

Detection with Enhanced Energy Windowing – Phase I Report



David Bass
Alexander Enders
December 2016

Approved for public release.
Distribution is unlimited.

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website <http://www.osti.gov/scitech/>

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Website <http://www.ntis.gov/help/ordermethods.aspx>

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Website <http://www.osti.gov/contact.html>

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Nuclear Security and Isotope Technology Division

**DETECTION WITH ENHANCED ENERGY WINDOWING
PHASE I REPORT**

David Bass
Alexander Enders

Date Published:
December 2016

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

1.	Introduction.....	1
1.1	Project Description.....	1
1.2	Phase I.....	1
1.3	Phase II.....	2
1.4	Documentation.....	2
2.	How to Use the DEEW Files	3
2.1	Extracting the DEEW.zip File	3
2.2	The Liberty BASIC Application	4
2.2.1	Application Setup.....	4
2.2.2	Running the Application	4
2.2.3	Input Files	4
2.2.4	Output Files.....	5
2.2.5	Source Code	5
2.3	The C# Application	6
2.3.1	Application Setup.....	6
2.3.2	Running the Application	6
2.3.3	Input Files	7
2.3.4	Output Files.....	7
2.3.5	Source Code	7
2.4	Comparing Outputs	7
2.4.1	Whitespace Differences	8
2.4.2	Rounding Differences	8
2.4.3	Output Path Difference	8
2.4.4	Date & Time Difference	8
2.5	Relevant Notes	8
2.5.1	Changing the Location of the Input Files Directory	8
2.5.2	Testing Subsets of Input Data	9
3.	Further Assistance	9

1. INTRODUCTION

This document reviews the progress of Phase I of the Detection with Enhanced Energy Windowing (DEEW) project. It contains a brief description of the project, instructions for setting up and running the applications, and guidance to help make reviewing the output files and source code easier.

1.1 PROJECT DESCRIPTION

The DEEW project is the implementation of software incorporating an algorithm conceived by Bruce D. Geelhood (Bruce). This algorithm reviews data generated by radiation portal monitors (RPMs), and utilizes advanced and novel techniques for detecting radiological and fissile material while not alarming on Naturally Occurring Radioactive Material (NORM). Independent testing by the Johns Hopkins Applied Physics Laboratory, documented in AOD-13-0337 (“Preliminary Report on Phase II of the PVT-AIP Program: Verification and Validation of Results from 256-Channel Energy Windowing Algorithms”, version 1.0, dated August 2013) indicated that the Enhanced Energy Windowing algorithm showed promise at reducing the probability of alarm in the stream of commerce compared to existing algorithms and other developmental algorithms, while still maintaining adequate sensitivity to threats.

Bruce originally implemented his algorithm using an application called Liberty BASIC, an interpreted programming language intended to help non-Software Development professionals create software applications. The Domestic Nuclear Detection Office (DNDO) then began working with Oak Ridge National Laboratory (ORNL) to re-implement the code in a more modern software development language (C#) so it can be utilized by radiological detection software and devices. Taking this measure provides a more efficient, secure and robust application, and broadens opportunities for technological advancement into Windows applications, Web applications, Web services, distributed components, and more.

The project has been broken into two phases. The goal of the first phase is to translate the existing Liberty BASIC code to C# to the degree it can be refactored in the second phase. The goal of the second phase is to refactor/re-design the C# code into a more componentized application, which other applications can interact with. The first phase of the project is completed, and this document and the accompanying data is intended to aid in reviewing the results of Phase I.

1.2 PHASE I

The translated C# replicates the algorithm Bruce accomplished in Liberty BASIC for the Roosville Stream of Commerce (SoC) input files. Running each application against this input data produces the same results. Specifically, there are three output files produced which should be nearly identical: the EW_BR log (aka, Background Ratio log), the Sigma log and the Spectra log. This document provides information on how to run the Liberty BASIC and C# applications, and compare their output results. Phase I is simply a translated version of what Bruce implemented in Liberty BASIC, with minimal re-design. Minor refactoring was done for basic file I/O (i.e., reading input files and writing output files), but the majority of the code has simply been translated from Liberty BASIC to C#. Regardless of not having re-architected the code structure yet (which is the goal of Phase II), this simple translation shows immediate benefits in performance (over 40 times faster), providing a friendlier graphical user interface for testing, and being written with a more flexible, modern and widely known language.

