WinACIF: A Telecom IC Support Tool Using Tcl/Tk

David Karoly, Todd Copeland, and David Gardner
Advanced Micro Devices

WinACIF:
A Telecom IC Support Tool Using Tcl/Tk

David Karoly
David.Karoly@amd.com

Todd Copeland
Todd.Copeland@amd.com

David Gardner
David.Gardner@amd.com

Advanced Micro Devices, Austin, Texas

Abstract

We discuss our use of Tcl/Tk to provide software support for telecommunications Integrated Circuits (ICs). Our Windows®-based Advanced Computer Interface (WinACIF) program works in concert with reconfigurable hardware based on Field Programmable Gate Arrays (FPGAs) to provide essential coordination in laboratory data collection and analysis of a device under test. WinACIF replaces several MS-DOS® based applications. Whereas the previous implementations suffered from the classic limitations of MS-DOS, WinACIF provides the flexibility and functionality of windowing applications by virtue of its Tcl/Tk roots. Tcl/Tk not only supplies more than ample power to create WinACIF, but also adds the benefit of saving valuable time otherwise spent learning a complex API. Run-time loaded Tcl extensions provide the flexibility to support various devices having diverse interfaces. A single Tcl/Tk script dynamically builds a Graphical User Interface (GUI) based on product configuration data retrieved from a data store. Additionally, we used canvas widgets to provide an intuitive interface. For the engineer who requires control beyond that afforded by our GUI, Tcl serves as WinACIF's command language.

1. Introduction

WinACIF is a 32-bit Windows application we designed to support AMD's SLAC™ family of integrated circuits. Telecommunications equipment manufacturers use the SLAC (Subscriber Line Audio-Processing Circuit) family of ICs to build telephony linecards that interface analog telephones with digitally switched networks. WinACIF users include AMD's engineering staff and customers. Using Tcl/Tk, we created a GUI that provides a readily understood and convenient means of programming the hundreds of features contained in the SLAC IC and displaying their current state. We utilized Tcl in conjunction with our application-specific extensions (implemented in C and C++) to script interactions with the IC. The application also configures and controls an interface board that generates and coordinates digital signals between the SLAC device and PC, and between the device and test equipment (Figure 1). Tk's canvas widget provides a handy way to build a dynamically configurable diagram that depicts the signal flow in a manner familiar to the users.

2. Application Description

AMD's SLAC devices are programmable codec/filter ICs. They are essentially embedded digital signal processors that perform A/D conversion, D/A conversion, filtering, compression and expansion functions. With the advent of AMD's advanced SLAC devices, the programmability now extends to control the line power feed, ringing, signaling and test functions. This programmability benefits the linecard designer by providing the flexibility to create one hardware design that satisfies the diverse requirements of multiple markets. This flexibility, in turn, benefits the telecommunication systems manufacturer by streamlining the manufacturing process and by reducing inventory, administrative costs and time required to address new markets.

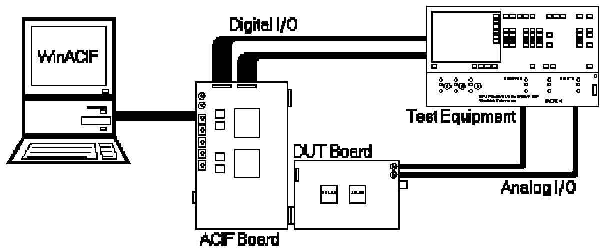


Figure 1. WinACIF lab bench scheme

The cost of all this flexibility is complexity. The most recent member of the SLAC product family has over 115 programmable command op-codes. Each op-code may be associated with a register containing up to 14 bytes of data. The data bytes themselves may contain multiple bit fields of varying widths, each controlling some programmable operating parameter of the device.

