



Star Dragon

Sean D. Wagle

And you thought the 64 could display only eight sprites. This short machine language game puts 16 on the screen. Or does it?

Sprites, those movable object blocks you create through a series of POKEs, are one of the most dazzling game elements available for the Commodore 64. You can create sprites of any shape, color them, move them, make them collide, even hide one behind another. But you can't have more than eight on the screen at any one time.

But with a bit of fancy dancing in machine language (ML), you can make it *seem* that there are more than eight sprites. "Star Dragon," an arcade-style game for the 64, uses a short machine language routine which temporarily moves all eight sprites to another location on the screen. *Temporarily* is the key word here, for the ML routine, listed as Program 1, "16," moves the sprites for only 1/60 second, then returns them to their original positions. As you'll see later, it's easy to

use this 16-sprite feature in your own games.

Head First

You need to type in and save two programs to play Star Dragon. First, enter Program 1—a BASIC loader which puts the 16-sprite routine in memory—and save it to disk. (See instructions below if you're using tape.) Use 16 as its filename. Next, enter Program 2, the game itself. Since it's written entirely in machine language, you'll need "MLX," found elsewhere in this issue, to type it in. After loading and running MLX, answer the prompts for the starting and ending addresses with:

Starting Address: C0F0
Ending Address: C79F

Use MLX to type in the Star Dragon data from Program 2 and

save it to disk as STAR.

It's a two-step process to run Star Dragon. First LOAD "16",8, then type RUN and press RETURN. You'll be prompted to enter a starting address. Enter 49152. After a moment, 16 sprites will appear on the screen. *While they're on the screen*, type LOAD "STAR",8,1, then press RETURN. Finally, when the message READY appears on the screen, enter SYS 49392, press RETURN, and the game begins. If that seems too complicated, use Program 3 to perform all these steps automatically. (When using Program 3, you must omit the {CLR} in line 100 of Program 1, as well as the PRINT "{CLR}": in line 290. Otherwise, the boot program will not work properly.)

The procedure is a bit different if you're using tape. It's necessary to load the data for Program 2 before starting the 16-sprite ML routine because the sprite routine's raster interrupts will disrupt tape loading. The easiest solution is to create a modified version of

Program 1. When typing in that program, replace lines 100 and 290 with those shown below:

```
100 PRINT "{CLR}":IF SA=0 THEN
  {SPACE}SA=49152:LOAD"STAR",1,1
290 POKE53178,255:POKE53179,255:POKE53281,0:SYS 49392
```

Then use MLX to type in Program 2. When you've entered all the data, save it immediately following the modified Program 1 on tape. Now, to run Star Dragon you need only load and run the modified Program 1.

The trail of star-like sprites, the Star Dragon, slithers and slides across the screen. The Dragon Gun at the bottom of the screen is your only defense.

Move the gun back and forth across the screen by pressing the + key to move left and the - key to move right. Hitting the CTRL key fires the gun. (A fourth key, SHIFT-LOCK, pauses the game for those times when your fingers tire.)

The object of the game is simple: Hit the dragon's head. If you hit anywhere else on the dragon, the shots just ricochet. Unfortunately, these bouncing bullets destroy your gun as well as the dragon. Don't let the dragon touch the gun, either—its poisonous barbs will ruin your weapon. You have three guns at hand. Lose all three and the game's over.

Scoring is straightforward. The first and second dragons of each level are worth 20 points when eradicated. The third is worth 60 points. Star Dragon has ten levels, with a total of 30 dragons.

16 Candles, 16 Sprites

The key to Star Dragon is what appears to be 16 sprites. Actually, there aren't more than eight on the screen at any one time, but since each group of eight is shifting every 1/60 second, a speed much too fast for our eyes to follow, it seems like there are 16. It's something like a movie, which displays 24 still frames a second. Each individual frame doesn't move, but put them together at that speed and the illusion is movement.

The machine language routine listed as Program 1 wedges itself into a new raster interrupt address. It's entirely relocatable, and ad-

vanced programmers may want to use it in their own games. Just include lines 100-240 and 300-410 from Program 1 in your program, and you'll have 16 sprites to work with instead of 8.