1.3 PHASE II

The Liberty Basic code is a single, monolithic, 13,000-line piece of code. While there are a few functions defined at the end of the program (less than 10), the entire file input, processing, and file output is performed in one very large module. Some variables are global, and some others are re-used for different purposes throughout the code. As a result, maintaining or improving any aspect of the code requires nearly God-like mastery of the entire code and all its interdependencies. Bruce has this mastery, but current technology is not sufficient to replicate Bruce. Breaking this large code into more manageable sections of code (objects) is the intent of Phase II, but this is a non-trivial effort as it requires a detailed understanding of what is being done in each portion of the code and how that can affect processing elsewhere in the code.

1.4 DOCUMENTATION

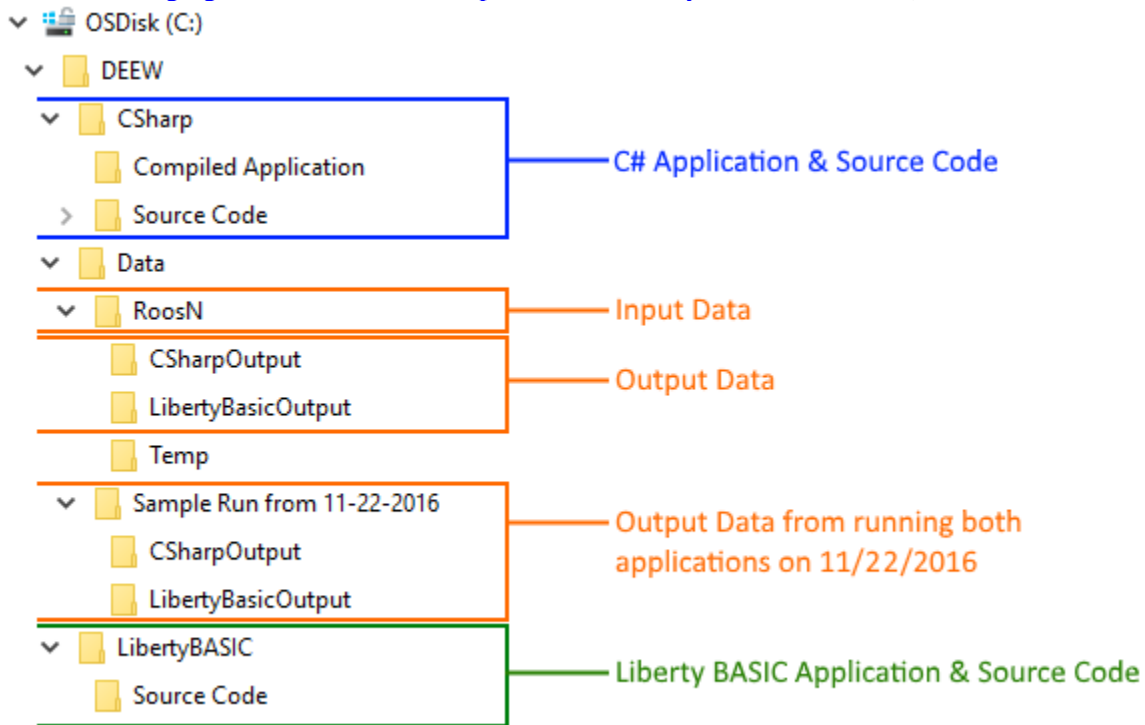
Documentation associated with the original Liberty BASIC code written by Bruce was included with the package of information sent with this document. Other than comments in the C# code files, this document is the only documentation available for the C# application and its code at this stage of the project.

2. HOW TO USE THE DEEW FILES

This section attempts to give guidance to set up and execute the Liberty BASIC and C# applications, and give some helpful tips for reviewing the three primary output files.

