Proposal: VDIFParse

Mars Buttfield-Addison

April 4, 2022

1 Purpose

Currently, the need to ingest and manipulate telescope data in the modern standard VDIF (or variant CODIF) formats is met by DiFX with its `mark5access` and `vdifio` libraries. However, the need for `mark5access` to also support several legacy formats has prevented it from using practices best suited to the VDIF format, at a significant cost to performance. The limited functionality of the `vdifio` library could not be incorporated into `mark5access` without suffering the inherent slowdown to multithreaded VDIF processing.

This presents an opportunity to create a new library that combines the functionality of the two previous libraries, designed especially for performance with the VDIF format. Coupled with a modern, easy-to-use API, such a library could also fill a gap that would allow quick creation of small VDIF processing programs. Herein proposes such a solution, called `VDIFParse`.

2 Use Cases

3 Technologies

Library is to be written in native C, in conformance with *at least* GNU90 ad C99. Dependencies will be kept to the C Standard Library. The library should target both OSX and Debian-based Linux.

4 API Design

Usage will centre around a struct type `InputStream`, which can be in one of two modes: `FileMode` or `StreamMode`.

In `FileMode`, the expected interaction is that a user will initialise an `InputStream` using the `open_file()` function and passing a filepath to a valid VDIF or CODIF file.

```
#include "vdifparse.h"

int main() {
    // open new stream (also parses first header)
    struct InputStream* in = open_file("file.vdif");


    close(in);
}
```

In `StreamMode`, the expected interaction is that a user will initialise an (initially empty) `InputStream` using the `open_stream()` function and then push raw data into it by monitoring a specific port or piping from another process.

```
#include "vdifparse.h"

int main() {
    // open new stream (most values remain unset)
    struct InputStream* in = open_stream();


    close(in);
}
```

5 Style and Conventions

- Type names are in PascalCase (e.g., `InputStream`).
- Function and variable names are in snake_case (e.g., `sample_rate`).

6 Collected Prompts for Feedback

1. Nowadays, the most popular architectures (x86-64, IA-32, etc.) all use little-endianness. Can you think of a device or prospective used that would still require support for big-endianness?

2. In the current form, it is technically possible to use one `InputStream` in both `FileMode` and `StreamMode` at different times (e.g., open a file and then later stream more data in). Can you think of any conceptual reasons to allow or disallow this?

3. In the style of `mark5access`, it may be advisable to add a domain marker such as a `vp_` prefix to all types or functions, to prevents overloading when imported into other projects. Can you think of any reason not to do this?