

Chapter 11 - Audio Architecture Overview

The picture side has several chips feeding one compositor. The audio side has several engines feeding one stereo mixer. Each engine owns its own register block on the shared bus. Several classic chip engines translate their direct tone registers through the SoundChip flexible voice path, so their register blocks are independent even when the generated voices share mixer channels.

The mixer runs at 44100 samples per second. It sums active engines, applies the global overdrive, filter, and reverb stages, then sends the result to the audio output.

11.1 The engine map

Engine	Chapter	Main use
SoundChip and SFX	12	Ten IE-native synth channels and 32 raw sample SFX channels
PSG	13	AY-style square/noise tones and envelopes
SN76489	14	Four-channel tone/noise latch chip
SID family	15	Three-voice wavetable/filter-style chip sound
TED audio	16	Two square/noise voices
POKEY	17	Four Atari-style tone/noise channels
AHX	18	AHX song playback
MOD	19	Four-channel tracker playback
WAV	20	PCM sample playback
MIDI/MUS and live MIDI	21	SME, MUS, and live MIDI events through RawlandMini
Paula DMA	22	Four DMA sample channels

Chapter 23 gives BASIC music recipes that combine these engines. The CPU chapters use different engines in their hand-entered machine-code examples so that the byte-entry programs are audibly different.

11.2 Which engine to choose

The IE-native SoundChip is the general synthesiser and effects core. The other engines are separate sound cards on the same mixer: some are best for period-chip tone colour, some for tracker songs, and some for sample playback.

Engine	Characteristic features	Use it when
SoundChip	Ten flexible channels; sine, triangle, saw, square, pulse, noise, ring modulation, hard sync, PWM, ADSR envelopes, sweeps, DAC writes, per-channel filter, global overdrive, filter, and reverb	You want IE-native synthesis, layered effects, or a shared flexible voice path for richer sounds
SFX	Thirty-two raw sample trigger channels with pointer, length, loop, rate, volume, and sample-format control	You have short PCM effects and want trigger-and-forget playback
PSG	Three square-tone channels, one noise source, and AY-style envelopes	You want crisp AY/YM arcade or microcomputer tones

Engine	Characteristic features	Use it when
SN76489	Three tone channels and one noise channel through a latch/data port	You want simple console-style square waves and noise
SID family	Three SID voices per chip, pulse width, ADSR, sync/ring options, and resonant filter behaviour	You want C64-style lead, bass, pulse, and filter movement
TED audio	Two compact square/noise voices	You want the small TED sound beside TED video
POKEY	Four Atari-style tone/noise channels and AUDCTL pairing modes	You want metallic, buzzy, or percussive Atari colour
AHX	Four synthetic music voices from AHX or THX module data	You want a chip-tune song format without sample data
MOD	Four-channel sample tracker playback with pattern and effect data	You want Amiga-style tracker music from one memory block
WAV	RIFF/WAVE PCM parsing, resampling, and stereo or mono output	You want recorded audio, speech, stings, or test tones
MIDI/MUS and live MIDI	SMF type 0/1 and MUS parsing, live note/programme/controller events, 16 MIDI channels, 10 active RawlandMini voices, GM-style programme families, drum channel, tempo, volume, loop, pause, and pitch bend	You want file-based melodic music or immediate note events with a built-in GM-style patch table
Paula DMA	Four low-level signed 8-bit sample DMA channels with period and completion bits	You want exact sample-buffer control or Amiga-style DMA timing

11.3 Master control

The master audio control register is AUDIO_CTRL at \$F0800.

Bit	Meaning
0	Enable audio output.
1	Freeze mixer state while changing presets.

Turn audio on before starting raw register examples:

```
10 POKE32 &H000F0800,1
```

The engines still accept register writes while audio is disabled; they become audible when bit 0 is set.

11.4 A first audio setup

This typed program starts the mixer, writes PSG and POKEY tone registers, then finishes with two SoundChip voices on channels 2 and 3. It puts a light low-pass filter and reverb over the resulting output:

```

10 REM FIRST AUDIO SETUP
20 POKE32 &H00F0800,1
30 PSG 0,142,12
40 POKEY 1,96,&HA8
50 SOUND 2,262,180,1,128
60 SOUND 3,330,140,2
70 ENVELOPE 2,4,8,200,12
80 ENVELOPE 3,4,8,200,12
90 GATE 2, ON
100 GATE 3, ON
110 SOUND FILTER 190,80,1
120 SOUND REVERB 90,120

```

Expected result: the SoundChip, PSG, and POKEY register blocks contain the shown values, and the filter and reverb affect the active mixed output. PSG and POKEY direct tones use the shared flexible voice path, so they can replace earlier flexible voice state rather than adding an unlimited separate set of voices. For that reason, this first tour writes PSG and POKEY before the final SoundChip chord.

