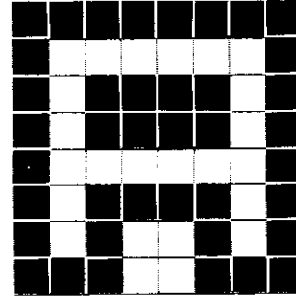
57 - 1F21213E02040408



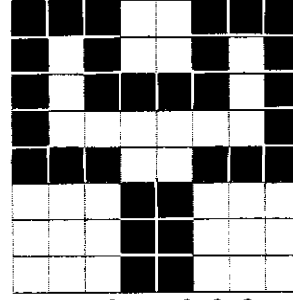
104 - FF81BD81BFA0A0E0



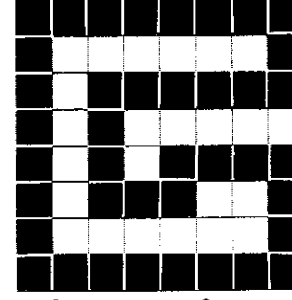
105 - E0A0A0A0A0BF81FF



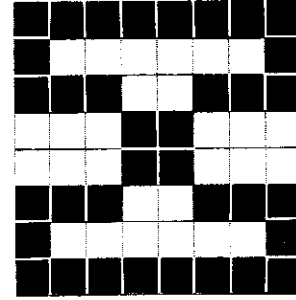
106 - FF81BDBD81BDA5E7



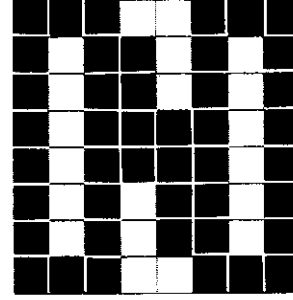
107 - E7A5BD81E7181818



108 - FF81BFA0AFB981FF



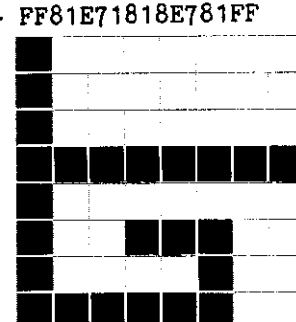
109 - FF81E71818E781FF



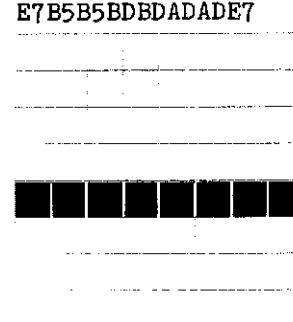
110 - E7B5B5BDBDADADE7



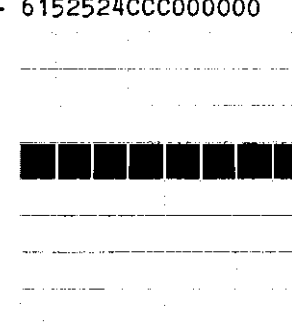
132 - 6152524CCC000000



