

Application Note BD-02

Using the Analog Waveform Tool as a General Purpose Graphing Utility The V-PLOT User Debug Command

The Analog Waveform Tool (AWT) has been developed by Credence Systems as a utility for generating and analyzing waveforms for use with the Quartet, Duo and SC212 test systems. It can be extremely useful, however, as a general purpose graphing utility for use with various types of test data. The AWT can replace vt_analyze, shmoo callback, shmoo, and post-processing of script-on datalogs.

This Application Note describes the use of the AWT to graph data, the file format of the AWAV file, and the use of VPLOT, a user debug command that does automatic graphing of Vdd vs. Idd data.

SHMOO CALLBACK FUNCTIONS

In the past, users who wished to plot the relationship between Vdd and Idd, or in the general case, a tester *parameter* vs. a tester *measurement*, would use a routine known as a shmoo callback. A shmoo callback employs a user written function to determine the character plotted by the shmoo software under the desired test conditions or results.

There are three (admittedly minor) drawbacks to this approach:

shmoo callback functions have several rules that must be followed - a lengthy, specific parameter list; routines to handle the first and last iterations; a return value corresponding to the character to be printed by shmoo.

a shmoo will run (x * y) iterations. Often, when plotting a tester measurement the user is only interested in the border, or “frontier”, of a V-I characteristic. A plot with (x * y) iterations using a resource such as a DPS can be extremely slow unless code is specifically written to stop testing after data is collected about the region of interest.

The shmoo plot display is a fairly simple ASCII plot. The AWT is capable of displaying data of greater subtlety.

THE AWT AS A GENERAL PURPOSE GRAPHING UTILITY

We can employ the “user debug command” technique and knowledge about the AWAV file format to take advantage of the AWT’s graphing capability. It is possible

to write a procedure that can be invoked from the Quartet/Duo debug environment that generates test data and puts it in the AWAV format. Then this data can be read in to the AWT and analyzed and printed.

The data is capable of conveying a great amount of subtlety, restricted only by the number of samples the user is willing to take - usually just a matter of time. In general, this will work much faster than the traditional shmoo plot as there is no display overhead associated with the initial data capture. In addition, most test engineers find adding a user debug command fairly simple in comparison with writing a shmoo callback. Lastly, a user debug command can be put in a library and invoked from any test program that initializes the pointer to it (see the attached example).

THE AWAV FILE FORMAT

The basic format of an AWAV file is simply a list of floating point numbers corresponding to y-axis values. Their position in the file (e.g. first, second, third data point) corresponds to the x-axis value. In addition, there is a header that must be present, consisting of:

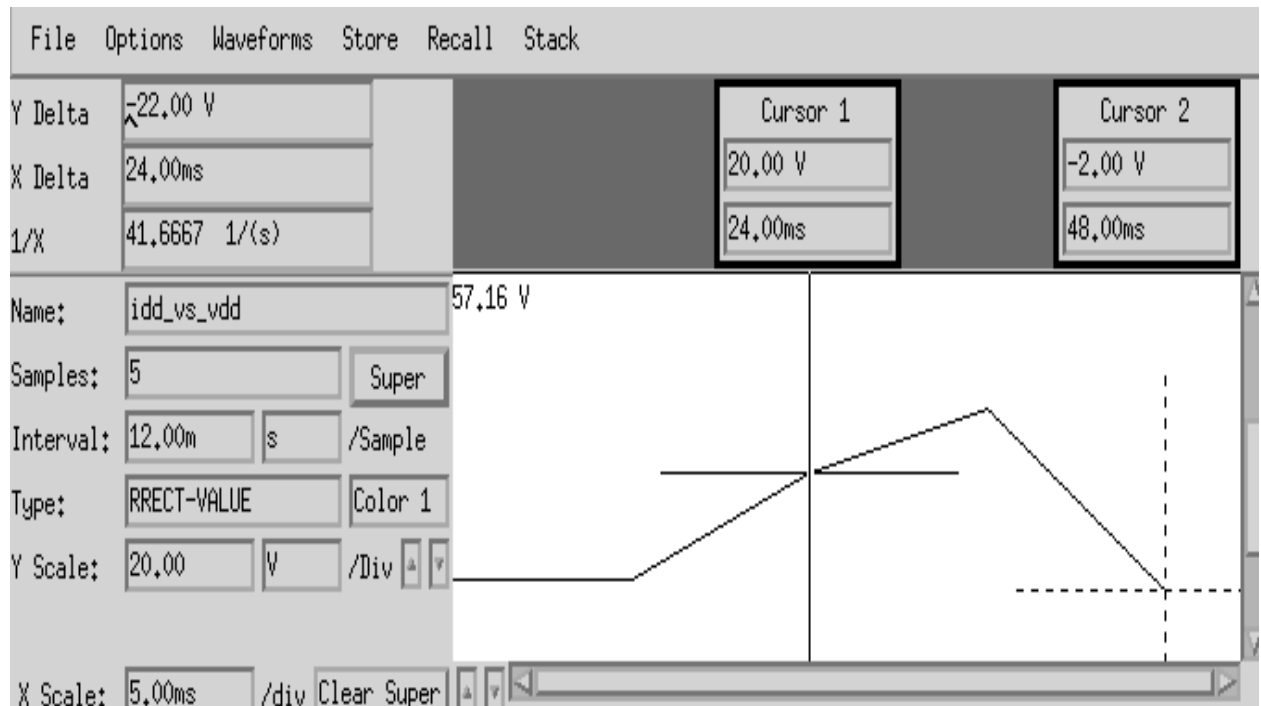
<u>Keyword</u>	<u>Description</u>	<u>Example</u>
version	file type and revision	version away 0 1 0;
date	current date	date 02 25 96;
time	current time	time 16 37 37;
name	name of away waveform	name = "idd_vs_vdd";
type	type of coordinate system - rrect (rectangular floating point), rlong (rectangular long integer), crect (complex floating point), polar *	type = rrect;
size	number of samples	size = 500;
sample_interval	size of x-axis steps	sample_interval = 0.012;
x_units	x-axis units	x_units = "s ";
y_units	y-axis units	y_units = "V ";

* See Toolbox Mixed Signal Supplement for a more detailed description of coordinate types.

The sample data must be preceded by the “pattern;” keyword and terminated with the “pattern_end;” keyword. The following example shows a very simple AWAV file and its associated display using the AWT:

Example:

```
version awav 0 1 0;  
date 02 25 96;  
time 16 37 37;  
name = "idd_vs_vdd";  
type = rrect;  
size = 5;  
sample_interval = 0.012;  
x_units = "s ";  
y_units = "V ";  
pattern;  
1.1e-01;  
1.0e-01;  
2.0e1;  
3.2e+01;  
-2.0;  
pattern_end;
```



VPLOT

We have using using the shmoo callback routine in the past to plot Vdd vs. Idd, an important piece of characterization data. Performance was slow and usually required several iterations to get the proper axis scaling. The user debug command “vplot” was added in order to implement the AWAV technique. Source code and representative plots for a TTL and a CMOS device are attached. In order to use vplot in your own test programs, the following steps are required:

- 1) Compile the source code for vplot and add to the appropriate library file using the Unix command *ar*:
- 2) Add the statement ‘user_init_personal’ to the device test program (this adds vplot and any other user debug commands). See the Toolbox Programming Manual for additional information on adding user debug commands.

