11 EXTENDING SUPERBASIC

In this chapter we are going to look at four programs, each of which extend the SuperBASIC language in some manner. 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, 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 '_code' 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 full assembly listings will be found to be helpful in a number of ways. First, they act as simple examples of SuperBASIC extension-file creation. Second, 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.

11.1 Using the programs

The procedures and functions within the four programs have to be initialized in order to inform SuperBASIC that they exist. The routine for doing this is demonstrated in the four programs, 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 filename,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.

11.2 Example 1 – CURSOR

This file contains two extension procedures, CURSEN and CURDIS. The short initialization routine is at the very beginning, starting at label EXTEN. The procedure definition table follows this, starting at label PROC_DEF.

The function INKEY$, within the QL ROM, does not enable the cursor. It is possible that an application program written in SuperBASIC may require the cursor to be enabled while polling the keyboard using INKEY$. CURSEN will perform just this very easily! First, set register D0 to SD_CURE, then set D3 to -1 and A0 to the appropriate channel ID, and, finally, perform a TRAP #3. The procedure CURDIS disables the cursor.

Unfortunately, though the principle is easy, life itself is harder than it should be. Owing to one of those inexplicable oversights that occur in new software, the routine in the QL ROM for finding the ID of a SuperBASIC channel is not vectored, and so you have to write it yourself! The complete routine, which will be required for any SuperBASIC function or procedure that uses channels, starts at CHANNEL. Sec.6.3 describes the channel structure.

Note how the channel is checked for being open in the code starting at CHAN_LOOK (just before CHAN_EXIT). First, the channel ID is placed into register A0 (because that is where it is needed!). Second, the least significant word of A0 is copied into a dummy register in order to set the flags. A closed or unopened channel is marked as -1 (long-word). The least significant word should be checked to be greater than or equal to zero for an open channel. The most significant word is, of course, the tag, and for an open channel it could take any value.

11.3 Example 2 – UTILS

Not all procedures are used purely for performing actions. Often they will find or calculate values to be returned. Usually this will be done by a function call to return one value, but, occasionally, more than one value will be required. In this latter case it becomes convenient to return the values through a procedure parameter list. The file in this example contains two functions (MEAN and NHEX$), and one procedure (TIME).

The MEAN function takes one (or more) values, coerced to floating point, and adds them together in a loop using the arithmetic routine RI_ADD. At this point the arithmetic stack has only one floating point number left on it. The number of values is then put on the stack, floated, and divided into the sum. Finally, the return argument address and type are set (D0 is set by the arithmetic routines) and the function returns.

The NHEX$ function is slightly more complex in that, when returning an odd length string, the start of the string must be aligned on a word boundary. First, the two arguments are fetched (in long integer form so there can be eight hex digits). The routine is a little bit careless in allocating the arithmetic stack, because CA_GTLIN will have taken at least 10 bytes to create the two long integers on the stack. Next, the

second long integer is converted (in place!) to eight hex digits. If an odd number of digits is required the characters are moved down by one (note that auto increment cannot be used as all references within the SuperBASIC area must be based on A6). Finally, the word holding the length of the string is put before the string and the return argument address and type are set.

The TIME procedure returns two values giving the time of day in hours and minutes. The critical part of the procedure is the TIME_SET routine. This returns an integer value into either a floating point variable or an integer variable. It will not return a value to a REPeat or FOR identifier. The first part of TIME_SET puts the integer on the arithmetic stack, and then checks if the parameter is either unset, or a variable. Next, it decides whether the integer on the stack needs to be converted into floating point. Finally, the value is assigned using BP_LET.

11.4 Example 3 – ARRAY

Arrays have a pre-defined allocation. Individual elements of an array may be set using BP_LET, but it is very easy to write procedures to modify complete arrays. The examples given here move part of an array (or sub-array) filling the space left behind with zeros. The procedure MAKE_ROOM creates room for extra entries in an array, and TAKE_ROOM removes entries.

The ROOM_SET routine first finds the amount of space to make (or take). It then works its way through the array pointers, finding the base address of the array, the length of each of the most significant elements (e.g., for the array A(20,3) the length is (3+1)*6 bytes), and the number of the elements which require moving.

The rest of the code in the MAKE_ROOM and TAKE_ROOM procedures simply perform block copying and block clearing, with all addressing being based on register A6.

To see the effect of, for example, the MAKE_ROOM routine, try the following (extended) SuperBASIC program:

```
DIM array(9,4)
FOR i=0 to 9
  FOR j=0 to 4
    array(i,j)=10*i+j
  END FOR j
END FOR i

PRINT array,
MAKE_ROOM array(3 to 8),2
PRINT array,
```

Now try a similar test, but this time use a string array.

11.5 Example 4 – JOBS

The final example of adding procedures to SuperBASIC is a set of job control procedures. There is one procedure to write out a list of all the jobs in the QL (JOBS), and there are four procedures to control jobs (SJOB, KJOB, RJOB, and PJOB). These latter four are very similar, the first (or only) two arguments are the job number and the job tag. The number is the index into the table of jobs, and the tag is the job's own identifier. These are combined to form the complete job ID required by QDOS calls.

The JOBS procedure to write a list of jobs has no parameters (except possibly a channel number) and the loop that scans the job tree is simple enough. After the jobs loop there is our old friend CHANNEL and finally the routine to format and write out the job information (JOB_INF).

The output is formatted by filling a buffer with the characters to be sent, then the line is sent to the required channel. The buffer used is the SuperBASIC interpreter's own buffer, which is at least 128 bytes long. The first two bytes are used to hold the integers which are to be added to the buffer. To add a number, the next integer is put in register D1, and the pointer to the end of the field is put in register A5. JOB_NUM puts the number in the start of the buffer and CN_ITOD is used to convert it to characters in the buffer. JOB_NUM then fills (with at least one space) up to A5.

Finally, the start of the job is checked for a standard header, and, if found, the characters of the program identification are copied into the buffer. Note the difference in handling the data in the jobs header (at an absolute address) and the interpreter buffer (address based on A6).

