30th June 2013

SAR Formation Flying

Annex 5. Developing a Satellite Navigation Receiver for the Space Mission

Document Version: V01_00

Dr. Nagaraj C Shivaramaiah, Dr. Jinghui Wu, Dr. Joon Wayn Cheong, Dr Mazher Choudhury, Kevin Parkinson
Australian Centre for Space Engineering Research (ACSER)
University of New South Wales, Sydney, 2052, Australia

Revision History

Version No.	Date	Author	Description of Change
V01_00	30th June 2013	Garada WP5 Team	Final Release

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY 6

2 INTRODUCTION AND GNSS RECEIVER OVERVIEW 7

3 DESIGN AND DEVELOPMENT METHODOLOGY 8

4 TEST METHODOLOGY 10

5 PROTOTYPE RECEIVER HARDWARE DESIGNS 46

6 SPACE-CERTIFIABLE RECEIVER HARDWARE DESIGN STRATEGIES USED 53

7 FPGA-BASED MULTI-GNSS BASEBAND LOGIC DESIGN 54

8 MULTI-GNSS SIGNAL PROCESSING AND THE DEVELOPMENT OF MATLAB BASED RECEIVER 58

9 FIRMWARE DEVELOPMENT AND DEBUGGING 76

10 TEST RESULT 91

11 GNSS RECEIVER OPERATION AND ENVIROMENTAL TESTING AT DLR 106

12 NAMURU RECEIVER COMMAND AND REPORTS 109

13 FIELD UPGRADE OF THE NAMURU RECEIVER FIRMWARE AND LOGIC 161

14 CONCLUSION AND FUTURE WORK 176

15 REFERENCES 179

TABLE OF CONTENTS (DETAILED)

1 EXECUTIVE SUMMARY 6

2 INTRODUCTION AND GNSS RECEIVER OVERVIEW 7

3 DESIGN AND DEVELOPMENT METHODOLOGY 8

4 TEST METHODOLOGY 10

4.1 Development of the Test tools 10

4.2 Test Equipment 11

4.3 Test cases for functional test 11

4.4 Test procedure: CASE_1 14

4.5 Test procedure: CASE_2 19

4.6 Test procedure: CASE_3 21

4.7 Test procedure: CASE_4 22

4.8 Position Computation 23

4.9 Relative position for Namuru V3.3 28

4.10 Test procedure: CASE_5 Relative PPS test 31

4.11 PPS test between Namuru and truth for Case 1 and Case 2 32

4.12 Test methodology for identifying receiver's performance 32

4.13 Test procedure: Cold start test 34

4.14 Test procedure: Warm start test 38

4.15 Test procedure: Hot start test 39

4.16 Cold start TTL 41

4.17 Warm start TTL 41

4.18 Hot start TTL 41

4.19 Acquisition Sensitivity 42

4.20 Tracking Sensitivity 44

5 PROTOTYPE RECEIVER HARDWARE DESIGNS 46

6 SPACE-CERTIFIABLE RECEIVER HARDWARE DESIGN STRATEGIES USED 53

6.1 Oscillator 53

6.2 Power supplies 53

6.3 Latch-up protection and recovery 54

6.4 Construction 54

6.5 Shielding 54

7 FPGA-BASED MULTI-GNSS BASEBAND LOGIC DESIGN 54

7.1 Baseband module overview 54

7.2 Baseband and processor interface 56

7.3 Garada baseband signal processor features and Summary 57

8 MULTI-GNSS SIGNAL PROCESSING AND THE DEVELOPMENT OF MATLAB BASED RECEIVER 58

8.1 Development considerations 58

8.2 PERFORMANCE ILLUSTRATIONS 62

8.3 GNSS Data Processing and Measurements Output in RINEX 3.0 Format 69

8.4 Results 72

8.5 Concluding remarks 76

9 FIRMWARE DEVELOPMENT AND DEBUGGING 76

9.1 Galileo E1 Firmware Architecture Overview 76

9.2 Software Version Control 78

9.3 Galileo Firmware-Baseband Interface 78

9.4 Galileo Signal Processing Module 78

9.5 Galileo E1-C Secondary Code Tracking 81

9.6 Galileo Measurement Processing Module 82

9.7 Satellite selection 83

9.8 Satellite Visibility Prediction for Galileo Channels 84

9.9 Satellite Visibility Prediction Results for GPS and Galileo Channels 86

9.10 EPH and ALM data base update 88

9.11 Integrated Position Calculation 88

9.12 Report update- through UART PC interface 89

9.13 Debugging Methodologies 89

9.14 MATLAB Scripts for Pseudorange and Carrier Phase Analysis 90

10 TEST RESULT 91

11 GNSS RECEIVER OPERATION AND ENVIROMENTAL TESTING AT DLR 106

11.1 Outline 106

11.2 Operational Issues 106

11.3 Flight model tests 107

11.4 Our tests 108

11.5 Concluding remarks 108

12 NAMURU RECEIVER COMMAND AND REPORTS 109

12.1 Serial Port Communications 109

12.2 Reports 109

12.3 NMEA Standard 109

12.4 Description of the commands 112

13 FIELD UPGRADE OF THE NAMURU RECEIVER FIRMWARE AND LOGIC 161

13.1 Bootloader Requirements 161

13.2 Bootloader Operation 161

13.3 Reprogramming the GPS Firmware 174

14 CONCLUSION AND FUTURE WORK 176

15 REFERENCES 179

1 Executive Summary

This document describes the design, development, testing and performance analysis of two satellite navigation receivers for space missions planned by Australia. The space-capable satellite navigation receiver development is part of the Australian Space Research Program funded Garada "SAR Formation Flying" project and combined with the Cubesat project "Biarri" funded by the Defence Science and Technology Organization (DSTO). Work package 5 in the Garada project proposed the development of a dual-system, dual-frequency Global Navigation Satellite System (GNSS) and a modified Namuru receiver for the Bluesat satellite. The latter was combined with the development of a single-frequency Global Positioning System (GPS) receiver was part of the Biarri project.

The scenario of the in-sight Biarri project requirements being a subset of the requirements of the long-term Garada GNSS receiver provided a good cohesive flow for the project process without which several successful outcomes would not have occurred. However, the single-frequency receiver, which is a far more sophisticated receiver than envisaged in the proposal, went well into the field-integration testing phase and required several hardware revisions, which impacted the multi-frequency receiver development timetable somewhat. More significant however, were the numerous hardware revisions of the multi-frequency board, which proved a very difficult design problem. Nevertheless, the entire development process added a great value to the UNSW Namuru family of GNSS receiver platforms which have been sought for various research projects around the world. Three new Namuru GNSS receiver platforms and several unique GNSS receiver attributes were generated in this process that will act as baselines for several space related projects in the near future.

The single-frequency Garada/Biarri receiver was tested almost all the way through to acceptance on an international space program. Very little remains to be done on that receiver at the end of the ASRP period. The multi-frequency receiver was able to demonstrate functionality both as a multi-system receiver, and a multi-frequency receiver, but not as an entirely integrated L1/ E1/ L5/ E5 receiver. That work will be completed with funds other than those provided by the ASRP and is expected to be complete by the end of 2013.

This document is organised as follows. Section 2 provides a brief overview of the project and the GNSS receivers in context. Section 3 describes the design and development methodology followed in this project followed by test methodology in Section 4. Section 0 provides detailed information about the hardware platforms and section 6 briefly provides the design strategies followed to ensure a space-certifiable hardware. Section 7 explains the design of Field Programmable Gate Array (FPGA) based baseband signal processing. Section 8 describes the development of a proof-of-concept oriented Matlab based receiver and section 9 details the firmware development and debug activities. Section 10 provides the test results followed by DLR space qualification test facility visit summary in section 11 and section 0 describes the Namuru receiver external world interface protocols. Section 13 describes a unique feature developed for the Namuru receiver, the field upgrade of the logic design and the firmware. Section 0 concludes the report and briefly describes the envisaged roadmap for the future followed by references in section 0.

2 Introduction and GNSS Receiver Overview

In this research, GNSS receivers using the new V3 Namuru platform have been developed for space operations. Although not fully space qualified in terms of radiation tolerance, they are designed to recover gracefully using redundancy to maintain operational status in the harsh environment of space. The "SAR Formation Flying" proposal promised two GNSS receivers: a dual-frequency, dual system receiver, and an upgraded Namuru for use on Bluesat. As the Garada project progressed, three platforms were developed in pursuit of that first goal, Namuru V3.1, V3.3 and V3.4, the latter two models both meeting the dual-frequency, dual-system criterion. At the beginning of the project, the Defence Science and Technology Organisation (DSTO) approached the team to develop a Cubesat receiver for the Biarri project. Therefore the effort in the original proposal dedicated to upgrading the earlier Namuru V2.4 was combined with the resources provided by DSTO to produce the Namuru V3.2. In addition, the previous generation receiver, the Namuru V2.4 and V2.5, were used extensively for initial experiments.

The operational failure risks associated with single event upsets (SEU) in volatile memory are reduced through the use of careful design and selection of components. Factors affecting power consumption are carefully minimised to avoid overheating at zero atmosphere. The receivers use custom designed printed circuits, with custom base-band logic in FPGA's and supporting application firmware. The receiver design uses non rad hard components throughout with some important features that have been added to improve reliability and provide fault tolerance.

Figure 2-1 shows the typical architecture of a GNSS receiver. The received signal is filtered, down-converted and passed through an Analogue-to-Digital-Converter (ADC) to obtain the Intermediate Frequency (IF) samples. The baseband signal processing (widely known as the correlator) is implemented in a programmable digital chip called Field Programmable Gate Array (FPGA). The correlator output is further processed, analysed, satellite navigation data is decoded and the user position, velocity and time are estimated using an embedded processor. In the case of Garada receiver, the custom-built multi-band RF front-end is configurable and the embedded processor lives within the FPGA fabric. The embedded processor on the other side talks to the mission computer for any system-level interactions.

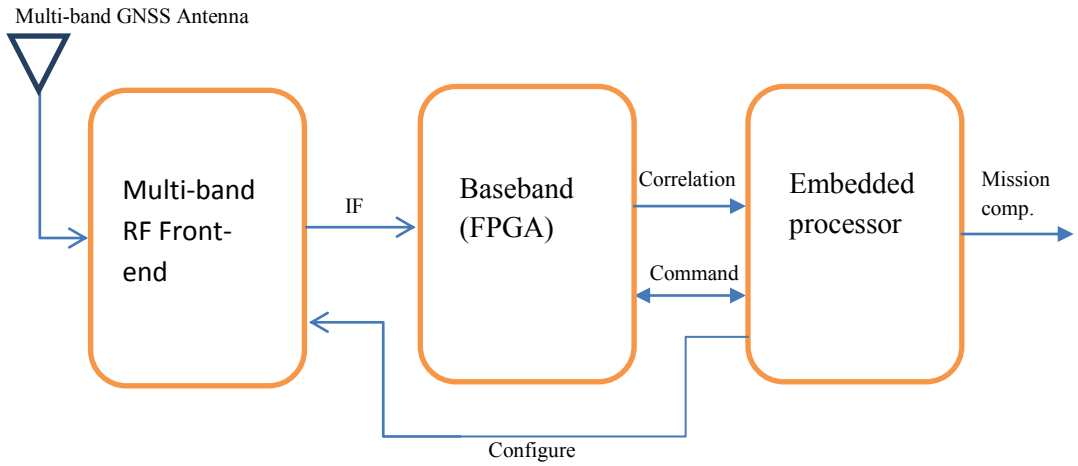


Figure 2-1 Block diagram of a typical multi-GNSS receiver

3 Design and Development Methodology

Upon the Garada project initiation during the project requirements and planning stage, the team evaluated different design methodologies that can lead to the final multi-signal multi-frequency prototype GNSS receiver. Since there are several stages in the receiver development life-cycle the goal was to prove and to cross reference (to a logically possible extent) the decisions taken in the hardware and firmware design and architecture, by a relatively well known and risk free development environment. The important development methodologies established towards this are shown in Table 3-1.

