Application Report
August 2004

User's Guide for Sonic MDIO Software

John Meenen *High Performance Analog*

ABSTRACT

This document explains the usage of the "Sonic" MDIO Software written by Texas Instruments Inc to provide a means of communication with an MDIO compatible device. It communicates through the serial port of the host PC. The signals get translated from a serial port format to the MDIO spec. format by a required MDIO Interface Board also developed by Texas Instruments Inc. The program provides read and write access to the MDIO port on the device, and also provides basic scripting functions to allow the user to write scripts that contain a sequence of MDIO commands for more complex tasks.

Contents

1 Introduction ... **3**
2 The Sequence Language ... **3**
2.1 Coding Guide ... 3
2.2 Sequence Command Reference ... 5
2.3 Important Notes ... 8
3 The User Interface ... **8**
3.1 Main Sequence Editor ... 9
3.2 Input/Output from MDIO ... 9
3.3 Immediate Read and Write ... 10
3.4 Macro Tabstrip ... 10
3.5 Statusbar ... 11
3.6 Base Conversion ... 11
3.7 File Menu ... 11
3.8 Settings Menu ... 11
3.9 Help Menu ... 12

Figures

Figure 1. Main Screen ... **9**
Figure 2. Macro Execution and Rename Pop-up Windows ... **10**
Figure 3. MDIO Interface Board Settings Menu ... **12**

1 Introduction

The Sonic MDIO Software was written to give the user the ability to communicate with an MDIO compatible device. The user tells the software how to communicate with the device by writing scripts of commands. These scripts are called “sequences.” The program allows the user to write a main script, and other sub-scripts called “macros.” This document explains the graphical user interface (GUI) and the sequence language.

2 The Sequence Language

This section is a language reference for the MDIO software. First it will discuss the conventional use of the language. This is followed by a definition of all the valid sequence commands.

2.1 Coding Guide

The MDIO software allows the user to write scripts that will perform the MDIO commands. The scripts are simply a list of commands that are executed sequentially. These scripts are called “sequences” or “macros.”

The program uses line-by-line execution. There is a main sequence which can operate independently, or call other macros. The macros can be though of as subroutines in that they are just a sequence of commands that may be executed multiple times.

Below is an example of code that shows a main sequence and two defined macros.

```
EXAMPLE.TXT

START
//THIS IS THE MAIN SEQUENCE
READ(8000)
WRITE(8002,ABCD)
READ(8002)
THIRD MACRO
DISPLAY(EXECUTING MAC2 NOW...)
MAC2
DISPLAY(READ 8000 ONCE MORE)
READ(8000)
STOP

MACDEF MACRO 1
//THIS IS THE DEFINITION OF "MACRO 1"
START
READ(8230,0000)
DISPLAY(REGISTER 8230 WAS JUST READ)
WRITE(8000,1234)
STOP

MACDEF MAC2
//THIS IS THE DEFINITION OF "MAC2"
START
DISPLAY(MAC2 NOW EXECUTING...)
MACRO 1
STOP

MACDEF THIRD MACRO
//THIS IS THE DEFINITION OF "THIRD MACRO"
START
DISPLAY(THIRD MACRO NOW EXECUTING...)
STOP
```

At this point don't be concerned with what the commands are doing or their syntax. This will be explained later in the chapter. For now concentrate on the modularity of the code. We see that all sequences (main or macros) contain a "start" and a "stop" command. Also, each macro is identified (or defined) by a "macdef <macroname>" statement preceding the macro.

Any sequence can call (or execute) another sequence by listing the macro's name in the sequence. An example is the main sequence calls "MACRO 1" on the fifth line. The only exception to this is that the main sequence cannot be called by a macro.

When the sequence is executed within the software it executes the main sequence only. If a macro is not called then it will not be executed. Defined macros do not have to be executed. In the example the last macro "Third Macro" is defined, but never called. So for this example "running the sequence" would mean that the software would execute the main sequence. It contains four commands followed by a call for "mac2" (the second defined macro). At this point it would pause execution of the main sequence, and jump to the "mac2" macro. Mac2 has 1 command, and then calls "Macro 1", so "Macro 1" is executed. After "Macro 1" is completed execution returns to "mac2" which ends at this point with the "stop" command, and execution is returned back to the main sequence. It completes the main sequence when it reaches the "stop" command, and execution is then completed.

