TEST EQUIPMENT PLUS

RF Recorder Demo Manual

Operation and Programming Guide

Andrew Montgomery, Justin Crooks

2/11/2013

Contents

Introduction

The RF Recorder demo application is an exploratory application showing the possibilities of the BB60A as a means for RF recording. With a 20MHz bandwidth and 80M sample rate the BB60A is capable of capturing a wide range of broadband signals. The API makes acquiring data trivial. One issue addressed in this demo is the rate of data saved to disk. At around 160MB/s, the rate exceeds that of traditional HDD drives. We must look to RAID configurations or SSDs to assist. The demo does not have much practical application at this point, but will interest programmers or individuals looking to determine whether the BB60A is the product they need for their application.

Additional Introduction for Programmers

The RF Recording demo is a nice introduction to programming the BB60A for real time use. In this case we show various aspects of interest, collecting data in the BB60A "Raw Pipe" mode (all 80M samples per second), a simple control panel to control the device, useful displays, and a simple saving scheme.

This program uses a commercial signal processing library from Intel called Integrated Performance Primitives. It is required to compile the project. The library cost is $100 per developer, or there is a free trial.

Read the section titled "Programming Guide" to get started.

Operation

Installation

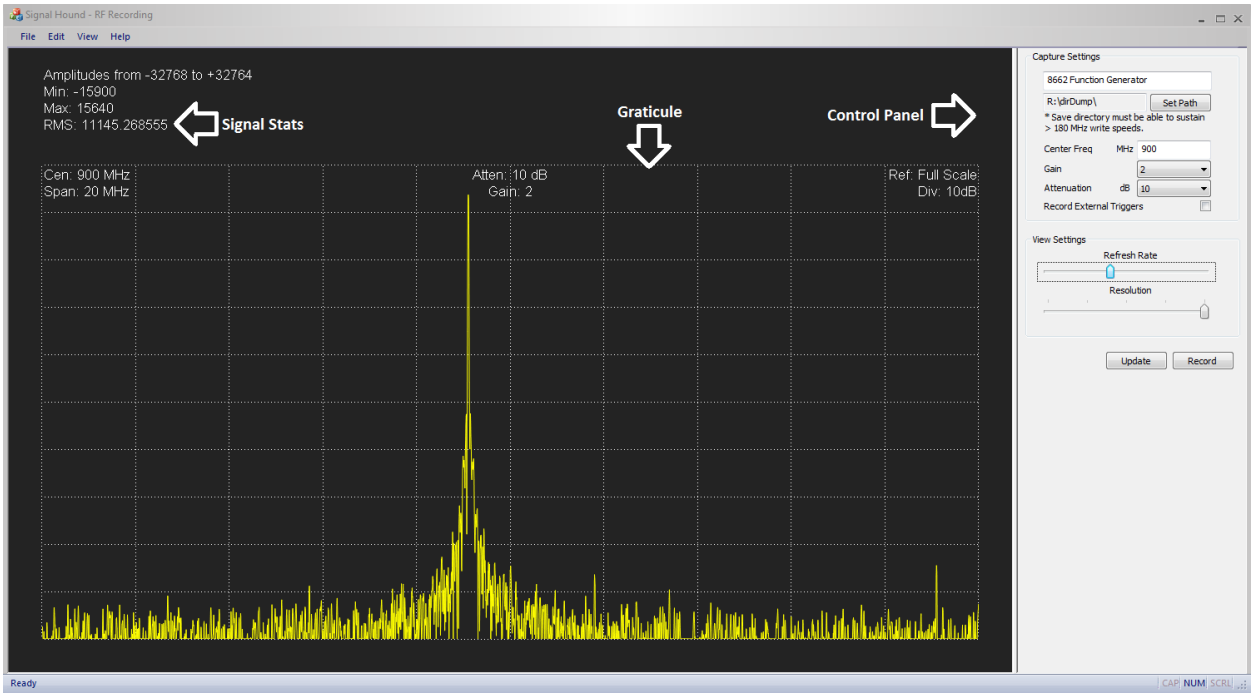
To run, the executable needs to be in the same directory as the API (“BB60APIW32.dll”) and the OpenGL extension library (“glew32.dll”). Both are included on the installation disk, though they may not be the latest versions. The API can also be downloaded from the Signal Hound website. The OpenGL extension library can be downloaded from glew.sourceforge.net. which can be found on the website or on the installation disk shipped with the BB60A.

1) If you installed the BB60A spectrum analyzer software from the software disk provided, the demo executable is found in the same directory as the main software.
2) If you have downloaded this demo from the website and you have the BB60A software installed, you only need to place the demo executable in the same directory as the main software executable.
3) If you downloaded the demo from the website and have not installed the main software you will need to install the main application on your system. Once installed place the demo executable in the main BB60A directory.

Running

The demo allows you to view

Upon running the program for the first time, you should see something similar to the image below if you have a BB60A attached. If no device is detected during program launch, the program will issue a warning and close.



A quick glance at various features of the display.

