WAVECREST Corporation

TESTING 270MHZ PLL CLOCK DEVICES ON A TRILLIUM ATE TESTER WITH THE *WAVECREST* DTS 2070

Application Note No. 125

WAVECREST Corporation continually engages in research related to product improvement. New material, production methods, and design refinements are introduced into existing products without notice as a routine expression of that philosophy. For this reason, any current *WAVECREST* product may differ in some respect from its published description but will always equal or exceed the original design specifications unless otherwise stated.

WAVECREST Corporation
A Technologies Company
7275 Bush Lake Road
Edina, Minnesota 55439
(612) 831-0030
(800) 733-7128
www.wavecrestcorp.com

Contents

Testing 270MHz PLL Clock Devices on a Trillium ATE Tester with the *WAVECREST* DTS 2070

Introduction

The *WAVECREST* DTS 2070 Digital Time System is used with many ATE testers to make the fast, accurate timing measurements for Phase Lock Loop (PLL) clock devices. Many PLL clock frequencies range from 1GHz to 16MHz and have specifications of 50ps to 150ps clock skew and 16ps to 300ps rms jitter. The PLL Clock DUT (Device Under Test) in this application example operates at 270MHz, has a 100ps clock skew, 50ps rms jitter specification and is free running.

Many ATE systems cannot make the frequency measurements and period jitter measurements necessary for this type of DUT, because of basic operational frequency, accuracy, and bandwidth (BW) limitations. Another problem for many ATE systems is they have difficulty with asynchronous events and must resort to using match loops to synchronize the DUT clock to the tester. For the following reasons, the WAVECREST DTS 2070 was selected to test this asynchronous PLL clock DUT:

1. The DTS 2070 measures 10ps accuracy and femtosecond resolution.
2. The DTS 2070 typical jitter noise floor is less than 5ps.
3. The DTS 2070 communicates with the instrumentation IEEE-488 GPIB bus.
4. The DTS 2070 can measure any asynchronous event, one shot or averaged.

This application note discusses using the *WAVECREST* DTS 2070 in a high performance IC testing PLL clock DUT application with the use of the Trillium ATE tester as controller for the GPIB communications. This paper also covers the design of the DUT board, and the GPIB commands to the *WAVECREST* DTS 2070 for setup and measurement. Since the DTS is connected to the Trillium, this note also covers some of the tester program language commands for GPIB communications.

DUT Board Design Considerations

The DUT board design considerations are as follows:

1. Maintaining a 50-ohm environment.
2. Crystal input XI and XO input circuitry.
3. Clock output circuitry and pull-up resistors.
4. Signal attenuation into the wave CH1 and CH2 inputs.

In response to customer demands, the *WAVECREST* DTS2070 Digital Time System was designed for use with many ATE testers to make the fast, accurate timing measurements the ATE systems cannot make—because of either basic accuracy or bandwidth (BW) limitations. Most ATE test systems use UNIX-based workstations like the SUN in a C/C++ environment with the NI-488.2M GPIB software driver and the NI interface. This paper helps the test engineer write a test program to measure critical timing parameters on ICs using the DTS with an ATE tester in an automated environment via the GPIB bus.

This paper explains how to:

1. Configure the GPIB device parameters in a NI-488 configuration file for the GPIB software driver.
2. Include files that must be declared in order to perform the necessary function calls of the GPIB software driver.
3. Initialize and set up the DTS with C/C++ software commands.
4. Utilize new, quick, easy-to-use setup and measurement macro commands for the DTS.

Some fundamental GPIB software commands must be performed for the controller/ instrument to communicate. This paper provides essential software examples that guide the test engineer through successful execution of program instructions over the GPIB.

ANSI/IEEE-488.2 Standard

The ANSI/IEEE Standard 488.2 was released in 1987 with a common set of commands that define protocol, error handling, status reporting, and data formats. All ATE test systems have a GPIB/IEEE-488 bus available for interfacing to instruments that adhere to these standards, whether the interface connects to the SBus slots or the SCSI port. The DTS 2070 instrument conforms to the standards of IEEE-488.1 and IEEE-488.2 and talks/listens to all ATE system controllers compliant to these standards.

