DAQ Data Translation Program

**Overview:**

The DAQ Data Translation Program facilitates the translation of plain text data to the mechanical team’s preferred csv file format. During this process it computes RPM data, translates analog readings to sensor outputs, and averages those sensor outputs down to defined polling rates. The program is installed via a python program distributed amongst GTOR team members, and, once installed, the program is able to update itself at the users request.

**How to Use:**

The program features a basic tkinter interface consisting of a couple of buttons and labels. The program features two main screens, home and the data translation tool, the first of which is pictured below.

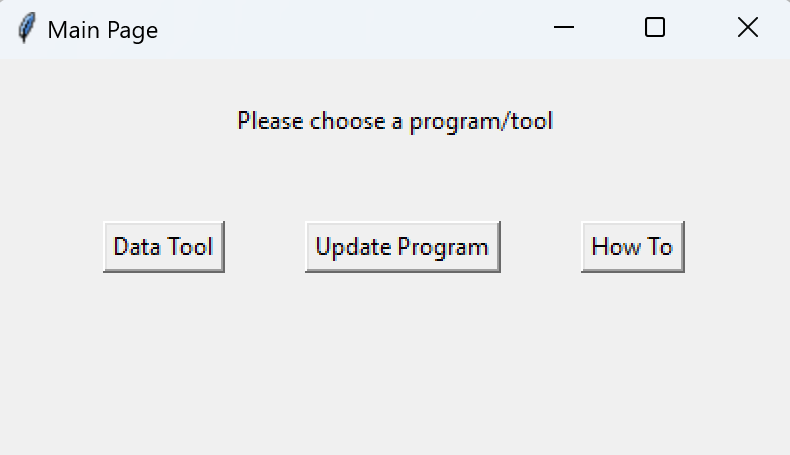


Figure 1 – Home Page: This page allows users to either open the data tool, update the program, or open the general how to guide.

If the user elects to update the program upon initial startup the tkinter GUI will close and a succession of terminal commands will be executed to redownload all the program files, scan them for their python libraries, and install/update all python libraries. The user is required to restart the program after performing such an operation.

If the user elects to open the how to guide a simple notepad window will open the HomeScreen.txt file packaged in the program’s Guides folder. This document contains basic information about the program and is updated on a regular basis to reflect any changes to the program.

If the user elects to open the data translation tool the following window will open.

A screenshot of a computer

Description automatically generated

Figure 2 – Initial Data Translation Tool Window: This is what the user sees immediately after pressing the Data Tool button.

This screen features a how to guide specific to the operation of this section of the program in addition to a choose file button. The choose file button opens a file explorer window to the current file path, allowing users to navigate to their desired data file located on the Mech E drive. Once selected the screen will appear as follows.

A screenshot of a computer

Description automatically generated

Figure 4 – Data Tool Screen With File Selected

After selecting a file the user is required to download it to their local storage to enable faster data handling (if it’s already downloaded (filepath is from the C drive) this step is not needed). This could be changed in the future, but the incredibly slow wifi in the SCC led to the decision to lock this down. After downloading the file, the user can select it from their installation folder which will bring up the following screen.

A screenshot of a computer

Description automatically generated

Figure 5 – Fully Fledged Data Tool Screen

This screen allows the user full access to all the Data Translation Tool’s features. Process Data will create a .csv file based on the parameters detailed in config (downloaded automatically in the previous section), Edit Config will open the config file in a notepad window to allow for customization, and Calculate Hertz Info will analyze the file and report a collection of statistics about it. The use of these features will not be further detailed in this guide as their specifics are likely to change with future updates.

**How It’s Made:**

The Data Translation Tool is made entirely in python. The DAQ.py file imports a succession of dependencies located in subfolders which each play a vital role in the program’s ability to function. The overall program structure goes as follows.

Figure 6 – Data Translation Tool Flow Chart

DAQ.py calls everything else from itself, with some subprocesses calling their own subprocesses to fulfill their requirements. Almost all subprocesses are created as a thread to allow for a parallelized work-flow, with serial operations only being performed when one needs to be completed before another can begin/continue.

Each of the files used in this program are thoroughly commented, and it is recommended for DAQ members to familiarize themselves with them before building additional functionality into the tool.

**Config Files**

The config files tell the program what sensors were on the car during testing, alongside information on how the data should be processed.

A computer screen shot of a computer code

Description automatically generated

Figure 7 – Example Config File

The aspects of the config entries are as follows.

1. Sensor Index: This is the index at which the sensor appears in data files and should never have to be changed by non-DAQ members.
2. Sensor Data Type: This specifies if the sensor uses a digital or analog transmission method.
3. Sensor Name: This is the name of sensor and is used by the program to know what data translation libraries need to be used.
4. Polling Rate: This is the polling rate the sensor’s data will be averaged to. To set this number use the “Calculate Hertz Info” to find the minimum polling rate seen throughout the data collection process. DO NOT SET THE POLLING RATE ABOVE THIS VALUE. As long as you don’t violate this rule, feel free to play around with this to either average out noise or get better resolution.
5. Num Teeth: This only applies to hall-effect sensors and should never be modified by non-DAQ members.

**Rules and Regulations:**

In order to add some consistency to the backend of this program a few formatting rules have been laid down.

1. Have all imports at the top of the file with a 1 line gap between the last important and the first comment, line of code, or quite literally anything. This is necessary as the installer loops through each file until it finds the first empty line.
2. Have all gui elements controlled by their relevant functions. This means that functions should be passed their necessary args (like a file path or whatev) in addition to the main data tool page and the new tkinter page created for the function call. This helps break the code up for easier debugging.
   1. Note how the only things passed to DataTranslator.translateData are the filepath of the data, the progress bar page, and the original data processing page. Look inside the DataTranslator.py file to see the creation of its labels and progress bar.A close-up of a logo

      Description automatically generated

Figure 8 – Example Code for function calls

**Update Log:**