10 GRAPHICS

In this chapter we are going to look at two graphics orientated executable programs (i.e., the programs are graphical in nature and created as jobs and executed by using the SuperBASIC EXEC command). The two programs are:

1. CLOCKF - Produces a real-time analogue clockface display

2. BALL - Produces a rolling multi-colour ball display

The first example, CLOCKF, is an example of standard graphics. In it we will see how the floating point package may be used to enable straight lines and arcs to be drawn. The second example, BALL, is orientated toward direct screen addressing. This type of graphics programming is the sort that will be found most often in games packages for special figures and moving items.

Each of the programs is listed in full as an assembler output list file, and preceded by a short description. The descriptions tend to rely upon the reader having read and understood previous examples (both in this chapter and the previous chapter), where appropriate. This keeps repetition to a minimum and enables you to get quickly to the new pertinent points. The source code of the programs, and the corresponding '_exec' files, are on one of the two Microdrive cartridges which can accompany this book. The assembler/editor package (described in Part 4) which was used to develop the programs is available on the other Microdrive cartridge.

The hexadecimal opcode listings could be used to enter the machine code directly into memory manually. Although this is long, tedious, and prone to error, it does at least give you the opportunity of trying the programs out without having to purchase an assembler package.

10.1 Screen memory layout

Before going on to look at the examples it is worth making sure that we know how the screen memory is organized. Pixel decoding is performed on a 'word' (i.e., 16-bit) data size system, as shown in Fig.10.1. In four colour mode each word represents eight pixels. In eight colour mode each word represents four pixels.

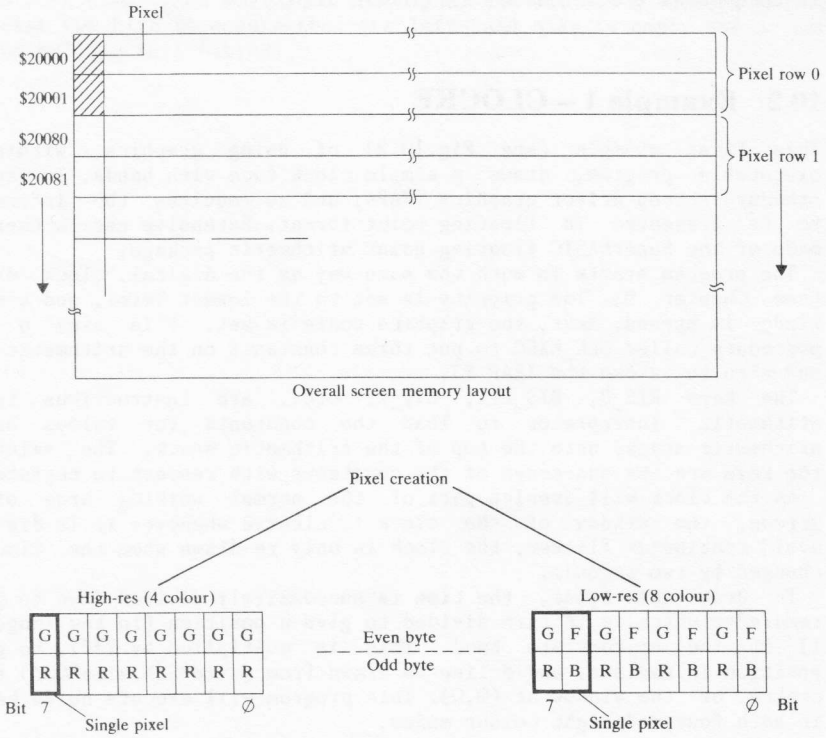

Figure 10.1 Screen memory layout

The screen memory begins at address $20000, and extends for 32K of RAM. The limited colour range in four colour mode means that no flash is available, and turning both green and red bits on will be interpreted by the hardware as white.

Note that 64 words make up any one row of pixels. In high-resolution mode (four colour mode) the screen is, therefore, 512 pixels wide. In low-resolution mode (eight colour mode) the screen is 256 pixels wide. In both modes the screen is 256 pixels high.