The engineer involved in the debug or application of the SLAC device faces a formidable testing and characterization task. This is particularly true when the design is intended to meet the requirements of several markets having diverse specifications. A great deal of lab bench experimentation and verification is required to ensure that the design meets the needs of all the targeted markets. WinACIF and its associated hardware provide a turn key evaluation platform that allows interactive manipulation of all the programmable features of the SLAC devices. This platform allows the engineer to easily explore the multiple "what if" scenarios inherent in linecard design. The development efforts of AMD's customers are streamlined and time to market minimized by having a tool that allows them to immediately begin evaluation of the SLAC device. They can begin the design of their application circuit and establish the proper SLAC device programming for their application without making an investment in building evaluation hardware.

WinACIF is often used in conjunction with WinSLAC. WinSLAC, a design synthesis tool, accepts the linecard design specifications and design constraints as input and produces an output text file containing the information required to configure the programmable filters of the SLAC device. WinACIF includes Tcl commands that convert this information into a stream of commands that will program the data into the SLAC device.

3. WinACIF Structure

Previously, we constructed support tools with the National Instruments' LabWindows® program under MS-DOS, adding C code to provide callback functionality. This implementation suffered from the following limitations:

- 640K of memory
- minimal functionality of widgets
- no window manager
- complex C code to create event loop and update widgets
- no scripting control mechanism
- hard-coded GUI layout
- overhead of multiple program maintenance
- difficulty of supporting new devices with multiple interface/command sets

The ability to extend Tcl with introspective commands written in C allowed us to realize our primary design goal of generalizing the ACIF software to support multiple devices having diverse interfaces.

The main Tcl script, acif.tcl, loads the core of the application, acif.dll. Acif.dll includes a data store comprised of arrays of C structures that contain the programming command sets for each of the SLAC devices. After loading acif.dll, the Tcl script uses one of the new commands, acifProducts, to ask the data store to return a list of available products. The Tcl script uses this list to construct an array of radiobuttons from which the user selects a product name. Upon selection acifProducts is executed again, this time with the chosen name as an argument. This execution configures pointers in the underlying C code, so that subsequent operations reference the appropriate section of the data store.