2.2 Sequence Command Reference

This section lists and defines all of the supported commands within a sequence. In this section arguments are surrounded with "<" and ">" (i.e. <mask>). If an argument is optional it will be surrounded with "[" and "]" (i.e. [<mask>]).

Start This command defines the start point of execution. Any text before a start command will be ignored.

Syntax: start

Stop This command defines the end point of execution. It also defines the end point of a loop (see repeat command definition). If a sequence contains a repeat command the first stop will define the end of the loop, and another stop command will be needed to end the sequence. See the example below.

Syntax: stop

Example:

```
start
    display(now in sequence)
    repeat 10
        display(now in loop)
        stop  //ends the loop
    display(now back in sequence)
stop  //ends the sequence
```

Display This command prints text to the output window. The output text color will be blue.

Syntax: display(<text>)

Example: display(Hello World!) *Prints "Hello World!" in blue text in output window.*

Read This command executes an MDIO “read” on the given register address. This command has two optional arguments that change the way the output is displayed. If the optional arguments are not given then read command will simply report the resulting data from the given register address (example 1). If a valid HEX value is given for the <expected data> argument then the read command will assume a <mask> value of “FFFF” and “read verify” of all 16 bits. The read command will report in green text or red text the resulting data and expected data. Green text means they compared equally, and red text means the resulting Hex value was different than the expected data. If all three arguments are given the read performs a “read verify” but only on the bits specified in the <mask> argument.

Syntax: read(<regaddr>[,<expected data>][,<mask>])

Example:
```
read(8000)
read(8000,0020)
read(8000,FFFF,1000)
```

Write This command executes an MDIO “write” on the given register address. This command has one optional argument. If the optional <mask> argument are not given then the write command will assume a <mask> value of “FFFF” and the register at <register address> will be written the exact value of <set/clear bits>. If all three arguments are given the write command will write only to the bits specified in the <mask> argument to the bit value of the corresponding bit in the <set/clear bits> argument.

Syntax: write(<register address>,<set/clear bits>,[<mask>]

Example:
```
write(8000,FFFF)
write(8000,F000,1001)
```

Repeat This command defines the start point of a loop. Loops can be nested, and can exist within macros. The repeat command has a required argument <Number of Repeats> which tells the software how many times to repeat a loop. The end of a loop is defined with a “Stop” command, which now makes it possible for a sequence to have multiple “stop” commands. (See stop definition for more detail)

Syntax: Repeat <Number of Repeats>

Example:
```
repeat 5  //Beginning of loop, run 5 times
    display(Looping Now…)  //Commands within loop
stop  //Marks the end of a loop
```

Echo This command sets or clears a flag in the software. The flag determines if a write command will print an output to the output textbox. If the flag is set then the write command will report an output. If the flag is cleared there is no report of write commands. To set the flag use the “ECHO ON” command. To clear the flag use the “ECHO OFF” command. The scope of the echo command is limited to the current sequence, and the default value for every sequence is echo on. Therefore, if you do not want any reports of write commands, and are calling several macros then you will need an “ECHO OFF” command at the top of each macro.

Syntax: Echo on

Echo off

Example:
```
Start
    echo off   //Clears the echo flag for this sequence
    write(8000,abcd)
Stop //End of sequence
```

SetDevAdd The user can now change the Device Address using the "SetDevAdd()" command instead of using the Settings Menu. The current Device Address is reflected in the status bar as well. The command has a required argument called <devadd>. This argument is assumed to be a Hexadecimal value unless the user uses base-casting (i.e. d'10 is decimal 10 and will be converted to Hex A, or H'A).