The routine in Program 1 moves the sprite registers from their normal locations to addresses 53170-53247. Take a look at the table below for the new register locations if you're programming with 16 sprites. You POKE values to these new locations just as you do for locations 53248-53294 when using the normal eight sprites—with a few exceptions.

One major difference is in the locations that control horizontal (x) position. In the normal eight-sprite system, visible horizontal positions range from 24-344. This requires a nine-bit value for a horizontal position—eight bits for each sprite in the even-numbered locations 53248-53262, plus a bit for the highest bit of each sprite's position in location 53264. Those familiar with programming sprites know that this causes a problem whenever a sprite crosses the "seam"—the point on the screen where the horizontal position value changed from 255 to 256. The 16-sprite routine avoids this problem by dividing the x-position by two. Thus, the sprite now disappears completely off the right edge of the screen when the value POKEd into locations 53184-53199 exceeds 172

(half the old value of 344).

Another difference is that the collision registers (53180-53183) contain only the value for those collisions occurring at the moment the register location is checked, instead of maintaining a value until read as is the case for the normal collision registers. Note that collisions between sprites 0-7 and sprites 8-15 cannot be detected. For example, if sprites 1, 5, 8, and 14 are all in the same position, then the collision between sprites 1 and 5 can be detected, as can the collision between sprites 8 and 14. However, there is no provision for recognizing the collision between sprites 1 and 14 or sprites 8 and 5.

The sprite data pointers (53232-53247) perform the same function as locations 2040-2047 in the eight-sprite system: They point to the 64-byte block of memory that defines the sprite's shape. For example, Program 1 loads these pointers with the value 11 (line 270), indicating that the sprite shape data is found at $11 * 64 = 704$ (see line 260).

The only real drawback to using this ML routine is that the sprites become partially transparent. However, this isn't noticeable if the sprite is on a dark background, such as gray, blue, or black. Notice that Star Dragon uses a black background.

See program listings on page 101.

Sprite Registers For 16-Sprite Routine

Location Hex	Location Decimal	Function
CFB2	53170	Sprites 0-7 multicolor off/on bits
CFB3	53171	Sprites 8-15 multicolor off/on bits
CFB4	53172	Sprites 0-7 x expansion bits
CFB5	53173	Sprites 8-15 x expansion bits
CFB6	53174	Sprites 0-7 y expansion bits
CFB7	53175	Sprites 8-15 y expansion bits
CFB8	53176	Sprites 0-7 sprite display priority
CFB9	53177	Sprites 8-15 sprite display priority
CFBA	53178	Bits to activate sprites 0-7
CFBB	53179	Bits to activate sprites 8-15
CFBC	53180	Collisions between sprites 0-7
CFBD	53181	Collisions between sprites 8-15
CFBE	53182	Sprites 0-7 collision with background
CFBF	53183	Sprites 8-15 collision with background
CFC0-CFCF	53184-53199	Sprites 0-15 x (horizontal) position
CFD0-CFDF	53200-53215	Sprites 0-15 y (vertical) position
CFE0-CFEF	53216-53231	Sprites 0-15 color registers
CFF0-CFFF	53232-53247	Sprites 0-15 data pointers

```

REM LDA #000
AH 520 DATA 032,213,255 :REM J
SR $FFD5
MH 530 DATA 169,131{5 SPACES}:
REM LDA #131
MX 540 DATA 141,002,003 :REM S
TA $0302
XH 550 DATA 169,164{5 SPACES}:
REM LDA #164
SP 560 DATA 141,003,003 :REM S
TA $0303
MP 570 DATA 134,045{5 SPACES}:
REM STX $2D
EG 580 DATA 132,046{5 SPACES}:
REM STY $2E
AA 590 IF BASIC=1 THEN GOSUB 8
60:GOTO 620
HD 600 LI=INT(LOC/256):L2=LOC-
(LI*256)
CM 610 PRINT#8,CHR$(76)CHR$(L2)
CHR$(L1);
RP 620 IF RN$="Y" THEN BU=58+L
EN(PN$)+1
CR 630 IF RN$="N" THEN BU=41+L
EN(PN$)+1
GE 640 IF BASIC=1 THEN BU=BU+1
1
GR 650 BL=88-BU
KC 660 FOR C=1 TO BL+1:PRINT#8
,CHR$(0);:NEXT C
KP 670 PRINT#8,CHR$(139)CHR$(2
27);
GH 680 B=679+LEN(PN$)+1
SD 690 LI=INT(B/256):L2=B-(LI*
256)
CA 700 PRINT#8,CHR$(L2)CHR$(L1
);
JP 710 CLOSE8
AS 720 GET#15,A$:S=ST
AJ 730 PRINT A$;
QE 740 IF S=0 THEN 720
KF 750 END
KP 760 L=LEN(LOC$)
ED 770 S=L-1
RK 780 FOR C=1 TO L
JR 790 I$=MID$(LOC$,C,1)
MD 800 IF I$<="9" THEN I$=STR$
(VAL(I$))
MB 810 IF I$>="A" THEN I$=STR$
(ASC(I$)-55)
QQ 820 LOC=LOC+VAL(I$)*16↑S
FK 830 S=S-1
AH 840 NEXT C
SM 850 RETURN
JC 860 FOR C=1 TO 14
JA 870 READ CODE
RE 880 PRINT#8,CHR$(CODE);
EP 890 NEXT C
CK 900 DATA 169,000{5 SPACES}:
REM LDA #000
EA 910 DATA 133,122{5 SPACES}:
REM STA $7A
EG 920 DATA 169,008{5 SPACES}:
REM LDA #008
RR 930 DATA 133,123{5 SPACES}:
REM STA $78
KP 940 DATA 032,096,166 :REM J
SR $A660
QG 950 DATA 076,174,167 :REM J
MP $A7AE
SD 960 RETURN