2.1 EXTRACTING THE DEEW.ZIP FILE

A compressed ZIP file (*DEEW.zip*) was delivered with this document. When extracted (unzipped), *DEEW.zip* will create the directory structure shown below. The directories identified in blue are related to the C# compiled application and its source code. Directories identified in green are related to the Liberty BASIC application source code. And those identified in orange are related to the input and output data for both applications. These files were intended to be extracted directly on the root drive (C:\). The default settings for both the Liberty BASIC and C# applications depend on the input files being in the directory structure laid out below; however, the location of the files can be changed relatively easily (see the section, [Changing the Location of the Input Files Directory](#) on how to do this).



Once unzipped, the *DEEW* folder contains three subdirectories: *CSharp*, *LibertyBASIC* and *Data*. The *CSharp* directory contains subdirectories holding the compiled executable used to run the C# application and a copy of the working source code. Similarly, the *LibertyBASIC* directory contains 1 subdirectory holding the functioning Liberty BASIC source code. Liberty BASIC is an interpreted language and its programs have to be run within the Liberty BASIC software application; therefore, there is no *Compiled Application* directory for this app.

The *Data* directory contains several important subdirectories. *RoosN* contains the data files used as input when running both the C# and Liberty BASIC application. Underneath *RoosN*, the *CSharpOutput* directory is where output files are written when the C# application is run, and the *LibertyBasicOutput* directory is where output files are written when the Liberty BASIC application is run. The *Temp* directory was included as a convenience for testing the applications on smaller subsets of input data (see the section, [Testing Subsets of Input Data](#) for guidance on how to do this). The other directory located

underneath *Data* is called *Sample Run from 11-22-2016*. This directory contains the three primary output files resulting from running both applications on November 22, 2016, using the files in the *RoosN* directory as input.

Important Note - Changing the Directory Structure

If the *DEEW.zip* file is unzipped anywhere other than directly on the C:\ drive, then both the Liberty BASIC and C# applications will need to have settings changed to point to the appropriate directory for where the input files reside. To change these settings, see the section, [Changing the Location of the Input Files Directory](#).


2.2 THE LIBERTY BASIC APPLICATION

2.2.1 Application Setup

If the *DEEW.zip* file was unzipped in a directory other than the root of the C:\ drive, or if the directory structure was altered in any way, then the Liberty BASIC application needs to know where the input files are located. To set this up, see the section, [Changing the Location of the Input Files Directory](#), before running the application.

Liberty BASIC must be installed in order to run this application.

2.2.2 Running the Application

Once Liberty BASIC is installed and running properly, load the file *EEW62.bas* in the Liberty BASIC Editor. This file is located in the *..\DEEW\LibertyBASIC\Source Code* directory. Once the file is loaded, simply press and hold the <Shift> key and press the <F5> key to run the application, or click the Run button in the tool bar (which looks like this: ).

Important Note – Liberty BASIC Running Time

The Liberty BASIC application takes roughly 7 to 9 hours to run to completion using the Roosville (RoosN) input files, depending on processing speed and available memory. See the section, [Testing Subsets of Input Data](#) for guidance on testing a subset of the input files in less time.

2.2.3 Input Files

The Liberty BASIC application uses the files located in the *..\DEEW\Data\RoosN* directory for input.

Important Note – Input Files

The input directory for the Liberty BASIC application must have a directory with the name “*RoosN*” as the final directory in the path. This is due to hard-coding of the directory name in the Liberty BASIC code for this application.

2.2.4 Output Files

The Liberty BASIC application will produce several output files. These output files will be placed in the `..\DEEW\Data\RoosN\LibertyBasicOutput` directory. If a `LibertyBasicOutput` directory does not exist where the input files are located, Liberty BASIC will try to create it (it will not check permissions or prompt first). There are three output files we're concerned with:

- `EEW_062Q111_RoosN_EW_BR_Log.csv`
- `EEW_062Q111_RoosN_Sigma_Log.csv`
- `EEW_062Q111_RoosN_Spectra_Log.csv`

These three files will be re-created after each execution of the Liberty BASIC application. The other files produced by the Liberty BASIC app are not relevant to whether or not the algorithm is performing correctly, and they are not needed.

