



***System Functions for  
Look-Up Tables***

*Antrim<sup>®</sup>-A/MS*

*December 2000*

---

© 1998-2000 Antrim Design Systems, Inc., 5550 Scotts Valley Drive, Scotts Valley, California 95066

All rights reserved. This document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this document may be reproduced in any form by any means without prior written authorization of Antrim Design Systems, Inc., and its licensors, if any.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

<p>The algorithm for numerical computation of the roots of a polynomial with complex coefficients is copyright D. Bini, Dipartimento di Matematica, Universita' di Pisa:</p> <p>NUMERICAL COMPUTATION OF THE ROOTS OF A POLYNOMIAL HAVING COMPLEX COEFFICIENTS, BASED ON ABERTH'S METHOD. Version 1.4, June 1996 D. Bini, Dipartimento di Matematica, Universita' di Pisa (bini@dm.unipi.it)</p> <p>*****</p> <p>All the [Numerical Computation of Roots] software contained in this library is protected by copyright. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.</p> <p>*****</p>
--

**TRADEMARKS:**

Antrim Design Systems and the Antrim logo are registered trademarks of Antrim Design Systems, Inc., in the United States, filed in Europe, Japan, and Asia, and may be protected as trademarks in other countries.

SPICE is a registered trademark of the Regents of the University of California, Berkeley. Parts of the Antrim product line are based on SPICE 3F5 with BSIM3v3 developed at the University of California, Berkeley, and copyrighted by the University of California. PCCTS (Purdue Compiler Construction Tool) is a third-party, public-domain product that was used in developing Antrim's Analog Artist Integration product. Verilog is a registered trademark of Cadence Design Systems, Inc. Verilog-A/MS is a registered trademark of Cadence Design Systems, Inc., and licensed to Open Verilog International and Antrim Design Systems, Inc. Undertow is a trademark of Veritools. MATLAB is a registered trademark of The MathWorks, Inc. SmartModel, SmartCircuit, and SWIFT are trademarks of Synopsys, Inc.

All other product, service, or company names mentioned herein are claimed as trademarks and trade names by their respective companies.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN, THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. ANTRIM DESIGN SYSTEMS, INC., MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Printed in the United States of America

[Rev. 2000.12.15]

# System Functions for Look-Up Tables

---



---

The proposed \$table system functions are used to manipulate look-up tables that depend on 1, 2, or 3 independent variables or parameter arrays (Table 1). The data is extracted either from ASCII files or from parameter arrays and is interpolated to model continuous behavior. Each system function operates in DC, AC, and transient analyses, and returns a single real value.

**Table 1: \$table System Functions**

System Functions	Argument(s)	Models continuous behavior from a table of data based on:
\$table1f()	<i>y_var, file, method</i>	a single independent variable <b>(see page 4)</b>
\$table2f()	<i>x_var, y_var, file, method</i>	two independent variables <b>(see page 6)</b>
\$table3f()	<i>w_var, x_var, y_var, file, method</i>	three independent variables <b>(see page 8)</b>
\$table1a()	<i>y, y_array, f_array, method</i>	an independent variable and a parameter array <b>(see page 11)</b>
\$table2a()	<i>x, y, x_array, y_array, f_array, method</i>	two independent variables and two parameter arrays <b>(see page 13)</b>
\$table3a()	<i>w, x, y, w_array, x_array, y_array, f_array, method</i>	three independent variables and three parameter arrays <b>(see page 15)</b>

---

For those system functions that reference a file name (*file*), the simulator searches the following locations in the order shown for the file containing the tabular data:

1. current directory
2. directories specified using the `+incdir+` named argument for the simulator command

**Note** – Using the `+incdir+` named argument, the location of the data files can remain fixed while the project directory changes.

