Table of Contents

Illustrations

List of Tables

SECTION 1

Introduction

“This document describes three table look-up methods for sine-wave generation . . .”

Sine-wave generators are used in communications and control applications (1,2). With the introduction of high-speed high-precision digital signal processors, stable and low distortion sine waves of any frequency can be produced digitally using some form of table look-up with interpolation to reduce distortion (3,4,5). This document describes three table look-up methods for sine-wave generation and provides the Total Harmonic Distortion (THD) performance and Maximum Synthesizable Frequency (MSF) for each case.

A routine for synthesizing sine waves having frequencies limited to integer multiples of the Fundamental Table Frequency (FTF) is described in **Section 3.1**. The MSF is highest using this approach. In **Section 3.2** a routine using only direct table look-up for synthesizing sine waves having frequencies which are fractional multiples of the FTF is described. This approach can be used to synthesize sine waves with frequencies which are not integer multiples of the FTF but they have substantially higher THD. A routine for synthesizing sine waves using table look-up with interpolation is described in **Section 4**. Sine waves with frequencies which are not limited to multiples of the FTF and yet have low THD are possibly using this synthesis approach. ■

SECTION 2

Background

"The maximum value delta can assume is N/2 since at least two samples per cycle are required to synthesize a sine wave without aliasing."

The values used for approximating a sine wave are stored in a table in memory as follows:

Table 2-1 *Sine-Wave Table Values*

i=N-1	sin[(N-1) • 360/N]
	•
	•
i→	sin[(i) • 360/N]
	•
	•
	sin[(2) • 360/N]
	sin[(1) • 360/N]
BASE ADDRESS (I=0)	sin[(0) • 360/N]

where:

N = *the table length*

i = *the index into the table;* $0 \le i \le N - 1$

sin[i•360/N] = *the value stored at the jth location in the table. i • 360/N is the angle, in degrees, for which the sine function is calculated. Throughout the remainder of this document, the abbreviation S[i] will be used to represent this function.*

Note that the length of the table can be traded off against software in that, except for the sign, only one quarter of the table values are unique. The frequency of the digital sine wave generated depends upon the time interval and the phase angle increment (delta, Δ) between successive table accesses.

If delta is unity (i.e., the entries are read sequentially) and the table is accessed every T seconds (T is referred to as the sampling interval), the FTF of the sine wave synthesized will be:

$$FTF = 1/NT \text{ Hz} \qquad \text{Eqn. 2-1}$$

On the other hand, if delta is greater than unity, e.g., every second (Δ = 2) or third (Δ = 3) entry is read, (see Figure 3-2) and the table is still accessed every T seconds, then the frequency of the sine wave synthesized will be:

$$f = \Delta \bullet FTF \text{ Hz}; \;\; \Delta \leq N/2 \qquad \text{Eqn. 2-2}$$

If delta is an integer value, only multiples of the FTF can be generated; whereas, if delta is allowed to be fractional, any frequency up to the MSF can be generated. The maximum value delta can assume is N/2 since at least two samples per cycle are required to synthesize a sine wave without aliasing.

The value of the sample output, x(n), will depend on the initial phase angle, phi (ϕ), and the time or sample index, n, as follows:

$$x(n) = \sin[\phi + n \bullet \Delta \bullet 360/N]; \quad n = 0, 1, 2, \ldots \qquad \text{Eqn. 2-3}$$

The THD of the synthesized sine wave depends upon the length of the table, N, the accuracy (number of bits of precision) of the data stored in the table and the value of delta. ■

SECTION 3

Direct Table Look-Up

3.1 Integer Delta Implementation

"When fractional values of delta are used, samples of points between table entries must be estimated using the table values."

This implementation is a direct table look-up method with delta being a positive integer number. Because delta is limited to being an integer all the required samples are contained within the table; no approximations are necessary. Figure 3-1 illustrates this method using N = 8 and Δ = 2.

The assembler listing for the SINe-Wave Generation Integer Delta ("SINWGID") routine is presented in Figure 3-2. The corresponding memory map is shown in Figure 3-3.

Although no assembler options are indicated in this listing, a number of options are available. Refer to the Macro Assembler Reference Manual for information (6).

The memory locations can also be changed to suit the user's needs. In this routine the sine table (N=256) is in internal Y ROM starting at address HEX 100 to minimize external accesses and preserve RAM for data variables. The actual 256 sine table values stored are given in **Appendix B**. The output location is chosen to be an address in external I/O space, Y:$FFE0.

The part of the routine that generates the sine samples is given in the form of a subroutine to facilitate calculation of the MSF. Only a MOVEP and a JMP instruction must be executed to output a sample point. This MOVEP instruction utilizes the modulo addressing capability of the DSP56001/2. Three address arithmetic unit registers are used in this routine; the address register, R1, contains the index into the sine table, the offset register, N1, contains the value of delta, and the modifier register, M1, contains the value N-1 to set up the modulo buffer.

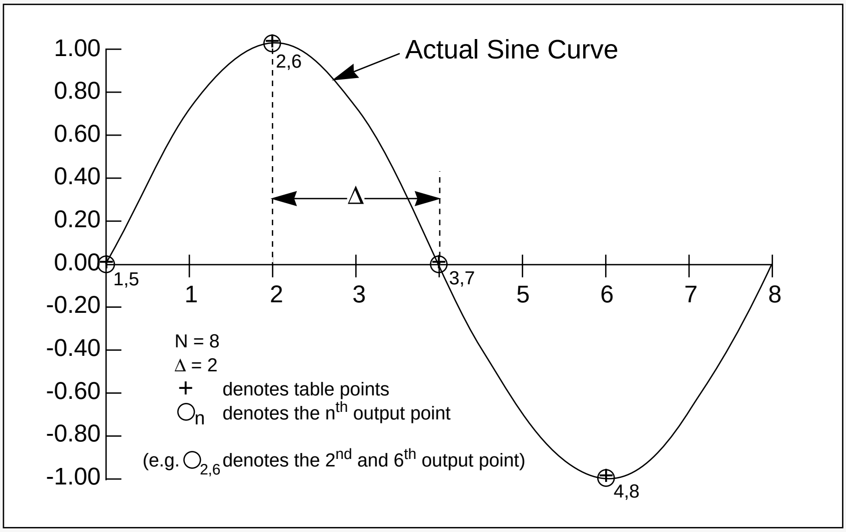

Figure 3-1 *Integer Delta*

NOTE: In Figure 3-2, the numbers in the parentheses at the far right of the subroutine portion of the listing indicate the number of instruction cycles required to complete the specific instruction. These numbers have been manually included as comments to ease the calculation of MSF in Eqn. 3-1.

```
Motorola DSP56000 Macro Cross Assembler Version 1.10 06-18-87 13:03:01 sinwgid.asm
Page 1


                 page 132,50,0,10
            ;SINWGID is a direct table look-up routine for SINE Wave Generation
            ;using Integer Delta values. The sine table is in on-chip ROM starting
            ;at address Y:$100 and contains 256 (N) entries which correspond to the
            ;sines of 256 equally spaced angles between 0 and 360 degrees.
            ;Integer delta values between 1 and N/2 can be used to step through the
            ;table. Here Delta=2 and is saved in N1. The table size is saved in M1.
            ;
   00000060          start   equ    $60                 ;starting address.
   00000100          sinep   equ    $100                ;sine table starting address.
   000000FF          mask    equ    255                 ;set mask = table size - 1.
   00000002          delta   equ    2                   ;set delta = 2
   0000FFE0          sinea   equ    $ffe0               ;address of output device.
   P:0060                    org    p:start
   P:0060   0506BA                  movec      #6, omr  ;enable on-chip ROM with sine table
   P:0061   61F400           move              #sinep,r1 ;initialize sine table pointer.
            000100
   P:0063   05FFA1           movec             #mask,m1 ;set up modulo N addressing.
   P:0064   390200           move              #delta,n1 ;offset equals Delta.
   P:0065   0BF080           jsr    sineg               ;jump to subroutine.
            000066
   P:0067   09C9E0 sineg     movep             y:(r1)+n1,y:sinea  ;output sample.      (2)
   P:0068   00000C           rts                                  ;return from subroutine.(2)
                             end                                  ;the end of listing.

0 Errors
0 Warnings
```