2.2.5 Source Code

The `..\DEEW\LibertyBASIC\Source Code` directory contains the following 4 files, which are discussed below:

- *EEW062a.lsn.bas.Original*
- *EEW62.bas*
- *RoundingIssues.bas*
- *Info.txt*

EEW062a.lsn.bas.Original

This is the original Liberty BASIC application written by Bruce Geelhood. It served as the source of the code to be translated to C# and has not been modified during this project.

EEW62.bas

This is also a copy of the Liberty BASIC application originally written by Bruce; however, it has had modifications made to it. As code was being translated into C#, changes were made to this code for several reasons. Many of the changes in the code are accompanied with a comment to explain the change. Here is a list of reasons for most (if not all) of the changes:

- A few bugs were found and corrected. All changes made to implement bug fixes are denoted with `"DAB: UPD:"` and followed by an explanation of why the change was made.
- The format of the some of the numerical output was modified. These changes were made so the format of certain numeric values would be consistent, and also to allow large numeric values to be shown (some numbers were being truncated). These changes are denoted with `"DAB: FMT:"` and are followed by an explanation of why the change was made.
- Changes were made to simplify the selection of the directory to process. Since Phase I of this project focuses on the Roosville Stream of Commerce (SoC) data, there is only one directory to process. Therefore, the original code was modified to process only this directory. These changes are noted with `"DAB: DIR:"`.
- Changes were made to send the output to a directory other than where the input files are located. This was done so the C# and Liberty BASIC applications could use the same input files. These changes are noted with `"DAB: OUT:"`.
- While tracking down discrepancies produced between the Liberty BASIC and C# output, there was a need to set breakpoints in the Liberty BASIC code. This was difficult in some cases where

'if' and 'for' blocks were combined on a single line of code, and a breakpoint could not identify when a block of code executed. Therefore, some blocks of 'if' and 'for' statements were broken out into separate lines to allow breakpoints to be set when the code was executed. Since these changes did not affect the flow or logic of the code, they are not commented.

- Some of the debugging and processing flags not needed for Phase I were turned off. These changes are mainly in the first 100 lines (or so) of code, and simply entail setting values to 0 or 1 to turn a setting on or off. These changes are not commented.

To ensure the integrity of the Liberty BASIC code, Bruce was informed when bug fixes were made and why they were being made. He was also sent a copy of this document and the modified Liberty BASIC code for his own evaluation.

RoundingIssues.bas

As soon as file comparisons began being made between the Liberty BASIC and C# application output files, the discovery was made of differences between the rounding of decimal values in the two languages (this was no surprise and was somewhat expected). This file is a short program written to demonstrate the rounding issues discovered in Liberty BASIC. For more information on this issue, see the [Rounding Differences](#) section.

Info.txt

This file mentions what the other three files in this directory are intended for. It was included to assist anyone who did not read this document in its entirety.

2.3 THE C# APPLICATION

2.3.1 Application Setup

If the *DEEW.zip* file was unzipped in a directory other than the root of the C:\ drive, or if the directory structure was altered in any way, then the C# application needs to know where the input files are located. To set this up, see the section, [Changing the Location of the Input Files Directory](#).

The C# application requires the Microsoft .NET Framework version 4.5.2 or later to be installed on the machine.

2.3.2 Running the Application

To run the C# application, simply double-click the *DEEW.exe* file in the *..\DEEW\CSharp\Compiled Application* directory. Below are a few noteworthy mentions for running the C# application.

- Ensure the "Directory of files to process" field displays the directory where the Roosville SoC files (i.e., the *RoosN* directory) are located. If it does not, see the section [Changing the Location of the Input Files Directory](#) to see how to change this setting.
- Since this application is not the final product, it does not implement much input validation or error handling. The UI was kept simple and put together rather quickly, so other user-friendly features were also kept to a minimum.

- The *Cancel* button in the user interface can be temperamental since the app is very IO intensive. If the *Cancel* button doesn't work on the first click, try clicking it again. Closing the app also cancels the current run.

