ION Software-Defined Radio Metadata Standard Final Report

Sanjeev Gunawardena, *Air Force Institute of Technology*
Alexander Rügamer, *Fraunhofer Institute for Integrated Circuits IIS*
Muhammad Subhan Hameed, Markel Arizabaleta, Thomas Pany, *Universität der Bundeswehr München*
Javier Arribas, *Centre Tecnològic de Telecomunicacions de Catalunya*

BIOGRAPHIES

Sanjeev Gunawardena is a research professor with the Autonomy and Navigation Technology (ANT) Center at the Air Force Institute of Technology at Wright-Patterson AFB, Ohio. He serves as co-chair for the ION GNSS SDR Metadata Standard working Group.

Alexander Rügamer received his Dipl.-Ing. (FH) degree in Electrical Engineering from the University of Applied Sciences Würzburg-Schweinfurt, Germany, in 2007. Since then he has been working at the Fraunhofer Institute for Integrated Circuits IIS in the Field of GNSS receiver development. He was promoted to Senior Engineer in February 2012. Since April 2013, he is head of a research group dealing with secure GNSS receivers and receivers for special applications. His main research interests focus on GNSS multi-band reception, integrated circuits and immunity to interference.

Muhammad Subhan Hameed studied Electrical Engineering at National University of Sciences and Technology (NUST) in Pakistan and is currently pursuing a masters degree in space science and technology from Technical University Munich (TUM). His research interests focus on satellite navigation and GNSS receiver technology.

Markel Arizabaleta is a research associate at the Universität der Bundeswehr München since 2017 and has a Master's Degree in Telecommunication Engineering. He has been working with the ION SDR metadata standard since 2017. His contributions to this effort includes updates to the standard document, normative software C++ code debugging and optimization, and performing consistency checks between the standard document and source code.

Thomas Pany is with the Universität der Bundeswehr München at the faculty of aerospace engineering and leads the navigation group within Institute of Space Technology and Space Applications (ISTA). He is working with GNSS since 1997 and with software radio technology since 2002. He has around 200 publications including one book and five patents. He serves as co-chair for the ION GNSS SDR Metadata Standard working Group.

Javier Arribas is Senior Researcher at the Statistical Inference Dept., Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain. He received the MSc in Telecommunication Engineering from La Salle University in 2004, and the PhD from UPC in 2012. His primary areas of interest include statistical signal processing, detection and estimation theory, GNSS, SDR receivers, FPGA prototyping and RF front-end design. He is the recipient of the 2015 EURASIP Best PhD Award.

ABSTRACT

The ION GNSS SDR Metadata Standard describes the formatting and other essential PNT-related parameters of sampled data streams and files. This allows processors to seamlessly consume such data without the need to input these parameters manually. The technical development phase of the initial version of the standard has now been deemed complete and is currently undergoing the last remaining procedural steps towards adoption as a formal standard by the Institute of Navigation. This paper reports on the activities of the working group since September 2018 and summarizes the final products of the standard. It also reports on examples of early adoption by academic, research, and open source SDR projects. This includes an example where, as intended, capabilities covered in the standard have been expanded to describe custom data embedded within sampled data streams. Also included is an example where the standard is used to describe non-GNSS data. Several examples of commercial adoption are also documented in this paper. The standard website (sdr.ion.org) includes a repository containing SDR files of various topologies and formats. The working group is currently in the process of providing the satnav observables for these datasets using well-known SDRs in order to serve as a reference for those working to adopt the standard products into their own software projects. This paper also includes results from this effort.

INTRODUCTION

The ION GNSS SDR Metadata Standard arose out of a need that was first discussed at the ION GNSS+ 2013 meeting in Nashville Tennessee, to come up with a standard way to describe the sampled data file formatting and other important parameters relevant to satnav SDR processing (referred to as GNSS SDR metadata), in such a way that an SDR could conceptually handle any of these files without having to manually enter these parameters. After 6 years of voluntary effort by the Standard Working Group, this vision has largely been realized and this free and open standard development has been deemed complete. All technical documentation and reports generated as part of this development can be found in http://sdr.ion.org/presentations-and-publications.html. The draft standard document can be found in https://github.com/IonMetadataWorkingGroup/GNSS-Metadata-Standard/tree/master/Specifications/documentation. This paper represents the final report on this effort. Activities since September 2018 and the final steps toward formal ratification are summarized below. An overview of the normative reference software and current status is described next. The remainder of the paper reports on current adoption and use cases of the standard by academia and research organizations, as well as availability of commercial products that have adopted the standard. Finally, early results on the effort to provide observation data to go with the test and verification datasets found in the ION SDR data repository are presented.

STATUS REPORT (ACTIVITIES SINCE SEPTEMBER 2018)

The second public request for comments (RFC2) was launched on September 24, 2018. Notice of RFC2 appeared in the ION GNSS+ 2018 printed program, was emailed to the ION distribution list (9,296 recipients), appeared in the Fall 2018 issue of the ION Newsletter and was also posted on the ION website. The deadline for submitting comments was December 31, 2018. The working group received a total of eight comments for RFC2. These were mostly related to minor issues in the standard document and were resolved through a revision. Letters acknowledging their comments and the working group's resolutions were sent to each RFC2 respondent. The comments, and the working group's responses and resolutions are reported in http://sdr.ion.org/RFC2_Comments_Responses.html.

During this period the working group also made minor revisions to the normative software and merged the development branch (which included revisions resulting from RFC2) into the master branch. More details on the normative software are given in the next section.

During this period the working group also made minor revisions to some of the metadata files associated with the reference datasets: http://sdr.ion.org/api-sample-data.html

Following the activities summarized above, the standard development effort was deemed complete. On August 21, an electronic poll was sent out to each voting member of the Working Group seeking their input on proceeding to the next step: the ION legal review, as stipulated in the terms of reference http://sdr.ion.org/TermsOfReference_042114.pdf. This procedure was discussed at our Working Group meeting during ION GNSS+ 2018:
https://s3.amazonaws.com/sdr.ion.org/Presentations_and_Publications/2018/WG-Presentation-Minutes-Attendees-2018.pdf. .
A reminder message was sent on August 29 to 41 contacts who had not yet voted, and this poll concluded on September 4, 2019. Out of the 57 voting members, there were 27 approvals and no disapprovals. However, this equates to a participation of 46%. During the ION GNSS+ meeting, informal conversations with voting members revealed that some had not gotten around to voting and others were on summer vacation and had not seen the email in time to vote. For these reasons, during our most recent meeting it was decided to re-open the vote to those who did not originally participate in order to increase participation. This new poll is likely to begin on or near October 7, 2019.

Note: the Working Group is comprised of satnav SDR subject matter experts from across the globe representing academia, government, and industry. In a few cases more than one member is affiliated with the same organization (e.g. due to change of employment since this effort commenced). A voting member is one that is designated to represent his/her affiliated organization.

Next Steps

Following the approval of the standard by a majority of the working group voting membership, the next step is the legal review by ION-appointed council. The legal review will scrutinize all documentation and records generated since this effort was started to determine whether all activities were performed objectively with adequate opportunity for public participation and discourse. After the legal review, that report and the standard products will be presented to ION Council for adoption as a formal ION standard. Once adopted, the standard will be made formal (as opposed to a draft) with revision number 1.0.

OVERVIEW OF NORMATIVE REFERENCE SOFTWARE AND CURRENT STATUS

The goal for developing the normative reference implementation that accompanies the formal standard document was to reduce to practice the conceptual design developed through consensus by the working group into an appropriate set of schema (according to industry best practices), and to develop the compliant software library implementation. The working group was fortunate to receive voluntary participation for this task from individuals representing established commercial GNSS SDR vendors. Indeed, the contributors to this repository, as evidenced by browsing through the commit history, represent satnav SDR architects and developers who are well known in the ION community.

Early on during the development of this standard, the working group decided to work on a publicly available reference library to be released along with the standard document. The goal was to promote early and widespread adoption of the standard by making it easy for vendors and researches to integrate standard compliance by integrating the library into their existing software. The scope of this effort was twofold: to develop a metadata interpreter and to develop a binary data converter using this metadata interpreter. Additional details about the normative software can be found in [1]. The GitHub repository for this open source software is https://github.com/IonMetadataWorkingGroup/GNSS-Metadata-Standard/tree/master.

Although major development of the normative software is deemed complete, the working group recognizes that minor issues and bugs may be discovered through usage. We plan to continue to maintain this repository and perform updates and revisions on an as-needed basis. Those wishing to volunteer and contribute to these efforts are encouraged to contact the working group co-chairs.

EXAMPLES OF STANDARD ADOPTION BY ACADEMIA AND RESEARCH

Even during its development phase, several research and academic organizations have adopted the standard and incorporated the normative software into their satnav SDR implementations and tools. This section includes a brief cross section of these, in no particular order.