Syntax: SetDevAdd(<devadd>)

Example:
```
setdevadd(01)       //Sets the Device Address to H’01 [PMA/PMD]
setdevadd(d’30)     //Sets the Device Address to H’1E [Vendor Specific 1]
setdevadd(b’100)    //Sets the Device Address to H’04 [PHY XS]
```

SetPhyAdd The user can now change the Physical Address using the "SetPhyAdd()" command instead of using the Settings Menu. The current Physical Address is reflected in the status bar as well. The command has a required argument called <phyadd>. This argument is assumed to be a Decimal value unless the user uses base-casting (i.e. h'10 is hexidecimal 10 and would not need to be converted before sending).

Syntax: SetPhyAdd(<phyadd>)

Example:
```
setphyadd(01)       //Sets the Device Address to H’01
setphyadd(h’1E)     //Sets the Device Address to H’1E
setphyadd(30)       //Sets the Device Address to H’1E
```

Pause This command will freeze execution of the sequence for the specified number of milliseconds. The maximum value is 65535ms (a little more than a minute and 5 seconds), and the minimum values is 1ms.

Syntax: Pause (100)

Example:
```
Start
    clause 22   //Sets the MDIO format to Clause 22
    write(????,abcd)
    pause(10)   //Pauses execution for 10ms
    clause 45   //Sets the MDIO format back to Clause 45
    read(8000)
Stop  //End of sequence
```

Clause This command sets the MDIO format to either IEEE 802.3 Clause 22 or Clause 45 depending on the argument given. The argument can be either 22 or 45. The scope of the clause command is limited to the current sequence, and the default value for every sequence is clause 45. So for a clause 22 device every sequence that is executed would need a “clause 22” command just after the “start” command.

Syntax:
```
Clause 22

Clause 45
```

Example:
```
Start
    clause 22   //Sets the MDIO format to Clause 22
    Write(????,abcd)
    clause 45   //Sets the MDIO format back to Clause 45
    read(8000)
Stop  //End of sequence
```

// This is the formal indication of a comment. Text following a “//” until end of line will NOT be printed to the output. It is used to comment code.

Syntax: `//<text>`

Example: `//This is a comment in the code`

2.3 Important Notes

One note is that text contained inside a sequence will be ignored if it is not a valid command, macro name, or part of a valid command. Care should be taken to use the “//” command to demark all comments. Anything other than valid commands or comments preceded by the “//” command should be removed to prevent the risk of the software from interpreting the text as a possible command.

Another note is that macro names CANNOT include a sequence command anywhere within the name. Therefore, a macro name of “SCRATCHPADREAD” is not correct, and will cause an error message if used.

3 The User Interface

This section describes the user interface, and explains how to use its features. Below is a screenshot of the GUI after starting the program.