Figure 11.1 Cursor enable/disable extensions

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

                                   0001 *H Extensions to BASIC
                                   0002 ;
                                   0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                   0004 ;
                                   0005 ; This program contains the following extensions:
                                   0006 ;
                                   0007 ; CURSEN  enables the cursor
                                   0008 ; CURDIS  disables the cursor
                                   0009 ;
00000000                           0010          ORG 0
                                   0011 ;
FFFFFFFC  =                        0012 ERR_OR    EQU    -4
FFFFFFFA  =                        0013 ERR_NO    EQU    -6
FFFFFFF1  =                        0014 ERR_BP    EQU    -15
0000000E  =                        0015 SD_CURE   EQU    $E
0000000F  =                        0016 SD_CURS   EQU    $F
00000110  =                        0017 BP_INIT   EQU    $110
00000112  =                        0018 CA_GTINT  EQU    $112
00000028  =                        0019 BV_VVBAS  EQU    $28
00000030  =                        0020 BV_CHBAS  EQU    $30
00000034  =                        0021 BV_CHP    EQU    $34
                                   0022 ;
                                   0023 ; Entry point for initialisation
                                   0024 ;
00000000 43FA000C                  0025 EXTEN:  LEA      PROC_DEF(PC),A1 ;get definitions
00000004 34780110                  0026         MOVE.W   BP_INIT,A2
00000008 4E92                      0027         JSR      (A2)
0000000A 7000                      0028         MOVEQ    #0,D0           ;no errors
0000000C 4E75                      0029         RTS                      ;back to SuperBASIC
                                   0030 ;
0000000E                           0031 PROC_DEF:
0000000E 0002                      0032         DEFW     2               ;2 procedures
00000010 001A                      0033 1%:     DEFW     CURSEN-1%       ;offset to entry
00000012 0643555253454E            0034         DEFB     6,'CURSEN'      ;6 characters in name
                                   0035         ALIGN
0000001A 0014                      0036 2%:     DEFW     CURDIS-2%
0000001C 06435552444953            0037         DEFB     6,'CURDIS'
                                   0038         ALIGN
00000024 0000                      0039         DEFW     0               ;end of procedures
00000026 0000                      0040         DEFW     0               ;0 functions
00000028 0000                      0041         DEFW     0               ;end of functions
                                   0042 ;
                                   0043 ; Enable the cursor
                                   0044 ;
0000002A 780E                      0045 CURSEN: MOVEQ    #SD_CURE,D4     ;set cursor enable key
0000002C 6002                      0046         BRA.S    CUR_COM
                                   0047 ;
                                   0048 ; Disable the cursor
                                   0049 ;
0000002E 780F                      0050 CURDIS: MOVEQ    #SD_CURS,D4     ;set cursor disable key
                                   0051 ;
00000030                           0052 CUR_COM:
00000030 6114                      0053         BSR.S    CHANNEL         ;use routine for ID in A0
00000032 660C                      0054         BNE.S    CUR_EXIT        ;... OK?
00000034 BBCB                      0055         CMP.L    A3,A5           ;should be no parameters
00000036 660A                      0056         BNE.S    ERRR_BP
00000038 1004                      0057         MOVE.B   D4,D0           ;set key in D0
0000003A 363CFFFF                  0058         MOVE.W   #-1,D3          ;set timeout
0000003E 4E43                      0059         TRAP     #3              ;(sets D0 to error code)
00000040                           0060 CUR_EXIT:
00000040 4E75                      0061         RTS
00000042                           0062 ERRR_BP:
00000042 70F1                      0063         MOVEQ    #ERR_BP,D0      ;bad parameter
00000044 4E75                      0064         RTS
                                   0065 ;
                                   0066 ; Set default or given channel
                                   0067 ; Call parameters   : A3 and A5 standard pointers to name
                                   0068 ;                     table for parameters
                                   0069 ; Return parameters : D6 pointer to channel table
                                   0070 ;                     A0 channel ID
                                   0071 ;
```

```
SBYTES
LEN : 140
RESPR : 256
```

```
00000046
00000046 7C01
00000048 BBCB
0000004A 6720

0000004C 08360007B801
00000052 6718

00000054 2F0D
00000056 2A4B
00000058 504D
0000005A 2F0D
0000005C 34780112
00000060 4E92
00000062 265F
00000064 2A5F
00000066 661C
00000068 3C369800

0000006C
0000006C CCFC0028
00000070 DCAE0030
00000074 BCAE0034
00000078 6C0C
0000007A 20766800
0000007E 3008
00000080 6B04
00000082 7000
00000084
00000084 4E75

00000086
00000086 70FA
00000088 4E75
```

```
0072 CHANNEL:
0073          MOVEQ   #1,D6           ;default is channel #1
0074          CMP.L   A3,A5           ;any parameters?
0075          BEQ.S   CHAN_LOOK       ;... no
0076 ;
0077          BTST    #7,1(A6,A3.L)   ;has 1st parameter a hash?
0078          BEQ.S   CHAN_LOOK       ;... no
0079 ;
0080          MOVE.L  A5,-(A7)        ;save top parameter pointer
0081          MOVE.L  A3,A5           ;set new top
0082          ADDQ    #8,A5           ;   to 8 bytes above bottom
0083          MOVE.L  A5,-(A7)        ;   (it will be new bottom)
0084          MOVE.W  CA_GTINT,A2     ;get an integer
0085          JSR     (A2)
0086          MOVE.L  (A7)+,A3        ;restore the pointers
0087          MOVE.L  (A7)+,A5        ;(doesn't affect cond codes)
0088          BNE.S   CHAN_EXIT       ;was it OK?
0089          MOVE.W  0(A6,A1.L),D6   ;get value in D6
0090 ;
0091 CHAN_LOOK:
0092          MULU    #$28,D6         ;make D6 (long) pointer to
0093          ADD.L   BV_CHBAS(A6),D6 ;channel table
0094          CMP.L   BV_CHP(A6),D6   ;is it within the table?
0095          BGE.S   ERR__NO         ;... no
0096          MOVE.L  0(A6,D6.L),A0   ;set channel ID
0097          MOVE.W  A0,D0           ;is it open?
0098          BMI.S   ERR__NO         ;... no
0099          MOVEQ   #0,D0           ;no error
0100 CHAN_EXIT:
0101          RTS
0102 ;
0103 ERR__NO:
0104          MOVEQ   #ERR_NO,D0      ;channel not open
0105          RTS
0106 ;
0107 END
```

```
Symbols:

00000110 BP_INIT    00000030 BV_CHBAS   00000034 BV_CHP     00000028 BV_VVBAS   00000112 CA_GTINT
00000046 CHANNEL    00000084 CHAN_EXI   0000006C CHAN_LOO   0000002E CURDIS     0000002A CURSEN
00000030 CUR_COM    00000040 CUR_EXIT   00000042 ERRR_BP    FFFFFFF1 ERR_BP     FFFFFFFA ERR_NO
FFFFFFFC ERR_OR     00000086 ERR__NO    00000000 EXTEN      0000000E PROC_DEF   0000000E SD_CURE
0000000F SD_CURS

