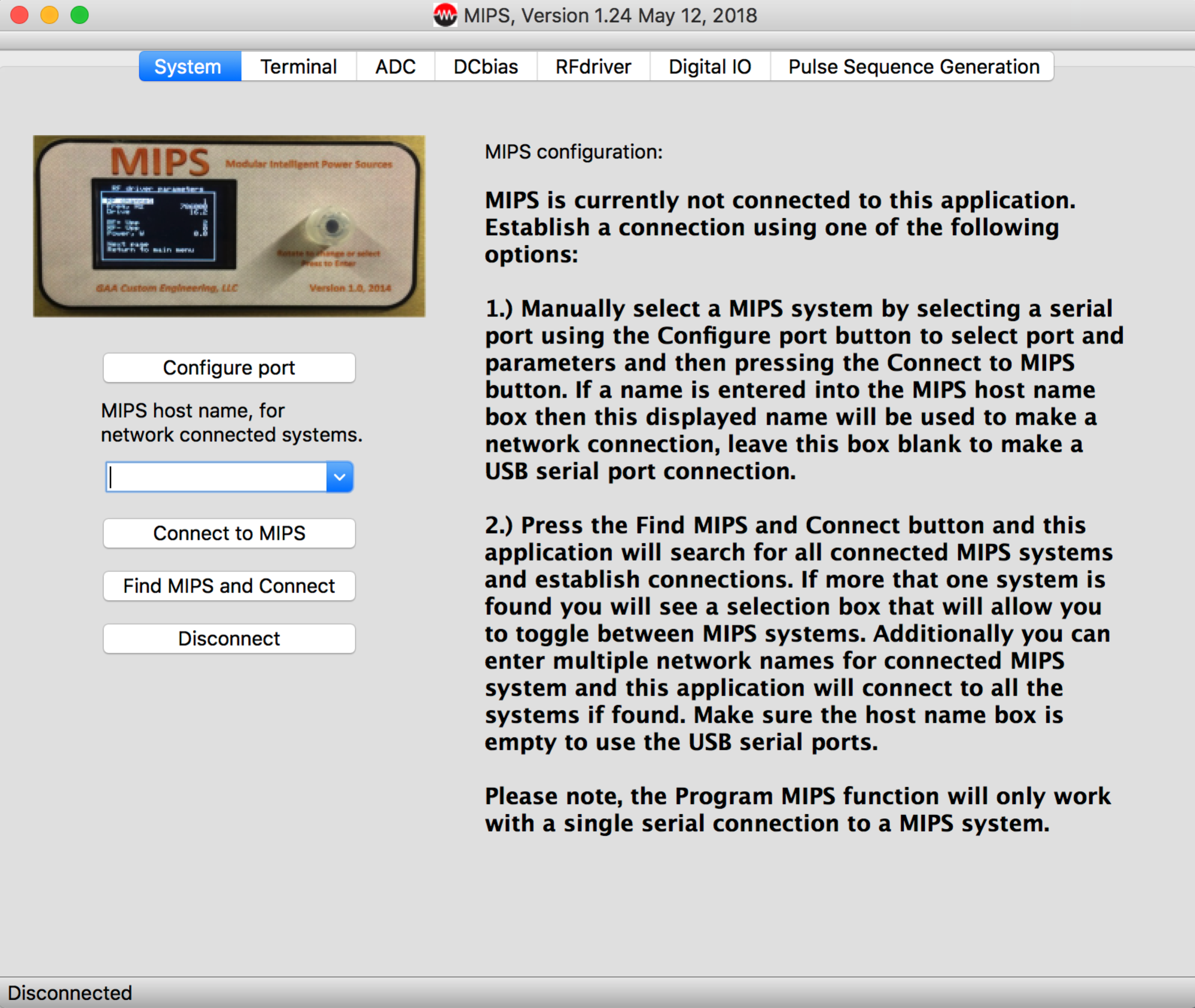
**MIPS Host Application**

**February 9, 2020**

**GAA Custom Electronics, LLC**



The MIPS host application is designed to communicate with a MIPS controller or multiple MIPS controllers. It allows you to monitor and control parameters in all the connected controllers. The MIPS host application supports two ways of interacting with your MIPS systems:

1. When the MIPS application connects to your system(s) you will see a series of tabs that allow you to control individual modules in your system(s). Much like the MIPS controller’s users interface it allows you to control each module. This mode of operation allows you to select a tab that represents a module and allows you to interact with the modules parameters. Most but not all of the MIPS modules are supported with tabs. You can also select the terminal tab and send commands to the connected MIPS system.
2. The MIPS application under the Tools menu selection allows you to display control panels. A control panel is an organization of MIPS controls that make sense for your specific application. A number of control panels have been included as examples. The user can define unique control panels including the ability to define your own graphics background. The custom control panel mode also supports multiple MIPS systems.

Below is a brief overview of the tab options on the main dialog box. Please note that when connected only the tabs for modules in your system will be shown:

System

This tab contains options for connecting to MIPS. When the app starts it is not connected and you will see a message indicating how to connect. MIPS can be connected to your host computer using a USB interface, WiFi, or an Ethernet connection. All MIPS systems have a USB interface, WiFi and Ethernet are options. If using a USB interface the host computer “sees” MIPS through a virtual com port. You will need to have drivers installed on PCs but MACs will work without any additional setup. The appendix in the help file gives more details on setting up this app. Use the configure port button to popup a settings dialog box that will allow you to select a com port, select the port and set the baudrate at 115,200. Do not change any other parameters. Press Apply to accept the changes and the dialog will close. You can now press Connect and the MIPS app will try to connect with MIPS and display the modules detected. If connecting via WiFi or Ethernet you can enter an IP address or name into the MIPS host name box. If this box is not blank then the app will attempt to connect via TCP/IP.

The MIPS application can connect to multiple systems. You can press the “Find MIPS and Connect” button and then the application will scan all the comm ports looking for MIPS controllers. A selection box will appear if multiple systems are found and then you can select the system you want to interact with. The selection box will list the MIPS controllers found by name. If the MIPS host name selection box has system IPs or names listed then this information will be used to make the connections.

ADC

The Arduino Due has several ADC channels that can be used for data collection. If your system has an ADC BNC on the rear panel then this tab will allow you to perform high-speed data collection from this input. The ADC input is 0 to 5 volts DC so if you have a signal with different range you will need to first rescale it with hardware. This option allows you to read at high rates (up to 600 KHz) and record very long vectors.

Terminal

The terminal window allows you to communicate with the MIPS system using any of the commands defined in the manual. The MIPS commands can also be found by selecting the “MIPS Commands” option under the Help menu. Remember the MIPS commands are case sensitive. The Save menu option will allow you to save all the data in the terminal window to a text file for documentation purposes. The Load menu option will load a test file and display its contents on the terminal window and also send the data to MIPS as well as displaying MIPS response. This allows you to define a setup script that can be sent to MIPS to simplify setup. You can paste commands from a text file into the terminal window to send a batch of commands to MIPS.

Digital IO

This tab allows monitoring and control of the digital IO in a MIPS system. The save and load menu options allow you to save the current state and load a previously saved state. The MIPS app does not automatically scan the input or output status, you will find an update button on this tab that will update all the parameters shown. Note that this tab shows all the digital IO that is possible on a MIPS system, your system likely has fewer digital IO option available for connection.

MIPS Trigger output can be used for a number of options including generation of a clock. This tab allows control of the Trigger output options.

The MIPS user interface can be controlled remotely using options on the tab, this is useful if you have a custom MIPS system with no control knob or you have floated your system and need to interact through the controllers UI.

DCbias

This tab allows monitoring and control of the DC bias outputs of your MIPS system. A total of four modules can be installed allowing up to 32 total channels. Controls on this tab allow you to monitor up to 24 of the channels and control the output voltages. This tab allows you to manually update the displayed parameters or you can select an auto update option to the parameters to update every few seconds. The DC output voltages can be changed by typing in a new value and pressing enter or by using the up and down arrow keys to change the values. If you hold the shift and or control button down the size of the voltage step will increase.

DC bias ground can be defined by entering a unique name the Group box for all the channels you with to group together. When you change one member of a group then all channels in the group will change by the same amount.

RFdriver

The RFdriver tab allows you to set the RF drive frequencies and drive levels for the number of modules in the system. You can also monitor the RF output level. An update button is provided to refresh all the controls displayed on this tab. Auto tune and Auto Re-tune buttons allow you to automatically tune a RF head to its load.

Auto tune will set the drive level to 10% and tune the RF heading by scanning the frequency range.

Auto retune will keep the current power setting and search around the current setting to fine the optimal setting.