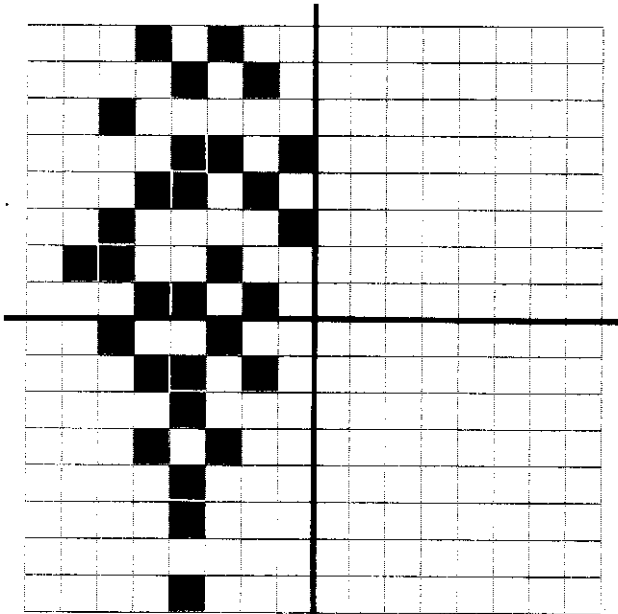
133 - 808080FF809C84FC



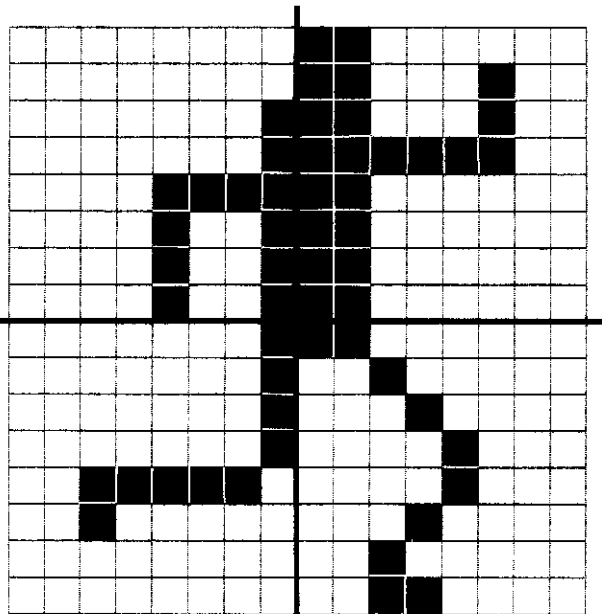
134 - 00000000FF000000



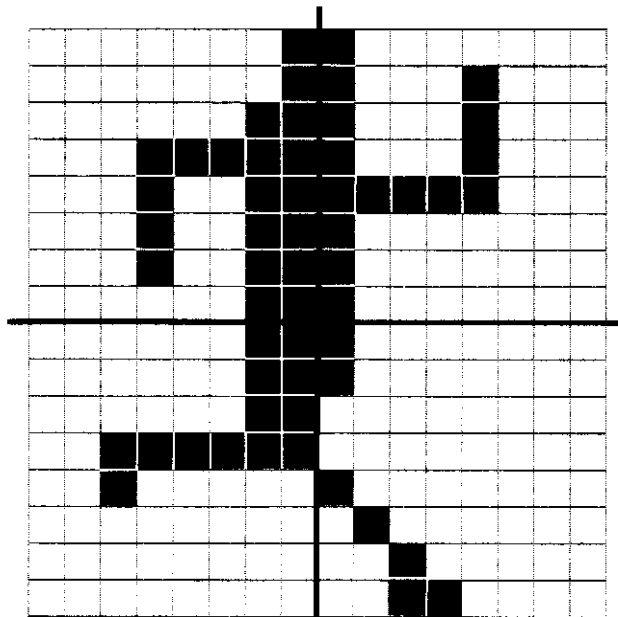
135 - 000000FF00000000



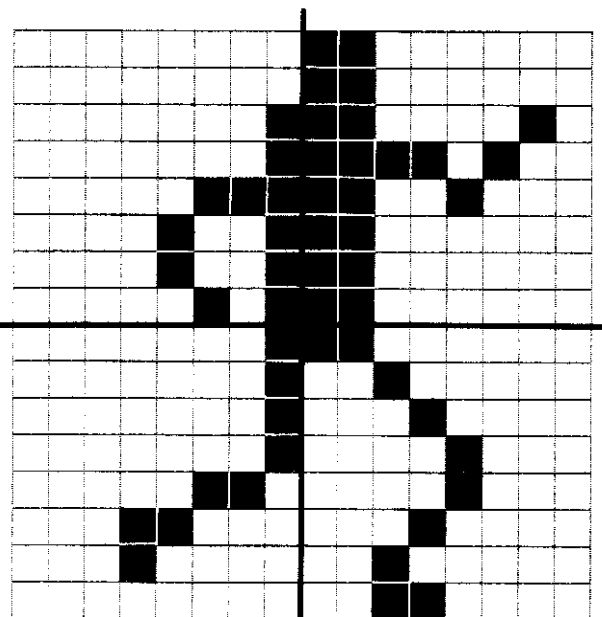
36 - 140A200D1A21641A
241A081408080008
0000000000000000
0000000000000000



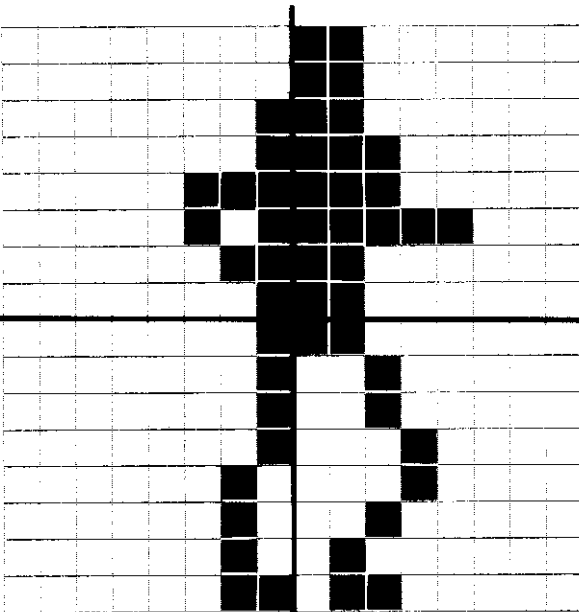
60 - 000001010F090909
010101013E200000
C0C4C4FCC0C0C0C0
C020100808102030



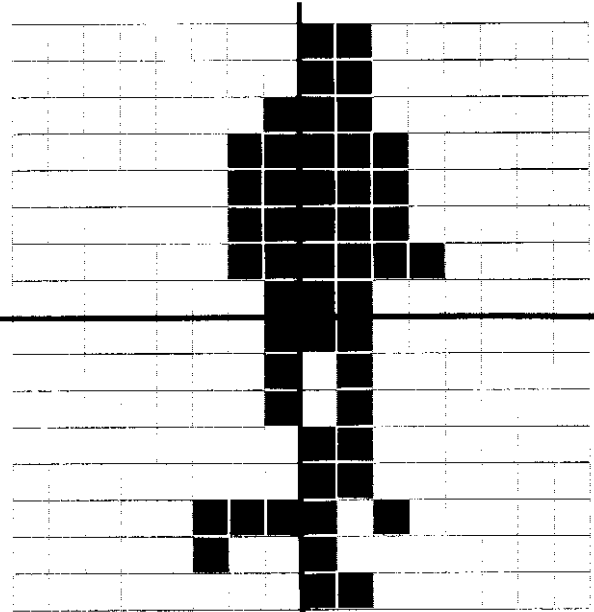
64 - 0101031F13131303
0303033F20000000
80888888F8808080
8080000080402030