11.5 SoundChip channel block

The IE-native SoundChip uses a flexible-channel layout. Each channel is \$40 bytes wide.

Offset	Field	Meaning
\$00	FREQ	Frequency in 16.8 fixed-point Hz.
\$04	VOL	Volume, 0 to 255.
\$08	CTRL	Gate and control bits.
\$0C	DUTY	PWM duty cycle.
\$10	SWEEP	Pitch sweep.
\$14	ATK	Attack.
\$18	DEC	Decay.
\$1C	SUS	Sustain.
\$20	REL	Release.
\$24	WAVE_TYPE	Waveform select.
\$28	PWM_CTRL	PWM rate and depth.
\$2C	NOISEMODE	Noise algorithm.
\$30	PHASE	Write to reset phase.
\$34	RINGMOD	Ring-modulation source.
\$38	SYNC	Hard-sync source.
\$3C	DAC	Signed 8-bit sample value.

The channel blocks are:

Channels	Base	End
0 to 3	\$F0A80	\$F0B7F

Channels	Base	End
4 to 6	\$F0C40	\$F0CFF
7 to 9	\$F0D40	\$F0DFF

SOUND, ENVELOPE, and GATE drive channels 0 to 3. The higher channels are direct-register channels used by expanded chip setups described later.

The SOUND `ch, freq, vol[, wave[, duty]]` keyword writes:

Argument	Register effect
ch	Selects channel 0 to 3.
freq	Stored as <code>freq * 256</code> in <code>FREQ</code> .
vol	Stored in <code>VOL</code> .
wave	Optional <code>WAVE_TYPE</code> .
duty	Optional <code>DUTY</code> .

11.6 SFX sample channels

The SFX block is for short raw samples stored in memory. The extended window has 32 channels at \$F2600 to \$F29FF, with stride \$20. The older \$F0E80 to \$F0EFF window remains as legacy aliases for channels 0 to 3.

Offset	Field	Meaning
\$00	SFX_PTR	Sample address.
\$04	SFX_LEN	Sample length in bytes.
\$08	SFX_LOOP_PTR	Loop start address.
\$0C	SFX_LOOP_LEN	Loop length.
\$10	SFX_FREQ	Playback rate in Hz.
\$14	SFX_VOL	Volume field. Values 0 to 255 set the audible level; larger values are clipped to 255.
\$16	SFX_PAN_RESERVED	Reserved pan field.
\$18	SFX_FORMAT	0 signed 8-bit, 1 unsigned 8-bit, 2 signed 16-bit.
\$1C	SFX_CTRL	Bit 0 trigger, bit 1 stop, bit 2 loop.

Status bits are exposed through `SFX_CTRL`: bit 0 means playing and bit 1 means error. On the 6502 and Z80, select the extended SFX bank and use the \$2600 to \$29FF bank-window aliases described by the include files. Chapter 12 gives the full typed setup for putting sample bytes in memory and triggering a channel.

11.7 Media loader

Longer songs and sample files can be started through the media loader. SOUND `PLAY "name"` writes these registers for you:

Address	Name	Purpose
\$F2300	MEDIA_NAME_PTR	Pointer to a null-terminated filename.

Address	Name	Purpose
\$F2304	MEDIA_SUBSONG	Subsong number.
\$F2308	MEDIA_CTRL	1 play, 2 stop.
\$F230C	MEDIA_STATUS	0 idle, 1 loading, 2 playing, 3 error.
\$F2310	MEDIA_TYPE	1 SID, 2 PSG, 3 TED, 4 AHX, 5 POKEY, 6 MOD, 7 WAV, 8 MIDI/MUS.
\$F2314	MEDIA_ERROR	0 ok, 1 not found, 2 bad format, 3 unsupported, 4 invalid path, 5 too large.

Native BASIC forms:

```
10 SOUND PLAY "SONG"
20 SOUND PLAY "SONG", 1
30 SOUND STOP
```

If loading fails, SOUND PLAY raises a BASIC error after the status poll reports MEDIA_STATUS value 3.

11.8 The file-player register rhythm

The file players all follow the same bus habit, even though their formats differ. A program puts the music or sample bytes in readable memory, writes a staged pointer and length, then writes a start bit to the player's control register. The player copies or parses that block before it becomes audible. While that work is in progress, the busy bit is the proof that the request is still alive.

