# PPOD Processing Software Update

M. Borgerson

9/17/2013

## Introduction

Until now, PPOD data uploaded from the instruments has been processed with **raw\_load\_ppod**. This program would accept either an input file name parameter or, if no name was provided, query the user for a file name to process. The user could process multiple files by writing a script to call **raw\_load\_ppod** with a series of file names. However, **raw\_load\_ppod** suffers from a number of problems:

* The program has grown by agglutination as new versions of the PPOD firmware have been developed. The program attempts to determine which version of the PPOD generated the data, then uses long sequences of ‘if’ statements to process each different file type. The result is an 850-line program singularly lacking in modularity and structure, which is very difficult to maintain.
* The program is poorly documented both from the standpoint of external documentation and internal comments.
* There are some rather subtle errors in the coding of the section which attempts to read the PPOD header data. One error, found just last week, caused the program to incorrectly delete eight characters from the header data for firmware version 5 files. I was lucky that this caused a program crash which led to the discovery of the error. For a slightly different header file, the bug could have deleted significant digits from a calibration constant, leading to incorrect output data throughout the file.
* As part of the file type determination, the program relies on specific file extensions. For example, HS PPOD files much have extensions of the form ‘HNNN’, while AirPPOD data must have extensions of the form ‘Annn’, where ‘nnn’ is the unit serial number. If you make a wrong choice when uploading the data, you may need to go back and change all the file extensions.
* Data processing for HSPPOD files is not very efficient. The algorithm reads data records one-by-one in a loop. While this is fairly efficient in C, it is much less efficient in MATLAB. Processing a single file takes about 37 seconds.

In an effort to overcome some of these problems, I have separated the parts of the program into functions of reasonable size, corrected some errors, and added comments.

## Updated Software

The primary goal of the new software is to provide the ability to process PPOD data without having to write new scripts, change the Matlab code, or follow a poorly documented set of requirements for data file organization.

The new software consists of a main function to collect file names and evaluate the header to determine the firmware version that generated the data. This function then calls separate functions to process each type of data file. Each file processed returns a structure with time and 1Hz pressure and temperature vectors. The main function concatenates the data from all the files and returns an output structure and generates a check plot of the pressure data. The program also displays a list of the files processed and their starting date and time.

### Main Program Function

### The main program function is longloadppod.m When you call the function with:

**myoutput = longloadppod**

it starts by prompting you for file names using a standard file dialog. You can navigate to any disk and directory if your Matlab Current Folder is not set to the desired data folder. This file dialog has multiselect enabled, so you can select more than one file to be processed. The easiest way to select multiple files is to click on the first, then Shift-Click on the last file. When you select ‘Open’ at the bottom right of the dialog, the function will start processing the files in the order in which they appear in the dialog. Figure 1 shows a number of files selected for processing.

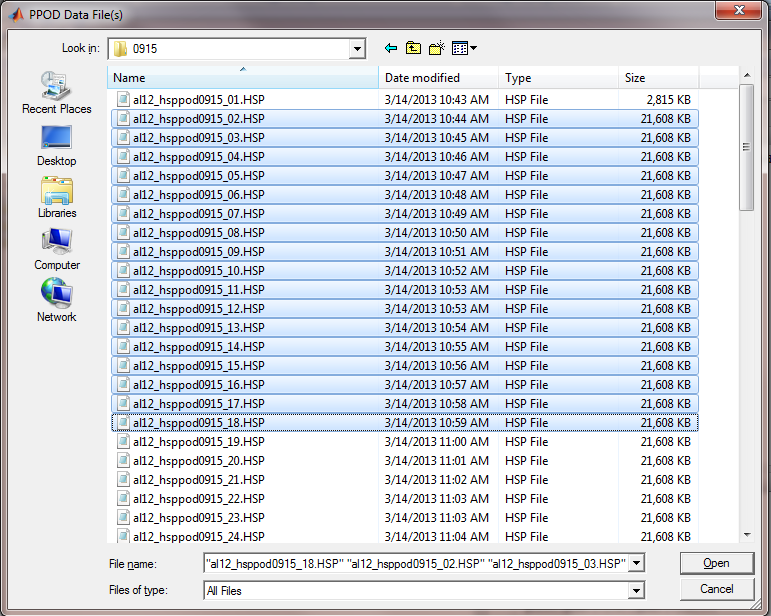


Figure 1. File Dialog in **longloadppod.m** with multiple files selected. Note that these HSPPOD files were not saved with a file extension showing the serial number and the file names do not contain the date of acquisition. This is due to a limitation in the early version of the host program used to upload the data.

Once you press ‘Open’, the program starts to read the files, convert the data, and generate the output structure. Figure 2 shows the Matlab window after the processing is complete.

