AUTOMATING THE SIA-3000

A beginner's reference guide

DRAFT

1 Introduction

The Wavecrest SIA-3000 and GigaView™ software have the ability to run automated tests or control the SIA remotely. There are a number of methods to achieve this: GPIB, Production API (PAPI), LabVIEW™, Remote GigaView™ and Visual Basic Macros.

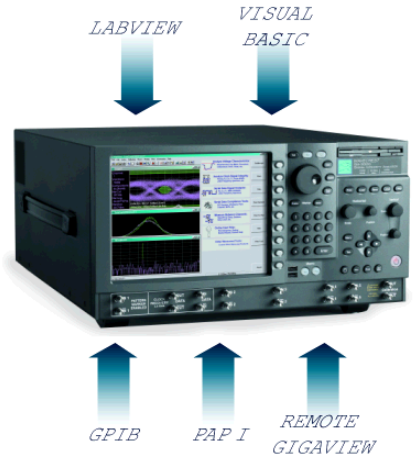
This paper is divided into sections describing the purpose and general implementation of each method. Additionally, example code is provided and some general applications of each implementation are described. It is beyond the scope of this paper to provide command references/definitions for all tools, commands or structures.

This paper should provide the reader with the knowledge to be able to make a choice of the most appropriate method taking into consideration, test times, programmer's experience or ease of use and application (lab or production). The code examples in this paper are mostly limited to basic clock measurements such as a period or statistics resulting from a histogram of period measurements.

Reference Documents:

- GigaView Software, see *"SIA-3000 User's Guide and Reference Manual", Doc# 200053-06 Rev A*
- GPIB Command Reference, see *"SIA-3000 GPIB Programming Guide", Doc# 200007-05 Rev A*
- LabVIEW Drivers, see *"SIA-3000 LabVIEW Driver Reference Guide", Doc# 200057-04 Rev A*
- Production API, see *"SIA-3000 PAPI Programming Guide"*, Doc# 200212-04 Rev A
- Detailed descriptions of each measurement tool from the perspective of the GigaView User Interface, see "*GigaView Getting Started Guides*"

Each approach has advantages or disadvantages depending on the situation in which the technique will be used. For example, a low level GPIB command set may require more time to understand and program—a negative—but it provides extremely fast measurements that are used in a production environment—a benefit. On the other hand Visual Basic Script Macros provide ease of use from the front panel, but would not typically be used in a production environment.



2 Overview of automation approaches

2.1 Basic Measures GPIB

There are actually three GPIB command sets that can be implemented. 'Basic Measures' is the lowest level. It provides essential signal measurements such as Period/Pk-Pk/1-sigma/Frequency/and Skew. It is also the fastest method and is used greatly in ATE or Production environments where very basic tests are required with very little test time.

2.2 Binary Packet GPIB

Binary Packet GPIB commands are essentially an expanded set of commands that allow access to the different tools used in the GigaView Software. This command set allows you to perform measurements from all of the tools and uses a binary packet that minimizes GPIB bus traffic. It optimizes speed but is more machine friendly than user friendly. This GPIB set is not often used in its 'raw' form but is the layer that underlies PAPI. For more on PAPI see section 2.4 and 3.4.

2.3 Tool Oriented GPIB

Tool Oriented GPIB commands are an expanded set similar to those in the GigaView software. This command set allows you to perform measurements from all of the tools. These GPIB commands are parsed into commands that correspond to settings in GigaView. Consequently, GPIB traffic increases because each tool must be set up by individual commands rather than the method employed by the Binary Packet. This implementation is more user-friendly than that of the Binary Packet and is the layer that underlies LabVIEW commands. For more on LabVIEW see section 2.5.

2.4 PAPI → Binary Packet GPIB

Production-API is a high level programming language that wraps the Binary Packet GPIB commands. This API makes it easy to program and use the GigaView tools or measurements while preserving the speed often required by production, ATE, or "rack-and-stack" applications.

2.5 LabVIEW → Tool Oriented GPIB

LabVIEW drivers are a high level programming language that wraps the Tool Oriented GPIB command set. These drivers allow LabVIEW users to integrate the SIA-3000 into an automated test environment. While not optimized for speed, the LabVIEW drivers allow easy integration with other instruments. These drivers are typically used in a lab or "rack-and-stack" setup.

2.6 Visual Basic Script Macros

Visual Basic Script Macros actually run directly on the SIA-3000 and so are not intended for remote automation. But, they provide an easy way to automate repetitive tests and log the results directly on the summary page of the GigaView tool. Using macros, the SIA can send GPIB commands to control other instruments, log results, save result plots and text as files. Macros will run on both the Windows® and UNIX versions of GigaView. Macros cannot be run from an external instrument and are typically used in a lab application.

2.7 Remote GigaView

Remote GigaView runs resident on a PC or Workstation that is external to the SIA-3000. It allows you to have the same functionality of the GigaView GUI that runs resident on the SIA but remotely with a GPIB connection and the SIA in 'listener' mode. This is especially useful when characterizing a test setup where the SIA is embedded in a rack or ATE system. Using GigaView Remote, you can verify that the signal under test is actually being detected by the SIA. In fact, SIA-3000 ATE versions do not have a front panel, but GigaView Remote allows the same functionality. This version is typically used in a lab or in an ATE/Production characterization or test characterization environment.

3 Detailed examples of automation approaches

3.1 'Basic Measures' GPIB

3.1.1 Reference Materials

GPIB Command Reference, see "SIA-3000 GPIB Programming Guide", Doc# 200007-05 Rev A

3.1.2 Applications of Basic Measures GPIB

These commands are very good for production applications. Of all measurements in the SIA they are the fastest, but are also not very comprehensive. This command set is usually used in ATE testers where only basic measurements are required and speed of test is critical.

3.1.3 Results to be expected

Basic Measures GPIB commands are limited to the following measurements and results returned:
Period +
PW +
PW -

Example code

The following example is typical of a simple measurement of the period of a clock signal. It is pseudo code because different operating systems and programming languages may have different requirements for some instructions. In general, this example should serve as a useful example.

```
// Pseudo - code to setup a period measurement - assumes channel 1
Send(0,5,":ACQ:FUNC PER",13,EOI);          // Period measurement
Send(0,5,":ACQ:COUN 1000(@1)",18,EOI);     // Set the sample count
Send(0,5,":CHAN1START:COUNT 1",19,EOI);    // First rising edge
Send(0,5,":CHAN1STOP:COUNT 2",18,EOI);     // To next rising edge
Send(0,5,":TRIG:SOURCE INTERNAL",21,EOI); // Arm off the signal itself
Send(0,5,":DISP:LEV 5050",14,EOI);         // 50% voltage threshold

// Pseudo-code to sample the signal to establish the voltage threshold
// This takes about 130ms, otherwise user voltages can be used

Send(0,5,":ACQ:LEV(@1)?",13,EOI);          // Request the "pulsefind"
Receive(0,5,Buffer,sizeof(Buffer),EOI);    // Go get the results

// The buffer will hold results (min voltage, max voltage) similar to the following:
:ACQUIRE:LEVEL -0.1082758 +0.8043081

// To establish user voltages use the following:
Send(0,5,":DISP:LEV USER",14,EOI);         // USER voltage threshold
Send(0,5,":CHANSTART:LEV -0.125",21,EOI); // First measurement edge
Send(0,5,":CHANSTOP:LEV -0.125",20,EOI);  // Next measurement edge

// To take the measurement use the following command
```

```
Send(0,5,":ACQ:ALL PER(@1)",16,EOI);      // Request the measurement
Receive(0,5,Buffer,sizeof(Buffer),EOI);   // Go get the results
```

