12 FILE HANDLING

In this chapter we are going to look at one, rather larger program, which extends the SuperBASIC language to cater for direct file handling. The program is listed in full as an assembler output list file, and is preceded by a short description. The description tends to rely upon the reader having read and understood the previous examples (in Chapter 11 especially). This keeps repetition to a minimum and enables you to get quickly to the new pertinent points. The source code of the program, and the corresponding '_code' file, 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.

12.1 Using the program

The procedures and functions within the program have to be initialized in order to inform SuperBASIC that they exist. The routine for doing this is demonstrated in the program, and discussed in Secs. 8.4 and 8.5. To physically link the procedures from SuperBASIC, a BOOT file could be created with commands in it of the form:

```
100 base=RESPR(size)
110 LBYTES mdvn_filer_code,base
120 CALL base
130 NEW
```

This first sets up a suitably sized slice of RAM in the resident procedure area. The '_code' file is then loaded into this area and CALLed. This will cause a jump to the start of the procedure file, which in turn simply executes the short initialization routine.

12.2 Example – FILER

In many systems, direct access file handling is made complex by the need to fit data into fixed length record structures. On the QL there is no such problem. It is possible to read or write any amount of data from anywhere in a file. It follows that it is easy to write procedures to handle fixed length records, but, as these would be primarily of historical interest, that is not performed here.

Procedures are defined to GET from and PUT onto files a list of arbitrary type values, to GET and PUT single bytes, and to position the file pointer. One function is defined to return the current file pointer. With these routines it is possible to set up sequential, indexed or linked list file structures. The file slaving algorithms in the QL mean that widely separated parts of a file may be accessed at the same time, and that the file positioning call can be used to pre-fetch parts of a file to increase the efficiency even further.

GET, PUT, BGET, and BPUT are purely re-directable, and will work with any I/O device (such as the NETwork or the SERial ports). BGET and BPUT can be emulated with the QL ROM procedures:

```
BGET #n,x   is the same as   x=CODE(INKEY$(#n,-1))
BPUT #n,x   is the same as   PRINT #n,CHR$(X);
```

The subsidiary routines in this group of procedures are the most important. There is, of course, the old faithful CHANNEL. FSTRG and SSTRG reduce the code size slightly by taking out the common part of getting and putting strings from and to a file. PUT_ON_A1 is a general purpose routine for fetching the value of a parameter of known type, and putting it on the arithmetic stack. SET_TYPE is used to check the parameters one at a time to ensure that each is actually a variable. It also sets up a type flag so that, for example, PUT can put two bytes on a file for an integer parameter, six bytes for a floating point, and '2+n' for a string.

The code would be considerably smaller if some tricks were used to convert the long branches into short branches. But the only tricks used in this code are to do with the stack (A7) handling. In some cases exits are made directly out of routines by just removing the return address by incrementing A7. This can save both code and time.

In the case of SET_POS and POS, the TRAPs are made with immediate return and both the 'not complete' and 'end of file' errors are suppressed. File positioning is always done immediately, and 'not complete' indicates that the required block of the file is, therefore, not yet in RAM.

There is one problem with POS, and that is it returns a long integer. There are no long integers in SuperBASIC! The long integer can be stored in a floating point form without loss of precision, and in this form it must be normalized. The normalization routine used here is fairly slow but simple. The number is simply shifted up until it overflows, and then it is shifted back by one. A faster normalize would be to try shifting by 16 places, then 8, 4, 2, and 1.