Signal source	**RF front-end**	**Baseband**	**Tracking and Navigation**	**Post processing**	**Primary purpose***
Matlab simulated	-NA-	Matlab	Matlab	RTKLib Pro	PoC
Spirent simulator	COTS data grabber (Nordnav / GNURadio)				PoC
	Namuru RF-FE V2.4	Legacy Digital baseband	Aquarius firmware		Dev / Dbg
	Namuru RF-FE V3.x	Digital Baseband	Modified Aquarius		Dev / Dbg / PE
Real signal	COTS data grabber (Nordnav / GNURadio)	Matlab	Matlab		PoC
	Namuru RF-FE V2.4	Legacy Digital baseband	Aquarius firmware		Dev / Dbg
	Namuru RF-FE	Digital Baseband	Modified Aquarius		Final prototype

Table 3-1 Development methodologies used in the project

**PoC = Proof of Concept, Dev = Development, Dbg = Debugging, PE = Testing and Performance Evaluation*

The different stages identified were (1) the signal source, (2) the RF-front end, (3) the baseband, (4) the tracking and navigation and (5) the post processing.

Apart from the option of getting the real signal through an Antenna which is the ultimate situation, the signal could be simulated (with limited variable parameters). Another method is to use a signal simulator which gives out the signal via an RF cable to feed directly into a board (bypassing the Antenna). Spirent GNSS signals simulator is very powerful and is the de-facto standard in the GNSS industry especially for receiver debugging and testing. Spirent simulator tools allow exhaustive

parameter configuration and scenario generations for various kinds of applications including the space.

If the signal is generated via Matlab software, the signal will already be in the sampled and digitized format and hence no RF front-end is required. In the case of signal obtained at RF (either simulated via Spirent or the real signal via an Antenna) the signal needs to be down-converted to IF before any digital processing can happen. In the final setup this is done by the Namuru RF front-end but instead of waiting for the final board, other options thought were to use COTS signal sampler. Two such signal samplers used were the NordNav and the USRP. The NordNav could get L1/E1 band signals and the USRP is a Software Defined Radio where the centre frequency and bandwidth is configurable (i.e. can work at L1/E1 or L5/E5a band). Signal collected from these signal samplers were fed to the Matlab processing software and were extremely helpful in proving several techniques and algorithms that eventually went into the final firmware on the embedded processor.

In the case of Matlab based processing, the tracking and navigation was also done in Matlab to generate the final RINEX file to feed to the RTKLib Pro. RTKLib Pro is a post-processing software developed by UNSW based on the open source RTKLib version. RTKLib Pro can handle multi-GNSS signals and RINEX 3.0 format input files. Some of the important outputs of the RTKLib Pro are the carrier-phase and pseudorange position solutions, ambiguity resolution statistics for GPS Galileo QZSS multi-frequency and multi-signal GNSS data.

The aim of the Matlab based receiver development is to allow the feasibility of using a unified platform for the processing of multi-GNSS signals for integrated positioning. The multi-software receiver platform discussed in this section is based on an open-source GPS receiver architecture. The user GUI interface is upgraded to allow the activation of multi-GNSS processing. Modifications for the GNSS open service signals at L1 need to deal with the unique features of each signal which include the PRN code modulation scheme, code period, data decoding method and system time difference. These differences lead to the modifications in the modules of acquisition, tracking and message extraction and position estimation.

Prior to the Garada Project, the School of Surveying and Geospatial Engineering (SAGE) has already established a solid foundation for real-time GPS receiver development. The flagship receiver produced was the dual front-end single-frequency GPS receiver, the Namuru V2.4 that was successfully flown on a sounding rocket experiment by the German Aerospace Agency (DLR). During the same time, a hardware-independent firmware known as the Aquarius was developed by Dr. Eamonn P Glennon based on the Namuru V2.x series of hardware. Aquarius was a full-fledged single frequency GPS L1 C/A firmware containing a proprietary RTOS, a full suite of signal acquisition and satellite tracking algorithms, and a Kalman-filter-based positioning engine.

After successfully acquiring funding for the Biarri project to develop centimeter-accurate carrier phase positioning capabilities on a different hardware (i.e. the Namuru V3.2) for in-orbit satellite scenarios, the Aquarius firmware was further enhanced to provide carrier phase measurement outputs and allow compatibility with the Namuru V3.2 hardware which is based on an ARM Cortex M3 CPU. The Namuru 2.x series of hardware used Altera NIOS II soft-core CPU. Furthermore, the GPS acquisition and tracking algorithms together with the position domain filters were optimised for in-orbit LEO satellite scenarios that have high signal dynamics. That is, the signals have very significant velocity and acceleration components that typical land-based receiver cannot tolerate. The in-orbit

acquisition process is enhanced by utilising the Almanac to provide Doppler frequency aiding. In addition, the order of the carrier tracking loop, code tracking loop and position domain filter are increased to cope with the increased signal dynamics.

To achieve the aims of the Garada project, the matured Aquarius firmware inherited to be used as a template to develop a receiver that supports the European Galileo E1 and E5a signals and the US GPS L5 signals. For firmware development purposes, the Aquarius is first expanded to include the Galileo E1 signals on the Namuru V2.4, and then ported onto the Namuru V3.3 to be further developed into a dual-system dual-frequency GNSS receiver.

GPS receiver firmware is very complex, and with this complexity comes the probability to introducing bugs into the code. While a thorough testing regime can assist in the elimination of such bugs, it is almost impossible to eliminate every bug. This introduces a requirement to allow upgrade of the firmware. In the past, UNSW Namuru receivers have been used for research and development, with the consequence that any required firmware changes were applied using JTAG and the C compiler toolset. However, programming receivers in this way is not possible for the receivers destined to operate in space and an alternative approach is therefore required. Development of the Biarri firmware upgrade feature was a solution to this problem.

4 Test methodology

Test methodology includes a detailed test plan and procedure for Namuru receivers. There are required tests as well as optional tests. In this report only the require tests are conducted. Development of the testing tools also gets the priority before starting the testing. As a result an effort was given to update or developing in house software for testing.

4.1 Development of the Test tools

Three major test tools were developed. Followings are short description of these tools

- To do post-processing, RTKLib_Pro (i.e. version 2.4.1b) software, an enhanced version of RTKLib (Takasu and Yasuda, 2009), was developed by ACSER with the capability of processing GPS and Galileo observations. At the beginning of this project, there was no post processing software which can process GPS and Galileo signals. As a result, RTKLib_Pro is developed.

- To convert proprietary message to RINEX format (i.e. most commonly used in GNSS environment), a tool named OBS2RINEX (version 2) is developed.

- Matlab based scripts (i.e. TestScripts V_01) was also developed for identifying, debugging as well as quantifying position accuracy and precision. These scripts can also be used for identifying receiver's performance, such as time to first fix, time to lock etc.

4.2 Test Equipment

4.2.1 Hardware requirement

Hardware	Availability(Y/N)	Note
Spirent GS8800		
Namuru V3.2		
RF cable		
SMA connector		
Connector to connect RF cable to Spirent		
RS422 connector to PC		
Timer Counter cable		
Timer Counter		

4.2.2 Software requirement

Software	Version	Availability(Y/N)	Note
Aquarius firmware			Installed in Namuru board
PosApp			Installed in the PC where Spirent is connected
VPSE	0.938.4.846		Installed in the data collection PC.
TestCase	V_01		
Matlab	7.9.0 or above		
TestScripts	V_01		

4.2.3 TestScripts' Input file requirement:

	Files	Note	Availability(Y/N)
Case 1	base_case1_XX.nmea	1. Generated after test case 2. XX is the file identifier.	
	base_case1_truth_xx_.csv		
Case2	base_case2_XX.nmea	Same as above	
	base_case2_truth_xx_.csv		
Case 3	base_case3_XX.nmea	Same as above	
	base_case3_truth_xx_.csv		

4.3 Test cases for functional test

In order to evaluate Namuru receiver's navigation/positioning performance, a simple test setup is proposed where simulated GPS signals are to be used. These simulated signals can be generated using Spirent Multi-Channel GNSS Simulator as defined later in this report.

As these receivers are to be used in space, their testing on the ground using real signals may not be under similar conditions as expected during their actual operation in space. For instance, it is not feasible to emulate Doppler on the ground that can be expected while operating on-board a spacecraft. Also, terrestrial testing may include effects such as multipath and atmospheric errors (e.g. tropospheric errors) that are not likely to be experienced in space.

Use of simulated signals allows replication of real scenarios as closely as possible by simulating environmental conditions experienced by a spacecraft, like realistic Doppler.

Simulated GPS signals were also selected for performance evaluation as they allow test repeatability and thus validation of test results.

Following sections define the hardware, simulator and the receiver setup.

4.3.1 Test cases

For Namuru positioning performance evaluation error-free testing is required. However, there are test cases which could be used to evaluate the receiver's performance in dynamic scenarios and those are optional. Following test cases need to be considered:

Required:	
Case_1:	Error-free position testing along with PPS test
Optional:	
Case _2:	Testing in presence of ionospheric errors only along with PPS test.
Case _3:	1 meter Carrier phase circle test.
Case _4:	Stationary test.
Case_5:	PPS testing

4.3.2 Folder structure of test cases

- TestCase_V_01
  - Case1
    - Base
      - Numarufunctiontest
    - Rover
      - Numarufunctiontest
  - Case2
    - Base
      - Numarufunctiontest
    - Rover
      - Numarufunctiontest
  - Case3
    - Base
      - Numarufunctiontest
    - supportEXE
      - SetupVSPE
      - rtklib_2.4.1
  - Case4
    - Rover
      - Numarufunctiontest
  - Data

* *TestCase_V_01/Data* will be referred as *<data_folder> in following sections.*

4.3.3 Hardware Setup for case 1 to case 4:

The proposed test hardware setup is shown in Figure 4-1 Test setupFigure 4-1. The Spirent simulator generates the L1 C/A GPS signals which are fed to Namuru GPS receiver via RF cable for position computation. The simulation parameters are controlled by PosApp software on PC1 that defines the simulation scenarios including GPS constellation, signal strength, simulation duration, receiver antenna characteristics and trajectory of the spacecraft on which the Namuru is to be installed. Namuru receiver, after processing the received GPS signal, generates: Pseudorange measurements, Carrier phase measurements, Pseudorange based absolute position, Timing pulse-per-second (PPS). Namuru receiver provides Pseudorange measurements, Carrier phase measurements, Pseudorange based absolute position, Timing pulse-per-second (PPS). PPS signal is outputted into designated pin (please see Namuru V3.2 user guide).Connection from receiver to 'Timer counter' are necessary. Port 1 of the 'Timer counter' is connected to Namuru's PPS and ground pin. Port2 of the 'Timer counter' is connected to Spirent's PPS output port.