```

SpeedScript-80

See instructions in article on page 77 before typing in.

Patch 1

```

289E:4C B6 08 8E 00 D6 D0 07 B0
28A6:48 A9 1F 8D 00 D6 68 2C 9A
28AE:00 D6 10 FB 8D 01 D6 60 F4

```

```

28B6:A9 00 A2 12 20 A1 08 E8 D1
28BE:A9 50 20 A1 08 AD 11 20 4F
28C6:85 FB AD 12 20 85 FC A2 63
28CE:01 A0 00 B1 FB 99 3C 03 A4
28D6:C8 29 7F C9 1F F0 13 C0 06
28DE:50 D0 F0 88 B1 FB 29 7F 81
28E6:C9 20 F0 05 88 D0 F5 A0 A6
28EE:4F C8 84 3B A0 00 B9 3C 12
28F6:03 20 A6 08 C8 C4 3B D0 C6
28FE:F5 18 98 65 FB 85 FB A5 4D
2906:FC 69 00 85 FC E0 01 D0 C7
290E:03 8C 10 20 20 28 09 E8 A5
2916:E0 19 F0 03 4C CF 08 A5 C4
291E:FB 8D 1B 20 A5 FC 8D 1C 8F
2926:20 60 C0 50 F0 08 A9 20 D8
292E:20 A6 08 C8 D0 F4 60 00 E2

```

Patch 2

```

2A4E:A9 00 A8 20 96 11 20 28 EF
2A56:09 A9 00 4C 96 11 00 00 57

```

Patch 3

```

315D:A9 FD 8D 30 D0 A9 08 20 26
3165:96 11 A9 20 A2 18 20 A1 E5
316D:08 A9 8F 20 A6 08 A0 08 D0
3175:A9 FF A2 1E 20 A1 08 88 03
317D:D0 FA 60 29 7F C9 20 90 99
3185:0F C9 40 90 08 E9 40 C9 25
318D:20 90 02 69 1F 20 A6 08 C9
3195:60 A2 12 20 A1 08 A9 00 95
319D:E8 4C A1 08 00 00 00 3C

```

Patch 4

```

3445:A9 FC 8D 30 D0 A9 00 4C EF
344D:96 11 20 CD BD A0 00 B9 50
3455:00 01 F0 06 20 A6 08 C8 F0
345D:D0 F5 60 30 20 A9 00 20 82
3465:96 11 A9 20 20 70 11 00 79