VPLOT SOURCE CODE

```
/****** COPYRIGHT 1996 Integrated Systems Test Digital *****/
/*****
FUNCTION: vplot.c
DESCRIPTION: Peter's routine for plotting Vdd vs. Idd and
formatting it for pretty AWT display.
INPUTS: None.
RETURN VALUE: SUCCESS on successful completion.
BUGS: Date and time in awav file header are not real.
FUTURE ENHANCEMENTS: Have not figured out x-scaling due to
no documentation on awav format, so must
use Vddmin = 0 to get correct x-scale. Note: This
was fixed in AWT Version 2 approximately July 1997

Revision History: 1.0 February 27, 1996 Initial Release
1.1 May 27, 1999 Added X-Offset feature (requires AWT version 2)
*****/
#include <box.h>
#include <elem.h>
#include <stdio.h>

#define NUM_SAMPLES 500

static int vplot()
{
double vddmin = 0; /* minimum Vdd value */
double orig_vddmin = 0; /* minimum Vdd value stored away */
double vddmax = 0; /* maximum Vdd value */
double idd[NUM_SAMPLES]; /* array to store Idd results */
double increment; /* the size of the Vdd increment */
FILE *awav; /* the awav output file */
char status[2]; /* the user answers yes or no */
int i; /* a counter for sample size */
int logging; /* save and restore datalog status */
int psnum; /* which power supply */
int num_samples; /* the number of x-axis points */
int didit = 1; /* success of a string compare */

/* turn off annoying datalogging */
logging = datalog_flag(DEFAULT_MODE, NULL);
datalog_flag(0, NULL);

/* User I/O */
while (didit = strcmp(status, "y") != 0)
{
printf("Enter Minimum Vdd Voltage: ");
scanf("%lf", &vddmin);
printf("Enter Maximum Vdd Voltage: ");
scanf("%lf", &vddmax);
}
```

```

printf("Enter Power Supply number: ");
scanf("%1d", &psnum);
printf("Enter number of samples (500 max): ");
scanf("%d", &num_samples);
printf("You entered %4.2lf for Vdd min and %4.2lf ", vddmin, vddmax);
printf(" for Vdd max on PS %d, using %d samples\n", psnum, num_samples);
printf("Is this correct? [y/n]");
scanf("%s", &status);
*status = tolower(*status);
}

printf("\nPlotting ...");
fflush(stdout);

/* the actual test */
increment = (vddmax - vddmin)/num_samples;
orig_vddmin = vddmin;
for (i = 0; i < num_samples; i++)
{
    set_voltage_supply(psnum, vddmin, 1.0, DELAY, NOCHANGE);
    if (i % 10 == 0)
    {
        printf(".");
        fflush(stdout);
    }
    idd_measure(psnum, &idd[i], 1);
    /* printf for any necessary debug */
    /*printf("idd = %f\n", idd[i]);*/
    vddmin = vddmin + increment;
}

/* Open a file and put in the header */
awav = fopen("vplot.awav", "wt");
fprintf(awav, "version awav 0 2 0;\n");
fprintf(awav, "date 02 25 96;\n");
fprintf(awav, "time 16 37 37;\n");
fprintf(awav, "name = 'idd_vs_vdd';\n");
fprintf(awav, "type = rrect;\n");
fprintf(awav, "size = %d;\n", num_samples);
fprintf(awav, "sample_interval = %lf;\n", increment);
fprintf(awav, "offset = %lf;\n", orig_vddmin);
fprintf(awav, "x_units = 'V';\n");
fprintf(awav, "y_units = 'A';\n");
/*fprintf(awav, "offset = %lf;\n", vddmin);*/
fprintf(awav, "pattern;\n");

/* Now enter the data */
for (i = 0; i < num_samples; i++)
    fprintf(awav, "%lf;\n", idd[i]);

/* Now put in the footer and close the file */
fprintf(awav, "pattern_end;\n");
fflush(awav);
fclose(awav);

/* and restore datalogging to its former self */
datalog_flag(logging, NULL);
printf(" Done\n");

/* the next statement gets rid of an unwanted <cr> that I
don't understand */
getc(stdin);
return (SUCCESS);
}

void
user_init_personal()
/* Add the following line to your device test program:
user_init_personal();
*/
{
void *cmdref;
cmdref = dbs_add_cmd("VPlot", "Plot Vdd vs. Idd", DBS_HELP_USER,
    "Plots Vdd vs. Idd to an AWAV (AWT compatible) file",
    "vplot", vplot);
}

```