10.2 Example 1 – CLOCKF

This first example (see Fig.10.2) of using graphics, within an executable program, draws a simple clock face with hands. It uses the standard screen driver graphics TRAPs, and so requires the information to be presented in floating point format. Extensive use is therefore made of the SuperBASIC floating point arithmetic package.

The program starts in much the same way as the digital clock example (see Chapter 9). The priority is set to the lowest level, and a screen window is opened. Next, the graphics scale is set. This uses a small procedure called CLK_EXEC to put three constants on the arithmetic stack and then to invoke the TRAP #3.

The keys RIS_0, RIS_2PI, RIS_1, etc., are instructions to the arithmetic interpreter to load the constants (or values on the arithmetic stack) onto the top of the arithmetic stack. The values of the keys are the addresses of the constants with respect to register A4.

As the clock will overlap part of the normal working area of the screen, the window of the clock is cleared whenever it is drawn. To avoid continuous flicker, the clock is only re-drawn when the time has changed by two seconds.

To draw the hands, the time is successively divided down to give a remainder which is in turn divided to give a position (in the range 0 to 1) of the appropriate hand. This is multiplied by 2*PI, to give a position in radians, and a line is drawn from (size*SIN,size*COS) to the centre of the window at (0,0). This program will execute quite happily in both four and eight colour modes.

10.3 Example 2 – BALL

This second example (see Fig.10.3) of graphics, within an executable program, accesses the screen directly. This illustrates one of the more complex forms of animation. Each representation of the object is not constrained to be in a limited number of pixel positions, and it can be panned to any position desired.

The bizarre screen organization in eight colour mode, where each pixel is represented by two bits (one bit if you ignore flash) in a byte at an even address, and two bits in the next byte, might appear to make panning an object rather difficult. Fortunately, the MOVEP instruction can be used to read or write alternate bytes! Within the drawing routine, all the green (with flash=0) bits are read into D6 and all the

red and blue bits into D7. Each register then holds the pixel information in the same order as it appears on the screen. After panning, masking, or any other operation, the pixels can then be written into the screen using more MOVEP instructions.

To draw the rolling ball, the ball is written to the same word address in the screen four times. Each time, a new rotation of the ball is used (four in total), and the ball is panned within the word by two bits extra each time. When moving to the right, the ball is always panned by at least two bits to ensure that the left-hand side is empty and no part of the ball is left behind.