// The buffer will hold results (avg, stdev, min, max) similar to the following:
:ACQUIRE:ALL +1.1082758e-009 +2.8043081e-12 +1.1006245e-009 +1.1163601e-009

//For skew measurements similar commands are used, except substitute the following:

```
Send(0,5,":ACQ:FUNC TPD++",13,EOI);       // TPD from rising to rising edge
Send(0,5,":ACQ:COUN 1000(@1,2)",20,EOI);  // Set the sample count, both channels
Send(0,5,":CHAN1START:COUNT 1",19,EOI);   // First rising edge, channel 1
Send(0,5,":CHAN2STOP:COUNT 1",18,EOI);    // First rising edge, channel 2
Send(0,5,":TRIG:SOURCE INTERNAL",21,EOI); // Arm off the signal itself
Send(0,5,":DISP:LEV 5050",14,EOI);        // 50% voltage threshold
```


// Pseudo-code to sample the signal to establish the voltage threshold
// This takes about 130ms, otherwise user voltages can be used

```
Send(0,5,":ACQ:LEV(@1,2)?",13,EOI);       //   Request   the   "pulsefind",   both
channels
Receive(0,5,Buffer,sizeof(Buffer),EOI);   // Go get the results
```

// The buffer will hold results (min voltage, max voltage) similar to the following:
:ACQUIRE:LEVEL -0.1082758 +0.8043081 -0.1006245 +0.1163601

// To take the measurement use the following command

```
Send(0,5,":ACQ:ALL TPD++(@1&2)",16,EOI);  // Measurement from Chan1 to Chan2
Receive(0,5,Buffer,sizeof(Buffer),EOI);   // Go get the results
```

'Binary Packet' GPIB

3.1.4 Reference Materials

- *GPIB Command Reference, see "SIA-3000 GPIB Programming Guide", Doc# 200007-05 Rev A*

3.1.5 Applications of Binary Packet GPIB

These commands are good for production applications, but are not typically implemented as stand alone code. Rather, the PAPI structures provide an interface between the familiar GigaView tools and the Binary Packet GPIB. PAPI utilizes Binary Packet GPIB that, without this interface, can be implemented but requires more development time.

3.1.6 Results to be expected

Binary Packet GPIB will return results from the familiar GigaView tools, but are typically implemented at a higher level with PAPI.

3.1.7 Example code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Standard acquire functions */
#define MIN_FUNC      1     /* Minimum valid function                */
#define FUNC_TPD_PP   1     /* TPD +/+                2-Chan         */
#define FUNC_TPD_MM   2     /* TPD -/-                2-Chan         */
#define FUNC_TPD_PM   3     /* TPD +/-                2-Chan         */
#define FUNC_TPD_MP   4     /* TPD -/+                2-Chan         */
#define FUNC_TT_P     5     /* Rising edge            1-Chan         */
```

```
#define FUNC_TT_M     6     /* Falling Edge           1-Chan          */
#define FUNC_PW_P     7     /* Positive pulse width   1-Chan          */
#define FUNC_PW_M     8     /* Negative pulse width   1-Chan          */
#define FUNC_PER      9     /* Period                 1-Chan          */
#define FUNC_FREQ     10    /* Frequency              1-Chan          */
#define FUNC_PER_M    11    /* Period minus           1-Chan          */
#define MAX_FUNC      11    /* Maximum valid function                 */
/* Rise/Fall edge designators */
#define MIN_EDGE      0     /* Minimum reference edge designator      */
#define EDGE_FALL     0     /* Measurement reference is falling edge  */
#define EDGE_RISE     1     /* Measurement reference is rising edge   */
#define EDGE_BOTH     2     /* Used for DDR in EYEH, DBUS, & FCMP     */
#define MAX_EDGE      2     /* Maximum reference edge designator      */
/* Pulsefind mode designators */
#define MIN_PFND      0     /* Minimum valid pulse-find designator    */
#define PFND_FLAT     0     /* Use flat algorithm for pulse-find calc */
#define PFND_PEAK     1     /* Use peak value for pulse-find calc     */
#define PFND_STRB     2     /* Use strobing method for pulse-find calc*/
#define MAX_PFND      2     /* Maximum valid pulse-find designator    */
/* Pulsefind percentage designators */
#define MIN_PCNT      0     /* Minimum valid percentage designator    */
#define PCNT_5050     0     /* Use 50/50 level for pulse-find calc    */
#define PCNT_1090     1     /* Use 10/90 level for pulse-find calc    */
#define PCNT_9010     2     /* Use 90/10 level for pulse-find calc    */
#define PCNT_USER     3     /* Do NOT perform pulse-find; manual mode */
#define PCNT_2080     4     /* Use 20/80 level for pulse-find calc    */
#define PCNT_8020     5     /* Use 80/20 level for pulse-find calc    */
#define MAX_PCNT      5     /* Maximum valid percentage designator    */
/* Arming mode designators */
#define MIN_MODE      0     /* Minimum arming mode designator         */
#define ARM_EXTRN     0     /* Arm using one of the external arms     */
#define ARM_START     1     /* Auto-arm on next start event           */
#define ARM_STOP      2     /* Auto-arm on next stop event            */

/* Structure with standard setup parameters */
typedef struct
  {
  long    lFuncNum;         /* Function to measure                    */
  long    lChanNum;         /* Channel to measure                     */
  long    lStrtCnt;         /* Channel start count                    */
  long    lStopCnt;         /* Channel stop count                     */
  long    lSampCnt;         /* Sample size                            */
  long    lPadLoc1;
  double  dStrtVlt;         /* Start voltage                          */
  double  dStopVlt;         /* Stop voltage                           */
  long    lExtnArm;         /* Arm when external is selected          */
  long    lPadLoc2;

  long    lOscTrig;         /* O-scope trigger                        */
  long    lOscEdge;         /* O-scope rise/fall trig                 */

  long    lFiltEnb;         /* Filter enable                          */
  long    lPadLoc3;
  double  dFiltMin;         /* Filter minimum                         */
  double  dFiltMax;         /* Filter maximum                         */

  long    lAutoArm;         /* Auto arm enable/mode                   */
  long    lArmEdge;         /* Arm rise/fall edge                     */
```

```
  long    lGatEdge;         /* Gate rise/fall edge                    */
  long    lPadLoc4;
  double  dArmVolt;         /* Arm user voltage                       */
  double  dGatVolt;         /* Gate voltage                           */
  long    lGateEnb;         /* Enable gating                          */
  long    lCmdFlag;         /* Command flag for timestamping, etc..   */

  long    lFndMode;         /* Pulse find mode                        */
  long    lFndPcnt;         /* Pulse find percent                     */
  long    lPadLoc5;
  long    lPadLoc6;
  long    lPadLoc7[2][6];

  long    lTimeOut;         /* Timeout in sec's, if negative it's ms  */
  long    lArmMove;         /* Arming delay in steps [can be +/-]     */
  long    lNotUsed[2];      /* DSM channel select                     */
  } PARM;