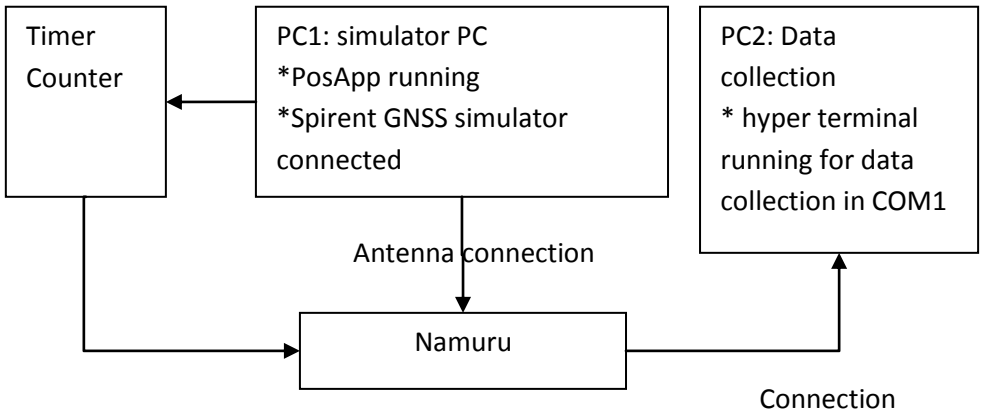


Figure 4-1 Test setup

Namuru supports standard NMEA messages for data output. Proprietary messages are also supported by the firmware for reporting information including acquisition assistance, receiver channel status, ephemeris, ionospheric corrections, etc. Using the Aquarius firmware, Namuru also supports special proprietary message providing data that can be used to generate RINEX files. This data allows performing higher accuracy positioning as compared to pseudorange based positioning.

These measurements are recorded by PC2, which connects to Namuru via RS422 interface. This data can then be used for position computation.

4.3.4 Hardware Setup for case 5: Relative PPS test between two Namuru boards

The proposed test hardware setup, for Relative PPS test between two receivers, is shown in Figure 4-2. Namuru outputted PPS signal into designated pin (please see Namuru V3.2 user guide).Connection from receivers to 'Timer counter' are necessary. Port 1 of the 'Timer

counter' is connected to 1st Namuru's PPS and ground pin. Port2 of the 'Timer counter' is connected to 2nd Namuru's PPS and ground pin.

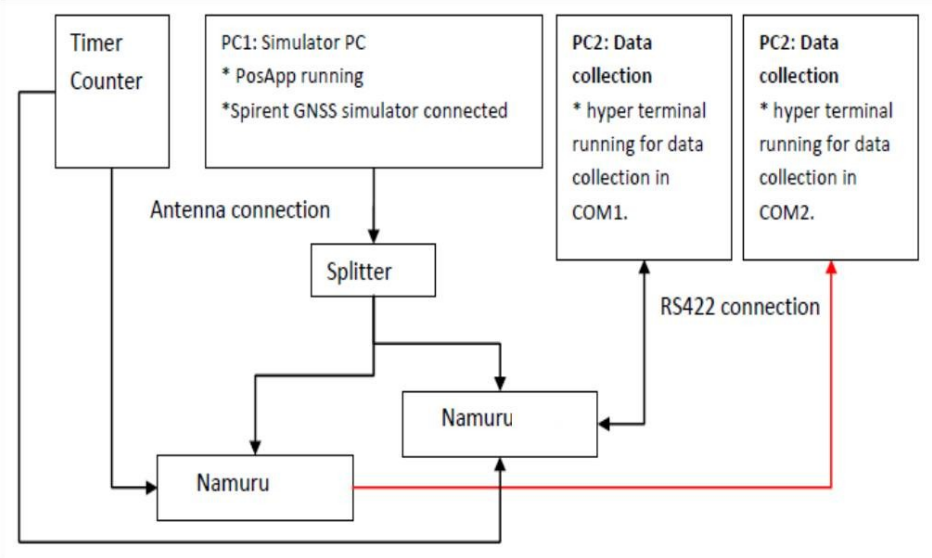


Figure 4-2: PPS output test.

4.4 Test procedure: CASE_1

4.4.1 Introduction

This is an error-free testing which means there is no ionospheric or ephemeris error is introduced.

4.4.2 Base spacecraft test

Step	Step description	Pass/Fail
	Spirent Setup:	
1	Open PosApp.	
2	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: *a. Go to File menu and click on Load.* *b. Browse to the folder named NamuruTest/case_1/Base/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_base.scn'.* *c. Click signal power button on the toolbar and adjust the signal power to 20 for all the channels.* *d. For truth data logging, select View->Data logging Control Window. Click load. Browse to 'datalog.qll.'*	
3	Connect Spirent's PPS output to port2 of Timer counter.	

4	Press 'Start' button from the toolbar.	
5	***Namuru Setup:***	
6	Connect the Namuru to a power source between 6-18V DC via the 6 pin connector.	
7	Connect the Namuru card to an RS422-USB adaptor via the 20 pin connector.	
8	Connect the RS422-USB adaptor to PC2.	
9	Connect Namuru's PPS and ground pin to port1 of Timer counter.	
10	Power up Namuru.	
11	Open HyperTerminal program and set the properties as 115200-8-N-1-N and connect to the system assigned port for RS422-USB adaptor. Start data logging on HyperTerminal by using the Capture Text option on Transfer menu. Name the data log file as base_caseN_XX.nmea, where XX is the file identifier and N is case number. Eg: For case 1 and file identifier 1 naming convention should be: base_case1_01.nmea	
12	Send the following command to the receiver for clearing up its non-volatile memory: $PNSWC,CLN To test if the non volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
13	Restart the Namuru board by switching off from power source and switch on.	
14	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uITFeaFc,4	
15	By default, Namuru provides the following messages but if nothing shows in HyperTerminal in 5 minutes then send following commands to Namuru for making sure that the desired data is output by the	

receive:

$GPGPQ,GGA,E
$GPGPQ,OBS,E
$GPGPQ,GRS,E
$GPGPQ,GSA,E
$GPGPQ,GSV,E
$GPGPQ,VTG,E
$GPGPQ,ZDA,E
$GPGPQ,XYZ,E
$GPGPQ,ION
$GPGPQ,UTC
$GPGPQ,PPS,E

Default message will be similar to:

After start of receiver

~~$PNSWR,FWV,BIARRI_MAZ_2012_09_04,1.2,1.06,20120904_155220*1C
~~$PNSWR,ION,663,52,05,02,FF,FF,27,04,FF,FA*19
~~$PNSWR,AAM,0,0,05,,,,,+2971.2,+40000,,000,00,0,0*39
~~$PNSWR,AAM,0,1,06,,,,,+2971.2,+40000,,000,00,0,0*3B
~~$PNSWR,AAM,0,2,08,,,,,+2971.2,+40000,,000,00,0,0*36
~~$PNSWR,AAM,0,3,09,,,,,+2971.2,+40000,,000,00,0,0*36
~~$PNSWR,AAM,0,4,12,,,,,+2971.2,+40000,,000,00,0,0*3B
~~$PNSWR,AAM,0,5,16,,,,,+2971.2,+40000,,000,00,0,0*3E
~~$PNSWR,AAM,0,6,25,,,,,+2971.2,+40000,,000,00,0,0*3D
~~$PNSWR,AAM,0,7,27,,,,,+2971.2,+40000,,000,00,0,0*3E
~~$PNSWR,AAM,0,8,28,,,,,+2971.2,+40000,,000,00,0,0*3E
~~$PNSWR,AAM,0,9,30,,,,,+2971.2,+40000,,000,00,0,0*36
~~$PNSWR,AAM,0,10,31,,,,,+2971.2,+40000,,000,00,0,0*0F
~~$PNSWR,AAM,2,0,193,,,,,+2971.2,+40000,,000,00,0,0*05
~~$PNSWR,UTC,000000,00000000,0,52,663,61,2,0*27

In case of cold start, within 7 minutes of receiver restarts following message will show in the HyperTerminal screen:

~~$PNSWR,OBS,2,1,1,407291.088000000,1650,0,0,0,0,0.142857*0E
~~$PNSWR,OBS,2,2,5,0,01b,407291.020226093,20318106.109,-373.703,0.000,00,000,50,337,33.5*6B
~~$GPZDA,170811.097,25,08,2011,+00,00*70
~~$GPGGA,000000.000000,0000.0005,S,00000.0048,W,0,00,,-

	0.8,M,0,M,,*4E ~~$GPGSA,A,3,,,,,,,,,,,,,*1C ~~$GPGSV,3,1,12,05,,,50,06,,,16,08,,,28,09,,,24*74 ~~$GPGSV,3,2,12,12,,,26,16,,,26,25,,,26,27,,,24*7F ~~$GPGSV,3,3,12,28,,,22,30,,,27,31,,,26,193,,,22*4B ~~$GPVTG,090.0,T,,M,00.0,N,00.0,K*69 ~~$PNSWR,XYZ,0,1650,0.000,0.0,0.0,0.0,0.000,0.000,0.000,0.0,0.000,*23 In case of cold start, within 32 minutes of receiver restarts following message will show in the HyperTerminal screen where it can be seen that Namuru has successfully produces first position fix: ~~$GPZDA,171748.112,25,08,2011,+00,00*7E ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,1,04,1.9,61298 2.6,M,0,M,,*77 ~~$GPGSA,A,3,5,6,8,21,,,,,,,,,4.4,1.9,3.9*08 ~~$GPGSV,3,1,12,05,,,50,06,,,50,08,,,50,04,,,24*74 ~~$GPGSV,3,2,12,03,,,26,19,,,25,02,,,20,07,,,25*73 ~~$GPGSV,3,3,12,21,,,50,17,,,20,11,,,27,193,,,24*40 ~~$GPGRS,171748.10,0,+0.0,+0.0,+0.0,+0.0,,,,,,,,*6E ~~$GPVTG,339.7,T,,M,14823.3,N,27452.7,K*60 ~~$PNSWR,XYZ,1,1650,407868.096,-590418.0,2875400.2,6325608.6,3901.182,-5821.454,3007.306,2487035.6,569.918,*14 Note: Position fix can be seen in GPGGA message 7th field(i.e. bold in following example) ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,**1**,04,1.9,61298 2.6,M,0,M,,*77	
16	If Namuru is not outputting messages, then send following command to update receiver clock as this time would be same as the scenario file. $GPZDA,170000.000,25,08,2011	
17	***Timer Counter Setup:***	

18	On the Timer counter port1 setup 1.5V trigger, DC coupling, 1.5V level set up, auto-level off	
19	On the Timer counter port2 setup 2.5V trigger, DC coupling, 2.5V level set up, auto-level off	
20	On the Timer counter select time difference and select '1-2'	
21	If no data showing in the screen then please select '2-1'	
22	Start data logging on Timer counter by select 'Data log' and provide the data log filename.	
23	***Final setup:***	
24	When simulation is finished, Spirent will open a dialog box with a default name for truth. Please change the name to <data folder>base_case1_truth_xx_.csv where XX is the file identifier.	
25	Stop data logging on HyperTerminal by using the Capture Text option on Transfer menu.	
26	Stop the data logging on Timer counter once the simulation is finished.	

.

4.4.3 Rover spacecraft test

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: a. Go to File menu and click on Load. b. Browse to the folder named NamuruTest/case_1/Rover/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_rover.scn'. c. Click signal power button on the toolbar and adjust the signal power to 20 for all the channels. d. For truth data logging, select View->Data logging Control Window. Click load. Browse to 'datalog.qll.'	