Figure 10.2 A real-time analogue clockface display

```
Graphics                                McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                                        0001 *H Graphics
                                        0002 ;
                                        0003 ; CLOCKF - clock face with hands
                                        0004 ;
                                        0005 ; Copyright (c) 1984 McGraw-Hill(UK)
                                        0006 ;
00000000                                0007         ORG 0
                                        0008 ;
FFFFFFFF  =                             0009 MYSELF   EQU   -1
00000005  =                             0010 MT_FRJOB EQU   $05
00000008  =                             0011 MT_SUSJB EQU   $08
0000000B  =                             0012 MT_PRIOR EQU   $0B
00000013  =                             0013 MT_RCLCK EQU   $13
00000020  =                             0014 SD_CLEAR EQU   $20
00000029  =                             0015 SD_SETIN EQU   $29
00000031  =                             0016 SD_LINE  EQU   $31
00000033  =                             0017 SD_ELIPS EQU   $33
00000034  =                             0018 SD_SCALE EQU   $34
                                        0019 ;
000000C8  =                             0020 UT_SCR   EQU   $C8
000000EC  =                             0021 CN_DATE  EQU   $EC
0000011C  =                             0022 RI_EXEC  EQU   $11C
0000011E  =                             0023 RI_EXECB EQU   $11E
00000008  =                             0024 RI_FLOAT EQU   $08
0000000A  =                             0025 RI_ADD   EQU   $0A
0000000E  =                             0026 RI_MULT  EQU   $0E
00000010  =                             0027 RI_DIV   EQU   $10
00000014  =                             0028 RI_NEG   EQU   $14
00000016  =                             0029 RI_DUP   EQU   $16
00000018  =                             0030 RI_COS   EQU   $18
0000001A  =                             0031 RI_SIN   EQU   $1A
                                        0032 ;
FFFFFFFA  =                             0033 RIS_0    EQU   -6
FFFFFFF4  =                             0034 RIS_2PI  EQU   -12
FFFFFFEE  =                             0035 RIS_1    EQU   -18
FFFFFFE8  =                             0036 RIS_SIZE EQU   -24
FFFFFFE2  =                             0037 RIS_ANG  EQU   -30
                                        0038 ;
                                        0039 ; Header bytes for debuggers etc.
                                        0040 ;
00000000 6012                           0041         BRA.S   CLOCK          ;branch to clock code
00000002 00000000                       0042         DEFL    0              ;pad out with 4 bytes
00000006 4AFB                           0043         DEFW    $4AFB          ;standard job flag
00000008 000A                           0044         DEFW    10
0000000A 436C6F636B2066616365 0045      DEFB    'Clock face'
                                        0046         ALIGN
                                        0047 ;
00000014 9DCE                           0048 CLOCK:  SUB.L   A6,A6          ;set A6 to zero forever
                                        0049 ;
00000016 700B                           0050         MOVEQ   #MT_PRIOR,D0   ;set priority
00000018 72FF                           0051         MOVEQ   #MYSELF,D1     ;... of this Job
0000001A 7401                           0052         MOVEQ   #1,D2          ;... to 1 (the lowest)
0000001C 4E41                           0053         TRAP    #1
                                        0054 ;
0000001E 347800C8                       0055         MOVE.W  UT_SCR,A2      ;open window for clock
00000022 43FA008C                       0056         LEA     SCR(PC),A1     ;address of definition
00000026 4E92                           0057         JSR     (A2)
00000028 2F08                           0058         MOVE.L  A0,-(A7)       ;save channel ID
                                        0059 ;
0000002A 43FA01C0                       0060         LEA     RIS_CONS(PC),A1 ;RI stack ptr to top of stack
0000002E 47FA00A4                       0061         LEA     SET_SCALE(PC),A3;pointer to scale block
00000032 49FA01CA                       0062         LEA     RIS_TOP(PC),A4 ;pointer to top of constants
00000036 7E34                           0063         MOVEQ   #SD_SCALE,D7   ;operation to set scale
00000038 6164                           0064         BSR.S   CLK_EXEC       ;set up RI and do graphics
                                        0065 ;
0000003A                                0066 CLOCK_LOOP:
0000003A 7008                           0067         MOVEQ   #MT_SUSJB,D0   ;suspend
0000003C 72FF                           0068         MOVEQ   #MYSELF,D1     ;me
0000003E 760A                           0069         MOVEQ   #10,D3         ;for 1/5 seconds
00000040 93C9                           0070         SUB.L   A1,A1          ;no flag
00000042 4E41                           0071         TRAP    #1
```

```
SEXEC
LEN : 512
DATA: 128
```

```
                                0072 ;
00000044 7013                   0073            MOVEQ   #MT_RCLCK,D0    ;read time into D1
00000046 4E41                   0074            TRAP    #1
00000048 E289                   0075            LSR.L   #1,D1           ;in two second units
0000004A B881                   0076            CMP.L   D1,D4           ;has the time changed?
0000004C 67EC                   0077            BEQ.S   CLOCK_LOOP      ;... no
0000004E 2801                   0078            MOVE.L  D1,D4
                                0079 ;
00000050 82FC5460               0080            DIVU    #21600,D1       ;reduce to 12 hour clock
00000054 4841                   0081            SWAP    D1
00000056 7C00                   0082            MOVEQ   #0,D6
00000058 3C01                   0083            MOVE.W  D1,D6           ;in all of D6
                                0084 ;
0000005A 7020                   0085            MOVEQ   #SD_CLEAR,D0    ;clear old clock face
0000005C 6148                   0086            BSR.S   CLK_TRAP3
                                0087 ;
0000005E 7E33                   0088            MOVEQ   #SD_ELIPS,D7    ;draw ellipse
00000060 43FA018A               0089            LEA     RIS_CONS(PC),A1 ;set RI stack ptr
00000064 47FA0075               0090            LEA     SET_CIRC(PC),A3 ;set circle block
00000068 6134                   0091            BSR.S   CLK_EXEC
                                0092 ;
0000006A 7A06                   0093            MOVEQ   #6,D5           ;start with yellow ink
0000006C 7E31                   0094            MOVEQ   #SD_LINE,D7     ;drawing lines
0000006E 47FA0071               0095            LEA     SET_LINE(PC),A3
00000072 4BFA0048               0096            LEA     CLK_DATA(PC),A5
00000076                        0097 CLK_HAND:
00000076 7029                   0098            MOVEQ   #SD_SETIN,D0    ;set colour of hands
00000078 3205                   0099            MOVE.W  D5,D1
0000007A 612A                   0100            BSR.S   CLK_TRAP3
                                0101 ;
0000007C 43FA016E               0102            LEA     RIS_CONS(PC),A1 ;reset RI stack pointer
00000080 4261                   0103            CLR.W   -(A1)
00000082 231D                   0104            MOVE.L  (A5)+,-(A1)     ;set size of hands
                                0105 ;
00000084 3306                   0106            MOVE.W  D6,-(A1)        ;put new position on stack
00000086 7008                   0107            MOVEQ   #RI_FLOAT,D0    ;float it
00000088 3478011C               0108            MOVE.W  RI_EXEC,A2
0000008C 4E92                   0109            JSR     (A2)
                                0110 ;
0000008E 8CDD                   0111            DIVU    (A5)+,D6        ;divide by next divisor
00000090 4246                   0112            CLR.W   D6
00000092 4846                   0113            SWAP    D6              ;remainder back in D6
                                0114 ;
00000094 331D                   0115            MOVE.W  (A5)+,-(A1)     ;put number of seconds
                                0116                                    ;around face on stack
00000096 6106                   0117            BSR.S   CLK_EXEC        ;and create the complete
                                0118                                    ;set of call parameters
                                0119 ;
00000098 5545                   0120            SUBQ    #2,D5           ;next ink colour
0000009A 6EDA                   0121            BGT.S   CLK_HAND
                                0122 ;
0000009C 609C                   0123            BRA.S   CLOCK_LOOP
                                0124 ;
0000009E                        0125 CLK_EXEC:
0000009E 3478011E               0126            MOVE.W  RI_EXECB,A2     ;execute instructions
000000A2 4E92                   0127            JSR     (A2)
000000A4 2007                   0128            MOVE.L  D7,D0           ;set IO key
000000A6                        0129 CLK_TRAP3:
000000A6 76FF                   0130            MOVEQ   #-1,D3          ;no timeout
000000A8 206F0004               0131            MOVE.L  4(A7),A0        ;set channel ID