Figure 3-2 *"SINWGID" Routine*

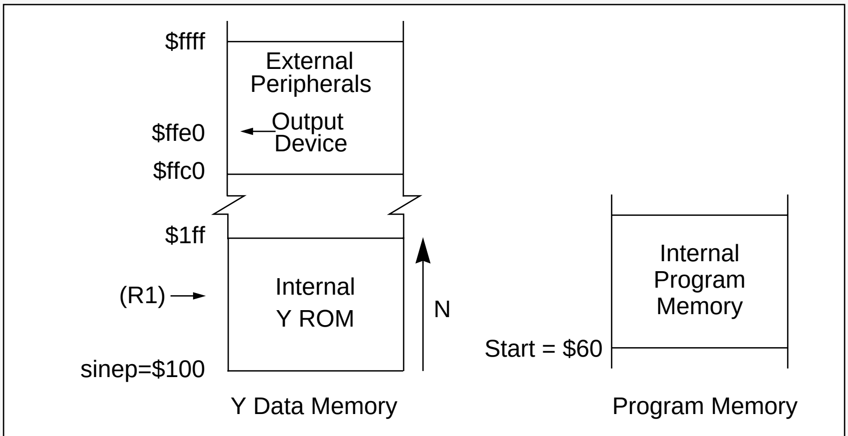

Figure 3-3 *"SINWGID" Memory Map*

Resources required for the “SINWGID” routine:

Data ROM	256	24-bit words for the sine table
Program Memory	6	24-bit words for the initialization
	2	24-bit words for the subroutine

The MSF for the given sine-wave subroutine would be achieved by replacing the RTS instruction with the JMP SINEG instruction. After performing the above replacement, MSF is given by:

$$MSF = \frac{1}{2 \bullet Icyc \bullet SIC} \qquad \text{Eqn. 3-1}$$

where:

SIC = the total number of Subroutine Instruction Cycles in “sineg”. SIC can be obtained by adding the numbers in the brackets incorporated in the subroutine

Icyc = the instruction cycle execution time

For example, in Eqn. 3-1, if Icyc = 50 ns and SIC = 4 cycles:

MSF = 1 /(2 • 50 • 4) MHz

= 2.5 MHz

The results above compare very favorably in terms of both program memory requirements and execution speed with those for other competitive products (7).

3.2 Real Delta Implementation

This implementation is a direct table look-up method with delta being a positive real number, that is, a number consisting of an integer and a fractional part. When fractional values of delta are used, samples of points between table entries must be estimated using the table values. The most straightforward estimation is to use the previous table entry. This approach is described in this section and illustrated in Figure 3-4 for N = 8 and Δ = 2.5.

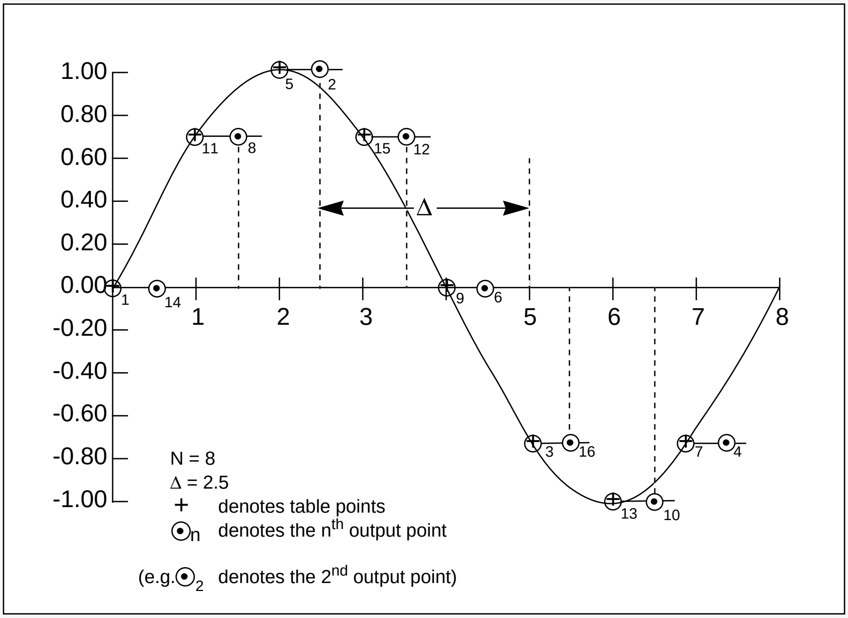

__Figure 3-4__ Real Data Without Interpolation

The assembler listing for the SINe-Wave Generation Real Delta (“SINWGRD”) routine is presented in Figure 3-5. The memory map (see Figure 3-6) is the same as that used for the SINWGID routine except that R4 is used to point to the output device. This saves one cycle since the previous sample can be output while the accumulator is being updated using the parallel move feature of the DSP56001/2. The programmer's model for the Data ALU is presented in Figure 3-7.

```
Motorola DSP56000 Macro Cross Assembler Version 1.10 06-18-87 13:03:27 sinwgrd.asm Page 1

               page    132,50,0,10
          ;SINWGRD is a direct table look-up routine for SINe Wave Generation using
          ;positive Real numbers for Delta. The sine table is in on-chip ROM starting at
          ;address r:$100 and contains 256 (N) entries corresponding to the sine values
          ;of 256 equally spaced angles between 0 and 360 degrees. N must be a power of
          ;two. Delta can be between 0.0 and N/2. Here delta = 2.5 and is saved in B1 and
          ;B0. The table size minus one, N-1, is saved in X1.
          ;
    00000060        start  equ $60                        ;program starting address.
    00000100        sinep  equ $100                       ;sine table starting address.
    000000FF        mask   equ $ff                        ;N - 1.
    2.500000        delta  equ 2.5                        ;delta = 2.5
    0000FFE0        sinea  equ $ffe0                      ;address of output device.
    P:0060                 org p:start
    P:0060   0506BA            movec   #6,omr             ;enable on-chip ROM with sine table
    P:0061   64F400        move        #sinea,r4          ;set up output pointer.
             00FFE0
    P:0063   51F400        move        #(delta-@cvi(delta)),b0;store fract. part in b0.
             400000
    P:0065   61F43A        asl b       #sinep, r1         ;eliminate sign bit, init. r1.
             000100
    P:0067   2D0200        move        #acvi(delta),b1    ;integer part of delta in b1.
    P:0068   390013        clr a       #0,n1              ;initialize acc. a and n1.
    P:0069   45F400        move        #>mask, x1         ;set modulo mask.
             0000FF
    P:006A   0BF080        jsr sineg                      ;jump to subroutine.
             00006C
    P:006C   4EE910 sineg  add b,a             y:(r1+n1),y0;update a, get sine value.   (2)
    P:006D   4E6466        and x1, a           y0,y:(r4)  ;mask a, output sample.       (1)
    P:006E   219900        move        a1,n1              ;update offset register nl.   (1)
    P:006F   00000C        rts                            ;return from subroutine.      (2)
                           end                            ;end of listing.
0 Errors
0 Warnings
```