```

Patch 5

```

C000:A9 12 85 FB A9 1E 85 FC 19
C008:A0 00 B1 FB 29 7F F0 14 0D
C010:C9 20 B0 02 A9 2A C9 40 7E
C018:90 0A 38 E9 40 C9 20 90 04
C020:03 18 69 20 91 FB C8 D0 37
C028:E1 A6 FC E0 20 F0 05 E8 A9
C030:86 FC D0 D6 A0 00 84 FB C5
C038:A9 08 85 FC B1 FB C9 20 42
C040:F0 0F C8 D0 F7 A6 FC E8 61
C048:86 FC E0 15 D0 EE 4C 76 0B
C050:C0 84 02 84 FD A5 FC 85 E1
C058:FE A0 01 B1 FD C9 D2 D0 4A
C060:10 C8 B1 FD C9 FF D0 09 2B
C068:A9 11 91 FD 88 A9 80 91 92
C070:FD A4 02 4C 42 C0 A9 4F D6
C078:8D E3 14 A9 14 8D E4 14 8B
C080:A9 06 8D FA 14 A9 A6 8D DB
C088:FC 14 A9 08 8D FD 14 A9 79
C090:20 8D B4 16 8D 85 18 A9 D9
C098:45 8D B5 16 A9 14 8D B6 A7
C0A0:16 A9 5D 8D 86 18 A9 11 15
C0A8:8D 87 18 A9 A6 8D 7C 09 DD
C0B0:A9 08 8D 7D 09 A0 04 A9 0F
C0B8:78 99 8B 0E 99 0A 16 A9 F9
C0C0:58 99 9B 0E 99 1A 16 88 13
C0C8:A9 EA 99 8B 0E 99 0A 16 C6
C0D0:99 9B 0E 99 1A 16 88 10 AB
C0D8:F1 A9 4F 8D 8D 1D 8D 06 DB
C0E0:1E A9 14 8D B9 1D 8D 07 9B
C0E8:1E A9 22 8D 8C 1B A9 16 3B
C0F0:8D 8D 1B A0 0B B9 1C C1 43
C0F8:99 24 16 88 10 F7 A0 08 45
C100:B9 28 C1 99 E5 14 88 10 DC
C108:F7 A9 EA 8D F9 14 8D FA 5E
C110:14 A9 28 8D FC 14 A9 09 7A
C118:8D FD 14 60 B9 45 20 F0 7E
C120:06 20 D2 FF C8 D0 F5 60 DE
C128:A9 36 85 01 98 18 69 43 0A
C130:A8 00 00 00 00 00 00 08

```

Star Dragon

Article on page 54.

BEFORE TYPING . . .

Before typing in programs, please refer to "How To Type In COMPUTE!'s GAZETTE Programs," which appears before the Program Listings.

Program 1: 16 Sprites

```

CD 100 INPUT"[CLR]{2 DOWN}STAR
TING ADDRESS";SA
AR 110 IFSA>53020THENPRINT"INP
UT AN ADDRESS LOWER THA
N 53021":RUN
SH 120 PRINT"WAIT..."
SD 130 REM >> PUT DATA AT STAR
TING ADDR <<
DH 140 FORT=0T0149:READV$
RA 150 L$=LEFT$(V$,1)
JD 160 IFASC(L$)>64THEN HN=ASC
(L$)-55
MX 170 IFASC(L$)<65THEN HN=ASC
(L$)-48
RA 180 R$=RIGHT$(V$,1)
PS 190 IFASC(R$)>64THEN LN=ASC
(R$)-55
JP 200 IFASC(R$)<65THEN LN=ASC
(R$)-48
CD 210 B=HN*16+LN:POKE SA+T,B:
NEXT
BX 220 REM >> WEDGE PROGRAM IN
TO RASTER <<
HH 230 POKE53265,27:POKE56333,
127:POKE788,((SA/256)-I
NT(SA/256))*256
RX 240 POKE789,SA/256:POKESA+1
09,PEEK(648)+3:POKE5327
4,129
XB 250 REM >>>> DISPLAY 16 SP
RITES <<<<<<
ED 260 FORT=53170T053247:POKET
,0:NEXT:FORT=0T063:POKE
704+T,255:NEXT
DE 270 FORT=0T015:POKE53232+T,
11
GG 280 POKE53216+T,1+RND(0)*8:
POKE53184+T,140-T*8:POK
E53200+T,60+T*10:NEXT
CE 290 POKE53178,255:POKE53179
,255:POKE53281,0:PRINT"
[CLR]":END
JS 300 REM >>>> 16 SPRITES HE
X DATA <<<<<<
AF 310 DATA A5,FD,29,01,AA,49,
01,A8,BD,B2,CF,8D,1C,D0
RM 320 DATA BD,B4,CF,8D,1D,D0,
BD,B6,CF,8D,17,D0,BD,B8
EG 330 DATA CF,8D,1B,D0,BD,BA,
CF,8D,15,D0,AD,1E,D0,99
PE 340 DATA BC,CF,AD,1F,D0,99,
BE,CF,A9,01,8D,19,D0,A5
JF 350 DATA FD,29,01,0A,0A,0A,
AA,A0,00,84,FE,A9,01,85
SQ 360 DATA FC,BD,C0,CF,0A,99,
00,D0,90,06,A5,FC,05,FE
BH 370 DATA 85,FE,BD,D0,CF,99,
01,D0,8A,84,FF,29,07,A8
BC 380 DATA BD,E0,CF,99,27,D0,
BD,F0,CF,99,F8,07,A4,FF
KQ 390 DATA 18,26,FC,E8,C8,C8,
C0,10,D0,CD,A5,FE,8D,10
XE 400 DATA D0,AD,1F,D0,E6,FD,
A9,00,8D,12,D0,AD,0D,DC