### `$table1f()` System Function

The general form for the `$table1f()` system function is:

```
$table1f(y_var, "file", method);
```

where the arguments and their meanings are as follows:

Argument	Description
<i>y_var</i>	independent variable
<i>file</i>	name of ASCII file containing tabular data
<i>method</i>	integer flag for interpolation method: 1=linear 3=natural cubic spline

The data in *file* consists of pairs of values, one pair per line, up to a maximum of 10,000 lines (pairs). The first number in the pair is the value of *y\_var*. The second number in the pair is the output value,  $f(y\_var)$ . The data does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced); also, the data points in the file may be freely formatted. The independent variable, *y\_var*, can be either increasing or decreasing, but must be monotonic.

**Note** – If the *y\_var* values are *not* monotonic, the program will automatically sort them such that they are.

## System Functions for Look-Up Tables

The interpolation method (*method*) is used to connect the points.

**Note** – If the cubic spline method is specified (i.e., *method* = 3), the interpolation method reverts to linear (i.e., *method* = 1) wherever the input variable exceeds the range of the table (as defined in the data file, *file*) because a natural cubic spline has unpredictable behavior outside its range.

Here is an example using `$tablelf()` in the context of a voltage-controlled oscillator (VCO):

```
module vco1(vin, vout, t1);
  inout vin, vout, t1;
  electrical vin, vout, t1;
  parameter real amp = 1.0;
  parameter real center_freq = 2.5e3;
  parameter real vco_gain = 1.0e4;
  real phase;

  analog begin
    V(t1) <+ $tablelf(V(vin), "inp.dat", 3);
    phase = idt(center_freq + vco_gain*V(t1));
    V(vout) <+ amp*abs(sin(6.28*phase));
  end
endmodule
```

`V(t1)` depends on `V(in)` according to the data listed in the file `inp.dat`. The data is interpolated using a natural cubic spline. The data in `inp.dat` might look like this (without the heading row):

<i>y_var</i>	<i>f(y_var)</i>	
0.0	1.0	← <i>first pair</i> : <code>V(in)</code> <code>V(t1)</code>
1.0	1.0	← <i>second pair</i>
2.0	1.1	← <i>third pair</i>
3.0	2.0	← <i>fourth pair</i>
3.5	4.0	← <i>fifth pair</i>
5.0	4.9	← <i>sixth pair</i>
6.1	5.0	← <i>seventh pair</i>

The first pair says, when `V(in)` is 0.0, `V(t1)` is 1.0; the second pair says, when `V(in)` is 1.0, `V(t1)` is also 1.0; and so on.

---

## \$table2f() System Function

The general form for the \$table2f() system function is:

```
$table2f(x_var, y_var, "file", method);
```

where the arguments and their meanings are as follows:

Argument	Description
<i>x_var</i>	first independent variable
<i>y_var</i>	second independent variable
<i>file</i>	name of ASCII file containing tabular data
<i>method</i>	integer flag for interpolation method: 1=linear 2=spline in <i>y_var</i> , linear in <i>x_var</i> 3=spline in <i>x_var</i> , linear in <i>y_var</i> 4=spline in both <i>x_var</i> and <i>y_var</i>

The data in *file* consists of triplets of values, one triplet per line, up to a maximum of 1e8 lines (triplets). The first number in the triplet is the value of *x\_var*. The second number in the triplet is the value of *y\_var*. The third number in the triplet is the output value,  $f(x\_var, y\_var)$ . The data does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced); also, the data points in the file may be freely formatted. Each axis value may be either increasing or decreasing, but must be monotonic.

<b>Note</b> – If the axis values are <i>not</i> monotonic, the program will automatically sort them such that they are.
---

The interpolation method (*method*) is used to connect the points. Points with the same *x\_var* form a column. Each column must contain at least two points (i.e., there must be two *y\_var* values for each *x\_var* column). The number and spacing of the *y\_var* points along each column may vary.