Figure 3-5 *“SINWGRD” Routine*

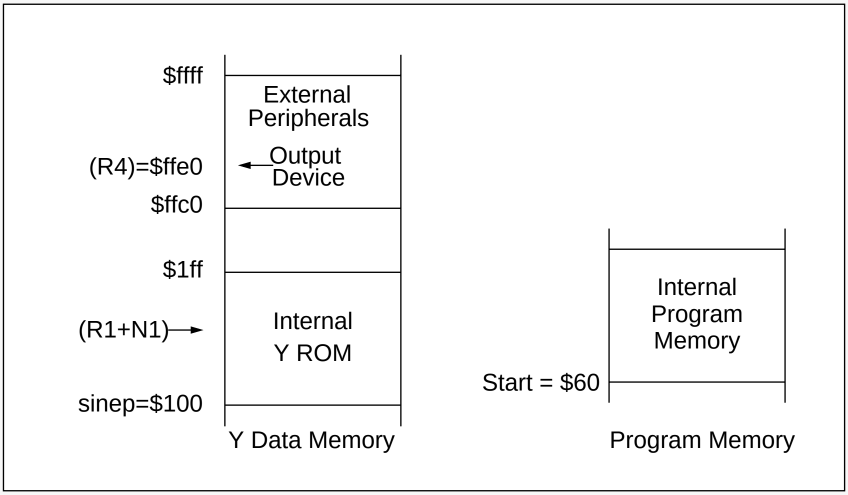

***Figure 3-6** “SINWGRD” Memory Map*

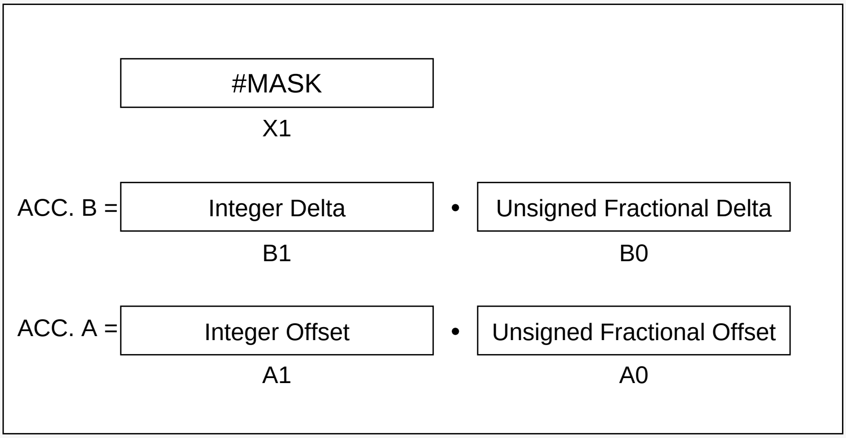

***Figure 3-7** “SINWGRD” Data ALU Programmer’s Model*

Given delta is now a real number, the inherent modulo addressing mode in the DSP56001/2. cannot be used and therefore a modulo addressing scheme must be implemented in software. The following is a description of one approach to implementing such a scheme. This approach uses both accumulators.

In this routine the integer part of delta is stored in the upper portion of an accumulator (B1 for the example given), and the fractional part is stored in the lower portion of the same accumulator (B0). Care should be taken to ensure that the fractional part is stored correctly. The number moved into B0 as a result of subtracting the integer portion from delta, i.e. Δ -@cvi(Δ), is a signed, i.e. positive, fraction. This is undesirable since the sign bit should be associated with the integer portion. Therefore a shift to the left must be performed to eliminate the sign bit leaving the unsigned fraction in B0.

The separation of integer and fractional portions is done by using the assembler “ConVert to Integer” built-in function (@cvi). The CVI function converts real numbers to integers by simply truncating the fractional part of the number. Therefore:

- @cvi(Δ) returns the integer portion of delta
- Δ-@cvi(Δ) returns the signed fractional part of delta

This assumes that delta is not a runtime variable. If delta is required to be a runtime variable, it can be passed by separating it into a signed integer and unsigned fraction which are loaded into B1 and B0 respectively.

A second accumulator, A in this example, contains the positive real number value used to load the offset register, N1, which is used to offset the pointer, R1. R1 is used to address the correct location in the sine table. Since the table index must be an integer, only A1 is moved to N1. Incrementing by delta, however, is done on both the integer and fractional parts of the A accumulator. It should be noted that the increment of the register R1 is done by indexing the register with the offset register N1, thus leaving the original value of R1 (base address) unchanged. Note that the binary point is at an imaginary point between the two 24-bit parts of the accumulators. Initially we have:

ACC. A =	0	•	0
	A1		A0

After the first addition we get:

ACC. A =	0+Integer Delta	•	Unsigned Fractional Delta
	A1		A0

To wrap the pointer, A1, around when it is incremented past the length of the table, a masking operation is performed on A. The mask, contained in X1, is the table length minus one, N-1, where N is restricted to be a power of two. This restriction is necessary to ensure that the mask consists of k least significant ones where k is defined by 2k = N. Using the above masking operation, the value of A1 and consequently N1, is restricted to be between 0 and N-1.

Resources required for the “SINWGRD” routine:

Data ROM	256	24-bit words for the sine table
Program Memory	11	24-bit words for the initialization
	4	24-bit words for the subroutine memory
MSF		1/(2 • 50 • 6) MHz
		= 1.667 MHz

3.3 Harmonic Distortion

Due to the fact that the sine wave generated is an approximation, not all of the energy is at the fundamental frequency; a certain amount of the energy of the generated samples falls into frequencies other than the fundamental. Those frequencies are:

1. Harmonic frequencies, hf, i.e. integer multiplies of the fundamental frequency, f
2. Subharmonic frequencies, sf, where s = h/d and h, d are integers

The resulting noise is measured in terms of THD, given by the following equation:

$$\text{THD} = \frac{\text{spurious harmonic energy}}{\text{total energy of the waveform}} \qquad \text{Eqn. 3-2}$$

When using the table look-up algorithms, harmonic distortion occurs from two distinct sources —

1. Quantization Error: Since the sine table values stored in memory are of finite word length (24 bits in this case) the sine values cannot be represented exactly. Quantization error is directly proportional to the word length. For example the sine of 45 degrees in decimal will be:

0.7070707	using 24 bits
0.7071	using 16 bits

2. Sampling Error: When points between table entries are sampled, i.e., delta is not an integer, then large errors are introduced because these points must be estimated from the table values. Since the sampling errors are derived from the table values, sampling errors are always greater than quantization errors.

The THD for different deltas and different table sizes is shown in Table 3-1(a), Table 3-1(b), and Table 3-2.