0000 error(s)  detected
61A2 bytes free
```

++++++++++++++++++++

Figure 11.2 General function/procedure parameter passing extensions

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

                               0001 *H Extra BASIC functions
                               0002 ;
                               0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                               0004 ;
                               0005 ; x=MEAN (value{,value})  returns the arithmetic
                               0006 ;                         mean of all the parameters
                               0007 ; x=NHEX$ (number of hex digits,number)
                               0008 ;                         converts number to hex string
                               0009 ;
                               0010 ; TIME hours,minutes      returns time of day (12 hr clock)
                               0011 ;
00000000                       0012          ORG 0
                               0013 ;
00000013  =                    0014 MT_RCLCK EQU    $13
000000FE  =                    0015 CN_ITOHL EQU    $FE
00000110  =                    0016 BP_INIT  EQU    $110
00000114  =                    0017 CA_GTFP  EQU    $114
00000118  =                    0018 CA_GTLIN EQU    $118
```

```
SBYTES
LEN : 300
RESPR : 512
```

```
0000011A  =                 0019 BV_CHRIX EQU    $11A
0000011C  =                 0020 RI_EXEC  EQU    $11C
00000120  =                 0021 BP_LET   EQU    $120
00000008  =                 0022 RI_FLOAT EQU    $08
0000000A  =                 0023 RI_ADD   EQU    $0A
00000010  =                 0024 RI_DIV   EQU    $10
                            0025 ;
00000058  =                 0026 BV_RIP   EQU    $58
                            0027 ;
FFFFFFF1  =                 0028 ERR_BP   EQU    -15
                            0029 ;
                            0030 ;Initialisation
                            0031 ;
00000000 43FA000C           0032         LEA     PROC_TAB(PC),A1 ;procedure definition table
00000004 34780110           0033         MOVE.W  BP_INIT,A2      ;add to BASIC's table
00000008 4E92               0034         JSR     (A2)
0000000A 7000               0035         MOVEQ   #0,D0           ;no - error
0000000C 4E75               0036         RTS
                            0037 ;
0000000E                    0038 PROC_TAB:
0000000E 0001               0039         DEFW    1               ;1 procedure
00000010 00A2               0040 1%:     DEFW    TIME-1%         ;offset
00000012 0454494D45         0041         DEFB    4,'TIME'
                            0042         ALIGN
00000018 0000               0043         DEFW    0               ;end of procedures
0000001A 0002               0044         DEFW    2               ;two functions
0000001C 0012               0045 2%:     DEFW    MEAN-2%
0000001E 044D45414E         0046         DEFB    4,'MEAN'
                            0047         ALIGN
00000024 0042               0048 3%:     DEFW    NHEX-3%
00000026 054E48455824       0049         DEFB    5,'NHEX$'
                            0050         ALIGN
0000002C 0000               0051         DEFW    0               ;end of functions
                            0052 ;
                            0053 ; MEAN function
                            0054 ;
0000002E 34780114           0055 MEAN:   MOVE.W  CA_GTFP,A2      ;get floating point numbers
00000032 4E92               0056         JSR     (A2)
00000034 662A               0057         BNE.S   MEAN_RTS        ;... oops
00000036 3803               0058         MOVE.W  D3,D4           ;save number of parameters
00000038 6728               0059         BEQ.S   ERRR_BP         ;... there were not any
0000003A 5543               0060         SUBQ.W  #2,D3           ;n-1 adds (adjust for DBRA)
0000003C 6D1C               0061         BLT.S   MEAN_SET        ;only one number - return it
0000003E 3478011C           0062         MOVE.W  RI_EXEC,A2      ;now use arithmetic package
00000042                    0063 ADD_LOOP:
00000042 700A               0064         MOVEQ   #RI_ADD,D0      ;add
00000044 4E92               0065         JSR     (A2)
00000046 6618               0066         BNE.S   MEAN_RTS        ;... oops
00000048 51CBFFF8           0067         DBRA    D3,ADD_LOOP     ;... and do another one
                            0068 ;
0000004C 5549               0069         SUBQ    #2,A1           ;number of parameters on stack
0000004E 3D849800           0070         MOVE.W  D4,0(A6,A1.L)
00000052 7008               0071         MOVEQ   #RI_FLOAT,D0    ;float it
00000054 4E92               0072         JSR     (A2)
00000056 7010               0073         MOVEQ   #RI_DIV,D0      ;and divide by it
00000058 4E92               0074         JSR     (A2)
0000005A                    0075 MEAN_SET:
0000005A 2D490058           0076         MOVE.L  A1,BV_RIP(A6)   ;set return argument address
0000005E 7802               0077         MOVEQ   #2,D4           ;and type
00000060                    0078 MEAN_RTS:
00000060 4E75               0079         RTS
00000062                    0080 ERRR_BP:
00000062 70F1               0081         MOVEQ   #ERR_BP,D0
00000064 4E75               0082         RTS
                            0083 ;
                            0084 ; Hex conversion
                            0085 ;
00000066 34780118           0086 NHEX:   MOVE.W  CA_GTLIN,A2     ;get two long integers
0000006A 4E92               0087         JSR     (A2)
0000006C 6642               0088         BNE.S   NHEX_RTS        ;oops
0000006E 5543               0089         SUBQ.W  #2,D3           ;we wanted two arguments