Here is an example using \$table2f() in the context of a voltage-controlled oscillator (VCO):

```
module vco2(in, out, vdd, t2);  
  in in, vdd;  
  out out;  
  inout t2;  
  electrical in, out, vdd, t2;  
  parameter real amp=1.0,center_freq=2.5e3,
```

## System Functions for Look-Up Tables

```
vco_gain=1.0e4;
real phase;

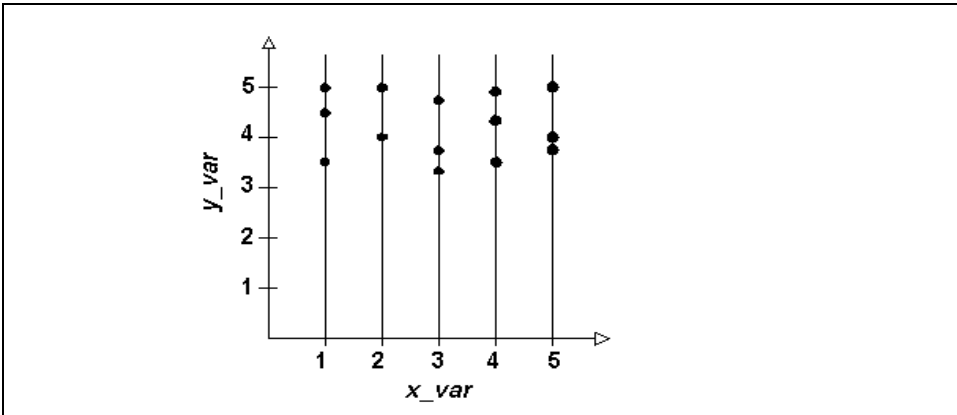
analog begin
    V(t2) <+ $table2f(V(in), V(vdd), "inp.dat", 1);
    phase = idt(center_freq + vco_gain*V(t2));
    V(out) <+ amp*abs(sin(`M_TWO_PI*phase));
end
endmodule
```

V(t2) depends on V(in) and V(vdd) according to the data listed in the file inp.dat. This example uses linear interpolation. The data in inp.dat might look like this (without the heading row):

x_var	y_var	f(x_var, y_var)	
1	3.5	1e6	} first column
1	4.5	1.1e6	
1	5	1.2e6	
2	4	2e6	} second column
2	5	2.1e6	
3	3.3	3e6	} third column
3	3.8	3.1e6	
3	4.8	3.2e6	
4	3.5	3.5e6	} fourth column
4	4.3	3.6e6	
4	4.9	3.7e6	
5	3.8	4e6	} fifth column
5	4	4.1e6	
5	5	4.2e6	

The data set above defines five columns; all except the second column contain three y\_var points (Figure 2-1013). The values for both x\_var and y\_var are increasing and monotonic.

**Note** – The number of y\_var points in each column may vary, as long as each column contains at least two y\_var data points.



**Figure 2-10: Sample Distribution of Columns and Points for `$table2f()`**

### **`$table3f()` System Function**

The general form for the `$table3f()` system function is:

```
$table3f(w_var, x_var, y_var, "file", method);
```

where the arguments and their meanings are as follows:

Argument	Description
<i>w_var</i>	first independent
<i>x_var</i>	second independent variable
<i>y_var</i>	third independent variable
<i>file</i>	name of ASCII file containing tabular data
<i>method</i>	integer flag for interpolation method: 1=linear 2=spline in <i>y_var</i> , linear in <i>x_var</i> , <i>w_var</i> 3=spline in <i>x_var</i> , linear in <i>y_var</i> , <i>w_var</i> 4=spline in <i>x_var</i> and <i>y_var</i> , linear in <i>w_var</i>

The data in *file* consists of quadruplets (set of four) of values, one quadruplet per line, up to a maximum of 1e8 lines (quadruplets). The first number in the quadruplet is the value of *w\_var*. The second number is the value of *x\_var*. The third number is the value of *y\_var*. The fourth number is the output value,  $f(w\_var, y\_var, x\_var)$ . The data does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced); also, the data points in the file may be freely

## System Functions for Look-Up Tables

---

formatted. Each axis value may be either increasing or decreasing, but must be monotonic.

**Note** – If the axis values are *not* monotonic, the program will automatically sort them such that they are.

The interpolation method (*method*) is used to connect the points. Points with the same *w\_var* form planes. Points with the same *x\_var* and *w\_var* form columns which are parallel to *y\_var*. All planes must have the same number of columns, and the *x\_var* of the columns in each plane must match. There must be at least two planes. The interpolation method (*method*) is used to connect the points.