Table 3-1(a) *Total Harmonic Distortion — Integer Delta*

N	$\Delta = 2$	$\Delta = 3$
8	$3.5527141 \cdot 10^{-15}$	$5.7949068 \cdot 10^{-15}$
16	$5.7949068 \cdot 10^{-15}$	$3.6274700 \cdot 10^{-15}$
32	$3.6274700 \cdot 10^{-15}$	$2.8356370 \cdot 10^{-15}$
64	$2.8356370 \cdot 10^{-15}$	$3.4157912 \cdot 10^{-15}$
128	$3.4157912 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$
256	$2.8659804 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
512	$2.6423040 \cdot 10^{-15}$	$2.6142553 \cdot 10^{-15}$
1024	$2.6142553 \cdot 10^{-15}$	$2.4857831 \cdot 10^{-15}$

Table 3-1(b) *Total Harmonic Distortion — Integer Delta*

Δ	N = 64	N = 256
1	$3.4157912 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
2	$2.8356370 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$
3	$3.4157912 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
4	$3.6274702 \cdot 10^{-15}$	$3.4157912 \cdot 10^{-15}$
5	$3.4157912 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
6	$2.8356370 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$
7	$3.4157912 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
8	$5.7949069 \cdot 10^{-15}$	$2.8356370 \cdot 10^{-15}$
9	$3.4157912 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
10	$2.8356370 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$

Table 3-2 *Total Harmonic Distortion — Real Delta*

	N = 64	N = 128	N = 256
	$2.8356370 \cdot 10^{-15}$	$3.4157912 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$
	$7.5141324 \cdot 10^{-04}$	$1.8539830 \cdot 10^{-04}$	$4.7061084 \cdot 10^{-05}$
	$6.0107522 \cdot 10^{-04}$	$1.4805426 \cdot 10^{-04}$	$3.7649080 \cdot 10^{-05}$
	$7.5141324 \cdot 10^{-04}$	$1.8539830 \cdot 10^{-04}$	$4.7061084 \cdot 10^{-05}$
	$3.4157912 \cdot 10^{-15}$	$2.8659804 \cdot 10^{-15}$	$2.6423040 \cdot 10^{-15}$
	$7.5141324 \cdot 10^{-04}$	$1.8539830 \cdot 10^{-04}$	$4.7061084 \cdot 10^{-05}$
5	$7.9041085 \cdot 10^{-04}$	$1.9724369 \cdot 10^{-04}$	$4.9414069 \cdot 10^{-05}$

The equations as well as the FORTRAN code used for calculating the THD in the above tables are included in **Appendix A**. With respect to Table 3-1(a), Table 3-1(b), and Table 3-2 some important observations and conclusions can be drawn:

1. For integer delta the THD for $N \leq 1024$ is of the same order of magnitude. This can be seen in Table 3-1(a) and Table 3-1(b). For integer deltas the only errors which cause distortion are quantization errors.

2. For integer delta and small N the THD is nonuniform. This is evident for $\Delta = 2$ and $N \leq 128$ or $\Delta = 3$ and $N \leq 64$ in Table 3-1(a). For small N, the quantization error distribution is not uniformly distributed between ±1/2 Least Significant Bit (LSB) thereby causing the THD values to be nonmonotonic.

3. For integer delta and large N the THD decreases monotonically with increasing N. This is evident for $\Delta = 2$ and $N \leq 128$ or $\Delta = 3$ and $N \leq 64$ in Table 3-1(a). For large N, the quantization error distribution will tend to be uniform between ± 1/2 LSB resulting in the monotonic behavior.

4. THD for odd deltas and any N is constant. Similarly, THD for even deltas and any N is constant with the exception of delta equal to a power of two greater than 2 (see Table 3-1(b)). The explanation rests with the observation that all N points are used for generating sine waves using odd deltas. The N/2 even points are used twice for generating sine waves using even deltas with the exception of delta being a power of two greater than 2. For this exception, the points used in generating the sine waves are separated by delta and are used delta times. As noted in **Appendix A**, for integer deltas the total number of points used in the THD calculations will be independent of delta and equal to N.

5. THD depends on the fractional part of delta. The THD for $\Delta = 2.50$ is less than either 2.25 or 2.75 (see Table 3-2), because the fractional part generates an integer every other access for $\Delta = 2.50$ and every other fourth access for 2.25 or 2.75. Therefore, every other sample is free of sampling error for $\Delta = 2.5$ but only every fourth sample is free of sampling error for the other two cases. Observe that the THD for 2.25, 2.75, and 8.25 is the same. The THD for 11.625 is slightly higher because the fractional part forms an integer only every eighth access.

6. The THD for non-integer deltas decreases with increasing table length as seen in Table 3-2. Consider the same delta entries for different N. By increasing the table length, the difference between table entries decreases, resulting in better approximations to the non-integer samples, hence reduced sampling errors. ■

SECTION 4

Table Look-Up with Linear Interpolation

"... the THD improves dramatically for non-integer values of delta when using linear interpolation."

In order to synthesize a sine wave of any frequency with low distortion, an interpolation method must be used together with table look-up. By using interpolation, sine values between table entries can be represented more accurately. The easiest interpolation technique to implement is linear interpolation. For linear interpolation the sine value for a point between successive table entries is assumed to lie on the straight line between the two values. This is illustrated in Figure 4-1 where $\Delta = 2.5$ and $N = 8$. Contrast this figure with Figure 3-4.

The algorithm used for linear interpolation is based on the equation of a straight line:

$$y = m \bullet x + b$$

where: *m = the slope of the line*
x = the x-coordinate value
b = the initial y value
y = the new y value

For linear interpolation:

m = S[i + 1] - S[i]; i.e., it is the slope of the line segment between successive table entries i and i + 1

b = S[i], the value of the sine table at the base address plus the j^{th} offset location

x = the fractional part of the pointer, $0 < x < 1.0$

y = S[i + x], the approximated sample value

Therefore, we have:

$$S[i+x] = S[i]+x \bullet \{S[i+1]-S[i]\}$$

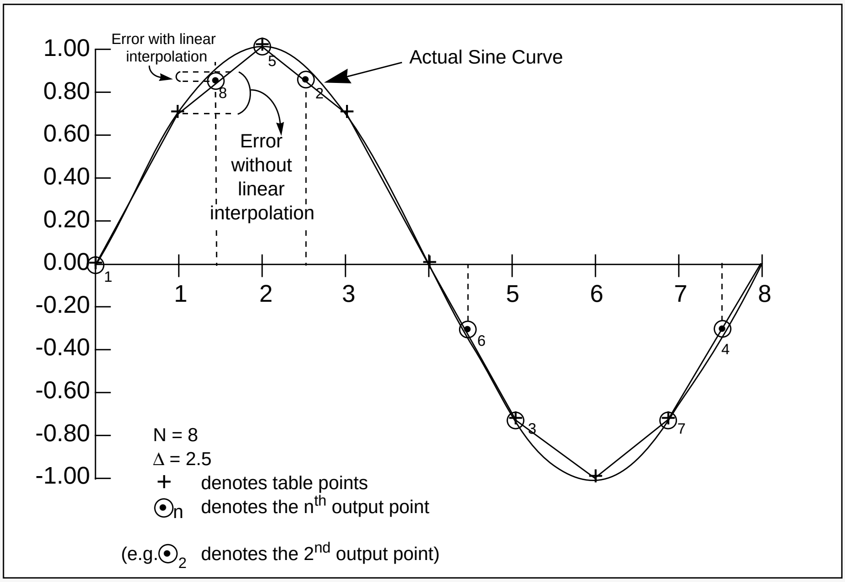

***Figure 4-1** Real Delta Interpolation*

The program flow diagram for the SINe-Wave Generation Linear Interpolation (“SINWGLI”) routine is presented in Figure 4-2.

