PART 3 Programming Examples

9 SIMPLE EXECUTABLE PROGRAMS

In this chapter we are going to look at three simple executable programs (i.e., the programs are created as jobs and executed by using the SuperBASIC EXEC command). The three programs are:

1. MESSAGE - Writes a message to the screen

2. CHOICE - Allows a keyboard input to select a choice of messages to be printed on the screen

3. CLOCKS - Produces a real-time digital clock display

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 '_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 full assembly listings will be found to be helpful in a number of ways. First, they act as simple examples of executable file program creation. Second, for those of you who are relatively new to assembly language programming with the 68000, they provide many examples of 68000 opcodes. Third, 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.

9.1 Example 1 – MESSAGE

This first example of an executable program (see Fig.9.1) illustrates several important points. The first is the amount of code (and junk?) that is required to create even a very simple program. When you write a program in a high-level language, such as SuperBASIC, most of this is hidden. In assembly language it is invariably all too clear.

At the start of the program there are declarations of a number of

constants that will be used within the program. These declarations are not absolutely necessary, but the use of the constants within the code, instead of actual values, is usually considered to be that nebulous thing called 'accepted good practice'. Clearly these declarations do not use any space in the program.

The declarations are followed by the job header. When the program is executed (via the SuperBASIC EXEC command), the program counter will be set to the start of the program, so that the first instruction to be executed will be the branch around the header. This is not a mistake! The header is there merely to identify the program. It is not essential and it may be omitted. The 'JOBS' example given in Chapter 11 will, however, produce a more informative output if a job has a header.

The first action of the program, starting at label MESSAGE is to set up a screen (a window without access to the keyboard). The address of the screen definition block is loaded into register A1 (using the LEA - load effective address instruction), and the utility routine UT_SCR is called to open it, set the colours, set the border, and clear it. Note that UT_SCR tests the error code in D0 before returning, so all that is required to be done on return is to branch to the end of the program if the condition codes are non-zero.

Next, the program sets up large characters (just for its own window!). UT_SCR returns the channel ID in register A0, so it is readily available for the TRAP #3 and the call to UT_MTEXT. The call to UT_MTEXT is just like the call to UT_SCR (i.e., the address of the message is loaded into A1 and the routine is called).

At the end of the program, the error code (from UT_SCR or UT_MTEXT) is moved to register D3, and the job is force removed from the QL.

9.2 Example 2 – CHOICE

The second example of a job additionally illustrates the reading of single characters from the keyboard, as well as error reporting. The program (shown in Fig.9.2) must be executed via the SuperBASIC EXEC_W command if the use of CTRL-C to switch input queues is to be avoided. (When there is more than one window that is expecting input, the keyboard can be switched from window to window. If you ever end up without a flashing cursor try typing CTRL-C).

In this program, which starts at label CHOICE, the utility UT_CON is used to open a console (a window with a keyboard queue). Next, a prompt is written to the new console with UT_MTEXT. Note that in early QDOS ROM versions UT_MTEXT does not test its own error return.

The cursor is now enabled. If there is no other console with a cursor enabled, the keyboard will be automatically directed to this console, and a cursor the size of one character will appear in the window.

A further TRAP is now made to read a byte into register D1. The time out is specified as 500 frames (10 seconds on a 50Hz system). If nothing is typed within 10 seconds, the trap will return with the error 'not complete'.

The program now assumes that the user has not typed either 1 or 2 and sets the appropriate error code (ERR_NF - 'not found'). The program then checks that D1 is not less than 1, then whether it is greater than 2.

Following this bit of check code there is some trickery! The program loads the address of the first message (this has an address register as a destination so the condition codes are not changed), then it checks to see if D1 had been less than 2, and, if not, loads the address of the second message.

9.3 Example 3 – CLOCKS

The third example of a simple executable job is a digital clock. This clock is located in the top right-hand side of the command window, wherever that may be.