00000070 66F0               0090         BNE.S   ERRR_BP         ;sorry it wasn't 2
00000072 28369800           0091         MOVE.L  0(A6,A1.L),D4   ;number of digits required
00000076 67EA               0092         BEQ.S   ERRR_BP         ;... what none!!
00000078 0C440008           0093         CMP.W   #8,D4           ;we can't do more than 8
0000007C 62E4               0094         BHI.S   ERRR_BP         ;(unsigned greater than)
                            0095 ;
0000007E 2049               0096         MOVE.L  A1,A0           ;2 long's, so room for 8 chars
```

```
00000080 5849
00000082 347800FE
00000086 4E92

00000088 08040000
0000008C 6712
0000008E 3204
00000090 92C1
00000092
00000092 1DB6980098FF
00000098 5249
0000009A 51C9FFF6
0000009E 5549
000000A0
000000A0 92C4
000000A2 5549
000000A4 3D849800
000000A8 2D490058
000000AC 7801
000000AE 7000
000000B0
000000B0 4E75


000000B2 7010
000000B4 D08B
000000B6 908D
000000B8 6662

000000BA 720C
000000BC 3478011A
000000C0 4E92

000000C2 7013
000000C4 4E41
000000C6 82FCA8C0
000000CA 4241
000000CC 4841
000000CE 82FC003C
000000D2 48C1
000000D4 82FC003C
000000D8 2801

000000DA 6108
000000DC 6640
000000DE 504B
000000E0 4844
000000E2 3204
000000E4
000000E4 226E0058
000000E8 92FC0002
000000EC 3D819800

000000F0 4A36B800
000000F4 6708
000000F6 0C360002B800
000000FC 661E
000000FE
000000FE 720F
00000100 C236B801
00000104 5501
00000106 6D14
00000108 6E08
0000010A 7008
0000010C 3478011C
00000110 4E92
00000112
00000112 2D490058
00000116 34780120
0000011A 4ED2
0000011C
0000011C 70F1
0000011E
0000011E 4E75
```

```
0097            ADDQ    #4,A1           ;now only 1 long word there
0098            MOVE.W  CN_ITOHL,A2     ;convert to 8 hex digits
0099            JSR     (A2)
0100                                    ;now the problems!!
0101            BTST    #0,D4           ;is it an odd length string
0102            BEQ.S   NHEX_SET_LEN    ;... no
0103            MOVE.W  D4,D1           ;yes, so have to move it
0104            SUB.W   D1,A1           ;move the pointer
0105 10%:                               ;move a digit
0106            MOVE.B  0(A6,A1.L),0-1(A6,A1.L)
0107            ADDQ    #1,A1
0108            DBRA    D1,10%          ;this moves D1+1 characters
0109            SUBQ    #2,A1           ;add 1 to A1, + 1 'cos it's moved
0110 NHEX_SET_LEN:
0111            SUB.W   D4,A1           ;move A1 to start of string
0112            SUBQ    #2,A1           ;... and a word further on
0113            MOVE.W  D4,0(A6,A1.L)   ;then put string length in
0114            MOVE.L  A1,BV_RIP(A6)   ;set arithmetic stack pointer
0115            MOVEQ   #1,D4           ;... and type string
0116            MOVEQ   #0,D0
0117 NHEX_RTS:
0118            RTS
0119 ;
0120 ; Procedure to return the time of day
0121 ;
0122 TIME:     MOVEQ   #16,D0          ;two parameters?
0123            ADD.L   A3,D0
0124            SUB.L   A5,D0
0125            BNE.S   ERRR_BP1        ;... no
0126 ;
0127            MOVEQ   #12,D1          ;space for 2 floating points
0128            MOVE.W  BV_CHRIX,A2     ;(not going to be very tidy)
0129            JSR     (A2)
0130 ;
0131            MOVEQ   #MT_RCLCK,D0    ;read clock
0132            TRAP    #1
0133            DIVU    #43200,D1       ;get half a day of seconds
0134            CLR.W   D1              ;... without the days
0135            SWAP    D1
0136            DIVU    #60,D1          ;then half a day of minutes
0137            EXT.L   D1              ;... without the seconds
0138            DIVU    #60,D1          ;split into hours and minutes
0139            MOVE.L  D1,D4           ;save minutes (top end of D4)
0140 ;
0141            BSR.S   TIME_SET        ;set one return parameter
0142            BNE.S   TIME_RTS        ;... oops
0143            ADDQ    #8,A3           ;move param. ptr to next
0144            SWAP    D4
0145            MOVE.W  D4,D1           ;set other return parameter
0146 TIME_SET:
0147            MOVE.L  BV_RIP(A6),A1
0148            SUB.W   #2,A1           ;put it on the RI stack
0149            MOVE.W  D1,0(A6,A1.L)
0150 ;
0151            TST.B   0(A6,A3.L)      ;is parameter unset?
0152            BEQ.S   TIME_TYPE       ;... yes, that's alright
0153            CMP.B   #2,0(A6,A3.L)   ;is it a variable?
0154            BNE.S   ERRR_BP1        ;... no, cannot set
0155 TIME_TYPE:
0156            MOVEQ   #$F,D1          ;mask out separators
0157            AND.B   1(A6,A3.L),D1
0158            SUBQ.B  #2,D1           ;see what type it is
0159            BLT.S   ERRR_BP1        ;null or string
0160            BGT.S   TIME_LET        ;integer - no conversion
0161            MOVEQ   #RI_FLOAT,D0    ;floating point - float it
0162            MOVE.W  RI_EXEC,A2
0163            JSR     (A2)
0164 TIME_LET:
0165            MOVE.L  A1,BV_RIP(A6)   ;RI stack ptr to value
0166            MOVE.W  BP_LET,A2       ;set value in data structure
0167            JMP     (A2)
0168 ERRR_BP1:
0169            MOVEQ   #ERR_BP,D0
0170 TIME_RTS:
0171            RTS
0172 ;
0173 END
```

```
Symbols:

00000042 ADD_LOOP   00000110 BP_INIT    00000120 BP_LET     0000011A BV_CHRIX   00000058 BV_RIP
00000114 CA_GTFP    00000118 CA_GTLIN   000000FE CN_ITOHL   00000062 ERRR_BP    0000011C ERRR_BP1
FFFFFFF1 ERR_BP     0000002E MEAN       00000060 MEAN_RTS   0000005A MEAN_SET   00000013 MT_RCLCK
00000066 NHEX       000000B0 NHEX_RTS   000000A0 NHEX_SET   0000000E PROC_TAB   0000000A RI_ADD
00000010 RI_DIV     0000011C RI_EXEC    00000008 RI_FLOAT   000000B2 TIME       00000112 TIME_LET
0000011E TIME_RTS   000000E4 TIME_SET   000000FE TIME_TYP

0000 error(s)  detected
6124 bytes free
```