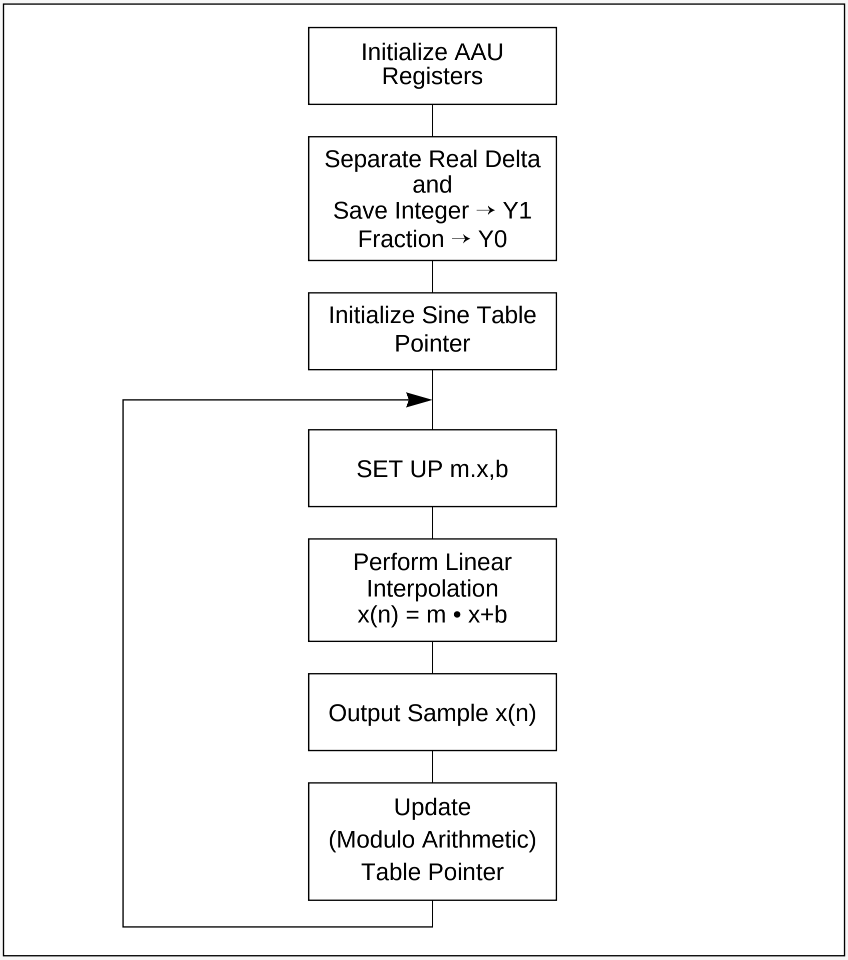

***Figure 4-2** “SINWGLI” Flow Diagram*

The assembler listing is given in Figure 4-3; the memory map is presented in Figure 4-4; and the Data ALU programmer's model is presented in Figure 4-5.

The sine table is stored in Y ROM starting at HEX 100. The address pointer, R1, points to the current sine value while address pointer, R2, points to the current plus one sine value, R2=R1+1. The slope between successive points is determined by subtracting the table entry pointed to by R2 from the table entry pointed to by R1.

For the “SINWGLI” routine, the fractional part of the linear approximation, x, is stored in the accumulator, A0, and the integer portion is stored in A1. A1 is “moved” into N1 which is used to index the R1 register. The same value that is moved in N1 is moved in N2 which is used to index R2. Note that in contrast to the “SINWGRD” routine, the content of A0 has to be right shifted so that it is in the correct positive signed fractional format prior to performing the multiplication to generate the linear approximation. Delta can be any positive real number between 0.0 and N/2. Accumulator B is first used to separate delta into a signed integer and an unsigned fraction (see “SINWGRD” routine), and then it is used to calculate the interpolated sample value. Note that an immediate long move (i.e., using the “>” sign) must be specified when saving the integer part of delta in Y1 because (@CVI(Δ)) would be interpreted as a signed fraction otherwise (8). The part of the routine that generates the sine samples is given in the form of a subroutine.

```
Motorola DSP56000 Macro Cross Assembler Version 1.10 06-18-87 17:19:09 sinwgli.asm Page 1
               page    132,50,0,10
    ; SINWGLI is a direct table look-up routine for SINe Wave. Generation using Linear Interpolation
    ; with any positive real delta greater than 0.0 and less than N/2.0. Here delta = 2.5 and is saved in y1 and y0.
    ; The sine table contains 256 (N) entries corresponding to the sines of 256 equally spacedangles
    ; between 0 and 360 degrees. N must be a power of two. The table size minus one, N-1, is saved in N0.
    00000060        start     equ     $60                                    ;program starting address
    00000100        sinep     equ     $100                                   ;sine table starting address
    000000FF        mask      equ     $ff                                    ;base address plus N minus one
    2.500000        delta     equ     2.5                                    ;delta = 2.5
    0000FFE0        sinea     equ     $ffe0                                  ;address of output device
    P:0060                    org     p:start
    P:0060  0506BA                    movec   #6, omr                        ;enable on-chip ROM with sine table
    P:0061  64F400            move            #sinea, r4                     ;set up output pointer
            00FFE0
    P:0063  51F400            move            #(delta @cvi(delta)),bo        ;store fraction part in b0
            400000
    P:0065  61F43A            asl     b       #sinep, r1                     ;eliminate sign bit, initialize r1
            000100
    P:0067  47F400            move            #>@cvi(delta),y1               ;integer part of Delta in y1
            000002
    P:0069  212600            move            b0,y0                          ;unsigned fractional Delta in y0
    P:0070  390013            clr     a       #0, n1                         ;initialize a and n1
    P:006A  38FF00            move            #mask, n0                      ;set up mask value
    P:006B  3A0000            move            #0, n2                         ;initialize n2
    P:006C  62F400            move            #sinep+1,r2                    ;initialize r2 with the value r1+1
            000101
    P:006E  0BF080            jsr     sineg                                  ;jump to subroutine
            000070
    P:0070  210F00  sineg     move            a0,b                           ;fraction part of ptr in b1, b0=0     (1)
    P:0071  4CE92B            lsr     b                     y:(r1+n1),x0     ;b1=signed fr. Delta, x0=sine         (2)
    P:0072  1FEA00            move            b1,x1         y:(r2+n2),b      ;sign. fract. in x1, sine in b1       (2)
    P:0073  20004C            sub     x0,b                                   ;subtr sines to obtain slope          (1)
    P:0074  1BE900            move            b,x0          y:(r1+n1),b      ;slope value to x0, sine in b1        (2)
    P:0075  2000AB            macr    x0,x1,b                                ;sample approximation = m*x+b         (1)
    P:0076  5F6430            add     y, a                  b,y:(r4)         ;update calculation, output sample    (1)
    P:0077  230500            move            n0,x1                          ;set up mask value in x1              (1)
    P:0078  200066            and     x1,a                                   ;mask a1 = modulo arithmetic          (1)
    P:0079  219900            move            a1,n1                          ;update offset register n1            (1)
    P:007A  219A00            move            a1,n2                          ;update offset register n2            (1)
    P:007B  00000C            rts                                            ;return from the subroutine           (1)
                              end                                            ;the end of the routine               (2)
0 Errors
```

Figure 4-3 *"SINWGLI" Routine*

Six address arithmetic unit registers are used in this routine; besides R1, R2, R4, N1, and N2, the modifier register M2 is used. The use of M2 is twofold. First, M2 is used to store the mask value before it is used in the AND masking operation. (This masking operation constrains R1 + N1 within the desired region in memory, i.e., implements modulo addressing.) Second, it is used to activate modulo arithmetic updates of R2 + N2. This is necessary to take care of the special case where R1 +N1 points to $1FF. Since R2 + N2 equals R1 + N1 + 1, it would point to $200 instead of $100 without modulo arithmetic.