/* Structure used for Clock Statistics window                         */
typedef struct
  {
  /* Input parameters  */
  PARM    tParm;              /* Contains acquisition parameters      */
  long    lPfnd;              /* Force a pulse-find before each measure */
  long    lQckMeas;           /* If true skip frequency and voltages  */
  /* Output parameters */
  long    lGood;              /* Flag indicates valid data in structure */
  long    lPad1;
  double  dPwPavg;            /* Contains the PW+ average value       */
  double  dPwPdev;            /* Contains the PW+ 1-Sigma value       */
  double  dPwPmin;            /* Contains the PW+ minimum value       */
  double  dPwPmax;            /* Contains the PW+ maximum value       */
  double  dPwMavg;            /* Contains the PW- average value       */
  double  dPwMdev;            /* Contains the PW- 1-Sigma value       */
  double  dPwMmin;            /* Contains the PW- minimum value       */
  double  dPwMmax;            /* Contains the PW- maximum value       */
  double  dPerPavg;           /* Contains the PER+ average value      */
  double  dPerPdev;           /* Contains the PER+ 1-Sigma value      */
  double  dPerPmin;           /* Contains the PER+ minimum value      */
  double  dPerPmax;           /* Contains the PER+ maximum value      */
  double  dPerMavg;           /* Contains the PER- average value      */
  double  dPerMdev;           /* Contains the PER- 1-Sigma value      */
  double  dPerMmin;           /* Contains the PER- minimum value      */
  double  dPerMmax;           /* Contains the PER- maximum value      */

  double  dDuty;              /* Contains the returned duty cycle     */
  double  dFreq;              /* Contains the carrier frequency       */
  double  dVmin;              /* Pulse-find Min voltage               */
  double  dVmax;              /* Pulse-find Max voltage               */
  } CLOK;

int main()
  {
  CLOK clok;
  long length;
  char buffer[8192];

  // Initialize the binary structure
```

```
memset(&clok, 0, sizeof(CLOK));
clok.tParm.lFuncNum = FUNC_PER;
clok.tParm.lChanNum = 1;
clok.tParm.lStrtCnt = 1;
clok.tParm.lStopCnt = 2;
clok.tParm.lSampCnt = 100;
clok.tParm.lAutoArm = ARM_STOP;
clok.tParm.lArmEdge = EDGE_RISE;
clok.tParm.lFndMode = PFND_PEAK;
clok.tParm.lFndPcnt = PCNT_5050;
clok.tParm.lTimeOut = 2;

// Determine the length of the binary packet length string
sprintf(buffer, "%i", sizeof(CLOK));
length = strlen(buffer);
// Create the header string
sprintf(buffer, ":ACQ:CLKSTAT #%i%i", length, sizeof(CLOK));
// Determine it's length
length = strlen(buffer);
// Append the binary packet to the end of the header string
memcpy(&buffer[length], &clok, sizeof(CLOK));
// Disable the header from being returned
Send(0, 5, ":SYST:HEAD OFF", 14, EOI);
// Send the binary packet command
Send(0, 5, buffer, length + sizeof(CLOK), EOI);
// Read the binary packet back with the results of the acquisition
Receive(0, 5, &clok, sizeof(CLOK), EOI);
// Print the results
printf("Per+ : %lf ns\n", clok.dPerPavg * 1e9);
printf("Per- : %lf ns\n", clok.dPerMavg * 1e9);
printf("PW+ : %lf ns\n", clok.dPwPavg * 1e9);
printf("PW- : %lf ns\n", clok.dPwMavg * 1e9);