```

BC 410 DATA 29,01,F0,03,4C,31,
EA,4C,BC,FE

Program 2: Star Dragon—Main Program

See instructions in article on page 54 before typing in.

```
C0F0:4C 6A C7 A4 15 8C 01 D4 28
C0F8:F0 02 C6 15 4C 00 C2 60 E5
C100:00 18 00 00 18 00 00 18 62
C108:00 00 3C 00 1E 3C 78 0F F4
C110:3C F0 07 FF E0 03 E7 C0 72
C118:03 C3 E0 3F BD FC FF BD BD
C120:FF 3F C3 FC 03 E7 E0 03 38
C128:FF E0 07 3C F0 1E 3C 38 39
C130:1E 3C 38 00 18 00 00 18 B1
C138:00 00 18 00 00 00 00 00 BE
C140:00 00 00 00 00 00 00 00 C3
C148:00 00 30 00 00 78 00 00 B3
C150:FC 00 01 FE 00 03 FF 00 6E
C158:00 78 00 00 78 00 03 7B 3F
C160:00 07 7B 80 0F 7B C0 1F 24
C168:7B E0 3F FF F0 7F FF F8 48
C170:7C FC FB 79 FE 78 73 FF E8
C178:38 63 FF 18 00 00 00 00 72
C180:93 05 0D 0D 0D 20 20 7E
C188:20 20 20 20 20 20 20 0C
C190:53 43 4F 52 45 3A 30 30 41
C198:30 30 20 4C 45 56 45 4C 63
C1A0:3A 42 0D 0D 9E 20 20 20 1A
C1A8:20 20 20 20 20 20 20 2C
C1B0:20 20 20 20 20 48 49 54 5B
C1B8:20 2B 0D 0D 20 20 20 6B
C1C0:20 20 20 20 20 20 54 4F DB
C1C8:20 50 4C 41 59 20 53 54 54
C1D0:41 52 44 52 41 47 4F 4E 4B
C1D8:00 40 80 C0 C0 30 0C 03 6A
C1E0:20 21 22 23 25 26 27 28 6B
C1E8:2A 2B 2C 2D 2F 30 31 32 73
C1F0:3A 35 36 37 39 3A 3B 3C 7B
C1F8:3E 3F 40 41 43 44 45 46 83
C200:A2 00 A2 00 BD 00 CF F0 A9
C208:44 C9 01 F0 14 BC D0 CF 56
C210:88 88 98 9D 0D CF C9 04 46
C218:B0 05 A9 00 9D 00 CF F0 E9
C220:2C BC 40 CF BD C0 CF 85 06
C228:AE BD 20 CF 20 53 C2 9D E6
C230:40 CF A5 AE 9D C0 CF BC B5
C238:50 CF BD D0 CF 85 AE B1 4E
C240:30 CF 20 53 C2 9D 50 CF 08
C248:A5 AE 9D D0 CF E8 E0 0F FF
C250:D0 B2 60 30 0A 0A 84 02 7D
C258:65 02 90 02 E6 AE 60 0A FF
C260:85 02 98 E5 02 B0 02 C6 38
C268:AE 60 4C 13 C3 A2 00 BD 7E
C270:00 CF C9 01 D0 F4 A0 00 CE
C278:84 AE 84 B0 A9 FF 85 AF 8F
C280:85 B1 FE 70 CF BD 70 CF 42
C288:D0 13 A9 FC 9D 70 CF DE 6D
C290:10 CF BD 10 CF D0 06 20 B8
C298:DD C2 4C 13 C3 BD C0 CF DE
C2A0:C9 12 B0 08 A0 00 84 AE E2