Figure 12.1 Direct file access procedures and functions

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

                                        0001 *H FILER
                                        0002 ;
                                        0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                        0004 ;
                                        0005 ; Direct access file handling
                                        0006 ;
                                        0007 ; This file contains the following extensions
                                        0008 ;
                                        0009 ; GET  #n {,variable}   get value(s) from file
                                        0010 ; BGET #n {,variable}   get byte(s) from file
                                        0011 ; PUT  #n {,value}      put values(s) on file
                                        0012 ; BPUT #n {,value}      put byte(s) on file
                                        0013 ; SET_POS #n,value      set file pointer
                                        0014 ; POS (#n)              function to find current
                                        0015 ;                       file pointer
                                        0016 ;
                                        0017 ;
00000000                                0018          ORG 0
                                        0019 ;
FFFFFFFF  =                             0020 ERR_NC   EQU    -1
FFFFFFFA  =                             0021 ERR_NO   EQU    -6
FFFFFFF6  =                             0022 ERR_EF   EQU    -10
FFFFFFF1  =                             0023 ERR_BP   EQU    -15
FFFFFFEE  =                             0024 ERR_OV   EQU    -18
00000001  =                             0025 IO_FBYTE EQU    $01
00000003  =                             0026 IO_FSTRG EQU    $03
00000005  =                             0027 IO_SBYTE EQU    $05
00000007  =                             0028 IO_SSTRG EQU    $07
00000042  =                             0029 FS_POSAB EQU    $42
00000043  =                             0030 FS_POSRE EQU    $43
00000110  =                             0031 BP_INIT  EQU    $110
00000112  =                             0032 CA_GTINT EQU    $112
00000114  =                             0033 CA_GTFP  EQU    $114
00000116  =                             0034 CA_GTSTR EQU    $116
00000118  =                             0035 CA_GTLIN EQU    $118
0000011A  =                             0036 BV_CHRIX EQU    $11A
0000011C  =                             0037 RI_EXEC  EQU    $11C
00000120  =                             0038 BP_LET   EQU    $120
00000008  =                             0039 RI_FLOAT EQU    $08
00000030  =                             0040 BV_CHBAS EQU    $30
00000034  =                             0041 BV_CHP   EQU    $34
00000058  =                             0042 BV_RIP   EQU    $58
                                        0043 ;
                                        0044 ; Entry point for initialization
                                        0045 ;
00000000 43FA000C                       0046 FILES:  LEA     PROC_DEF(PC),A1 ;ptr to proc defs
00000004 34780110                       0047         MOVE.W  BP_INIT,A2
00000008 4E92                           0048         JSR     (A2)
0000000A 7000                           0049         MOVEQ   #0,D0           ;no errors
0000000C 4E75                           0050         RTS                     ;back to BASIC
                                        0051 ;
0000000E                                0052 PROC_DEF:
0000000E 0005                           0053         DEFW    5               ;5 procedures
00000010 0032                           0054 1%:     DEFW    GET-1%          ;offset to entry
00000012 03474554                       0055         DEFB    3,'GET'
                                        0056         ALIGN
00000016 0088                           0057 2%:     DEFW    BGET-2%
00000018 0442474554                     0058         DEFB    4,'BGET'
                                        0059         ALIGN
0000001E 00BC                           0060 3%:     DEFW    PUT-3%
00000020 03505554                       0061         DEFB    3,'PUT'
                                        0062         ALIGN
00000024 00EE                           0063 4%:     DEFW    BPUT-4%
00000026 0442505554                     0064         DEFB    4,'BPUT'
                                        0065         ALIGN
0000002C 010E                           0066 5%:     DEFW    SET_POS-5%
0000002E 075345545F504F53               0067         DEFB    7,'SET_POS'
                                        0068         ALIGN
00000036 0000                           0069         DEFW    0               ;end of procedures
00000038 0001                           0070         DEFW    1               ;1 function
0000003A 011C                           0071 6%:     DEFW    FPOS-6%
0000003C 03504F53                       0072         DEFB    3,'POS'
```