4	Connect Spirent's PPS output to port2 of Timer counter.	
5	Press 'Start' button from the toolbar.	
6	***Namuru Setup:***	
7	Please follow section 4.4.2 steps 6 to step 10.	
8	Open HyperTerminal program and set the properties as 115200-8-N-1-N and connect to the system assigned port for RS422-USB adaptor. Start data logging on HyperTerminal by using the Capture Text option on Transfer menu. Name the data log file as rover_caseN_XX.nmea, where XX is the file identifier and N is case number. Eg: For case 1 and file identifier 1 naming convention should be: rover_case1_01.nmea	
9	Please follow section 4.4.2 steps 12 to step 22.	
10	When simulation is finished, Spirent will open a dialog box with a default name for truth. Please change the name to rover_case1_truth_xx_.csv where XX is the file identifier.	
11	Please follow section 4.4.2 steps 25 and step 26.	

4.5 Test procedure: CASE_2

4.5.1 Introduction

In this case, effects of ionospheric errors on the positioning solution are studied. Following are the steps for testing in presence of ionospheric errors.

4.5.2 Base spacecraft test

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: *a. Go to File menu and click on Load.* *b. Browse to the folder named NamuruTest/case_2/Base/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_base.scn'.* *c. Click signal power button on the toolbar and adjust the*	

	signal power to 20 for all the channels. *d. For truth data logging, select 'Data log' icon from the top panel and browse to 'datalog.qll'*	
4	Connect Spirent's PPS output to port2 of Timer counter.	
5	Press 'Start' button from the toolbar.	
6	***Namuru Setup:***	
7	Please follow section 4.4.2 steps 6 to step 13.	
8	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. *$PNSWR,CFG,5,3,uiTF,4*	
9	Please follow section 4.4.2 steps 15 to step 22.	
10	When simulation is finish, Spirent will open a dialog box with a default name for truth. Please change the name to base_case2_truth_xx_.csv where XX is the file identifier.	
11	Stop data logging on HyperTerminal by using the Capture Text option on Transfer menu.	
12	Stop the data logging on Timer counter once the simulation is finished.	

.

4.5.3 Rover spacecraft test

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: *a. Go to File menu and click on Load.* *b. Browse to the folder named NamuruTest/case_2/Rover/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_rover.scn'. Details of the scenario file and there setup procedure is explained in Appendix B.* *c. Click signal power button on the toolbar and adjust the signal power to 20 for all the channels.* *d. For truth data logging, select 'Data log' icon from the*	

	top panel and browse to 'datalog.qll'	
4	Press 'Start' button from the toolbar.	
5	***Namuru Setup:***	
6	Repeat section 4.4.2 step 7 to 9.	
7	When simulation is finish, Spirent will open a dialog box with a default name for truth. Please change the name to rover_case2_truth_xx_.csv where XX is the file identifier.	
8	Repeat section 4.4.2 step 11 to 12.	

4.6 Test procedure: CASE_3

4.6.1 Introduction

In this case, 5 meter circle is simulated which will show the carrier phase performance in ground. This is an alternative use of carrier phase test in orbit situation. Initially a virtual COM port is created using VSPE and then rtcm message is outputted to COM port. This message is stored into a file which later used as a base station observation for carrier phase [positioning.

Following are the steps for setting up the scenario and perform testing.

4.6.2 Test setup

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Install VSPE provided in TestCase_V_01t/case_3/supportEXE/VPSE/	
3	Open VSPE.exe	
4	Run 'com8_virtual_115200.vspe' using File-> open-> com8_virtual_115200.vspe'	
5	Open hyper terminal and connect to COM port 8, set the properties as 115200-8-N-1-N and connect	
6	Start data logging on HyperTerminal by using the Capture Text option on Transfer menu. Name the data log file as rover_case5_XX.rtcm, where XX is the file identifier.	
7	Open PosApp.	
8	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: a. *Go to File menu and click on Load.* b. *Browse to the folder named*	

	NamuruTest/case_5/Rover/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest.scn'. *c. Click signal power button on the toolbar and adjust the signal power to 20 for all the channels.* *d. For truth data logging, select 'Data log' icon from the top panel and browse to 'datalog.qll'* ***e.** Open Menu->RS232 setting. Setup comport 8 and RTCM output from pull down menu.*	
9	Press 'Start' button from the toolbar.	
10	***Namuru Setup:***	
11	Please follow section 4.4.2 step 6 to step 8.	
12	Please follow section 4.4.2 step 10 to step 13.	
13	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uiTF,1	
14	Please follow section 4.4.2 step 15 to step 16.	
15	When simulation is finish, Spirent will open a dialog box with a default name for truth. Please change the name to base_case3_truth_xx_.csv where XX is the file identifier.	

4.7 Test procedure: CASE_4

4.7.1 Introduction

In this case, a stationary scenario is simulated which will elaborate stand alone accuracy and timing. Following are the steps for setting up the scenario and perform testing.

4.7.2 Stationary test

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: a. Go to File menu and click on Load. b. Browse to the folder named	

	NamuruTest/case_6/Rover/numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_base.scn'. c. Click signal power button on the toolbar and adjust the signal power to 20 for all the channels. d. For truth data logging, select 'Data log' icon from the top panel and browse to 'datalog.qll'	
4	Press 'Start' button from the toolbar.	
5	**Namuru Setup:**	
6	Please follow section 4.4.2 step 6 to step 8.	
7	Please follow section 4.4.2 step 10 to step 13.	
8	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uiTF,1	
9	Please follow section 4.4.2 step 15 to step 16.	
10	When simulation is finish, Spirent will open a dialog box with a default name for truth. Please change the name to base_case4_truth_xx_.csv where XX is the file identifier.	

4.8 Position Computation

4.8.1 Absolute Position

4.8.1.1 Test procedure for CASE_1

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named "TestScripts_v_01"[i.e. provided by ASCER,UNSW]	
3	Provide the file names for following variables according to the name. rover.nmea, rover.truth Note: * rover have two Matlab cell structures that holds the file names of	

	the NMEA file and truth file. Eg: rover.nmea='H:\Data\ rover_case1_01.nmea'; rover.truth='H:\Data\ rover_case1_truth_01.csv';	
4	Give following command absPositionTest(rover) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.8.1.2 Test procedure for CASE_2

Step description	Pass/Fail
Please follow section 4.8.1.1 step 1 to step 4 for those files generated during case2 (i.e. rover_case2_01.nmea and rover_case2_truth_01.csv).	

4.8.1.3 Test procedure for CASE_3

Step description	Pass/Fail
Please follow section 4.8.1.1 step 1 to step 4 for those files generated during case3 (i.e. rover_case3_01.nmea and rover_case3_truth_01.csv).	

4.8.1.4 Test procedure for CASE_4

Step description	Pass/Fail
Please follow section 4.8.1.1 step 1 to step 4 for those files generated during case4 (i.e. rover_case4_01.nmea and rover_case2_truth_04.csv).	

4.8.2 Absolute position acceptance

Any test described in the sections from 4.8.1 should have 0±10m. In such cases, absolute position test can be marked as pass. Otherwise should be marked as failed.

Example:

A sample output of section 6.1.1.6 is follows:

MEAN : -10.9058 -8.84245 70.2516
Std(1 sigma) : +/-4.785 10.359 15.1893

This result can be interpreted as position component X, Y and Z have mean accuracy of -11, -9m, 70m with standard deviation of ±4.8m, ±10.4m, ±15.2m, respectively. In this case absolute position test failed. On the other hand, following sample output shows relative position test passed where position component X, Y and Z have mean accuracy of -1, -1m, 1m with standard deviation of ±2.6m, ±4.9m, ±5.8m, respectively. According to FPS, acceptability of absolute position test depends on the zero mean from the truth and standard deviation should be less than 10cm. However, zero mean is not always achievable during test.

MEAN	:	-0.9058	-0.84245	1.2516
Std(1 sigma)	:	+/-2.5753	4.8977	5.8173

4.8.3 Relative position for Namuru V2.4 and V3.2

4.8.3.1 Test procedure for CASE_1

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
3	Provide the file names for following variables according to the name. base.nmea, base.truth, rover.nmea, rover.truth Note: * base and rover are two Matlab cell structures that holds the file names of the nmea file and truthfile. Ex: base.nmea='H:\Data\ base_case1_01.nmea'; base.truth='H:\Data\ base_case1_truth_01.csv'; rover.nmea='H:\Data\ rover_case1_01.nmea'; rover.truth='H:\Data\ rover_case1_truth_01.csv';	
4	Give following command relPositionTest(base,rover) *This program will generate a generate graphs as well as provides the statistics from the truth. At first this program converts nmea file to rinex file and then it uses ’RTKLIB’ program to generate relative position.	
5	Observe the mean and standard deviation column of the accuracy output. **Standard deviation value should be below 10cm** * Appendix F shows a sample of graphs and statistics.	

4.8.3.2 Test procedure for CASE_2

Step description	Pass/Fail
Please follow section 4.8.3.1 step 1 to step 5 for those files generated during case2.	

4.8.3.3 Test procedure for CASE_3

Step	Step description	Pass/Fail
1	Open rtklib_2.4.1/bin/rtkconv.exe	
2	From “RTCM,RCV RAW or RINEX OBS ?” input option select the rover_case3_XX.rtcm file which is generated in section 4.6.	
3	Press ‘Convert’. It will generate “rover_case5_XX.obs”.	
4	Open TestCase_V_01\case5\supportEXE/OBS2RINEX_MFC.exe	
5	From “Path and Name of the Input File” input option browse to “rover_case3_XX.obs” which will generated “Case3.N” & “Case3.O” file.	
6	“Station Name” input option type “case3”	
7	Click “Go” button and wait until “Congratulation” message is popped up.	
8	Open rtklib_2.4.1/bin/ rtkpost_mkl.exe	
9	Click “Option” button. From “Setting1” tab ”Positioning Mode” pull down menu select “Kinematic”. ”Ionosphere Correction” pull down menu select “OFF”. ”Troposphere Correction” pull down menu select “OFF”.	
10	Click “OK” button.	
11	From “RINEX OBS: Rover” input option browse to ”case3.O”.	
12	From “RINEX OBS: Base Station” input option browse to “rover_case3_XX.obs”	
13	From “RINEX *NAV/CLK,SP3,IONEX or SBS/EMS” input option browse to ”case3.N” which is generated in section 4.6.	
14	Click “ Execute” button. Wait until execution is finished.	
15	*Click Plot button to visualize the result*	

4.8.4 Relative position acceptance

Any test described in the sections from 4.8.3 should have around 0±10cm. In such cases, relative position test can be marked as pass. Otherwise relative position test should be marked as failed.

Example:

A sample output is follows:

MEAN : -1.19543 -3.95915 -11.4199
Std(1 sigma) : +/-7.49242 10.541 33.2504

This result can be interpreted as position component X, Y and Z have mean accuracy of -1.2m, -4m, 11m with standard deviation of ±7.5m, ±10.5m, ±33m, respectively. In this case relative position test failed. On the other hand, following sample output shows relative position test passed where position component X, Y and Z have mean accuracy of 0.002m, 0.005m, 0.002m with standard deviation of ±0.03m, ±0.02m, ±0.04m, respectively. According to FPS, acceptability of relative position test depends on the zero mean and standard deviation should be less than 10cm.