Twave

This tab allows you to control all parameters for a dual channel Twave system. Additionally, all the compressor parameters are displayed and can be controlled including the multi-pass compression table. The Twave tab has context sensitive help so if you select help when this tab is displayed you will see instructions on how to define the compression parameters and how to use the multi-pass compression table.

ARB

This tab allows you to control all parameters for an up to your channel ARB system. Additionally, all the compressor parameters are displayed and can be controlled including the multi-pass compression table. The ARB tab has context sensitive help so if you select help when this tab is displayed you will see instructions on how to define the compression parameters and how to use the multi-pass compression table.

FAIMS

The FAIMS tab allows you to enable and disable FAIMS and monitor the output voltages. All the FAIMS tuning and setup is performed using the FAIMS controller's user interface. The FAIMS tab allows you to monitor and control the CV and BIAS voltages as well as the offset value. The offset value allows you to define the range for CV and BIAS. Changing offset will not change the output voltage, just the range. With offset set at 0 the CV and BIAS channels have a range of -250 to 250V. The offset will allow you to center this range about any value from -250 to 250V. For example if you set the offset to 100V then the range will be -150 to 350V.

The FAIMS tab also supports three CV scanning option:

1. CV parking feature that is coupled with a LC separation. You can define a list of retention times and windows with their associated CV and BIAS values. The system will automatically adjust the CV and BIAS values as the LC separation progresses. Please see the context sensitive help in the FAIMS tab for more detail.
2. Linear scanning. This function allows you to define a starting and ending CV voltage and duration in seconds for the scan. When the scan starts the MIPS FAIMS system will automatically scan the CV from the starting voltage to the ending voltage in a linear fashion. Additionally, the defined scan can be repeated the number of times defined in loop. The scan can be started manually or via an external.
3. Step scanning. This function allows you to define a starting and ending CV voltage, number of steps, and step duration for the scan. When the scan starts the MIPS FAIMS system will automatically scan the CV from the starting voltage to the ending voltage in a stepwise fashion. Additionally the defined scan can be repeated the number of times defined in loop. The scan can be started manually or via an external trigger.

Pulse Sequence Generation

One of the most powerful features of MIPS is its ability to define complex pulse sequences. You can define pulse sequences that change digital outputs and DCbias outputs at user defined times. You are able to define an internal clock or use an external clock to control the pulse sequences and you trigger the execution using a software command or an external hardware trigger. This tab allows defining downloading and executing pulse sequences. This tab has context sensitive help that will define all the options in more detail.

The MIPS application also contains a menu bar with a number of useful options described below:

Help

The Help menu heading has both a general help selection and a MIPS commands option. The general help will display help for the specific tab selected or for the control panel if one is displayed. The MIPS command help option will list all MIPS commands and these can be entered using the Terminal interface by selecting the Terminal tab.

Under the Help menu option you will also find a Properties option, if you select this option you will see a dialog that allows you to select a number of options that define the behavior of the MIPS system, please see the properties section of this document for more details.

File

The File menu heading contains two options, Load and Save. These functions allow you to load and save parameters to a file on your computer. These functions will load and save parameters on the tab you have selected or if you have one of the control panels displayed the load and save will apply to the control panel. When you select load or save you will see a file selection dialog allowing you to define the file of your choice.

Terminal

When the terminal tab is select a number of options in the Terminal menu selection will be available and described below:

Clear

Select this option to clear the Terminal screen, this will remove all displayed text / message history.

Message repeat

Selecting this option will display a dialog box that allows you to define a message to be sent every second to MIPS.

Get file from MIPS

The MIPS controller has a SD memory card to save configuration data. This command allows you to read a file from the SD card to a location on your host computer. You can use the DIR command to see all the file names saved on the SD card.

Send file to MIPS

This command allows you to write a file to the SD card from a selected file on your host computer. Note, this function will over write a file with no warning!

Read EEPROM

Each module in your MIPS controller has an EEPROM used to save the modules calibration and configuration parameters. This command allows you to select a module and save the data to a file on your host computer.

Write EEPROM

Each module in your MIPS controller has an EEPROM used to save the modules calibration and configuration parameters. This command allows you to write data from a file on your host computer to the select module’s EEPROM.

The following three options are designed to communicate with the Arduino; Due that is located on the ARB module. This can be done by connecting the MIPS application directly to the ARB Arduino board or by using the TWITALK command. The TWITALK command redirect communications traffic through the MIPS controller to the select ARB module. Note, these commands were designed for factory setup and test and you should not need to use them.

Read ARB Flash

ARB modules have an Arduino Due on each module. This Due stores its configuration data in its Flash memory. This command allows you to read and save the Flash contents to a file on you host computer.

Write ARB Flash

ARB modules have a Arduino Due on each module. This Due stores its configuration data in its Flash memory. This command allows you to write from a file on your host computer to the Flash on an ARB module.

ARB Upload

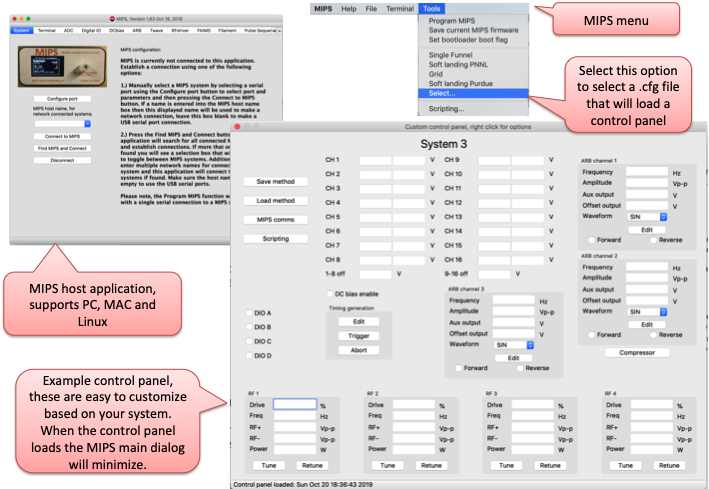
This command allows you to update the program on the ARB Due from a selected file on the host computer.

Tools

The Tools menu heading contains functions that enable loading new firmware into the MIPS controller as well as control panel options. Appendix B outlines the steps to install new firmware in MIPS. The control panels are a visual display of a system with MIPS controls positioned at logic locations and labeled for their specific application. At the time of this writing there are several custom control control panels developed for various customers. We will work with you to develop a custom control panel (usually at no cost) or you can use the user defined control panel option. The Select… menu option allows you to select a user defined control panel that can be defined by a simple text file. When you press this select option you will see a file selection dialog asking you to select a .cfg file. This file defines your custom control panel. Please see Custom Control Panel section of this document for more details on this file format.

**Custom Control Panels**

User defined control panels are created by a simple text file that the user generates. This text file contains instructions that the MIPS application uses to create a control panel that describes your system and how the MIPS signals are used to control your instrument. MIPS control panels can integrate signals from multiple MIPS systems; this allows you to create very complex systems. Control panels display a single page or screen but you can create several control panels for a single system if you wish and you can nest control panels.



To load a control panel select the “Select…” option under the Tools menu and then select your configuration file, extension .cfg is recommended. MIPS will then open this file and create the control panel.

Make sure you have first connected to your MIPS systems using the “Find MIPS and Connect” button on the System tab. This will search your ports and connect to your MIPS systems. This function will read the MIPS system names and these names will be used in map the signals from the name you define to the specific MIPS box. So make sure you have defined unique names for each of your MIPS systems.

After you have selected your control panel file the main MIPS dialog will switch to the System tab and minimize. You should not use the main dialog when running a control panel.

You can create a control panel configuration file using any text editor to enter the configuration commands defined below. The MIPS app install files include an example configuration file named example.cfg. This example cfg file illustrates most of the features and provides a framework you can use to build your own control panel. If you are using the MAC version of the MIPS application you will find the examples in the MIPS application bundle in the examples folder. We recommend you copy the examples from the app bundle to a folder on your MAC.