Flexiband/GTEC front-end by Fraunhofer IIS

The Flexiband/GTEC front-end has been developed by Fraunhofer IIS and commercially available via TeleOrbit. It has been designed as a mobile USB3.0 data recorder device supporting three configurable RF-tuners per unit that can be tuned to different frequency bands within the L-Band or S-Band or to the same band but then having separate RF inputs. A common and synchronized sampling clock of 81 MHz is used to digitize the analogue baseband data from the RF-tuners with 8 bit I/Q. Since all ADCs are driven from the same reference clock, all samples of one or of a combination of Flexiband units are synchronized within one sample. The data streams are provided to an FPGA that carries out digital filtering, down-sampling, reduction of the bit-width as well as multiplexing of the data-stream including embedding of checksums to ensure or to at least detect potential data losses. This digital-data stream is then transferred via a USB3.0 controller to a PC. Continuous streaming rates of 1.296 GBit/s have been realized in many projects; higher data rates are possible too, however, the I/O capacity of the PC is often the bottleneck [2].

The Flexiband's multiplexed raw data stream is saved to hard-disk together with a XML-file (*.sdrx) containing its description according to the ION Metadata Standard. Some exemplary samples of different configurations can be found on http://www.iis.fraunhofer.de/flexiband and http://sdr.ion.org/api-sample-data.html.

Multi Sensor Navigation Analysis Tool (MuSNAT) by Universität der Bundeswehr München

MuSNAT is a high-end GNSS software-defined transceiver that has been developed to fulfil modern day requirements of navigation application areas. The website for MuSNAT is https://www.unibw.de/lrt9/lrt-9.2/software-packages/musnat. Along with conventional GNSS receiver functionality (acquisition, tracking, position computation), MuSNAT supports multi-frequency, multi-GNSS and multi-sensor processing which makes it suitable for a wide range of applications areas including GNSS signal quality monitoring/analysis, positioning in RTK or PPP modes as well as integration of GNSS and other navigation technologies such as INS or LiDAR.

At the backend of MuSNAT's processing chain is the MuSNAT Core that processes the input GNSS and other sensor data and logs all the results into an SQL database. A dedicated graphical user interface then allows the user to visualize the data in a time synchronous way in order to identify relationships between the input signals and receiver performance parameters.

For reducing the overall processing time, MuSNAT also supports a parallel execution mode that shifts the processing load of correlation computation from CPU to an Nvidia CUDA compatible GPU.

It is also possible to use MuSNAT as a software-defined transmitter by reverting the data flow inside the MuSNAT Core. This enables MuSNAT to generate a full constellation GNSS-Signal with variable signal power and navigation message that can be broadcasted using a digital-to-analog converter. Hence, MuSNAT realizes the GNSS-transceiver concept.

The ION metadata standard is fully supported in MuSNAT in addition to the legacy reading routines from separated IF sample files (one for each frequency band). This enhancement gives us the possibility to process data from virtually any GNSS signal source or from signals of opportunity like Long Term Evolution (LTE) signals. The file reader found in the ION normative software is still slower than our internal reading routine but future work on it will also target the ION reader's performance improvement [3].

GNSS-SDR
GNSS-SDR is an open source project that implements a global navigation satellite system software defined receiver in C++ [4]. With GNSS-SDR, users can build a GNSS software receiver by creating a graph where the nodes are signal processing blocks and the lines represent the data flow between them. The software provides an interface to different suitable RF front-ends and implements all the receiver's chain up to the navigation solution. Its design allows any kind of customization, including interchangeability of signal sources, signal processing algorithms, interoperability with other systems, output formats, and offers interfaces to all the intermediate signals, parameters and variables.

The goal is to provide efficient and truly reusable code, easy to read and maintain, with fewer bugs, and producing highly optimized executables in a variety of hardware platforms and operating systems. In that sense, the challenge consists of defining a gentle balance between level of abstraction and performance, addressing:

- Concurrency
- Efficiency
- Performance
- Portability
- Ability to run in real-time or in post-processing
- Extendibility

The GNSS-SDR software receiver runs in a common personal computer and provides interfaces through USB and Ethernet buses to a variety of either commercially available or custom-made RF front-ends, adapting the processing algorithms to different sampling frequencies, intermediate frequencies and sample resolutions. It also can process raw data samples stored in a file. The software performs signal acquisition and tracking of the available satellite signals, decodes the navigation message and computes the observables needed by positioning algorithms, which ultimately compute the navigation solution. It is designed to facilitate the inclusion of new signal processing techniques, offering an easy way to measure their impact in the overall receiver performance. Testing of all the processes is conducted both by the systematic functional validation of every single software block and by experimental validation of the complete receiver using both real and synthetic signals. The processing output can be stored in Receiver Independent Exchange Format (RINEX), used by most geodetic processing software for GNSS, or transmitted as RTCM 3.2 messages through a TCP/IP server in real-time. Navigation results are stored in KML and GeoJSON formats. For more information and setup details, the reader is referred to the project website https://gnss-sdr.org/

Although GNSS-SDR does not currently incorporate the ION metadata standard, support is planned for a future revision. Later in this paper, GNSS-SDR and MuSNAT are used in a zero baseline configuration on a reference data set to compare and validate observables (i.e. range measurements). The ultimate goal of this effort is to provide observables data in RINEX format for all the reference SDR datasets located in the ION repository (sdr.ion.org) to compare also with other GNSS SDR like pyChips and MuSNAT.

pyChips
pyChips is a post-processing satnav SDR developed primarily to serve as a free and open-source educational tool. It was initially used in the ION GNSS+ 2018 tutorial titled "Hands-on Introduction to GNSS Software Receivers and Signal Processing" (https://www.ion.org/gnss/upload/GNSS18OnsiteProg.pdf). pyChips is now available on GitHub: https://github.com/sanjeevg2/pyChips.

Python has become one of the top-5 programming languages during recent years and continues to gain popularity within science and engineering communities. Major reasons for this rapid adoption seem to be the fact that Python itself is free and several integrated development environments (IDEs) are also freely available. Many of these IDEs support state-of-the-art productivity enhancing features such as code completion and background code analysis and highlighting for errors and style. Although Python is a high-level interpreted language, powerful packages such as numpy (https://numpy.org/) help to accelerate numerical computation performance. For example, the initial version of pyChips ran a single GPS-SPS tracking channel operating on a GN3S data file (16 MHz sample rate, 8-bit samples represented as 2-bit sign-magnitude values) faster than real-time on a 2015 Intel Core-i7 class laptop.

Encouraged by the relative ease of learning and coding in Python and the pleasantly surprising performance of the initial version, it was decided to expand capability to support multi-constellation satnav (GPS and Galileo for starters) on multiple bands (L1/E1 and L5/E5a). Due to the higher bandwidth of signals in the L5/E5a band, data collection systems typically employ packed samples to keep the data rate and file sizes reasonably small. For example, the LabSat 3 Wideband device packs 1, 2 or 3-bit baseband samples from up to three data streams into 64-bit unsigned integers before writing to disk [5]. It was discovered that decoding such packed data formats natively in Python is slow to the point that the SDR is unusable (decoding a 1 ms block of wideband dual frequency LabSat3 Wideband data takes on the order of seconds)! The numpy bit-shifting functions seem to be the bottleneck responsible for this slow performance.

Fortunately, as with most high-level languages, Python includes a foreign function library called ‘ctypes’ that allows calling functions in dynamically linked libraries (DLLs) or shared libraries (.SO files in Linux). (https://docs.python.org/3/library/ctypes.html). Using this interface, functions performing low-level operations can be developed in C/C++ and compiled to run optimally in the host CPU, or even bridged into higher performance computing platforms such as GPUs and FPGAs.

pyChips currently employs ctypes to perform the SDR file reading and low-level sample decoding functions. The approach is shown in Figure 1. The ION Metadata library is wrapped into a C-callable DLL named MetadataCLib.dll. This DLL converts the metadata object tree in the ION library into a hierarchy of structures. The metadata for the target SDR file found within these structures is parsed, and a suitable low-level sample decoding handler is attached at runtime. This approach of using a custom handler for the expected SDR file format helps to improve performance. Currently decoders for a few file formats have been developed (GN3S, Ohio University TRIGR, LabSat3 Wideband, and USRP). However, since the project is open source, users can easily add handlers to support other file formats. This is part of the reason why the metadata parsing and file reading/decoding were split into two DLLs: MetadataCLib.dll and cChips.dll, respectively. Only the latter needs to be revised and compiled to add support for a new SDR file format.