```
SBYTES
LEN : 620
RESPR : 1024
```

```
00000040 0000                0073          DEFW    0               ;end of functions
                             0074 ;
                             0075 ; Get items from a file
                             0076 ;
00000042 6100018C            0077 GET:     BSR     CHAN_SET
00000046                     0078 GET_LOOP:
00000046 61000198            0079          BSR     CHK_RI6         ;check for room on RI stack
0000004A 610001A2            0080          BSR     TYPE_SET        ;get type of next
0000004E 6E32                0081          BGT.S   GET_INT         ;... integer
00000050 6734                0082          BEQ.S   GET_FP          ;... floating point
                             0083 ;
00000052 7402                0084          MOVEQ   #2,D2           ;get length of string
00000054 61000160            0085          BSR     FSTRG_PUSH
00000058 6600014C            0086          BNE     EXIT_8          ;oops
0000005C 38369800            0087          MOVE.W  0(A6,A1.L),D4   ;save it
00000060 7200                0088          MOVEQ   #0,D1
00000062 3204                0089          MOVE.W  D4,D1           ;round up to even byte
00000064 5241                0090          ADDQ.W  #1,D1
00000066 08810000            0091          BCLR    #0,D1
0000006A 2A01                0092          MOVE.L  D1,D5           ;save rounded value
0000006C 61000174            0093          BSR     CHK_RI          ;check for room for string
00000070 92C5                0094          SUB.W   D5,A1           ;move stack pointer down
00000072 3D849800            0095          MOVE.W  D4,0(A6,A1.L)   ;and put length in
00000076 5449                0096          ADDQ    #2,A1
00000078 3404                0097          MOVE.W  D4,D2           ;fetch characters of string
0000007A 6100013C            0098          BSR     FSTRG
0000007E 5549                0099          SUBQ    #2,A1           ;and include string length
00000080 600A                0100          BRA.S   GET_LET         ;in return value.
                             0101 ;
00000082 7402                0102 GET_INT:MOVEQ    #2,D2           ;get two bytes
00000084 6002                0103          BRA.S   GET_BYTES
00000086 7406                0104 GET_FP: MOVEQ    #6,D2           ;get six bytes
                             0105 ;
00000088                     0106 GET_BYTES:
00000088 6100012C            0107          BSR     FSTRG_PUSH      ;push bytes onto A1
0000008C                     0108 GET_LET:
0000008C 66000118            0109          BNE     EXIT_8          ;was there a read error
00000090 2D490058            0110          MOVE.L  A1,BV_RIP(A6)   ;set stack pointer
00000094 34780120            0111          MOVE.W  BP_LET,A2       ;and assign value
00000098 4E92                0112          JSR     (A2)
0000009A 504B                0113          ADDQ    #8,A3           ;move to next parameter
0000009C 60A8                0114          BRA.S   GET_LOOP        ;carry on
                             0115 ;
                             0116 ; Get a byte (and convert to FP if necessary)
                             0117 ;
0000009E 61000130            0118 BGET:    BSR     CHAN_SET        ;set up channel ID etc.
000000A2                     0119 BGET_LOOP:
000000A2 6100013C            0120          BSR     CHK_RI6         ;check for room for 1 FP
000000A6 61000146            0121          BSR     TYPE_SET        ;find type
000000AA 6D0000F8            0122          BLT     ERRR_BP         ;... won't do string
000000AE 1C01                0123          MOVE.B  D1,D6           ;save type flag
000000B0 7401                0124          MOVEQ   #1,D2           ;get one byte
000000B2 61000102            0125          BSR     FSTRG_PUSH
000000B6 660000EE            0126          BNE     EXIT_8          ;oops
000000BA 5349                0127          SUBQ    #1,A1           ;put zero byte on stack
000000BC 42369800            0128          CLR.B   0(A6,A1.L)
000000C0 4A06                0129          TST.B   D6              ;was FP required?
000000C2 6E08                0130          BGT.S   BGET_LET        ;... no
000000C4 7008                0131          MOVEQ   #RI_FLOAT,D0    ;... yes, float it
000000C6 3478011C            0132          MOVE.W  RI_EXEC,A2
000000CA 4E92                0133          JSR     (A2)
000000CC                     0134 BGET_LET:
000000CC 2D490058            0135          MOVE.L  A1,BV_RIP(A6)   ;set arithmetic SP
000000D0 34780120            0136          MOVE.W  BP_LET,A2       ;assign value
000000D4 4E92                0137          JSR     (A2)
000000D6 504B                0138          ADDQ    #8,A3           ;move to next parameter
000000D8 60C8                0139          BRA.S   BGET_LOOP
                             0140 ;
                             0141 ; Put data onto file
                             0142 ;
000000DA 610000F4            0143 PUT:     BSR     CHAN_SET        ;set up the channel ID etc.
000000DE                     0144 PUT_LOOP:
000000DE 6100010E            0145          BSR     TYPE_SET        ;find the type
000000E2 6712                0146          BEQ.S   PUT_FP          ;floating point
000000E4 6E1C                0147          BGT.S   PUT_INT         ;integer
                             0148 ;
000000E6 34780116            0149          MOVE.W  CA_GTSTR,A2     ;get a string
000000EA 610000BE            0150          BSR     PUT_ON_A1       ;just one
000000EE 34369800            0151          MOVE.W  0(A6,A1.L),D2   ;find length
```