return 0;
}
```

3.2 'Tool Oriented' GPIB

3.2.1 Reference Materials

- *GPIB Command Reference, see "SIA-3000 GPIB Programming Guide", Doc#* 200007-05 Rev A
- *GigaView Software information, see "SIA-3000 User's Guide and Reference Manual", Doc# 200053-06 Rev A*
- *Gigaview Tools, see Getting Started Guides*

3.2.2 Applications of Tool Oriented GPIB

For results from tools that go beyond the 'Basic Measures GPIB', the Tool Oriented GPIB provides a more full command set of measurement tools. When certain functionality of a tool needs to be accessed or setup, these commands provide that capability. This slows the measurement as compared to Binary Packet/PAPI but provides a more user-friendly development environment.

3.2.3 Results to be expected

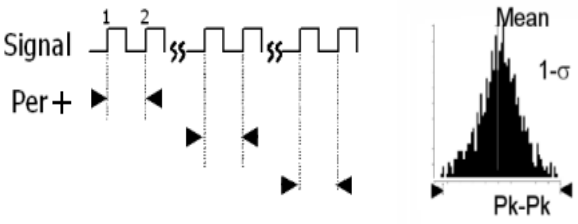
Most results from all tools can be retrieved or calculated from retrieved results. Individual commands are required to set up each tool's advanced options.

3.2.4 Example code

The following example shows a GPIB command sequence for the SIA-3000 that will use the Histogram tool for acquiring Accumulated and Latest Mean, Minimum, Maximum, Peak to Peak, and Standard Deviation Values. The corresponding controls in the GigaView User Interface are shown as reference to aid in understanding the relationship between the GPIB commands and GigaView Control settings. Note that using the GPIB commands does not require also setting the GUI controls.

3.2.4.1 Using Histogram tool with Tool Oriented GPIB commands

The histogram tool measures edge-to- edge on a single channel. It can be configured to measure any combination of edges on a clock signal such as Rising→Rising (Per+) as is shown in the example to the right. Or Falling→falling (Per -), Rising→Falling (PW+), Falling→Rising (PW-). It can also be configured to measure across some number of edges, such as a Rising edge to the third Rising edge.



The first section of this example consists of 10 GPIB steps spanning 3 sections. The 3 sections are:

(1) Initialization & Configuration
(2) Take Measurement
(3) Retrieve Results

Initialization and Configuration should only be done once to set up the instrument before starting a measurement. Taking a Measurement and Retrieving Results may be repeated in cycles to take multiple passes and acquire statistics along the way.

The last part of this document, *Configuration Options*, consists of optional information that may be applied to the 10-step sequence for custom configuration and external arming. Following the 10-step sequence will allow you to retrieve results. The :HISTOGRAM function can be configured in many ways and this paper shows just a brief example. For a detailed reference of GPIB commands refer to the "SIA-3000 GPIB Programming Guide 200007-01RevA".

3.2.4.2 Initialization & Configuration

1. Initialize the device with the following string:

   ```
   :SYST:COMPOFF;:SYST:HEADOFF;:SYST:LONGON;:SYST:ENDLIT;*ESE
   255;*SRE 255
   ```

2. Set the Defaults for the Histogram tool with the following string:

   ```
   :HISTOGRAM:DEFAULT
   ```

3. Set the Channel to be analyzed by the Histogram tool. The following string would set channel 1 as the input. Replace 1 with a different number to select a different channel.

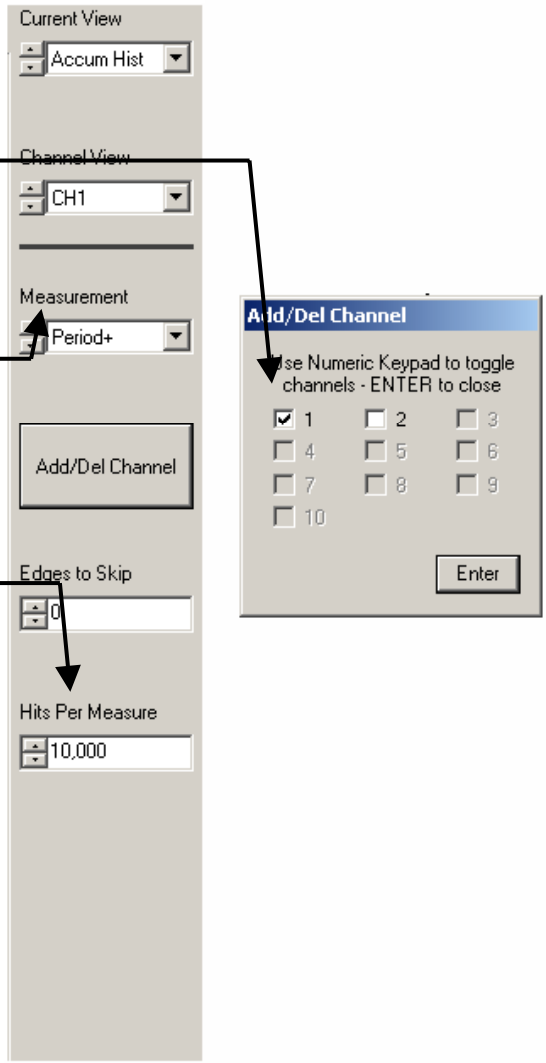
   ```
   :HIST:PARAM:CHAN1
   ```

4. Set the Histogram Tool to measure from rising edge to rising edge with the following string:

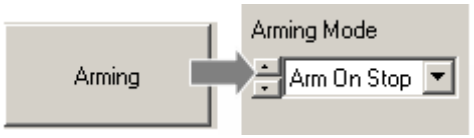
   ```
   :HIST:PARAM:FUNC PER+
   ```

5. Set the number of hits per measure with the following string. This example will set 10000 hits per pass.

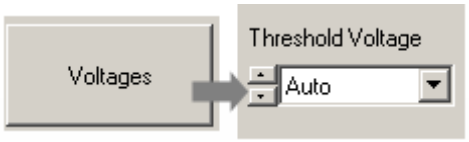
   ```
   :HIST:PARAM:SAMP10000
   ```



6. Send the following string to set internal Arm on Stop mode:
   `:HIST:PARAM:ARM:MODE STOP` →



7. Send the following string to set automatic threshold voltage:
   `HIST:PARAM:THRESHOLD 5050` →



3.2.4.3 Take Measurement

8. Make a Histogram Pass with the following string:
   `:HISTOGRAM:ACQUIRE;*OPC` →

Note: To take multiple passes, simply send the above string repeatedly (One pass per string sent)

3.2.4.4 Retrieve Results

9. Send each of the following strings one by one, reading the return value before sending the next string. The return information will be an ASCII representation of the values. The table to the right shows the corresponding values in the GigaView Histogram Summary page.

`:HISTOGRAM:MEAN?`
Read Accumulative Mean Value
`:HISTOGRAM:MINIMUM?`
Read Accumulative Minimum Value
`:HISTOGRAM:MAXIMUM?`
Read Accumulative Maximum Value
`:HISTOGRAM:STDDEV?`
Read Accumulative 1-Sigma Value
`:HISTOGRAM:PKTOPK?`
Read Accumulative Pk-Pk Value
`:HISTOGRAM:LATEST:MEAN?`
Read Latest Mean Value
`:HISTOGRAM:LATEST:MINIMUM?`
Read Latest Minimum Value
`:HISTOGRAM:LATEST:MAXIMUM?`
Read Latest Maximum Value
`:HISTOGRAM:LATEST:STDDEV?`
Read Latest 1-Sigma Value
`:HISTOGRAM:LATEST:PKTOPK?`
Read Latest Pk-Pk Value

Histogram Summary

	CH1
TJ	179.897ps
DJ (pk-pk)	44.968ps
Lt-rmsJ	9.987ps
Rt-rmsJ	10.345ps
Avg-rmsJ	10.166ps
Chi-Squared Lt	4.212452
Chi-Squared Rt	232.221815
# Passes	19
Accum Period+	20.000797ns
Accum Min	19.941569ns
Accum Max	20.067933ns
Accum 1-Sigma	15.74ps
Accum Pk-Pk	126.364ps
Accum Hits	190,000
Latest Period+	20.001377ns
Latest Min	19.945841ns
Latest Max	20.060022ns
Latest 1-Sigma	15.75ps
Latest Pk-Pk	114.181ps
Latest Hits	10,000
Start Voltage	130mV
Stop Voltage	130mV

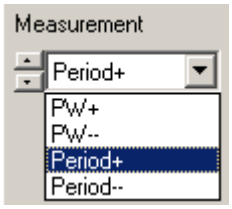
10. Repeat steps (8) and (9) to acquire another pass and retrieve the latest and accumulated values including the new pass.

3.2.4.5 Configuration Options

These are not necessary for the example sequence above, but are alternatives that may be used to configure the device differently.

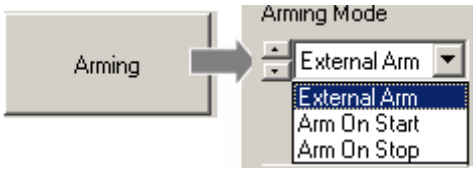
These would replace (4) above to select different edges. PER- is falling to falling, PW+ is rising to falling, and PW- is falling to rising.

```
:HIST:PARAM:FUNC PER-
:HIST:PARAM:FUNC PW+
:HIST:PARAM:FUNC PW-
```



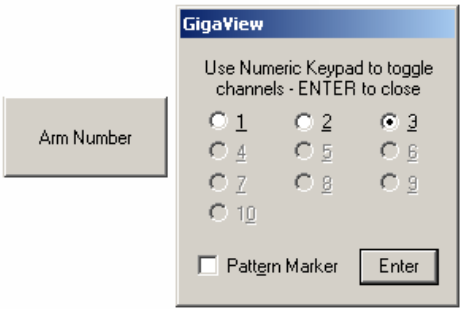
These would replace (6) above. The last item selects external arming which requires many additional commands.

```
:HIST:PARAM:ARM:MODE START
:HIST:PARAM:ARM:MODE EXTERNAL
```



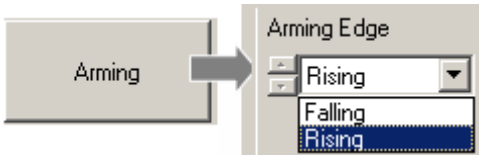
If external arming is selected, the following string would set channel 3 as the arming channel:

```
:HIST:PARAM:ARM:CHAN3
```



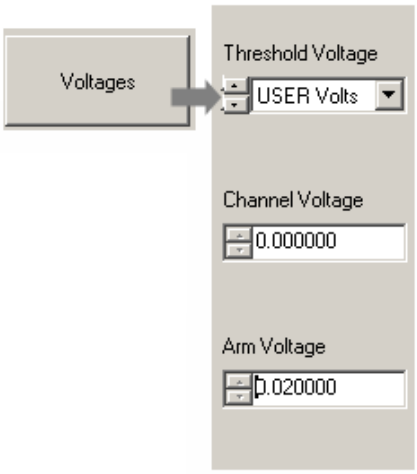
Select one of these two commands to select arming on the rising or falling edge of the arm signal.