Here is an example using `$table3f()` in the context of a voltage-controlled oscillator (VCO):

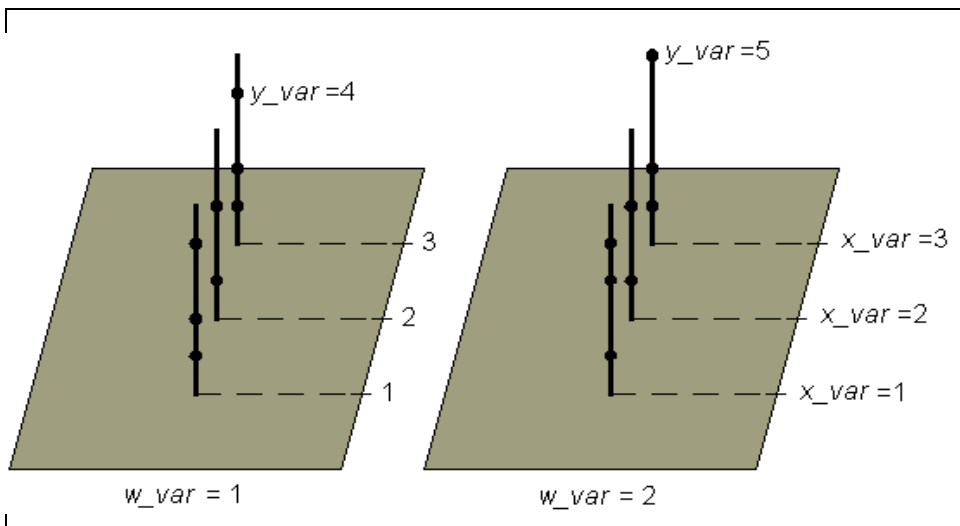
```
module vco3(in, out, vdd, plane, t3);
    in in, vdd, plane;
    out out;
    inout t3;
    electrical in, out, vdd, plane, t3;
    parameter real amp=1.0,center_freq=2.5e3,
        vco_gain=1.0e4;
    real phase;

    analog begin
        V(t3) <+ $table3f(V(in), V(vdd), V(plane),
            "inp.dat", 1);
        phase = idt(center_freq + vco_gain*V(t3));
        V(out) <+ amp*abs(sin(`M_TWO_PI*phase));
    end
endmodule
```

$V(t3)$  depends on  $V(in)$ ,  $V(vdd)$ , and  $V(plane)$  according to the data listed in the file `inp.dat`. This example uses linear interpolation. The data in `inp.dat` might look like this (without the heading row):

<i>w_var</i>	<i>x_var</i>	<i>y_var</i>	<i>f(w_var, x_var, y_var)</i>
1	1	1	3
1	1	2	5
1	1	4	6
1	2	1	7
1	2	3	5
1	3	1	4
1	3	2	6
1	3	4	7
2	1	1	4
2	1	3	6
2	1	4	7
2	2	1	8
2	2	3	6
2	3	1	5
2	3	2	7
2	3	5	8

For the above data, there are two planes: one corresponding to  $w\_var=1$  and the other corresponding to  $w\_var=2$ . Each plane contains three columns, defined by  $x\_var=1, 2,$  and  $3$  for each  $w\_var$  (Figure 2-1114).



**Figure 2-11: Sample Planes and Columns for `$table3f()`**

### **`$table1a()` System Function**

The general form for the `$table1a()` system function is:

```
$table1a(y_var, y_array, f_array, method);
```

where the arguments and their meanings are as follows:

Argument	Description
<code>y_var</code>	real variable used to determine value in <code>y_array</code>
<code>y_array</code>	parameter array of real values
<code>f_array</code>	parameter array of real values representing $f(y\_array)$
<code>method</code>	integer flag for interpolation method: 1=linear 3=natural cubic spline

The real variable `y_var` is used to interpolate a value from the parameter array `y_array`. The interpolated real value from the parameter array `y_array` is then used to determine the corresponding [interpolated] result from the real values in the parameter array `f_array`, which is  $f(y\_array)$ .