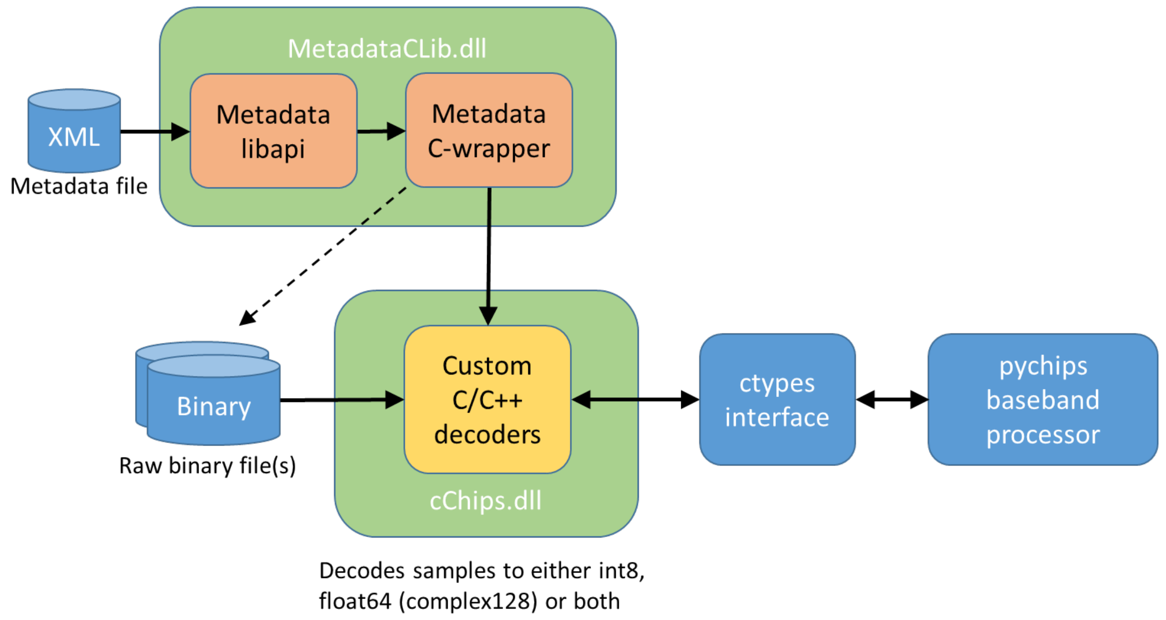


Figure 1: Block diagram illustrating the interfaces to the ION metadata standard library and the custom file decoders used in pyChips

It is also important to note that the custom handlers have a common input/output interface and decode the packed samples into 8-bit integer and/or 64-bit floating point data types (real or complex depending on the front-end architecture). This is also determined at runtime depending on which types are actually used in pyChips. In general, integer decoding is active when correlation operations are enabled. Floating point decoding is active only when the spectrum analyzer is running (which uses numpy's floating point FFT functions). This avoids runtime typecasting in the python/numpy environment which also helps to improve performance.

By employing the ION Metadata Standard and associated normative software, pyChips can be revised easily to support any SDR file format as needed (more specifically this actually means revising the cChips.dll code).

EXAMPLES OF INDUSTRY ADOPTION AND COMMERCIAL PRODUCTS USING STANDARD

The standard has seen early adoption by several commercially available products as listed below. Additional examples of adoption can also be found in:
https://s3.amazonaws.com/sdr.ion.org/Presentations_and_Publications/2018/Ruegamer__ION_Metadata_Examples_v03.pdf

- GTEC© GNSS Radio Frequency Front-End and MGSE Record and replay device from TeleOrbit GmbH: https://teleorbit.eu/en/satnav/gtec/

- SX3 GNSS Software Receiver from IFEN GmbH: https://www.ifen.com/products/sx3-gnss-software-receiver/

- GIPSIE (GNSS multisystem performance simulation environment) GNSS Simulator from TeleConsult Austria: https://www.tca.at/produkte/gnss-processing/gipsie

EXAMPLE OF EXTENSION TO INCLUDE NON-STANDARD INFORMATION

This section describes an example of extending the capability of the standard to include data that is not part of the defined set of metadata classes. In the working group's perspective, this is an expected and encouraged use case, and is the mechanism whereby the standard may evolve in future revisions to cover additional capabilities that are currently not considered to be the core set of essential metadata classes. It should be noted however that the present working group's scope is limited only to the initial version of the standard.

Flexiband Front-End with Embedded U-Blox Data

One RF tuner in the Flexiband front-end described previously in this paper can be exchanged with a COTS u-blox NEO-M8T timing receiver to be included in the Flexiband housing, as shown in Figure 2. This u-blox receiver has the ability to trigger the recording of all other units (both internal data streams as well of external Flexiband data recorder units) with respect to its absolute GNSS-based time reference and its PPS pulse. Moreover, the u-blox ubx-binary protocol itself is embedded in the digital data stream w.r.t. the PPS, as shown in Figure 3.

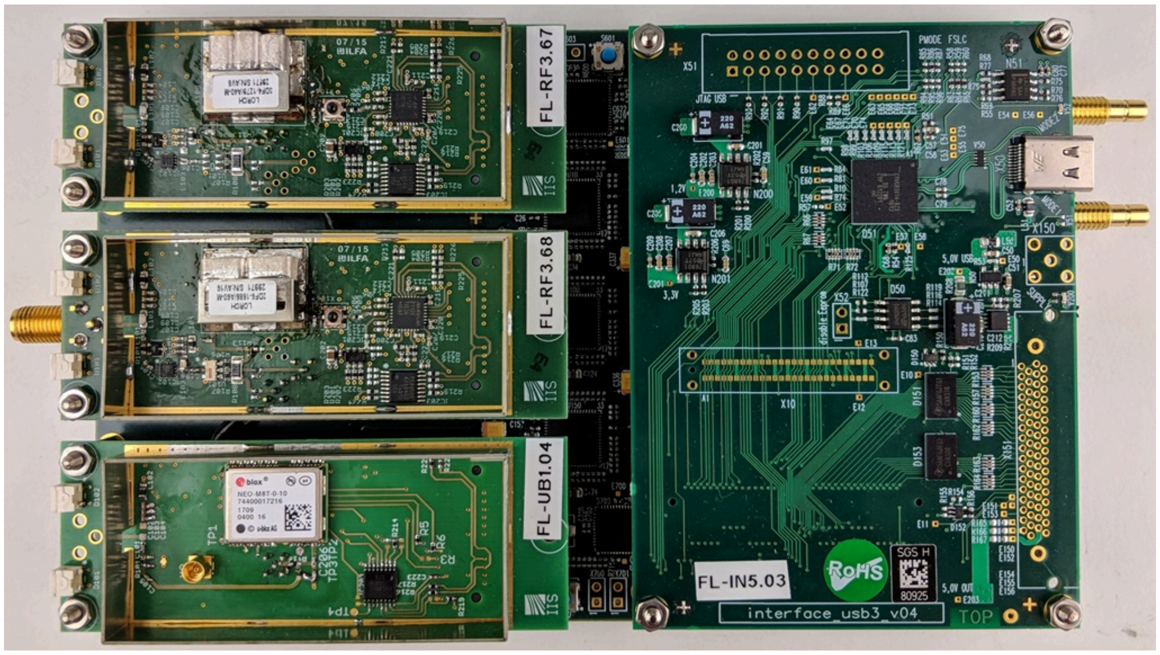


Figure 2: Flexiband data recorder with two RF-tuners (top and middle , left side) and one integrated u-blox receiver instead of an RF tuner (bottom, left side)

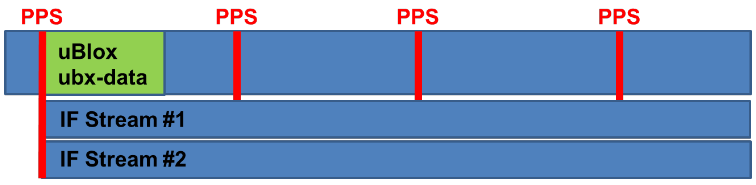


Figure 3: IF front-end data streams aligned to the u-blox PPS and with ubx embedded information

This provides many advantages in the post processing: Firstly, one always has an absolute timing reference with the accuracy of the u-blox PPS pulse (thus <100ns). Secondly, absolute position information is available. Thirdly, one can use the ubx multi-GNSS information of GPS, Galileo, GLONASS, and BeiDou as assistance data for its own processing of the raw IF data streams.

In this particular design, two RF tuners are used to receive a Galileo E1 and E6 data stream with 50 MHz bandwidth and 81 MHz of sampling rate using 4 bit I/Q quantization. The data streams are multiplexed to blocks of 1024 bytes length, as outlined in Figure 4. With 6 bytes header (2 bytes preamble and 4 bytes serving as a counter to detect packet loss) and 6 bytes footer (2 byte data padding and 4 byte CRC), 506 bytes for the E1 and E6 raw data are available.

To embed the u-blox ubx binary protocol (in this case, the NAV-SVINFO, NAV-PVT and RXM-RAWX measurements, adding up to 580 byte if 10 satellites are received) into the data stream, the two padding bytes of the footer are reused: the first byte of the footer indicates with its magic number ‘E6’ that a ubx binary data byte follows. Later on, the ubx message can be extracted and reconstructed from the raw, multiplexed data stream providing assisted information on the recorded raw data samples.