Below are the commands that you can use in the control panel .cfg file:

* *#*, any line starting with a # character is considered a comment and ignored.
* *size,x,y* defines the size of the control panel in pixels
* *image,filename.jpg* defines a background image, make sure you include a full path to the file. The image is stretched to fit the size of your control panel.
* *Help,filename* define a full path and file name to a text file that contains help information for the control panel you are creating. This help file can be displayed by right clicking your mouse on the control panel and then selecting the “This control panel help” option. You can use any text editor to create the help file and include information specific to your application.
* *TextLabel,this is a label,font,x,y* This command will place a label at a user defined location and use the font size defined.
* *Shutdown,button name,x,y* This command will add a System shutdown button to your control panel at the location defined. Pressing this button will turn off all DC bias voltages and reduce all RF drive level to 0.
* *SaveLoad,save name, load name,x,y* This command will add save and load buttons to the control panel. This allows you to save and restore all your system settings.
* *InitParms,filename.ini* This command will open the filename and execute all the MIPS commands found. This is intended to perform setup when the control panel loads. The MIPS commands in the file must be formatted as follows: MIPSname,command.
* *SendMessage,mname,command* This command allows you to send a message string to the MIPS system named *mname*. You can send any valid MIPS command that is expected to return a value. This command will return the parameter requested by *command* or “?” if the request cannot be processed.
* *SendCommand,mname,command* This command allows you to send a message string to the MIPS system named *mname*. You can send any valid MIPS command that is not expected to return a value. This command will return nothing if the *command* is processed or “?” if the request cannot be processed.
* *MIPScomms,x,y* This command will add a button to your control panel that when pressed will allow you to send commands to a select MIPS system.
* *DCBgroups,x,y* This command places a DCbias groups box on the control panel. This box allows you to define DC bias channels that are group together; changing one member of a group will change them all by the same amount.
* *Script,x,y* This command places a scripting box on your control panel. This allows you to define a script to control your instrument.
* *Cpanel,name,control panel file,x,y* This command will place a button on the control panel that will allow you to load a new control panel when the button is pressed. Name defines the text in the button, control panel file is a full system path and file name that points to the control panel configuration file and x,y define the button location.
* *RFchannel,name,MIPS name,chan,x,y* This command defines a RF channel, its name, the MIPS system name, the MIPS system RF channel number, and its x,y location. The command will place an RF channel control box on your control panel. This control box will allow you to set the RF parameters and monitor the RF levels and power.
* *RFCchannel,name,MIPS name,chan,x,y* This command defines a RF channel, its name, the MIPS system name, the MIPS system RF channel number, and its x,y location. The command will place an RF channel control box on your control panel. This control box will allow you to set the RF parameters and monitor the RF levels and power. This command include the option for closed loop control of the RF channel.
* *DCBchannel,name, MIPS name, chan,x,y* This command defines a DC bias channel, its name, the MIPS system name, the MIPS system DC bias channel number, and its x,y location.
* *DCBoffset,name,MIPS name,x,y* This command defines a DC bias channel offset control, its name, the MIPS system name, and its x,y location.
* *DIOchannel,name,MIPS name,chan,x,y* This command defines a DIO channel, its name, the MIPS system name, the MIPS system DIO channel number, and its x,y location.
* *ESIchannel,name,MIPS name, chan,x,y* This command defines a ESI channel, its name, the MIPS system name, the MIPS ESI channel number, and its x,y location.
* *ARBchannel,name,MIPS name,chan,x,y* This command defines a ARB channel, its name, the MIPS system name, the MIPS ARB channel number, and its x,y location.
* *Compressor,name,MIPS name,x,y* This command will place a button on the control panel that will display a Compression dialog box when pressed. This dialog box will allow you to control all the SLIM compression parameters including multi-pass compression tables. You define the button *name*, the *MIPS name* where the signals are generated, and its *x,y* location.
* *IFT,name,MIPS name,x,y* This command display an IFT (Ion Trapping Funnel) dialog box on your control panel. This allows you to generate the timing needed to control a trapping funnel. You define the IFT name, the MIPS name where the signals are generated, and its x,y location.
* *Grid1,name* This command has to follow the IFT command. This command defines the name of the signal used for Grid 1.
* *Grid2,name* This command has to follow the IFT command. This command defines the name of the signal used for Grid 2.
* *Grid3,name* This command has to follow the IFT command. This command defines the name of the signal used for Grid 3.
* *Enable,chan* This command has to follow the IFT command. This command will cause the IFT timing generation to include an enable signal to go active during the acquire interval. The chan is the MIPS channel number, i.e. A
* *Timing,name,MIPS name,x,y* This command defines a pulse timing generator that is more general purpose than the IFT generator described above. This timing generator allows the user to define multiple events that control dc voltages and digital output signals. You should only define this timing generator or the IFT, never both in the same control panel. You define the timing control *name*, the *MIPS name* where the signals are generated, and its *x,y* location. You will see a small dialog on your control panel with three buttons for editing the timing (pops up a new dialog), trigging, and aborting a timing sequence.
* *EventControl,name,Ename,x,y* This control allows you to edit an event in the Timing Generator described above. Editing the event means you can set the voltage of a timing event that you defined in the timing generator. You can define multiple event controls and the event control statements must directly follow the timing generator statement you wish to reference. All of the event controls that follow a timing generator statement will reference its events using the event names you define. The event control is designed to allow changing event values in the MIPS system while the system is in the pulse generation mode. The parameter *name* defines the control name on your control panel, *Ename* defines the event you are referencing in the timing generator, for example if you have an event named Fill you wish to reference then you have two options for *Ename*, Fill.Value or Fill.ValueOff. These names are case sensitive. When you press the pulse sequence trigger button the event controls will be updated with the current event values that are sent to the MIPS hardware and then you will be able to change this values as needed. The location of the event control on you control panel is defined using the *x* and *y* values.
* *Acquire,program name,filename,tofscans* This command is used to enable a program to acquire data. The parameter “program name” defines the full path and name of an external application that is called to acquire data. If the parameter filename is defined then the user will be asked for a location to save the date and the external program will be passed a filename. If the parameter tofscans is present then the total number of tof scans to acquire is calculated and passed to the external program.
* *RFAMP,name,MIPS name,chan,x,y* This command defines a RFamp channel, its name, the MIPS system name, the MIPS RFamp channel (or module) number, and its x,y location. This will place a dialog box on your control panel that allows control of the RF amplifier as well as the QUAD mass filter.
* *ADCchannel,name,MIPS name,chan,x,y* This command defines an ADC channel, its name, the MIPS system name, the MIPS ADC channel number, and its x,y location. This will place a dialog box on your control panel that allows monitoring the ADC value. Optionally you can add four additional parameters to convert the ADC raw data to engineering units. The additional parameters are *m,b,units,format* the raw ADC value is multiplied by *m* and then *b* is added to convert the value as needed. The parameter *units* will display on the control panel after the ADC dialog box. The string *format* is a c style print format string to allow you to define how you want the data presented in the dialog box.
* *DACchannel,name,MIPS name,chan,x,y* This command defines a DAC channel, its name, the MIPS system name, the MIPS DAC channel number, and its x,y location. This will place a dialog box on your control panel that allows control the DAC value. Optionally you can add four additional parameters to convert the DAC raw voltages to engineering units. The additional parameters are *m,b,units,format* the raw DAC voltage is multiplied by *m* and then *b* is added to convert the value as needed. The parameter *units* will display on the control panel after the DAC dialog box. The string *format* is a c style print format string to allow you to define how you want the data presented in the dialog box.
* *Groupbox,name,width,height,x,y* This command will create a group box, name will define the groupbox label, width and height define the groupbox size in pixels, and x and y define its location in pixels. All the commands that follow this groupbox command will place controls in the groupbox and the x, y locations will be relative to the upper left corner of the groupbox. This capability allows you to make logical collections of controls. When you need to reference a control inside a groupbox you will reference the control by preceding its name with the groupbox name followed by a “dot”.
* *Groupboxend* This command signals the end of a groupbox.
* *TCPserver*,*port number* This command enables TCP/IP connection to the control panel on the port number defined in this command. When this command is defined the MIPS application listens for commands on the port number defined. Most of the controls on your control panel can be read and written by an external system. Please see the TCP/IP Server section of this manual for a detailed discussion of this capability.
* *Scriptbutton,name,Script file name, x,y*  This command allows you to place a button on the control panel that will automatically load and execute a script when the button is pressed. Name defines the text displayed in the button, Script file name is a full path and file name that points to the script file to load and execute when the button is pressed, and x,y define the button location.
* *CallOnUpdate,value.* This command must follow a ScriptButton command. Value can be TRUE or FALSE. If TRUE then this previous ScriptButton will be activated on every update of the Control Panel.
* *Ccontrol,name,MIPS name, Type, Gcmd,Scmd,RBcmd,Units,x,y,Dtype* This command allows you to define various types of controls for virtually any MIPS command. Name is the label for the control; if it’s a button then name will be the text in the button. MIPS name is the name of the connected MIPS system that will send and receive messages for his control. Type defines the type of control you wish to create, LineEdit, CheckBox, Button, or ComboBox. Gcmd is the MIPS command to read the current value of the control from MIPS. If the command contains commas you need to replace them with underscores, these will later be replaced with commas when sent to MIPS. You can leave this entry blank if not used in your control. Scmd is the MIPS command to send the current control value to MIPS. If the command contains commas you need to replace them with underscores, these will later be replaced with commas when sent to MIPS. You can leave this entry blank if not used in your control. RBcmd is the MIPS command to get the read back value for your control. If the command contains commas you need to replace them with underscores, these will later be replaced with commas when sent to MIPS. You can leave this entry blank if not used in your control. Units defines the control units and only applies in LineEdit controls. x,y define the control location. Dtype is optional and defines the data type. This only applies to LineEdit boxes and is double by default, set to String if you want to allow text to be entered into a LineEdit box.
* *ComboBoxList,a,b,c,…* This command defines the ComboBox list of options. This command must follow a Ccontrol command that defines the ComboBox control. The list of options should match the MIPS command syntax for the command you are using.
* *Device,name,filename,units,x,y.* This command will add a control to your control panel who’s data is read from an external device that is connected with a serial com port. The control label is defined by *name*, *filename* is the full path and filename to a file that contains the serial interface details. The label *units* will follow the control and *x,y*, define the location of the control on your control panel.