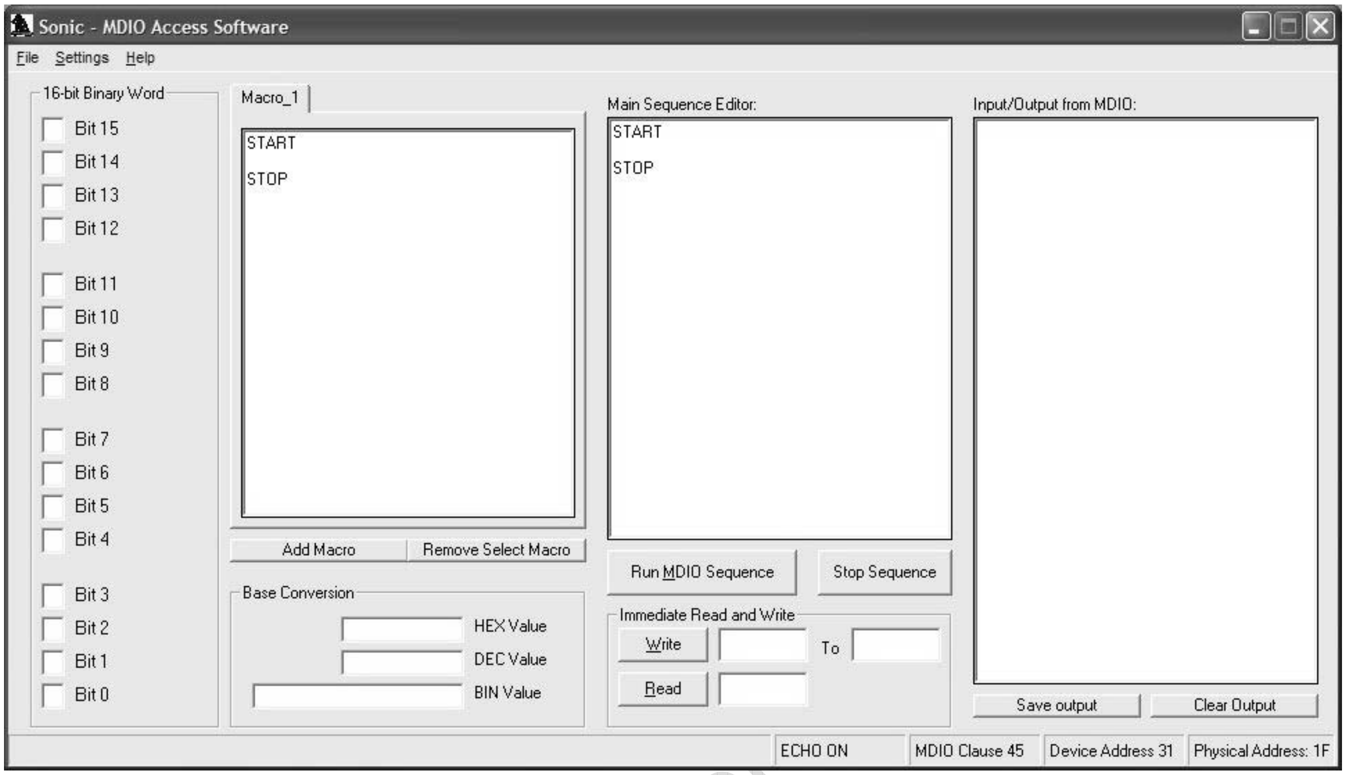


Figure 1. Main Screen

3.1 Main Sequence Editor

We will start with the "Main Sequence Editor" textbox. This is where the main sequence is typed. The "Run MDIO Sequence" button is just below on the left side of the textbox. When this button is clicked the software will execute the sequence. The button to the right is labeled "Stop Sequence." This button allows the user to abort the execution of a sequence at anytime.

3.2 Input/Output from MDIO

Output from any command during execution is printed into the "Input/Output from MDIO" textbox to the right of the main sequence editor. This textbox is read-only, but the buttons below it allow you to save the output to a file and to clear the output. You can also highlight and copy text from the textbox. Some commands have color-coded output to help quickly read the results. When you save the output you have the choice of standard text format or rich-text format. Saving in rich-text format will keep the color and font encoding.

3.3 Immediate Read and Write

The section below the main sequence editor, labeled “Immediate Read and Write,” can be used to issue a single read or write command. To read a register’s content type the register’s address (in HEX) in the textbox to the right of the “Read” command, and then press Enter or click the “Read” button. The register’s content will be printed to the output textbox. To write a value to a register type the HEX value into the textbox to the right of the “Write” button, and the address of the register (in HEX) in the next textbox. Then press Enter or click the “Write” button. An output report will be printed to the output box.

3.4 Macro Tabstrip

To the left of the main sequence editor is a tab with a textbox. This is where macros are stored. The two buttons below the tab allow you to add new macros, or to remove the currently selected macro. When you have multiple macros defined, and want to select a different macro left-click on the tab name you want to select. If you have more macros defined than will fit across the top of the editor a left and right arrow button will automatically pop-up. Use the left and right arrow to scroll through the macro tabs. Right-click on any selected tab and a pop-up window will appear. The figure below shows the pop-up windows.