Since the u-blox receiver and the front-end use their own reference clock, after some time, the raw samples of the front-end might not be aligned to the PPS pulse anymore. Therefore, the information to which data word the PPS pulse occurs is also embedded into the footer. Thus, if the PPS pulse occurred within one block, its relative position is encoded into the data bytes of the footer. This provides the new relation of the raw data to the u-blox PPS with a resolution of 1/sampling rate, in this case approx. 12 ns (1/81 MHz). Moreover, if the footer indicates the PPS positon and this led to a collision with the ubx binary message, the ubx binary would be shifted to the next packet, in order to keep it consistent and avoid data loss. Lastly, in case byte 1013 and 1014 are 0xFF 0xFF, no PPS occurred in that block and no u-blox ubx information is included. Figure 5 shows an example metadata file for this use case.

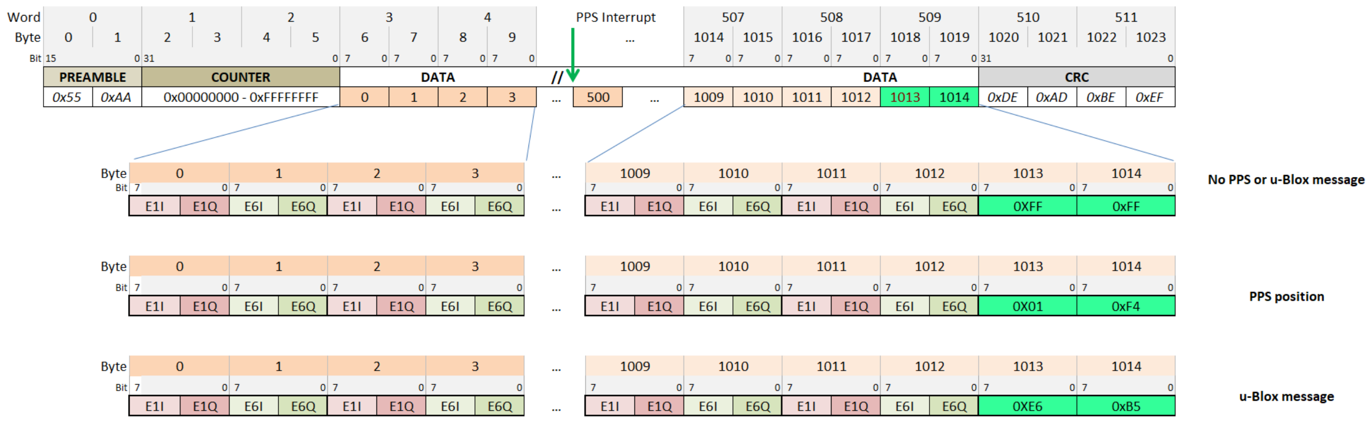
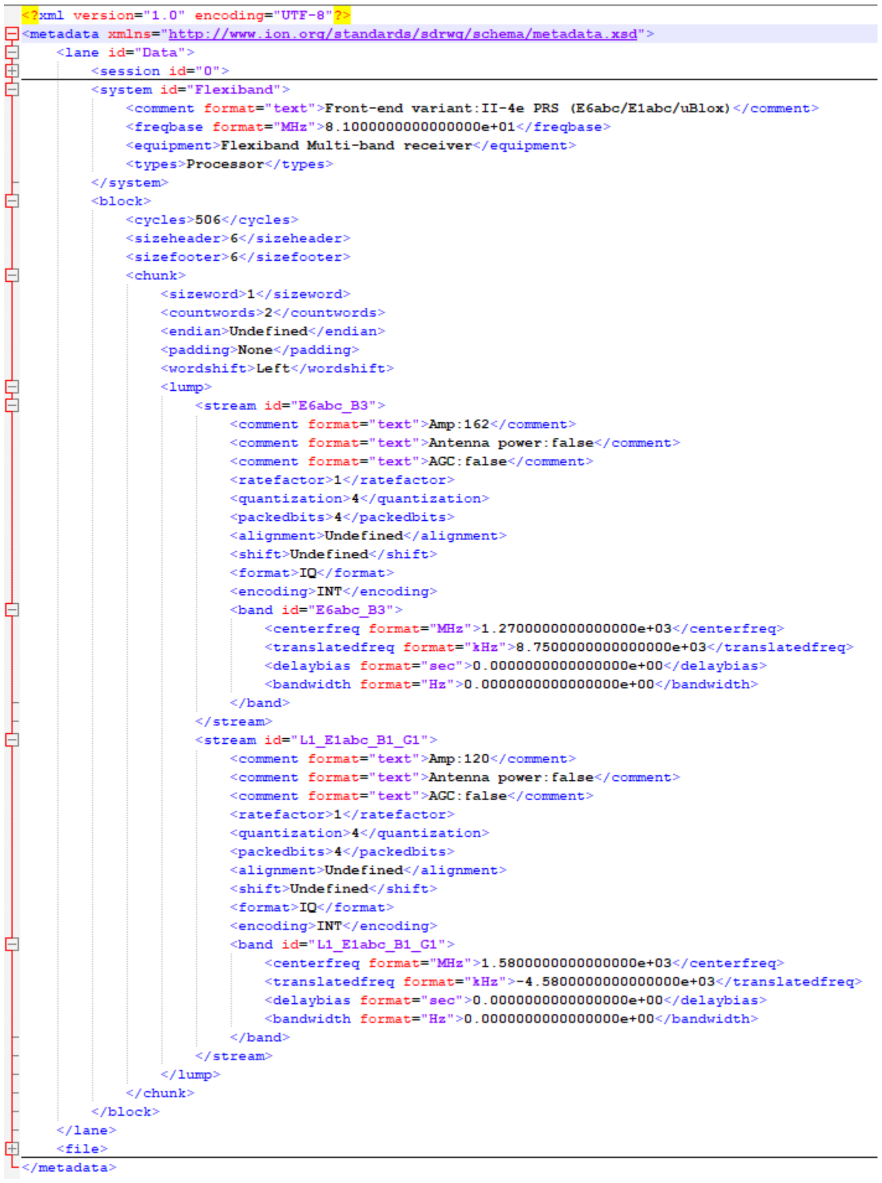


Figure 4: Outline of the used multiplex protocol



```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://www.ion.org/standards/sdrwg/schema/metadata.xsd">
    <lane id="Data">
        <session id="0">
        <system id="Flexiband">
            <comment format="text">Front-end variant:II-4e PRS (E6abc/E1abc/uBlox)</comment>
            <freqbase format="MHz">8.1000000000000000e+01</freqbase>
            <equipment>Flexiband Multi-band receiver</equipment>
            <types>Processor</types>
        </system>
        <block>
            <cycles>506</cycles>
            <sizeheader>6</sizeheader>
            <sizefooter>6</sizefooter>
            <chunk>
                <sizeword>1</sizeword>
                <countwords>2</countwords>
                <endian>Undefined</endian>
                <padding>None</padding>
                <wordshift>Left</wordshift>
                <lump>
                    <stream id="E6abc_B3">
                        <comment format="text">Amp:162</comment>
                        <comment format="text">Antenna power:false</comment>
                        <comment format="text">AGC:false</comment>
                        <ratefactor>1</ratefactor>
                        <quantization>4</quantization>
                        <packedbits>4</packedbits>
                        <alignment>Undefined</alignment>
                        <shift>Undefined</shift>
                        <format>IQ</format>
                        <encoding>INT</encoding>
                        <band id="E6abc_B3">
                            <centerfreq format="MHz">1.2700000000000000e+03</centerfreq>
                            <translatedfreq format="kHz">8.7500000000000000e+03</translatedfreq>
                            <delaybias format="sec">0.0000000000000000e+00</delaybias>
                            <bandwidth format="Hz">0.0000000000000000e+00</bandwidth>
                        </band>
                    </stream>
                    <stream id="L1_E1abc_B1_G1">
                        <comment format="text">Amp:120</comment>
                        <comment format="text">Antenna power:false</comment>
                        <comment format="text">AGC:false</comment>
                        <ratefactor>1</ratefactor>
                        <quantization>4</quantization>
                        <packedbits>4</packedbits>
                        <alignment>Undefined</alignment>
                        <shift>Undefined</shift>
                        <format>IQ</format>
                        <encoding>INT</encoding>
                        <band id="L1_E1abc_B1_G1">
                            <centerfreq format="MHz">1.5800000000000000e+03</centerfreq>
                            <translatedfreq format="kHz">-4.5800000000000000e+03</translatedfreq>
                            <delaybias format="sec">0.0000000000000000e+00</delaybias>
                            <bandwidth format="Hz">0.0000000000000000e+00</bandwidth>
                        </band>
                    </stream>
                </lump>
            </chunk>
        </block>
    </lane>
    <file>
</metadata>
```

Figure 5: Metadata example for the E1/E6/u-blox design

In conclusion, this demonstrates that thanks to the flexible design of the ION Metadata Standard also such proprietary extensions can easily be introduced. Similarly, also other data like INS information could be embedded. However, if the data rate of the auxiliary information exceeds the few hundred bytes like in the ubx example given before, a separated stream within the multiplexed raw data should be defined and used.