C2A8:A0 7F 84 AF C9 9C 90 08 D3
C2B0:A0 80 84 AE A0 FF 84 AF DF
C2B8:BD D0 CF C9 30 B0 08 A0 DC
C2C0:00 84 B0 A0 7F 84 B1 C9 C2
C2C8:DA 90 08 A0 80 84 B0 A0 03
C2D0:FF 84 B1 C0 00 F0 3C 20 16
C2D8:DD C2 38 B0 36 BC 60 CF 45
C2E0:B9 00 CE 05 AE 25 AF 9D 74
C2E8:20 CF B9 01 CE 05 B0 25 CA
C2F0:B1 9D 30 CF B9 01 CE 05 2E
C2F8:B0 25 B1 9D 30 CF BD 10 7C
C300:CF F0 0F B9 02 CE 9D 10 BF
C308:CF BD 60 CF 18 69 04 9D FB
C310:60 CF 60 E8 E0 0F B0 03 FD
C318:4C 6F C2 60 4C F6 C3 A2 68
C320:00 BD 00 CC C9 00 F0 F4 09
C328:C9 02 00 05 A9 00 9D 00 FF
C330:CC BD 80 CB 85 AE BD C0 7D
C338:CB 85 AF A9 00 A8 91 AE 0C
C340:BD C0 CD 85 AD A5 AD F0 39
```

```
C348:54 A0 00 BD 40 CD C9 04 CE
C350:B0 01 C8 C9 9E 90 01 C8 28
C358:BD 80 CD C9 C8 90 01 C8 88
C360:0F 00 F0 08 A9 00 9D 00 6F
C368:CC 4C F6 C3 BD 40 CD 85 94
C370:AE BC C0 CC BD 40 CC 20 0C
C378:53 C2 9D C0 CC A5 AE 9D 12
C380:40 CD BD 80 CD 85 AE BC F9
C388:00 CD BD 80 CC 20 53 C2 93
C390:9D 00 CD A5 AE 9D 80 CD B5
C398:C6 AD 4C 45 C3 BD 80 CD B0
C3A0:4A 4A 4A 48 29 03 86 02 12
C3A8:A8 B9 D8 C1 85 AE 68 A8 8A
C3B0:B9 E0 C1 85 AF BD 40 CD A0
C3B8:18 65 AE 85 AE 90 02 E6 76
C3C0:AF BD 40 CD 18 65 AE 85 AD
C3C8:AE 90 02 E6 AF 29 F8 85 14
C3D0:AE BD 80 CD 29 07 A8 BD 80
C3D8:40 CD 29 03 AA BD DC C1 11
C3E0:11 AE 91 AE A6 02 18 98 BF
C3E8:65 AE 90 02 E6 AF 9D 80 B2
C3F0:CB A5 AF 9D C0 CB E8 E0 7F
C3F8:40 F0 03 4C 21 C3 60 AC 87
C400:CF CF A5 CB C9 28 D0 06 6D
C408:C0 E0 90 02 88 88 C9 2B CC
C410:D0 06 C0 A0 B0 02 C8 C8 8D
C418:8C CF CF AD 8D 02 C5 16 C6
C420:85 16 F0 07 C9 04 D0 03 83
C428:20 2C C4 60 A2 00 BD 00 FB
C430:CC D0 21 A9 06 9D C0 CD 09
C438:A9 00 9D 40 CC A9 FF 9D F8
C440:80 CC AD CF CF 38 E9 08 2B
C448:9D 40 CD A9 B0 9D 80 CD CF
C450:FE 00 CC 60 E8 E0 4D D0 15
C458:D5 60 A2 00 BD 00 CC F0 B1
C460:5B BD C0 CD C9 03 F0 54 8C
C468:BD 40 CD 85 B0 BD 80 CD 3E
C470:85 B1 86 02 A2 00 86 AF EB
C478:BD 00 CF C9 01 D0 43 BD 07