Device file format

# comment

Port,COM1

Setting,9600,8,n,1

EOL,CR,LF

Init

End Init

Read command,R0

Scan format,%f

M,1

B,0

**Scripting**

The MIPS application allows users to write their own scripts, MIPS provides support for application scripting with ECMAScript.

**ECMAScript** is a trademarked scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262. It was created to standardize JavaScript, so as to foster multiple independent implementations.

Here is a link to a useful Java Script tutorial site: https://www.w3schools.com/js/default.asp

The scripting capability has been customized by exposing the class “mips” to the scripting system. This class has a number of functions that allow you to communicate with the MIPS application and any connected MIPS controllers.

There are two ways to enable the scripting system in the MIPS application; from the Tools menu of the main MIPS dialog or from a custom user control panel if the control panel has enabled scripting. The functions available from the mips class are different depending on where the scripting is enabled.

When scripting is enabled you will see a scripting dialog box popup that will allow you to enter your script into a text window, load a script from a file or save your script to a file. You will also fine a execute button to start you script and an abort button to stop a running script. Below the execute button is a status message area that will display any errors or final results from your script. A simple example is to enter 4+5 into the scripting text window and press Execute, you will see 9 in the status message area.

Use caution writing scripts, do not write a tight loop such as while(true); this will never release the processor and lock up the application. A loop as shown above will not allow the abort button to work because the button press will never be processed. Calling function in the mips class allows the system to process events and enables the abort capability so modifying the tight loop example to while(true) mips.msDelay(1); will not lock the system up and you and press the abort button to exit. If you end all of your scripts with status = “done”; then you will see “done” in the status area when your script finishes.

Below is a list of functions available from the mips class, these functions are always available regardless of where the scripting system was started:

mips. SendCommand(name,cmd);

This function will send a command to the named MIPS system. The function is useful when you have multiple connected MIPS system and allows you to direct a message to a specific system. The argument “name” is a string defining the MIPS system name and cmd is any valid MIPS command string. This function it intended to send commands that do not return data. Example mips.SendCommand(“MIPS-1”,“SDCB,1,10\n”); will set DCbias channel 1 to 10 volts on the MIPS system named MIPS-1.

Returns true if the command was accepted and false if rejected.

mips. SendMess(name,message);

This function will send a command to the named MIPS system. The function is useful when you have multiple connected MIPS system and allows you to direct a message to a specific system. The argument “name” is a string defining the MIPS system name and cmd is any valid MIPS command string. This function it intended to send commands that will return data. Example mips.SendMess(“MIPS-1”, “GVER\n”); will return the current MIPS firmware version string for the MIPS system named MIPS-1.

Returns a string containing the reply from MIPS.

mips. msDelay(delay);

This function will delay for the number of milliseconds defined by the integer delay. This function will not return until the delay time has expired. While your script is delaying the MIPS application will use this time to process events. Example; mips.msDelay(1000);

mips.statusMessage(message);

This function will display a message on the status bar of the main MIPS dialog. This is the bottom line of the MIPS application dialog. This is useful for your script to communicate its status to the user.

mips.popupMessage(message);

This function will popup a dialog box containing the message defined by “message”. This popup will stop script processing and wait for the user to press the “OK” button.

mips.popupYesNoMessage(message);

This function will popup a message with Yes and No response buttons. The message displayed is defined by “message” and should be in the form of a yes or no questions. This function will not return until the user and pressed the yes or no button.

mips.popupUserInput(title, message);

This function allows a script to popup a user input dialog box. This function allows you to define the title of the dialog box and a message asking for the type of input you expect from the user. This function returns a string that contains the user response.

Below is a list of functions available from the mips class that are only available when the scripting system is started from the MIPS Tool menu:

mips. SendCommand(cmd);

This function will send a command string to the connected MIPS system. Any valid command string can be sent to a MIPS system. Make sure you terminate the command with the end of line character \n. This function it intended to send commands that do not return data. Example mips.SendCommand(“SDCB,1,10\n”); will set DCbias channel 1 to 10 volts.

Returns true if the command was accepted and false if rejected.

mips. SendMess(cmd);

This function will send a command string to the connected MIPS system. Any valid command string can be sent to a MIPS system. Make sure you terminate the command with the end of line character \n. This function it intended to send commands that will return data. Example mips.SendMess( “GVER\n”); will return the current MIPS firmware version string.

Returns a string containing the reply from MIPS.

Below is a list of functions available from the mips class that are only available when the scripting system is started from a custom control panel:

mips.Save(filename);

This command will save the current settings on your control panel to a method file that you define by filename. The argument filename should include the full path to the file you wish to save. All of the settings are saved to filename and this file is in text format so you can read it with any text editor. The settings saved include all the voltages, RF frequencies and drive levels.

This function returns a status string indicating the results of the save operation.

mips.Load(filename);

This command will load a settings file and update all the parameters on your control panel. The filename define the full path and file name to the settings file you wish to load. After all the parameters have been loaded the system will delay for 3 seconds to allow everything to stabilize before resuming screen updates.

This function returns a status string indicating the results of the load operation.

mips.SystemShutdown();

This function is designed to put your system is a safe shutdown state while saving the operating settings before shutting down. This function will disable the DCbias voltages by turning off the supply and also set all the RF drive levels to zero as well as setting ESI voltages to zero. The system will pause screen updates for 3 seconds after the function is called.

mips.SystemEnable();

This function will restore the system from a shutdown state. When the system is shutdown all the operating settings are remembered, this function will restore those operating parameters and enable system operation. The system will pause screen updates for 3 seconds after the function is called.

mips.Acquire(filepath);

This function is designed to start a data acquisition and acts similar to pressing the Trigger button on the IFT control box on your control panel. This function requires your control panel to have defined the IFT option. The filepath parameter defines where the files will be saved from the acquire operation. If the filepath is already present then the final folder name in the path will have a sequence number added or advanced to insure the path is unique. It is required that the filepath is present up to the final folder name. After the filepath is created the current settings will be saved to the defined path and the Acquire application will be started, this application name is defined in your control panel configuration file. The console IO to and from the Acquire program is directed to a dialog box that will appear after this function is called. The MIPS application will wait for the Acquire application to issue a “Ready” message and then the pulse sequence will be started causing the data collection to begin. The Acquire program is a command line application and it can communicate with MIPS by printing output to its console. The Acquire application can plot data on the displayed console dialog by issuing a plot command and then following it up with a plot data. An example control panel and Acquire program is included with the MIPS app, this is a c++ program written is QT and illustrates how the acquire program needs to be designed. After the Acquire program finishes the contents of the acquire console is saved to the defined filepath. This function does not block and will return to your script after the call is made and the process is started.

This function will return true if the process started properly or false if any error is detected.

mips.isAcquiring();

This function will allow your script to test if the Acquire program is currently running. The Acquire function defined above does not block so you can use this function is a loop to wait for the acquire application to finish.