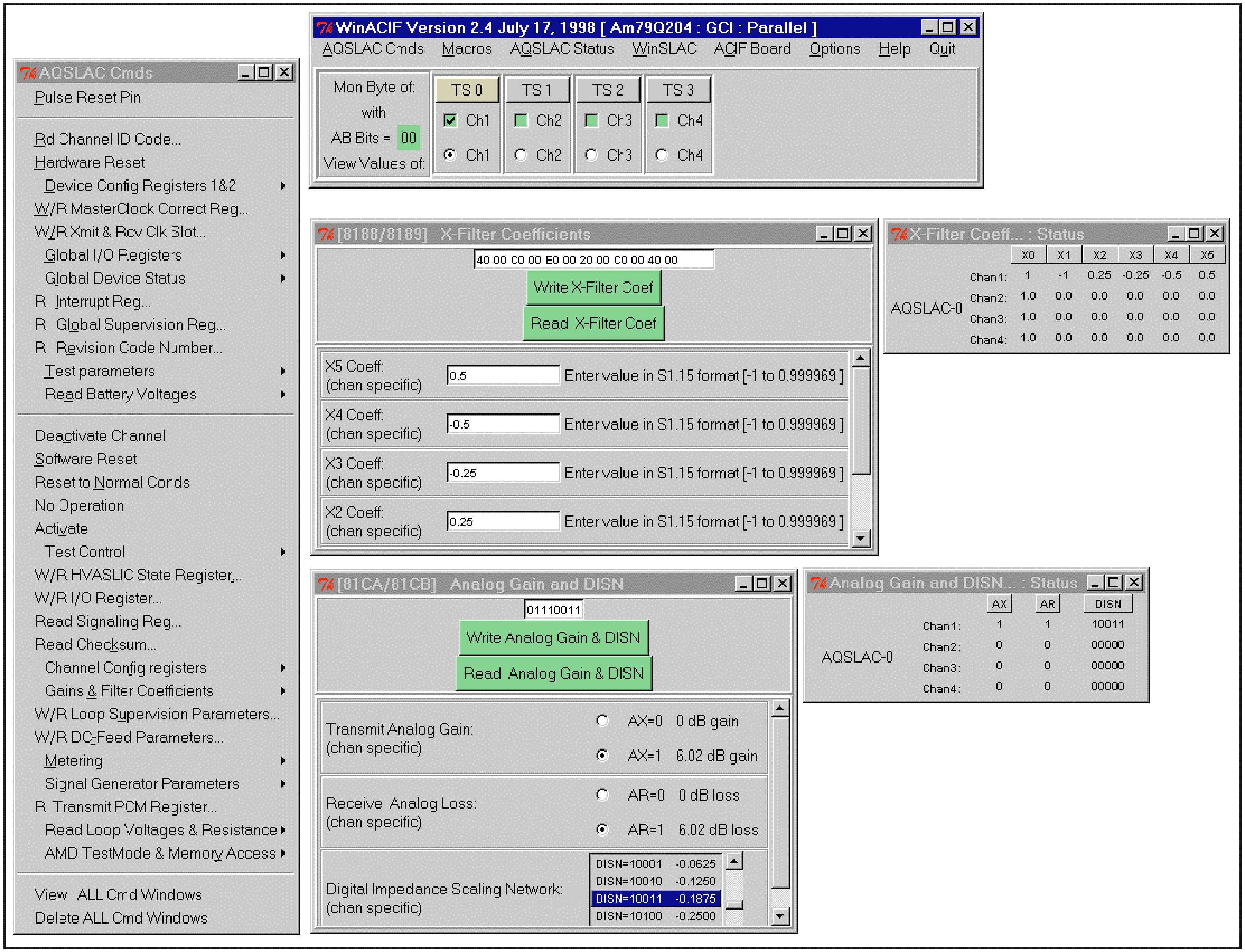
Throughout the program, we used this strategy of dynamically creating the GUI based on data obtained from the underlying data store. Depending on the argument sent with the command, our Tcl commands either return configuration information from the data store, or perform some related operation, such as writing or reading from the SLAC device. This structure made possible the writing of generic Tcl code that created different GUIs depending on the information in the data store.

In addition to acif.dll, the application may load one of several other DLLs, depending on the options selected. For example, the SLAC devices have a variety of hardware interfaces. Some new devices even offer the choice between two interfaces. Downloading a different configuration file into the FPGAs (Field Programmable Gate Arrays) on the ACIF board redefines the interface logic. Acif.dll reconfigures the software by unloading the old driver DLL and loading the one that supports the new configuration. The new driver DLL defines new Tcl commands that control the interface logic implemented in the FPGAs.

The various DLLs and the application-specific Tcl commands they provide are summarized in Appendix A.

4. SLAC IC Command Windows

To make our users as comfortable as possible with the new WinACIF tool, we designed our GUI to resemble the device data sheets with which they are so familiar. After selecting a product name, the user views WinACIF's main menu, which contains eight menu items. The first item contains the SLAC commands. When the user selects this item, a pull-down menu appears listing command descriptions for programming the selected SLAC device. If the user chooses a command with no associated register, WinACIF will execute the instruction directly upon selection from this menu. Otherwise, if the user selects a command that reads from or writes to a register, it will launch a top level window where the user can specify or view the data parameters in the associated register.