The first action of the program, which starts at label CLOCK, is to set register A6 to zero. This is done because the program will never need A6, but the date conversion routine uses A6 as a base address. If A6 is zero, then the date conversion will use absolute addresses.

The next action of the job is to set its own priority to 1. This makes it a background job, placing a very low load on the machine. Next, the program opens a window and sets the colours. This is done explicity because, at this point, the program does not know where the window is going to be! Note that UT_SCR would clear the arbitrary window if it were used to open it.

The loop (beginning at the local label 10%) starts off with a TRAP to suspend the job. As the priority of the job is already the lowest possible, this might seem rather irrelevant. In fact, since the job never has to wait for I/O, it would take a higher priority than a job which is always having to wait for I/O. Suspending the job reduces the load on the machine even further.

The next operation in the loop is, perhaps, rather strange. It is a screen driver EXTOP. The purpose of EXTOPs is to allow application programs to add functions to the standard screen driver. In this case, the EXTOP code is a routine called GET_WIND. GET_WIND is written as if it is part of a device driver. Within the routine the register A0 points to the screen definition block for the jobs window (it is no longer the ID) and A6 points to the base of the system variables. The first action in the routine is to find the base address of the channel 0 window. It transfers the X and Y origins to its own definition block, sets the X (dependent on character size) and Y size, and re-calculates the X origin (given as: window_0_X_origin + window_0_width - own_window_width). See Sec.6.3 for a discussion of screen channel definition blocks.

The time is read into register D1 and converted to a standard string by CN_DATE. The standard string format has the byte count in the first word of the string, so this is moved to D2 (the I/O call string length register) before writing the string to the window. Both MT_SUSJB and MT_RCLCK destroy A0, so it is necessary to restore A0 before calling the TRAP #3s.

The last routine in the program is a simple routine to kill the job if the error code is non-zero.

Figure 9.1 MESSAGE – Simple message program

```
Job to write a message          McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                                0001 *H Job to write a message
                                0002 ;
                                0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                0004 ;
00000000                        0005         ORG 0
                                0006 ;
FFFFFFFF  =                     0007 MYSELF   EQU    -1
00000005  =                     0008 MT_FRJOB EQU    $05
0000002D  =                     0009 SD_SETSZ EQU    $2D
000000C8  =                     0010 UT_SCR   EQU    $C8
000000D0  =                     0011 UT_MTEXT EQU    $D0
                                0012 ;
                                0013 ; Header for debuggers etc.
                                0014 ;
00000000 6010                   0015         BRA.S   MESSAGE          ;branch to code
00000002 00000000               0016         DEFL    0
00000006 4AFB                   0017         DEFW    $4AFB            ;standard header
00000008 0007                   0018         DEFW    7
0000000A 4D657373616765         0019         DEFB    'Message'
                                0020         ALIGN
                                0021 ;
00000012                        0022 MESSAGE:
00000012 43FA0026               0023         LEA     SCR(PC),A1       ;set up a screen
00000016 387800C8               0024         MOVE.W  UT_SCR,A4
0000001A 4E94                   0025         JSR     (A4)
0000001C 6614                   0026         BNE.S   SUICIDE          ;check error return
                                0027 ;
0000001E 702D                   0028         MOVEQ   #SD_SETSZ,D0     ;set character size
00000020 7203                   0029         MOVEQ   #3,D1            ;wide
00000022 7401                   0030         MOVEQ   #1,D2            ;tall
00000024 76FF                   0031         MOVEQ   #-1,D3           ;no timeout
00000026 4E43                   0032         TRAP    #3
                                0033 ;
00000028 43FA001C               0034         LEA     HALLO(PC),A1     ;write a message
0000002C 387800D0               0035         MOVE.W  UT_MTEXT,A4
00000030 4E94                   0036         JSR     (A4)
                                0037 ;
00000032                        0038 SUICIDE:
00000032 2600                   0039         MOVE.L  D0,D3            ;notify any error
00000034 7005                   0040         MOVEQ   #MT_FRJOB,D0     ;force remove
00000036 72FF                   0041         MOVEQ   #MYSELF,D1       ;myself
00000038 4E41                   0042         TRAP    #1
0000003A                        0043 SCR:
0000003A FF                     0044         DEFB    $FF              ;checkerboard border
0000003B 04                     0045         DEFB    $04              ;4 pixels wide
0000003C 04                     0046         DEFB    $04              ;green background
0000003D 00                     0047         DEFB    $00              ;black letters
0000003E 00C8                   0048         DEFW    200              ;200 pixels wide
00000040 0023                   0049         DEFW    35               ;35  high
00000042 009C                   0050         DEFW    156              ;in the middle
00000044 0064                   0051         DEFW    100
00000046                        0052 HALLO:
00000046 0005                   0053         DEFW    5
00000048 48616C6C6F             0054         DEFB    'Hallo'
                                0055 ;
                                0056 END

Symbols:

00000046 HALLO      00000012 MESSAGE    00000005 MT_FRJOB   FFFFFFFF MYSELF     0000003A SCR
0000002D SD_SETSZ   00000032 SUICIDE    000000D0 UT_MTEXT   000000C8 UT_SCR

0000 error(s)  detected
6270 bytes free
```

