

Chapter 20 - The WAV Sample Player

The WAV player plays RIFF/WAVE PCM data from memory. It is the direct choice for sampled effects, speech, short stings, and already recorded music. The program supplies a complete WAV block, including header and sample bytes; the player parses it, resamples it, and writes the result to two adjacent SoundChip DAC channels.

Use MIDI/MUS playback in Chapter 21 when the source is note music rather than sampled audio. Use Paula DMA in Chapter 22 when you want lower-level timing control. Use the SFX sample trigger in Chapter 12 when you already have raw PCM bytes and do not want a WAV header.

20.1 First sound

Type this program. It builds a tiny 8000 Hz, 16-bit mono WAV in memory and loops it through the WAV player.

```
10 REM TINY WAV BEEP
20 POKE32 &H000F0800,1
30 A=&H00110000:N=800:L=44+N*2
40 REM CLEAR THE WAV BLOCK
50 FOR I=0 TO L-1
60 POKE8 A+I,0
70 NEXT I
80 REM COPY THE 44 BYTE RIFF HEADER
90 FOR I=0 TO 43
100 READ B
110 POKE8 A+I,B
120 NEXT I
130 REM 800 SAMPLES OF 440 HZ SINE
140 FOR I=0 TO N-1
150 V=INT(SIN(I*TWOPI*440/8000)*22000)
160 IF V<0 THEN V=V+65536
170 L0=V-INT(V/256)*256
180 HI=INT(V/256)
190 POKE8 A+44+I*2,L0
200 POKE8 A+45+I*2,HI
210 NEXT I
220 REM OUTPUT PAIR, VOLUME, FORCE MONO
230 POKE8 &H000F0BF0,0
240 POKE8 &H000F0BF1,220
250 POKE8 &H000F0BF2,220
260 POKE8 &H000F0BF3,1
270 REM POINTER, LENGTH, START PLUS LOOP
280 POKE32 &H000F0BD8,A
290 POKE32 &H000F0BDC,L
300 POKE32 &H000F0BE0,5
310 FOR T=1 TO 3000
320 NEXT T
330 PRINT PEEK32(&H000F0BE4)
350 DATA 82,73,70,70,100,6,0,0,87,65,86,69
360 DATA 102,109,116,32,16,0,0,0,1,0,1,0
370 DATA 64,31,0,0,128,62,0,0,2,0,16,0
380 DATA 100,97,116,97,64,6,0,0
```

You should hear a looping tone. Line 330 prints a status byte. Bit 0 is usually set because the sound is playing.

Lines 40 to 70 clear the memory block, then lines 80 to 120 copy the header from the DATA statements. Lines 130 to 210 write signed 16-bit samples in little-endian order. Lines 220 to 300 choose DAC channels, set both volumes, force mono output, stage the pointer and length, and start with looping enabled.

20.2 What the WAV player accepts

Item	Value
Container	RIFF/WAVE
Audio format	PCM
Extensible format	16-bit PCM only
Channels	Mono or stereo
8-bit samples	Unsigned PCM, converted to signed output
16-bit samples	Signed little-endian PCM
Sample rate	Any non-zero rate declared in the header
Output	Two adjacent SoundChip DAC channels
Default output	Channel pair 0 and 1, volume 255, force mono on

The parser accepts extra RIFF chunks and skips them. It rejects floating point audio, more than two channels, unsupported bit depths, zero sample frames, bad byte-rate or block-align fields, truncated chunks, and missing fmt or data chunks.

20.3 Register block

The player register block is at \$F0BD8-\$F0BF3.

Address	Name	Access	Purpose
\$F0BD8	WAV_PLAY_PTR	write/read	Low 32 bits of the WAV block address.
\$F0BDC	WAV_PLAY_LEN	write/read	Length of the WAV block, in bytes.
\$F0BE0	WAV_PLAY_CTRL	write/read	Start, stop, loop, pause, and resume control.
\$F0BE4	WAV_PLAY_STATUS	read	Busy, error, pause, and stereo-output status.
\$F0BE8	WAV_POSITION	read	Current source-frame position.
\$F0BEC	WAV_PLAY_PTR_HI	write/read	High 32 bits of the WAV block address.
\$F0BF0	WAV_CHANNEL_BASE	write/read	Left DAC channel; the right DAC uses base plus one.
\$F0BF1	WAV_VOLUME_L	write/read	Left volume, 0-255.
\$F0BF2	WAV_VOLUME_R	write/read	Right volume, 0-255.
\$F0BF3	WAV_FLAGS	write/read	Bit 0 forces mono output.