---

The data defined in the arrays does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced). The real values in *y\_array* and *f\_array* can be either increasing or decreasing, but must be monotonic.

**Note** – If the *y\_array* or *f\_array* values are *not* monotonic, the program will automatically sort them such that they are.

The interpolation method (*method*) is used to connect the points.

**Note** – If the cubic spline method is specified (i.e., *method* = 3), the interpolation method reverts to linear (i.e., *method* = 1) wherever the input variable exceeds the range of the table (as defined in the parameter arrays *y\_array* and *f\_array*) because a natural cubic spline has unpredictable behavior outside its range.

Compare the following example using `$table1a()` in the context of a voltage-controlled oscillator (VCO) with the similar example using `$table1f()` (see `$table1f()` *System Function* on page 4):

```
module vco1(vin, vout, t1);
  inout vin, vout, t1;
  electrical vin, vout, t1;
  parameter real amp = 1.0;
  parameter real center_freq = 2.5e3;
  parameter real vco_gain = 1.0e4;
  real phase;

  parameter real y1[0:6]=
    {0.0, 1.0, 2.0, 3.0, 3.5, 5.0, 6.1};
  parameter real d1[0:6]=
    {1.0, 1.0, 1.1, 2.0, 4.0, 4.9, 5.0};

  analog begin
    V(t1) <+ $table1a(V(vin),y1,d1,3);
    phase = idt(center_freq + vco_gain*V(t1));
    V(vout) <+ amp*abs(sin(6.28*phase));
  end
endmodule
```

### \$table2a() System Function

The general form for the \$table2a() system function is:

```
$table2a(x, y, x_array, y_array, f_array, method);
```

where the arguments and their meanings are as follows:

Argument	Description
<i>x</i>	real variable used to determine value in <i>x_array</i>
<i>y</i>	real variable used to determine value in <i>y_array</i>
<i>x_array</i>	first parameter array of real values
<i>y_array</i>	second parameter array of real values
<i>f_array</i>	parameter array of real values representing $f(x\_array, y\_array)$
<i>method</i>	integer flag for interpolation method: 1=linear 2=spline in <i>y_array</i> , linear in <i>x_array</i> 3=spline in <i>x_array</i> , linear in <i>y_array</i> 4=spline in both <i>x_array</i> and <i>y_array</i>

The real variables *x* and *y* are used to interpolate values from the parameter arrays *x\_array* and *y\_array*. The interpolated real values from those parameter arrays are then used to determine the corresponding [interpolated] result from the real values in the parameter array *f\_array*, which is  $f(x\_array, y\_array)$ .

The data defined in the arrays does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced). Each axis value may be either increasing or decreasing, but must be monotonic.

**Note** – If the axis values are *not* monotonic, the program will automatically sort them such that they are.

The interpolation method (*method*) is used to connect the points. Points with the same *x\_array* value form a column. Each column must contain at least two points (i.e., there must be two *y\_array* values for each *x\_array* column). The number and spacing of the *y\_array* points along each column may vary.

Compare the following example using \$table2a() in the context of a voltage-controlled oscillator (VCO) with the similar example using \$table2f() (see \$table2f() System Function on page 6):

```
module vco2(in, out, vdd, t2);
```

---

```

in in, vdd;
out out;
inout t2;
electrical in, out, vdd, t2;
parameter real amp=1.0,center_freq=2.5e3,
    vco_gain=1.0e4;
real phase;

parameter real x2[0:13]={1.0, 1.0, 1.0,
                        2.0, 2.0,
                        3.0, 3.0, 3.0,
                        4.0, 4.0, 4.0,
                        5.0, 5.0, 5.0};
parameter real y2[0:13]={3.5, 4.5, 5.0,
                        4.0, 5.0,
                        3.3, 3.8, 4.8,
                        3.5, 4.3, 4.9,
                        3.8, 4.0, 5.0};
parameter real d2[0:13]={1.0e6, 1.1e6, 1.2e6,
                        2.0e6, 2.1e6,
                        3.0e6, 3.1e6, 3.2e6,
                        3.5e6, 3.6e6, 3.7e6,
                        4.0e6, 4.1e6, 4.2e6};

analog begin
    V(t2) <+ $table2a(V(in),V(vdd), x2,y2,d2, 1);
    phase = idt(center_freq + vco_gain*V(t2));
    V(out) <+ amp*abs(sin(`M_TWO_PI*phase));
end
endmodule