Include Files

At the beginning of the DTS module are “include” files that need to be declared and are essential to execute the high level function calls of the gpib.drv. It is important to use the proper syntax for the function calls such as ibrd or ibwrt. Without the include file, the call would be meaningless and the compiler would detect an error warning. The include file is a preprocessor directive, so the compiler can perform preparatory work on the source code before compiling.

The include files are as follows:

```
#include  <stdio.h>
#include <strings.h>
#include  "cib.h"          /* gpib ibxx function library for c */
#include  "dts2070.h"    /* special calls supplied by WAVE */
```

Other include files, depending on system software, are:

```
#include <teradyne.h>  /* custom ni driver j953 type testers */
#include  "globals.h"
#include  <sgtty.h>
#include  "ni_gpib.h"     /* custom ni driver for mt/lt1101 type testers */
#include <stdlib.h>
#include <mtbox.h>
#include  "ugpib.h"       /* custom ni driver  se212 type testers */
#include "agile.h"
#include  "ugpib.h"       /* custom ni driver s9000 type testers */
#include  "gpib.h"        /*  ni driver in C for sun */
struct device device
static int dts_no;
static char rdbck[256];
```

Data Types

The programmer needs knowledge of data types to properly declare variables for the DTS. All voltage parameters are real and declared as double. All burst measurements including AVERage and JITTer, are floating point and declared as double. All samples are integer using modifier long and declared as long.

Termination Characters

All serial stream data transfers must be null-terminated by a GPIB EOI or the NL character. The character for a NL (new line) is a ASCII 0A (hex), NULL is a ASCII 00 (hex) and EOI (end or identify) is a hardware line. This signifies when the GPIB transmission is terminated. Therefore, the parser automatically detects the character and terminates the data transfer.

Initialization Function

The DTS needs to be initialized only once at the beginning of the program. Without the initialization and configuration, the controller/instrument will not communicate. The common commands at initialization are sent to the DTS by high level function calls of the NI488.2 software driver (gpib.drv). The calls of ibrsp (serial poll response), ibsta (command status), ibrd (read), ibwrt (write), ibtmo (time-out), ibpad (primary address), ibsad (secondary address), ibeos (end of string), ibeot (eoi on last byte), ibclr (clear bus) are all part of this driver (gpib.drv).

The *dts_initialize* function is essential to: call the config file "dts2070," communicate to GPIB address 05, bring the instrument on line (ibonl), clear the GPIB bus (ibclr) and status registers, and allow a 2.0 second delay for the clear to take effect. This *ts_initialize* function works for all testers, and the code structure should be used as follows:

```
void dts_initialize( ) {
  if ((dts_no = ibfind ("dts2070")) < 0)
  {
    printf (" GPIB initialization FAILED for BAD address or \n");
    printf (" GPIB cable connection or \n");
  ibonl (dts_no, 0);                          /*  takes instrument off line = 0  */
  }
 else {
  ioctl (dts_no, IBGET, &device);             /*  optional statement associated with ibfind device  */
  ibtmo (dts_no, T10s);                       /*  time out 10 sec.  */
  ibpad (dts_no, 5);                          /*  primary address = 5  */
  ibsad (dts_no, 0);                          /*  secondary address = 0  */
  ibeos (dts_no, 0);                          /*  ignore eos char on read = 0  */
  ibeot (dts_no, 1);                          /*  send eoi on last byte of write = 1  */
  ibonl (dts_no, 1);                          /*  instrument on line = 1  */
   if ( ibclr (dts_no) & ERR)
        printf ( " ERROR in BUS CLEAR \n");   /* clear gpib bus  */
  delay ( 2.0)                                /*  delay 2 sec. to allow both bus to clear  */

  ibwrt (dts_no, "*CLS; :SYST:HEADOFF; :SYST:LONGOFF; *ESE125; *SRE49",51);
  ibwrt (dts_no, ":ACQ:COUN100; :SYST:WAVSQU",26);
  if (ibsta & ERR) printf (" IBSTA ERROR \n" );
  }
 return;
 }
```

The ibwrt call passes the arguments of the GPIB device address with the ASCII command string and size of the string. Each ASCII command performs other commands that clear the status register (`*CLS`), turn header information off (`:SYST:HEAD OFF`), and follow the abbreviated truncation rule of four characters for alpha mnemonics (`:SYST:LONG OFF`) as used throughout this application example. The event status (`*ESE125`) and service request enable (`*SRE49`) commands set up masking to obtain status information during serial polling.

