MUSIC LEX FILE

As Read From the HP 82161 Cassette Unit.

This is a LEX file from HP which implements a simple "organ" on the HP-75. When various keys are pressed, it sounds a tone corresponding with the note on an organ keyboard. It is essentially self contained, and works by waiting for a keypress, looking up the key in a table to get a frequency, and sounding a tone accordingly until the key is released.

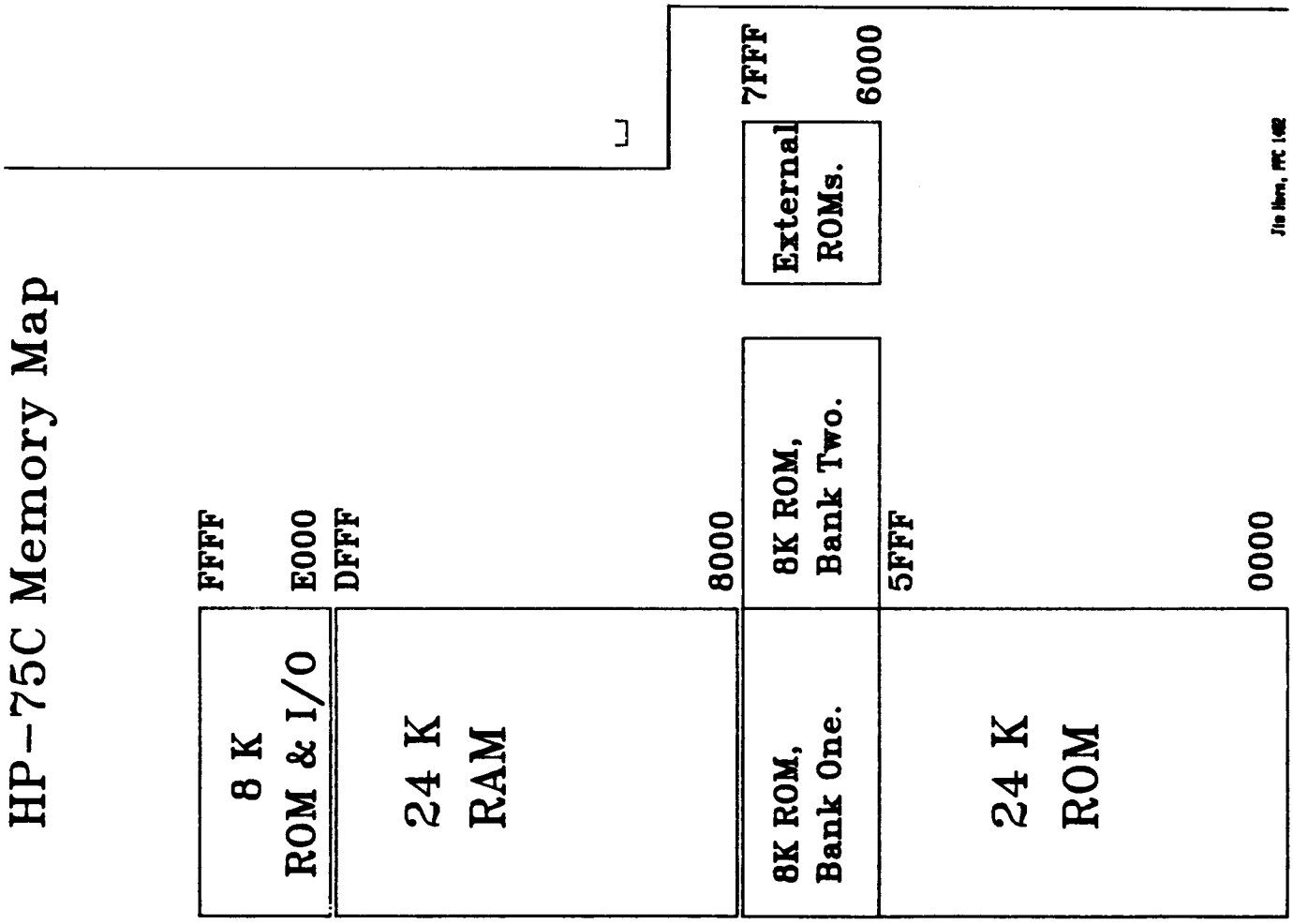
This program was disassembled by Jim Horn (1402). It is not necessarily correct, nor is it in full compliance with the HP-80 Series assemblers, although I used them as a guide. I do use Hexadecimal rather than Octal. Any and all corrections and additions will be appreciated.