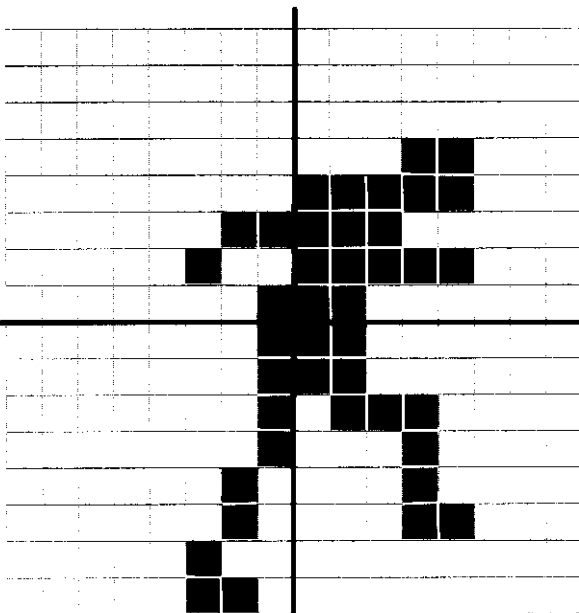
68 - 0000010107090905
0101010106181000
C0C0C2F4C8C0C0C0
C020100808102030



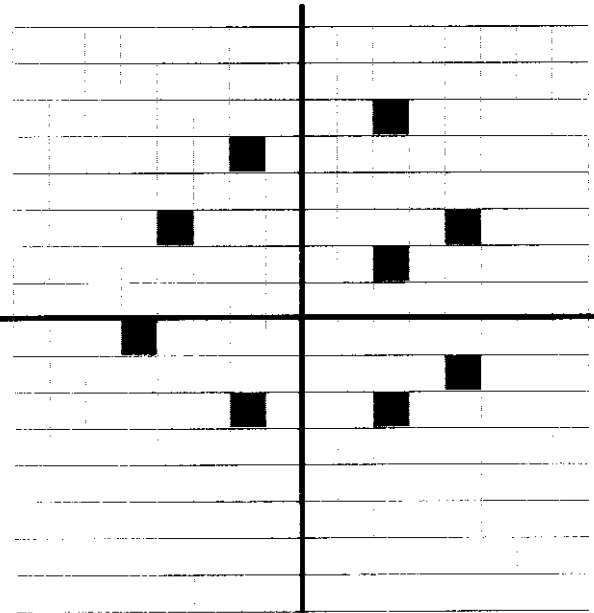
72 - 0000010107050301
 0101010102020203
 C0C0C0E0E0F8C0C0
 C020201010204060



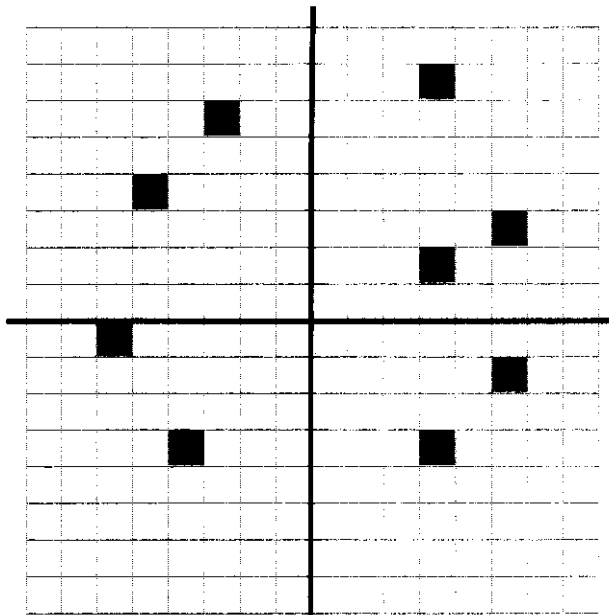
76 - 0000010303030301
 0101010000070400
 C0C0C0E0E0E0F0C0
 C04040C0C0A080C0