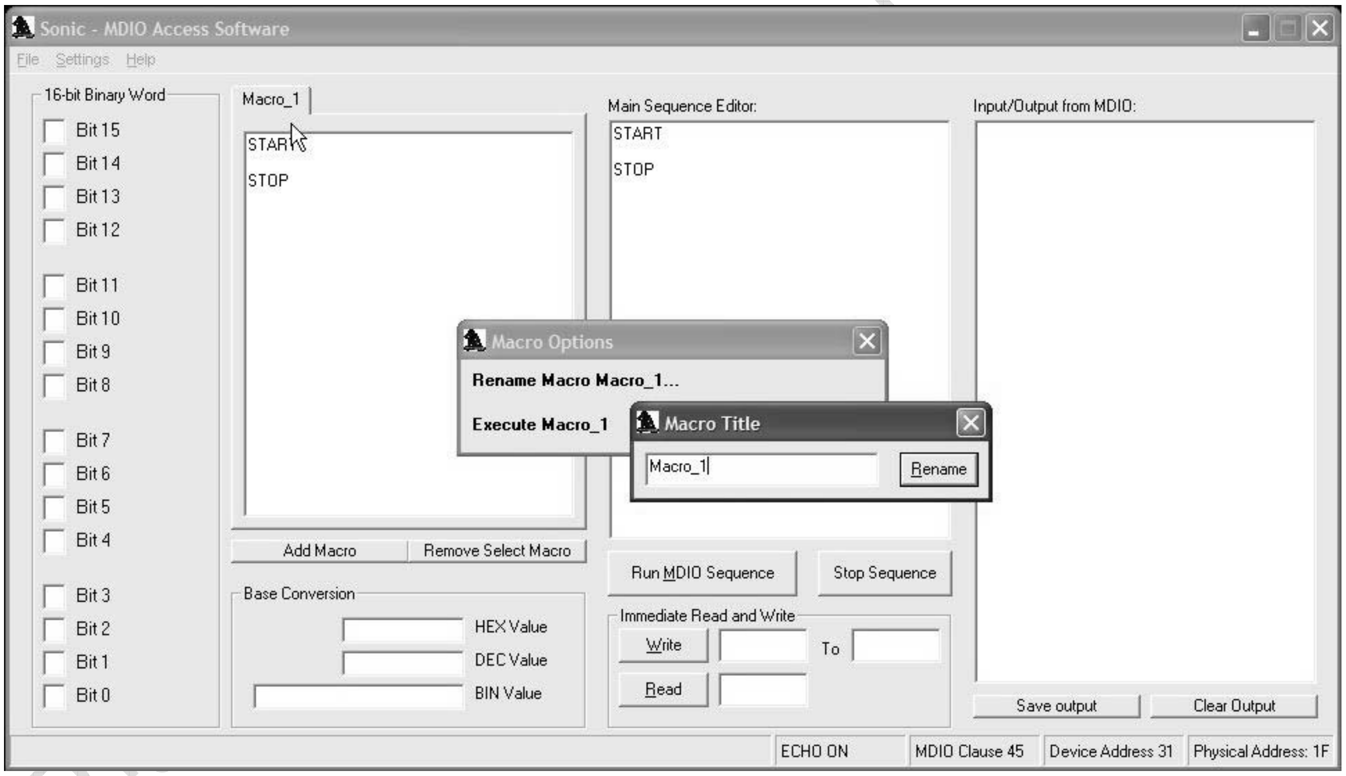


Figure 2. Macro Execution and Rename Pop-up Windows

This window gives you two options: rename the macro and execute the macro. Clicking on the rename option allows you to type a new name for that macro. Clicking the execute option will execute that single selected macro.

3.5 Statusbar

The statusbar for the program is located at the bottom of the main program's window. There are four panels in the statusbar. The first panel is for general information (i.e. when running a sequence this panel will show "Executing Sequence…").

The second and third panel double as indicators and toggle buttons. The second panel is for the "echo" command. It shows the current status of the echo flag at anytime. The echo status can be changed during sequence execution by either executing an "echo off" or "echo on" command within the sequence, or it can be toggled at any point by simply clicking on the panel in the status bar.

The third panel is for the MDIO Clause mode. It too can be clicked at any point to toggle the MDIO mode (i.e. clause 45 or clause 22), or it can be changed within the sequence with a "clause 45" or "clause 22" command.

The last panel shows the Device Address. This window will update anytime the user changes the device address either by changing it in the Settings menu, or by using the *SetDevAdd()* command in an MDIO sequence.