++++++++++++++++++++

Figure 11.3 Array manipulation procedures

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

                                    0001 *H Extensions to BASIC
                                    0002 ;
                                    0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                    0004 ;
                                    0005 ; This file contains the following extensions
                                    0006 ;
                                    0007 ; MAKE_ROOM array,n      makes room for n extra entries
                                    0008 ;                        in an array
                                    0009 ; TAKE_ROOM array,n      takes n entries out of an array
                                    0010 ;
00000000                            0011          ORG 0
                                    0012 ;
FFFFFFFC  =                         0013 ERR_OR    EQU      -4
FFFFFFFA  =                         0014 ERR_NO    EQU      -6
FFFFFFF1  =                         0015 ERR_BP    EQU      -15
0000000E  =                         0016 SD_CURE   EQU      $E
0000000F  =                         0017 SD_CURS   EQU      $F
00000110  =                         0018 BP_INIT   EQU      $110
00000112  =                         0019 CA_GTINT  EQU      $112
00000028  =                         0020 BV_VVBAS  EQU      $28
00000030  =                         0021 BV_CHBAS  EQU      $30
00000034  =                         0022 BV_CHP    EQU      $34
                                    0023 ;
                                    0024 ; entry point for initialisation
                                    0025 ;
00000000 43FA000C                   0026 EXTEN:  LEA     PROC_DEF(PC),A1 ;get procedure table
00000004 34780110                   0027         MOVE.W  BP_INIT,A2
00000008 4E92                       0028         JSR     (A2)
0000000A 7000                       0029         MOVEQ   #0,D0           ;no errors
0000000C 4E75                       0030         RTS                     ;back to BASIC
                                    0031 ;
0000000E                            0032 PROC_DEF:
0000000E 0004                       0033         DEFW    4               ;2 only - but long names
00000010 001E                       0034 1%:     DEFW    MAKE_ROOM-1%    ;offset to code
00000012 094D414B455F524F4F4D       0035         DEFB    9,'MAKE_ROOM'   ;name
                                    0036         ALIGN
0000001C 0038                       0037 2%:     DEFW    TAKE_ROOM-2%
0000001E 0954414B455F524F4F4D       0038         DEFB    9,'TAKE_ROOM'
00000028 0000                       0039         DEFW    0               ;end of procedures
0000002A 0000                       0040         DEFW    0               ;0 functions
0000002C 0000                       0041         DEFW    0               ;end of functions
                                    0042 ;
                                    0043 ;
                                    0044 ; Array manipulation procedures
                                    0045 ;
0000002E                            0046 MAKE_ROOM:
0000002E 6146                       0047         BSR.S   ROOM_SET        ;set pointers to array
00000030 6620                       0048         BNE.S   MAKE_RTS
00000032 4BF55800                   0049         LEA     0(A5,D5.L),A5   ;make space - from top down
00000036 49F54800                   0050         LEA     0(A5,D4.L),A4   ;set destination pointer
0000003A                            0051 MAKE_MOVE:
0000003A 554C                       0052         SUBQ    #2,A4           ;predecrement
0000003C 554D                       0053         SUBQ    #2,A5           ;and move up
0000003E 3DB6D800C800               0054         MOVE.W  0(A6,A5.L),0(A6,A4.L)
```

SBYTES
LEN : 210
RESPR : 256

```
00000044 5545                0055          SUBQ    #2,D5
00000046 62F2                0056          BHI.S   MAKE_MOVE
00000048                     0057 MAKE_CLEAR:
00000048 554C                0058          SUBQ    #2,A4           ;predecrement
0000004A 4276C800            0059          CLR.W   0(A6,A4.L)      ;and clear the left over bit
0000004E 5544                0060          SUBQ    #2,D4
00000050 62F6                0061          BHI.S   MAKE_CLEAR
00000052                     0062 MAKE_RTS:
00000052 4E75                0063          RTS
                             0064 ;
00000054                     0065 TAKE_ROOM:
00000054 6120                0066          BSR.S   ROOM_SET        ;set pointers to array
00000056 661C                0067          BNE.S   TAKE_RTS
00000058 49F54800            0068          LEA     0(A5,D4.L),A4   ;set source pointer
0000005C                     0069 TAKE_MOVE:                       ;move down
0000005C 3DB6C800D800        0070          MOVE.W  0(A6,A4.L),0(A6,A5.L)
00000062 544C                0071          ADDQ    #2,A4           ;and postincrement
00000064 544D                0072          ADDQ    #2,A5
00000066 5545                0073          SUBQ    #2,D5
00000068 62F2                0074          BHI.S   TAKE_MOVE
0000006A                     0075 TAKE_CLEAR:
0000006A 4276D800            0076          CLR.W   0(A6,A5.L)      ;clear bit at the top
0000006E 544D                0077          ADDQ    #2,A5           ;and postincrement
00000070 5544                0078          SUBQ    #2,D4
00000072 62F6                0079          BHI.S   TAKE_CLEAR
00000074                     0080 TAKE_RTS:
00000074 4E75                0081          RTS
                             0082 ;
                             0083 ; General setup for room routines
                             0084 ; returns D4 distance to move
                             0085 ;         D5 amount to move
                             0086 ;         A5 base address of array
                             0087 ;
00000076                     0088 ROOM_SET:
00000076 504B                0089          ADDQ    #8,A3           ;ignore array for moment
00000078 BBCB                0090          CMP.L   A3,A5           ;any arguments left?
0000007A 6F56                0091          BLE.S   ERR_BP1
0000007C 34780112            0092          MOVE.W  CA_GTINT,A2     ;we need one integer
00000080 4E92                0093          JSR     (A2)
00000082 6648                0094          BNE.S   ROOM_RTS        ;oops
00000084 5343                0095          SUBQ.W  #1,D3           ;just one
00000086 664A                0096          BNE.S   ERR_BP1         ;oops
00000088 38369800            0097          MOVE.W  0(A6,A1.L),D4   ;set distance to move
0000008C 6F40                0098          BLE.S   ERR_OR          ;oops
                             0099 ;
0000008E 514B                0100          SUBQ    #8,A3
00000090 0C360003B800        0101          CMP.B   #3,0(A6,A3.L)   ;it must be an array
00000096 663A                0102          BNE.S   ERR_BP1
00000098 720F                0103          MOVEQ   #$F,D1          ;mask out separators
0000009A C236B801            0104          AND.B   1(A6,A3.L),D1   ;when we get array type
                             0105 ;
0000009E 2A6E0028            0106          MOVE.L  BV_VVBAS(A6),A5 ;get base of VV area
000000A2 2876B804            0107          MOVE.L  4(A6,A3.L),A4
000000A6 D9CD                0108          ADD.L   A5,A4           ;and so base of descriptor
000000A8 DBF6C800            0109          ADD.L   0(A6,A4.L),A5   ;... and base of array
                             0110 ;
000000AC 3C36C808            0111          MOVE.W  8(A6,A4.L),D6   ;get element length
000000B0 5501                0112          SUBQ.B  #2,D1           ;adjust for array type
000000B2 6D08                0113          BLT.S   ROOM_SIZE       ;nothing for strings
000000B4 6E04                0114          BGT.S   ROOM_BY_2       ;*2 for integers
000000B6 CCFC0003            0115          MULU    #3,D6           ;*6 for floating point
000000BA                     0116 ROOM_BY_2:
000000BA DC46                0117          ADD.W   D6,D6           ;assume element length < 64k
000000BC                     0118 ROOM_SIZE:
000000BC 3A36C806            0119          MOVE.W  6(A6,A4.L),D5   ;get total nr of elements
000000C0 5245                0120          ADDQ.W  #1,D5           ;max. dimension +1
000000C2 9A44                0121          SUB.W   D4,D5           ;thus nr of elements to move
000000C4 6F08                0122          BLE.S   ERR_OR
000000C6 C8C6                0123          MULU    D6,D4           ;conv. dist. to move to bytes
000000C8 CAC6                0124          MULU    D6,D5           ;and number of bytes to move
                             0125 ;
000000CA 7000                0126          MOVEQ   #0,D0
000000CC                     0127 ROOM_RTS:
000000CC 4E75                0128          RTS
000000CE                     0129 ERR_OR:
000000CE 70FC                0130          MOVEQ   #ERR_OR,D0
000000D0 4E75                0131          RTS
000000D2                     0132 ERR_BP1:
```