```
000000F2 5442
000000F4 6016
000000F6
000000F6 34780114
000000FA 610000AE
000000FE 7406
00000100 600A
00000102
00000102 34780112
00000106 610000A2
0000010A 7402
0000010C
0000010C 610000AE
00000110 60CC



00000112 610000BC
00000116
00000116 B7EF0004
0000011A 6700008A
0000011E 34780112
00000122 61000086
00000126 4A369800
0000012A 6704
0000012C 70EE
0000012E 6076
00000130
00000130 5249
00000132 7401
00000134 61000086
00000138 60DC



0000013A
0000013A 61000094
0000013E 34780118
00000142 4E92
00000144 5343
00000146 665C
00000148 22369800
0000014C 7042
0000014E 7600
00000150 2057
00000152 4E43
00000154 603A



00000156 6178
00000158 BBCB
0000015A 6648
0000015C 61000082
00000160 2849
00000162 7043
00000164 7200
00000166 7600
00000168 2057
0000016A 4E43
0000016C 224C
0000016E 5D49
00000170 42769800
00000174 383C0820
00000178
00000178 5344
0000017A E381
0000017C 6708
0000017E 68F8
00000180 E291
00000182 3D849800
00000186
00000186 2D819802
0000018A 2D490058
0000018E 7802



00000190 4A80
```

```
0152            ADDQ.W  #2,D2           ;put length/string on file
0153            BRA.S   PUT_FILE
0154 PUT_FP:
0155            MOVE.W  CA_GTFP,A2      ;get a floating point
0156            BSR     PUT_ON_A1       ;just one
0157            MOVEQ   #6,D2           ;and put 6 bytes on file
0158            BRA.S   PUT_FILE
0159 PUT_INT:
0160            MOVE.W  CA_GTINT,A2     ;get an integer
0161            BSR     PUT_ON_A1       ;just one
0162            MOVEQ   #2,D2           ;and put 2 bytes on file
0163 PUT_FILE:
0164            BSR     SSTRG           ;put bytes on file
0165            BRA.S   PUT_LOOP        ;carry on
0166 ;
0167 ; Put a byte on the file
0168 ;
0169 BPUT:    BSR     CHAN_SET        ;set up channel ID etc.
0170 BPUT_LOOP:
0171            CMP.L   4(A7),A3        ;end of list?
0172            BEQ     EXIT_8          ;yes (D0 already set)
0173            MOVE.W  CA_GTINT,A2     ;get an integer
0174            BSR     PUT_ON_A1       ;just one
0175            TST.B   0(A6,A1.L)      ;MSByte must be zero
0176            BEQ.S   BPUT_FILE       ;good
0177            MOVEQ   #ERR_OV,D0      ;no, call it overflow
0178            BRA.S   EXIT_8
0179 BPUT_FILE:
0180            ADDQ    #1,A1           ;just the LSByte
0181            MOVEQ   #1,D2
0182            BSR     SSTRG           ;onto the file
0183            BRA.S   BPUT_LOOP
0184 ;
0185 ; Set the file pointer
0186 ;
0187 SET_POS:
0188            BSR     CHAN_SET        ;set up channel ID etc.
0189            MOVE.W  CA_GTLIN,A2     ;and get a long integer
0190            JSR     (A2)
0191            SUBQ.W  #1,D3           ;just one
0192            BNE.S   ERRR_BP
0193            MOVE.L  0(A6,A1.L),D1   ;set file pointer
0194            MOVEQ   #FS_POSAB,D0    ;position file absolute
0195            MOVEQ   #0,D3           ;return immediately
0196            MOVE.L  (A7),A0         ;set channel ID
0197            TRAP    #3
0198            BRA.S   POS_D0          ;check for valid D0 return
0199 ;
0200 ; Get file pointer
0201 ;
0202 FPOS:    BSR.S   CHAN_SET        ;set up channel ID etc
0203            CMP.L   A3,A5           ;should be no other parameters
0204            BNE.S   ERRR_BP
0205            BSR     CHK_RI6         ;make room for return value (FP)
0206            MOVE.L  A1,A4           ;save RI stack pointer
0207            MOVEQ   #FS_POSRE,D0    ;position file relative
0208            MOVEQ   #0,D1           ;by no bytes
0209            MOVEQ   #0,D3           ;and return immediately
0210            MOVE.L  (A7),A0         ;set channel ID
0211            TRAP    #3
0212            MOVE.L  A4,A1           ;restore RI stack pointer
0213            SUBQ    #6,A1           ;put
0214            CLR.W   0(A6,A1.L)      ;zero exponent on
0215            MOVE.W  #$0820,D4       ;set unnormalised exponent (+1)
0216 POS_NORM:
0217            SUBQ.W  #1,D4           ;reduce exponent
0218            ASL.L   #1,D1           ;and multiply mantissa by 2
0219            BEQ.S   POS_MANT        ;... if zero do not carry on
0220            BVC.S   POS_NORM        ;if not overflowed, try again
0221            ROXR.L  #1,D1           ;restore mantissa to non overflowed
0222            MOVE.W  D4,0(A6,A1.L)   ;put actual exponent on RI stack
0223 POS_MANT:
0224            MOVE.L  D1,2(A6,A1.L)   ;and mantissa
0225            MOVE.L  A1,BV_RIP(A6)   ;set RI stack pointer
0226            MOVEQ   #2,D4           ;and return type
0227 ;
0228 ; Test D0 return from file position calls
0229 ;
0230 POS_D0: TST.L   D0               ;OK?
```