This function returns true if Acquire is running else false is returned.

mips.DismissAcquire();

This function will destroy the Acquire program popup console IO dialog box. This console IO dialog box is not automatically destroyed when the Acquire application finished so this function should be used to remove the dialog.

mips. UpdateHalted(stop)

This function will enable or disable the control panel parameter updating. If stop is true then updating will be stopped, if false updating will be enabled. Note, if you exit your script or abort with updates halted they will not automatically be re-enabled.

mips. WaitForUpdate()

This function will wait to the next control panel parameter update to complete before returning.

mips. ReadCSVfile(filename,delimiter)

This function will open the file passed by filename and read its contents. The delimiter parameter defines the character used to separate variables on each line of the file. This function returns the number of lines read from the file.

mips. ReadCSVentry(line,entry)

This function allows you to read a specific entry from a CSV file. Before calling this function you must first load the CSV file using the ReadCSVfile function. The line and entry parameters define the parameter that will be returned. The first line in the file is defined as 0 and the first entry on a line is entry 0. The returned value is a string and the string is empty for invalid lines or entries.

mips. Command(cmd)

This function allows you to read and write any value on your control panel. You can reference the TCP/IP Server section of this manual to see the syntax used to read and write values, this syntax is identical to the TCP/IP communications. The cmd parameter is a string that contains the command and this function returns a string. If you have a control panel value named Inlet you can read its value by setting cmd to “Inlet”, you can set the value to 10 by setting cmd to ”Inlet=10”.

mips. CreatePlot(Title,Yaxis,Xaxis,NumPlots)

This function allows you to create a plot. This command will initialize the plot by defining its title, the Yaxis and Yaxis names and the number of plots you want to generate. NumPlots can be set to 1 or 2. NumPlots defines the number of parameters that will be shown on the plot.

mips. PlotCommand(cmd)

This function allows you to plot data after the CreatePlot function has been called to initialize a plot. A single plot can contain several graphs and you can use the right and left arrow keys to scroll through the graphs. The plot also contains a menu with a number of options including help, right click anywhere on the plot to see and select an option. The cmd parameter is a string and this string can contain a number of plot commands that are outlined below:

NormalCursor

This command will set the cursor to normal mode.

Refresh

This command will re-plot the currently data.

Save,filename

Saves all the plot data to the filename you define.

Load,filename

Loads a plot file and displays the data.

Scan,parm,start,stop

A plot can contain a series of scans. This command defines the parameter (parm) that is being scanned from the starting value (start) to the terminal value (stop). This command will label the individual scans in a plot file.

Xrange,Time

The Xrange command defines the plots axis and additionally the plot mode. If the Xrange,Time command is issued the plot mode will be time on the X axis and will act as a strip chart recorder. The X axis will be adjusted as needed when additional data points are added to the plot.

Xrange,min,max

This command defines the range for the X axis of a plot. Data points will be added to a plot between the minimum (min) and maximum (max) values entered.

Xrange,min,max,dmin,dmax

This Xrange command allows you to map data with X axis values between min and max to a displayed value range of dmin to dmax. This allows you to map the raw X axis value to a display range of your choice.

Yrange,min,max

The Yrange command allows you to define the Y axis plot range for the data you will be ploting.

Clear

Clears all the plot data.

Newgraph,num,length

After the plot is created you need to add graphs or plots that will display your data. This command allows you to define the number of plots, 1 or 2, and the length of the plot in number of points.

PlotPoint,time,val1,val1

This command allows you to add a point to a time plot defined with the XRange,Time command. As points are added the X axis range will automatically be adjusted. This allows the display it act as a strip chart recorder.

Addpoint,num,x,y1,y2

This command add points to a plot at the defined X and Y axis location.

Plot1,name

The plotting capability allow up to two plots to be display on a graph. This command allows you to name plot 1.

Plot2,name

The plotting capability allow up to two plots to be display on a graph. This command allows you to name plot 2.

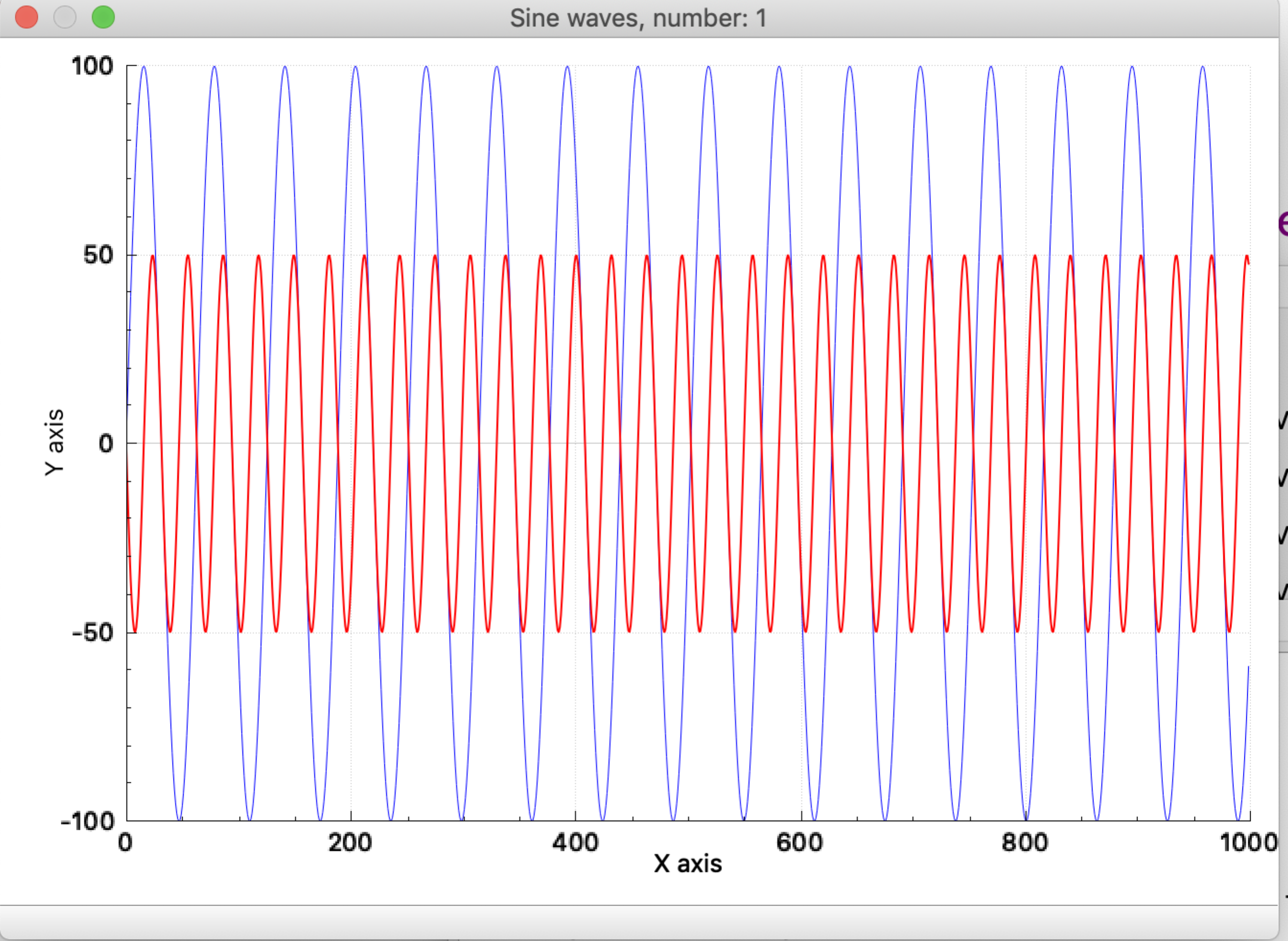
Plot

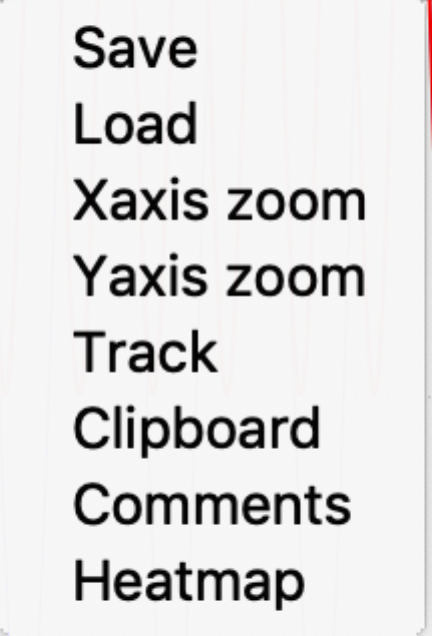
This command will plot the points added to a plot on the displayed graph.

Title,name

This command defines the plot title.

You will find a couple plot scripts in the example script section that illustrates the commands needed to generate plots. Below is the plot generated by the first example plot script.



If you right click anywhere in the plot area you will see the following popup menu. This menu provides a number of options that allow you to interact with the plot.

Save

Saves the current plot data to a file, when this option is selected you will see a file save dialog popup, select the file name you wish and select save.

Load

The load option will allow you to load a new plot file and overwrite the one currently being displayed. When you select this option a file select dialog will appear allowing you to select the file you wish to load.

Xaxis zoom

Yaxis zoom

Select one or both of the zoom options, when selected the option will have a check mark in front of its name. When enabled the zoom option allows you to use your mouse to zoom in or out and scroll the axis you have enabled.

Track

When selected the track option will show the X, and Y location of the cursor on the displayed plot.

Clipboard

Selecting this option will copy the current plot to your computers clipboard.