```
000000D2 70F1                0133          MOVEQ   #ERR_BP,D0
000000D4 4E75                0134          RTS
                             0135 ;
                             0136 END

Symbols:

00000110 BP_INIT    00000030 BV_CHBAS   00000034 BV_CHP     00000028 BV_VVBAS   00000112 CA_GTI
FFFFFFF1 ERR_BP     FFFFFFFA ERR_NO     FFFFFFFC ERR_OR     000000D2 ERR_BP1    000000CE ERR_O
00000000 EXTEN      00000048 MAKE_CLE   0000003A MAKE_MOV   0000002E MAKE_ROO   00000052 MAKE_R
0000000E PROC_DEF   000000BA ROOM_BY_   000000CC ROOM_RTS   00000076 ROOM_SET   000000BC ROOM_S
0000000E SD_CURE    0000000F SD_CURS    0000006A TAKE_CLE   0000005C TAKE_MOV   00000054 TAKE_R
00000074 TAKE_RTS

0000 error(s)  detected
6152 bytes free
```

++++++++++++++++++++

Figure 11.4 Job control/display procedures

```
Job control for BASIC          McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                               0001 *H Job control for BASIC
                               0002 ;
                               0003 ; Copyright (c) 1984 McGraw-Hill
                               0004 ;
                               0005 ; This file contains the following extensions:
                               0006 ;
                               0007 ; JOBS [#n]              lists the current jobs
                               0008 ; SJOB nr,tag,time       suppresses a job
                               0009 ; KJOB nr,tag            kills a job
                               0010 ; RJOB nr,tag            releases a job
                               0011 ; PJOB nr,tag,priority   sets a job's priority
                               0012 ;
00000000                       0013          ORG 0
                               0014 ;
FFFFFFFA  =                    0015 ERR_NO    EQU    -6
FFFFFFF1  =                    0016 ERR_BP    EQU    -15
00000002  =                    0017 MT_JINF   EQU    $02
00000005  =                    0018 MT_FRJOB  EQU    $05
00000008  =                    0019 MT_SUSJB  EQU    $08
00000009  =                    0020 MT_RELJB  EQU    $09
0000000B  =                    0021 MT_PRIOR  EQU    $0B
000000F2  =                    0022 CN_ITOD   EQU    $F2
00000110  =                    0023 BP_INIT   EQU    $110
00000112  =                    0024 CA_GTINT  EQU    $112
00000000  =                    0025 BV_BFBAS  EQU    $00
00000030  =                    0026 BV_CHBAS  EQU    $30
00000034  =                    0027 BV_CHP    EQU    $34
00000007  =                    0028 IO_SSTRG  EQU    $07
                               0029 ;
                               0030 ; Entry point for initialisation
                               0031 ;
00000000 43FA000C              0032 JOB:     LEA     PROC_DEF(PC),A1 ;get table pointer
00000004 34780110              0033          MOVE.W  BP_INIT,A2
00000008 4E92                  0034          JSR     (A2)
0000000A 7000                  0035          MOVEQ   #0,D0           ;no errors
0000000C 4E75                  0036          RTS                     ;back to SuperBASIC
                               0037 ;
0000000E                       0038 PROC_DEF:
0000000E 0005                  0039          DEFW    5               ;5 procedures
00000010 0064                  0040 1%:      DEFW    JOBS-1%         ;offset to entry
00000012 044A4F4253            0041          DEFB    4,'JOBS'        ;4 character name
                               0042          ALIGN
00000018 0026                  0043 2%:      DEFW    SJOB-2%
0000001A 04534A4F42            0044          DEFB    4,'SJOB'
                               0045          ALIGN
00000020 0024                  0046 3%:      DEFW    KJOB-3%
00000022 044B4A4F42            0047          DEFB    4,'KJOB'
```

SBYTES
LEN : 400
RESPR : 512

```
00000028 0022
0000002A 04524A4F42

00000030 0020
00000032 04504A4F42

00000038 0000
0000003A 0000
0000003C 0000

0000003E 7808
00000040 7A03
00000042 6010
00000044 7805
00000046 7A02
00000048 600A
0000004A 7809
0000004C 7A02
0000004E 6004
00000050 780B
00000052 7A03

00000054
00000054 34780112
00000058 4E92
0000005A 6644
0000005C B645
0000005E 6642
00000060 22369800
00000064 4841
00000066 36369804
0000006A 3403
0000006C 2004
0000006E 93C9
00000070 4E41
00000072 4E75



00000074 6130
00000076 6628
00000078 BBCB
0000007A 6626
0000007C 2848
0000007E 43FA00F0
00000082 3419
00000084 610000C4
00000088 7200

0000008A 2801
0000008C 7002
0000008E 7400
00000090 4E41
00000092 4A80
00000094 660A
00000096 2A01
00000098 6150
0000009A 6604
0000009C 2205
0000009E 66EA