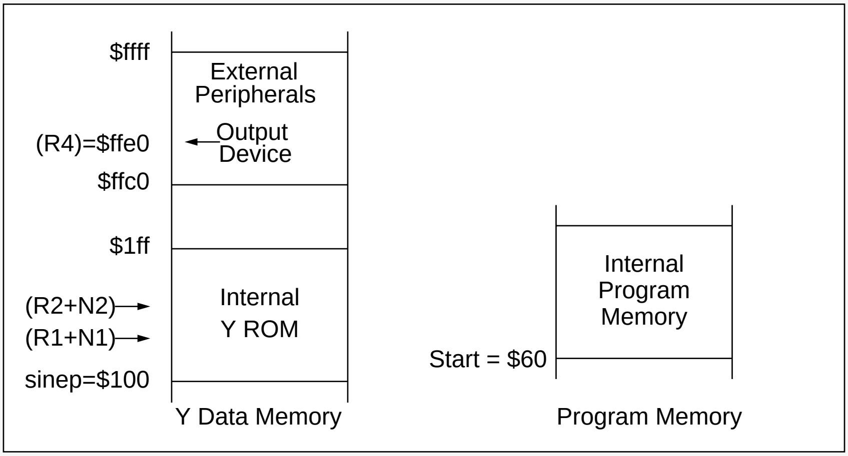

Figure 4-4 *"SINWGLI" Memory Map*

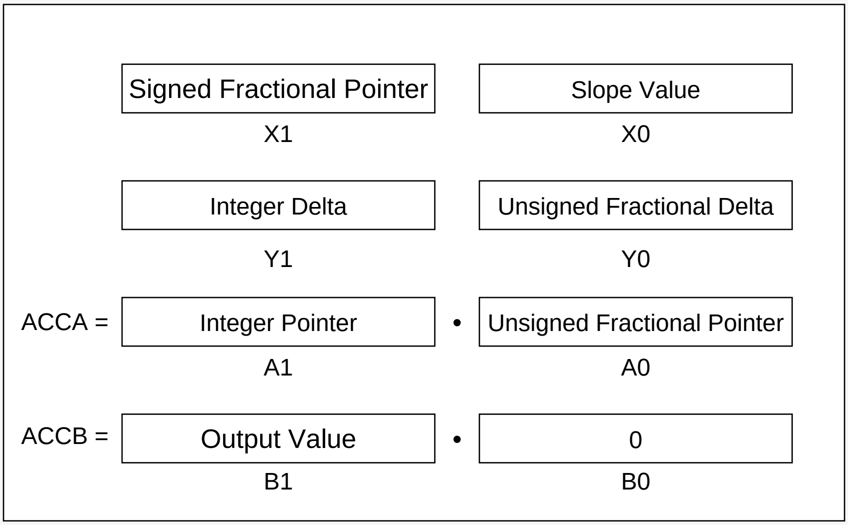

Figure 4-5 *"SINWGLI" Data ALU Programmer's Model*

Resources required for the "SINWGLI" routine:

Data ROM	256	24-bit words for the sine table
Program Memory	17	24-bit words for the initialization
	12	24-bit words for the program memory
MSF		1/(2 • 50 • 16) MHz = 0.625 MHz

4.1 Harmonic Distortion

As expected the THD improves dramatically for non-integer values of delta when using linear interpolation. This is of course because the intermediate points are better approximated. The performance for integer deltas remains the same as that for the direct table look-up methods already discussed. The results for the THD using linear interpolation are shown in Table 4-1 and these results should be directly compared with the results in Table 3-2.

Table 4-1 *Total Harmonic Distortion — Real Delta*

Δ	**N = 64**	**N = 128**	**N = 256**
2.00	$2.8356370 \bullet 10^{-15}$	$3.4157912 \bullet 10^{-15}$	$2.8659804 \bullet 10^{-15}$
2.25	$2.0443282 \bullet 10^{-07}$	$1.2762312 \bullet 10^{-08}$	$7.9741605 \bullet 10^{-10}$
2.50	$3.6316863 \bullet 10^{-07}$	$2.2683957 \bullet 10^{-08}$	$1.4175620 \bullet 10^{-09}$
2.75	$2.0443282 \bullet 10^{-07}$	$1.2762312 \bullet 10^{-08}$	$7\ 9741605 \bullet 10^{-10}$
3.00	$3.4157912 \bullet 10^{-15}$	$2.8659804 \bullet 10^{-15}$	$2.6423040 \bullet 10^{-15}$
8.25	$2.0443282 \bullet 10^{-07}$	$1.2762312 \bullet 10^{-08}$	$7.9741605 \bullet 10^{-10}$
11.625	$1.4913862 \bullet 10^{-07}$	$9.3069933 \bullet 10^{-09}$	$5.8146748 \bullet 10^{-10}$

With respect to Table 4-1 some important observations and conclusions can be drawn:

1. THD is reduced by 103 by using linear interpolation with direct table look-up. This can be verified by comparing Table 3-2 with Table 4-1. This is because an extra correctional value is added to the sine value obtained from the sine table. This correctional value is the product of the slope at the specific point and the fractional part of the pointer (i.e., m-x).

2. THD depends only on the fractional part of delta when linear interpolation is used. It is a maximum for the fractional part = 0.5 and symmetrical about this midpoint. This is evident in Table 4-1. This is because the linear approximation is poorest at the midpoint. ■

APPENDIX A

Computation of Total Harmonic Distortion (THD)

"To determine the precision needed to calculate THD, consider the inherent symmetry in the DFT."

The equation for calculating THD (given in Eqn. 3-2) can be rewritten as:

$$\text{THD} = \frac{ET - EF}{ET} \qquad \text{Eqn. A-1}$$

where: *ET = the total energy of the wave*
EF = the energy of the fundamental frequency

For accurate and correct results, the above energy terms must be calculated over a full cycle of the synthesized sine wave. In the case of direct table look-up, a full cycle may require several passes through the sine table. The number of passes depends on the value of delta used.

A full cycle will be synthesized for the smallest n for which $n \bullet \Delta$ is evenly divisible by N. For example, if the table length, N = 128, and the step size, $\Delta = 2.5 = 5/2$, then a complete cycle occurs for n = 256, since $256 \bullet 2.5/128 = 5$. Figure 3-4 illustrates an example for N = 8 and $\Delta = 2.5$. The $(2 \bullet 8 =)$ 16 points required for the THD calculation are shown on the figure.

In general, if $\Delta = A/B$, where A and B are relatively prime numbers, then the minimum number of samples, N', which must be output to synthesize a full cycle is $N' = B \bullet N$.

The total energy, ET, in a cycle of length BN ($BN = B \bullet N$) is given by:

$$ET = \sum_{i=0}^{BN-1} x(i) \cdot x(i) \qquad \text{Eqn. A-2}$$

where: x(i) *is the* i^{th} *sample of the sine-wave sequence*

The amount of energy in the fundamental frequency, EF, over the same period is given by:

$$EF = (1/BN) \bullet (|X(A)|^2 + |X(BN - A)|^2)$$
$$= (2/BN) \bullet (|X(A)|^2) \quad \text{for a real sequence}$$

Eqn. A-3

where the X(k) are terms of the Discrete Fourier Transform (DFT) defined by the following equation:

$$X(k) = \sum_{n=0}^{BN-1} x(n) e^{[-j(2 \bullet \pi / BN) \bullet n \bullet k]} \qquad \text{Eqn. A-4}$$

The x(n) values used to calculate THD in Table 3-1(a), Table 3-1(b), Table 3-2, and Table 4-1 are based on actual values computed by the DSP56001/2 for the 3 sample sine-wave generator programs described in this document. The THD computation was carried out using VAX VMS FORTRAN and by using the formulas given above with double precision floating point arithmetic(9).