000000AC 4E43                   0132            TRAP    #3
000000AE 4E75                   0133            RTS
                                0134 ;
000000B0                        0135 SCR:
000000B0 00000000               0136            DEFL    0               ;no border, black on black
000000B4 003C                   0137            DEFW    60              ;3:2 pixel aspect ratio
000000B6 0028                   0138            DEFW    40
000000B8 01C4                   0139            DEFW    512-60          ;top RHS (512-32-60 for TV mod
000000BA 0000                   0140            DEFW    0               ;       (16 for TV mode)
                                0141 ;
                                0142 ; data for clock face in 2 second units
                                0143 ;
000000BC                        0144 CLK_DATA:
000000BC 08006000               0145            DEFL    $08006000       ;hour hand size = 3/4
000000C0 0708                   0146            DEFW    1800            ;1800 units per hour
000000C2 5460                   0147            DEFW    21600           ;nr of units around the face
000000C4 08007000               0148            DEFL    $08007000       ;minute hand size = 7/8
000000C8 001E                   0149            DEFW    30              ;30 units per minute
```

```
000000CA 0708
000000CC 08014000
000000D0 0001
000000D2 001E

000000D4
000000D4 EE
000000D5 EE
000000D6 0A
000000D7 EE
000000D8 14
000000D9 16
000000DA 00
000000DB
000000DB FA
000000DC FA
000000DD EE
000000DE EE
000000DF FA
000000E0 00
000000E1
000000E1 08

000000E2 10
000000E3 F4
000000E4 0E
000000E5 E2
000000E6 1A
000000E7 E8
000000E8 0E
000000E9 E2
000000EA 18
000000EB E8
000000EC 0E
000000ED FA
000000EE FA
000000EF 00

000000F0
000000F0 00000000000000000000
0000010C 00000000000000000000
0000012C 00000000000000000000
0000014C 00000000000000000000
0000016C 00000000000000000000
0000018C 00000000000000000000
000001AC 00000000000000000000
000001CC 00000000000000000000

000001EC
000001EC 0801
000001EE 40000000
000001F2 0803
000001F4 6487ED51
000001F8 0000
000001FA 00000000
000001FE
```

```
0150            DEFW    1800            ;nr of units round the face
0151            DEFL    $08014000       ;second hand size = 1
0152            DEFW    1               ;1 unit per unit
0153            DEFW    30              ;30 units around the face
0154 ;
0155 SET_SCALE:
0156            DEFB    RIS_1           ;stack   1
0157            DEFB    RIS_1           ;        1,1
0158            DEFB    RI_ADD          ;        2
0159            DEFB    RIS_1           ;        2,1
0160            DEFB    RI_NEG          ;        2,-1
0161            DEFB    RI_DUP          ;        2,-1,-1
0162            DEFB    0
0163 SET_CIRC:
0164            DEFB    RIS_0           ;stack   0
0165            DEFB    RIS_0           ;        0,0
0166            DEFB    RIS_1           ;        0,0,1
0167            DEFB    RIS_1           ;        0,0,1,1
0168            DEFB    RIS_0           ;        0,0,1,1,0
0169            DEFB    0
0170 SET_LINE:
0171            DEFB    RI_FLOAT        ;stack  size, nr,
0172                                    ;   divisor * nr of divs
0173            DEFB    RI_DIV          ;       size, position (0 to 1)
0174            DEFB    RIS_2PI         ;       size, position, 2*PI
0175            DEFB    RI_MULT         ;       size, angle
0176            DEFB    RIS_ANG         ;       ..... angle
0177            DEFB    RI_SIN          ;             x0
0178            DEFB    RIS_SIZE        ;             x0, size
0179            DEFB    RI_MULT         ;             x
0180            DEFB    RIS_ANG         ;             x, angle
0181            DEFB    RI_COS          ;             x, y0
0182            DEFB    RIS_SIZE        ;             x, y0, size
0183            DEFB    RI_MULT         ;             x, y
0184            DEFB    RIS_0           ;             x, y, 0
0185            DEFB    RIS_0           ;             x, y, 0, 0
0186            DEFB    0
0187            ALIGN
0188 ;
0189 RI_STACK:
0190            DEFL    0,0,0,0,0,0,0,0 ;240 bytes plus space
0191            DEFL    0,0,0,0,0,0,0,0
0192            DEFL    0,0,0,0,0,0,0,0
0193            DEFL    0,0,0,0,0,0,0,0
0194            DEFL    0,0,0,0,0,0,0,0
0195            DEFL    0,0,0,0,0,0,0,0
0196            DEFL    0,0,0,0,0,0,0,0
0197            DEFL    0,0,0,0,0,0,0,0
0198 ;
0199 RIS_CONS:
0200            DEFW    $0801           ;one
0201            DEFL    $40000000
0202            DEFW    $0803           ;2*PI
0203            DEFL    $6487ED51
0204            DEFW    $0000           ;0
0205            DEFL    $00000000
0206 RIS_TOP:
0207 ;
0208 END
```