MEAN : 0.0022 0.0045 0.0021
Std(1 sigma) : +/-0.0312 0 .0212 0.0400

4.9 Relative position for Namuru V3.3

4.9.1 Test procedure for CASE_1

Step	Step description	Pass/Fail
1	Open command prompt from windows start->Accessories	
2	Navigate to the directory using 'cd' command where obs2rinex.exe is stored. eg: cd H:\Data	
3	Issue following command. obs2rinex -ibase_case1_01.nmea -sbase_case1_01_ *Note: this command will generate two rinex file. 1. base_case1_01_XXX.11O 2. base_case1_01_XXX.11N	
4	Issue following command. obs2rinex -irover_case1_01.nmea -srover_case1_01_ *Note: this command will generate two rinex file. 1. rover_case1_01_XXX.11O 2. rover_case1_01_XXX.11N	
5	Please follow the section 'Data Processing with RTKLIB' from Manual_RTKLIBPro_spacecraft relative positioning document to	

	generate relative positions files using rover_case1_01_XXX.11O, base_case1_01_XXX.11O and rover_case1_01_XXX.11N. ***Note: please put the output file name as 'Relative_case1_01.pos'**	
6	Open Matlab program.	
7	Browse to the folder named "TestScripts_v_01"[i.e. provided by ASCER,UNSW]	
8	Provide the file names for following variables according to the name. relative.pos, truth.base, truth.rover Note: * base and rover are two Matlab cell structures that holds the file names of the nmea file and truthfile. Ex: relative.pos='H:\Data\ Relative_case1_01.pos '; truth. base='H:\Data\ base_case1_truth_01.csv'; truth. rover='H:\Data\ rover_case1_truth_01.csv';	
9	Give following command relPositionTest_garada(truth, relative) *This program will generate a generate graphs as well as provides the statistics from the truth. At first this program converts nmea file to rinex file and then it uses 'RTKLIBPro' program to generate relative position.	
10	Observe the mean and standard deviation column of the accuracy output. **Standard deviation value should be below 3cm** * Appendix F shows a sample of graphs and statistics.	

4.9.2 Test procedure for CASE_2

Step description	Pass/Fail
Please follow section 4.9.1 step 1 to step 10 for those files generated during case2.	

4.9.3 Test procedure for CASE_3

Step description	Pass/Fail
Please follow section 4.8.3 step 1 to step 7 for those files generated during case3.	
Open rtklib_Pro/bin/ RtkLib_Pro.exe	
Please follow section 4.8.3 step 9 to step 15 for those files generated during case3.	

4.9.4 Relative position acceptance

Test described in the sections from 4.9.1 to 4.9.23 should have around 0±3cm. In such cases, relative position test can be marked as pass. Otherwise relative position test should be marked as failed.

Example:

A sample output is follows:

MEAN : -1.19543 -3.95915 -11.4199
Std(1 sigma) : +/-7.49242 10.541 33.2504

This result can be interpreted as position component X, Y and Z have mean accuracy of -1.2m, -4m, 11m with standard deviation of ±7.5m, ±10.5m, ±33m, respectively. In this case relative position test failed. On the other hand, following sample output shows relative position test passed where position component X, Y and Z have mean accuracy of 0.002m, 0.005m, 0.002m with standard deviation of ±0.03m, ±0.02m, ±0.04m, respectively. According to FPS, acceptability of relative position test depends on the zero mean and standard deviation should be less than 3cm.

MEAN : 0.0022 0.0045 0.0021
Std(1 sigma) : +/-0.0212 0 .0212 0.0290

4.10 Test procedure: CASE_5 Relative PPS test

4.10.1 Test procedure

Step	Step description	Pass/Fail
1	Follow section 4.4.2 for 1st Namuru.	
2	Follow section 4.4.2 for 2nd Namuru.	
3	Connect 1st Namuru's PPS and ground pin to port1 of Timer counter.	
4	Connect 2nd Namuru's PPS and ground pin to port2 of Timer counter.	
5	On the Timer counter port1 setup 1.5V trigger, DC coupling, 1.5V level set up, auto-level off	
6	On the Timer counter port2 setup 1.5V trigger, DC coupling, 1.5V level set up, auto-level off	
7	On the Timer counter select time difference and select '1-2'	
8	If no data showing in the screen then please select '2-1'	
9	Start data logging on Timer counter by select 'Data log' and provide the data log filename.	
10	Stop the data logging once the simulation is finished.	
11	Stop the data logging on Timer counter once the simulation is finished.	

4.10.2 Acceptance Test data

Step description	Pass/Fail
Observe the mean and standard deviation collected data from Timer counter. ***Mean value should be below 20ns.***	

4.11 PPS test between Namuru and truth for Case 1 and Case 2

4.11.1 Acceptance Test data

Step description	Pass/Fail
Observe the mean and standard deviation collected data from Timer counter. **Mean value should be below 20ns.**	

4.12 Test methodology for identifying receiver's performance

The performance testing includes typical receiver performance characteristics such as time to first fix (TTFF) for cold, warm and hot start, time to lock (TTL) as well as acquisition and reacquisition performance and receiver sensitivity.

4.12.1 Test Equipment

4.12.1.1 Hardware requirement

Hardware	Availability(Y/N)	Note
Spirent GS8800		
Namuru V3.2		
RF cable		
SMA connector		
Connector to connect RF cable to Spirent		
RS422 connector to PC		

4.12.1.2 Software requirement

Software	Version	Availability(Y/N)	Note
Aquarius firmware			Installed in Namuru board
PosApp			Installed in the PC where Spirent is connected
TestCase	V_01		
Matlab	7.9.0 or above		
TestScripts	V_01		

4.12.1.3 TestScripts' Input file requirement:

	Files	Note	Availability(Y/N)
Case 1	TTFF_ColdStart_XX.nmea	1. Generated after test case 2. XX is the file identifier.	
Case2	TTFF_HotStart_XX.nmea	Same as above	
Case 3	TTFF_WarmStart_XX.nmea	Same as above	

4.12.1.4 Test case for performance test

In order to evaluate Namuru receiver's navigation/positioning performance, a simple test setup is proposed where simulated GPS signals are to be used. These simulated signals can be generated using Spirent Multi-Channel GNSS Simulator as defined later in this report.

As these receivers are to be used in space, their testing on the ground using real signals may not be under similar conditions as expected during their actual operation in space. For instance, it is not feasible to emulate Doppler on the ground that can be expected while operating on-board a spacecraft. Also, terrestrial testing may include effects such as multipath and atmospheric errors (e.g. tropospheric errors) that are not likely to be experienced in space.

Use of simulated signals allows replication of real scenarios as closely as possible by simulating environmental conditions experienced by a spacecraft, like realistic Doppler. Simulated GPS signals were also selected for performance evaluation as they allow test repeatability and thus validation of test results.

Following sections define the hardware, simulator and the receiver setup.

4.12.1.5 Test cases

For Namuru positioning performance evaluation error-free testing is required. However, there are test cases which could be used to evaluate the receiver's performance in dynamic scenarios and those are optional. Following test cases need to be considered:

Required:	
Case_1:	Cold start TTFF performance.
Case_2:	Warm start TTFF performance.
Case_3:	Hot start TTFF performance.
Case_4:	Cold start TTL performance.
Case_5:	Warm start TTL performance.
Case_6:	Hot start TTL performance.
Case_7:	Acquisition sensitivity.
Case_8:	Tracking sensitivity.

4.12.1.6 Hardware Setup for all cases:

This section is same as previous section 4.3.3.

4.13 Test procedure: Cold start test

4.13.1 Introduction

The Namuru spaceborne receiver supports a cold, hot or warm start in Low Earth Orbit (LEO) using the time returned from an in-built real-time clock (RTC) and data stored in non-volatile memory (NVM). This test is to identify the cold start TTFF and TTL.

4.13.2 Test procedure

Step	Step description	Pass/Fail
	Spirent Setup:	
1	Open PosApp.	
2	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: *a. Go to File menu and click on Load.* *b. Browse to the folder named NamuruTest/ numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_base.scn'. Details of the scenario file and there setup procedure is explained in Appendix A.* *c. Click signal power button on the toolbar and adjust the signal power to 5 for all the channels.*	
3	Press 'Start' button from the toolbar.	
4	***Namuru Setup:***	
5	Connect the Namuru to a power source between 6-18V DC via the 6 pin connector.	
6	Connect the Namuru to an RS422-USB adaptor via the 20 pin connector.	
7	Connect the RS422-USB adaptor to PC2.	
8	Power up Namuru.	
9	a. Create a folder named "Testdata" in PC2 (see: Figure 1). b. Open Termite program c. Click 'Setting' tab. d. Set the following properties i. Port: system assigned port for RS422-USB adaptor. ii. Baud rate: 115200 iii. Data bits: 8 iv. Stop bits: 1 v. Parity: none vi. Flow control: none	

	vii. Forward: none viii. Tick “Append CR-LF”, “Local echo” and “Word wrap” ix. In “Plug-ins” box tick “Logfile”. This will open an window for data logging. Put TTFF_ColdStart_XX.nmea in the filename section and locate 'Testdata’ folder to save.	
10	Send the following command to the receiver for clearing up its non-volatile memory: $PNSWC,CLN To test if the non volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
11	Restart the Namuru board by switching off from power source and switch on.	
12	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uITFEAfC,4 Note: This command will prevent restoring alamanc, ephemeris, time and last known position. This setup for cold start.	
13	Send the following command to the receiver for restarting automatically for 100 times while following cold start requirement. $PNSWC,TST,START,100,1200	
14	By default, Namuru provides the following messages but if nothing shows in HyperTerminal in 7 minutes then send following commands to Namuru for making sure that the desired data is output by the receive: $GPGPQ,GGA,E $GPGPQ,OBS,E $GPGPQ,GRS,E $GPGPQ,GSA,E $GPGPQ,GSV,E $GPGPQ,VTG,E $GPGPQ,ZDA,E $GPGPQ,XYZ,E $GPGPQ,ION $GPGPQ,UTC	