The VAX VMS FORTRAN source code used for the computation of THD is given in Figure A-1.

Some important details concerning the computation of the THD should be noted. ENERG3 and (ENERG2-EFUND) ideally should have the same value. However, a difference may occur because ENERG2 is much greater than EFUND. The result of the subtraction will be inaccurate due to numerical precision limitations. To determine the precision needed to calculate THD, consider the inherent symmetry in the DFT. For a real sequence x(n), the DFT sequence, X(k), exhibits the following symmetries (10):

1. Re{X(k)} = Re{X(N - k)}
2. Im{X(k)} = - Im{X(N - k)}
3. Mag{X(k)} = Mag{X(N - k)}
4. Arg{X(k)} = Arg{X(N - k)}

where: N is the length of the DFT

Therefore, using symmetry 3 as a check for the correct results for N = 128, we should have:

Mag{X(1)} = Mag{X(127)}

Mag{X(2)} = Mag{X(126)}

•

•

•

Mag{X(n)} = Mag{X(128 - n)}

This, in turn, means that the following equalities should hold:

$$\cos(2 \bullet \pi \bullet k \bullet n / N) = \cos(2 \bullet \pi \bullet (N - k) \bullet n / N)$$
$$\sin(2 \bullet \pi \bullet k \bullet n / N) = -\sin(2 \bullet \pi \bullet (N - k) \bullet n / N)$$

Eqn. A-5

where: *n = 0,1,2, . . BN-1*

k = 0,1,2, . . BN-1

If single precision accuracy is used in calculating the THD, the above equalities do **NOT** hold. For example,

let k = 1
n = 128
ARG1 = 2 • π • n • 1/128
ARG2 = 2 • π • n • 127/128

Then, as shown in Table A-1, cos (ARG1) does not equal cos (ARG2) when truncating to only 8 digits. If, however, double precision is used, the equalities of Eqn. A-5 are satisifed, as shown in Table A-2 even when truncating to 10 digits.

Table A-1 *Single Precision Accuracy*

n	cos(ARG1)	cos(ARG2)	sin(ARG1)	-sin(ARG2)
125	0.9891766	0.9891724	0.1467302	-0.1467580
126	0.9951848	0.9951788	0.0980167	-0.0980776
127	0.9987954	0.9987938	0.0490676	-0.0491002

Table A-2 *Double Precision Accuracy*

n	cos(ARG1)	cos(ARG2)	sin(ARG1)	-sin(ARG2)
125	0.989176509	0.989176509	0.146730474	-0.146730474
126	0.995147266	0.995147266	0.098017140	-0.098017140
127	0.998795456	0.998795456	0.049067674	-0.049067674

```
C   Program THD.FOR Using double precision real arithmetic
C   Program to calculate THD for digital sine-wave generation without interpolation.

    IMPLICIT COMPLEX*16(X)
    REAL*8 ENERGl,ENERG2,ENERG3,S(2048),TWOPI,ISS(2048)
    DIMENSION X(0:2048)
    INTEGER*4 IS(2048)
    TYPE *,' ENTER TABLE SIZE (UP TO 2048)'
    ACCEPT *,NN
    TYPE *,'ENTER INPUT HEX FILENAME'
C   the hex values are assumed to be 11 on each line, each consisting of 6 characters C
and having a blank space in between them. This is the format of the "LOD" file
C   that is generated by the assembler [5].
    READ(1,110)(IS(I),I=l,NN)
110 FORMAT(ll(Z6,X))
    DO 120 I=l,NN

    IF(BTTEST(IS(I),23)) THEN
         IS(I)=JIOR(IS(I),'FF000000'X)
    ENDIF
120 CONTINUE

    DO 100 I=2,NN
    ISS(I)=IS(I)
100 CONTINUE
    TYPE *,' ENTER DELTA, BN, A'
    ACCEPT *,DELTA,BN,A
    AINDEX=0.0
    DO 200 IABC=l,BN
    INDEX=INT(AINDEX)
    S(IABC)=DBLE(ISS(INDEX+1))
    AINDEX=AINDEX+DELTA
    IF (AINDEX.GT.FLOAT(N)) AINDEX=AINDEX-FLOAT(N)
200 CONTINUE
    TWOPI=8.0*DATAN(l.D0)
    DO 400 IK=0,BN-l
    XSUM=DCMPLX(0.D0,OD0)
    DO 300 IN=0,BN-l
    XARG=DCMPLX(0.D0,-TWOPI*DBLE(IN)*DBLE(IK)/DBLE(BN))
300 XSUM=XSUM+DCMPLX(S(IN+1),0.D0)*CDEXP(XARG))
    X(IK)=XSUM
400 CONTINUE
    DO 500 IK=0,BN-l
500 ENERGl=ENERGl+CDABS(X(IK))**2.D0)
    ENERGl=ENERGl/DBLE(BN)
    DO 505 IR=0,BN-l
    IF((IK.EQ.(A)).OR.(IK.EQ.(BN-INT(A)))) GOTO 505
    ENERG3=ENERG3+(CDABS(X(IK))**2.D0)
505 CONTINUE
    ENERG3=ENERG3/DBLE(BN)
    energ3 should be the same as EFUND
C   DO 600 I=l,BN
600 ENERG2=ENERG2+(S(I)*S(I))
    TYPE *,' ENERGl=',ENERGl,' ENERG2=',ENERG2,'
    $ENERG3=',ENERG3
    EFUND=((CDABS(X(A))**2)+(CDABS(X(BN-A))**2))/DBLE(BN)
    TYPE *,' THD=',ENERG3/ENERG2,'EFUND=',EFUND
    END
```

Figure A-1 *VAX VMS FORTRAN Source Code for THD Computation*

APPENDIX B

The Sine Table

Table B-1 gives the sine entries for N = 256. The hexadecimal values are given in the signed fractional format used for the DSP56001. "$" denotes a hexadecimal value.

Table B-1 *Sine Table*

ADDRESS	VALUE
S_01	$03242B
S_02	$0647D9
S_03	$096A90
S_04	$0C8BD3
S_05	$0FAB27
S_06	$12C810
S_07	$15E214
S_08	$18F8B8
S_09	$1C0B82
S_0A	$1F19F9
S_0B	$2223A5
S_0C	$25280C
S_0D	$2826B9
S_0E	$2B1F35
S_0F	$2E110A
S_10	$30FBC5
S_11	$33DEF3
S_12	$36BA20
S_13	$398CDD
S_14	$3C56BA
S_15	$3F174A
S_16	$41CE1E
S_17	$447ACD
S_18	$471CED
S_19	$49B415
S 1A	$4C3FE0
S_1B	$4EBFE9
S_1C	$5133CD
S_1D	$539B2B
S_1E	$55F5A5
S_1F	$5842DD
S_20	$5A827A
S_21	$5CB421
S_22	$5ED77D

Table B-1 *Sine Table (continued)*

ADDRESS	VALUE
S_23	$60EC38
S_24	$62F202
S_25	$64E889
S_26	$66CF81
S_27	$68A69F
S_28	$6A6D99
S_29	$6C2429
S_2A	$6DCA0D
S_2B	$6F5F03
S_2C	$70E2CC
S_2D	$72552D
S_2E	$73B5EC
S_2F	$7504D3
S_30	$7641AF
S_31	$776C4F
S_32	$788484
S_33	$798A24
S_34	$7A7D05
S_35	$7B5D04
S_36	$7C29FC
S_37	$7CE3CF
S_38	$7D8A5F
S_39	$7E1D94
S_3A	$7E9D56
S_3B	$7F0992

