Chapter 3

HMSL Operation

Abstract

This chapter describes the operation of HMSL including its menus, various *operating modes* and the *main control words for starting and stopping* the HMSL scheduler. Some of this material (like that on HMSL.SCAN) may be more technical than most people will need, so you may just want to skim this. However, the information on HMSL's operating modes, and in particular, the words **HMSL.START, HMSL.PLAY**, and **HMSL** is important for beginning users.

HMSL Operating Modes

HMSL has three operating modes: *Forth, HMSL* and *Forth+HMSL* mode.

Forth Mode

When you first run HMSL, you are in *Forth mode*. You can *enter Forth commands* to examine or manipulate data structures, compile programs, send MIDI commands, etc. If you enter the command:

```
HMSL    ( or )
some-morph  HMSL.PLAY
```

then you will be in...

HMSL Mode

In this mode the ASCII keyboard will not respond, the HMSL scheduler is running without looking for keystrokes (except a selected few). You can hit 'Q' to get back to Forth Mode, or if you load a KEY-PARSER (see end of this chapter) you can trigger your own commands using single keystrokes.

In HMSL mode, the HMSL window is open. This allows use of the Shape Editor or other screens. Screen can be selected from the "Screens" menu.

When you enter this mode, the **HMSL scheduler start running.** This scheduler, known affectionately as the *polymorphous executive*, allows several jobs or players to run simultaneously. This means you can play several melodies in parallel with other activities like algorithmic composition. You might think of this as a "play mode". If a player is started in this mode it will play to completion. If you start a player in Forth Mode, you will hear nothing. This is because the player is ready to play but it needs the polymorphous executive to keep it active.

[More advanced technical description]: The polymorphous executive, or *PE*, continuously calls a routine called HMSL.SCAN which "keeps alive" several activities. HMSL.SCAN does the following:

1) Sends a TASK: message to all *active* jobs and players. If it is time to do something, for example play a note, then that activity will occur.

2) Executes any events posted using POST.EVENT that are ready to occur.

3) Call SELF.CLOCK, a deferred word used internally by USE.SELF.TIMER. See the chapter on Time and Scheduling.

4) Call MIDI.PARSE.MANY to see if any MIDI Input needs parsing.

5) Call HMSL.CHECK.EVENT to see if any mouse events have occured. This keeps the HMSL screens like the Shape Editor, alive.

6) Check the ASCII keyboard using ?TERMINAL. If a key has been pressed, it is passed to a KEY-PARSER if there is one. See below.

You can get back into Forth Mode in four ways:

1) Click on the close box in the HMSL window.

2) Select "Stop" from the HMSL menu.

3) Hit 'Q' on the ASCII keyboard. (Amiga users will need to click in the JForth window first.)

4) Set the variable QUIT-HMSL to true. This could be done by a job or a control grid. It is how the items 1,2 & 3 stop HMSL.

Forth+HMSL Mode

This is a combination of Forth and HMSL modes. In this mode, the HMSL scheduler is running. Jobs and players can be started and run. The ASCII keyboard is *also available* for entering Forth commands. You may wonder why we don't use this mode all the time. When HMSL and Forth are both active, they both run a little slower. Typing in this mode is almost painfully slow because the HMSL scheduler is taking up so much of the computers time.

You can enter this mode by entering:

```
HMSL.START
```

You can get back to Forth Mode by entering:

```
HMSL.STOP
```

or by using one of methods 1,2 or 4 from above. In some of the tutorials we use this mode because you can enter commands that affect morphs while they are running. This is handy for experimenting or debugging. It may also be handy for performance situations where you want compete control over the piece as it runs, or want to type in new contents of variables, start things running from the ASCII keyboard, and so on.

In this mode a morph can be started and run simply by sending it a **START:** message. For example, to start a player:

```
OB.PLAYER  PL1 ( instantiate a player )
PREFAB:  PL1  ( setup a simple melody )
1000 PUT.REPEAT:  PL1  ( make it play a long time )
HMSL.START (start HMSL+FORTH mode )
START:  PL1 ( start the player running )
PRINT:  PL1
STOP:  PL1 ( stop the player )
HMSL.STOP
```

You may need to resize or move windows to be able to type Forth commands.

HMSL Menus

When you are in HMSL Mode or Forth+HMSL Mode, you will see two menus appear. The Macintosh version will also have other menus visible. The two new menus are "HMSL" and "Screens".

HMSL Menu

At the moment, this only has two commands in it. If you have some ideas of other commands you would like to see in here, let us know.

Reset - Calls HMSL.RESET.

This will stop anything playing. Clears lots of things in HMSL. See HMSL.RESET in the HMSL Controls reference section below.

Stop - Stops HMSL

Selecting this item will put you back in Forth mode. See HMSL.STOP in the HMSL Controls reference section below.

Screens Menu

This menu selects an *interactive control screen*. There are two predefined screens which are described here. If you design your own screen, its name will appear in the menu. See the chapter on Interactive Controls and Screens.

Shape Editor - Displays the Shape Editor

This will display the Shape Editor which allows you to graphically edit shapes. See the chapter on Shapes for more detail.

Action Table - Displays the Action Table

This will display the Action Table which allows you to control Actions. Actions are objects that have custom stimuli and responses. They can be used for interactive real time performance. See the chapter on Perform for more information.