EXAMPLE OF METADATA SUPPORT FOR DATA COLLECTED FROM REMOTE SITES

IRNSS S-/L-band measurement campaign in Hanoi, Vietnam

The Southeast Asia region has one of the highest navigation satellite reception destinies in the world. With also the regional component of BeiDou available, and the regional navigation satellite systems (RNSS) QZSS and IRNSS/NavIC, typically, more than 50 navigation satellites are receivable [6]. Within the H2020 project "BELS+" TeleOrbit and Fraunhofer had the opportunity to setup its MGSE "Multi-GNSS Simulator & Test Environment" at the NAVIS laboratory at the University of Hanoi, Vietnam. To be able to assess the full L-band and S-band navigation signals, dedicated antennas have been installed on the rooftop of NAVIS, too.

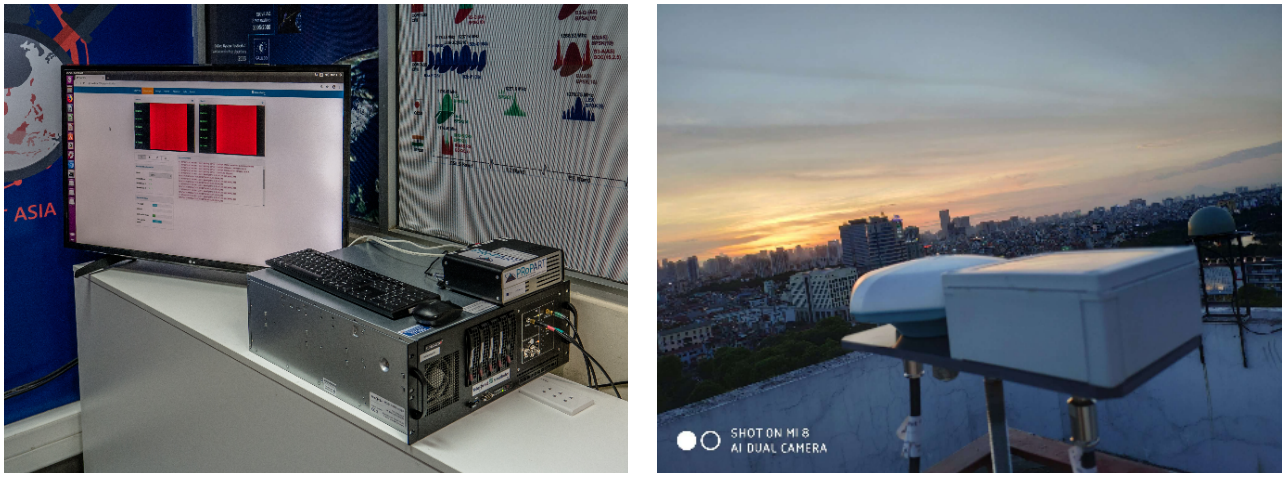


Figure 6: MGSE recording and replay system (left), L- and S-band antenna at NAVIS, Hanoi (right)

For the data recording unit, the previously introduced Flexiband/GTEC has been integrated into the MGSE housing. Moreover, the date recorder was extended with a specific S-band recording module covering the 16.5 MHz bandwidth from 2483.5 – 2500.0 MHz of the IRNSS/NavIC of India.

Three front-end configurations were realized, in order to support different applications (see Figure 7):

- Single Band, high bandwidth, high dynamic for interference assessments:
  - L1, 54 MHz BW, 81 MHz fs, 8 bit I/Q
- Dual Band, high bandwidth, medium dynamic for dual-band interference assessments:
  - L1, 54 MHz BW, 81 MHz fs, 4 bit I/Q
  - E5, 54 MHz BW, 81 MHz fs, 4 bit I/Q
- Triple Band, high bandwidth full IRNSS/NavIC support:
  - L1, 54 MHz BW, 81 MHz fs, 4 bit I/Q
  - L5, 38 MHz BW, 40.5 MHz fs, 2 bit I/Q
  - S-Band, 18 MHz BW, 20.25 fs, 4 bit I/Q

Via the FPGA firmware of the data recorder, the specific front-end configuration is determined. The raw data are provided together with the Meta Data file representing the individual multiplexed data formats. Thanks to a remote access to the system running in Hanoi, raw data recordings can be made at different instances in time providing a unique opportunity e.g. to analyze ionosphere scintillation effects and their impact on the super wide IRNSS L5 – S-band combination.

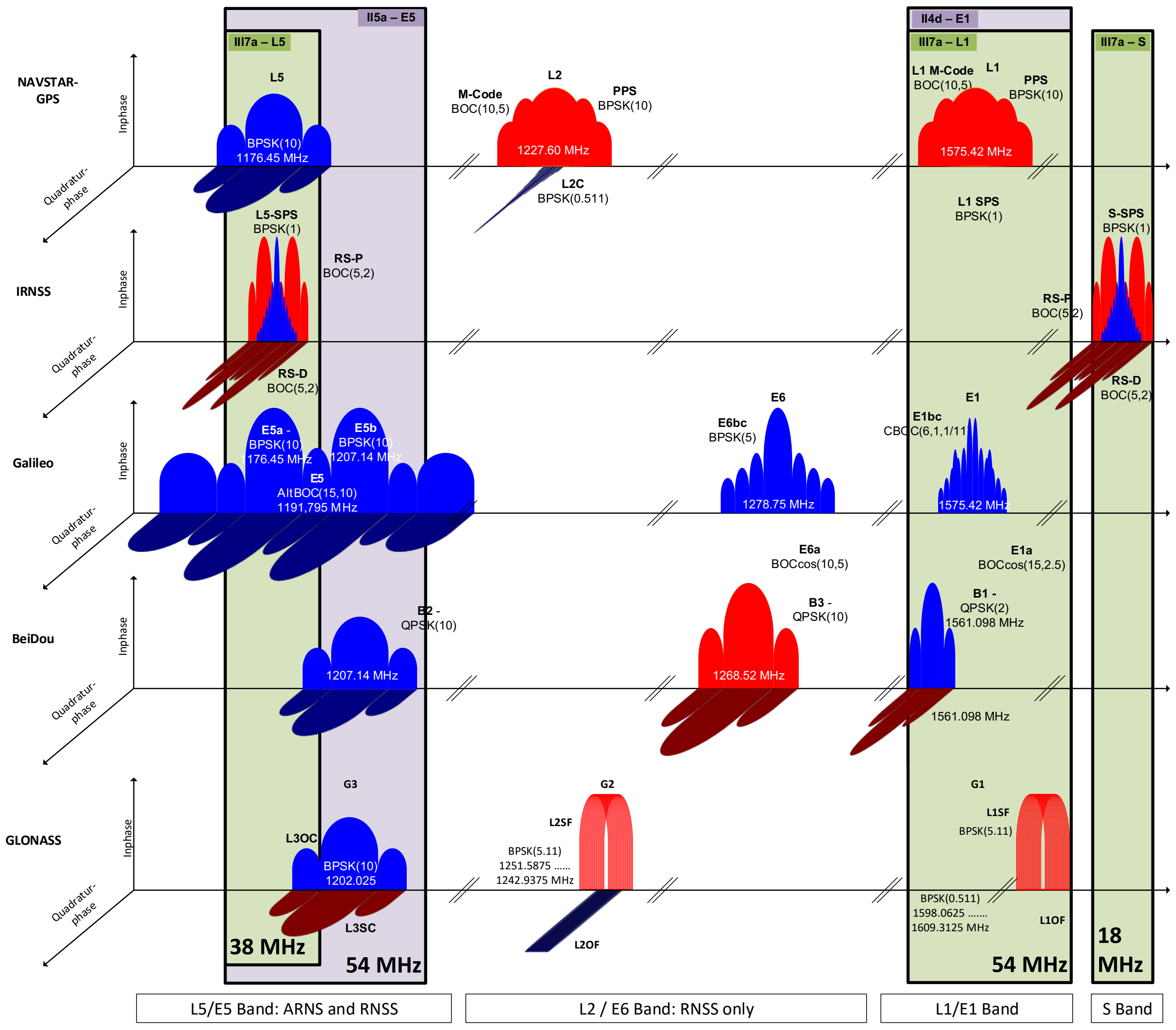
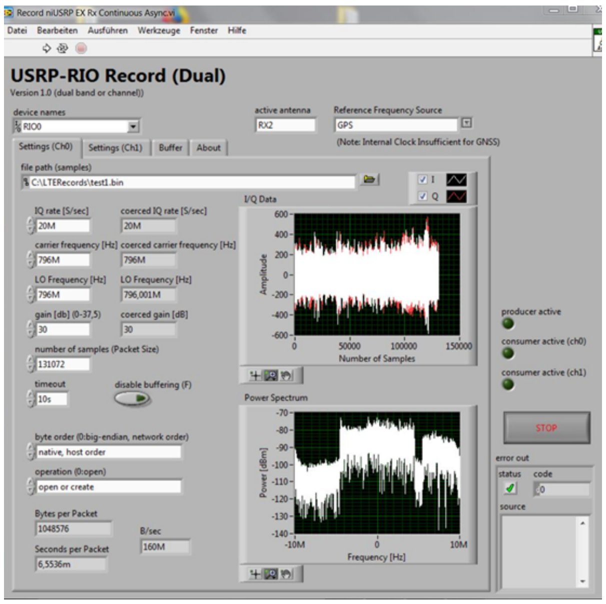


Figure 7: Configurations used for the Hanoi tests

EXAMPLE OF METADATA STANDARD USED TO DESCRIBE NON-GNSS DATA

As was initially envisioned, the metadata standard is not limited to GNSS signals in the L-band but can be applied to describe signals of opportunity for use in PNT. This section describes the application of the standard for describing a long-term evolution (LTE) cellular signal collection using a USRP RIO device from National Instruments and logging software developed at UniBwM [7].

