

Copyright (c) 2026 Zayn Otley. All rights reserved.

Chapter 19 - MOD Playback

MOD playback is the sample-based music engine. A MOD carries its sample data, pattern rows, song order, and effects inside one block of memory. You point the player at that block, give its length, choose an output filter if wanted, then start it.

Unlike PSG, SID, TED, POKEY, and the other chip chapters, MOD is not a register-by-register synthesiser. Its register block controls a player. The musical notes live inside the module data.

19.1 First sound

Type this program. It builds a tiny four-voice module in memory, starts the A500-style output filter, and plays a soft chord. No file or assembler is needed.

```

10 REM TINY FOUR VOICE MOD
20 POKE32 &H00F0800,1
30 A=&H00100000:L=2236
40 REM CLEAR HEADER, PATTERN, AND SAMPLE AREA
50 FOR I=0 TO L-1
60 POKE8 A+I,0
70 NEXT I
80 REM SONG NAME
90 T$="IE MOD CHORD"
100 FOR I=1 TO LEN(T$)
110 POKE8 A+I-1,ASC(MID$(T$,I,1))
120 NEXT I
130 REM FOUR 32 BYTE LOOPED SAMPLES
140 FOR S=0 TO 3
150 D=20+S*30
160 POKE8 A+D+22,0
170 POKE8 A+D+23,16
180 POKE8 A+D+25,48
190 POKE8 A+D+29,16
200 NEXT S
210 REM SONG LENGTH AND M.K. FORMAT ID
220 POKE8 A+950,1
230 POKE8 A+1080,77
240 POKE8 A+1081,46
250 POKE8 A+1082,75
260 POKE8 A+1083,46
270 REM FIRST ROW: C-2, E-2, G-2, C-3
280 POKE8 A+1084,1
290 POKE8 A+1085,172
300 POKE8 A+1086,16
310 POKE8 A+1088,1
320 POKE8 A+1089,83
330 POKE8 A+1090,32
340 POKE8 A+1092,1
350 POKE8 A+1093,29
360 POKE8 A+1094,48
370 POKE8 A+1096,0
380 POKE8 A+1097,214
390 POKE8 A+1098,64
400 REM FOUR SHORT WAVEFORMS
410 FOR S=0 TO 3
420 FOR I=0 TO 31
430 V=INT(SIN(I*TWOPI*(S+1)/32)*90)
440 IF V<0 THEN V=V+256
450 POKE8 A+2108+S*32+I,V
460 NEXT I
470 NEXT S
480 REM FILTER, START, THEN CHECK STATUS
490 SOUND MOD FILTER 1
500 SOUND MOD PLAY A,L
510 FOR T=1 TO 3000
520 NEXT T
530 PRINT MOD STATUS

```

You should hear a short four-voice chord. Line 530 normally prints 1 while the player is still running, or 0 if the short module has already ended.

The module built above has a standard 1084-byte header, one 1024-byte pattern, and four 32-byte looped samples. The first pattern row encodes four notes: C-2, E-2, G-2, and C-3.

Lines 40 to 70 clear the whole block, because zero is a useful default throughout the MOD header. Lines 130 to 200 describe four small samples: length 16 words, volume 48, and loop length 16 words. Lines 270 to 390 write only the non-zero bytes of the first pattern row; the cleared bytes remain as effect zero and parameter zero. Lines 400 to 470 create the sample data, and lines 490 to 530 select the filter, start playback, and read status.

19.2 What MOD can play

Item	Value
Usual channel count	4
Accepted channel IDs	M.K., 4CHN, FLT4, M!K!, 6CHN, 8CHN, FLT8, OCTA, CD81, OKTA, and nnCH up to 32 channels
Pattern rows	64 rows per pattern
Sample format	Signed 8-bit sample bytes inside the module
Sample slots	31 standard sample descriptors
Default speed	6 ticks per row
Default tempo	125 BPM
Output filters	0 none, 1 A500, 2 A1200

Four-channel modules play directly. Wider modules are accepted and mixed down through the four MOD output paths.

19.3 Register block

The player register block is at \$F0BC0-\$F0BD7.

Address	Name	Access	Purpose
\$F0BC0	MOD_PLAY_PTR	write/read	Start address of the module data in memory.
\$F0BC4	MOD_PLAY_LEN	write/read	Length of the module data, in bytes.
\$F0BC8	MOD_PLAY_CTRL	write/read	Control and busy state.
\$F0BCC	MOD_PLAY_STATUS	read	Playback status and error state.
\$F0BD0	MOD_FILTER_MODEL	write/read	Output filter model.
\$F0BD4	MOD_POSITION	read	Current song-position index.