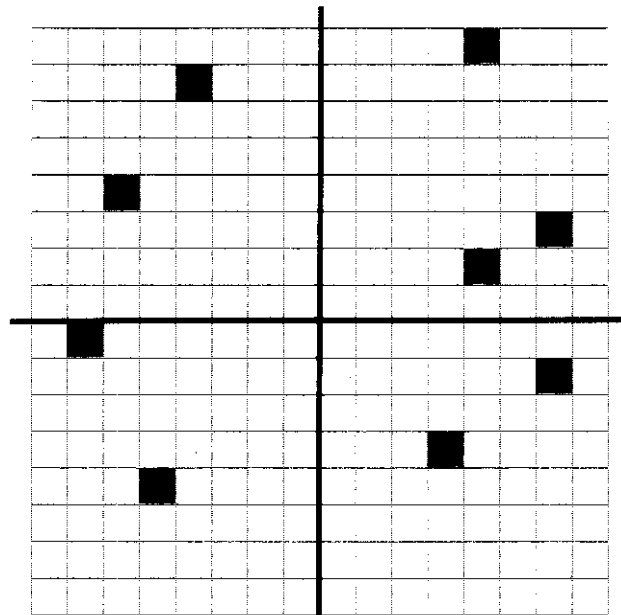
80 - 0000000000030401
 0101010102020406
 00000018F8E0F8C0
 C0C0701010180000



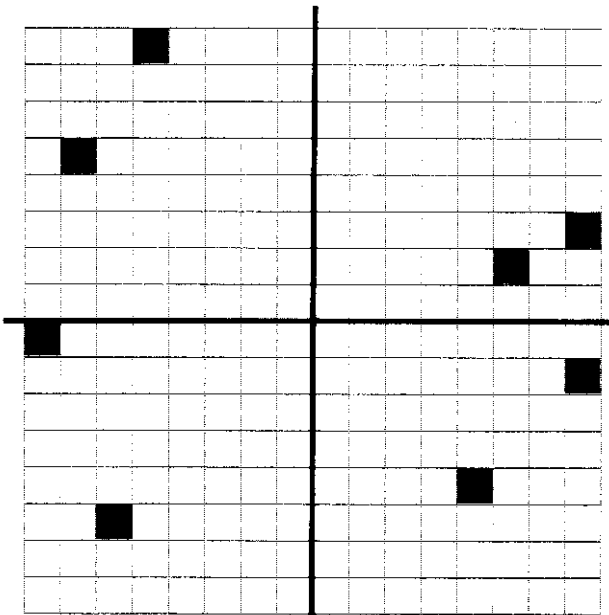
84 - 0000000200080000
 1000020000000000
 0000200000082000
 0008200000000000



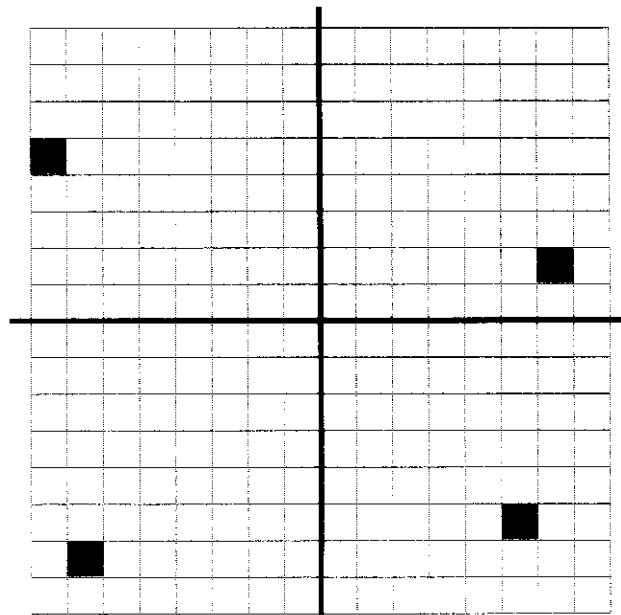
88 - 0000040010000000
 2000000800000000
 0010000000041000
 0004001000000000



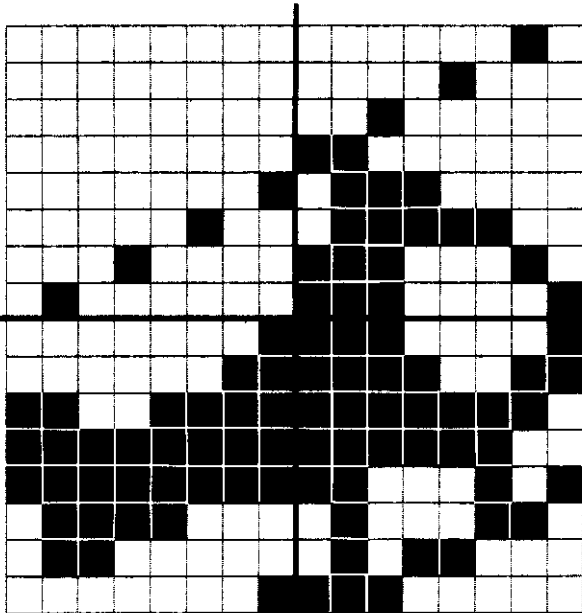
92 - 0008000020000000
 4000000010000000
 0800000000020800
 0002001000000000