The application of the metadata standard for these types of signals is straight forward and Figure 8 shows the data collection application of the USRP RIO that has been configured for the LTE signal. Figure 9 shows the metadata standard file for the LTE signal. Finally, Figure 10 shows the metadata parameters decoded from the metadata file in MuSNAT. The data will be made available at sdr.ion.org together with an indication which LTE transmitters are contained in the samples.



USRP-RIO Record (Dual)
Version 1.0 (dual band or channel))

device names: RIO0
active antenna: RX2
Reference Frequency Source: GPS
(Note: Internal Clock Insufficient for GNSS)

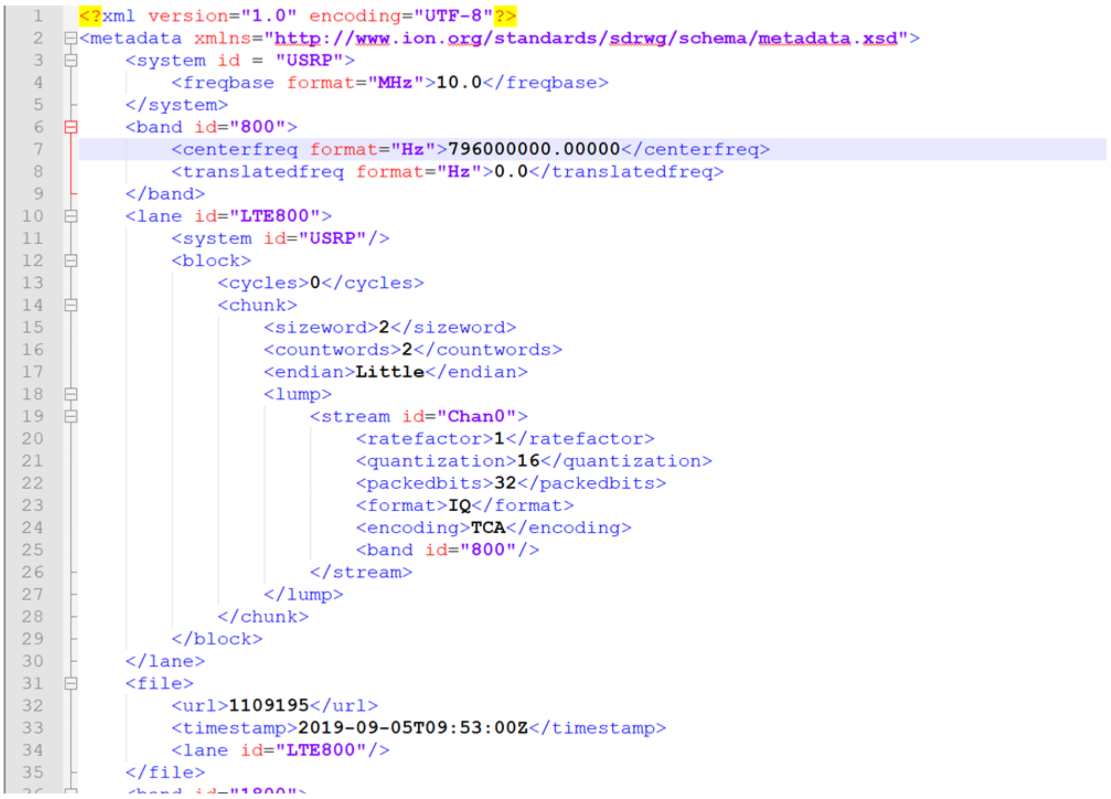
file path (samples): C:\LTERecords\test1.bin

IQ rate [S/sec]: 20M	coerced IQ rate [S/sec]: 20M
carrier frequency [Hz]: 796M	coerced carrier frequency [Hz]: 796M
LO Frequency [Hz]: 796M	LO Frequency [Hz]: 796,001M
gain [db] (0-37,5): 30	coerced gain [dB]: 30
number of samples (Packet Size): 131072	
timeout: 10s	disable buffering (F)

byte order (0:big-endian, network order): native, host order
operation (0:open): open or create
Bytes per Packet: 1048576
B/sec: 160M
Seconds per Packet: 6,5536m

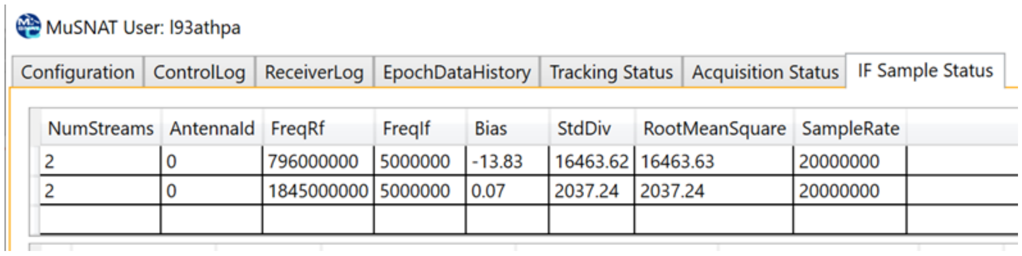
error out
status: ✓ code: 0
source

Figure 8: data collection application of the USRP RIO



```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://www.ion.org/standards/sdrwg/schema/metadata.xsd">
    <system id = "USRP">
        <freqbase format="MHz">10.0</freqbase>
    </system>
    <band id="800">
        <centerfreq format="Hz">796000000.00000</centerfreq>
        <translatedfreq format="Hz">0.0</translatedfreq>
    </band>
    <lane id="LTE800">
        <system id="USRP"/>
        <block>
            <cycles>0</cycles>
            <chunk>
                <sizeword>2</sizeword>
                <countwords>2</countwords>
                <endian>Little</endian>
                <lump>
                    <stream id="Chan0">
                        <ratefactor>1</ratefactor>
                        <quantization>16</quantization>
                        <packedbits>32</packedbits>
                        <format>IQ</format>
                        <encoding>TCA</encoding>
                        <band id="800"/>
                    </stream>
                </lump>
            </chunk>
        </block>
    </lane>
    <file>
        <url>1109195</url>
        <timestamp>2019-09-05T09:53:00Z</timestamp>
        <lane id="LTE800"/>
    </file>
    <band id="1800">
```

Figure 9: Metadata file for the LTE signal capture



NumStreams	AntennaId	FreqRf	FreqIf	Bias	StdDiv	RootMeanSquare	SampleRate
2	0	796000000	5000000	-13.83	16463.62	16463.63	20000000
2	0	1845000000	5000000	0.07	2037.24	2037.24	20000000

Figure 10: MuSNAT IF Sample Status Tab

RINEX OBSERVATION FILES FOR NORMATIVE SDR DATA REPOSITORY AT SDR.ION.ORG

In this section we discuss the experimental set-up and results of a zero-baseline test performed between MuSNAT and GNSS-SDR. The test was realized by using the same IQ data stream as the input signal source for both the software receivers. The purpose of this test is to evaluate the consistency of observables from two independently developed SDRs when presented with the same input file. The ultimate goal of this effort is to perform this analysis for all the SDR datasets contained in the ION data repository and to make available the observation data in RINEX format in order to serve as a reference for those who will be using these datasets to incorporate and test the metadata standard libraries into their own SDRs.

The IQ data set was downloaded from Fraunhofer's reference data archive [8]. The data set is 210 sec in duration and was realized by using Fraunhofer Flexiband USB Front-end to record IQ data samples at a sampling rate of 20 MHz for two frequency bands covering GPS L1, L2 and Galileo E1. A third frequency band was recorded with a sampling rate of 40 MHz covering GPS L5 and Galileo E5a.