Mon Byte of: with AB Bits = 00

	TS 0	TS 1	TS 2	TS 3
Mon Byte of:	[✓ Ch1]	Ch2	Ch3	Ch4
View Values of:	[✓ Ch1]	Ch2	Ch3	Ch4

[8188/8189] X-Filter Coefficients

40 00 C0 00 E0 00 20 00 C0 00 40 00

X5 Coeff: (chan specific)	0.5	Enter value in S1.15 format [-1 to 0.999969]
X4 Coeff: (chan specific)	-0.5	Enter value in S1.15 format [-1 to 0.999969]
X3 Coeff: (chan specific)	-0.25	Enter value in S1.15 format [-1 to 0.999969]
X2 Coeff: (chan specific)	0.25	Enter value in S1.15 format [-1 to 0.999969]

X-Filter Coeff... : Status

AQSLAC-0	X0	X1	X2	X3	X4	X5
Chan1:	1	-1	0.25	-0.25	-0.5	0.5
Chan2:	1.0	0.0	0.0	0.0	0.0	0.0
Chan3:	1.0	0.0	0.0	0.0	0.0	0.0
Chan4:	1.0	0.0	0.0	0.0	0.0	0.0

[81CA/81CB] Analog Gain and DISN

01110011

Transmit Analog Gain: (chan specific)	AX=0 0 dB gain	[✓ AX=1 6.02 dB gain]
Receive Analog Loss: (chan specific)	AR=0 0 dB loss	[✓ AR=1 6.02 dB loss]

Digital Impedance Scaling Network: (chan specific)

```
DISN=10001  -0.0625
DISN=10010  -0.1250
DISN=10011  -0.1875
DISN=10100  -0.2500
```

Analog Gain and DISN... : Status

AQSLAC-0	AX	AR	DISN
Chan1:	1	1	10011
Chan2:	0	0	00000
Chan3:	0	0	00000
Chan4:	0	0	00000

Figure 2. Desktop showing main window (top center), tear-off command menu (left), two command windows (center) and two status windows (right).

When the user has configured the desired combination of parameter values using the command window, they can write the corresponding data bytes to the SLAC device by clicking the write command button. We divided the command window into three main sections:

- An entry widget that displays the data in either binary or hexadecimal form. The user has the option of specifying the data byte by typing into this entry widget.
- Two command buttons, each sending a particular command op-code to the SLAC device. Typically these buttons correspond to read and write operations.
- A series of widgets on a scrollable frame that display the parameters programmed by the data bytes and allow individual modification of each. The scrollable frame, created with the canvas widget, makes it possible for the user to display the parameter of interest, even if the window is resized to be smaller. The user may thereby optimize their use of screen real estate.

Figure 2 shows the main window with the pulldown menu torn off, two of the possible command windows open on the desktop, and two status windows. We will discuss status windows in section 5.

Some other development tool kits require that an image be manually created for each window in the application. Instead, we wrote a data driven Tcl procedure that is used to build all of the command windows. An introspective mechanism queries the data store to determine the desired data format, the widget types, and all other information to place in the window. Since the data store defines hundreds of variations on the basic window, we avoid an enormous development and maintenance burden through this approach. Should a need arise to modify the presentation of the command windows, the edits to the one Tcl procedure will automatically apply to all of the command windows for all the products supported on WinACIF.

The write command button executes the acifDut command with arguments defining the operation and the data bytes. The acifDut command abstracts all the details of communicating with the hardware and the maintenance of the data structures. The first argument to acifDut is a command mnemonic that identifies the operation. The second argument is the user-specified data bytes. The C code that implements acifDut uses Tcl’s hash table facility to map the mnemonic, the first argument, to a pointer referencing the associated information within the data store. This provides the hexadecimal op-code as well as information needed to validate the data bytes, and to parse them into bit fields corresponding to the various parameters.

5. Maintaining and Displaying SLAC Device State

Interacting with a device with over 700 programmable parameters would be impossible without a means of recalling and displaying the current settings. The ability of some SLAC devices to support multiple telephone channels further complicates the tracking of the configuration variables. Some parameters have a channel-specific scope, such as the gain of an individual channel, while others have a global scope within the device, such as the choice of an incoming clock frequency. The data store contains information concerning the scope of each parameter.