C480:C0 CF E9 0C C5 B0 B0 2D DB
C488:69 C0 C5 B0 90 27 BD D0 FA
C490:CF E9 30 C5 B1 B0 1E 69 D4
C498:14 C5 B1 90 18 E4 AF D0 61
C4A0:26 A9 02 9D 00 CF A9 2A 7E
C4A8:85 15 86 AA A6 02 A9 F0 37
C4B0:9D 80 CD A6 AA E8 E0 0F 17
C4B8:D0 BE A6 02 E8 E0 40 D0 6B
C4C0:9B 60 E6 AF D0 EF 60 20 2F
C4C8:97 E0 A6 02 A9 03 9D C0 A0
C4D0:CD A5 8E 9D 40 CC A9 7F 5E
C4D8:9D 80 CC 4C BA C4 AD 0E 02
C4E0:CF C9 02 F0 03 4C 63 C5 E9
C4E8:AD CE CF E9 08 85 B0 AD FA
C4F0:DE CF E9 28 85 B1 A2 00 D5
C4F8:86 02 20 97 E0 A6 02 A5 0F
C500:8E 9D 40 CC A5 8F 9D 80 36
C508:CC A9 01 9D 00 CC A5 8D 6A
C510:29 03 9D C0 CD A5 B0 9D B4
C518:40 CD A5 B1 9D 80 CD E8 7A
C520:E0 40 D0 D4 A2 00 8E BA 80
C528:CF A2 80 8E BB CF A2 00 9F
C530:86 96 20 1F C3 A6 96 E8 69
C538:8E 01 D4 D0 F3 EE FD 9F E9
C540:AE FD 9F E0 03 90 16 EE 1A
C548:FF 9F A9 00 8D FD 9F AD 42
C550:FF 9F C9 0B F0 0A A9 22 D2
C558:85 B1 20 BD C5 4C 3A C6 80
C560:4C 23 C7 AD BF CF 30 29 75
C568:AD BF CF 30 24 A2 00 BD 21
C570:C0 CF E9 05 CD CF CF B0 BD
C578:10 69 0B CD CF CF 90 09 8C
C580:BD D0 CF C9 D6 90 02 B0 63
C588:08 E8 E0 0F D0 E1 4C 22 28
C590:C6 A9 08 8D 01 D4 A2 00 64
C598:A0 00 C8 D0 FD EE 20 D0 57
C5A0:E8 D0 F5 8E 01 D4 EE FA B0
C5A8:9F AD FA 9F C9 03 B0 03 87
C5B0:4C 3A C6 8E BA CF 8E BB A0
C5B8:CF A9 58 85 B1 A9 15 8D E5
C5C0:18 D0 A9 1B 8D 11 D0 A2 68
C5C8:00 AD FD 9F 0A 18 69 30 2D
C5D0:8D 98 C1 AD FF 9F 38 69 B4
C5D8:3F 8D A1 C1 E9 10 C9 3A 15
C5E0:90 07 EE 96 C1 E9 0A B0 38
C5E8:F5 8D 97 C1 A2 00 BD 80 F2
```

```
C5F0:C1 20 D2 FF E8 E4 B1 D0 CE
C5F8:F5 85 A2 0A C5 A2 D0 FC 2D
C600:A5 B1 C9 58 F0 01 60 A5 7D
C608:CB C9 28 D0 FA 4C F0 C0 AB
C610:20 6D C2 AD FF 9F 85 B4 7A
C618:20 F3 C0 C6 B4 D0 F9 4C 60
C620:DE C4 20 FF C3 20 1F C3 F2
C628:20 5A C4 AD 8D 02 C9 01 D8
C630:D0 DE A9 00 8D 01 D4 4C 79
C638:2B C6 A9 20 85 B1 A9 00 8A
C640:85 B0 A8 AA 91 B0 C8 D0 2E
C648:FB E8 E6 B1 E0 20 D0 F4 24
C650:AA 9D D0 CF 9D 00 CC E8 07