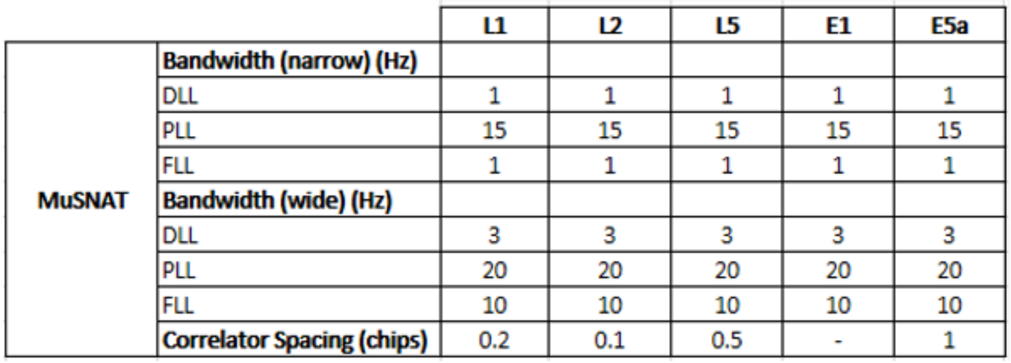
Here we present a comparison between the receiver configuration paradigms of both the software receivers which is summarized in Table 1. Following the ION GNSS Metadata standardization, MuSNAT can read front-end related parameters from an ION GNSS SDR XML file along with the receiver-specific parameters from MuSNAT's own configuration file. The user executes MuSNAT as an SDR processor which invokes the ION GNSS SDR XML file to obtain the required front-end parameters using the open-source ION SDR C++ API. GNSS-SDR reads both front-end related parameters and receiver specific parameters from the same configuration file. Table 1 compares the two approaches used to parse the metadata of a given SDR data file and read and process its contents using an SDR processor.

Table 1: - Comparison between ION GNSS SDR XML and GNSS-SDR Configuration files

ION GNSS SDR XML File	GNSS-SDR Configuration File
XML-schema based file format	Structure/Class based variable assignment
Uses open-source ION SDR C++ API at back-end	Uses open-source *GNSS-SDR* configuration API at back-end
Only implements front-end related (signal source) configuration. The receiver related configuration is implemented within the GNSS SDR processor.	Implements both front-end (signal source) and receiver related configuration in the same file
Provides interface between a generic GNSS SDR processor and sampled front-end data stream	Provides interface between user and *GNSS-SDR* processing chain
User launches GNSS SDR processor which invokes ION SDR XML file.	User launches configuration file through command line

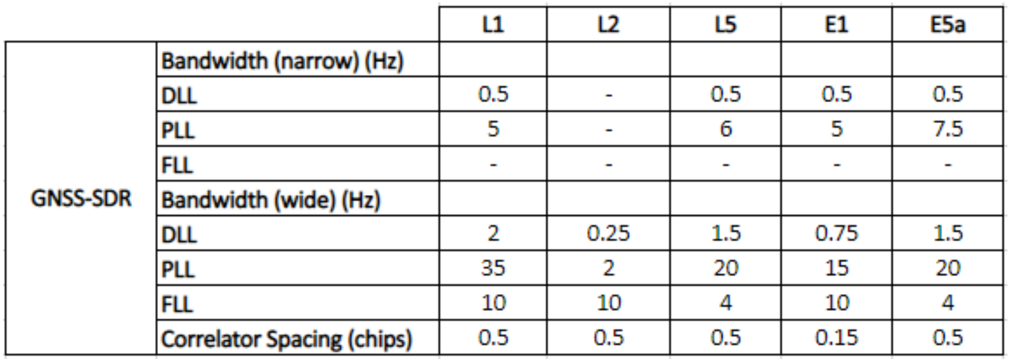
We now present a comparison between the receiver-specific parameters defined within the configuration of both software receivers. Table 2 and Table 3 show the loop bandwidths and correlator spacing used in MuSNAT and GNSS-SDR respectively. MuSNAT was configured to track L1 C/A, L2 CM, L5 as well as Galileo E1 B/C and E5a in wave-formed based tracking mode which computes the early-minus-late correlation at the level of the replica signals. The correlator configuration consisted of 16-bit prompt (P) and early-minus-late (D) correlators in precise correlation mode for low offset tolerance and less reuse of reference signals. GNSS-SDR was configured to track GPS L1 C/A, L2 C, L5 as well as Galileo E1 BC and E5a with predefined correlator spacings (the default correlator spacing used in GNSS SDR is 0.5 chips).

Table 2: - MUSNAT loop bandwidths and correlator spacings



		L1	L2	L5	E1	E5a
MuSNAT	**Bandwidth (narrow) (Hz)**					
	DLL	1	1	1	1	1
	PLL	15	15	15	15	15
	FLL	1	1	1	1	1
	Bandwidth (wide) (Hz)					
	DLL	3	3	3	3	3
	PLL	20	20	20	20	20
	FLL	10	10	10	10	10
	Correlator Spacing (chips)	0.2	0.1	0.5	-	1

Table 3: - GNSS-SDR loop bandwidths and correlator spacings



		L1	L2	L5	E1	E5a
GNSS-SDR	Bandwidth (narrow) (Hz)					
	DLL	0.5	-	0.5	0.5	0.5
	PLL	5	-	6	5	7.5
	FLL	-	-	-	-	-
	Bandwidth (wide) (Hz)					
	DLL	2	0.25	1.5	0.75	1.5
	PLL	35	2	20	15	20
	FLL	10	10	4	10	4
	Correlator Spacing (chips)	0.5	0.5	0.5	0.15	0.5

For post-processing, double-difference measurements were computed using RTKLib [9] from RINEX observation files generated from both the software receivers. The RTKLib campaign was configured to be in Fixed mode with the GNSS-SDR set as the reference receiver and the frequencies were set to L1+L2+L5+E5b to utilize all available signals. Thereby, a position fix was achieved for 32.1 % of the complete duration. The code/phase residuals and carrier-to-noise ratios were exported as text files for each signal and MATLAB was used to obtain the code/phase noise plots (see Figures 8-11).

The phase noise results indicate a similar carrier phase tracking performance of MuSNAT and GNSS-SDR with the phase residuals within 2 mm for L5, E1 and E5a signals. The code residuals, however indicate a larger deviation of code tracking performance with the residuals being within 2 m for L1 and 4 m for E1. The larger deviation in code residuals can be attributed to the difference in DLL bandwidths of both the receivers but further investigation is needed to validate this claim.