	$GPGPQ,PPS,E Default message will be similar to: After start of receiver ~~$PNSWR,FWV,AQURIUS,1.5.2,2.1.2,20130524_124708*1C ~~$PNSWR,ION,663,52,05,02,FF,FF,27,04,FF,FA*19 ~~$PNSWR,AAM,0,0,05,,,,,+2971.2,+40000,,000,00,0,0*39 ~~$PNSWR,AAM,0,1,06,,,,,+2971.2,+40000,,000,00,0,0*3B ~~$PNSWR,AAM,0,2,08,,,,,+2971.2,+40000,,000,00,0,0*36 ~~$PNSWR,AAM,0,3,09,,,,,+2971.2,+40000,,000,00,0,0*36 ~~$PNSWR,AAM,0,4,12,,,,,+2971.2,+40000,,000,00,0,0*3B ~~$PNSWR,AAM,0,5,16,,,,,+2971.2,+40000,,000,00,0,0*3E ~~$PNSWR,AAM,0,6,25,,,,,+2971.2,+40000,,000,00,0,0*3D ~~$PNSWR,AAM,0,7,27,,,,,+2971.2,+40000,,000,00,0,0*3E ~~$PNSWR,AAM,0,8,28,,,,,+2971.2,+40000,,000,00,0,0*3E ~~$PNSWR,AAM,0,9,30,,,,,+2971.2,+40000,,000,00,0,0*36 ~~$PNSWR,AAM,0,10,31,,,,,+2971.2,+40000,,000,00,0,0*0F ~~$PNSWR,AAM,2,0,193,,,,,+2971.2,+40000,,000,00,0,0*05 ~~$PNSWR,UTC,000000,00000000,0,52,663,61,2,0*27 In case of cold start, within 7to 10 minutes of receiver restarts following message will show in the HyperTerminal screen: ~~$PNSWR,OBS,2,1,1,407291.088000000,1650,0,0,0,0,0.142857*0E ~~$PNSWR,OBS,2,2,G,1,5,0,01b,407291.020226093,20318106.109,-373.703,0.000,00,000,50,337,33.5*6B ~~$GPZDA,170811.097,25,08,2011,+00,00*70 ~~$GPGGA,000000.000000,0000.0005,S,00000.0048,W,0,00,,-0.8,M,0,M,,*4E ~~$GPGSA,A,3,,,,,,,,,,,,,,*1C ~~$GPGSV,3,1,12,05,,,50,06,,,16,08,,,28,09,,,24*74 ~~$GPGSV,3,2,12,12,,,26,16,,,26,25,,,26,27,,,24*7F ~~$GPGSV,3,3,12,28,,,22,30,,,27,31,,,26,193,,,22*4B ~~$GPVTG,090.0,T,,M,00.0,N,00.0,K*69 ~~$PNSWR,XYZ,0,1650,0.000,0.0,0.0,0.0,0.000,0.000,0.000,0.0,0.000,*23 In case of cold start, within 32 minutes of receiver restarts following	

	message will show in the HyperTerminal screen wher it can be seen that Namuru has successfully produces first position fix: ~~$GPZDA,171748.112,25,08,2011,+00,00*7E ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,1,04,1.9,61298 2.6,M,0,M,,*77 ~~$GPGSA,A,3,5,6,8,21,,,,,,,,,4.4,1.9,3.9*08 ~~$GPGSV,3,1,12,05,,,50,06,,,50,08,,,50,04,,,24*74 ~~$GPGSV,3,2,12,03,,,26,19,,,25,02,,,20,07,,,25*73 ~~$GPGSV,3,3,12,21,,,50,17,,,20,11,,,27,193,,,24*40 ~~$GPGRS,171748.10,0,+0.0,+0.0,+0.0,+0.0,,,,,,,,*6E ~~$GPVTG,339.7,T,,M,14823.3,N,27452.7,K*60 ~~$PNSWR,XYZ,1,1650,407868.096,-590418.0,2875400.2,6325608.6,3901.182,-5821.454,3007.306,2487035.6,569.918,*14 Note: Position fix can be seen in GPGGA message 7th field(i.e. bold in following example) ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,**1**,04,1.9,61298 2.6,M,0,M,,*77	
15	***Final setup:***	
16	Stop data logging on Termite.	

.

4.13.3 TTFF identification

Step	Step description	Pass/Fail
5	Open Matlab program.	
6	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
7	Provide the nmea file(i.e. ColdStart_XX.nmea) name for following variable. filename='H:\Data\ ColdStart _01.nmea';	
8	Give following command coldstart(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.14 Test procedure: Warm start test

4.14.1 Introduction

The Namuru V3.2 spaceborne GPS receiver supports a warm start in LEO using the time returned from an in-built RTC and data stored in NVM. This test is to identify the warm start TTFF.

4.14.2 Test procedure

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Please follow section 4.13.2 steps 3 to step 4.	
4	***Namuru Setup:***	
5	Please follow section 4.13.2 steps 5 to step 8.	
6	Please follow section 4.13.2 steps 9(i) to (ix). However, file name should be TTFF_WarmStart_XX.nmea in the filename section.	
7	Send the following command to the receiver for clearing up its non-volatile memory: $PNSWC,CLN To test if the non-volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
8	Restart the Namuru board by switching off from power source and switch on.	
9	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uITEafc,4 Note: This command will prevent restoring ephemeris. This setup for warm start.	

10	Send the following command to the receiver for restarting automatically for 100 times while following cold start requirement. $PNSWC,TST,START,100,600	
11	Please follow section 4.13.2 steps 12 to step 16.	

4.14.3 TTFF identification

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named "TestScripts_v_01"[i.e. provided by ASCER,UNSW]	
3	Provide the nmea file(i.e. WarmStart_XX.nmea) name for following variable. filename='H:\Data\ WarmStart_01.nmea';	
4	Give following command warmstart(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.15 Test procedure: Hot start test

4.15.1 Introduction

This test is to identify the hot start TTFF.

4.15.2 Test procedure

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Please follow section 4.13.2 steps 3 to step 4.	
4	***Namuru Setup:***	
5	Please follow section 4.13.2 steps 5 to step 8.	
6	Please follow section 4.13.2 steps 9(i) to (ix). However, file name should be TTFF_HotStart_XX.nmea in the filename section.	
7	Send the following command to the receiver for clearing up its non-	

	volatile memory: $PNSWC,CLN To test if the non volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
8	Restart the Namuru board by switching off from power source and switch on.	
9	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uITFeafc,4 Note: This command will restore ephemeris, almanac, time and last known position. This setup for warm start.	
10	Send the following command to the receiver for restarting automatically for 100 times while following cold start requirement. $PNSWC,TST,START,100,600	
11	Please follow section 4.13.2 steps 12 to step 16.	

4.15.3 TTFF identification

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
3	Provide the nmea file(i.e. WarmStart_XX.nmea) name for following variable. filename='H:\Data\ WarmStart_01.nmea';	
4	Give following command hotstart(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.16 Cold start TTL

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
3	Provide the nmea file(i.e. ColdSart_XX.nmea) name for following variable. filename='H:\Data\ ColdSart _01.nmea';	
4	Give following command TTL(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.17 Warm start TTL

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
3	Provide the nmea file(i.e. WarmSart_XX.nmea) name for following variable. filename='H:\Data\ ColdSart _01.nmea';	
4	Give following command TTL(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.18 Hot start TTL

Step	Step description	Pass/Fail
1	Open Matlab program.	
2	Browse to the folder named “TestScripts_v_01”[i.e. provided by ASCER,UNSW]	
3	Provide the nmea file(i.e. HotSart_XX.nmea) name for following variable.	

	filename='H:\Data\ ColdSart _01.nmea';	
4	Give following command TTL(filename) *This program will generate a generate graphs as well as provides the statistics from the truth.	

4.19 Acquisition Sensitivity

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Load a scenario file: This will be the main (root) file for the simulation scenario as defined above. This can be done as follows: *a. Go to File menu and click on Load.* *b. Browse to the folder named NamuruTest/ numarufunctiontest (i.e. provided by ACSER,UNSW) and select the file named 'numarufunctiontest_base.scn'.* *c. Click signal power button on the toolbar and adjust the signal power to 5 for all the channels.*	
4	***NamuruV3.2 Setup:***	
5	Please follow section 4.13.2 steps 5 to step 8.	
6	Please follow section 4.13.2 steps 9(i) to (ix). However, file name should be ACQ_ColdStart_XX.nmea in the filename section.	
7	Send the following command to the receiver for clearing up its non-volatile memory: $PNSWC,CLN To test if the non volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
8	Restart the Namuru board by switching off from power source and switch on.	

9	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. $PNSWR,CFG,5,3,uITeafc,4 Note: This command will prevent restoring ephemeris. This setup for warm start.	
10	Wait for Namuru to start outputting navigation solution. That means third filed of $PNSWR,XYZ message will be '1'. It will be similar like followings E.G: ~~$PNSWR,OBS,5,1,578,407868.088000000,1650,0,0,0,0,0.396828*0D ~~$PNSWR,OBS,5,2,5,0,01f,407868.016966246,21295383.625,3381.320,0.000,00,000,50,337,32.9*7C ~~$PNSWR,OBS,5,3,6,1,01f,407868.020792380,20148337.531,-5879.297,0.000,00,000,50,337,32.9*55 ~~$PNSWR,OBS,5,4,8,2,01f,407868.019665176,20486264.828,-4600.695,0.000,00,000,50,337,32.9*55 ~~$PNSWR,OBS,5,5,21,8,01f,407868.027859430,18029689.453,44.961,0.000,00,000,50,337,32.9*48 ~~$GPZDA,171748.112,25,08,2011,+00,00*7E ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,1,04,1.9,612982.6,M,0,M,,*77 ~~$GPGSA,A,3,5,6,8,21,,,,,,,,,4.4,1.9,3.9*08 ~~$GPGSV,3,1,12,05,,,50,06,,,50,08,,,50,04,,,24*74 ~~$GPGSV,3,2,12,03,,,26,19,,,25,02,,,20,07,,,25*73 ~~$GPGSV,3,3,12,21,,,50,17,,,20,11,,,27,193,,,24*40 ~~$GPGRS,171748.10,0,+0.0,+0.0,+0.0,+0.0,,,,,,,,*6E ~~$GPVTG,339.7,T,,M,14823.3,N,27452.7,K*60 **~~$PNSWR,XYZ,*1*,1650,407868.096,-590418.0,2875400.2,6325608.6,3901.182,-5821.454,3007.306,2487035.6,569.918,*14**	
11	Once this message shows '1', record the slider value.	
12	Repeat step 1 to 11 with reduced signal power. Signal power In PosApp application should be reduced to identify acquisition sensitivity. Click on the signal power window and then Click on the slider. Step down signal power using keyboards '↓' key. Every time '↓' key is pressed signal power is down by 0.2 dBm.	

13	Repeat this test 20 times and take average and standard deviation.	

4.20 Tracking Sensitivity

Step	Step description	Pass/Fail
1	***Spirent Setup:***	
2	Open PosApp.	
3	Please follow section 4.13.2 steps 3 to step 4.	
4	***Namuru Setup:***	
5	Please follow section 4.13.2 steps 5 to step 8.	
6	Please follow section 4.13.2 steps 9(i) to (ix). However, file name should be ACQ_ColdStart_XX.nmea in the filename section.	
7	Send the following command to the receiver for clearing up its non-volatile memory: $PNSWC,CLN To test if the non volatile memory is clean or not, following commands can be sent to receiver: $GPGPQ,ALM $GPGPQ,EPH Receiver should output nothing as neither almanac nor ephemeris data is stored.	
8	Restart the Namuru board by switching off from power source and switch on.	
9	Send the following command to the receiver for setting up GPS SV visibility mask angle as 5° and solution mode as 3D. **$PNSWR,CFG,5,3,uITeafc,4** Note: This command will prevent restoring ephemeris. This setup for warm start.	
10	Wait for Namuru to start outputting navigation solution. That means third filed of $PNSWR,XYZ message will be '1'. It will be similar like followings E.G: ~~$PNSWR,OBS,5,1,578,407868.088000000,1650,0,0,0,0,0.396828*0D	

	~~$PNSWR,OBS,5,2,5,0,01f,407868.016966246,21295383.625,3381.320,0.000,00,000,50,337,32.9*7C ~~$PNSWR,OBS,5,3,6,1,01f,407868.020792380,20148337.531,-5879.297,0.000,00,000,50,337,32.9*55 ~~$PNSWR,OBS,5,4,8,2,01f,407868.019665176,20486264.828,-4600.695,0.000,00,000,50,337,32.9*55 ~~$PNSWR,OBS,5,5,21,8,01f,407868.027859430,18029689.453,44.961,0.000,00,000,50,337,32.9*48 ~~$GPZDA,171748.112,25,08,2011,+00,00*7E ~~$GPGGA,171748.096296,6514.4113,N,10136.2100,E,1,04,1.9,612982.6,M,0,M,,*77 ~~$GPGSA,A,3,5,6,8,21,,,,,,,,,4.4,1.9,3.9*08 ~~$GPGSV,3,1,12,05,,,50,06,,,50,08,,,50,04,,,24*74 ~~$GPGSV,3,2,12,03,,,26,19,,,25,02,,,20,07,,,25*73 ~~$GPGSV,3,3,12,21,,,50,17,,,20,11,,,27,193,,,24*40 ~~$GPGRS,171748.10,0,+0.0,+0.0,+0.0,+0.0,,,,,,,,*6E ~~$GPVTG,339.7,T,,M,14823.3,N,27452.7,K*60 **~~$PNSWR,XYZ,*1*,1650,407868.096,-590418.0,2875400.2,6325608.6,3901.182,-5821.454,3007.306,2487035.6,569.918,*14**	
11	In PosApp application, GNSS signal power will be reduced to identify tracking sensitivity. Click on the signal power window and then Click on the slider. Step down signal power using keyboards '↓' key. Every time '↓' key is pressed signal power is down by 0.2 dBm. After every press wait for 1 minute and observe third filed of '**$PNSWR,XYZ'** message. Once this message shows '0', record the slider value.	
12	Repeat step 11 20 times and take average and standard deviation.	