```
00000192 6712                0231          BEQ.S   EXIT_8          ;yes
00000194 72FF                0232          MOVEQ   #ERR_NC,D1      ;not complete is quite normal
00000196 B081                0233          CMP.L   D1,D0
00000198 6706                0234          BEQ.S   EXIT_OK8
0000019A 72F6                0235          MOVEQ   #ERR_EF,D1      ;end of file is still OK
0000019C B081                0236          CMP.L   D1,D0
0000019E 6606                0237          BNE.S   EXIT_8          ;no, it is not end of file
000001A0                     0238 EXIT_OK8:
000001A0 7000                0239          MOVEQ   #0,D0           ;yes, set no error
000001A2 6002                0240          BRA.S   EXIT_8
                             0241 ;
000001A4                     0242 ERRR_BP:
000001A4 70F1                0243          MOVEQ   #ERR_BP,D0      ;bad parameter
000001A6                     0244 EXIT_8:
000001A6 504F                0245          ADDQ    #8,A7           ;remove channel ID and top of
000001A8 4E75                0246          RTS                     ;parameter list
                             0247 ;
                             0248 ; Put next item on the A1 stack (A2 set to CA_GT...)
                             0249 ;
000001AA                     0250 PUT_ON_A1:
000001AA 4BEB0008            0251          LEA     8(A3),A5        ;get just one item
000001AE 4E92                0252          JSR     (A2)            ;call appropriate type
000001B0 264D                0253          MOVE.L  A5,A3           ;move onto next item
000001B2 6642                0254          BNE.S   EXIT_12         ;remove return and channel ID
000001B4 4E75                0255          RTS                     ;and saved A5
                             0256 ;
                             0257 ; Fetch bytes
                             0258 ;
000001B6                     0259 FSTRG_PUSH:
000001B6 92C2                0260          SUB.W   D2,A1           ;make room on A1 stack
000001B8 7003                0261 FSTRG:   MOVEQ   #IO_FSTRG,D0    ;fetch known number of bytes
000001BA 6002                0262          BRA.S   TRAP4_3
                             0263 ;
                             0264 ; Send bytes
                             0265 ;
000001BC 7007                0266 SSTRG:   MOVEQ   #IO_SSTRG,D0    ;send known number of bytes
                             0267 ;
000001BE                     0268 TRAP4_3:
000001BE 4E44                0269          TRAP    #4              ;A1 relative to A6
000001C0 76FF                0270          MOVEQ   #-1,D3          ;no timeout
000001C2 206F0004            0271          MOVE.L  4(A7),A0        ;set channel ID
000001C6 4E43                0272          TRAP    #3
000001C8 92C1                0273          SUB.W   D1,A1           ;restore A1 to original
000001CA 4A80                0274          TST.L   D0              ;check error return
000001CC 6628                0275          BNE.S   EXIT_12
000001CE 4E75                0276          RTS
                             0277 ;
                             0278 ; Set up and save channel ID and top of parameter list
                             0279 ;
000001D0                     0280 CHAN_SET:
000001D0 6144                0281          BSR.S   CHANNEL         ;get channel ID
000001D2 6608                0282          BNE.S   GET_OUT         ;return directly
000001D4 2257                0283          MOVE.L  (A7),A1         ;get return address
000001D6 2E8D                0284          MOVE.L  A5,(A7)         ;save top of parameter list
000001D8 2F08                0285          MOVE.L  A0,-(A7)        ;and channel ID
000001DA 4ED1                0286          JMP     (A1)            ;and return
                             0287 ;
000001DC                     0288 GET_OUT:
000001DC 584F                0289          ADDQ    #4,A7           ;remove one return address
000001DE 4E75                0290          RTS                     ;and return to BASIC
                             0291 ;
                             0292 ; Check for room for 6 or D1 bytes on RI stack
                             0293 ;
000001E0                     0294 CHK_RI6:
000001E0 7206                0295          MOVEQ   #6,D1           ;check for 6 bytes
000001E2                     0296 CHK_RI:
000001E2 3478011A            0297          MOVE.W  BV_CHRIX,A2     ;check for room on RI stack
000001E6 4E92                0298          JSR     (A2)
000001E8 226E0058            0299          MOVE.L  BV_RIP(A6),A1   ;set RI stack pointer
000001EC 4E75                0300          RTS
                             0301 ;
                             0302 ; Set up type of next parameter
                             0303 ;
000001EE                     0304 TYPE_SET:
000001EE B7EF0008            0305          CMP.L   8(A7),A3        ;end of parameter list?
000001F2 6D06                0306          BLT.S   TYPE_VAR        ;... no
000001F4 7000                0307          MOVEQ   #0,D0           ;... yes, OK
000001F6                     0308 EXIT_12:
000001F6 584F                0309          ADDQ    #4,A7           ;remove return
```