Signal Stats: This section of the view shows you characteristics of the incoming ADC data. The statistics are not performed over the entire data set, but a minimal sample size per incoming packet. You can see the minimum and maximum values reported by the ADC as well as the Root Mean Square. These values can help you acquire a signal of the best dynamic range and to avoid compression.

Graticule: The graticule shows you the results of a single FFT on the incoming data. The display is in the log scale with 10 dB divisions and shown in full scale. This can be useful in maximizing the dynamic range of the incoming signal. The top line represents the maximum potential input given the current settings. Compression happens if the signal reaches the top line.

Control Panel: The control panel allows you to change the operational settings as well as set up a signal capture. You can change the center frequency, and the gain and attenuation settings. Gain and attenuation should be adjusted to ensure the best dynamic range.

You can give your signal captures a name, specify the location of the save(must be hard drive capable of > 180MB/s write speed), and whether or not to store the triggers.

NOTE: All changes are not active until update is pressed.

Read the section “Recording” to learn more about the record button.

Recording

The record button on the control panel instantly begins a record session. Before you press it, ensure all settings are up to date. Ensure your capture title and save folder are current. Attempting to save to a hard drive with less than 180MB/s write speed will result in undefined operation. The file name chosen for the saves is constructed based on the current time. This ensures no two file names will conflict.

Recording begins immediately after pressing the record button. The data is saved to disk in raw binary format, a 16 bit signed integer for each sample. All files are prefixed with the constructed file name. An XML file is also created with the same prefix. It contains all settings, capture title, and any triggers that appear during the recording session. (A high frequency input trigger signal will result in a large xml file)

Programming Guide

This demo is a MFC application project in Visual Studio 2008.

Setup and Requirements

If all you desire is to view the code, nothing is needed but a text editor, preferably a source code editor.

If you wish to compile this program, this is a list of what is needed.

1) Microsoft Visual Studio 2008 or later, with the capability to compile MFC applications.
2) Intel IPP Signal Processing Library. A free trial version is available which provides a developer with access to all development tools.

3) BB60APIW32.dll/.lib, glew32.dll/.lib

Program Organization

There are four main parts to this demo application.

1) Interfacing the BB60A device
2) Handling the control panel
3) Drawing the signal and text
4) Saving the data

The main thread of execution is left for Win32 callbacks and drawing, but an additional thread is created and runs the "MainThread" function in the RFDoc class. It is here where all program logic and flow is found.

Interfacing the BB60A

To best understand this section on interfacing it is recommended you have some precursor knowledge of the BB60A API. Reading the API manual is recommended.

This demo contains a light C++ class wrapper around the API. This encapsulation is recommended for large projects, it relocates all API related functionality to one object, and allows you to pass around a device object easily. In this case our wrapper provides the functionality of opening the device, configuring it with a simple settings class, and retrieving the raw data.

The Control Panel

The control panel (Control.cpp/h) contains its own copy of the device and view settings which are manipulated through the control panel objects. When Update is pressed, the control panel tells the Document(main class/thread of execution) about the new settings. The record button tells the Document to begin recording.

Drawing

Drawing is done in RFView. We use OpenGL to render text and the signal. To update the display call the Invalidate() method on the RFView class. The view is updated in the "RFDoc::MainThread" routine, and is updated at a rate determined by the user through the control panel refresh rate slider.

Saving

Once the Record button is pressed, the document prepares the DiskController class responsible for saving the data, and prepares the XML descriptor file. The XML file is generated with a small wrapper class around the rapidXML library. The DiskController allows you to queue up data to save, and in a separate thread, the class writes out the data to disk. Due to variances in disk I/O it is unreasonable for saving to be a blocking call. It is important to save in a separate thread, and allow a significant queue to build to account for any difficulties.

In this program we save unsigned shorts converted from the returned floats, which we save up to 1GB per file (1792 returned packets). Sequential files are suffixed with an integer counter. File names are prefixed with the same name as the xml file.

What's Next

The purpose of this program is to show the feasibility of recording RF with the BB60A. We introduce some simple methods for viewing the data and optimizing the capture as well as providing a very simple file format. In a successful end user application, the file format should be carefully chosen, as it increases the potential for other applications to interface the captures and eases development around RF viewing and RF editing software. We really like the XML descriptor file as it removes a lot of information which would need to be stored at the top of the first file, and it easily editable later with more than simple values. An XML file could choreograph multiple files in a more elegant RF capture format.

It is easy to see how one might extend the functionality of this program to viewing and editing the RF captures. Similar statistics and views would still apply. Loading the data from memory is actually implemented in the DiskController class but not used. Disk I/O requirements are reduced when reading from disk as real-time playback may not be required. Keeping some sort of moving/shifting pipe of data from the disk with a viewing range. A circular buffer would be useful, as well as some controls for the user to move through the data.

Contact Information

For programming related questions please contact Andrew Montgomery at aj@teplus.com. For hardware specific questions relating to the operation of the BB60A please contact justin@teplus.com. We are interested in hearing feedback, criticism, and praise. If you would like to see a new demo, let us know, we might be able to help.