ADDRESS	VALUE
S_3C	$7F6237
S_3D	$7FA737
S_3E	$7FD888
S_3F	$7FF622
S_40	$7FFFFF
S_41	$7FF622
S_42	$7FD888
S_43	$7FA737
S_44	$7F6237
S_45	$7F0992
S_46	$7E9D56
S_47	$7E1D94
S_48	$7D8A5F
S_49	$7CE3CF
S_4A	$7C29FC
S_4B	$7B5D04
S_4C	$7A7D05
S_4D	$798A24
S_4E	$788484
S_4F	$776C4F
S_50	$7641AF
S_51	$7504D3
S_52	$73B5EC
S_53	$72552D
S_54	$70E2CC

Table B-1 Sine Table *(continued)*

ADDRESS	VALUE
S_55	$6F5F03
S_56	$6DCA0D
S_57	$6C2429
S_58	$6A6D99
S_59	$68A69F
S_5A	$66CF81
S_5B	$64E889
S_5C	$62F202
S_5D	$60EC38
S_5E	$5ED77D
S_5F	$5CB421
S_60	$5A827A
S_61	$5842DD
S_62	$55F5A5
S_63	$539B2B
S_64	$5133CD
S_65	$4EBFE9
S_66	$4C3FE0
S_67	$49B415
S_68	$471CED
S_69	$447ACD
S_6A	$41CE1E
S_6B	$3F174A
S_6C	$3C56BA
S_6D	$398CDD

ADDRESS	VALUE
S_6E	$36BA20
S_6F	$33DEF3
S_70	$30FBC5
S_71	$2E110A
S_72	$2B1F35
S_73	$2826B9
S_74	$25280C
S_75	$2223A5
S_76	$1F19F9
S_77	$1C0B82
S_78	$18F8B8
S_79	$15E214
S_7A	$12C810
S_7B	$0FABD
S_7C	$0C8BD3
S_7D	$096A90
S_7E	$0647D9
S_7F	$03242B
S_80	$000000
S_81	$FCDBD5
S_82	$F9B827
S_83	$F69570
S_84	$F3742D
S_85	$F054D9
S_86	$ED37F0

Table B-1 *Sine Table (continued)*

ADDRESS	VALUE
S_87	$EA1DEC
S_88	$E70748
S_89	$E3F47E
S_8A	$E0E607
S_8B	$DDDC5B
S_8C	$DAD7F4
S_8D	$D7D947
S_8E	$D4E0CB
S_8F	$D1EEF6
S_90	$CF043B
S_91	$CC210D
S_92	$C945E0
S_93	$C67323
S_94	$C3A946
S_95	$C0E8B6
S_96	$BE31E2
S_97	$BB8533
S_98	$B8E313
S_99	$B64BEB
S_9A	$B3C020
S_9B	$B14017
S_9C	$AECC33
S_9D	$AC64D5
S_9E	$AA0A5B
S_9F	$A7BD23

ADDRESS	VALUE
S_A0	$A57D86
S_A1	$A34BDF
S_A2	$A12883
S_A3	$9F13C8
S_A4	$9D0DFE
S_A5	$9B1777
S_A6	$99307F
S_A7	$975961
S_A8	$959267
S_A9	$93DBD7
S_AA	$9235F3
S_AB	$90A0FD
S_AC	$8F1D34
S_AD	$8DAAD3
S_AE	$8C4A14
S_AF	$8AFB2D
S_B0	$89BE51
S_B1	$8893B1
S_B2	$877B7C
S_B3	$8675DC
S_B4	$8582FB
S_B5	$84A2FC
S_B6	$83D604
S_B7	$831C31
S_B8	$8275A1

Table B-1 *Sine Table (continued)*

ADDRESS	VALUE
S_B9	$81E26C
S_BA	$8162AA
S_BB	$80F66E
S_BC	$809DC9
S_BD	$8058C9
S_BE	$802778
S_BF	$8009DE
S_C0	$800000
S_C1	$8009DE
S_C2	$802778
S_C3	$8058C9
S_C4	$809DC9
S_C5	$80F66E
S_C6	$8162AA
S_C7	$81E26C
S_C8	$8275A1
S_C9	$831C31
S_CA	$83D604
S_CB	$84A2FC
S_CC	$8582FB
S_CD	$8675DC
S_CE	$877B7C
S_CF	$8893B1
S_D0	$89BE51
S_D1	$8AFB2D

ADDRESS	VALUE
S_D2	$8C4A14
S_D3	$8DAAD3
S_D4	$8F1D34
S_D5	$90A0FD
S_D6	$9235F3
S_D7	$93DBD7
S_D8	$959267
S_D9	$975961
S_DA	$99307F
S_DB	$9B1777
S_DC	$9D0DFE
S_DD	$9F13C8
S_DE	$A12883
S_DF	$A34BDF
S_E0	$A57D86
S_E1	$A7BD23
S_E2	$AA0A5B
S_E3	$AC64D5
S_E4	$AECC33
S_E5	$B14017
S_E6	$B3C020
S_E7	$B64BEB
S_E8	$B8E313
S_E9	$BB8533
S_EA	$BE31E2

Table B-1 *Sine Table (continued)*

ADDRESS	VALUE
S_EB	$C0E8B6
S_EC	$C3A946
S_ED	$C67323
S_EE	$C945E0
S_EF	$CC210D
S_F0	$CF043B
S_F1	$D1EEF6
S_F2	$D4E0CB
S_F3	$D7D947
S_F4	$DAD7F4
S_F5	$DDDC5B
S_F6	$E0E607
S_F7	$E3F47E
S_F8	$E70748
S_F9	$EA1DEC
S_FA	$ED37F0
S_FB	$F054D9
S_FC	$F3742D
S_FD	$F69570
S_FE	$F9B827
S_FF	$FCDBD5

REFERENCES

1. J. Tierney, “Digital Frequency Synthesizers”, Chapter V of *Frequency Synthesis: Techniques and Applications*, J. Gorski-Popiel, ed., N.Y., IEEE Press, 1975.

2. T. F. Quatieri, R. J. McAulay, “Speech Transformations Based on a Sinusoidal Representation”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-34, 1449-1464, December 1986.

3. D. L. Duttweiler and D. G. Messerschmitt, “Analysis of Digitally Generated Sinusoids with Application to A/D and D/A Converter Testing”, *IEEE Transactions on Communications*, vol. COM-26, pp. 669-675, May 1978.

4. S. Mehrgardt, “Noise Spectra of Digital Sine Generators Using the Table-Lookup Method”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP31, pp. 1037-1039, August 1983.

5. J. Tierney, C. M. Rader, and B. Gold, “A Digital Frequency Synthesizer”, *IEEE Transactions on Audio and Electroacoustics*, vol. AU-19, pp. 48-56, March 1971.

6. *DSP56000 Macro Assembler Reference Manual*, Motorola Inc., 1986.

7. Texas Instruments Application Note, *Precision Digital Sine-Wave Generation with the TMS32010*, February 1984.

8. *DSP56000 Digital Signal Processor User's Manual*, Motorola Inc., 1986.

9. D. Borth, *Motorola Internal Memo on Digital Signal Processing*, June 1986.

10. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, 1975.