Upgraded Software for the W8TEE/K2ZIA
Hacker's Guide - Version 03.0

John Price – WA2FZW

License Information

Introduction

The W8TEE/K2ZIA antenna analyzer was originally developed as a club project for the Milford Amateur Radio Club. The original author of the software, Jack Purdum, published the design and code online on the Yahoo SoftwareControlledHamRadio group (which has now been moved to the SoftwareControlledHamRadio group on Groups.io). Jack also published an article about the project in the November 2017 issue of QST.

My main objective in modifying the software was to make it work on the 6 meter band (which requires replacing the AD9850 DDS with the higher frequency AD9851). In the process of going through the original code to figure out how to accomplish this, I did find a number of potential and actual bugs in the code (Definition: Working Software – Software with only undiscovered bugs) and came up with some enhancements to make it easier to use. Then I got a little carried away!

This document provides an overview of what each of the functions that make up the software do at a very high level for anyone who wants to attempt their own modifications. You will find rather detailed descriptions of what each function does in the comments in the software itself.

My_Analyzer.h

This header file contains all the definitions of things that anyone might want (or need) to change in order for the analyzer to operate with the various hardware options available.

It includes the following definitions for the various DDS options:

```
#define AD9850_DDS    // Using the AD9850 DDS board
#define AD9851_DDS    // Using the AD9851 DDS board
#define AD8307_SWR    // Using AD8307 detectors
```

Definitions related to adding the 6 meter and/or “Custom” bands:

```
#define ADD_6_METERS             // Enable 6 meter operation
#define LOW_6M_EDGE 50000U       // 50 MHz * 1000
#define HIGH_6M_EDGE 54000U      // 54 MHz * 1000
#define ADD_CUSTOM               // Enable the "Custom" band
#define LOW_C_EDGE     1000U     //  1MHz * 1000
#define HIGH_C_EDGE   30000U     // 30MHz * 1000
```

Definitions of the parameters that control the “Repeat Scan” function:

```
#define DEFAULT_COUNT 50         // Initial default count
#define REPEAT_INCREMENT 10      // How much to change the count
#define MIN_REPEAT_COUNT 10      // Minimum repetition count
#define MAX_REPEAT_COUNT 100     // Maximum repetition count
#define SCAN_PAUSE 1000          // Length of pause between scans
```

Whether or not the “Fine Tune” button had been added:

```
#define    FT_INSTALLED
```

If you want the “Examine” function to operate automatically after using the “Single Scan”, “Repeat Scans” and “View Plot” functions, un-comment the following line.

```
#define AUTO_EXAMINE
```

If the definition is commented out, the “Examine” feature will still be activated by simply moving the encoder knob one click one way or the other.

EPROM Address Map

The following shows the addresses of things stored in the EEPROM and the symbolic names used in the program to reference them.

Address	Symbol	Purpose in Life
0000 – 0001	SWR_MINS_SET	Indicates minimum SWRs have been set
0002 – 0003	SWR_MINS_ADDRESS	Saved SWR readings - 160M
0004 - 0005		Saved SWR readings - 80M
0006 - 0007		Saved SWR readings - 60M
0008 - 0009		Saved SWR readings - 40M
0010 - 0011		Saved SWR readings - 30M
0012 - 0013		Saved SWR readings - 20M
0014 - 0015		Saved SWR readings - 17M
0016 - 0017		Saved SWR readings - 15M
0018 - 0019		Saved SWR readings - 12M
0020 - 0021		Saved SWR readings - 10M
0022 - 0023		Saved SWR readings - 6M
0024 - 0025		Saved SWR readings - Custom
0050 – 0051	ACTIVE_BAND_SET	Indicates active band data is saved
0052 – 0053	ACTIVE_BAND_INDEX	Saved active band index setting
0054 – 0055	ACTIVE_BAND_BOTTOM	Saved active band low freq setting
0056 – 0057	ACTIVE_BAND_TOP	Saved active band high freq setting
0090 – 0091	NEXT_SD_FILE_NUMBER	Saved next file sequence number
0092	EEPROM_NEXT	Next available EEPROM address

Main Program Functions

The first two functions are the standard Arduino functions:

setup()	Initializes all of the things needed to make it work
loop()	Runs continuously; basically processes the menu functions

The functions described in the following sections are grouped according to their primary role (although they may have secondary roles). The order they are listed in is the same order as which they appear in the .ino file.

Initialization Functions

The functions in this group primarily deal with setting various variables that make everything else work.

SetActiveBand()	Sets the active band an startup and may be invoked from the “Analysis” menu.
SetBandEdge()	Sets the upper and lower scan frequency range
GetBandEdge()	Gets the upper and lower scan frequency range When needed
ReadActiveBandData()	Reads the saved band and frequency information from the EEPROM.
SaveActiveBandData()	Saves active band and frequency information to the EEPROM.
SetEEPROMMins()	Sets the saved minimum SWR readings in the EEPROM.
ReadEEPROMMins()	Reads the saved minimum SWR readings from the EEPROM.

Menu Processing Functions

These functions process the main and sub-menus

ShowMainMenu()	Display the top level menu.
AlterMenuOption()	Controls which main menu item is selected and makes a selection when the encoder switch is pushed.
ShowSubMenu()	Displays one of the sub-menus.