Symbols:

```
000000BC CLK_DATA   0000009E CLK_EXEC   00000076 CLK_HAND   000000A6 CLK_TRAP   00000014 CLOCK
0000003A CLOCK_LO   000000EC CN_DATE    00000005 MT_FRJOB   0000000B MT_PRIOR   00000013 MT_RCLCK
00000008 MT_SUSJB   FFFFFFFF MYSELF     FFFFFFFA RIS_0      FFFFFFEE RIS_1      FFFFFFF4 RIS_2PI
FFFFFFE2 RIS_ANG    000001EC RIS_CONS   FFFFFFE8 RIS_SIZE   000001FE RIS_TOP    0000000A RI_ADD
00000018 RI_COS     00000010 RI_DIV     00000016 RI_DUP     0000011C RI_EXEC    0000011E RI_EXECB
00000008 RI_FLOAT   0000000E RI_MULT    00000014 RI_NEG     0000001A RI_SIN     000000F0 RI_STACK
000000B0 SCR        00000020 SD_CLEAR   00000033 SD_ELIPS   00000031 SD_LINE    00000034 SD_SCALE
00000029 SD_SETIN   000000DB SET_CIRC   000000E1 SET_LINE   000000D4 SET_SCAL   000000C8 UT_SCR
```

```
0000 error(s)  detected
6080 bytes free
```