Stop cancels the current request. A later valid start supersedes an older in-flight start, so a program should read the status register after changing pointer, length, loop, pause, or volume fields. The individual chapters give the exact register names, extra fields, and error bits for AHX, PSG file playback, SID file playback, TED and POKEY players, MOD, WAV, and MIDI/MUS. The live MIDI port is the exception inside this family: it is an event stream, so each byte is consumed as it is written instead of staging a file block first.

11.9 Global effects

The global effects are shared by every engine.

Register	Address	Range	Purpose
FILTER_CUTOFF	\$F0820	0 to 255	Filter cutoff.
FILTER_RESONANCE	\$F0824	0 to 255	Resonance.
FILTER_TYPE	\$F0828	0 to 3	0 off, 1 low-pass, 2 high-pass, 3 band-pass.
FILTER_MOD_SOURCE	\$F082C	0 to 3	Modulation source channel.
FILTER_MOD_AMOUNT	\$F0830	0 to 255	Modulation amount.
OVERDRIVE_CTRL	\$F0A40	0 to 255	Drive amount.
REVERB_MIX	\$F0A50	0 to 255	Dry/wet mix.
REVERB_DECAY	\$F0A54	0 to 255	Tail length.

The BASIC forms are:

```

10 SOUND FILTER 200,128,1
20 SOUND FILTER MOD 2,200
30 SOUND OVERDRIVE 80
40 SOUND REVERB 90,140

```

Effect changes are immediate. To make several changes as one audible step, set AUDIO_CTRL bit 1, write the effect registers, then clear bit 1 again while leaving bit 0 set.

11.10 Plus processing paths

Several engines have a **Plus** switch. Plus mode is not a second register map and it is not a different chip. It is an enhanced output processing path for that engine: the tone, envelope, player, and data registers keep their normal meanings, while the mixer uses a different gain curve, per-voice balance, smoothing, drive, stereo spread, or room processing as implemented by that engine.

The common programming rule is simple:

1. Start the engine in its normal way.
2. Switch Plus on with the engine's BASIC command or control register.
3. Read the control register back if you need proof.
4. Switch Plus off without rewriting the sound registers.

Engine	BASIC form	Control register
PSG	PSG PLUS ON / PSG PLUS OFF	PSG_PLUS_CTRL at \$F0C20
SID	SID PLUS ON / SID PLUS OFF	SID_PLUS_CTRL at \$F0E19
TED	TED PLUS ON / TED PLUS OFF	TED_PLUS_CTRL at \$F0F05
POKEY	POKEY PLUS ON / POKEY PLUS OFF	POKEY_PLUS_CTRL at \$F0D09
AHX	AHX PLUS ON / AHX PLUS OFF	AHX_PLUS_CTRL at \$F0B80

Individual chapters give the short compare listing and the engine-specific audible difference. This chapter is the rule that keeps those listings from becoming five copies of the same explanation.

11.11 BASIC and direct access map

Form	Engine or block
SOUND ch, freq, vol[, wave[, duty]]	SoundChip channel 0 to 3.
ENVELOPE ch, atk, dec, sus, rel	SoundChip ADSR.
GATE ch, ON / GATE ch, OFF	SoundChip gate bit.
SOUND WAVE ch, type	SoundChip waveform select.
SOUND NOISE ch, mode	SoundChip noise mode.
SOUND SWEEP ch, enable, period, shift	SoundChip sweep.
SOUND SYNC ch, source	SoundChip hard sync.
SOUND RINGMOD ch, source	SoundChip ring modulation.
SOUND FILTER ...	Global filter.

Form	Engine or block
SOUND REVERB ...	Global reverb.
SOUND OVERDRIVE ...	Global overdrive.
SOUND PLAY ... / SOUND STOP	Media loader.
PSG ...	PSG engine.
POKE8 &H000F0C30,value	SN76489 latch/data port.
SID ...	SID family.
TED ... audio forms	TED audio.
POKEY ...	POKEY.
AHX ...	AHX playback.
SOUND MOD ... / MOD STATUS	MOD playback.
SOUND PLAY or raw WAV registers	WAV playback.
SOUND PLAY or raw MIDI player registers	MIDI/MUS playback.
MIDI NOTE, MIDI PROG, MIDI CTRL, MIDI SEND, MIDI RESET	Live MIDI events through RawlandMini.

11.12 Limits

- The mixer sums engines; it does not reserve exclusive ownership of the output for any one engine.
- Engine volume ranges are engine-specific; each engine chapter gives its own register format.
- Global effects apply to the full mix.
- Media loading is asynchronous at the register level, so check MEDIA_STATUS if you drive the media loader with POKE32.
- BASIC SOUND PLAY includes a bounded status poll and reports a BASIC error if loading reaches the error state.

Chapter 12 covers the SoundChip and SFX blocks in detail.