Important Note – C# Running Time

The C# application takes roughly 7 to 9 minutes to run to completion using the Roosville (RoosN) input files, depending on processing speed and available memory. See the section, [Testing Subsets of Input Data](#) for guidance on testing a subset of the input files in less time.

2.3.3 Input Files

The C# application uses the files located in the `..\DEEW\Data\RoosN` directory for input.

2.3.4 Output Files

The C# application will produce several output files. These output files will be placed in the `..\DEEW\Data\RoosN\CSharpOutput` directory. If a *CSharpOutput* directory does not exist where the input files are located, the application will try to create it (it will not check permissions or prompt first). We are concerned with the same three output files from this app:

- `EEW_062Q111_RoosN_EW_BR_Log.csv`
- `EEW_062Q111_RoosN_Sigma_Log.csv`
- `EEW_062Q111_RoosN_Spectra_Log.csv`

These three files will be re-created after each execution of the C# application. The only other file produced by the C# application is the Error Log file, and it is not relevant or needed.

2.3.5 Source Code

In the `..\DEEW\CSharp\Source Code` directory is a Visual Studio 2015 (VS 2015) solution file (.sln) called *Deew.sln*. Loading this solution file using VS 2015 will load all of the source code for this application.

As mentioned in the introduction of this document, this implementation of the C# code is mostly a direct translation of the Liberty BASIC code, with minimal refactoring or re-designing. This translation is intended to show Bruce's algorithm can be replicated in a more modern language, and a future version of the software will be refactored and optimized. Therefore, the architecture of the C# code will look similar to the Liberty BASIC code.

2.4 COMPARING OUTPUTS

To make comparing the output files easier, the file names were kept the same and they were placed into two different subdirectories, side-by-side (i.e., the *CSharpOutput* and *LibertyBASICOuput* directories). Due to the size of the files, a file comparison tool should be used when comparing them. While translating the code into the C# application, the *Beyond Compare* tool was used.

There are a few known differences between the output files, which are described below.

2.4.1 Whitespace Differences

One known difference, which can easily be ignored in most file comparison tools, is the occasional discrepancy in spacing (having an extra space in one output file but not the other). With this known difference, it is usually helpful to compare the files as text files and ignore minor differences such as these.

2.4.2 Rounding Differences

There are differences in some numbers where the last digit in a decimal is different by 1 value. For example, the value *4.1234* in Liberty BASIC's output may show up as *4.1235* in C#'s output. Code was written in C# to correct rounding issues when values are formatted for output, but the code in Liberty BASIC was kept in its original form. Therefore, portions of the output files do not match up exactly due to this rounding discrepancy, and show values off by 1 decimal position. The program (*RoundingIssues.bas* in the *Liberty ..\DEEW\LibertyBASIC\Source Code* directory) was written to demonstrate this issue.

2.4.3 Output Path Difference

The path pointing to the output directory will be different, as one points to the *LibertyBasicOutput* path and the other to *CSharpOutput*.

2.4.4 Date & Time Difference

The date and time the application was run will be different. This is only located at the top of the Sigma and Spectra log files.