```
:HIST:PARAM:ARM:SLOP RISE
:HIST:PARAM:ARM:SLOP FALL
```



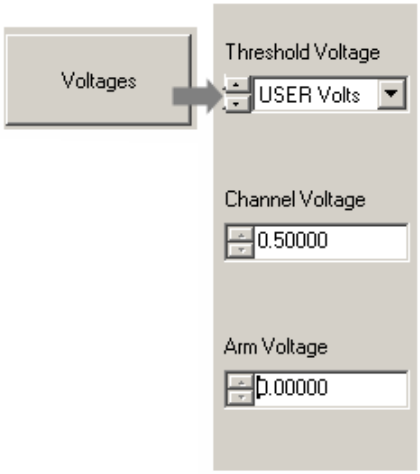
This string would configure the arming level of the external arming signal to 20 mV.

```
:HIST:PARAM:ARM:VOLT 0.02
```



The following sequence of commands would be sent to select a user threshold for the clock signal that is 0.500 volts. All 3 strings would be sent instead of (7) above.

```
:HIST:PARAM:THRESHOLD USER
:HIST:PARAM:START:VOLT 0.500
:HIST:PARAM:STOP:VOLT 0.500
```



3.2.4.6 Conclusion of Tool Oriented GPIB example

The :HISTOGRAM function can be configured in many ways and this paper shows just a brief example. For a detailed reference of GPIB commands refer to the "SIA-3000 GPIB Programming Guide 200007-05 Rev A". Other configurations include enabling TailFit to measure Random (RJ) and Deterministic (DJ) jitter, setting time filters, and "keep-out" regions.

3.3 PAPI → Binary Packet GPIB

3.3.1 Reference Materials

- *GPIB Command Reference, see "SIA-3000 GPIB Programming Guide", Doc# 200007-01Rev A*
- *GigaView Software information, see "SIA-3000 User's Guide and Reference Manual", Doc# 200053-02 Rev A*
- *PAPI Command Reference, see " SIA-3000 PAPI Programming Guide", Doc #200212-04 Rev A*

3.3.2 Applications of PAPI

PAPI is typically used in ATE or production test applications. It is fast because it is a layer that is wrapped around the Binary Packet GPIB command set.

3.3.3 Results to be expected

The Histogram tool is used for capturing statistical data on a simple time measurement such as period, propagation delay, slew rate, pulse width, rise time, fall time, pulse-width, duty cycle, period jitter, lane-skew jitter and slew rate jitter. Use Adjacent Cycle Jitter Tool for cycle-to-cycle jitter and Adjacent Cycle Jitter.

3.3.4 Example code

Measurement Commands, Utility Structures, and Structure Definitions are not covered deeply in this example. For complete details on using the PAPI commands, refer to the "PAPI Programming Manual" #200212-04 Rev A.

This section should provide an idea of the use of PAPI commands.

All measurements are handled in by passing a measurement parameter structure to a calling function, which initiates the measurement. Section 3.2 of the PAPI manual outlines the commands that are used to initiate a measurement and to retrieve the data from the instrument. The commands in Section 3.2 of the PAPI manual are completely independent of the window to be used and are used with all of the measurement windows.

The basic process for creating a measurement follows these basic steps:

1. Initialize a window structure. This means that memory must be allocated, variables declared and the structure set to defaults.
2. Modify any structure elements as needed for the given measurement. Typical modifications include channel number, pattern file name (if data), number of measurements and triggering information.
3. Call a measurement command. Use one of the measurement commands from Section 3.2 and pass it the window structure defined in 1 and 2.
4. Parse the window structure for the results. Once the measurement is completed, the command will return any error messages or a SIA_SUCCESS if measurement was completed successfully.
5. DONE.

If the program is to be done in a production environment, some attention needs to be paid to the memory handling. In step 1, we allocated memory for the structure. If this is done repeatedly without clearing the memory, this will result in a memory overflow error during run time. This can be avoided by either moving the memory declarations to a section of the program that is executed only once. Be sure to execute an appropriate FCNL_Clr----

command when the structure is no longer needed. This only needs to be done once at the end of the program. Alternatively, memory can be allocated and cleared on a per-run basis although this will have a huge impact on test time.

3.3.4.1 Structure used for Histogram Window

The histogram tool is used for displaying the statistical distribution of a given measurement. Measurements made with this tool are limited to repetitive signal measurements such as clock period, duty cycle, pulse width, rise time, fall time, propagation delay and frequency. This tool is typically used for displaying the statistical distribution of thousands of measurements. Important distribution parameters can be calculated based on the data including: RMS, peak to peak, Random Jitter (RJ), Deterministic Jitter (DJ) and Total Jitter (TJ).


```
typedef struct
      {
      PARM    tParm;                //INPUTS
double  dUnitInt;
long    lPassCnt, lErrProb;
long    lTailFit, lForcFit;
long    lMinHits;
long    lFndEftv;
long    lMinEftv, lMaxEftv;
long    lAutoFix;

long    lGood, lPad1;                 //OUTPUTS
long    lNormCnt;
double  dNormMin, dNormMax;
double  dNormAvg, dNormSig;
long    lPad2;
long    lAcumCnt;
double  dAcumMin, dAcumMax;
double  dAcumAvg, dAcumSig;
long    lBinNumb, lPad3;
double  dLtSigma[PREVSIGMA];
double  dRtSigma[PREVSIGMA];
PLTD    tNorm;
PLTD    tAcum;
PLTD    tMaxi;
PLTD    tBath;
PLTD    tEftv;
TFIT    tTfit;
} HIST;
```

EXAMPLE

```
:
#define TRUE 1
static HIST histogram;              //declare histogram to be a structure of
                                    //type HIST
memset(&histogram, 0, sizeof(HIST)); //clear the memory for histogram str.
FCNL_DefHist (&histogram);          //set histogram structures to default
                                    //values
histogram.tparm.lChanNum = 1;       //capture waveform on channel 1
histogram.tparm.lFuncNum = FUNC_PER; //set measurement to be period
histogram.tparm.lStrtCnt = 1;       //measure from first edge to second
histogram.tparm.lStpCnt = 2;        //edge
```