AlterMenuDepth()	Controls which sub-menu item is selected and makes a selection when the encoder switch is pushed.
DoAnalysis()	Processes selections in the “Analysis” menu.
DoOptions()	Processes selections in the “View/Save” menu.
DoMaintenance()	Processes selections in the “Maintenance” menu.

Command Processing Functions

The functions in this group are responsible for executing the individual commands initiated via sub-menu selections.

DoNewScan()	Performs and displays the results of a single scan between the preset frequency ranges.
RepeatScan()	Repeats a scan between the preset frequency ranges a specified number of times.
Examine()	This is not exactly a command processing function, but it’s called after “DoNewScan”, “RepeatScan” and “ViewOldPlot” to allow the operator to examine the SWR at frequencies determined by moving the encoder knob.
DoSingleFrequency()	Monitors the SWR at one specific frequency (which can be changed while monitoring). It also includes an “analog” SWR meter function.
SaveScan()	Saves the data from the most recent scan to the SD card.
ViewOldPlot()	Displays the contents of a saved scan exactly as it was displayed when originally performed.
PlotOverlay()	Can be used to display the results of a saved scan on the same graph as a live scan or another saved one.
ViewTable()	Displays the contents of a saved scan file in a tabular format.
PlotToSerial()	Sends the contents of a saved scan file verbatim to the Arduino IDE’s serial monitor.

DrawBarChart()	Draws the bar chart of the saved minimum SWR values.
DeleteSingleFile()	Deletes a selected single file from the SD card.
DeleteAllFiles()	Deletes all of the files from the SD card.
ResetFileSeqNumber()	Resets the next file sequence number provided there are no saved scan files on the SD card.
EraseEEPROM()	Completely erases the contents of the EEPROM except the next file sequence number.
ReadEEPROM()	Displays the contents of the EEPROM on the Arduino IDE's serial monitor.

Formatting & Display Functions

These functions either format specific data items or display specific things on the TFT display:

Splash()	Displays the startup information and credits.
FormatFrequency()	Formats the internal frequency into an ASCII string with either 2 or 3 decimal places.
FormatSWR()	Formats the internal SWR into an ASCII string.
Format()	General number to ASCII function (eventually will be eliminated).
PaintText()	Writes text strings of various sizes on the screen.
EraseText()	Erases text from the screen.
GraphAxis()	Plots the background for the scan plots.
GraphPoints()	Plots the actual scan data.
MarkMinimum()	Puts the little red '+' characters at the minimum SWR points on the scan graphs.
PaintMeter()	Paints the "analog" SWR meter on the display.

MovePointer()	Controls the movement of the pointer on the “analog” SWR meter.
ShowAndScroll()	Part of the “View Table” function; decides which page should be displayed and controls scrolling between pages.
DrawTable()	Displays a single page of the “View Table” output.
DisplayFrequency()	Used to display frequencies with the carat (‘^’) character under the digit that rotating the encoder will change.
PaintHeading()	Paints the headings on the plot and table outputs.

DDS Control Functions

These functions manipulate the DDS:

SendFrequency()	Sets the DDS output to a specified frequency.
DDS_Down()	Puts the DDS to sleep.
DDS_Up()	Wakes the DDS up when needed.
GetNextPoint()	Gets the next SWR/frequency pair when performing a live scan.
ReadSWRValue()	Reads and computes the VSWR.

SD Card Related Functions

These all have to do with things related to using the SD card:

Mount_SD()	Mounts the SD card at startup or if a card wasn’t installed at startup. Still has a bug (see the *User Manual*)
CountFiles()	Counts the numner of “SCANnn” files on the card.
ShowFiles()	Displays a list of the “SCANnn” files on the card.

SelectFile()	Allows the operator to select a single file from the displayed list.
SortFiles()	Sorts the files by name before displaying them.
ConfirmDelete()	Gives the operator the ability to cancel out of or confirm deletion of a single file.
ConfirmDeleteAll()	Gives the operator the ability to cancel out of or confirm deleting all files.
ReadScanDataFile()	Reads the scan data from a saved file.
WriteScanData()	Writes the current scan to an SD file.
Display_SD_Err()	General error display for SD problems.
Display_SD_Err_2()	Specific sequence of error messages when there are no “SCANnn” files on the card.
Display_SD_Err_4()	Specific sequence of error messages when there is no SD card present.

Interrupt Processing Functions

These functions handle the actual interrupts from the encoder and “Fine Tune” button (if installed) and the subsequent processing of those interrupts.

ReadEncoder()	Handles and processes interrupts generated by rotating the encoder knob.
ResetEncoder()	Clears the flags resulting from moving the encoder.
FT_Interrupt()	Handles the direct interrupts from the “Fine Tune” button (if installed).
ReadFT()	Processes interrupts from the “Fine Tune” button and handles the contact bounce problem.
ResetFT()	Resets the variables associated with the “Fine Tune” button.

Miscellaneous Functions

These two really don’t neatly fit into any of the previous categories:

ConfirmAction()	Used where the operator is given the option to cancel out of a previously selected function such as deleting a file or erasing the EEPROM.
DisplayScanStruct()	Conditionalized on the definition of DEBUG, this function displays the contents of the “scan” structure on the Arduino IDE’s serial monitor.