The instrument commands (`:ACQ:COUN100`) set the sample size to 100 and (`:SYST:WAVSQU`) set the pulse find to square wave for VOH/VOL measurements. The ibsta call exchanges protocol to query status of command transfer to the DTS instrument and reports if any ERR occurred (ERRor is a NI-488 gpib.drv declared variable and has built-in error reporting). Once the function is complete to initialize, invoke header format, setup masking for status reporting, acquire count, and report error status; the next task is read a response.

Identify Function

First, a response (ibrd) must be read from the GPIB. An *****IDN? command ensures

the GPIB communication of the DTS. The IDeNtify query returns a string “WAVE, DTS2070, 1”. While waiting for the reply, performing a serial poll on the status byte register to check that the MAV bit is set high is suggested. The Message AVailable status bit indicates that the GPIB has data to transfer and can be read only after the bit is high. Without the serial poll, the data may not be available and the read would be meaningless.

```
void dts_identify( ) {
  ibwrt (dts_no, "*IDN?",5);
  if (ibsta & ERR) printf (" IBSTA ERROR \n" );
        int poll_status;
        poll_status = 0;
                while (!(poll_status  & 0x 10))
                {
                 ibrsp(dts_no,&poll_status);
                }
        ibrd (dts_no, rdbck,40);
        printf (wave idn = %s \n",rdbck);
 return;
 }
```

Pulse Find

The DTS pulse find capability measures the peak voltages SINe or the VOH/VOL SQUare on the CH1 and CH2 signals selected by the command `:SYST:WAVSQU`. To initiate a pulse find (`:ACQ:LEV`) command, the instrument measures the signal high level and low level. This function is necessary to automatically determine the 20 to 80% voltage percentage points for TT+ based on the signal VOL/VOH in the `:ACQ:RUNTT+` application.

```
void dts_levtt( ) {
       ibwrt (dts_no,":SYST:WAVSQU; :TER?; :ACQ:LEV; :TER?",36);
                while (!(poll_status  & 0x 10))
                        ibrsp(dts_no,&poll_status);
        brd(dts-no, vmax-str, 20);
return;
}
```

The `:CHANSTOP:MAX?` query returns VOH data. This function is necessary not only for determining amplitude verification of an output signal VOL and VOH, but it is also necessary for determining overshoot and/or undershoot parameters.

Once the proper response is returned from the DTS, the next step is to use `:SYSTEM:MACRO` commands to set up the DTS.

Setup Function

The `:SYST:MAC` command is fast, easy-to-use, and maintains an exact sequence of commands necessary to complete a setup that specify functions, channels, arming mode, and threshold voltages. The three setups below for functions TPD++, PW+, TT+ are stored in memory under `*SAVel` and a `OPeration Complete` command (`*OPC?`) queries a response from the DTS to status byte the MAV bit when ready.

The setup function writes important instructions to the DTS that will assign proper arming modes with channel information and threshold voltage values to accomplish a measurement. Maintaining an exact sequence of commands as structured in the `:SYST:MAC` command is recommended. Always follows a sequence of function, channel, arming and voltage, especially during the first basic measurement setup. Later, you may want to change arming, count cycle or even threshold voltages, and with a basic setup, simple changes are sufficient.

```
void dts - setup{
      ibwrt(dts_no,":SYST:MAC/TPD++/BOTH/AUT/STARTFIRST/ /2.000/2.000/", 53);
      ibwrt(dts_no,":SYST:MAC/PW+/ 1 /AUT/STARTFIRST/ /2.000/2.000/', 51);
      ibwrt( dts_no,": SYST: MAC/TT+/ 2 /AUTSTOP/20 80///', 36);
      ibwrt(dts_no,"*SAV1; OPC?", 5);
                  while (!(poll-status & Ox 10))
                  {
                  ibrsp(dts_no,&poll_status);
                  }
            ibrd(dts_no, opcck, 5);
            printf (opc = %d \n", opcck);
return;
}
```