When commands are sent to the SLAC device, the C code responsible for interfacing to the hardware also updates a Tcl array called DUT (Device Under Test). This array records the values of all of the SLAC device parameters programmed by the user through the command windows. DUT's element names correspond to the various programmable parameters of the SLAC device. We made DUT read-only to insure that the content accurately reflects the state of the SLAC device. The Tcl script cannot alter the contents of DUT except by executing the command which programs the SLAC device. We store the parameter values in the array DUT in a format meaningful to the user. To convert the bit-field values used by the device into an understandable format, such as decibels, amperes, ohms, etc., the C code uses data supplied by the data store.

Using this strategy, we can easily create a display of the current device configuration. We merely set up a label widget that displays the desired elements from the DUT array. The main menu of WinACIF provides the user with the ability to view these status windows on the desktop (Figure 2). These windows show the current value of a parameter or values of a group of related parameters across all of the channels in the device.

6. Variable Traces and Bindings

In the implementation of the command window, Tcl’s variable trace feature provides a convenient mechanism for maintaining the correct relationships between the widgets and the underlying data structures. When the window is created, temporary variables are created for the parameter modification widgets in the bottom section of the window. These variables are initialized with the appropriate parameter values from the DUT array. The temporary variables allow the user to setup any combination of changes to the parameter values, independent of the actual, read-only values stored in the DUT array. Variable traces on the temporary variables trigger the execution of code that maintains the data byte display in the top of the window. A Tcl command extension is called that uses information from the data store to convert the set of parameter values into their corresponding bit-fields and assemble these bit fields to create the data bytes.

A user may also type into the data byte entry, thereby triggering an update of the parameter values displayed in the widgets below. The user may then write the new data bytes to the SLAC device by clicking on the Write button. This action calls a Tcl command extension that programs the SLAC device and updates the values in the DUT array to reflect the new programming of the device.

As discussed in a later section, the user can also program the SLAC device independent of the command window by executing a script containing the acifDut command. In this case, the values displayed in any affected command window must be updated. Variable traces placed on the appropriate elements of the DUT array achieve this effect. When the acifDut command changes the value in the DUT array, the trace fires, executing code that copies the value into the corresponding temporary variable. This updates the widgets and also fires a trace that executes the code that updates the data byte entry.

We rely on Tcl's variable trace mechanism to maintain the correct relationships among the DUT array, the parameter widgets, and the data byte display. The event-driven nature of this code makes good documentation essential.

7. Graphical Board Control

Tcl's canvas widget provided us the opportunity to represent signal flow in a graphical format that is intuitive to our users. The SLAC devices may use one of several different interfaces to transmit data. WinACIF depicts a particular interface by drawing a graphical representation of the signal flow and switching arrangements on a canvas widget. Figure 3 shows one of the possible interfaces. This figure depicts the SLAC device transmitting and receiving a serial bit stream that is logically grouped into 8 bytes. The bytes labeled B1 and B2 consist of digitized audio data that can be captured for conversion to parallel form and routed to or from a piece of telecom test equipment. The window provides interactive control over the evaluation board hardware that performs the signal routing and conversion.

Rather than embedding radio or checkbutton widgets in the canvas, we tagged groups of lines and assigned bindings to the tags to reconfigure the evaluation board. For example, the row of arrows pointing up from the top row of bytes works together to comprise a one-of-four selector switch. Clicking on one of these arrows changes the selection by executing an AMD-created Tcl command that programs the appropriate hardware register. The Tcl command also updates a read-only array, the acifBoard array, that tracks board state. The black arrow indicates the current choice.

The Tcl code executed by the binding event can be as elaborate as needed to achieve the desired effect. In the bottom row of bytes in Figure 3, each of the B1 and B2 boxes has two arrows feeding into it. These arrows comprise a "one, the other or neither" selector switch. These kinds of switch arrangements make intuitive sense to the engineers who use WinACIF.