SEXEC
LEN : 80
DATA: 64

Figure 9.2 CHOICE – Select a message program

```
Job to write one of 2 messages  McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                                0001 *H Job to write one of 2 messages
                                0002 ;
                                0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                0004 ;
00000000                        0005          ORG 0
                                0006 ;
FFFFFFFF  =                     0007 MYSELF    EQU    -1
FFFFFFF9  =                     0008 ERR_NF    EQU    -7
00000005  =                     0009 MT_FRJOB  EQU    $05
00000001  =                     0010 IO_FBYTE  EQU    $01
0000000E  =                     0011 SD_CURE   EQU    $0E
000000C6  =                     0012 UT_CON    EQU    $C6
000000D0  =                     0013 UT_MTEXT  EQU    $D0
                                0014 ;
                                0015 ; Header for debuggers etc.
                                0016 ;
00000000 600E                   0017          BRA.S   CHOICE          ;branch to code
00000002 00000000               0018          DEFL    0
00000006 4AFB                   0019          DEFW    $4AFB           ;standard header
00000008 0006                   0020          DEFW    6
0000000A 43686F696365           0021          DEFB    'Choice'
                                0022          ALIGN
                                0023 ;
00000010 43FA004A               0024 CHOICE: LEA      CON(PC),A1      ;set up a screen
00000014 387800C6               0025          MOVE.W  UT_CON,A4
00000018 4E94                   0026          JSR     (A4)
0000001A 6638                   0027          BNE.S   SUICIDE         ;check error return
                                0028 ;
0000001C 43FA004A               0029          LEA     MESSAGE(PC),A1  ;write prompt
00000020 387800D0               0030          MOVE.W  UT_MTEXT,A4
00000024 4E94                   0031          JSR     (A4)
00000026 4A80                   0032          TST.L   D0
00000028 662A                   0033          BNE.S   SUICIDE         ;check error return
                                0034 ;
0000002A 700E                   0035          MOVEQ   #SD_CURE,D0     ;enable cursor
0000002C 76FF                   0036          MOVEQ   #-1,D3
0000002E 4E43                   0037          TRAP    #3
                                0038 ;
00000030 7001                   0039          MOVEQ   #IO_FBYTE,D0    ;fetch a byte
00000032 363C01F4               0040          MOVE.W  #500,D3         ;wait 10s for reply
00000036 4E43                   0041          TRAP    #3
00000038 4A80                   0042          TST.L   D0              ;check error return
0000003A 6618                   0043          BNE.S   SUICIDE
                                0044 ;
0000003C 70F9                   0045          MOVEQ   #ERR_NF,D0      ;assume reply is in error
0000003E 04010031               0046          SUB.B   #'1',D1         ;compare against 1
00000042 6D10                   0047          BLT.S   SUICIDE         ;... it's too small
00000044 5301                   0048          SUBQ.B  #1,D1           ;compare against 2
00000046 6E0C                   0049          BGT.S   SUICIDE         ;... it's too large
00000048 43FA002E               0050          LEA     MESS1(PC),A1    ;assume message 1
0000004C 6D04                   0051          BLT.S   WRITE           ;was it 1?
0000004E 43FA0034               0052          LEA     MESS2(PC),A1    ;no, it is message 2
00000052                        0053 WRITE:
00000052 4E94                   0054          JSR     (A4)            ;write message
                                0055 ;
00000054                        0056 SUICIDE:
00000054 2600                   0057          MOVE.L  D0,D3           ;notify any error
00000056 7005                   0058          MOVEQ   #MT_FRJOB,D0    ;force remove
00000058 72FF                   0059          MOVEQ   #MYSELF,D1      ;myself
0000005A 4E41                   0060          TRAP    #1
                                0061 ;
0000005C 7F                     0062 CON:     DEFB    $7F             ;horizontal stripes
0000005D 04                     0063          DEFB    $04             ;4 pixels wide
0000005E 02                     0064          DEFB    $02             ;red background
0000005F 07                     0065          DEFB    $07             ;white letters
00000060 00C8                   0066          DEFW    200             ;200 pixels wide
00000062 0023                   0067          DEFW    35              ;35  high
00000064 009C                   0068          DEFW    156             ;in the middle
00000066 0064                   0069          DEFW    100
                                0070 ;
00000068 000E                   0071 MESSAGE:DEFW     14
0000006A 4B657920696E2031206F   0072          DEFB    'Key in 1 or 2'
```