5 Prototype Receiver Hardware Designs

This section describes the architecture, design challenges and development path of the Namuru V3 receivers.

Table 5-1 shows all the receiver platforms used at some stages of the project development. As mentioned earlier, Namuru V2.4 and V2.5 were used for initial experimentation with some of the software algorithms as they were time tested and available in good numbers. V3.1 is a L1-band-only receiver and design-wise a subset of the V3.3 receiver. The "Lego" receiver is a dual-band receiver built with COTS modules. The V3.1 and Lego receivers were used as part of the contingency plan during V3.3 hardware development delays, which were significant. V3.4 is functionally similar to V3.3 but with two front end bands combined into a single chip.

Hardware	RF Front-end		FPGA device	Processor
	L1/E1	L5/E5		
V2.4	GP2015	NA	Altera EP2C50	NIOS II/f
V2.5	GP2015	NA	Altera EP2C50	NIOS II/f
V3.1	BL2627	NA	Altera EP4CGX50	NIOS II/f ← Contingency Plan
V3.2 (Biarri)	GP2015	NA	Actel ProASIC A3	ARM Cortex M3
V3.3	**BL2627**	**GD1030**	**Altera EP4CGX150**	**2x NIOS II/f**
V3.4	BL2627	GD1040	Altera EP4CGX150	2x NIOS II/f
Lego	MAX2769	MAX2112	Altera EP4CE115	2x NIOS II/f ← Contingency Plan

Garada project specific →

Table 5-1 Namuru receiver platforms overview

*** Note: (1) GP2015 has 2MHz bandwidth and cannot receive the full E1 signal. (2) BL2627 has 6MHz bandwidth and can receive the basic (BOC) component of the E1 signal. (3) GD1030 can be configured to receive the full E1 signal (CBOC / TMBOC) with a bandwidth of 15 MHz.*

Following sub-sections provide details about V3.1, V3.2 V3.3 and V3.4 hardware.

a. Namuru V3.2 hardware

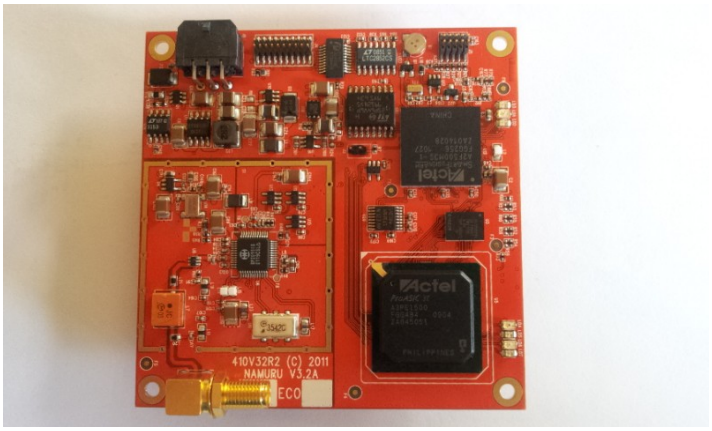


Figure 5-1 Namuru V3.2 board

Figure 5-1 shows the picture of a Namuru V3.2 board. It is a mechanically balanced, 6-layer PCB with surface mount components and the dimensions are matched to a Colony II Cubesat payload. The RF is shielded to minimize the interference effects on the receiver. The latch up protection circuitry resets the receiver in case of high inrush current due to radiation in space. The functional blocks of V3.1 receiver are shown in Figure 5-2. The receiver comprises of L1/E1 RF front-end ASIC with 2 MHz bandwidth, an Actel ProASIC FPGA (A3PC1500) for the baseband signal processing and an ARM CPU + Flash (Actel Smartfusion FPGA A2F500), a VTCXO with 10MHz +/- 1.5 ppm, 1 MByte SRAM, Battery backed RTC, RS422 & GPIOs. The RF front-end is GP2015, a well proven widely used chip developed from Zarlink Semiconductors. The functional block diagram is shown in Figure 5-2.

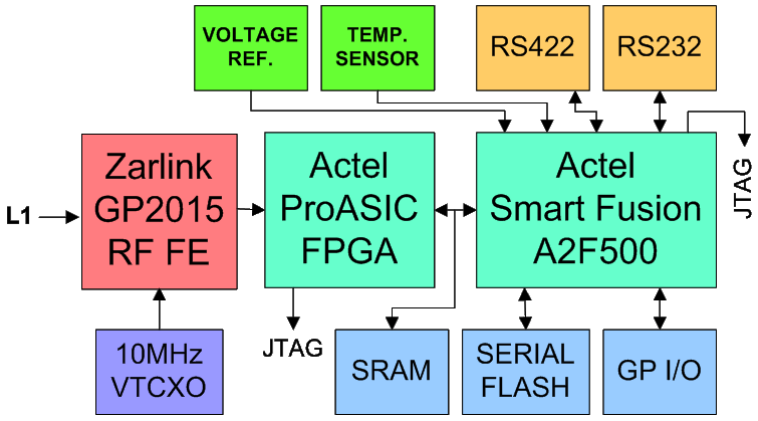


Figure 5-2 Functional block diagram of V3.2

b. Namuru V3.1 hardware

Figure 5-3 shows the picture of a Namuru V3.1 board. It is a credit card sized 6-layer PCB with surface mount components. The RF is shielded to minimize the interference effects on the receiver. The prototype version runs off the USB power. The functional blocks of V3.1 receiver are shown in

Figure 5-4. The receiver comprises of L1/E1 RF front-end ASIC with 6MHz bandwidth, an Altera Cyclone 4 FPGA, a VTCXO with 16.368MHz +/- 1.5 ppm, 1 MByte SRAM, Battery backed RTC, FRAM, RS232, USB & GPIOs. The RF front-end BL2627 was developed by General Dynamics, NZ.

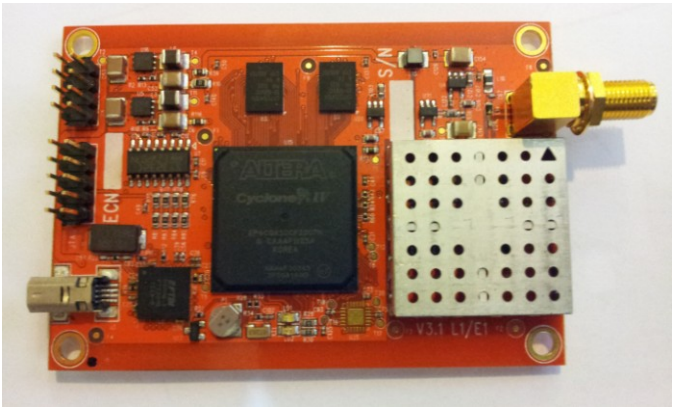


Figure 5-3 Namuru V3.1 board

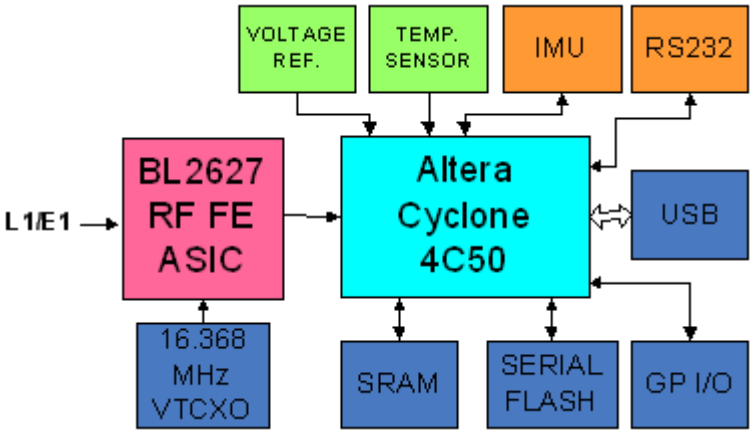


Figure 5-4 Namuru V3.1 architecture

c. Namuru V3.3 and V3.4 Hardware

Figure 5-5 shows the pictures of Namuru V3.3 and V3.4 boards. These boards have three RF Front End channels out of which two can be configured to receive any of the L1/E1, L2, L5/E5a, E6, GLONASS and the other RF channel is fixed to L1/E1. All the RF channels run from a single reference V-TCXO at 16.368 MHz. The FPGA is Altera Cyclone 4 GX150 and it has enough resources to accommodate baseband signal processing modules for signals from all the three RF channels. The boards have a battery backed RTC, Serial 128K x 8 FRAM, RS232, USB & GPIOs. The fixed-band RF front-end is BL2627 same as in the V3.1 board. The other two channels have a tunable RF front-end GD1030 developed by General Dynamics, NZ. On the V3.4 board, functionalities of two GD1030 chips are built into a single chip GD1040. V3.4 differs to V3.3 also in the number of ADC output bits which is 2 in the case of V3.3 and 12 in the case of V3.4.

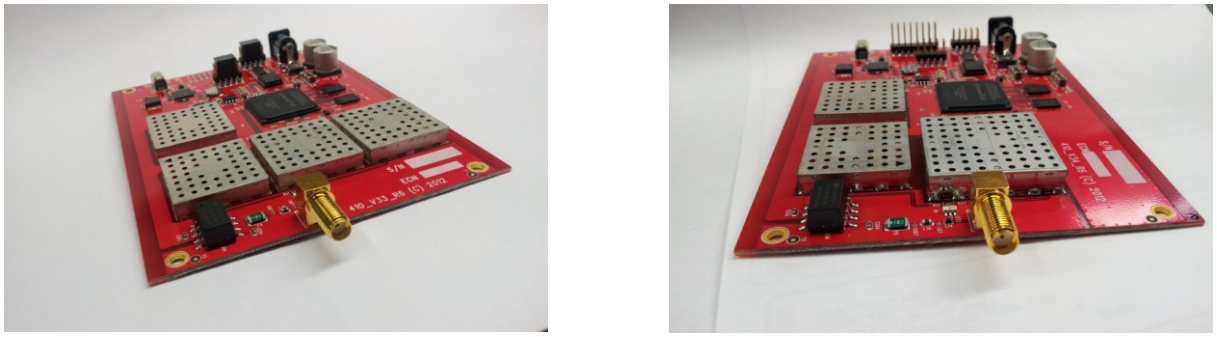


Figure 5-5 Namuru V3.3 (left) and Namuru V3.4 (right)

The digital sections of V3.3 and V3.4 are similar in nature and the architecture of the digital section is shown in Figure 5-6. The embedded processor is the NIOS II/f RISC processor from Altera. NIOS processors are also used in all the V2 family boards at UNSW.