000000A0
000000A0 4E75

000000A2
000000A2 70F1
000000A4 4E75









000000A6
000000A6 7C01
000000A8 BBCB
```

```
0048            ALIGN
0049 4%:        DEFW    RJOB-4%
0050            DEFB    4,'RJOB'
0051            ALIGN
0052 5%:        DEFW    PJOB-5%
0053            DEFB    4,'PJOB'
0054            ALIGN
0055            DEFW    0               ;end of procedures
0056            DEFW    0               ;0 functions
0057            DEFW    0               ;end of functions
0058 ;
0059 SJOB:      MOVEQ   #MT_SUSJB,D4    ;suspend job
0060            MOVEQ   #3,D5           ;get 3 parameters
0061            BRA.S   JOB_COMMON
0062 KJOB:      MOVEQ   #MT_FRJOB,D4    ;force remove job
0063            MOVEQ   #2,D5           ;get 2 parameters
0064            BRA.S   JOB_COMMON
0065 RJOB:      MOVEQ   #MT_RELJB,D4    ;release job
0066            MOVEQ   #2,D5           ;get 2 parameters
0067            BRA.S   JOB_COMMON
0068 PJOB:      MOVEQ   #MT_PRIOR,D4    ;set job priority
0069            MOVEQ   #3,D5           ;get 3 parameters
0070 ;
0071 JOB_COMMON:
0072            MOVE.W  CA_GTINT,A2     ;get some integers
0073            JSR     (A2)
0074            BNE.S   JOB_EXIT
0075            CMP.W   D5,D3           ;got the right number?
0076            BNE.S   ERRR_BP
0077            MOVE.L  0(A6,A1.L),D1   ;get job ID and tag
0078            SWAP    D1              ;(in the right order)
0079            MOVE.W  4(A6,A1.L),D3   ;get timeout (SJOB)
0080            MOVE.W  D3,D2           ;or priority (PJOB)
0081            MOVE.L  D4,D0           ;set operation key
0082            SUB.L   A1,A1           ;flag address (SJOB) = 0
0083            TRAP    #1
0084            RTS
0085 ;
0086 ; Write a list of jobs to selected or default channel
0087 ;
0088 JOBS:      BSR.S   CHANNEL
0089            BNE.S   JOB_EXIT        ;... OK?
0090            CMP.L   A3,A5           ;should be no parameters
0091            BNE.S   ERRR_BP
0092            MOVE.L  A0,A4           ;save channel ID
0093            LEA     JOB_HEAD(PC),A1 ;write out a header
0094            MOVE.W  (A1)+,D2        ;... set length
0095            BSR     JOB_WRITE
0096            MOVEQ   #0,D1           ;start at job 0
0097 ;
0098 1%:        MOVE.L  D1,D4           ;save this job ID
0099            MOVEQ   #MT_JINF,D0     ;get job information
0100            MOVEQ   #0,D2           ;scan the whole tree
0101            TRAP    #1
0102            TST.L   D0              ;give up if an error
0103            BNE.S   JOB_EXIT
0104            MOVE.L  D1,D5           ;save next job ID
0105            BSR.S   JOB_INF         ;output information on job
0106            BNE.S   JOB_EXIT
0107            MOVE.L  D5,D1           ;if next job is not
0108            BNE.S   1%              ;zero, carry on
0109 ;
0110 JOB_EXIT:
0111            RTS
0112 ;
0113 ERRR_BP:
0114            MOVEQ   #ERR_BP,D0      ;bad parameter
0115            RTS
0116 ;
0117 ; Set default or given channel
0118 ; Call parameters   : A3 and A5 standard pointers to name
0119 ;                     table for parameters
0120 ; Return parameters : D6 pointer to channel table
0121 ;                     A0 channel ID
0122 ;
0123 CHANNEL:
0124            MOVEQ   #1,D6           ;default is channel #1
0125            CMP.L   A3,A5           ;any parameters?
```