C658:E0 BA D0 F5 AA 86 02 20 12
C660:97 E0 A5 8E A6 02 29 BF DE
C668:9D 00 CE E8 D0 EF A2 00 B8
C670:BD 02 CE 29 1F 9D 02 CE 0C
C678:E8 E8 E8 D0 F2 A5 8F 8D
C680:4A 4A 69 20 85 AC A5 AC CB
C688:9D C0 CF A9 40 9D D0 CF 93
C690:86 02 20 97 E0 A6 02 A5 AA
C698:8F 29 07 69 02 9D E0 CF C7
C6A0:FE 00 CF A9 0D 9D F0 CF D2
C6A8:A9 F0 9D 70 CF E8 E0 0F F4
C6B0:D0 D4 A9 FF 8D BA CF 8D 95
C6B8:BB CF A9 0E 8D FF CF A2 DC
C6C0:00 A0 01 98 9D 10 CF E8 D5
C6C8:C8 E0 0F D0 F6 A9 E0 8D 8F
C6D0:DF CF A9 80 8D CF FF A9 D4
C6D8:3B 8D 11 D0 A9 1D 8D 18 8B
C6E0:D0 A9 01 8D EF CF A2 00 3E
C6E8:A9 10 9D 00 04 9D 00 05 9E
C6F0:9D 00 06 9D E8 06 E8 D0 E9
C6F8:F1 A9 0F 8D 18 D4 A9 0F 1B
C700:8D 05 D4 A9 F5 8D 06 D4 93
C708:A9 81 8D 04 D4 A2 00 A9 99
C710:30 9D 96 C1 E8 E0 04 D0 B1
C718:F6 AD FF 9F 0A 8D 20 D0 20
C720:4C 10 C6 A2 00 A9 93 20 CA
C728:D2 FF A9 1B 8D 11 D0 A9 04
C730:15 8D 18 D0 BD 52 C7 9D 22
C738:20 D9 9D 20 05 E8 E0 18 A9
C740:D0 F2 CA 86 A2 CA A5 A2 E4
C748:8D 21 D0 E4 A2 D0 F7 4C E3
C750:B9 C5 07 0F 0F 04 20 17 DF
C758:0F 12 0B 2C 20 04 12 01 4E
C760:07 0F 0E 13 0C 01 19 05 C5
C768:12 21 A2 00 A9 20 9D C0 67
C770:CB BD 00 C1 9D 40 03 E8 4E
C778:E0 80 D0 F0 A9 00 8D FA 25
C780:9F 8D FD 9F A9 01 8D FF 69
C788:9F A9 50 8D CF CF 8D 21 2F
C790:D0 8D 20 D0 4C 3A C6 00 D5
C798:00 00 00 00 EA EA EA EA EB
```

Program 3: Star Dragon Booter

```
EP 100 POKE 53280,2:POKE 53281
,0:PRINT"[CLR]{2 DOWN}"
TAB(9)"[73]{RVS} BOOTING
DRAGON STAR [43]:PRINT
KC 110 PRINT"LOAD "CHR$(34)":0:
16"CHR$(34)",8":PRINT"
[4 DOWN]RUN"
RQ 120 PRINT"[2 DOWN]
[18 RIGHT]49152"
HG 130 PRINT"[3 DOWN]LOAD "CHR
$(34)":0:STAR"CHR$(34)",
8,1":PRINT"[4 DOWN]SYS
[SPACE]49392"
FQ 140 PRINT"[21 UP]"
BP 150 FOR I=0 TO 4:POKE 631+I
,13:NEXT:POKE 198,5
```

MonoTones

Article on page 68.

```
RF 10 POKE53280,0:POKE53281,0:
DIM C$(15):FORA=0TO15:RE
ADB:C$(A)=CHR$(B):NEXT
AE 20 A$="THIS IS REALLY A GLO
```