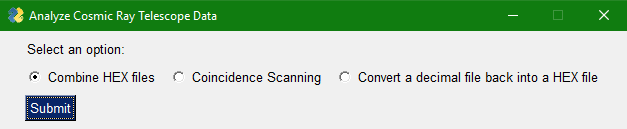
**Analyze CR Data.py**

**Contents**

1. **Combine Hex Files**
2. **Coincidence Scanning**
3. **Convert from Decimal back to HEX**

**Overview**

To operate the program, the program uses a GUI (graphical user interface) to communicate with the user. This GUI was designed using the PySimpleGUI library, which is a wrapper for Python’s built-in Tkinter GUI library. Some infrequently used functions accept terminal-based input instead of GUI input. This documentation is subdivided into 3 major categories because of how the program is really a combination of these 3 separate options:

****

1. This function itself is subdivided into three major components. Mainly, this includes: combining HEX files into one large HEX file, converting the contents of that HEX file back into decimal, and then generating an excel file to display the contents of the decimal file. While all of this happens, the function processes potential GPS strings and timeskip errors in the data.
2. This function accepts a folder of two or more decimal text files as input. There cannot be any other files or directories in that folder, or the program will fail. Those files will be scanned for detection coincidences within a scan window. The scan window is a quantized duration of time in units of the circuit boards smallest clock increment (which is 1 / 244.1 Hz or about 4ms).
3. This function was made for users who accidentally deleted a HEX file, and only have the decimal file. This program creates a copy of that file in HEX again.

**Advanced Description**

1. The user must open one of the many small HEX files using the GUI. The program will use this file to locate the rest of the small HEX files, which are assumed to be in the same folder. The program identifies each HEX file by its file name, which must follow the pattern: “F\*.txt” where “\*” is a wildcard symbol. A wildcard symbol is a variable representing any string of text; in our case the wildcard represents an integer. The program also prompts the user for a name for the new output data. This name is used for the name of the new folder, and for the names of all output files (a combined HEX file, a decimal file, and an excel file) as well as the light curve. That folder will be saved to the same directory that contains the smaller HEX files. At the user’s convenience, the program can delete all of the smaller HEX files for the user because they are now obsolete. Note that no data correction occurs when the HEX file is combined; correction only occurs during the conversion from HEX to decimal. Statistics from the timeskip corrections are printed at the top of the decimal file. Additionally, possible GPS strings in the data are parsed and written to the top of the decimal file during the conversion from HEX. Both the GPS strings and the timeskip corrections are later copied to the excel file. The user can manually select each GPS string that will be parsed, in case some of them are inaccurate. It is important to mention that the GPS strings are parsed according to a timestamp that was polled from the circuit right after the collection of the GPS string.

Now let’s talk about the excel file. The program has been designed to take care of some of the data analysis in excel. Particularly, a light curve is generated from bins of the data. This light curve is a histogram graph of the counts in each bin as the recording goes on. A bin refers to a duration of time (say from *t*=0s to *t*=300s, and then the next bin is *t*=300s to *t*=600s, and so on) during which detections are counted. For example, there may have been 550 detections that occurred within the first 300s, and we would say that the first bin had 550 counts. The user may select the bin size, but the default and recommended size is 337.5s. This particular bin size allows a target frequency of 1/86400Hz to be a discrete frequency on the frequency spectrum from the Fast-Fourier Transform (FFT). This frequency corresponds to a signal that has a 24hr period, and we believe this frequency is correlated to a periodic diurnal effect where the muon flux changes depending upon the time of day. Some statistics for the light curve are also generated. The FFT is not a part of the program, it is performed using Excel’s data analysis package.

In addition to generating a light curve, the program will generate statistics on the time intervals between subsequent detections on another sheet. Note that presently, these statistics can take a long time to generate and this function should be turned off for larger data sets. The time interval data will be graphed as a histogram as well. Because muon detections are independent events that occur with a near-constant true rate, the graph of the intervals is expected to behave like a decaying exponential. Therefore, a log-scaled graph of counts within each interval is expected to be linear. Seeing that these interval graphs follow the expected pattern helps confirm that we are observing muons, and this data appears to be explained very well by these models.

1. Coincidence analysis is performed by combining the timestamps within each decimal file that is present in the selected folder. Each timestamp is associated with the file or telescope it came from by an identifying integer (0, 1, 2...). Afterwards, those timestamps and their associated integers are compiled into an ordered list, which is sorted by increasing timestamps. At this point, how the coincidences are chosen from that sorted list can become complicated. There have been many variations upon how this sorted list is processed, and these variations should dependent upon the hardware and dead time. All versions require the user to decide upon a “scan window” that describes the maximum time separation between coincident detections. Note that this time separation ought to be quantized by the circuit-board’s sub-second clock oscillation period. Therefore, the scan window has been represented as an integer argument or parameter. Note that the scan window is a range that begins from the first timestamp in a coincident detection. A scan window of “0” corresponds to only detections from different telescopes that have the exact same timestamp. Muons are expected to arrive at the same time on the order of microseconds, but due to electronic delays and complications a scan window of about 4 is needed to capture all of the data, but most of the coincidences are simultaneous. See the appendix for a more detailed description of the current scanning algorithm.
2. This function contains some strange code because it was written to work with older versions of decimal files without even prompting the user. It has even become obsolete now because it is rare that the user will delete the HEX file now, and no other parts of the program require the HEX file.