The need to write a program to draw on the canvas rather than using an interactive "what-you-see-is-what-you-get" drawing tool often dismays newcomers to Tcl/Tk. In reality, we find defining the graphic programmatically is an asset. The interface depicted in Figure 3 has an alternate mode of operation that transmits 16 rather than 8 bytes. The Tcl procedure draws the graphic based on the number of bytes passed as an argument. This approach makes it easy to redraw a different arrangement.

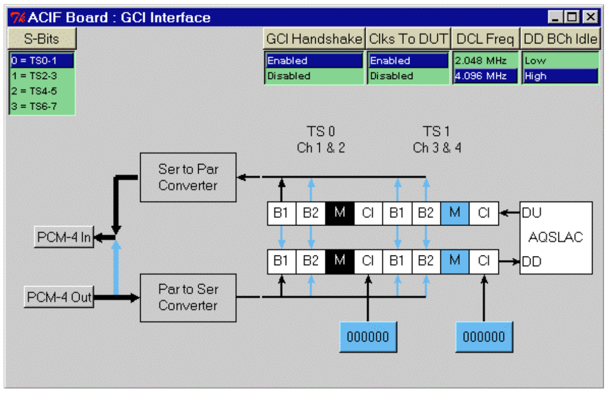


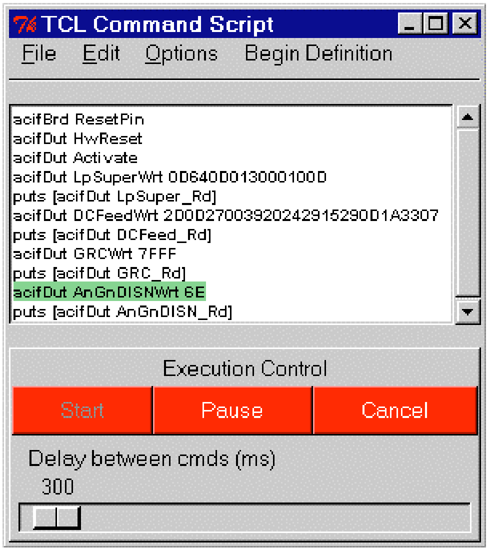
Figure 3. Board programming window

8. User Scripting: More Than a Macro

The user interactions with WinACIF discussed so far all operate directly on the programming of the SLAC device or on the ACIF hardware. Testing of a particular linecard configuration may require the programming of hundreds of commands. The window in Figure 4 provides the ability to reproduce this set of operations at a later time without laboriously repeating the exact sequence of mouse clicks and button presses.

As the user interacts with the other windows of the GUI, this window can capture the Tcl commands that program the SLAC device and the ACIF board. The user does not need to know the command names or how to specify their arguments. Note, however, the command names and arguments have been designed to be self-explanatory to users familiar with the SLAC device.

We based the Command Script Window on the text widget. The user can modify the commands using cut and paste operations or by placing the insertion cursor into the middle of the sequence and inserting additional commands by interacting with the other windows of the GUI. The user can replay the script from this window with a user variable delay between commands and insert breakpoints into the script. We provided an option that repetitively sends the commands in a loop useful for debugging certain aspects of SLAC device operation. The user may save and load these command sequences as files.



```
acifBrd ResetPin
acifDut HwReset
acifDut Activate
acifDut LpSuperWrt 0D640D013000100D
puts [acifDut LpSuper_Rd]
acifDut DCFeedWrt 2D0D27003920242915290D1A3307
puts [acifDut DCFeed_Rd]
acifDut GRCWrt 7FFF
puts [acifDut GRC_Rd]
acifDut AnGnDISNWrt 6E
puts [acifDut AnGnDISN_Rd]
```

Figure 4. Command script window

The above describes more than a macro and replay mechanism because the user is saving an independent Tcl script. For the engineer who requires more sophisticated control, this provides an avenue for writing Tcl scripts that can completely automate testing processes. Certain testing scenarios may require looping, conditional branching, or file IO. He can use the full power of the Tcl interpreter to solve these problems. Tcl's simple syntax, which in many areas is similar to C, reassures the users that the learning curve will not take too much time away from already busy schedules. The Tcl extensions implemented in WinACIF abstract the details of communicating with the SLAC device and the ACIF hardware. Each engineer can now craft customized scripts, share those scripts with others, and even submit a valuable addition for incorporation into WinACIF.

