**GNSS SDR READER API**

**QUICK START GUIDE**

**Introduction**

Software Defined Radio (SDR) data collection systems and processing platforms designed for Global Navigation Satellite System (GNSS) receiver applications have been growing in popularity over the last several years. After collection of data, these systems output datasets of various formats. In the past, converting and decoding different datasets to other formats has been difficult and error-prone.

To remedy this, the ION GNSS SDR Standard Working Group developed a standard that would make decoding of these files significantly easier. For each dataset produced by a receiver application, a corresponding XML document would be made that contained important metadata about these datasets, such as the frequency of the streams and their encodings. An XML parser was written by [Name] that would extract the data from a metadata file and put it in a C++ object.

This API builds upon the API written by [Name]. With the GNSS SDR Reader API, you can point to a metadata file, and assuming that a corresponding SDR file exists, the samples from that SDR file will be decoded and placed into an output stream. The data from that output stream can be fed into other programs, such as [Tom’s Waterfall Plotter].

For more information about these XML metadata files, go to [Link.Com]

**Build Instructions**

On Windows, simply import into visual studio. On Linux, navigate to the root directory, then execute g++ ….

**Program Design**

Basically, this program relies on a multithreaded approach. One thread moves data from the SDR file into an intermediate buffer, and the other takes data from the intermediate buffer, decodes it, and puts it in an output buffer.

**Basic Usage**

In your application, be sure to include the GnssReader:

#include "GnssReader.h"

To start, create a GNSSReader Object. The minimum amount of arguments that must be supplied is four.

GNSSReader(const char\* pathToFile,uint64\_t readSize,uint64\_t buffSize,uint64\_t streamSize, uint64\_t blockTotal = -1,const char\*\* addlPaths = NULL, uint64\_t pathCount = 0);

Where

const char\* pathToFile Represents the absolute path to the metadata file

uint64\_t readSize Represents the transfer size in bytes from the SDR to the Intermedate Buffer.

uint64\_t buffSize Represents the size of the intermediate buffer, in terms of readSize.

uint64\_t streamSize Represents the size of the Outputstream(s).

Optional Arguments:

uint64\_t blockTotal = -1 Represents the amount of blocks to read (default: -1 for all)

const char\*\* addlPaths = NULL, Represents additional paths to look for SDR files on (default: NULL for none)

uint64\_t pathCount = 0, Represents the count of addlPaths

GnssReader can throw std::exceptions at runtime if there are XML parsing errors, so be sure to catch any it may throw.

Assuming no exceptions were thrown, the next thing to do is to allocate memory to create the output streams. The class function

GnssReader::makeDecStreams();

does this. Now, it is time to start decoding. Call

test4.startAsThread();