MOD_PLAY_CTRL accepts these bits when written:

Bit	Value	Meaning
0	1	Start using the staged pointer and length.
1	2	Stop playback and clear the player error flag.
2	4	Loop the song when it reaches its restart point or end.

When read back, MOD_PLAY_CTRL uses bit 0 as the load-busy flag and bit 2 as the current loop flag. Starting a module copies the bytes from memory and parses them before playback begins, so the busy bit can briefly be set before

MOD_PLAY_STATUS reports playing.

MOD_PLAY_STATUS uses these bits:

Bit	Meaning
0	The player is producing MOD audio.
1	The last start request failed.

19.4 BASIC commands

Use these forms from BASIC:

```
SOUND MOD PLAY address,length
SOUND MOD STOP
SOUND MOD FILTER model
MOD STATUS
```

Always give a length with SOUND MOD PLAY unless you have deliberately staged MOD_PLAY_LEN yourself. MOD STATUS returns the same status byte as \$F0BCC, so the normal tests are:

```
10 S=MOD STATUS
20 IF S AND 2 THEN PRINT "MOD ERROR":END
30 IF S AND 1 THEN PRINT "MOD PLAYING"
```

To stop the current module:

```
10 SOUND MOD STOP
20 PRINT MOD STATUS
```

19.5 Data format

A MOD block begins with a 1084-byte header.

Offset	Length	Meaning
0	20	Song name, padded with zero bytes.
20	930	Thirty-one sample descriptors, 30 bytes each.
950	1	Song length, 1-128 positions.
951	1	Restart position.
952	128	Pattern table.
1080	4	Format ID.
1084	varies	Pattern data, then sample data.

Sample descriptor length, loop start, and loop length are stored as big-endian word counts in the module, not byte counts. Multiply them by two to get bytes. Sample bytes are signed: \$00 is silence, \$7F is a large positive sample, and \$80 is a large negative sample.

Each pattern row stores four bytes per channel:

Byte	Bits	Meaning
0	high nibble	Sample number high bits.
0	low nibble	Period high bits.
1	all bits	Period low byte.
2	high nibble	Sample number low bits.
2	low nibble	Effect number.
3	all bits	Effect parameter.

For example, sample 1, period 428, effect 0, parameter 0 is:

```
01 AC 10 00
```

That is the first note in the program at the start of this chapter.

19.6 Filters

SOUND MOD FILTER 0 disables the model filter. 1 selects the A500 analogue-output curve at about 4.5 kHz. 2 selects the A1200 curve at about 28 kHz.

```
10 REM MOD FILTER READBACK
20 SOUND MOD FILTER 1
30 PRINT PEEK32(&H000F0BD0)
40 SOUND MOD FILTER 2
50 PRINT PEEK32(&H000F0BD0)
```

Use only models 0, 1, and 2. Other values are reserved.

The two PRINT lines read the filter-model register after each command. This is a simple way to check which output curve the MOD player will use.

Some modules use the ProTracker LED-filter effect. The model filter must be non-zero before the LED-filter stage can affect the sound.

19.7 Raw register start

The BASIC command is easiest, but raw POKE32 gives access to the loop bit. This example assumes the module bytes are already at \$100000 and the module length is 2236 bytes, as built by the first program.

```
10 REM MOD RAW LOOP START
20 POKE32 &H000F0800,1
30 REM POINTER, LENGTH, FILTER
40 POKE32 &H000F0BC0,&H00100000
50 POKE32 &H000F0BC4,2236
60 POKE32 &H000F0BD0,1
70 REM START WITH LOOP BIT SET
80 POKE32 &H000F0BC8,5
90 PRINT PEEK32(&H000F0BCC)
```

Line 80 writes start plus loop. Line 90 prints a status byte with bit 0 set once playback is active.

Use the raw form when you need the loop bit. The BASIC SOUND MOD PLAY command starts once; the raw control value 5 is start bit 0 plus loop bit 2.

19.8 Errors and Limits

The error bit is set when a start request has no readable memory behind it, uses a zero length, names an unknown format ID, is too short for the header or pattern data, or otherwise cannot be parsed as a supported MOD.

Stopping the player clears the error bit. Starting a new module clears the old error first, then sets it again only if the new start fails.

The player owns the sound paths while a MOD is playing. If another engine writes to the same flexible DAC channels at the same time, the latest writer wins for that sample period. Stop MOD playback before using those DAC channels for your own sample output.