9. Conclusions

The port of Tcl/Tk to the Windows' platform greatly accelerated the evolution of the WinACIF software from a MS-DOS to a Windows application. The difficulty and time required to learn the basic skills to be a proficient Windows developer has barred most engineers from developing their own GUI-based applications. The conventional toolkits make GUI development a field for software specialists. Charles Petzold says it best,

> "…please do not fear that you are missing some vital part of your brain that is required to be a successful Windows programmer. This initial confusion is normal, and don't let anyone tell you differently. Windows-based programming is strange. It's weird, it's warped, it's awkward, it's convoluted, it's mind-boggling…"[1]

We find Tcl/Tk to be an empowering tool which bypasses most of these difficulties thereby drastically shortening the learning curve and the development time required to build GUI-based applications. For engineers whose focus has been the application domain, Tcl/Tk's ease makes Windows development possible.

By defining the GUI programmatically using Tcl/Tk script, we structured this application in a more powerful and flexible manner. The ability to extend Tcl with C code allows implementation of complicated data structures and interfaces to hardware. Dynamic reconfiguration for various products and interfaces could then be implemented, consolidating support for the entire product family in one program. The application is readily extendable to provide support for future products. Additionally, Tk's canvas widget provides a powerful tool for delivering information in a graphical form familiar to our users.

10. References

[1] Petzold, Charles. *Programming Windows 95: The Definitive Developers Guide to the Windows 95 API*. Microsoft Press, 1996. ISBN 1-55615-676-6.

Appendix A : Tcl Command Extensions used in WinACIF

ACIF.DLL	
defines the following Tcl command extensions related to programming the SLAC device:	
AcifProducts	Returns information describing the SLAC products and interfaces that are supported by WinACIF. Used to select the product/interface and configure WinACIF accordingly.
acifDut	Returns information describing the command set of the SLAC device. Writes or reads SLAC device control interface and updates Tcl array DUT which tracks the state of the SLAC device.
AcifDutEntryCheck	Checks validity of data byte string for proper number of binary or hexadecimal digits.
AcifDutEntryFromVals	Converts parameter values to bits and combines them to create data byte string.
AcifDutEntryToNewVals	Fractures data byte string into parameter values and updates temporary variables for command window widgets.
AcifDutEntryToHex	Converts value to hex form without spaces.
AcifParmEntryCheck	Validates the data when an entry widget is used to enter a parameter value.
acifExit	Cleans up DLLs and exits.
acifDown	Loads the proper DLL for downloading a WinSLAC file.
AcifSelectPort	Returns the list of available communication ports. Selects a port to use.
AcifFileStatus	Returns the compilation date and time of WinACIF's currently loaded DLLs.
AcifBinToVal	Converts a string of binary digits to a specified decimal format.
AcifValToBin	Converts a decimal value to a string of binary digits according to a specified format.
PAUSE	Executes pause when encountered in a user-defined script.
PCM.DLL	
defines the following Tcl command extensions related to controlling the interface board when implementing a PCM/MPI interface:	
acifBrd *option value*	Programs *value* into the ACIF board hardware. Updates the corresponding element of the acifBoard associative array. *Option* identifies one of many registers contained in the FPGA logic. Each of these registers controls some aspect of the operation of a PCM/MPI interface.
GCI.DLL	
defines the following Tcl command extensions related to controlling the interface board when implementing a GCI interface:	
acifBrd *option value*	Programs *value* into the ACIF board hardware. Updates the corresponding element of the acifBoard associative array. *Option* identifies one of many registers contained in the FPGA logic. Each of these registers controls some aspect of the operation of a GCI interface.