2.5 RELEVANT NOTES

2.5.1 Changing the Location of the Input Files Directory

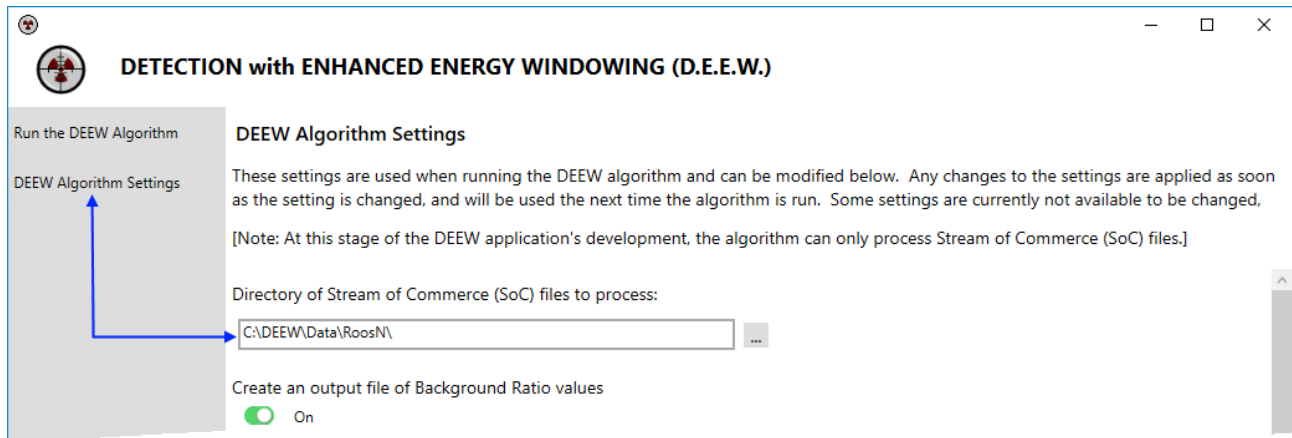
The default settings for both the Liberty BASIC and C# applications rely on the input files being in the *C:\DEEW\Data\RoosN* directory; however, the location of the input files can be changed. If the input files are placed in a different directory, a couple of settings in the C# and Liberty BASIC applications will need to be modified to point to the input files.

To change the input directory in the Liberty BASIC application, follow these steps:

1. Open the file *EEW62.bas* in the Liberty BASIC Editor.
2. The beginning of the file has a few comments. Line 15 is the first line of executable code and should start with the following: `LogDir$ = "C:\Deew\Data\RoosN\"`
3. On this line of code, change the directory to point to where the input files are located.
4. The Liberty BASIC application will now look for the input files in the newly designated path.

To change the input directory in the C# application, follow these steps (see the image of a partial screenshot below for visual details):

1. Run the *DEEW.exe* application.
2. Click the *DEEW Algorithm Settings* button in the left-hand pane.
3. Update the first setting, which should be *Directory of Stream of Commerce (SoC) files to process*. This field can be updated by typing a directory name in or using the ellipsis button (`...`) to select a directory. If the directory entered does not exist, the border of this setting will turn red.
4. The C# application will now look for the input files in the newly designated path.



2.5.2 Testing Subsets of Input Data

Running the applications against the entire set of Roosville data (roughly 2,500 occupancies) takes considerable time, particularly running the Liberty BASIC app. In addition to running them against the entire set of data, they can be run against a subset of the data. Included in the directory structure the *DEEW.zip* file created is a directory called *Temp*, located under the *Data\RoosN* directory. This was set up as a convenience to move data files into. In order to test a subset of the data, simply move files from the *RoosN* directory into the *Temp* directory. The applications will only process the files in the *RoosN* directory, not those in *Temp*. When doing this, it is important to move all three files associated with each occupancy. For example, to test 10 occupancies, leave 10 of the **.backgd.csv*, **.info.txt* and **.raw.csv* files in the *RoosN* directory (for a total of 30 files), and move the rest to the *Temp* directory. The next time the applications run, both of them will process only the 10 occupancies in the *RoosN* directory. Move the files back from the *Temp* to the *RoosN* directory to have them processed again.

Important Note – Moving Data Files

When moving files between the *RoosN* and *Temp* directories, it is important to move all three file types (**.backgd.csv*, **.info.txt* and **.raw.csv*) for any given occupancy. For example, to move the occupancy *Roosville.MT.4._D20100621_T230821_O0055_CRO256_U_PVT* from *RoosN* to *Temp*, the following files must be moved to avoid errors during file processing:

- *Roosville.MT.4._D20100621_T230821_O0055_CRO256_U_PVT.backgd.csv*
- *Roosville.MT.4._D20100621_T230821_O0055_CRO256_U_PVT.info.txt*
- *Roosville.MT.4._D20100621_T230821_O0055_CRO256_U_PVT.raw.csv*

3. FURTHER ASSISTANCE

The contact information below can be used should any questions or issues arise during the review.

Project Manager: Alex Enders
 Phone: (865) 574-9083
 Email: endersal@ornl.gov

Software Engineer: David Bass
 Phone: (865) 576-0009
 Email: bassda@ornl.gov