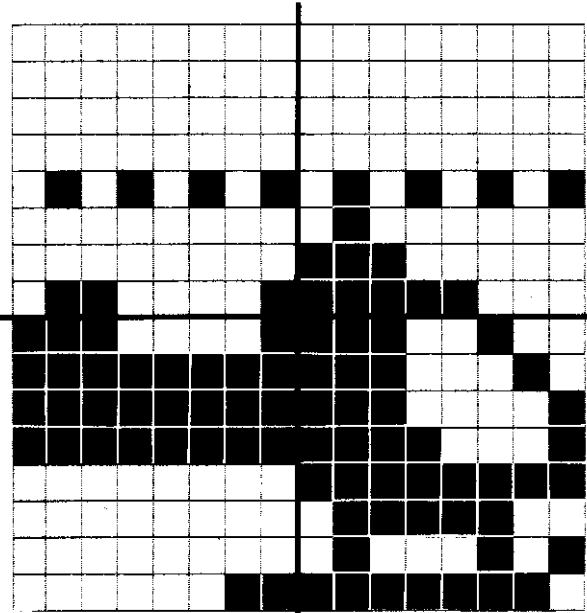
96 - 1000004000000000
 8000000000200000
 000000000010400
 0001000008000000



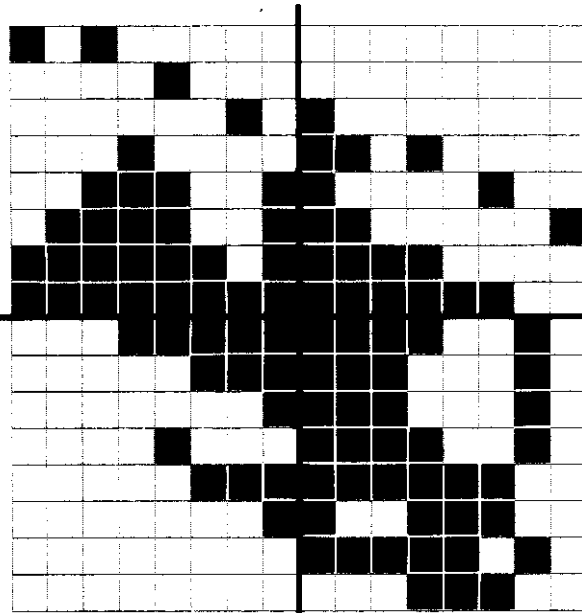
100 - 0000008000000000
 0000000000004000
 0000000000000200
 0000000000004000



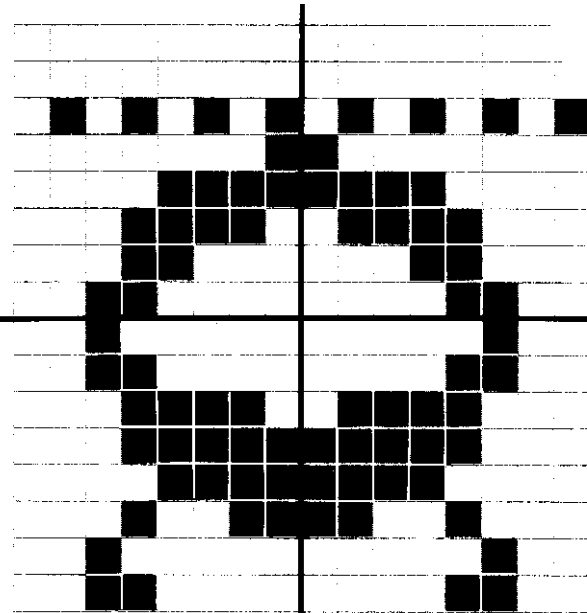
104 - 000000001041040
0103CFFFFFF786001
020820C0707CE2E1
E1F3FEFCC54658E0



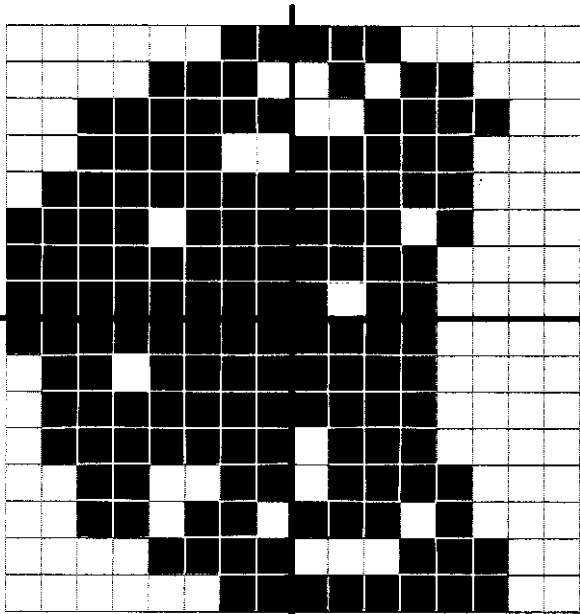
108 - 0000000055000061
E1FFFFFF00000003
000000005540E0F8
E4E2E1F1FF7C45FE