```
histogram.tparm.lSampCnt = 10,000;   //measure 10,000 samples per burst
histogram.lPassCnt = 0;              //reset pass count to zero
histogram.lTailFit = TRUE;           //indicate TailFit desired
histogram.lMinHits = 50,000;         //don't attempt a TailFit until at least
                                     //50,000 measurements are
                                     //accumulated
histogram.lAutoFix = TRUE;           //perform pulse find initially if needed.
FCNL_RqstPkt ( ApiDevId, &histogram, WIND_HIST );  //execute the measurement.
FCNL_RqstAll ( ApiDevId, &histogram, WIND_HIST );   //get plot data
```

3.3.4.2 Conclusion of PAPI Histogram example

The example shows the typical code required to make a histogram measurement. This structure will then send a binary packet over GPIB to configure and make a measurement with the SIA-3000. The binary packet minimizes GPIB traffic and so is optimized for a production environment.

3.4 LabView → Tool Oriented GPIB

3.4.1 Reference Materials

- *LabVIEW drivers, see "SIA-3000 LabVIEW Driver Reference Guide", Doc #200057-00 Rev A*
- *GigaView Software information, see "SIA-3000 User's Guide and Reference Manual", Doc# 200053-02 Rev A*

3.4.2 Applications of LabVIEW

LabVIEW Drivers are specially formatted National Instruments LabVIEW SubVIs which permit the developer to send commands to configure, acquire data and download results from a test instrument. In the case of the SIA-3000, a SCPI Compliant GPIB interface (Tool Oriented GPIB) is provided as a hardware interface. This is supported through LabVIEW using its internal VISA (Virtual Instrument System Architecture) implementation.

Drivers are available for the SIA-3000 to allow remote control of a subset of the GigaView tools. These drivers follow a standard format which passes around an error cluster for LabVIEW and GPIB error handling, as well as a VISA session ID which is a specially formatted string reference containing the GPIB address information.

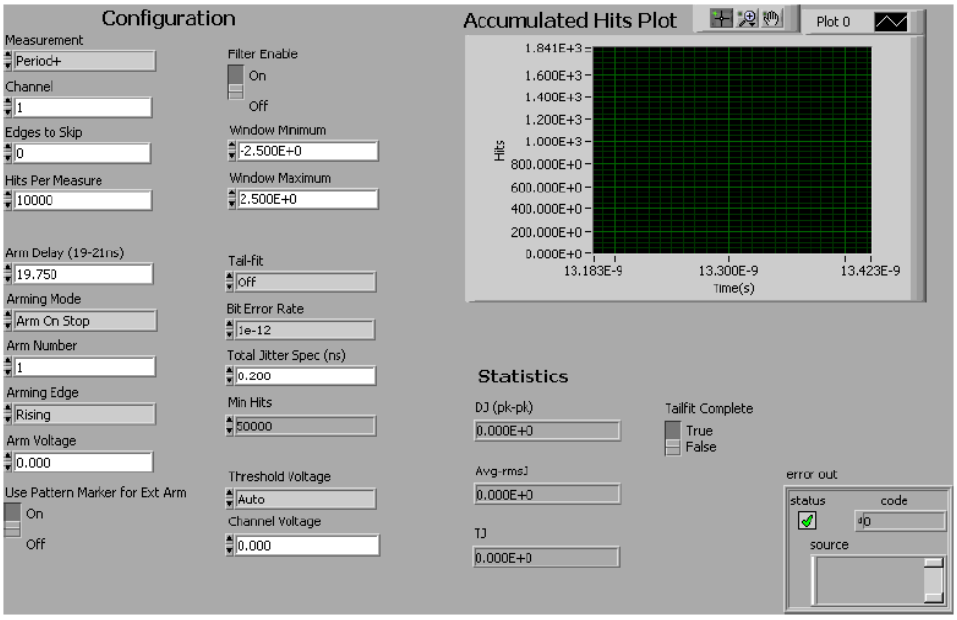
3.4.3 Results to be expected

The output of the LabVIEW drivers can be similar to that of the output of GigaView tools from the front panel of the SIA-3000.

3.4.4 Example code

For all of the available tools, the process of controlling the instrument consists of the following steps:

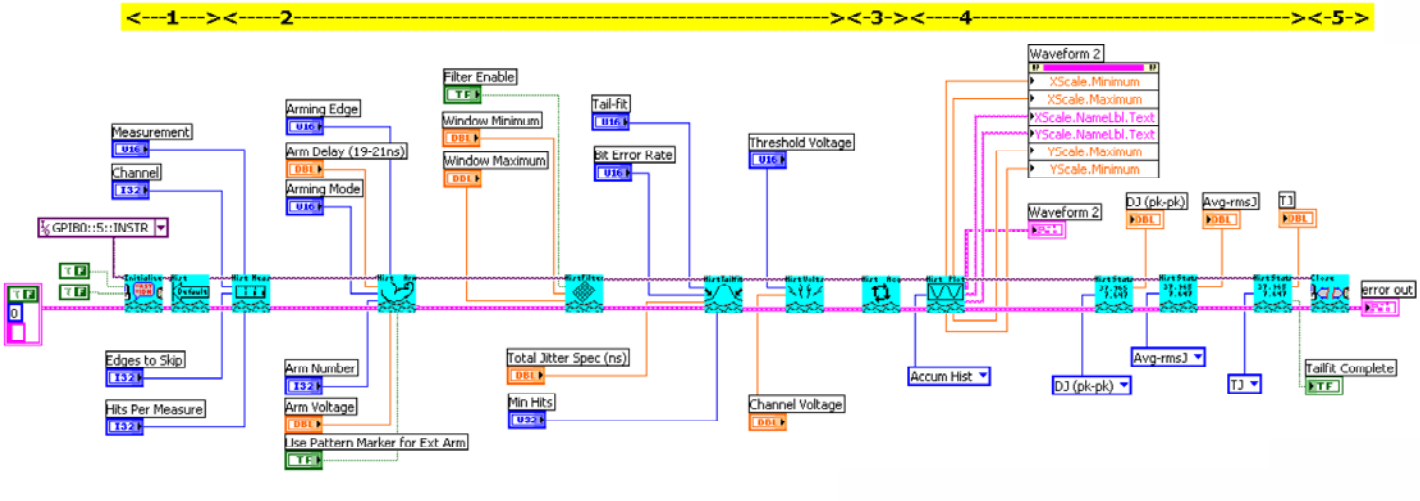
1. Initialization
2. Configuration
3. Acquisition
4. Results Downloading
5. Close Connection



Front Panel—Note that everything to the left is a configuration parameter (Step 2) and everything to the right is a result statistic (Step 4)

The following is a simple example written in LabVIEW to show these 5 steps and how one might go about acquiring data from the SIA-3000. There are many calls that must be made

in succession to get the desired data for the different tools. This specific example is for histogram tool, but the same basic format can be applied to any of the tools.



Block Diagram – Note that the 5 steps are labeled with the yellow LabVIEW comment placed above. This shows which calls go with which steps from left to right.

In this LabVIEW code example, first the SIA-3000 GPIB connection is initialized on address 5. ID Query and Reset are hard coded to false, so a simple initialization will occur with no verification. The defaults are set with the second call and initialization for the Histogram tool is complete.

For the second step, many parameters are wired to additional inputs for the configuration parameters. All the calls in step 2 do the same thing: configure setup parameters on the SIA-3000.

The third step consists of one call to the Histogram Acquire driver. This tells the SIA-3000 to perform a single pass of Data Acquisition. Note: to cycle, this driver may be placed in a loop that will terminate based on number of iterations, or a TailFit™ completion. This is beyond the scope of a simple LabVIEW example, but in step 4 we will address a mechanism for determining if a TailFit has been successfully completed.

The fourth step shows the two types of result reporting common to all tools: Plots and Statistics. This example displays the resulting Accumulated Histogram plot and the three basic Jitter statistics. Notice that there is a TailFit complete output on the statistics drivers. Any call to the statistics driver will return this flag indicating TRUE if a TailFit has been successfully completed and FALSE if it has not. Until a TailFit is completed, successive calls to the Histogram Acquire driver would be needed in order to get valid Jitter statistics. For a real world application, an acquisition loop is needed that will terminate only when a TailFit has been completed.

For more information about LabVIEW, visit the National Instruments website (http://www.ni.com). And for more information about using the SIA-3000 LabVIEW Drivers, see “SIA-3000 LabVIEW Driver Reference Guide”, Doc# 200057-04 Rev A.

3.5 Visual Basic Script Macros

3.5.1 Reference Materials

3.5.2 Applications of VB macros

With Macros you can:
Log results from repeated tests on the Summary page.
Open and configure GigaView measurement tools.
Control other devices with GPIB commands.
Save results or plot graphics to a file.

Possible Applications:
Jitter Tolerance Test Set using GPIB to control Pattern Generator, Periodic Jitter and Random Jitter sources
Characterization Test Set using GPIB to vary device voltage, current or temperature

3.5.3 Results to be expected

One of the most useful capabilities is logging results from repeated tests. With GigaView, any value or multiple values from the summary pages of any tool can be logged. The results are annotated to the same summary page.

3.5.4 Example code

The example macro below takes a value from the summary page of the current active tool and displays that logged value as an Annotation on the summary page. The macro will run the tool repetitively a set number of times. It prompts you to title the logged results, choose a number of passes, choose a pause between passes, and choose the row and column of the value to be logged. When initially running the macro, it is good to first do one pass to ensure that the correct value is logged. A macro will run to completion unless additional commands are added for pauses or breaks.

To use this macro:
Open a tool and go to the summary page. (The macro in Example 1 is very general and does not open or configure a tool. Therefore, it will operate on any active summary page. The macro can be easily modified to add commands to open a specific tool and set the view to the summary page)
Load the Macro (in the top menu choose Macro|Load…) and select the macro from the dialog box.
Or, choose Macro|Edit… and manually type in the macro in the editor.
Decide which value you want from the current active summary page (see Example 2).
Count which row the value is in (the summary page title is Row 0)
Count which column the value is in (The first column with the text is Column 0)
Now run the macro. When prompted, enter the values needed.
Set the “number of passes” to a value of how many runs or cycles of the tool you want. (Note that because the logging is into the Annotation, there is a limit on how much can be logged ~32K characters.)

Once the macro is complete, the contents of the summary page can be saved and pasted into a spreadsheet program.

Macro Example Code

Visi.modify command sets the current tool view to the summary page. The macro will not run without the summary page being active.

Visi.Prompt command asks user for values.

ROW, COL variables are the coordinates on the summary page to be logged.

N is the number of times to run the test.

Visi.Summary command chooses a value from the summary page.

RESULT variable is created with the logged value and a time stamp.

Visi.Annotate command puts the RESULT variable's contents into the annotated text on the summary page.