The "addresses" shown on the left are the relative byte locations on the tape. Note that the information stored on the tape is the CAT data of 18 bytes, followed by the memory image of the program itself. The first two bytes of the CAT information point to where the program was in memory. Since the program assumes its beginning ("START") is at relative position zero, much of it has "-START" to set the addresses aright.

While the program is only 240 bytes long, when the CAT info is added, it totals 258 bytes, which takes up two records on the tape (all of the first, and the first two of the second). When using a tape, the unused bytes of the last record are the same as the bytes of the previous record, as the buffer's data there is unaffected.

* First, the CAT information...

CONTINUED ON NEXT PAGE



```
$000:  C9 9C              ! Location in RAM.
$002:  F0 00              ! Length of binary +240
$004:  8D                 ! 1000 1101 binary.
$005:  4C                 ! "L" = Lex file.
$006:  93 73 F3 9B        ! 16:15, 11/28/82
$00A:  4D 55 53 49        ! "MUSIC   "
$00E:  43 20 20 20

* Start of the data from memory:

* First is a set of pointers, the added BASIC
* commands, and any error numbers and messages
* which the LEX file adds to the system.

                   ROMPTR    EQU $82A3       ! Points to current code.
                   KEYBOARD  EQU $FF02       ! Keyboard input address.
                   SPEAKER   EQU $FF80       ! Speaker address.

$012:  09 80       START     HEX 09,80
$014:  0A 00                 DEF RUNTIME-START
$016:  12 00                 DEF NAMES-START
$018:  0C 00                 DEF PARSE-START
$01A:  18 00                 DEF ERRORS-START
$01C:  1B 00       RUNTIME   DEF RUN-START     ! Also defines INIT.
$01E:  2B 00       PARSE     DEF MAIN-START
$020:  1C 00                 DEF RUNAGAIN-START
$022:  FF FF                 HEX FF,FF         ! End of pointers.

$024:  4D 55 53    NAMES     ASP "MUSIC"       ! "MUSIC" command.
$027:  49 C3
$029:  FF                    HEX FF            ! End of commands.

$02A:  00          ERRORS    HEX 00            ! Error code (none).
$02B:  FF                    HEX FF            ! End of errors.

* The following is the code, starting with the setup routine.
* I make no promises of the accuracy of the decoding in this part.

$02C:  1F                    HEX 1F
$02D:  9E          RUN       RTN

$02E:  6C A9 B4 09 RUNAGAIN  LDMI R54,$09B4
$032:  80                    ELB R54
$033:  01 0A E5              PUMD R54,+R12
$036:  CE 6D 46              JSB $466D
$039:  A3                    STM R54,R12
$03A:  0C                    ARP R14
$03B:  9E                    RTN

* This is the main routine.

$03C:  A1                    HEX A1            ! Code attributes.

$03D:  98          MAIN      BIN
$03E:  52 92                 CLB R22
$040:  B2 02 FF              STBD R22,KEYBOARD
$043:  B1 A3 82              LDMD R22,ROMPTR   ! Get code address,
$046:  CB 9E 00              ADM R22,KEYMAP-START ! find table address.
$049:  66 B1 02 FF KEYLOOP   LDMD R46,KEYBOARD ! Check the keyboard.
$04D:  02 A3                 STM R46,R02       ! Save for lookup.
$04F:  42 CF 20 00           ANMS R02,$0020    ! Mask keypress bit?
$053:  F7 F4       K2        JZR KEYLOOP       ! No key - wait.
$055:  66 10 A3              STM R46,R20
$058:  CE A6 02              JSB $02A6
$05B:  51 C8 80              CMB R21,=$80      ! ATTN key?