Figure 10.3 A rolling multi-colour ball display

```
Rolling ball                    McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                                0001 *H Rolling ball
                                0002 ;
                                0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                0004 ;
00000000                        0005          ORG 0
                                0006 ;
00000008  =                     0007 MT_SUSJB EQU    $08
FFFFFFFF  =                     0008 MYSELF   EQU    -1
                                0009 ;
                                0010 ; Header for debuggers etc
                                0011 ;
00000000 600C                   0012          BRA.S   BALL            ;branch to code
00000002 00000000               0013          DEFL    0
00000006 4AFB                   0014          DEFW    $4AFB           ;standard header
00000008 0004                   0015          DEFW    4
0000000A 42616C6C               0016          DEFB    'Ball'
                                0017 ;
0000000E 4BF900020810           0018 BALL:    LEA     $20810,A5       ;start addr of line
                                0019 ;
                                0020 ; First we move right (rolling the ball)
                                0021 ;
00000014 722D                   0022          MOVEQ   #$2D,D1         ;draw 46 complete cycles
00000016 49FA008E               0023 1%:      LEA     BALL_DATA(PC),A4;start of data for cycle
0000001A 7003                   0024          MOVEQ   #3,D0           ;each cycle is 4 long
0000001C 7802                   0025          MOVEQ   #2,D4           ;initial shift is 2
0000001E 6140                   0026 2%:      BSR.S   DRAW_WAIT       ;wait and draw ball
00000020 5444                   0027          ADDQ    #2,D4           ;shift it right a bit
00000022 51C8FFFA               0028          DBRA    D0,2%
                                0029 ;
00000026 544D                   0030          ADDQ    #2,A5           ;move to next word in screen
00000028 51C9FFEC               0031          DBRA    D1,1%
                                0032 ;
                                0033 ; Now we spin a stationary ball
                                0034 ;
0000002C 7808                   0035          MOVEQ   #8,D4           ;with shift of 8
0000002E 554D                   0036          SUBQ    #2,A5           ;at last position
00000030 7213                   0037          MOVEQ   #$13,D1         ;20 complete cycles
00000032 49FA0072               0038 3%:      LEA     BALL_DATA(PC),A4;start of data for cycle
00000036 7003                   0039          MOVEQ   #3,D0           ;each cycle is 4 long
00000038 6126                   0040 4%:      BSR.S   DRAW_WAIT       ;wait and draw ball
0000003A 51C8FFFC               0041          DBRA    D0,4%
0000003E 51C9FFF2               0042          DBRA    D1,3%
                                0043 ;
                                0044 ; Now we move left (rolling the ball backwards)
                                0045 ;
00000042 722D                   0046          MOVEQ   #$2D,D1
00000044 49FA00E4               0047 5%:      LEA     BALL_LAST(PC),A4;start of data for cycle
00000048 7003                   0048          MOVEQ   #3,D0           ;each cycle is 4 long
0000004A 7806                   0049          MOVEQ   #6,D4           ;initial shift is 6
0000004C 6112                   0050 6%:      BSR.S   DRAW_WAIT       ;wait and draw ball
0000004E 5544                   0051          SUBQ    #2,D4           ;shift it left a bit
00000050 98FC0058               0052          SUB.W   #88,A4          ;move back by two objects
00000054 51C8FFF6               0053          DBRA    D0,6%
                                0054 ;
00000058 554D                   0055          SUBQ    #2,A5           ;move to previous word in screen
0000005A 51C9FFE8               0056          DBRA    D1,5%
                                0057 ;
0000005E 60AE                   0058          BRA.S   BALL            ;keep repeating!!!
                                0059 ;
                                0060 ; Wait a while to get smooth movement
                                0061 ;
00000060                        0062 DRAW_WAIT:
00000060 3F00                   0063          MOVE.W  D0,-(A7)        ;save the counters
00000062 3F01                   0064          MOVE.W  D1,-(A7)
00000064 7008                   0065          MOVEQ   #MT_SUSJB,D0    ;suspend
00000066 72FF                   0066          MOVEQ   #MYSELF,D1      ;myself
00000068 7602                   0067          MOVEQ   #2,D3           ;for 2 frames
0000006A 93C9                   0068          SUB.L   A1,A1
0000006C 4E41                   0069          TRAP    #1
0000006E 321F                   0070          MOVE.W  (A7)+,D1
```

```
SEXEC
LEN : 350
DATA:  64
```

```
00000070 301F                    0071            MOVE.W  (A7)+,D0
                                 0072 ;
                                 0073 ; Routine to put an 8*11 pixel object into the screen
                                 0074 ;
                                 0075 ; (A4) is a block of 11 long words holding the
                                 0076 ;      pixels of the object
                                 0077 ; (A5) is the top LHS of the object in the screen
                                 0078 ;      which is shifted right by D4 (512 mode) pixels
                                 0079 ;