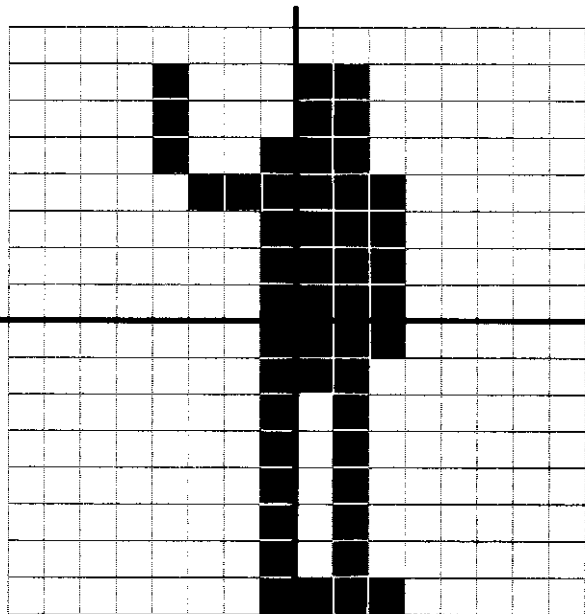
112 - A00802103979FDFE
1F07010807010000
000080D084C1F0FC
F2E2E2F2FC9CFA1C



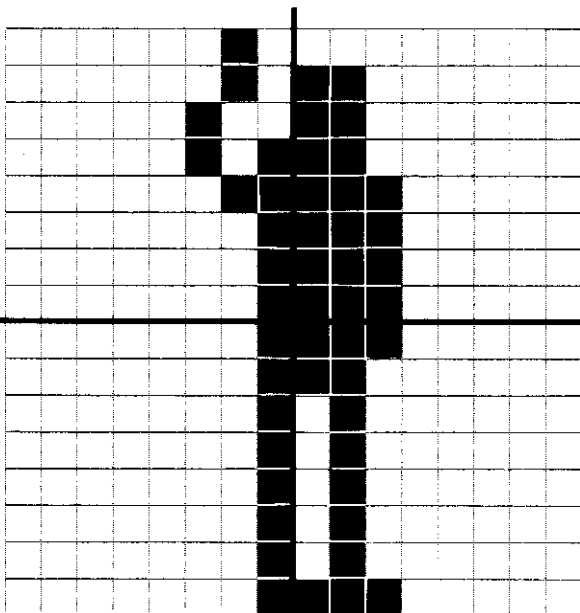
116 - 000055010F1E1830
20301E1F0F132030
00005580F078180C
040C78F8F0C8040C



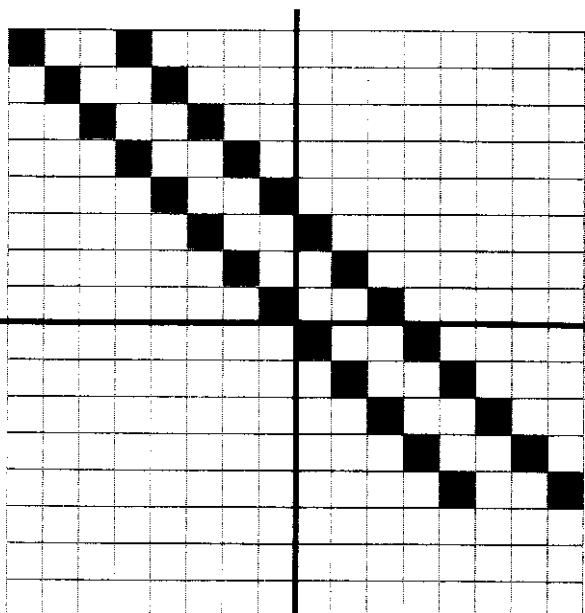
120 - 030E3F3C7FF7FFFF
 FF6F7F7F33360F03
 E0583CF8F8E8F0B0
 F0F0F07078E81CFC



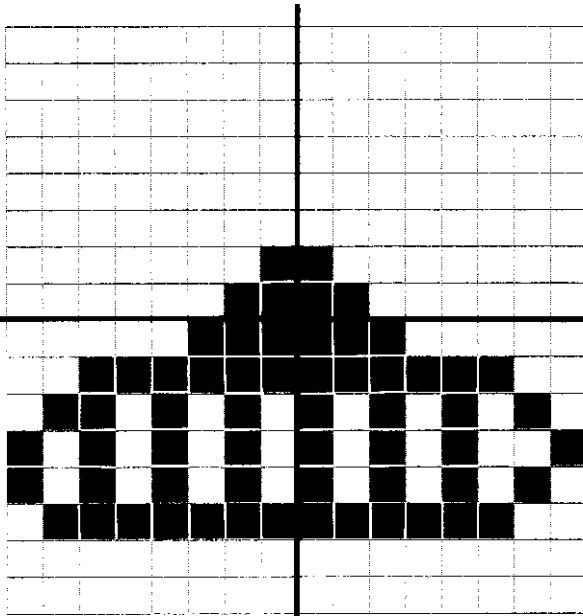
124 - 0008080907010101
 0101010101010101
 00C0C0C0E0E0E0E0
 E0C04040404040E0