Use channel bases 0-8. The player needs two adjacent DAC channels.

20.4 Control and status bits

Write these values to WAV_PLAY_CTRL:

Bit	Value	Meaning
0	1	Start playback using the staged pointer and length.
1	2	Stop playback.
2	4	Loop when the end is reached. Combine with start as value 5.
3	8	Pause without resetting the source position.
4	16	Apply the loop bit to the current playback without restarting.

To resume after pause, write 0 to WAV_PLAY_CTRL. Writing 1 starts again from the staged pointer and length.

Read WAV_PLAY_CTRL for control state:

Bit	Meaning
0	Load is busy or playback is active.
2	Loop is enabled.
3	Pause is enabled.

Read WAV_PLAY_STATUS for player state:

Bit	Meaning
0	Busy or playing.
1	Last start request failed.
2	Paused.
3	Stereo output path is active.

The error bit is cleared by a successful new start. Stop does not clear an old error; start a valid WAV or reset the player state to clear it.

20.5 Setup order

From a clean state:

1. Enable the audio mixer by writing 1 to \$F0800.
2. Put the WAV bytes in memory.
3. Write WAV_PLAY_PTR and WAV_PLAY_LEN.
4. Write WAV_PLAY_PTR_HI if the block address needs high bits.
5. Choose WAV_CHANNEL_BASE, WAV_VOLUME_L, WAV_VOLUME_R, and WAV_FLAGS.
6. Write 1 to WAV_PLAY_CTRL, or 5 for start plus loop.

The player copies the supplied bytes when start is written. Changing the source memory after that does not change the sound that is already playing. Change the registers and start again if you want a different sample.

20.6 Header and sample bytes

RIFF/WAVE stores multi-byte fields little-endian. In the first program, line 350 starts with RIFF, then a file size of 1636 bytes after the first eight bytes. Line 360 writes the fmt chunk for one-channel 16-bit PCM. Line 370 writes the sample rate 8000, byte rate 16000, block align 2, and bit depth 16. Line 380 writes the data chunk size, 1600 bytes.

For 16-bit samples, write low byte first:

```
10 V=22000
20 POKE8 A+44,V-INT(V/256)*256
30 POKE8 A+45,INT(V/256)
```

For 8-bit PCM WAV data, the file byte is unsigned. \$80 is silence, \$FF is positive, and \$00 is negative.

20.7 Volume, stereo, and channels

This example changes the output pair and makes the right side quieter:

```
10 REM WAV OUTPUT SETTINGS
20 POKE8 &H000F0BF0,2
30 POKE8 &H000F0BF1,255
40 POKE8 &H000F0BF2,80
50 POKE8 &H000F0BF3,0
```

Line 20 selects DAC channels 2 and 3. Line 50 clears force mono, so a stereo WAV keeps its left and right streams. A mono WAV still feeds both DAC channels with the same sample.

Changing the channel base while a sound is playing releases the old DAC pair and moves playback to the new pair. Any other sound engine writing to the same DAC channels will compete with the WAV player, so choose a pair that your program is not using for manual DAC output.

20.8 Pause, resume, loop, and stop

Pause:

```
10 REM WAV PAUSE
20 POKE32 &H000F0BE0,8
30 PRINT PEEK32(&H000F0BE4)
```

Resume without retriggering:

```
10 REM WAV RESUME
20 POKE32 &H000F0BE0,0
```

Change the loop flag while the current sound continues:

```
10 REM WAV LOOP ON WITHOUT RESTART
20 POKE32 &H000F0BE0,20
```

20 is loop plus apply-only. To turn looping off without restarting, write 16.

Stop:

```
10 REM WAV STOP
20 POKE32 &H000F0BE0,2
```

Stop releases the DAC channels used by the player.

20.9 Errors and Limits

If bit 1 of WAV_PLAY_STATUS is set, check these first:

- WAV_PLAY_LEN must be non-zero.
- The memory range must be readable.
- The block must begin with RIFF and contain WAVE.
- The fmt and data chunks must both be present.
- The file must be PCM, mono or stereo, 8-bit or 16-bit.
- The byte rate and block align must match the sample rate, channel count, and bit depth.

WAV_POSITION reports a source-frame index. It advances according to the file's sample rate, not the number of output samples written to the mixer. When looping is enabled, the reported position wraps inside the source length.