Comments

Comments will display the comments for the displayed plot, please note that all the plotting commands are saved in the comments section so you will see all the plot raw data, you can also add your own comments. Start all your comments line with a # character to make sure they are not read as plot commands when the plot file is loaded.

Heatmap

This option will display the data as a heat map. This option is intended for applications where a plot contains a series of graphs and allows you to visualize all the data in a plot.

Example scripts:

Below are a few example scripts you can use to learn how the system works and build on to address your specific needs. Scripting is a powerful capability and can allow you to automate the process of collecting data while scanning parameter values to speed the process of characterization. You will find these example scripts in the MIPS install director on your system.

Example 1: This is a simple example this illustrates the message IO options. After this process starts it will count from 1 to 10 on the status bar and then allow you to do it again or stop.

mips.popupMessage("Press OK to start this demo");

mips.statusMessage("here we go!");

while(true)

{

for(i=0;i<10;i++)

{

mips.msDelay(1000);

mips.statusMessage(i+1);

}

if(!mips.popupYesNoMessage("That was fun, want to do it again?")) break;

}

mips.statusMessage("finished");

status = “done”;

Example 2: This script will acquire 5 datasets and must be run from a control panel. You can use the example control panel to test this script. You will need to make sure the filepath is correct on your system.

mips.popupMessage("Press OK to start demo");

// Read 5 data files.

for(i=0;i<5;i++)

{

mips.Acquire("/Users/GAA/desktop/test.0001");

while(mips.isAcquiring()) mips.msDelay(1000);

mips.DismissAcquire();

}

status = "done";

Example 3: This example script illustrates the use of the popupUserInput and changing a MIPS voltage in a data collection loop. This script assumes we are connected to a MIPS controller named MIPS. The script must be run from a custom control panel and have an IFT defined.

mips.popupMessage("Press OK to start demo");

count = mips.popupUserInput("Setup","Enter number of dataset to record:");

startV = mips.popupUserInput("Setup","Enter inital voltage:");

stepV = mips.popupUserInput("Setup","Enter voltage step:");

for(i=0;i<count;i++)

{

voltage = startV + stepV \* i;

mips.statusMessage(voltage);

mips.SendCommand("MIPS", "SDCB,1," + voltage + "\n");

mips.msDelay(100);

mips.Acquire("/Users/GAA/desktop/test.0001");

while(mips.isAcquiring()) mips.msDelay(1000);

mips.DismissAcquire();

}

status = "done";

Example 4: This example script illustrates the use of the plot capability and will display two sine waves. The script must be run from a custom control panel.

mips.CreatePlot("Test plot","Y axis","X axis",2)

mips.PlotCommand("Title,Sine waves");

mips.PlotCommand("NewGraph,2,1000");

mips.PlotCommand("Xrange,0,1000");

mips.PlotCommand("Yrange,-100,100");

for(i=1;i<1000;i++)

{

j = i / 2;

k=100\* Math.sin(i/10);

l = -50 \* Math.sin(i/5);

mips.PlotCommand("Addpoint," + i.toString() + "," + i.toString() + "," + k.toString() + "," + l.toString());

}

mips.PlotCommand("Plot");

mips.PlotCommand("Refresh");

Example 5: This example script illustrates the use of the plot capability and will display two random graphs. This examples show how to generate a strip chart plot with real time update every time a point is added to the plot. The script must be run from a custom control panel.

mips.CreatePlot("Test plot","Y axis","X axis",2)

mips.PlotCommand("Title,Random real time data");

mips.PlotCommand("NewGraph,2,1000");

mips.PlotCommand("Xrange,Time");

for(i=1;i<1000;i++)

{

k=Math.random();

l = k\*2.0 + 3

mips.PlotCommand("PlotPoint," + i.toString() + "," + l.toString() + "," + k.toString());

mips.msDelay(1000);

}

These examples give you an idea of what is possible using your own scripts to define specialized experiments. Please let us know if you need additional features or find bugs.

**TCP/IP server**

The TCP/IP server allows your program to control your MIPS systems that are connected through the MIPS application’s custom control panel. This is done using a TCP/IP socket connection and the controlling program can be on a different computer or on the local system. The TCP/IP server is enabled by a MIPS control panel that has the following command in its configuration file:

TCPserver,9999

For this example the MIPS control panel will listen to port 9999 for commands to set or monitor values on your control panel. You can test this capability using the telnet application and connecting to localhost at port 9999 when your control panel is running. All commands are \n terminated ASCII string, all responses from MIPS are also \n terminated ASCII strings.

The TCP/IP commands used are based on the names you used in the custom control panel to define your controls. Please note the commands are case sensitive. Listed below are the details for each of the control types supported by the TCP/IP server.

ADC channels:

The ADC channels are given a name in the custom control panel configuration file. This name is used to read ADC voltage, for example if the ADC channel is named *IonCurrent* then the following commands are valid:

*IonCurrent* Returns the *IonCurrent* value.

ARB channels:

The ARB channels are given a name in the custom control panel configuration file. This name is used to read and write the ARB channel parameters, for example if the ARB channel is named *ARB 1* then the following commands are valid:

*ARB 1.Frequency* Returns the *ARB 1* frequency setting.

*ARB 1.Amplitude* Returns the *ARB 1* Amplitude value, Vp-p.

*ARB 1.Aux output* Returns the *ARB 1* auxiliary output voltage.

*ARB 1.Offset output* Returns the *ARB 1* offset voltage.

*ARB 1.Forward* Returns the *ARB 1* forward flag, TRUE or FALSE.

*ARB 1.Reverse* Returns the *ARB 1* reverse flag, TRUE or FALSE.

*ARB 1.Waveform* Returns the *ARB 1* waveform type, SIN, RAMP, TRI, PULSE, or ARB.

*ARB 1.Frequency=value* Sets the *ARB 1* frequency setting.

*ARB 1.Amplitude=value* Sets the *ARB 1* Amplitude value, Vp-p.

*ARB 1.Aux output=value* Sets the *ARB 1* auxiliary output voltage.

*ARB 1.Offset output=value* Sets the *ARB 1* offset voltage.

*ARB 1.Forward=value* Sets the *ARB 1* forward flag, TRUE or FALSE.

*ARB 1.Reverse* *=value* Sets the *ARB 1* reverse flag, TRUE or FALSE.

*ARB 1.Waveform=value* Sets the *ARB 1* waveform type, SIN, RAMP, TRI, PULSE, or ARB.

Compressor:

The ARB compressor is given a name in the custom control panel configuration file. This name is used to read and write the compression parameters, for example if the ARB compressor is named *Compress* then the following commands are valid:

*Compress.Order* Returns the compression order.

*Compress.Table* Returns the multi-pass compression table.

*Compress.Compress time* Returns the compression time in milli-seconds.

*Compress.Trigger delay* Returns the trigger delay in milli-seconds.

*Compress.Normal time* Returns the normal time in milli-seconds.

*Compress.Non compress time*

Returns the non compress time in milli-seconds.

*Compress.Compress* Returns the compress flag, TRUE or FALSE.

*Compress.Normal* Returns the normal flag, TRUE or FALSE.

*Compress.Open* Returns the Open flag, TRUE or FALSE.

*Compress.Close* Returns the Close flag, TRUE or FALSE.

*Compress.Trigger* Issues a trigger command to the compressor.

*Compress.Order=value* Sets the compression order to value.

*Compress.Table=value* Sets the multi-pass compression table to value.

*Compress.Compress time=value*

Sets the compression time to value in milli-seconds.

*Compress.Trigger delay=value*

Sets the trigger delay to value in milli-seconds.

*Compress.Normal time=value*

Sets the normal time to value in milli-seconds.

*Compress.Non compress time=value*

Sets the non compress time to value in milli-seconds.

*Compress.Compress=value* Sets the compress flag to value, TRUE or FALSE.

*Compress.Normal=value* Sets the normal flag to value, TRUE or FALSE.

*Compress.Open=value* Sets the Open flag to value, TRUE or FALSE.

*Compress.Close=value* Sets the Close flag to value, TRUE or FALSE.

DAC channels:

The DAC channels are given a name in the custom control panel configuration file. This name is used to read and write the DAC voltage settings, for example if the DAC channel is named *Bias* then the following commands are valid:

*Bias* Returns the *Bias* voltage setting.

*Bias=value* Sets the *Bias* DAC voltage to value.

DCbias channels:

The DCbias channels are given a name in the custom control panel configuration file. This name is used to read and write the DCbias voltage settings and monitor the readback values, for example if the DCbias channel is named *Inlet* then the following commands are valid:

*Inlet* Returns the *Inlet* voltage setting.

*Inlet.readback* Returns the *Inlet* voltage readback value.

*Inlet=value* Sets the *Inlet* DCbias voltage to value.

