

Chapter 17 - POKEY

POKEY is a four-channel sound chip with byte frequency dividers, per-channel distortion, shared clock-routing control, high-pass filter options, and a readable polynomial random tap. It is good for clean square tones, metallic noise, buzzing percussion, and 16-bit paired bass voices.

The first useful program is one pure channel:

```
10 REM POKEY FIRST TONE
20 POKE32 &H00F0800,1
30 REM NORMAL 64 KHZ CLOCK PATH
40 POKEY CTRL 0
50 REM CHANNEL 1, PURE TONE, FULL VOLUME
60 POKEY 1,121,&HAF
```

&HAF is distortion 5 shifted into bits 5 to 7, plus volume 15. With AUDCTL clear, channel 1 uses the 64 kHz clock path.

Try changing the divider in line 60 to 60. The channel rises in pitch because POKEY divides the clock by the register value plus one.

17.1 Shape of the chip

Item	Value
Channels	4, numbered 1 to 4 in BASIC
Channel registers	AUDF divider byte and AUDC control byte
Distortion	Eight modes from pure tone to polynomial noise
Volume	Four bits per channel
Clock sources	64 kHz path, 15 kHz path, and 1.79 MHz paths for channels 1 and 3
Pairing	Channels 1+2 and 3+4 can form 16-bit periods
Filter options	High-pass channel 1 by channel 3, and channel 2 by channel 4
Random tap	POKEY_RANDOM at \$F0D0A

17.2 Register block

POKEY registers are byte-wide from \$F0D00 to \$F0D0A.

Address	Name	Purpose
\$F0D00	POKEY_AUDF1	Channel 1 divider
\$F0D01	POKEY_AUDC1	Channel 1 control
\$F0D02	POKEY_AUDF2	Channel 2 divider
\$F0D03	POKEY_AUDC2	Channel 2 control

Address	Name	Purpose
\$F0D04	POKEY_AUDF3	Channel 3 divider
\$F0D05	POKEY_AUDC3	Channel 3 control
\$F0D06	POKEY_AUDF4	Channel 4 divider
\$F0D07	POKEY_AUDC4	Channel 4 control
\$F0D08	POKEY_AUDCTL	Shared audio control
\$F0D09	POKEY_PLUS_CTRL	Bit 0 enables POKEY Plus
\$F0D0A	POKEY_RANDOM	Read-only random byte

Use POKE8 for raw register work. The BASIC POKEY helper writes the same byte registers.

17.3 Channel control

Each AUDC byte has volume, volume-only mode, and distortion:

Bits	Field	Meaning
0 to 3	VOLUME	Level 0 to 15
4	VOLUME_ONLY	Output the volume level without oscillator pitch
5 to 7	DISTORTION	Select one of eight timbres

Distortion values:

Value	Timbre
0	17-bit polynomial mixed with 5-bit polynomial
1	5-bit polynomial
2	17-bit polynomial mixed with 4-bit polynomial
3	5-bit polynomial mixed with 4-bit polynomial
4	17-bit polynomial noise
5	Pure square tone
6	4-bit polynomial buzz
7	17-bit polynomial plus pulse

POKEY ch,divider,ctrl accepts BASIC channel numbers 1 to 4. Out-of-range channel numbers are ignored.

```

10 REM POKEY NOISE HIT
20 POKE32 &H00F0800,1
30 REM UNPAIRED CHANNELS
40 POKEY CTRL 0
50 FOR V=15 TO 0 STEP -1
60 REM DISTORTION 4 PLUS VOLUME V
70 POKEY 2,8,&H80+V
80 FOR Q=1 TO 60
90 NEXT Q
100 NEXT V
110 POKEY 2,8,&H80

```

Expected result: channel 2 makes a short white-noise style hit that fades out. &H80 selects distortion 4; the low nibble is the changing volume.

Line 40 clears AUDCTL, so channel 2 uses the normal clock. Line 70 builds an AUDC2 byte from distortion 4 and the current volume. The final line leaves the same noise timbre selected, but with volume zero.

Try changing the divider in line 70 from 8 to 3 for a sharper hit.

17.4 AUDCTL

POKEY_AUDCTL changes clock sources, channel pairing, polynomial length, and high-pass routing.

Bit	Field	Meaning
0	CLOCK_15KHZ	Use the 15 kHz clock path instead of the 64 kHz path
1	HIPASS_CH1	High-pass channel 1 using channel 3 as cutoff source
2	HIPASS_CH2	High-pass channel 2 using channel 4 as cutoff source
3	CH4_BY_CH3	Join channels 3 and 4 into a 16-bit period
4	CH2_BY_CH1	Join channels 1 and 2 into a 16-bit period
5	CH3_179MHZ	Channel 3 uses the 1.79 MHz clock
6	CH1_179MHZ	Channel 1 uses the 1.79 MHz clock
7	POLY9	Use 9-bit polynomial behaviour instead of 17-bit behaviour