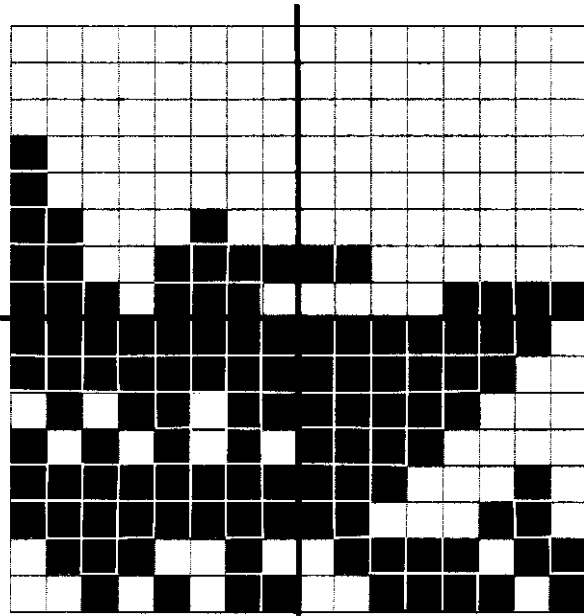
128 - 0202040503010101
 0101010101010101
 00C0C0C0E0E0E0E0
 E0C04040404040E0



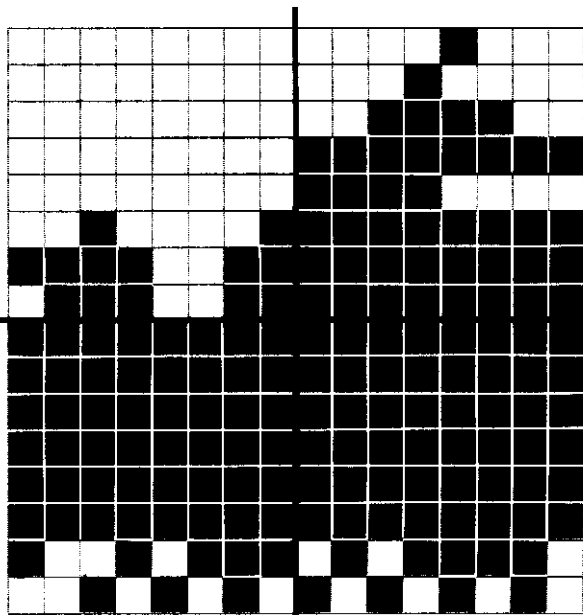
136 - 9048241209040201
 0000000000000000
 0000000000804020
 9048241209000000



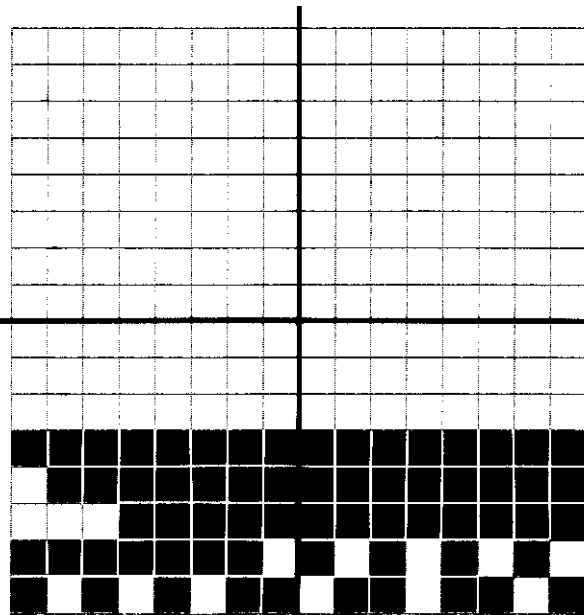
140 - 000000000000103
 073F6AAAAA7F0000
 00000000000080C0
 EOFCAAA9AAFC0000



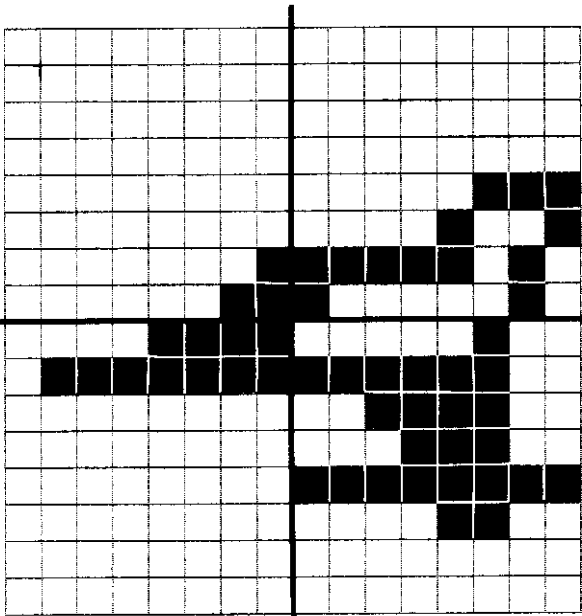
60 - 0000008080C4CFEE
 FFFF5BAAFFFF722B
 000000000000C00F
 FEFCF8F0E2C67B3D