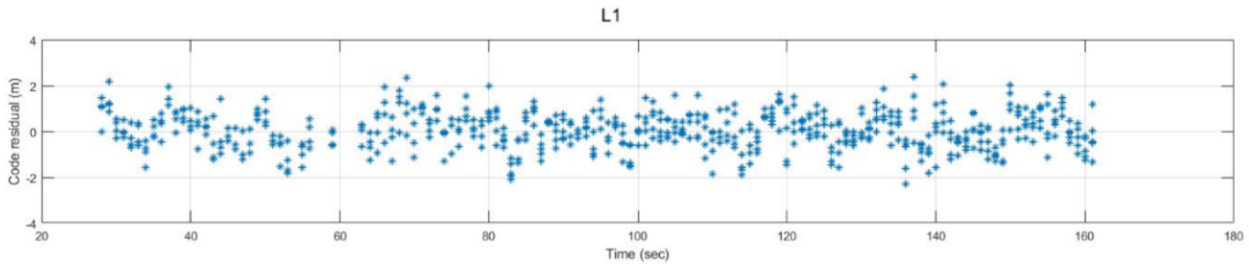


Figure 11: L1 Code Noise

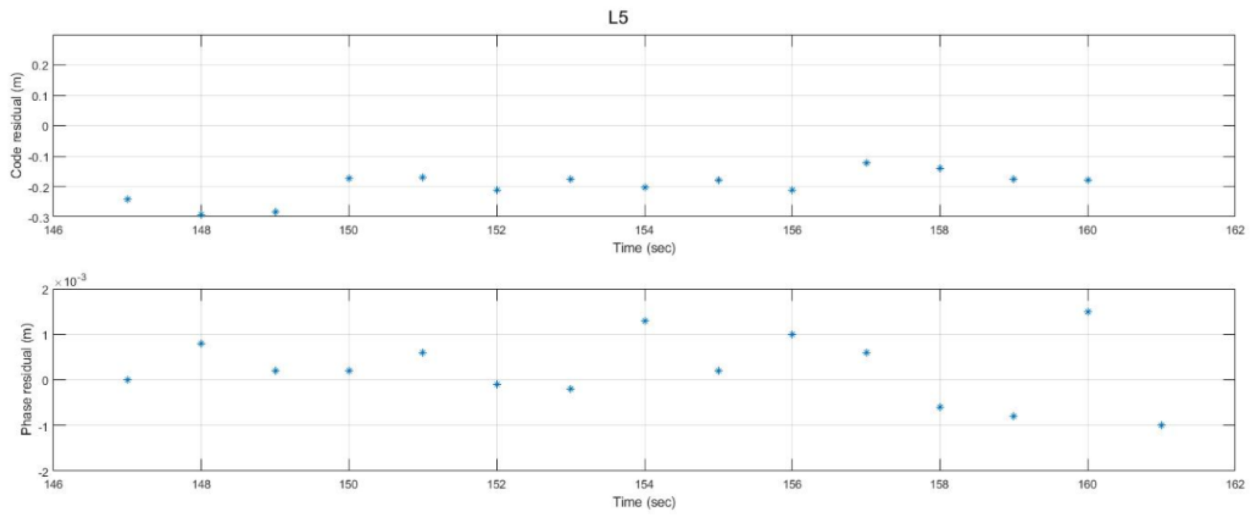


Figure 12: L5 Code Noise and Phase Noise

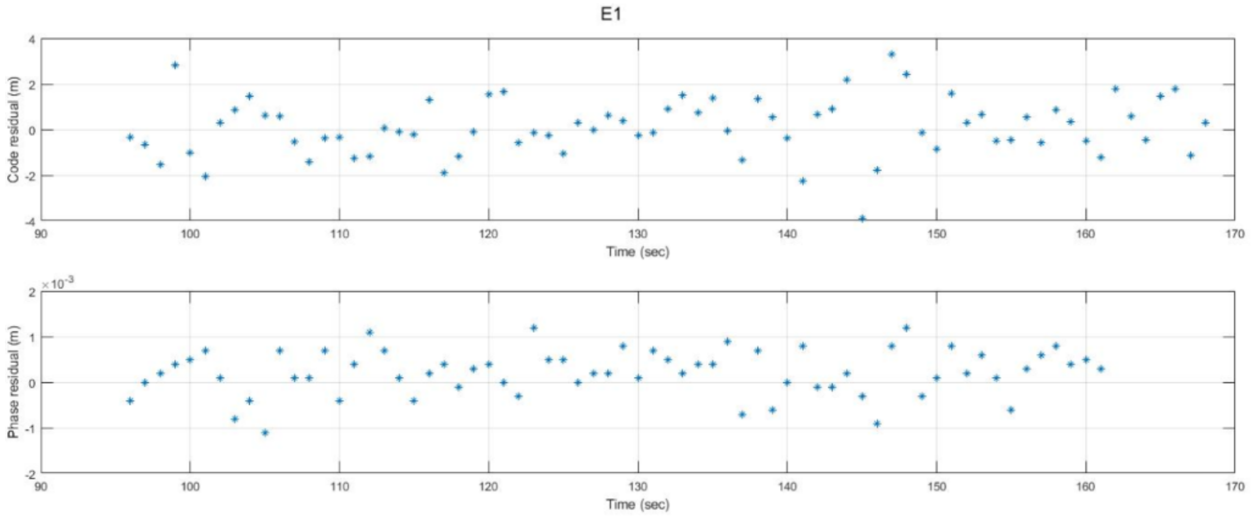


Figure 13: E1 Code Noise and Phase Noise

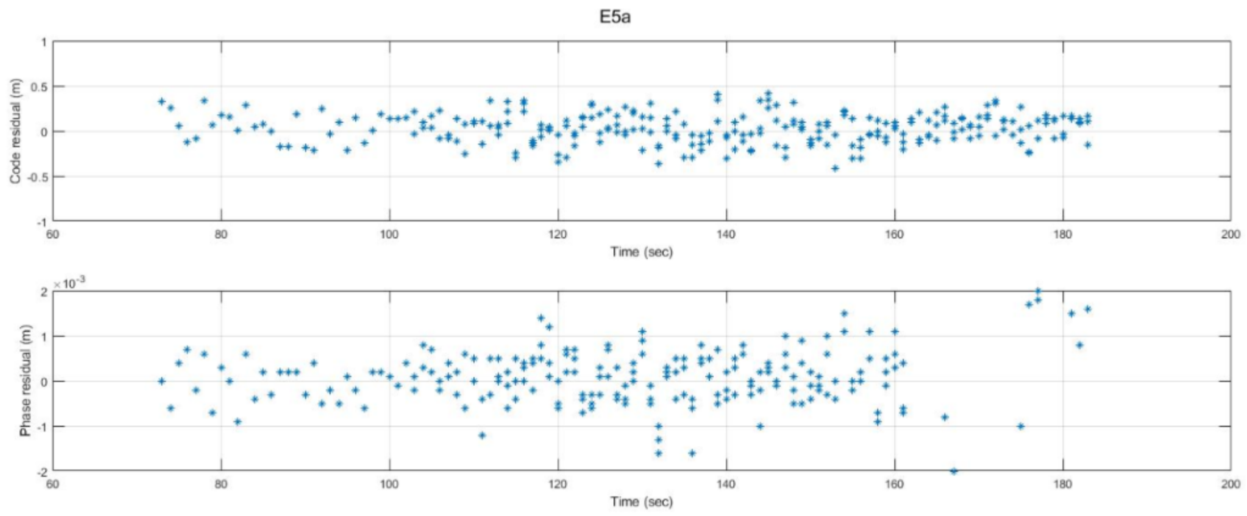


Figure 14: E5a Code Noise and Phase Noise

SUMMARY

This paper represents the working group's final report on the ION SDR Metadata standard. As of this writing, development of the initial version of the standard has been deemed complete. A few more procedural steps remain towards adoption as a formal standard by the Institute of Navigation. The draft standard has already been adopted by several research and academic organizations and is been used in innovative ways, as described in this paper. In addition, the Working Group is pleased to see the availability of several commercial products that have also adopted the standard. We hope that this voluntary effort by the ION SDR community and the products that were developed will continue to be useful for years to come.

ACKNOWLEDGMENTS

The Working Group gratefully acknowledges the support of the ION staff during these six years. Without your help this work would not have been possible. ION leadership and Council is thanked for their continuous encouragement and support. Our over-enthusiastic timeline outlined in the Terms of Reference was way off given a voluntary effort such as this, and we appreciate Council's patience and understanding as we slowly but surely crossed the finish line. We also acknowledge the work of many students, interns, and staff members in each of our organizations that worked on the many aspects of the standard. During these six years, at various times, many members of the working group gave their time and energy and worked tirelessly to make this effort ultimately a success. Some of these members have left the PNT community and no longer active. We sincerely appreciate all you did. Thank you!

The MGSE project has been supported and co-financed by ESA under Contract No. 4000116130. Part of the development of GNSS-SDR has been supported by the Generalitat de Catalunya through projects 2017 SGR 1479 and the Spanish Government through grants TEC2015-69868-C2-2-R and RTI2018-099722-B-I00. The test and measurement campaign at the NAVIS center in Hanoi, Vietnam, has been co-funded by the H2020 BELS+ project.

REFERENCES

1. J. Curran, M. Arizabaleta, T. Pany, S. Gunawardena,"The Institute of Navigation's GNSS SDR Metadata Standard," InsideGNSS, November/December 2017. https://insidegnss.com/wp-content/uploads/2018/01/novdec17-STANDARDS.pdf, accessed September 2019.
2. A. Rügamer, F. Förster, M. Stahl, G. Rohmer, "A Flexible and Portable Multiband GNSS front-end System," Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012), Nashville, TN, September 2012, pp. 2378-2389.
3. T. Pany, D. Dötterböck, H. Gomez-Martinez, M. Subhan Hammed, F. Hörkner, T. Kraus, D. Maier, D. Sanchez-Morales, A. Schütz, P. Klima, D. Ebert, "The Multi-Sensor Navigation Analysis Tool (MuSNAT) – Architecture, LiDAR, GPU/CPU-GNSS-Signalprocessing," Proc. ION GNSS+ 2019, Miami Florida, September 2019.
4. C. Fernández-Prades, J. Arribas, P. Closas, C. Avilés, L. Esteve, GNSS-SDR: an open source tool for researchers and developers , in Proceedings of the ION GNSS Conference 2011, September 19-23, 2011, Portland, Oregon (USA).
5. Racelogic Inc., "LabSat 3 Wideband: Racelogic's compact yet powerful multi-constellation, multi-frequency GNSS device testing system," https://www.labsat.co.uk/index.php/en/products/labsat-3-wideband, accessed September 2019.
6. A. Rügamer and D. Seybold, "Multi-GNSS Measurement Campaign in Southeast Asia using the MGSE-System," 2018 European Navigation Conference (ENC), Gothenburg, 2018, pp. 9-18. doi: 10.1109/EURONAV.2018.8433226
7. National Instruments Inc., "Overview of the NI USRP RIO Software Defined Radio," https://www.ni.com/en-us/innovations/white-papers/14/overview-of-the-ni-usrp-rio-software-defined-radio.html, accessed September 2019
8. ftp://ftp.iis.fraunhofer.de/flexiband/reference-data/L125_III1b_210s.usb.zip
9. T.Takasu, RTKLIB: Open Source Program Package for RTK-GPS, FOSS4G 2009 Tokyo, Japan, November 2, 2009