SEXEC
LEN : 150
DATA: 64

```
00000077 0A                          0073          DEFB    $A              ;new line
                                     0074          ALIGN
00000078 000A                        0075 MESS1:   DEFW    10
0000007A 57656C6C20646F6E6521        0076          DEFB    'Well done!'
                                     0077          ALIGN
00000084 000A                        0078 MESS2:   DEFW    10
00000086 5665727920676F6F6421        0079          DEFB    'Very good!'
                                     0080 ;
                                     0081 END


Symbols:

00000010 CHOICE     0000005C CON         FFFFFFF9 ERR_NF      00000001 IO_FBYTE    00000078 MESS1
00000084 MESS2      00000068 MESSAGE     00000005 MT_FRJOB    FFFFFFFF MYSELF      0000000E SD_CUR
00000054 SUICIDE    000000C6 UT_CON      000000D0 UT_MTEXT    00000052 WRITE

0000 error(s)  detected
6220 bytes free
```

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

Figure 9.3 CLOCKS – Real-time digital clock display

```
Clock - clock in window 0       McGraw-Hill(UK) 68000 Ass v1.0A   Page: 0001

                                0001 *H Clock - clock in window 0
                                0002 ;
                                0003 ; Copyright (c) 1984 McGraw-Hill(UK)
                                0004 ;
00000000                        0005          ORG 0
                                0006 ;
FFFFFFFF  =                     0007 MYSELF   EQU   -1
00000005  =                     0008 MT_FRJOB EQU   $05
00000008  =                     0009 MT_SUSJB EQU   $08
0000000B  =                     0010 MT_PRIOR EQU   $0B
00000013  =                     0011 MT_RCLCK EQU   $13
00000001  =                     0012 IO_OPEN  EQU   $01
00000007  =                     0013 IO_SSTRG EQU   $07
00000009  =                     0014 SD_EXTOP EQU   $09
00000011  =                     0015 SD_TAB   EQU   $11
00000028  =                     0016 SD_SETST EQU   $28
00000029  =                     0017 SD_SETIN EQU   $29
                                0018 ;
00000078  =                     0019 SV_CHBAS EQU   $78
                                0020 ;
00000018  =                     0021 SD_XMIN  EQU   $18
0000001C  =                     0022 SD_XSIZE EQU   $1C
0000001E  =                     0023 SD_YSIZE EQU   $1E
00000026  =                     0024 SD_XINC  EQU   $26
                                0025 ;
000000EC  =                     0026 CN_DATE  EQU   $EC
                                0027 ;
                                0028 ; Insert standard header ID for any debuggers etc.
                                0029 ;
00000000 600E                   0030          BRA.S   CLOCK           ;branch to clock code
00000002 00000000               0031          DEFL    0               ;pad out with 4 bytes
00000006 4AFB                   0032          DEFW    $4AFB           ;standard job flag
00000008 0005                   0033          DEFW    5               ;name is 5 bytes long
0000000A 436C6F636B             0034          DEFB    'Clock'
                                0035          ALIGN
                                0036 ;
00000010 9DCE                   0037 CLOCK:   SUB.L   A6,A6           ;don't need A6 so clear it
00000012 700B                   0038          MOVEQ   #MT_PRIOR,D0    ;set priority