```

### `$table3a()` System Function

The general form for the `$table3a()` system function is:

```
$table3a(w, x, y, w_array, x_array, y_array, f_array,
         method);
```

where the arguments and their meanings are as follows:

Argument	Description
<code>w</code>	real variable used to determine value in <code>w_array</code>
<code>x</code>	real variable used to determine value in <code>x_array</code>
<code>y</code>	real variable used to determine value in <code>y_array</code>
<code>w_array</code>	first parameter array of real values
<code>x_array</code>	second parameter array of real values
<code>y_array</code>	third parameter array of real values
<code>f_array</code>	parameter array of real values representing $f(w\_array, x\_array, y\_array)$
<code>method</code>	integer flag for interpolation method: 1=linear 2=spline in <code>y_array</code> , linear in <code>x_array</code> , <code>w_array</code> 3=spline in <code>x_array</code> , linear in <code>y_array</code> , <code>w_array</code> 4=spline in <code>x_array</code> and <code>y_array</code> , linear in <code>w_array</code>

The data in consists of quadruplets (set of four) of values, one quadruplet per line, up to a maximum of 1e8 lines (quadruplets). The first number in the quadruplet is the value of `w_var`. The second number is the value of `x_var`. The third number is the value of `y_var`. The fourth number is the output value,  $f(w\_var, y\_var, x\_var)$ .

The data defined in the arrays does *not* have to adhere to a uniform grid (i.e., it can be nonuniformly spaced). Each axis value may be either increasing or decreasing, but must be monotonic.

**Note** – If the axis values are *not* monotonic, the program will automatically sort them such that they are.

The interpolation method (`method`) is used to connect the points. Points with the same `w_array` values form planes. Points with the same `x_array` values and `w_array` values form columns which are parallel to `y_array`. All planes must have the same number of columns, and the `x_array` value of the columns in

---

each plane must match. There must be at least two planes. The interpolation method (*method*) is used to connect the points.

Compare the following example using `$table3a()` in the context of a voltage-controlled oscillator (VCO) with the similar example using `$table3f()` (see `$table3f()` *System Function* on page 8):

```
module vco3(in, out, vdd, plane, t3);
  in in, vdd, plane;
  out out;
  inout t3;
  electrical in, out, vdd, plane, t3;
  parameter real amp=1.0,center_freq=2.5e3,
    vco_gain=1.0e4;
  real phase;

  parameter real w3[0:15]={1.0, 1.0, 1.0, 1.0,
                           1.0, 1.0, 1.0, 1.0,
                           2.0, 2.0, 2.0, 2.0,
                           2.0, 2.0, 2.0, 2.0};

  parameter real x3[0:15]={1.0, 1.0, 1.0, 2.0,
                           2.0, 3.0, 3.0, 3.0,
                           1.0, 1.0, 1.0, 2.0,
                           2.0, 3.0, 3.0, 3.0};

  parameter real y3[0:15]={1.0, 2.0, 4.0, 1.0,
                           3.0, 1.0, 2.0, 4.0,
                           1.0, 3.0, 4.0, 1.0,
                           3.0, 1.0, 2.0, 5.0};

  parameter real d3[0:15]={3.0, 5.0, 6.0, 7.0,
                           5.0, 4.0, 6.0, 7.0,
                           4.0, 6.0, 7.0, 8.0,
                           6.0, 5.0, 7.0, 8.0};

  analog begin
    V(t3) <+ $table3a(V(in), V(vdd), V(plane),
                    w3, x3, d3, 1);
    phase = idt(center_freq + vco_gain*V(t3));
    V(out) <+ amp*abs(sin(`M_TWO_PI*phase));
  end
endmodule
```