```
000000AA 6720                   0126            BEQ.S   CHAN_LOOK       ;... no
                                0127 ;
000000AC 08360007B801           0128            BTST    #7,1(A6,A3.L)   ;has 1st parameter a hash?
000000B2 6718                   0129            BEQ.S   CHAN_LOOK       ;... no
                                0130 ;
000000B4 2F0D                   0131            MOVE.L  A5,-(A7)        ;save top parameter pointer
000000B6 2A4B                   0132            MOVE.L  A3,A5           ;set new top
000000B8 504D                   0133            ADDQ    #8,A5           ;   to 8 bytes above bottom
000000BA 2F0D                   0134            MOVE.L  A5,-(A7)        ;   (it will be new bottom)
000000BC 34780112               0135            MOVE.W  CA_GTINT,A2     ;get an integer
000000C0 4E92                   0136            JSR     (A2)
000000C2 265F                   0137            MOVE.L  (A7)+,A3        ;restore the pointers
000000C4 2A5F                   0138            MOVE.L  (A7)+,A5        ;(doesn't affect cond codes)
000000C6 661C                   0139            BNE.S   CHAN_EXIT       ;was it OK?
000000C8 3C369800               0140            MOVE.W  0(A6,A1.L),D6   ;get value in D6
                                0141 ;
000000CC
000000CC CCFC0028               0142 CHAN_LOOK:
000000D0 DCAE0030               0143            MULU    #$28,D6         ;make D6 (long) pointer to
000000D4 BCAE0034               0144            ADD.L   BV_CHBAS(A6),D6 ;channel table
000000D8 6C0C                   0145            CMP.L   BV_CHP(A6),D6   ;is it within the table?
000000DA 20766800               0146            BGE.S   ERRR_NO         ;... no
000000DE 3008                   0147            MOVE.L  0(A6,D6.L),A0   ;set channel ID
000000E0 6B04                   0148            MOVE.W  A0,D0           ;is channel open?
000000E2 7000                   0149            BMI.S   ERRR_NO         ;... no
000000E4                        0150            MOVEQ   #0,D0           ;no error
000000E4 4E75                   0151 CHAN_EXIT:
                                0152            RTS
                                0153 ;
000000E6                        0154 ERRR_NO:
000000E6 70FA                   0155            MOVEQ   #ERR_NO,D0      ;channel not open
000000E8 4E75                   0156            RTS
                                0157 ;
                                0158 ; Routine to format and write out job information
                                0159 ;
000000EA                        0160 JOB_INF:
000000EA 2C02                   0161            MOVE.L  D2,D6           ;we are about to smash
000000EC 2E03                   0162            MOVE.L  D3,D7           ;the registers,
000000EE 2F08                   0163            MOVE.L  A0,-(A7)        ;and the job address
                                0164 ;
000000F0 206E0000               0165            MOVE.L  BV_BFBAS(A6),A0 ;use the BASIC buffer
000000F4 5448                   0166            ADDQ    #2,A0           ;leave room at bottom for
000000F6 2248                   0167            MOVE.L  A0,A1           ;a RI stack for 1 integer
000000F8 2A48                   0168            MOVE.L  A0,A5           ;and set our field pointer
                                0169 ;
000000FA 3204                   0170            MOVE.W  D4,D1           ;first the job number
000000FC 584D                   0171            ADDQ    #4,A5           ;in field of 4 characters
000000FE 6156                   0172            BSR.S   JOB_NUM
00000100 2204                   0173            MOVE.L  D4,D1           ;now the job tag
00000102 4841                   0174            SWAP    D1              ;which is in the msw
00000104 5E4D                   0175            ADDQ    #7,A5           ;in a field of 7
00000106 614E                   0176            BSR.S   JOB_NUM
00000108 3206                   0177            MOVE.W  D6,D1           ;now the owner number
0000010A 5C4D                   0178            ADDQ    #6,A5           ;in a field of 5+1
0000010C 6148                   0179            BSR.S   JOB_NUM
0000010E 4A87                   0180            TST.L   D7              ;now check if suspended
00000110 6A06                   0181            BPL.S   JOB_W_PR
00000112 1DBC005388FF           0182            MOVE.B  #'S',0-1(A6,A0.L);yes, put in S flag
00000118                        0183 JOB_W_PR:
00000118 7200                   0184            MOVEQ   #0,D1
0000011A 1207                   0185            MOVE.B  D7,D1           ;now priority (byte)
0000011C 584D                   0186            ADDQ    #4,A5           ;in a field of 4
0000011E 6136                   0187            BSR.S   JOB_NUM
                                0188 ;
                                0189 ; All the numbers are in - now check for a name
                                0190 ; (max 60 characters)
                                0191 ;
00000120 7416                   0192            MOVEQ   #22,D2          ;21 chars are in buffer (+LF)
00000122 245F                   0193            MOVE.L  (A7)+,A2        ;restore job base address
00000124 5C4A                   0194            ADDQ    #6,A2           ;check bytes 6 and 7
00000126 0C5A4AFB               0195            CMP.W   #$4AFB,(A2)+    ;... for flag
0000012A 6616                   0196            BNE.S   JOB_INF_DONE    ;no flag
0000012C 321A                   0197            MOVE.W  (A2)+,D1        ;get length
0000012E 0C41003C               0198            CMP.W   #60,D1          ;is it too long?
00000132 620E                   0199            BHI.S   JOB_INF_DONE    ;... yes, forget it
00000134 D441                   0200            ADD.W   D1,D2           ;now we've some more
00000136 6006                   0201            BRA.S   2%              ;characters to go in.
00000138 1D9A8800               0202 1%:        MOVE.B  (A2)+,0(A6,A0.L);copy characters of name
0000013C 5248                   0203            ADDQ    #1,A0
```

```
0000013E 51C9FFF8              0204 2%:     DBRA    D1,1%
                               0205 ;
00000142                       0206 JOB_INF_DONE:
00000142 1DBC000A8800          0207         MOVE.B  #$A,0(A6,A0.L)  ;put <LF> at end
00000148 4E44                  0208         TRAP    #4              ;A1 is relative to A6
0000014A                       0209 JOB_WRITE:
0000014A 7007                  0210         MOVEQ   #IO_SSTRG,D0    ;send string
0000014C 76FF                  0211         MOVEQ   #-1,D3          ;no timeout
0000014E 204C                  0212         MOVE.L  A4,A0           ;restore channel ID
00000150 4E43                  0213         TRAP    #3
00000152 4A80                  0214         TST.L   D0              ;check error
00000154 4E75                  0215         RTS
                               0216 ;
                               0217 ; Put an integer into a line and space along to end of field
                               0218 ;       (A6,A1.L) points to base of buffer, A1 is preserved
                               0219 ;       (A6,A0.L) points to buffer
                               0220 ;       (A6,A5.L) points to end of field
                               0221 ;
00000156                       0222 JOB_NUM:
00000156 5549                  0223         SUBQ    #2,A1           ;make room for integer
00000158 3D819800              0224         MOVE.W  D1,0(A6,A1.L)   ;and put integer in
0000015C 347800F2              0225         MOVE.W  CN_ITOD,A2      ;convert integer to decimal
00000160 4E92                  0226         JSR     (A2)
00000162                       0227 JOB_N_LOOP:
00000162 1DBC00208800          0228         MOVE.B  #' ',0(A6,A0.L) ;move a space in
00000168 5248                  0229         ADDQ    #1,A0           ;and move buffer pointer on
0000016A BBC8                  0230         CMP.L   A0,A5           ;have we filled field yet?
0000016C 62F4                  0231         BHI.S   JOB_N_LOOP      ;try again
0000016E 4E75                  0232         RTS
                               0233 ;
                               0234 ; JOBS heading line
                               0235 ;
00000170                       0236 JOB_HEAD:
00000170 001A                  0237         DEFW    26
00000172 4A6F62207461736B2020 0238         DEFB    'Job tag    owner priority',$A
                               0239 ;
                               0240 END
```

Symbols:

```
00000110 BP_INIT     00000000 BV_BFBAS    00000030 BV_CHBAS    00000034 BV_CHP      00000112 CA_GTINT
000000A6 CHANNEL     000000E4 CHAN_EXI    000000CC CHAN_LOO    000000F2 CN_ITOD     000000A2 ERRR_BP
000000E6 ERRR_NO     FFFFFFF1 ERR_BP      FFFFFFFA ERR_NO      00000007 IO_SSTRG    00000000 JOB
00000074 JOBS        00000054 JOB_COMM    000000A0 JOB_EXIT    00000170 JOB_HEAD    000000EA JOB_INF
00000142 JOB_INF_    00000156 JOB_NUM     00000162 JOB_N_LO    0000014A JOB_WRIT    00000118 JOB_W_PR
00000044 KJOB        00000005 MT_FRJOB    00000002 MT_JINF     0000000B MT_PRIOR    00000009 MT_RELJB
00000008 MT_SUSJB    00000050 PJOB        0000000E PROC_DEF    0000004A RJOB        0000003E SJOB
```

```
0000 error(s)  detected
6098 bytes free
```