00000014 72FF                   0039          MOVEQ   #MYSELF,D1      ;... of this Job
00000016 7401                   0040          MOVEQ   #1,D2           ;... to 1 (the lowest)
00000018 4E41                   0041          TRAP    #1
                                0042 ;
0000001A 7001                   0043          MOVEQ   #IO_OPEN,D0     ;open window for clock
0000001C 72FF                   0044          MOVEQ   #MYSELF,D1      ;... owned by this Job
0000001E 7600                   0045          MOVEQ   #0,D3           ;... (it's a device)
00000020 41FA0084               0046          LEA     SCR(PC),A0      ;address of name
00000024 4E42                   0047          TRAP    #2
```

```
SEXEC
LEN : 200
DATA:  90
```

```
00000026 6148                0048          BSR.S   OOPS            ;any errors?
00000028 2848                0049          MOVE.L  A0,A4           ;save channel ID
                             0050 ;
0000002A 7028                0051          MOVEQ   #SD_SETST,D0    ;set strip
0000002C 7210                0052          MOVEQ   #$10,D1         ;... to Burgandy
0000002E 76FF                0053          MOVEQ   #-1,D3
00000030 4E43                0054          TRAP    #3
                             0055 ;
00000032 7029                0056          MOVEQ   #SD_SETIN,D0    ;set ink
00000034 7204                0057          MOVEQ   #$4,D1          ;... to green
00000036 4E43                0058          TRAP    #3
                             0059 ;
00000038 7008                0060 10%:     MOVEQ   #MT_SUSJB,D0    ;suspend
0000003A 72FF                0061          MOVEQ   #MYSELF,D1      ;myself
0000003C 760A                0062          MOVEQ   #10,D3          ;for 1/5 second
0000003E 93C9                0063          SUB.L   A1,A1           ;no flag address
00000040 4E41                0064          TRAP    #1
                             0065 ;
00000042 7009                0066          MOVEQ   #SD_EXTOP,D0    ;find where to put window
00000044 76FF                0067          MOVEQ   #-1,D3          ;wait until complete
00000046 204C                0068          MOVE.L  A4,A0           ;set channel
00000048 45FA0034            0069          LEA     GET_WIND(PC),A2
0000004C 4E43                0070          TRAP    #3
                             0071 ;
0000004E 7011                0072          MOVEQ   #SD_TAB,D0      ;reset to start of line
00000050 7200                0073          MOVEQ   #0,D1
00000052 4E43                0074          TRAP    #3
                             0075 ;
00000054 7013                0076          MOVEQ   #MT_RCLCK,D0    ;now read time into D1
00000056 4E41                0077          TRAP    #1
                             0078 ;
00000058 43FA0068            0079          LEA     BUF_TOP(PC),A1  ;use buffer from top down
0000005C 347800EC            0080          MOVE.W  CN_DATE,A2      ;to convert date into
00000060 4E92                0081          JSR     (A2)
                             0082 ;
00000062 7007                0083          MOVEQ   #IO_SSTRG,D0    ;now send the result
00000064 3419                0084          MOVE.W  (A1)+,D2        ;of 20 characters
00000066 76FF                0085          MOVEQ   #-1,D3          ;... with no timeout