**Appendix**

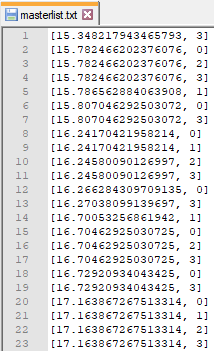
**Time Corrections**

Three consecutive timestamps are compared to infer if a correction needs to be made. The corrections are performed under the assumption that there are “timeskip” errors where a timestamp might skip to the next second by mistake or be a second behind by mistake. Note that we try to avoid overcorrecting the data. The first time is assumed to be reasonable within every comparison. Then, expected times for the second and third timestamp are computed based upon a fourth timestamp. This means the program is not intended to handle an errant fourth timestamp, and if too many data corrections occur then the data may be compromised by poor clock circuitry. Typically, there are only about 2 corrections performed for every 1000 detections. The fourth timestamp is only checked if an issue was found with the first three timestamps. Let timeA represent the first timestamp, timeB is the second timestamp, and so on. There are never any changes made to the *sub-seconds* of a timestamp, only the seconds. Also, none of the timestamp changes result in cascading effects where all subsequent timestamps are increased or decreased by 1 second. Here are the scenarios leading up to correction and the corrections that are performed:

|  |  |
| --- | --- |
| **Situation** | **Outcome** |
|  | ***None*** |
|  | ***None, cannot be processed*** |
|  | *timeC* inherits the seconds of *timeD*  *timeB* inherits the seconds of *timeA* |
|  | *timeB* inherits the seconds of *timeA*  *timeB* inherits the seconds of *timeC* |
|  |  |

**Coincidence Scanning Algorithm**

Here is an example of what the “master” combined list of timestamps looks like before it is processed. However, the full list might be millions of detections long.

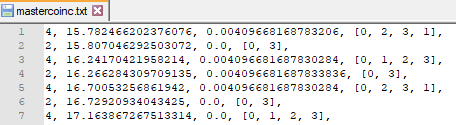


each entry before the comma is the timestamp in seconds, and each entry after the comma is a label for that telescope. You might notice that lines 15-17 all contain the exact same timestamp, but the detections came from 3 different telescopes. Then, line 14 contains a similar timestamp from a fourth timestamp that is only about 4 milliseconds behind. It is the objective of the program to count these 4 detections as a coincidence, knowing that these detections all came from a simultaneous pulser signal. A coincidence is defined as near-simultaneous detections that come from unique telescopes.

In reality, it is possible for muon showers to occur within close time proximity to each other. For this reason, we begin constructing a list of coincidences by starting with a small scan window of 0, meaning only simultaneous coincidences are counted and stored in a list. This list is full of elements containing 4 entries each, where each entry contains information about the coincidence. After searching through the whole list and collecting all simultaneous detections, the program deletes all of these simultaneous detections from its list before searching this list again for detections that have a time separation of 1 unit, and so on.

On its subsequent searches, the program first seeks to expand the size of its simultaneous coincidences. For example, the program would have identified lines 15-17 as being coincident on its first search. On its second search, this coincidence will be expanded to include lines 14-17. After checking for the expansion of already seen coincidences, the program looks for completely new coincidences. For example, lines 12 and 13 would not have been counted as a coincidence on the first search of the program with a window of 0 because those are not simultaneous detections. However, on the second search with a window of 1, those detections will be counted as a new coincidence. To reiterate, every time a detection is added to a coincidence, it is removed from future searches of any kind. The searches occur with increasing time, which biases the list. Suppose one telescope had a detection both 4ms before a simultaneous coincidence and 4ms after that coincidence. Only the first detection would be considered part of the coincidence due to time-order bias.

Here is how the coincidence list looks when that portion of the master list is processed:



The output coincidence data has the following structure:

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Telescopes | Beginning of the coincidence (s) | Time range of the coincidence (s) | Telescope-Identifying Numbers |

where each of these elements are separated by a comma. The number of telescopes will always just be the length of the telescope-identifying numbers. The beginning of the coincidence will always be the timestamp of the earliest detection in that coincidence. The last timestamp in that coincidence could be calculated by adding the coincidence time range to the earliest detection.

There exist older versions of this coincidence-scanning function (called scan\_times in the program). Some of them were designed to be more generous when scanning for coincidences, where coincidences might be allowed to “overlap” and the same detections from one coincidence can be detections in another coincidence.