DCbias offset channels:

The DCbias offset channels are given a name in the custom control panel configuration file. This name is used to read and write the DCbias offset voltage settings, for example if the DCbias offset channel is named *CH1-8 offset* then the following commands are valid:

*CH1-8 offset* Returns the *CH1-8 offset* voltage setting.

*CH1-8 offset =value* Sets the *CH1-8 offset* voltage to value.

DCbias enable channels:

The DCbias enable channels are given a name in the custom control panel configuration file. This name is used to read and write the DCbias enable state, for example if the DCbias enable channel is named *CH1-8 enable* then the following commands are valid:

*CH1-8 enable* Returns the *CH1-8 enable* state, ON or OFF.

*CH1-8 enable =value* Sets the *CH1-8 enable* state to value, ON or OFF.

DIO channels:

The DIO channels are given a name in the custom control panel configuration file. This name is used to read and write the DIO channel’s state, for example if the DIO channel is named *Enable* then the following commands are valid:

*Enable* Returns the *Enable* state, 0 or 1.

*Enable =value* Sets the *Enable* state to value, 0 or 1.

ESI channels:

The ESI channels are given a name in the custom control panel configuration file. This name is used to read and write the ESI channel’s parameters, for example if the ESI channel is named *ESI* then the following commands are valid:

*ESI* Returns the *ESI* voltage setting.

*ESI.readback* Returns the *ESI* actual voltage.

*ESI.ena* Returns the *ESI* enable status, ON or OFF.

*ESI=value* Sets the *ESI* output voltage to value, in volts.

*ESI,ena=value* Sets the *ESI* enable status to value, ON or OFF.

RFdriver channels:

The RFdriver channels are given a name in the custom control panel configuration file. This name is used to read and write the RFdriver parameters, for example if the RFdriver channel is named *HPIF* then the following commands are valid:

*HPIF.Drive* Returns the *HPIF* drive setting.

*HPIF.Freq* Returns the *HPIF* frequency.

*HPIF.RF+* Returns the *HPIF* RF+ output voltage p-p.

*HPIF.RF-* Returns the *HPIF* RF output voltage p-p.

*HPIF.Power* Return the *HPIF* RF head power in watts.

*HPIF.Drive=value* Sets the *HPIF* RF drive level to value, 0 to 100 percent.

*HPIF.Freq=value* Sets the *HPIF* RF drive frequency to value, Hz.

Timing generation:

The Timing generation channel is given a name in the custom control panel configuration file. This name is used to read and write the Timing generation parameters, for example if the Timing generation channel is named *Timing* then the following commands are valid:

*Timing.isTblMode* Return TRUE if the system is in table mode, returns FALSE is not in table mode.

*Timing.Time mode, in mS* Returns the time mode, TRUE or FALSE

*Timing.Ext Clock Freq* Returns the external clock frequency value

*Timing.Mux order* Returns the multiplexing order

*Timing.Frame.Start* Returns the *Timing* frame start value.

*Timing.Frame.Width* Returns the *Timing* frame width value.

*Timing.Frame.Accumulations*

Returns the *Timing* frame number of accumulations.

*Timing.Frame.Enable* Returns the *Timing* frame output enable signal. The selected signal is high during the frame acquire period. Value values are: <blank indicates none>, A, B, C, D.

*Timing.Table* Returns the *Timing* pulse sequence table.

*Timing.Event.Select event* Returns the *Timing* selected event name. The valid values are defined in the event selection box.

*Timing.Event.Signal* Returns the *Timing* selected event signal name, this is the output signal affected by this event. The valid values are defined by the DCbias channels and DIO channels.

*Timing.Event.Start* Returns the *Timing* selected event start value.

*Timing.Event.Width* Returns the *Timing* selected event width value.

*Timing.Event.Value* Returns the *Timing* selected event output active value.

*Timing.Event.Value,off* Returns the *Timing* selected event output in-active value.

*Timing.Event.StartT* Returns the calculated event Start time in counts used in the table command.

*Timing.Event.Channel* Returns the MIPS channel for the event.

*Timing.Clock source* Returns the *Timing* clock source, valid values are: Ext, ExtN, ExtS, 42000000, 10500000, 2652000, 656250.

*Timing.Trigger source* Returns the *Timing* trigger source, valid values are: Software, Edge, Pos, Neg.

*Timing.Generate* Selects the *Timing* generate button and causes the generation of a timing table based on the current parameters.

*Timing.Download* Selects the *Timing* download button and causes the current table to be downloaded to the MIPS system.

*Timing.Time mode, in mS=value*

Set the Time mode check box, value = TRUE to enable or FALSE to disable.

*Timing.Ext Clock Freq=value*

Sets the external clock frequency, value = frequency in Hz.

*Timing.Mux order=value*

Sets the multiplex mode order, value = None, 4 Bit, 5 Bit, 6 Bit, 7 Bit, 8 Bit, or 9 Bit.

*Timing.Frame.Start=value* Sets the *Timing* frame start value.

*Timing.Frame.Width=value* Sets the *Timing* frame width value.

*Timing.Frame.Accumulations=value*

Sets the *Timing* frame number of accumulations.

*Timing.Frame.Enable=value* Sets the *Timing* frame output enable signal. The selected signal is high during the frame acquire period. Value values are: <blank indications none>, A, B, C, D.

*Timing.Table=value* Sets the *Timing* pulse sequence table.

*Timing.Event.Select event=value*

Sets the *Timing* selected event name. The valid values are defined in the event selection box. Enter a unique name to create a new event.

*Timing.Event.Start=value* Sets the *Timing* selected event start value.

*Timing.Event.Width=value* Sets the *Timing* selected event width value.

*Timing.Event.Value=value* Sets the *Timing* selected event output active value.

*Timing.Event.Value,off=value*

Sets the *Timing* selected event output in-active value.

*Timing.Clock source=value* Sets the *Timing* clock source, valid values are: Ext, ExtN, ExtS, 42000000, 10500000, 2652000, 656250.

*Timing.Trigger source=value*

Sets the *Timing* trigger source, valid values are: Software, Edge, Pos, Neg.

Ccontol:

The TCP/IP interface can also be used to read and write to custom controls generated with the Ccontrol command in your control panel configuration file (cfg file). The type of the custom control will define the commands supported; below you will find the general response based on the custom control type:

LineEdit

A LineEdit control can have a number of options depending on how it is defined in your cfg file. Sending the name of the LineEdit box will return its value, setting its name equal to a value will set the LineEdit box if it has output capability and sending the name.readback will return the LineEdit readback value if it has been configured for readback capability.

CheckBox

Sending the CheckBox name will return TRUE or FALSE, TRUE if the checkbox has been checked.

You can set the CheckBox by setting it equal to TRUE or FALSE, for example Name = TRUE.

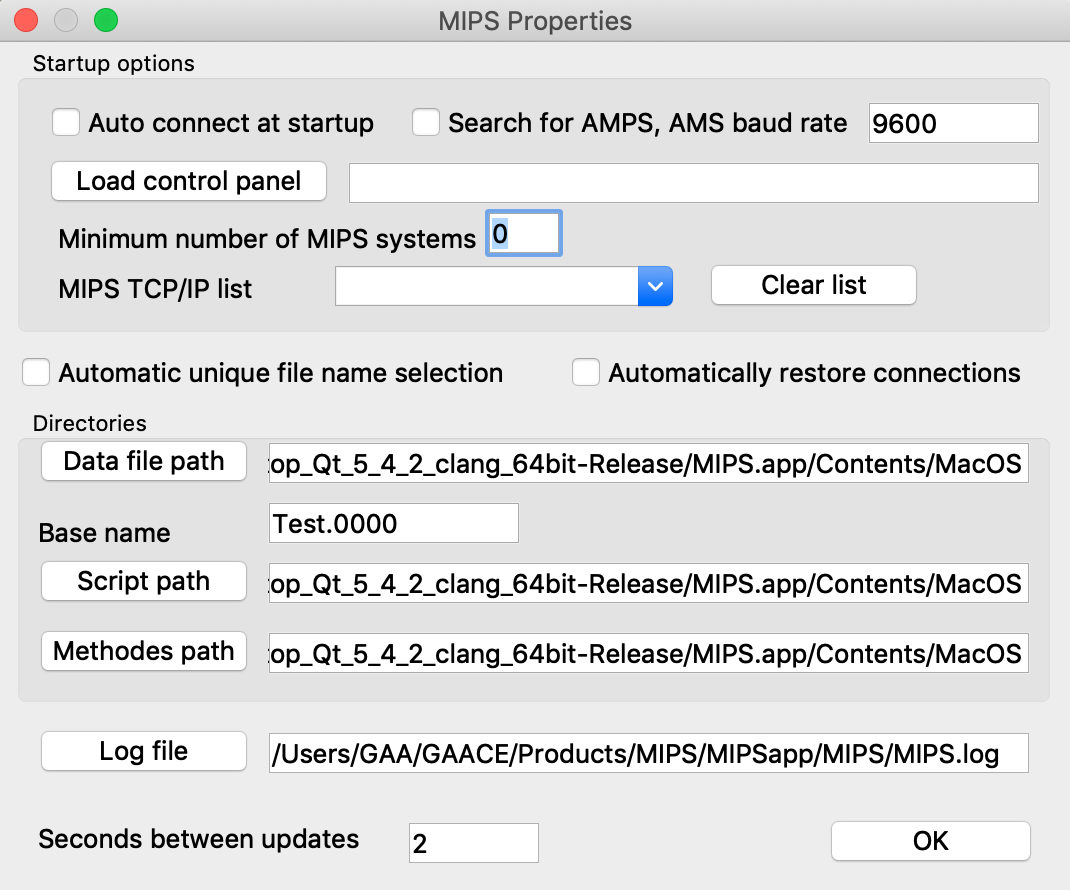
Button

Sending the name of the button is equivalent to pressing the button on the control panel.