00000068 204C                0086          MOVE.L  A4,A0           ;to our window
0000006A 4E43                0087          TRAP    #3
0000006C 6102                0088          BSR.S   OOPS            ;any errors?
0000006E 60C8                0089          BRA.S   10%
                             0090 ;
                             0091 ; Check for error on IO call
                             0092 ;
00000070 4A80                0093 OOPS:    TST.L   D0              ;has an error occurred
00000072 6708                0094          BEQ.S   OK              ;... no
00000074 2600                0095          MOVE.L  D0,D3           ;... yes - notify it
00000076 7005                0096          MOVEQ   #MT_FRJOB,D0    ;remove Job
00000078 72FF                0097          MOVEQ   #MYSELF,D1      ;... yes, this one
0000007A 4E41                0098          TRAP    #1              ;(we should not get back
                             0099                                  ; from this)
                             0100 ;
0000007C 4E75                0101 OK:      RTS
                             0102 ;
                             0103 ; This routine works out the window required to overlap
                             0104 ; window 0 at top RHS. This code forms part of a device
                             0105 ; driver and is in supervisor mode.
                             0106 ;
0000007E                     0107 GET_WIND:
0000007E 246E0078            0108          MOVE.L  SV_CHBAS(A6),A2
00000082 2452                0109          MOVE.L  (A2),A2         ;get origin X,Y
00000084 216A00180018        0110          MOVE.L  SD_XMIN(A2),SD_XMIN(A0)
0000008A 7014                0111          MOVEQ   #20,D0          ;20 characters
0000008C C0E80026            0112          MULU    SD_XINC(A0),D0  ;... of current width
00000090 3140001C            0113          MOVE.W  D0,SD_XSIZE(A0) ;set size X
00000094 317C000A001E        0114          MOVE.W  #10,SD_YSIZE(A0);... and Y
0000009A 906A001C            0115          SUB.W   SD_XSIZE(A2),D0 ;find X origin
0000009E 91680018            0116          SUB.W   D0,SD_XMIN(A0)  ;... from RHS
000000A2 7000                0117          MOVEQ   #0,D0           ;no error
000000A4 4E75                0118          RTS
                             0119 ;
000000A6 0003                0120 SCR:     DEFW    3               ;name of output device
000000A8 534352              0121          DEFB    'SCR'
                             0122          ALIGN
000000AC                     0123 BUFFER:
                             0124          DEFS    22              ;this is for CN_DATE
000000C2                     0125 BUF_TOP:
```

```
                                0126 ;
                                0127 END

      Symbols:

      000000AC BUFFER     000000C2 BUF_TOP    00000010 CLOCK      000000EC CN_DATE    0000007E GET_WIND
      00000001 IO_OPEN    00000007 IO_SSTRG   00000005 MT_FRJOB   0000000B MT_PRIOR   00000013 MT_RCLCK
      00000008 MT_SUSJB   FFFFFFFF MYSELF     0000007C OK         00000070 OOPS       000000A6 SCR
      00000009 SD_EXTOP   00000029 SD_SETIN   00000028 SD_SETST   00000011 SD_TAB     00000026 SD_XINC
      00000018 SD_XMIN    0000001C SD_XSIZE   0000001E SD_YSIZE   00000078 SV_CHBAS

0000 error(s)  detected
6179 bytes free
```