```
Dim Visi
Set Visi = CreateObject ("Visi.Application")
Visi.Show
Visi.Maximize
Visi.Modify "Current View", "Summary"
RESULT = Visi.Prompt("Enter title:") & Chr(13) & Chr(10)
COUNT = Visi.Prompt("Number of passes:")
WAIT = Visi.Prompt("Time between passes (sec):")
ROW=Visi.Prompt("Row #?")
COL=Visi.Prompt("Column #?")
For N = 1 to Count
  Visi.SingleStop
  RESULT = RESULT _
    & Visi.SummaryCell (ROW, COL) & Chr(9) _
    & Visi.TimeStamp & Chr(13) & Chr(10)
  Visi.Annotate RESULT
  Visi.Wait WAIT
Next
```

Example 1. The General Data-Logging Macro Text

To Save the results to a file, use the "Save" command and enter a path and filename. So, in the example above the summary page can be automatically saved.

After Next, enter this line:
Visi.Save "C:\Test.txt"
The page will be saved on the C: drive as a text file named 'Test'.

The Macro example in Example 1 does not open a tool. It assumes a tool is already open. The 'Modify' current view command will change the view of the currently open active tool to the summary page. The summary page must be active for the SummaryCell command to work. The example below shows a histogram summary

page to the right. The values of Row and Column chosen follow the example. Counting starts with zero, and text/blank lines are counted.

Histogram Example	Column 0 Column 1
Row 0	Histogram Summary
Row 1	Chan1
Row 2	TJ 84.506ps
Row 3	DJ (pk-pk) 36.218ps
Row 4	Lt-rmsJ 3.717ps
Row 5	Rt-rmsJ 3.495ps
Row 6	Avg-rmsJ 3.606ps
Row 7	Chi-Squared Lt 4.019386
Row 8	Chi-Squared Rt 6.376021
Row 9	# Passes 70
Row 10	Accum Period+ 2.00339ns
We want to run the test 3 times, so when prompted for “number of passes”, enter 3.	Accum Min 1.973799ns Accum Max 2.039736ns Accum 1-Sigma 11.217ps Accum Pk-Pk 65.937ps
We have chosen to log “Accum Period+”. The coordinates on the summary page are Row 10, Column 1. When prompted, enter ROWS = 10, COLS = 1	Accum Hits 700,000 Latest Period+ 2.003814ns Latest Min 1.977645ns Latest Max 2.034498ns Latest 1-Sigma 10.76ps Latest Pk-Pk 56.853ps Latest Hits 10,000 Start Voltage -0.025198V Stop Voltage -0.025198V
The logged results are displayed on the bottom of the summary page. These values can be pasted into a spreadsheet program.	Logged Results: ---------- 2.00339ps 2.00456ps 2.00153ps

Example 2. Example of logged results on summary page

The example could be modified to open specific tools and modify their settings. Often, the easiest way to do this is to record a macro and then paste the pertinent lines of code into the example code.

The coordinates on the summary page are static and not relative. Therefore, in some tools that have variable length summary pages, some values may not be able to be logged. One example is the High Frequency Modulation tool. This summary page has a variable length list of all spikes detected on the FFT. Therefore values appearing

below the list will not always be at the same coordinate and will not be able to be logged. Most tools do not have variable length lists on the summary page.

To make the macro display a more descriptive result, you can add text strings to the output, which is displayed on the summary annotation. See Example 3 and 4 below.

This example is written to work only with the Known Pattern with Marker tool. It provides a more descriptive output, and assigns variable names to the coordinates of interest on the summary page.

Results displayed show the logged measurements in a labeled table.

Variables defined for the coordinates on the summary page

```
Dim Visi
Set Visi = CreateObject ("Visi.Application")
Visi.Show
Visi.Maximize
Visi.Modify "Current View", "Summary"
Count = Visi.Prompt("Number of repeats:")
Result = "TJ=" & Chr(9) & "DCD+ISI=" & CHR(9) & "PJ=" &_
CHR(9) & "DJ=" & CHR(9) & "RJ="& CHR(9) &  COUNT &_
" Repeats" & Chr(13) & Chr(10)
For N = 1 to Count
  Visi.SingleStop
TJ = Visi.SummaryCell (3,1)
DCDISI = Visi.SummaryCell (4,1)
PJ = Visi.SummaryCell (5,1)
DJ = Visi.SummaryCell (6,1)
RJ = Visi.SummaryCell (7,1)
 Result = Result _
    & TJ & Chr(9) & DCDISI & CHR(9) & PJ & CHR(9) & DJ &_
CHR(9) & RJ & Chr(9) & "Run #" & N & Chr(13) & Chr(10)
  Visi.Annotate Result
  Visi.Wait 0
Next
```

Example 3. Known pattern with marker data logging

