Microproject at Vidsel Test Range

ASTERIX Radar Formats

Table of Contents

1. Background

At FMV Vidsel, we use radar systems almost every day. From weather balloons to aircraft and missiles, receiving, processing, and interpreting the data formats produced by both our own and external systems is critical in our line of work.

In the past, radar formats have often been developed in-house by the various companies that produce the systems themselves, and standards have not been widely adopted. In recent years, this has changed, and we now have a standardized set of formats across the European Union that practically all radar systems support and use – ASTERIX.

In order for us at Vidsel to provide our customers with what they need, as well as ensure we fly and test safely and securely, we have seen a need to implement and support these standardized formats in our C3 (command and control) systems. To that end, we have developed an internal library for this purpose. The library is presently relatively minimal, and only supports the most critical formats published in ASTERIX – this is a capability we want to extend and improve, such that we can offer even more options for both our customers and ourselves.

The library is currently used in active production, and will continue to be used in critical systems for years to come.

2. Description

2.1 Purpose

This project involves implementing a new radar format in the existing library, using the publicly available ASTERIX specifications and our internal project documentation. The new format will cover one of the categories of the ASTERIX specification, and will be used to interface with new radar systems at the base.

2.2 Scope

In full, the project consists of two aspects.

- Implementation of the format
- Thorough end-to-end testing of the format

Implementation will consist of studying the specifications published by EUROCONTROL (the authors of ASTERIX), and translating them into idiomatic C++ in our internal library. The library already has two categories implemented, and the new code should be based on and follow their structure and implementation as closely as possible.

After (and during) implementation, integration with our automated testing suite will be required. Since the software is and will be part of systems that deal with flight safety in both excercies and live firings, verifying the integrity and robustness of the software is a critical part of development.

This project covers the implementation and intergration of a single ASTERIX category.

3. Requirements

The paragraphs below describe the technical requirements for the project.

3.1 Technical Requirements

3.1.1 Shall

The added software surface shall

- Fully implement an ASTERIX radar format category
- Fully and automatically test the data variants within the format
- Follow the idiomatic and syntactic styles and regulations within the existing codebase
- Be written in C++, targeting the 17th C++ Standard
- Be compilable with GCC 8.3.0
- Be compilable with the existing CMake build system
- Not introduce any additional software or library dependencies into the library

A style guide for the library will be included with the source code. This guide contains a similar section to this one, listing shall/should/may requirements.

During development, the submitted software shall

- Be submitted to FMV via the Git source control system
- Be cryptographically signed with a verified PGP key (for git commits)

3.1.2 Should

The added software should

- Have an automatic code coverage rating of 90% or higher; insofar as it is reasonable and viable

3.1.3 May

No requirements are defined in this section.

3.2 Knowledge Requirements

The paragraphs below describe the knowledge requirements for the applicant.

3.2.1 Shall

The applicant shall (e.g, knowledge required in order to be considered for the project)

- Be confident with idiomatic C++17
- Be confident with basic git operations (add, commit, revert, merge…)
- Be fluent in both reading and writing English

3.2.2 Should

The applicant should (e.g, knowledge that *significantly* improves your fit for the project)

- Be familiar with the CMake build system
- Be familiar with Google Test (GTest)
- Be familiar with reading and writing of binary data streams
- Be familiar with reading and comprehending technical documentation in English

3.2.3 May

The applicant may (e.g, knowledge that improves your fit for the project)

- Be familiar with cryptographic public-key signing
- Be familiar with Linux; in particular, CentOS- and Debian- based systems

3.3 Schedule Requirements

Implementing the format category according to 3.1 is estimated to take four (4) weeks of full-time work. The total project time is therefore, from VTR’s point of view, three to four (3-4) months from the start date, taking into account regular course plans and other engagements.

Status updates will be reported to the project manager on a regular basis, decided on between the applicant and the project manager.

A final date whereupon the project should be fully completed according to the requirements above will be decided on between FMV and the applicant together with the signing of the contract.

3.4 Project Completion

Completion of the project is dependent on fully passing both the integration and automated testing. The submitted source code will be compiled, tested, and integrated at the base with our systems.

After completion of the project, the applicant will be invited to come to the base and hold a presentation of their work to the involved personnel.

4. Provided Tooling and Support

FMV will provide the following software, licences, hardware, and documentation for the duration of the project.

4.1 Software

- JetBrains CLion
- Access to a shared GitHub organization
- Full existing library source code

GCC, the 17th C++ Standard, and other libraries and softwares that the library depends on are not listed here, as they are all gratis open-source software. Installation instructions for the aforementioned softwares are available on their respective sites, or through your operation system's package manager.

4.2 Hardware

No hardware will be provided for this project.

4.3 Documentation

- EUROCONTROL Specification for Surveillance Data Exchange
- Software development guidelines and style requirements
- Sample files (.pcap, .bin, etc) of the format, recorded from live systems

The structure and design of the existing library is documented through in-source Doxygen comments, as well as a README file in the root of the repository.

Relevant EUROCONTROL specifications and documents for the category to be implemented will be provided at the start of the project.

All available specifications and documents can be downloaded freely from EUROCONTROL at https://www.eurocontrol.int/library.

4.4 Point of Contact

Project Manager	Jarl Gullberg
Email	gullberg.jarl@fmv.se
Phone	+46 73 644 96 64

To get in touch with the project manager, use e-mail as the primary contact point. Phone calls should be considered a secondary options, available only during office hours (07:00 to 16:00).

Questions will be answered directly when possible, or forwarded to the appropriate person at VTR.

5. Attachments

1. EUROCONTROL Specification for Surveillance Data Exchange – Part 1
2. EUROCONTROL Specification for Surveillance Data Exchange – Part 4, Category 048, Monoradar Target Reports

Attachment 1 describes the general structure and format of the ASTERIX data protocol. Attachment 2 is the full specification for Category 048, Monoradar Target Reports, one of the protocols that has already been implemented in the library. This attachment is provided as a reference, and does not describe the format that this project will cover implementation of.