```
000001F8 60AC                0310          BRA.S   EXIT_8          ;and exit
000001FA                     0311 TYPE_VAR:
000001FA 1236B800            0312          MOVE.B  0(A6,A3.L),D1   ;get name type
000001FE 343C00C5            0313          MOVE.W  #$C5,D2         ;mask of acceptable types
00000202 0302                0314          BTST    D1,D2
00000204 670C                0315          BEQ.S   TYPE_BP         ;not permissable
00000206 720F                0316          MOVEQ   #$F,D1          ;mask out separators
00000208 C236B801            0317          AND.B   1(A6,A3.L),D1
0000020C 6704                0318          BEQ.S   TYPE_BP         ;it is null
0000020E 5501                0319          SUBQ.B  #2,D1           ;set -ve for string, 0 for FP,
00000210 4E75                0320          RTS                     ;+ve for integer
00000212                     0321 TYPE_BP:
00000212 584F                0322          ADDQ    #4,A7           ;remove return
00000214 608E                0323          BRA.S   ERRR_BP         ;and bad parameter
                             0324 ;
                             0325 ; Set default or given channel
                             0326 ; Call parameters    : A3 and A5 standard pointers to name
                             0327 ;                     table for parameters
                             0328 ; Return parameters : D6 pointer to channel table
                             0329 ;                     A0 channel ID
                             0330 ;
00000216                     0331 CHANNEL:
00000216 7C01                0332          MOVEQ   #1,D6           ;default is channel #1
00000218 BBCB                0333          CMP.L   A3,A5           ;are there any parameters?
0000021A 6720                0334          BEQ.S   CHAN_LOOK       ;... no
                             0335 ;
0000021C 08360007B801        0336          BTST    #7,1(A6,A3.L)   ;has first parameter a hash?
00000222 6718                0337          BEQ.S   CHAN_LOOK       ;... no
                             0338 ;
00000224 2F0D                0339          MOVE.L  A5,-(A7)        ;save top parameter pointer
00000226 2A4B                0340          MOVE.L  A3,A5           ;set new top
00000228 504D                0341          ADDQ    #8,A5           ;to 8 bytes above bottom
0000022A 2F0D                0342          MOVE.L  A5,-(A7)        ;(when done, it's new bottom)
0000022C 34780112            0343          MOVE.W  CA_GTINT,A2     ;get an integer
00000230 4E92                0344          JSR     (A2)
00000232 265F                0345          MOVE.L  (A7)+,A3        ;restore parameter pointers
00000234 2A5F                0346          MOVE.L  (A7)+,A5        ;(doesn't affect cond codes)
00000236 661C                0347          BNE.S   CHAN_EXIT       ;was it OK?
00000238 3C369800            0348          MOVE.W  0(A6,A1.L),D6   ;D6 to replace the default
                             0349 ;
0000023C                     0350 CHAN_LOOK:
0000023C CCFC0028            0351          MULU    #$28,D6         ;D6 (long) ptr to chan table
00000240 DCAE0030            0352          ADD.L   BV_CHBAS(A6),D6
00000244 BCAE0034            0353          CMP.L   BV_CHP(A6),D6   ;is it within the table?
00000248 6C0C                0354          BGE.S   ERRR_NO         ;... no
0000024A 20766800            0355          MOVE.L  0(A6,D6.L),A0   ;set channel ID
0000024E 3008                0356          MOVE.W  A0,D0           ;is it open?
00000250 6B04                0357          BMI.S   ERRR_NO         ;... no
00000252 7000                0358          MOVEQ   #0,D0           ;no error
00000254                     0359 CHAN_EXIT:
00000254 4E75                0360          RTS
00000256                     0361 ERRR_NO:
00000256 70FA                0362          MOVEQ   #ERR_NO,D0      ;channel not open
00000258 4E75                0363          RTS
                             0364 ;
                             0365 END
```

Symbols:

```
0000009E BGET          000000CC BGET_LET      000000A2 BGET_LOO      00000112 BPUT          00000130 BPUT_FIL
00000116 BPUT_LOO      00000110 BP_INIT       00000120 BP_LET        00000030 BV_CHBAS      00000034 BV_CHP
0000011A BV_CHRIX      00000058 BV_RIP        00000114 CA_GTFP       00000112 CA_GTINT      00000118 CA_GTLIN
00000116 CA_GTSTR      00000216 CHANNEL       00000254 CHAN_EXI      0000023C CHAN_LOO      000001D0 CHAN_SET
000001E2 CHK_RI        000001E0 CHK_RI6       000001A4 ERRR_BP       00000256 ERRR_NO       FFFFFFF1 ERR_BP
FFFFFFF6 ERR_EF        FFFFFFFF ERR_NC        FFFFFFFA ERR_NO        FFFFFFEE ERR_OV        000001F6 EXIT_12
000001A6 EXIT_8        000001A0 EXIT_OK8      00000000 FILES         00000156 FPOS          000001B8 FSTRG
000001B6 FSTRG_PU      00000042 FS_POSAB      00000043 FS_POSRE      00000042 GET           00000088 GET_BYTE
00000086 GET_FP        00000082 GET_INT       0000008C GET_LET       00000046 GET_LOOP      000001DC GET_OUT
00000001 IO_FBYTE      00000003 IO_FSTRG      00000005 IO_SBYTE      00000007 IO_SSTRG      00000190 POS_DO
00000186 POS_MANT      00000178 POS_NORM      0000000E PROC_DEF      000000DA PUT           0000010C PUT_FILE
000000F6 PUT_FP        00000102 PUT_INT       000000DE PUT_LOOP      000001AA PUT_ON_A      0000011C RI_EXEC
00000008 RI_FLOAT      0000013A SET_POS       000001BC SSTRG         000001BE TRAP4_3       00000212 TYPE_BP
000001EE TYPE_SET      000001FA TYPE_VAR
```

```
0000 error(s)  detected
5EA6 bytes free
```