```
Known Pattern w/Marker Summary
        Channel1
Per+
TJ              136.135ps
DCD + ISI       117.49ps
PJ (pk-pk)      2.73ps
DJ (pk-pk)      120.221ps
RJ (1-sigma)    1.237ps

Voltage         -0mV
Bit Rate        2.499994Gbit/s

TJ=             DCD+ISI=    PJ=         DJ=          RJ=      Run= 6
137.664ps       118.899ps   2.728ps     121.627ps    1.246ps
137.009ps       117.534ps   2.683ps     120.217ps    1.299ps
136.713ps       117.656ps   2.722ps     120.379ps    1.264ps
136.229ps       117.477ps   2.687ps     120.164ps    1.249ps
136.164ps       117.912ps   2.7ps       120.612ps    1.209ps
136.135ps       117.49ps    2.73ps      120.221ps    1.237ps

Per+ FFT Spikes     2.7ps/1.2GHz
                    1.5ps/925.1MHz
                    1.4ps/20.1MHz
```

Logged results

Example 4. Output displayed by macro in Example 3

It is possible to use macros to control other instruments by GPIB. The following example shows how to sweep the frequency output of a generator and measure the signal with the SIA-3000.

For this example, we want to sweep a range of frequencies from a signal generator. The Example will use the Histogram to measure and log the period of the clock at each frequency step. This setup could be used as part of a tolerance test to add known amounts of jitter to a device under test. See Figure 2.

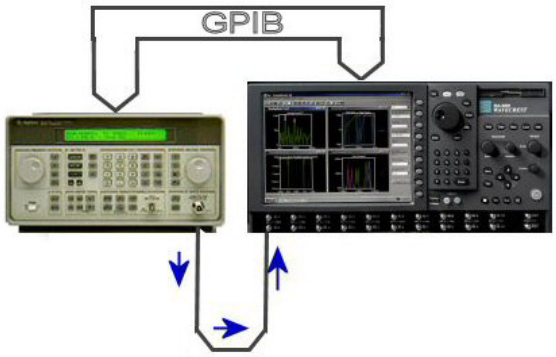


Figure 1. Hardware Setup for the Frequency Sweep

The Hardware Setup
Instruments used are the Wavecrest SIA-3000, Agilent 8648D Signal Generator.
The Instruments are connected by GPIB. The output of the Device Under Test (DUT) is summed with the output of the Agilent 8648D. The SIA-3000 must be set to “Controller” mode in the Edit|configuration menu.

The Macro Setup

The macro in Example 5 is written to prompt for a Start Frequency, Stop Frequency, number of frequency points and the value to be logged.

Row/col set chooses the value to be displayed.

StartFreq/StopFreq sets the range to sweep the 8648D.

DevID, GpibSend and GpibClose send command to the 8648D

8648D output is incremented by GPIB

Measurement result is displayed on summary annotation

```
Dim Visi
Set Visi = CreateObject ("Visi.Application")
Visi.Show
Result = Visi.Prompt("Enter title:") & Chr(13) & Chr(10)
Row = Visi.Prompt ("ROW #?")
Col=Visi.Prompt ("COL #?")
StartFreq = Visi.Prompt("Start Frequency:")
StopFreq = Visi.Prompt("Stop Frequency:")
Points=Visi.Prompt ("Points?")
Increment= (StopFreq-StartFreq)/Points
DevID = Visi.GpibOpen (" ", 0, 7)
Visi.GpibSend DevID, "FREQ:CW " & StartFreq & "MHZ"
Visi.GpibClose DevID
For N = 1 to Points
  Visi.SingleStop
DevID = Visi.GpibOpen (" ", 0, 7)
Visi.GpibSend DevID, "FREQ:CW " & StartFreq & "MHZ"
Visi.GpibClose DevID
startfreq=startfreq+increment
Result = Result _
    & Visi.SummaryCell (row,col) & Chr(9) _
& Chr(13) & Chr(10)
  Visi.Annotate Result
  Visi.Wait 1
Next
```

Example 5. Controlling external instruments with GPIB and logging results.

A tolerance test set could be created by modifying this macro and adding additional test equipment. A general example would be to add jitter to a pattern generator. Using a random noise source, Random Jitter (RJ) can be added. A sinusoidal source would add deterministic jitter in the form of periodic jitter.

In Conclusion, VB macros allow you to log results from measurements on the summary page using the annotation command. The results can be displayed and formatted in an easy to read format, showing only the results in which you are interested. Finally, using macros to control other test equipment from the SIA-3000, you can design full bench-top characterization tests.

3.6 GigaView Remote

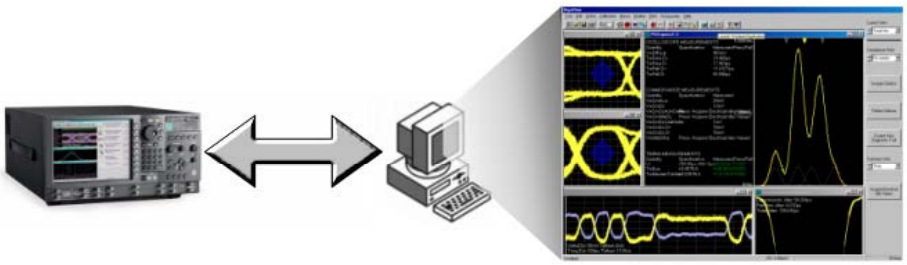
3.6.1 Reference Materials

- *GigaView Software information, see "SIA-3000 User's Guide and Reference Manual", Doc# 200053-06 Rev A*
- *GigaView Tools, see "Navigating GigaView" Getting Started Guide*

3.6.2 Applications of GigaView Remote

GigaView Remote allows you to have all of the functionality of GigaView, but use a remote PC or workstation to control the SIA-3000. GigaView Remote is installed on a computer and requires a valid license (the same as the license of the SIA-3000 to be controlled). A GPIB card and cable are needed to connect the PC and the SIA-3000.

In a lab, GigaView Remote is useful because data can be directly obtained by your PC and saved where you are performing signal analysis. In the case that the SIA-3000 is used for classified research, no results or settings are stored on the hard-drive with GigaView Remote; it strictly uses GPIB commands to control and configure the instrument from a host PC. If the SIA is being used in a "rack-and-stack" setup, GigaView Remote allows you to quickly compare the results of your GPIB or LabVIEW code with the results from GigaView. This is a great aid in test code characterization and development.



4 Conclusion

This document has pointed out some of the different ways of automating the SIA-3000. There are trade-offs for the various approaches. The discussion of the applications of each and the example code should leave the programmer with the ability to understand which approach is best for them. While this paper does not include complete documentation of everything needed to begin, it provides references to relevant guides or manuals for each application.