$05E:  F7 3D                 JZR ATTNKEY       ! Quit if it is.
$060:  C8 7B                 CMB R21,=$7B      ! Key >"z"?
$062:  FB E5                 JCY KEYLOOP       ! Ignore if so.
$064:  CC 2A                 SBB R21,=$2A      ! Key <"*"?
$066:  F4 E1                 JPS KEYLOOP       ! If so, skip it.
$068:  14 A2                 STB R21,R24
$06A:  55 92                 CLB R25
$06C:  54 12 C3              ADM R24,R22
$06F:  56 B0 80              LDBD R26,=$80
$072:  FF 17                 JRN $08B
$074:  A2                    STB R26,R22
$075:  CA 01                 ADB R26,=$01
$077:  5A 14 A4              LDBD R32,R24
$07A:  F7 CD                 JZR KEYLOOP       ! Ignore silent keys.
$07C:  1B A2                 STB R26,R33
$07E:  5B CC 03              SBB R33,=$03
$081:  5C                    DRP R34
$082:  1A A1       TONELOOP  LDM R34,R32
$084:  8A          WAIT      DCB R34
$085:  F6 FD                 JNZ WAIT
$087:  56 B2 80 FF           STBD R26,SPEAKER
$08B:  5D                    DRP R35
$08C:  8A          WAIT2     DCB R35
$08D:  F6 FD                 JNZ WAIT2
$08F:  57 B2 80 FF           STBD R27,SPEAKER
$093:  5C B1 02 FF           LDMD R34,KEYBOARD
$097:  26 C1                 CMM R34,R46
$099:  F7 E7                 JZR TONELOOP
$09B:  F0 AC                 JMP KEYLOOP
$09D:  66 B1 02 FF ATTNKEY   LDMD R46,KEYBOARD ! Is ATTN still down?
$0A1:  CF 20 00              ANMS R46,$0020
$0A4:  F6 F7                 JNZ ATTNKEY       ! Wait 'til released.
$0A6:  92                    CLB R46           ! Send $00,$02 to
$0A7:  B2 02 FF              STBD R46,KEYBOARD ! the keyboard
$0AA:  A8 02                 LDB R46,=02       ! and end.
$0AC:  B2 02 FF              STBD R46,KEYBOARD
$0AF:  9E                    RTN

* KEYMAP:  The following table holds the periods for the
* different keys, stored one byte per key in ASCII order.  Note
* that unused keys such as the upper case letters have 0 for
* their values, to keep them silent.  To find the period value
* for a particular key, add the ASCII representation of the key
* to KEYTABLE-$2A.  The results are the address of the value.

$0B0:  15 00 3A 00 KEYMAP    BYT 21,0,58,0     ! * + , -
$0B4:  33 2D 18 00           BYT 51,45,24,0    ! . / 0 1
$0B8:  36 30 00 28           BYT 54,48,0,40    ! 2 3 4 5
$0BC:  24 20 00 1B           BYT 36,32,0,27    ! 6 7 8 9
$0C0:  00 30 00 00           BYT 0,48,0,0      ! : ; < =
$0C4:  00 00 00 00           BYT 0,0,0,0       ! > ? @ A
$0C8:  00 00 00 00           BYT 0,0,0,0       ! B C D E
$0CC:  00 00 00 00           BYT 0,0,0,0       ! F G H I
$0D0:  00 00 00 00           BYT 0,0,0,0       ! J K L M
$0D4:  00 00 00 00           BYT 0,0,0,0       ! N O P Q
$0D8:  00 00 00 00           BYT 0,0,0,0       ! R S T U
$0DC:  00 00 00 00           BYT 0,0,0,0       ! V W X Y
$0E0:  00 00 00 00           BYT 0,0,0,0       ! Z [ \ ]
$0E4:  00 00 00 00           BYT 0,0,0,0       ! ^ _ ` a
$0E8:  4D 5B 61 2D           BYT 77,91,97,45   ! b c d e
$0EC:  00 52 49 1C           BYT 0,82,73,24    ! f g h i
$0F0:  41 00 36 3D           BYT 65,0,54,61    ! j k l m
$0F4:  45 19 16 3A           BYT 69,25,22,58   ! n o p q
$0F8:  2B 6E 26 1E           BYT 43,110,38,30  ! r s t u
$0FC:  57 33 67 22           BYT 87,51,103,34  ! v w x y
$100:  74                    BYT 116           ! z

$101:  AE                    HEX AE            ! What's this?

! Jim Horn, PPC 1402
```

END LINE