The save/recall commands store all 10 functions, channels and voltages with **one** save command, therefore, in this example the three setups TPD++, PW+ and TT+ are in memory under `*SAV1`. The query `*OPC?` and serial polling of the status byte register for a MAV bit high assures the save is complete when **ibrd** response OPC = 1. Without the serial polling of the status byte, the **ibrd** could read "0" signifying the `*SAV1` was not completed. Wait for the serial poll status, then do the **ibrd**.

Table 1 summarizes the three setups:

Function	Channel	Arming	Enable Mode	Trigger	Voltage l	Voltage2
TPD++	Both	Auto Arm	Start First	User	2.000	2.000
PW+	CH 1	Auto Arm	Start First	User	2.000	2.000
TT+	CH 2	Auto Arm	Stop	20% - 80%	Vol	Voh

Table 1. System Macro Command Setups

The three setups stored in memory can later be recalled by the command `*RCL1` in conjunction with the `:AQRuire:RUN` function command.

Measurement Commands

The measurement macro commands are very fast and easy to use. The quick measure command (`:ACQuire:RUN PW+`) not only recalls the total setup for PW+ including channel, voltage, and arming mode parameters; but also executes a measurement and returns the AVERage and JITTer values.

```
void dts-measpw( ) {
    ibwrt(dts-no,":ACQ:RUNPW+", 11);
                  while (!(poll-status &Ox 1 0)
                  {
                  ibrsp(dts-no,&poll_status);
                  }
      lbrd(dts-no, resp, 100);
          sscanf ( &resp, "%f%f', &average, &jitter);
          printf("average = %e \n", average);
          printf("jitter = %e \n", jitter);
return;
}
```

The need for additional commands and polling are eliminated by the (`:ACQuire:RUN PW+`) command. The command performs internally, by the DTS firmware, a setup recall for PW+, burst measurement and return measurement value. The program need only serial poll the status byte and wait for the **MAV** ready.

System Arming Macro

Be aware that all the examples are in the **auto arming mode** for auto arm on stop, auto arm start first, and auto arm stop first. If your application needs to use external arming with arming on pulse count; a fast and easy to use `:SYST:ARM` macro command is available. The `:SYST:ARM` command specifies trigger source, arming enable mode, channel arm, arm threshold voltages, arm slope and arm count and has up to 10 arming arguments.

ibwrt*(dts-no,":SYST:ARM/EXT/STARTFIRST/ARMl/ / / /POS/ /2/ P", 46);*

The :SYST:ARM macro commands are setting up the EXTernal arming mode with enable sequence in the STARTFIRST mode. The ARM1 input will trigger off the start arming input threshold voltage (determined by the pulse find) and the Arm l slope will be POSitive and arm on start count of 2.

Program Example

The C++ Functions and commands of the DTS written in this application and incorporated in the test program are shown in table no. 2.

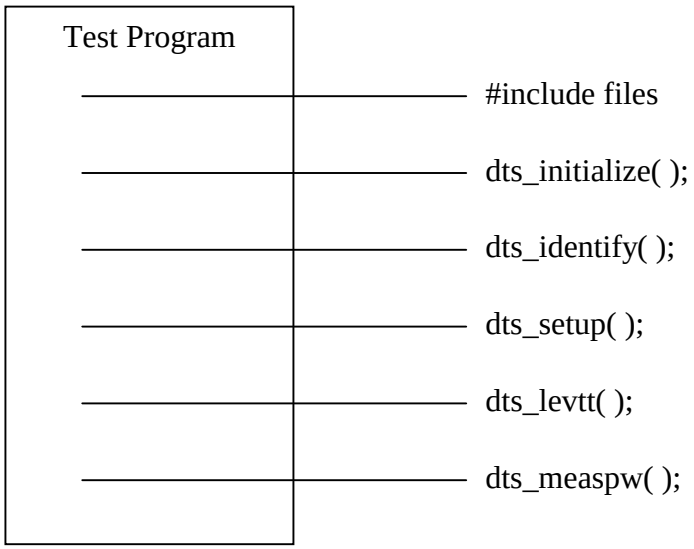

Table No. 2 C++ Functions in test program.