HMSL Control Reference

Some of these words are fairly technical. If you are not already familiar with how to use HMSL, you may want to try some of the tutorials first. The most important words here are: HMSL , HMSL.START , HMSL.STOP , and HMSL.PLAY

BYE (-- , leave HMSL and Forth entirely)

Calls HMSL.TERM to clean everything up. Completely exits HMSL.

HMSL (-- , start HMSL running)

When HMSL is running, the HMSL window appears and the HMSL screens are activated. The polymorphous executive is started which provides the scheduling operations. The word HMSL is called when a piece can be started from a screen, eg. actions, or is driven by MIDI input using the MIDI parser.

When you start HMSL using this word, the KEY-PARSER will be active. See below.

HMSL.CLOSE (-- , close graphics window)

You may want to manually close the HMSL window if your code crashes HMSL and it doesn't manage to close it by itself.

HMSL-GRAPHICS (-- var-addr , use window when true)

When HMSL is started, the window will only open if HMSL-GRAPHICS is TRUE. This is the default condition. If you do not want graphics, set this variable to FALSE. This can be useful if you are running a piece that doesn't use any on-screen graphics.

```
FALSE  HMSL-GRAPHICS !    ( or )
HMSL-GRAPHICS  OFF
```

HMSL.EDIT.PLAY (shape -- , edit a shape while playing)

Adds the shape to the SHAPE-HOLDER then plays it using a stock player called SE-PLAYER. SE-PLAYER is executed by calling HMSL.PLAY .

HMSL.EXEC (morph -- , executes morph)

This is like HMSL.PLAY but it quits automatically when there are no more morphs playing.

HMSL.INIT (-- initialize HMSL system)

This does all the initialization required to run HMSL. It starts the real time clock running, sets up MIDI, and allocates memory for various things like the MIDI-ALLOCATOR. It is called automatically when you start HMSL if you answer 'Y' to the startup question. You cannot SAVE-FORTH after having done HMSL.INIT because HMSL will come up next time thinking it is already initialized but it won't really be initialized. HMSL.INIT initializes all of the HMSL subsystems by calling SYS.INIT. SYS.INIT is redefined by every system that needs to be initialized. By having SYS.INIT call SYS.INIT inside itself, and then adding one's own customizations, a chain of initializations can be performed. If the variable IF-DEBUG is TRUE, a diagnostic message will print as HMSL initializes each section. Here is how a system might define SYS.INIT.

```
:  SYS.INIT  ( -- ) sys.init  mything.init ;
```

HMSL.OPEN (-- , open HMSL graphics window)

This is handy if you want to test graphics commands, or test drawing Control Grids or Screens.

HMSL.PLAY (morph -- , execute the morph then call HMSL)

A START: message is sent to the morph then HMSL is called, allowing the hierarchy to be executed in time. If a piece that involves a hierarchy of morphs, then typically the top morph in the hierarchy would be passed to HMSL.PLAY.

```
MAIN-COLLECTION  HMSL.PLAY
```

HMSL.RESET (-- , resets most of HMSL)

If things get "messed up" for some reason, you can sometimes fix things by calling HMSL.RESET. It calls SYS.RESET then calls the USER.RESET chain that allows you to have user applications reset. Among other things, this will clear the action table, clear ACTOBJ and abort any active players or jobs, clear AMIGA-ALLOCATOR , clear MIDI-ALLOCATOR , clear SHAPE-HOLDER, silence AMIGA local sound.

HMSL.SCAN (-- , INTERNAL , keep HMSL alive)

This is the primary word for the polymorphous executive. Users normally do not call this directly. It will TASK: any active player or job, check for MIDI Parser events and graphic events. This is called continuously in a loop to keep the HMSL scheduler alive.

HMSL.START (-- , start HMSL running w/ keyboard active)

This opens the HMSL window and starts the polymorphous executive just like calling the HMSL word. This word, however, leaves the ASCII keyboard active. You can then START: or PRINT: morphs while a piece is running. This is very useful for debugging.

HMSL.START keeps the keyboard active by redefining KEY and EMIT. They will call HMSL.SCAN as well as perform their normal tasks of getting keyboard input and outputting to the screen. The ASCII keyboard will seem sluggish (respond slowly) because the music is given a much higher priority.

HMSL.STOP (-- , stops HMSL)

Opposite of HMSL.START

HMSL.TERM (-- , clean up HMSL system)

This is called automatically when you enter BYE or QUIT HMSL from the menu.

Controlling HMSL using ASCII Keystrokes

If you enter HMSL, HMSL runs but the keyboard is not active. You cannot enter Forth commands. There is, however, a way to use the keyboard to parse ASCII keystrokes. If you put the CFA of a function in the variable KEY-PARSER, then HMSL passes any keys hit to that parser. The characters are converted to UPPER CASE before being passed. Here is an example of using the parser to start and stop a player from the keyboard.

```
OB.PLAYER PL1
PREFAB: PL1  ( quick set up of player, shape, instrument )
: PARSE.KEY  ( char  -- )
   CASE
      ASCII B OF  START:  PL1  ENDOF
      ASCII E OF STOP: PL1 ENDOF
   ENDCASE
;
'C  PARSE.KEY  KEY-PARSER  !
      ( the use of ‘c is explained in the Forth glossary )
HMSL
```

Now hit 'B' to **b**egin playing the player and 'E' to **e**nd the playing.

!!Warning - Don't forget to set KEY-PARSER to zero when you FORGET PARSE.KEY!!

KEY-PARSER (-- var-addr , contains CFA of keyboard parser)

your.parser (upper-case-char -- , do your thing)