ComboBox

**Properties page**

The properties page allows you to define parameters that will control the MIPS app behavior. To view and edit the Properties Select the Help menu options and then the properties option; you will then see the following dialog:



The parameters entered or changed are saved to a properties file when the OK button is selected. These parameters will automatically load when the MIPS app starts. Below is a description of all the options found on this properties page:

*Auto connect at startup*

If this box is checked then the MIPS app will automatically search for connected MIPS system after startup. MIPS system connected via USB are scanned only is the TCP/IP list is empty, if this list is not empty then only the defined TPC/IP addresses are scanned.

*Search for AMPS*

If check this option will direct the MIPS app to search for and connect to PNNL AMPS systems at the baud rate defined.

*Load control panel*

Pressing this button will allow you to select a control panel configuration file (.cfg) to automatically load when MIPS starts.

*Minimum number of MIPS systems*

This value allows you to define the minimum number of MIPS system that must be found before the control panel will load. This allows you to make sure the systems are connected before the control panel will start.

*MIPS TCP/IP list*

You can enter a list of TCP/IP addresses that you would like MIPS to scan for connected MIPS systems in this box. If you list TCP/IP addresses then the USB ports will not be scanned.

*Automatic unique file name selection*

Checking this box will cause the data files names to be unique by adding a sequence number to the base file name or incrementing the sequence number if its present. This function only applies to the data acquisition capability of the Timing Generator.

*Automatically restore connections*

Checking this box will cause the MIPS app to try and restore any broken USB connections when the control panel is running.

*Data file path*

The data file path is used by the acquire capability when you press the trigger button on the timing control. This path defines the location where the data file folders will be generated and files saved. This data file path must be present on your system.

*Base name*

The data file path and this base name define the folder where the acquire results will be saved. This Base name path does not need to be present on your system and it will be created at the data file path location. If the automatic unique file name option is selected then the MIPS system will add a sequence number to the base name if one is not already present.

*Script path*

This path defines the default location or folder for the script file on your system. These script files are optionally used in the control panels.

*Methods path*

This path defines the location of method files on your system. Method files are saved and loaded by the control panels and contain all the user defined parameters. This files allow you to save all your settings.

*Log file*

This button allows you to define a log file. This log file is used by the control panel to record events and any errors that occur. These files can be very useful in debugging the system issues.

*Seconds between updates*

This option allows you to define the rate that the parameters will update on a control panel. The default is every second and this may be too fast for complex multi-MIPS systems. Please note that if you change this values the change will not take effect until you terminate the MIPS app and restart.

**Appendix A: MIPS install details**

There are two version of the MIPS application, one for use on a MAC and another for use on a PC. The install files for both can be found on my google drive (https://drive.google.com/open?id=0B3IchPRNYqYIcjZhdGFVMVR1VzQ) in the MIPSapp directory. This application is a work in progress and it provides an example how to communicate with the MIPS system. This application is written in Qt and the source code can be found on my github account here: https://github.com/GordonAnderson

Install instructions

MAC:

It is very easy to install this application on a MAC, simply copy the MIPS.app file found in the MAC directory to your application directory or wherever you would like. Click on the MIPS.app file to start it up. The MIPS.app file is a bundle and you can open it with Finder. You will find a folder called MIPS inside the bundel, copy this folder to your home folder on your MAC and then you will be able to run the example control panel and scripts.

PC:

Installation is a bit more complex on a PC running windows. Windows systems do not have the needed drivers to communicate with the MIPS system using the arduino due controller. To install the MIPS application perform the following steps:

1. Copy the PC directory and all its contents including the sub directories and there contents to your windows PC and name the directory MIPS.
2. Run the program MIPS.exe. If you get the following error: “*The Program can't start because MSVCR110.dll is missing from your computer. Try reinstalling the program to fix this problem.*” then you need to run the vcredist\_x86.exe application to install the needed windows drivers. After you install the drivers the MIPS.exe application should run.
3. The next step is to make sure the proper USB drivers are installed, this Arduino page outlines the install process: <https://www.arduino.cc/en/Guide/ArduinoDue#toc10>
4. You do not need to download the drivers because they are already in the PC directory that you copied to you system in step 1. When instructed to locate the driver’s directory use the one in the PC directory.

The MIPS host computer application allows you to upload new MIPS firmware as well as control most MIPS functions from your computer.

**Appendix B: Instructions for Updating the MIPS Firmware.**

These instructions define how to use the MIPS application to load new firmware to your MIPS system. Before you attempt to load new firmware make sure you can communicate with your MIPS system using the MIPS application. You will find two versions of the application, one designed to run on a PC and one for a MAC. My google drive (https://drive.google.com/open?id=0B3IchPRNYqYIcjZhdGFVMVR1VzQ) has a MIPSapp dir has all the install files for the two versions and instructions on how to install the app. Test the MIPS app by using the Terminal tab to send commands to the MIPS system, for example send GVER to return the version of the firmware. Note the MIPS commands are case sensitive. The current version of MIPS firmware is located on my Google drive along with this readme file. The firmware files always end in .bin. Below are the instructions for updating your systems firmware: To save the current version:

1. Connect to your MIPS system and make sure you press the connect button to make the port connection. Pick a fast baud rate, 115,200.
2. Disconnect the RF heads from the MIPS box. It is a good idea to disconnect all signals from the MIPS box but the RF heads must be disconnected.
3. Select the “Save current MIPS firmware” option and follow the prompts.
4. In the end you should have a file holding the current version of your MIPS system. Save this file.

To program the new version:

1. Make sure you have a copy of the MIPS firmware file, it will be named something like: MIPS-V1.133.bin. This is the file we will use to update your systems.
2. Exit the MIPS app and then restart it.
3. Connect to your MIPS system and make sure you press the connect button to make the port connection. Pick a fast baud rate, 115,200.
4. Disconnect the RF heads from the MIPS box. It is a good idea to disconnect all signals from the MIPS box but the RF heads must be disconnected.
5. Select the “Program MIPS” option and follow the prompts.
6. When the programming operation finishes you will see the MIPS box reboot.

If something goes wrong:

This process is not fool proof and it is not uncommon for it to fail for one reason or another. This section provides a little background on how the download process works and how to recover from a failure.

The first step in the download or reading of the firmware process is to signal the ARM processor to remain in the boolloader on power up and not to run the firmware. Next the ARM processor is rebooted. At this point the firmware is still present in the ARM processor’s flash but will not run. Then the firmware can be read and saved or erased and a new version uploaded. The bootloader function is native to the ATMEL ARM processor and a program called bossac is called by the MIPS app to do the programming and reading of firmware. The bossca app’s output it then redirected to the terminal screen in the MIPS app.

If the process fails most likely the bootloader flag is cleared so the ARM processor stays in the boolloader mode and will not run the MIPS app. This renders the MIPS system inoperable. If this happens you can recover by the following procedure:

1. Disconnect from MIPS by selecting the System tab and then press the Disconnect button.
2. Exit the MIPS app and then restart the app.
3. Configure the serial port and note that the com port has likely changed, that is because the ARM processor’s bootloader is now running and not the MIPS firmware.
4. After you configure the serial port press the Connect button. Note that the system will generate error messages (timeouts) and show them in the status bar at the bottom of the dialog box. This is ok; we are connecting only because it is required by the MIPS app to do the firmware update.
5. Select the Terminal tab.
6. Now you can use one of the options under Tools to program new firmware, read the current version or just set the bootloader flag. Note that you do not need to set the bootloader flag when programming a new version or reading the current version, these functions automatically set the flag when finished.