For an unpaired channel:

```
frequency = base_clock * 0.5 / (divider + 1)
```

For a paired channel, the low channel is the audible master. Its period is:

```
period = low_divider + 256 * high_divider
frequency = base_clock * 0.5 / (period + 1)
```

This example uses channels 1 and 2 as one 16-bit voice:

```
10 REM POKEY 16 BIT BASS
20 POKE32 &H00F0800,1
30 REM JOIN CHANNELS 1 AND 2
40 POKEY CTRL &H50
50 POKEY 1,&H40,&HAF
60 REM CHANNEL 2 HOLDS THE HIGH PERIOD BYTE
70 POKEY 2,&H20,0
80 FOR T=1 TO 3000
90 NEXT T
100 POKEY 1,&H40,&HA0
```

&H50 joins channels 1 and 2 and gives channel 1 the 1.79 MHz clock. Channel 2 supplies the high period byte and is muted.

Line 50 writes the low byte and audible control for the pair. Line 70 writes the high byte and keeps channel 2 silent. Line 100 clears the volume nibble while leaving the pure-tone distortion selected.

Try changing line 70 to `POKEY 2,&H10,0`; the paired period is shorter and the bass rises.

17.5 Random tap

Reading `POKEY_RANDOM` advances and returns a byte from the polynomial random source:

```
10 REM POKEY RANDOM BYTES
20 REM EACH READ ADVANCES THE POLYNOMIAL
30 FOR I=1 TO 8
40 PRINT PEEK8(&H00F0D0A)
50 NEXT I
```

Expected result: eight byte values in the range 0 to 255.

The random tap is read-only. There is no setup register for this example; each `PEEK8` returns the next byte from POKEY's polynomial state.

Try changing the loop limit from 8 to 16 to print a longer sample.

17.6 POKEY Plus

POKEY Plus follows the shared Plus rule from Chapter 11. `POKEY PLUS ON` writes 1 to `POKEY_PLUS_CTRL` at `$F0D09`; `POKEY PLUS OFF` writes 0. The register map and distortion numbers stay the same. The POKEY-specific difference is a changed output curve, per-channel mix gains, oversampling, light drive, and a short room effect.

```

10 REM POKEY PLUS COMPARE
20 POKE32 &H00F0800,1
30 POKEY CTRL 0
40 POKEY 1,121,&HAF
50 REM LISTEN TO PLAIN POKEY FIRST
60 FOR T=1 TO 2500
70 NEXT T
80 POKEY PLUS ON
90 PRINT PEEK8(&H00F0D09)
100 REM NOW LISTEN TO POKEY PLUS
110 FOR T=1 TO 2500
120 NEXT T
130 POKEY PLUS OFF
140 PRINT PEEK8(&H00F0D09)
150 POKEY 1,121,&HA0

```

Lines 80 and 130 switch only the output processing. The POKEY registers and distortion numbers stay the same. Lines 90 and 140 print 1 and then 0 to show the control byte.

Try changing the control byte in line 40 from &HAF to &H6F; the Plus comparison uses a buzzy distortion.

17.7 Player registers

The POKEY player streams POKEY register music from memory. The BASIC command is `SAP PLAY` because `SAP` is the common file format for Atari POKEY music. This is still the POKEY playback path; `SAP` is not a separate sound chip in this book.

Address	Name	Purpose
\$F0D10	SAP_PLAY_PTR	Start address of the music data
\$F0D14	SAP_PLAY_LEN	Length in bytes
\$F0D18	SAP_PLAY_CTRL	Write 1 start, 2 stop, 5 start loop
\$F0D1C	SAP_PLAY_STATUS	Bit 0 busy, bit 1 error
\$F0D20	SAP_SUBSONG	Subsong number

```

10 REM POKEY MEMORY PLAYBACK
20 REM START SUBSONG 0 FROM MEMORY
30 SAP PLAY &H0020000,8192,0
40 S=POKEY STATUS
50 PRINT S
60 IF S AND 2 THEN PRINT "POKEY ERROR"

```

If the memory block is valid POKEY music data, `POKEY STATUS` reports busy while playback is active. If the pointer, length, or data is invalid, bit 1 is set.

Line 30 writes the pointer, length, subsong, and start command. Lines 40 to 60 read the playback status and report only errors. They do not stop playback; a valid POKEY tune should keep playing until it ends, loops, or a later stop command is typed.

To stop POKEY playback later:

```
10 SAP STOP
20 PRINT POKEY STATUS
```

17.8 Side effects and limits

Writing an AUDF divider retriggers the matching channel. In a 16-bit pair, the low-numbered channel is the audible master and the high-numbered channel is muted. AUDCTL high-pass bits apply filter settings to channels 1 and 2 using channels 3 and 4 as cutoff sources.

POKEY_RANDOM ignores writes. Reading it advances the random state. Raw POKEY registers are byte-wide; use POKE8 for direct byte writes and the 32-bit player registers only for playback pointer, length, control, and status.

The next chapter covers AHX, a four-voice tracker-style music engine.