

# Chapter 48 - From Floating Point To Tables

---

BASIC can write the idea directly:

```
A=A+0.03
Z=Z+0.01
CA=COS(A)/SC
SA=SIN(A)/SC
```

Machine code usually does not do that inside the frame loop. It keeps small accumulators, uses tables, and writes the same 16.16 values to the blitter.

The goal is not to change the effect. The goal is to change how the CPU arrives at the six Mode 7 numbers.

## 48.1 Phase Accumulators

---

A floating-point phase wraps at  $2\pi$ :

```
A=A+0.03:IF A>6.28318 THEN A=A-6.28318
```

The native rotozoomers use a 16-bit accumulator. The high byte is a table index from 0 to 255; the low byte is the fractional part.

```
accumulator:  aaaa aaaa ffff ffff
                |      | |      |
                index  fraction
```

Adding 313 each frame is close to the BASIC 0.03 radians step:

```
0.03 * 256 / (2*pi) * 256 = about 313
```

Adding 104 each frame is close to the BASIC 0.01 zoom-phase step.

## 48.2 Sine And Cosine

---

The table has 256 signed entries. A full turn is one trip through the table.

```
sin = table[index]
cos = table[(index + 64) AND 255]
```

The 64 offset is a quarter turn. A cosine is a sine shifted by a quarter of the table.

The table values use 8.8 scale. A value near 256 means about 1.0. A value near 0 means 0.0. A value near -256 means about -1.0.

## 48.3 Reciprocal Zoom

The BASIC version divides by scale:

```
CA=COS(A)/SC
SA=SIN(A)/SC
```

Integer code prefers multiplication. The native versions keep a reciprocal table, then multiply:

```
CA = cos_table_value * recip_table_value
SA = sin_table_value * recip_table_value
```

The product is shifted into the 16.16 range required by the blitter.

## 48.4 The Same Six Outputs

Whether the CPU used BASIC floating point or integer tables, it must write the same six values:

| Value | Meaning                       |
|-------|-------------------------------|
| U0    | Starting U coordinate.        |
| V0    | Starting V coordinate.        |
| duCol | U step for one screen column. |
| dvCol | V step for one screen column. |
| duRow | U step for one screen row.    |
| dvRow | V step for one screen row.    |

That is the bridge from BASIC to assembly. The mathematics is the same. Only the arithmetic method changes.

## 48.5 A Table Probe In BASIC

This small BASIC listing shows the indexing idea without writing a full table:

```
10 REM TABLE INDEX PROBE
20 A=0
30 FOR F=1 TO 12
40 I=INT(A/256) AND 255
50 C=(I+64) AND 255
60 PRINT "IDX ";I;" COS ";C
70 A=A+313
80 NEXT F
```

Expected result: the index advances by about one entry per frame, with a fractional remainder carried in the low byte.

## 48.6 Why This Matters

Tables make the assembly listings readable:

- The frame loop advances angle and scale.

- The high byte selects sine and reciprocal entries.
- The CPU combines them into CA and SA.
- The CPU derives  $U\theta$ ,  $V\theta$ , and the four deltas.
- The blitter draws the frame.

Chapter 49 shows those steps in the IE64 and IE32 rotozoomers.