3.6 Base Conversion

The last "Base Conversion" section is located below the macro tabstrip. This allows the user to type, or drag and drop, a valid numerical expression into the textbox labeled as the appropriate numerical base. It will automatically convert the number to the other two listed numerical formats. It will also populate the 16 textboxes labeled as "Bits[0-15]" on the left side of the window with the binary value for each bit.

If the user drags (or types) a HEX value of "ABCD" into the "HEX Value" textbox the program will report the decimal value (43981), and the binary value (1010101111001101) in the textboxes below.

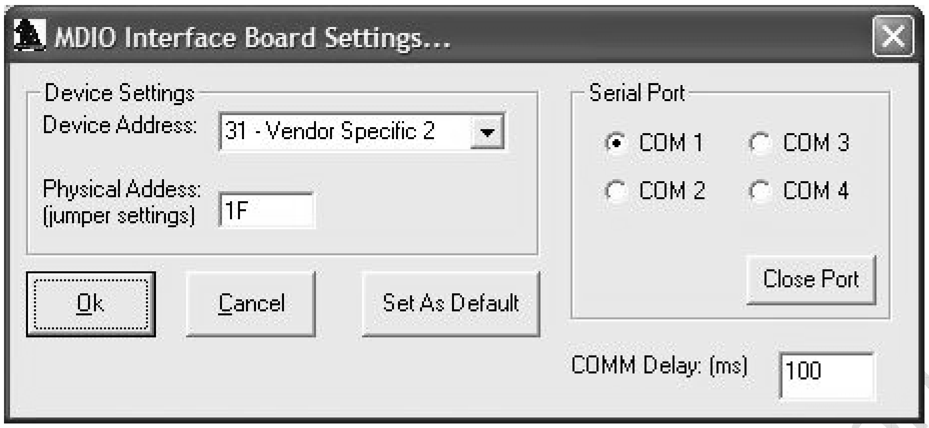
If the user drags a value over a textbox (HEX, DEC, or BIN) and it is not a valid expression for that base the program will not allow the user to drop the value into the textbox.

3.7 File Menu

The File menu allows the user to open, save, and close sequence files and to exit the program. A sequence file stores the main sequence and all defined macros. The files are saved in plain text format, and can be edited with any text editor. The format of the text file is explained in the "Sequence Language" section of this document.

3.8 Settings Menu

Clicking on the Settings menu will pop-up a separate window called "MDIO Interface Board Settings…" This window is shown in the figure below.


MDIO Interface Board Settings...

Device Settings
Device Address: 31 - Vendor Specific 2
Physical Addess: (jumper settings) 1F

Serial Port
(●) COM 1 ( ) COM 3
( ) COM 2 ( ) COM 4

COMM Delay: (ms) 100

Figure 3. MDIO Interface Board Settings Menu

The settings menu defines both the device settings as well as the serial port settings for the PC running this program. This software requires an MDIO Interface Board in order to communicate with the device. The program communicates with the board through a serial port. The board translates the serial port data into an MDIO datastream to send to the device. When data comes from the device the process works in reverse.

The “Serial Port” radio buttons lets the user select which serial port will be used to communicate with the MDIO Interface board.

The “Close Port” button allows the user to close an open serial port without having to exit the program. This is useful if the user wants to also use another program at the same time which also uses the serial port.

The “COMM Delay” setting controls a delay value that is used by the program between communication commands. The minimum value is 1ms, and the maximum value 10,000ms.

The “Device Address” setting is a device value that is defined in the device design, and determined by the device’s function according to IEEE 802.3. The default value is “1E” which is a vendor specific address. “04” would be used for a PHY device.

The “Physical Address” setting is a device value that is typically programmable to allow multiple devices on a single MDIO chain. On an EVM board this address is usually programmable via jumpers on the board.

Before you can execute a sequence or an immediate read/write you must select the serial port to use to communicate with the MDIO Interface Board. Click on the Settings menu and select the serial port, set the physical and device addresses, and click Ok.

3.9 Help Menu

The help menu tells the user what version of the software they are using. It also displays the readme.txt file that explains new features and known issues. During installation of the software the readme.txt file is saved in the same directory as the program.