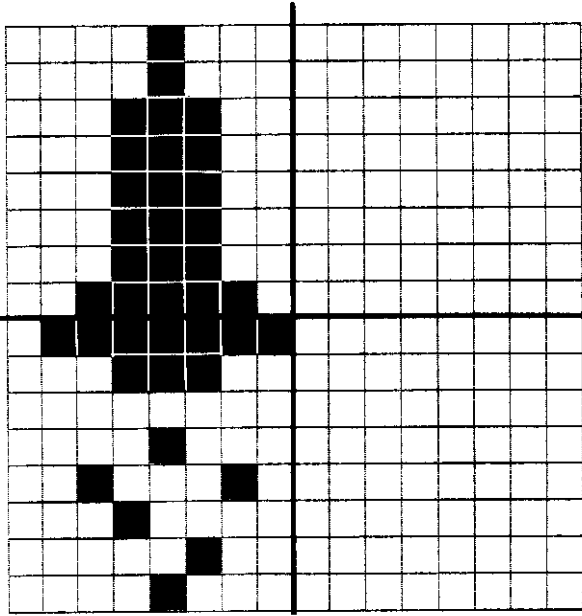
64 - 000000000021F373
 FFFFFFFFFF972A
 08103CFF0FFFFFFF
 FFFFFFFFFF5EAA



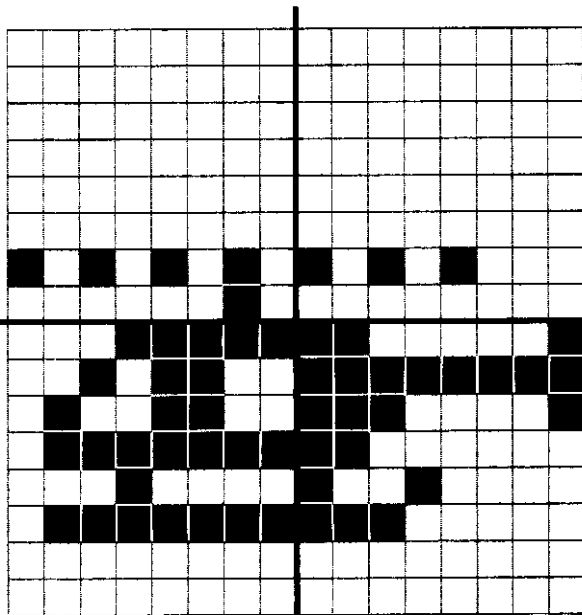
68 - 0000000000000000
 000000FF7F1FFEAB
 0000000000000000
 000000FFFFFFAA6D



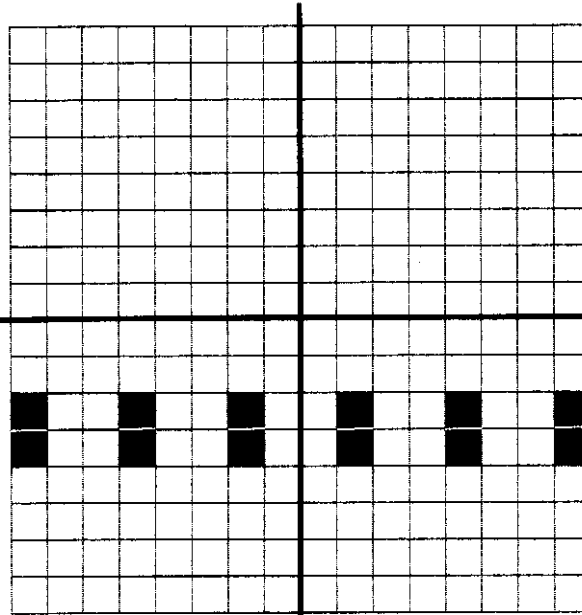
60 - 000000000000103
 0F7F00000000000
 00000000709FA82
 04FC3C1CFF0C0000



60 - 08081C1C1C1C1C3E
 7F1C000822100408
 000000000000000
 0000000000000000



60 - 000000000000AA02
 1F2C4C7F107F0000
 000000000000A800
 C1FFE1C090E00000



64 - 0000000000000000
 0000929200000000
 0000000000000000
 0000494900000000

MILLERS GRAPHICS - LIMITED WARRANTY

Millers Graphics warrants the Night Mission program and Book, which it manufactures, to be free from defects in materials and workmanship for a period of 90 days from the date of purchase.

During the 90 day warranty period Millers Graphics will replace any defective products at no additional charge, provided the product is returned, shipping prepaid to Millers Graphics. The Purchaser is responsible for insuring any product so returned and assumes the risk of loss during shipping.

Ship to:

Millers Graphics
1475 W. Cypress Ave.
San Dimas, California 91773

WARRANTY COVERAGE

The NIGHT MISSION Cassette and Book are warranted against defective material and workmanship. THIS WARRANTY IS VOID IF THE PRODUCT HAS BEEN DAMAGED BY ACCIDENT, UNREASONABLE USE, NEGLIGENCE, TAMPERING, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIALS OR WORKMANSHIP.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE 90 DAY PERIOD. MILLERS GRAPHICS SHALL NOT BE LIABLE FOR LOSS OR USE OF THE SOFTWARE OR BOOK, OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USE.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusion may not apply to you in those states.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.



MILLERS GRAPHICS
1475 W. Cypress Ave.
San Dimas, CA 91773