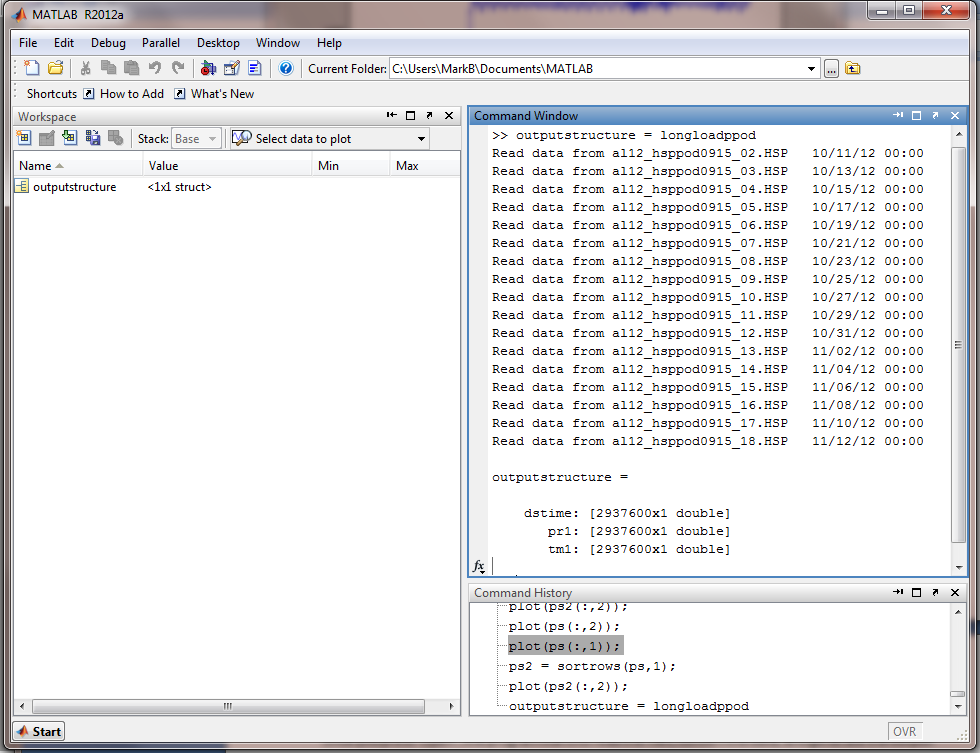


Figure 2. Matlab window after running longloadppod.m.

Since there is no semicolon after the command, Matlab has shown the contents of the output structure. The structure contains concatenated dstime, pr1, and tm1 vectors. The command window also shows a list of the files processed and the starting data of the data. You can copy and paste this data into a text file for later reference. The HSPPOD files also contains 20Hz pressure data, that is not saved by this version of the program.

Figure 3 shows the check plot for this data set. The standard zoom tools can be used to examine a subset of the data. Note that selecting ‘print’ for this figure can result in a slow print process, as Matlab will have to scale all 2.9 million data points as it sends the data to the printer.

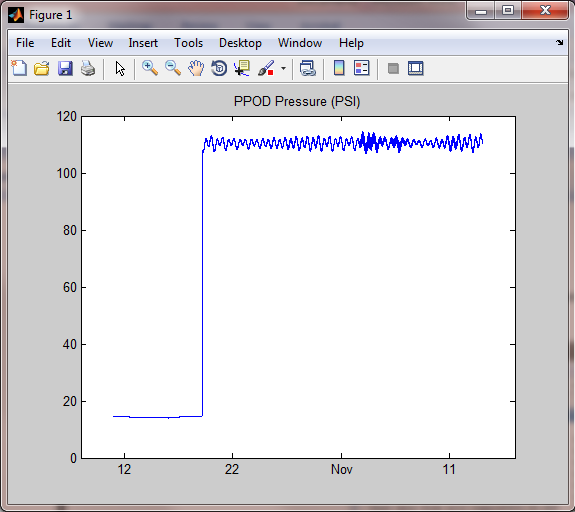


Figure 3. Check plot produced by **longloadppod.m**. This particular data set shows the deployment of the HSPPOD on a lander in the Gulf of Alaska in 2012 and a moderate storm generating large wave signals at about 5 November.

After you have processed a data set, you can save the concatenated output as a .mat file by selecting the data structure and using the Matlab ‘Save’ command. In this case myoutput.mat is about 30MB in length.

The first 20 files take about 2.2 seconds each to process. After that there are occasional hesitations as Matlab allocates more memory for the concatenated data structure.

### Support Functions

In the same file as **longloadppod**.**m** is **FindPPODVersion.m**. This function is called with the file name of the first selected file. It reads the header from the file and returns an integer, either 0, 5 or 6, which is the firmware version that wrote the file. A return value of zero indicates that the data cannot be processed by this version of the software.

**FindPPODVersion**.**m** calls the function **pp6header.m**. This function accepts an input parameter, fid, which is the file ID of an open PPOD file. It reads and parses the header data in the first 8K bytes of the data file. **pp6header.m** returns a data structure containing all the elements of the header. When called by **FindPPODVersion.m**, only the firmware string field is used. The function looks for a string characteristic of firmware version 5 or 6 and returns an appropriate value.

**readppod5.m and readppod6.m** One or the other of these functions is called by longloadppod.m, depending on the version number returned by **FindPPODVersion.m** Each function accepts a file name from those generated by the open file dialog. The function reads the header again, using **pp6header.m**. This time the program uses the calibration coefficients from the header to process the data read from the file. The processed data is returned to the calling function (**longloadppod.m**) as a structure with many different elements. However, only dstime, pr1, and tm1 are concatenated into the final output file.

Both **readppod5.m** and **readppod6.m** call the function **convert\_ppod6.** This function accepts as inputs the header data structure and vectors containing the pressure and temperature periods extracted from the data file. It returns an array with the pressure and temperature data calculated using the Paroscientific algorithm.

**readppod5.m** and **readppod6.m** are under 100 lines in length—about half of which is comments. They have been written to be part of modular processing system, but each of them can be called in standalone operations by calling them with a file name as a parameter. When new versions of the PPOD firmware are written, a new processing function can be easily adapted and integrated into **longloadppod.m** with the addition of about 5 lines of code.

## Limitations

(To be determined as I get more experience with the code and it gets tested by other users.)

## Using the Software

In order to use **longloadppod.m**, you will need to copy the program and its support functions to a folder in your Matlab path. You will need to copy the following programs:

**longloadppod6.m**

**readppod5.m**

**readppod6.m**

**pp6header.m**

**convert\_ppod6**

You can then call **longloadppod6.m** and specify an output structure:

**mydata – longloadppod6;**