                                 0080 ;  A4 is updated by 11 on each call
                                 0081 ;  D5 to D7 are scratch
                                 0082 ;
00000072                         0083 DRAW_8X11:
00000072 2F0D                    0084            MOVE.L  A5,-(A7)        ;preserve screen address
00000074 7A0A                    0085            MOVEQ   #10,D5          ;draw 11 lines
00000076                         0086 DRAW_LOOP:
00000076 7C00                    0087            MOVEQ   #0,D6           ;clear (top ends of)
00000078 7E00                    0088            MOVEQ   #0,D7           ;      working registers
0000007A 0D0C0000                0089            MOVEP.W 0(A4),D6        ;move green and flash into D6
0000007E 0F0C0001                0090            MOVEP.W 1(A4),D7        ;and red and blue into D7
00000082 E8BE                    0091            ROR.L   D4,D6           ;move object round
00000084 E8BF                    0092            ROR.L   D4,D7
00000086 0D8D0000                0093            MOVEP.W D6,0(A5)        ;and put it into the screen
0000008A 0F8D0001                0094            MOVEP.W D7,1(A5)
0000008E E19E                    0095            ROL.L   #8,D6           ;move the missing bit of the
00000090 E19F                    0096            ROL.L   #8,D7           ;object back
00000092 584D                    0097            ADDQ    #4,A5
00000094 1AC6                    0098            MOVE.B  D6,(A5)+        ;and put it into the screen
00000096 1A87                    0099            MOVE.B  D7,(A5)         ;as well
                                 0100 ;
00000098 584C                    0101            ADDQ    #4,A4           ;move to next line of the object
0000009A DAFC007B                0102            ADD.W   #$80-5,A5       ;and of the screen
0000009E 51CDFFD6                0103            DBRA    D5,DRAW_LOOP
                                 0104 ;
000000A2 2A5F                    0105            MOVE.L  (A7)+,A5        ;restore the screen address
000000A4 4E75                    0106            RTS
                                 0107 ;
000000A6                         0108 BALL_DATA:
000000A6 000AA0F0                0109            DEFL    $000AA0F0
000000AA 203AA0F8                0110            DEFL    $203AA0F8
000000AE 283E80E8                0111            DEFL    $283E80E8
000000B2 28BE82EB                0112            DEFL    $28BE82EB
000000B6 02AB2ABF                0113            DEFL    $02AB2ABF
000000BA 22BB22BB                0114            DEFL    $22BB22BB
000000BE A8FE80EA                0115            DEFL    $A8FE80EA
000000C2 82EB28BE                0116            DEFL    $82EB28BE
000000C6 022B28BC                0117            DEFL    $022B28BC
000000CA 0A2F08AC                0118            DEFL    $0A2F08AC
000000CE 0A0F00A0                0119            DEFL    $0A0F00A0
                                 0120 ;
000000D2 080E20B0                0121            DEFL    $080E20B0
000000D6 283E28BC                0122            DEFL    $283E28BC
000000DA 082E20B8                0123            DEFL    $082E20B8
000000DE 02AB80EA                0124            DEFL    $02AB80EA
000000E2 A2FB8AEF                0125            DEFL    $A2FB8AEF
000000E6 A8FE2ABF                0126            DEFL    $A8FE2ABF
000000EA A2FB8AEF                0127            DEFL    $A2FB8AEF
000000EE 02AB80EA                0128            DEFL    $02AB80EA
000000F2 082E20B8                0129            DEFL    $082E20B8
000000F6 283E28BC                0130            DEFL    $283E28BC
000000FA 080E20B0                0131            DEFL    $080E20B0
                                 0132 ;
000000FE 0A0F00A0                0133            DEFL    $0A0F00A0
00000102 0A2F08AC                0134            DEFL    $0A2F08AC
00000106 022B28BC                0135            DEFL    $022B28BC
0000010A 82EB28BE                0136            DEFL    $82EB28BE
0000010E A8FE80EA                0137            DEFL    $A8FE80EA
00000112 88EE88EE                0138            DEFL    $88EE88EE
00000116 02AB2ABF                0139            DEFL    $02AB2ABF
0000011A 28BE82EB                0140            DEFL    $28BE82EB
0000011E 283E80E8                0141            DEFL    $283E80E8
00000122 203AA0F8                0142            DEFL    $203AA0F8
00000126 000AA0F0                0143            DEFL    $000AA0F0
0000012A                         0144 BALL_LAST:
0000012A 020B80E0                0145            DEFL    $020B80E0
0000012E 022B80E8                0146            DEFL    $022B80E8
00000132 223B88EC                0147            DEFL    $223B88EC
00000136 A8FE2ABF                0148            DEFL    $A8FE2ABF
```

```
0000013A 08AE20BA          0149          DEFL    $08AE20BA
0000013E 02AB80EA          0150          DEFL    $02AB80EA
00000142 08AE20BA          0151          DEFL    $08AE20BA
00000146 A8FE2ABF          0152          DEFL    $A8FE2ABF
0000014A 223B88EC          0153          DEFL    $223B88EC
0000014E 022B80E8          0154          DEFL    $022B80E8
00000152 020B80E0          0155          DEFL    $020B80E0
                           0156 ;
                           0157 END

    Symbols:

    0000000E BALL          000000A6 BALL_DAT     0000012A BALL_LAS     00000072 DRAW_8X1     00000076 DRAW_LOO
    00000060 DRAW_WAI      00000008 MT_SUSJB     FFFFFFFF MYSELF

0000 error(s)  detected
6256 bytes free
```