The arming enable modes are **Auto** arm on **START FIRST** for both the measurement of the prop delay (TPD++) of channel 1 rising edge at 2.0vdc with respect to channel 2 rising edge at 2.0vdc, and the measurement of pulse width (PW+) of Channel 1 signal (see table no. 3 for waveform examples). The arming mode enables the measurement of "START before STOP" or START FIRST and is one of four special DTS arming enable modes. The second arming mode, **Auto Arm on Stop**, assures the stop pulse will be measured for rise time (TT+) at 20% to 80% of vol/voh. The DTS will measure any random event or signal and with the arming enable mode the user can control which event is measured first.

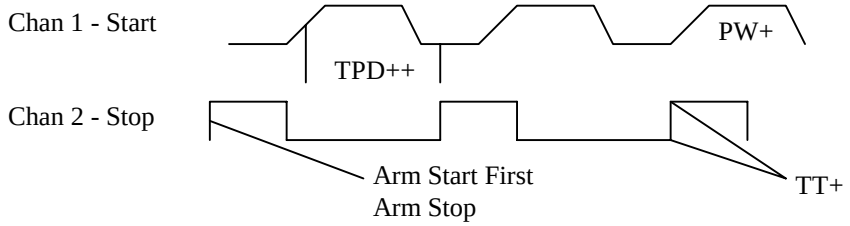

Table No. 3 Waveform Examples

Conclusions

The test commands of the DTS written in this application are for a ATE Test system using a UNIX in a C/C++ environment with the NI-488.2M GPIB software driver.

The WAVECREST Digital Time Scope makes the fast, accurate one-shot time interval measurements necessary for automated testing. This paper will help the Test Engineer organize and structure his test program; so he can properly accomplish status reporting, serial polling, exchange protocol, convert measurement data formats, and utilize easy-to-use macros. The need to measure critical timing parameters on **IC**s for propagation delays (TPD), pulse width (PW) duty cycle, and rise times (TT+) with 10ps accuracy are demonstrated in the application discussed in this document. With the measurement goals and budget achieved, the user is able to write software to initialize the DTS via the GPIB bus, save measurement setups, execute measurements and obtain critical device jitter data.

References

Mitchell Waite and Stephen Prata, New C Primer Plus, Second Edition, Sams Publishing, Carmel, IN; 1993.

National Instruments Corp., NI-488.2 M Software Reference Manual, 1992.

Kerry Newcom, Streaming Data Speeds Up IEEE 488 Bus, Evaluation Engineering, Nokomis, FL, June, 1993.

Herbert Schildt, ANSI C Made Easy, Osborne Mcgraw-Hill, Berkeley, CA.; 1990.

WAVECREST Corporation, Applications Note No. 110; Verifying ATE SYSTEM ACCURACY and Deskew, June, 1993.

Operators Guide, Digital Time Scope DTS 2070. WAVECREST Corporation, Edina, MN.

Anatole Oiczak, UNIX System V Quick Reference Guide Release 4, ASP, San Jose, CA; 1993.

IEEE-488 Interface Guide, Digital Time Scope DTS2070. WAVECREST Corporation, Edina, MN.

WAVECREST Corporation
World Headquarters
7275 Bush Lake Road
Edina, MN 55439
(612) 831-0030
FAX: (612) 831-4474
Toll Free: 1-800-733-7128
www.wavecrestcorp.com

WAVECREST Corporation
West Coast Office:
1735 Technology Drive, Suite 400
San Jose, CA 95110
(408) 436-9000
FAX: (408) 436-9001
1-800-821-2272