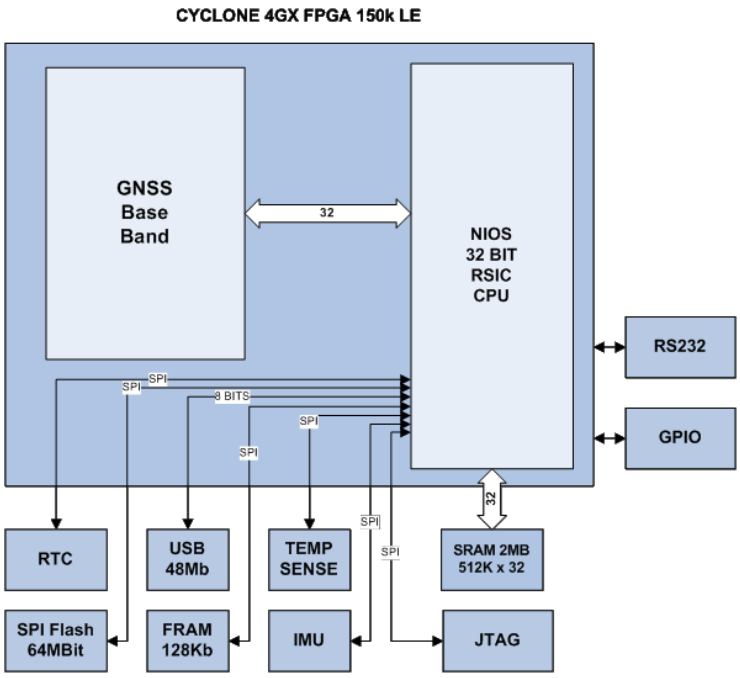


Figure 5-6 Digital section architcture in V3.3 and V3.4 boards

Figure 5-7 shows the Namuru V3.3 board with the cans open. The fourth shield part is the clock distribution circuitry implemented to provide synchronized clocks across the board for the RF front-ends and the digital section.

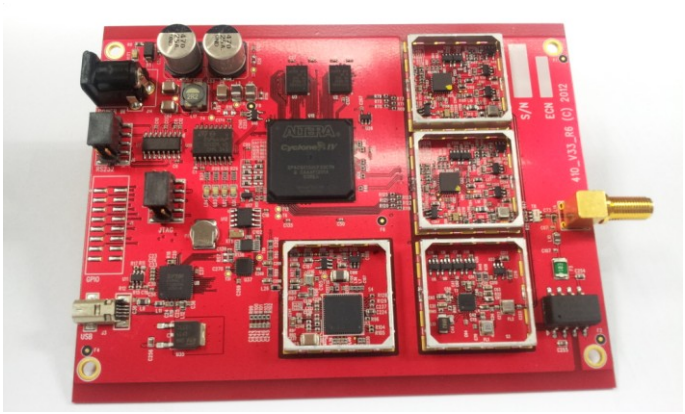


Figure 5-7 Namuru V3.3 board

The signal from the antenna passed through as splitter and the output of the splitter feeds all the RF front-ends on the board. The PLL synthesizer helps distribute the system clock across the board as mentioned earlier. The digitized IF signals from all the front-ends are then taken into the FPGA for the baseband processing. The FPGA-processor subsystem with the required peripherals delivers the PVT solution to the mission computer. The functional block diagram of V3.3 is shown in Figure 5-8.

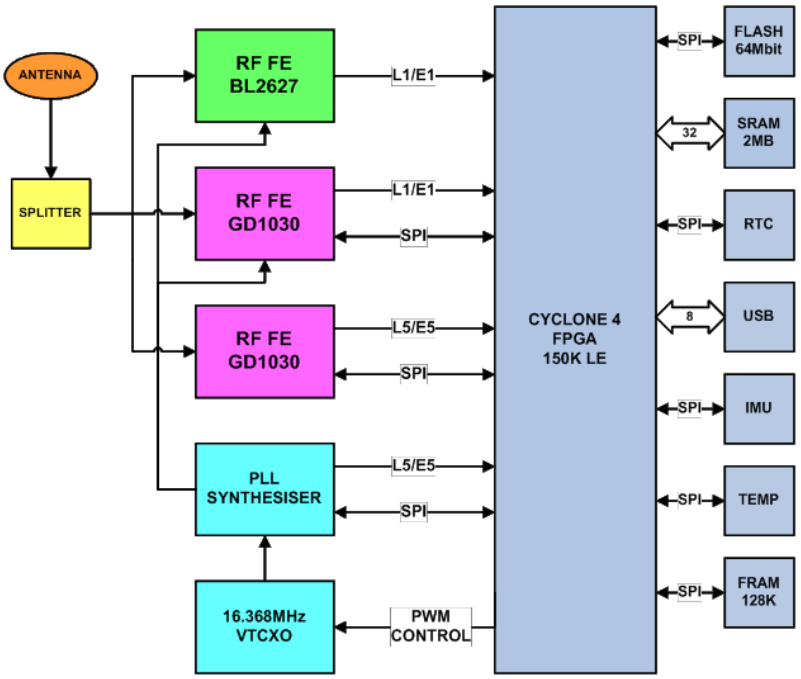


Figure 5-8 Namuru V3.3 Functional blocks

Figure 5-9 and Figure 5-10 show the board layout and functional bock diagram of V3.4. Note that the two GD1030 functionalities are combined in the GD1040 chip. GD1040 is in effect an integrated dual-channel RF front-end.

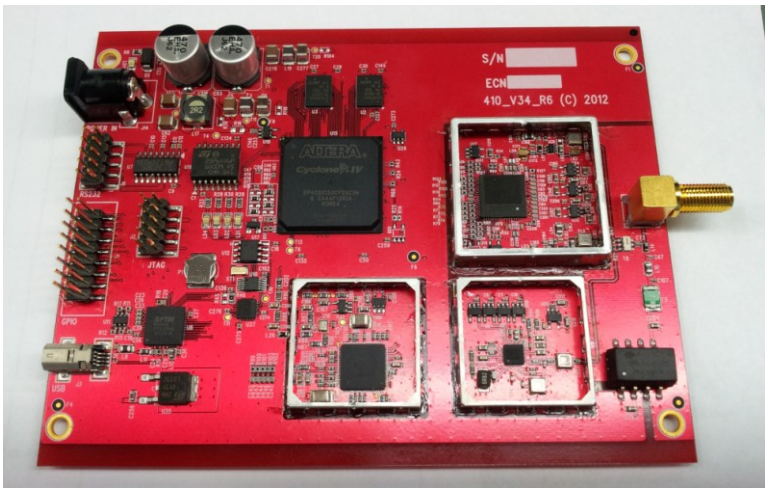


Figure 5-9 Namuru V3.4 Hardware

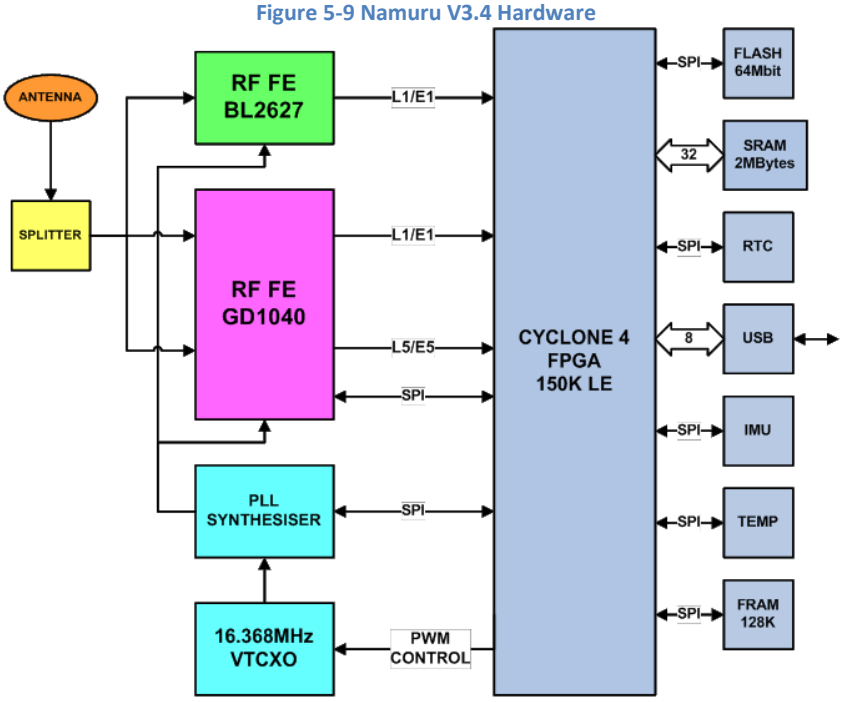


Figure 5-10 Namuru V3.4 Functional Block Diagram

A picture (obtained from design tools) of the GD1030 ASIC is shown in Figure 5-11. GD1030 is a QFN 32-pin package with dimensions measuring 5x5mm. The die is 2x2mm developed with the 130nm CMOS technology and the core voltage is 1.8V.

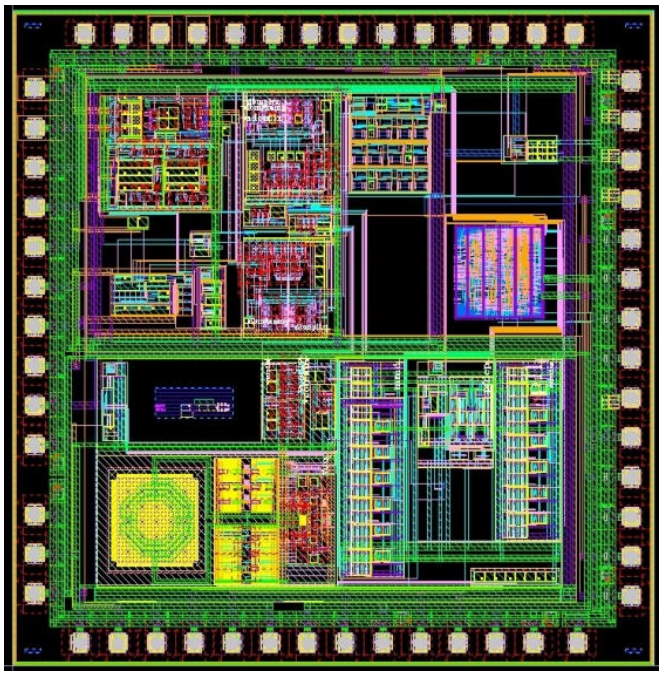


Figure 5-11 inside the GD1030 ASIC

d. Lego receiver setup

During the course of the project development, while waiting for the V3.3 hardware, the receiver design team built dual-band hardware with two tuneable RF evaluation modules (EVMs) and FPGA development board. Building of this system involved some board-to-board connections and the boards themselves consumed some physical space. This system was termed Lego receiver due to its 'building blocks' nature of the design. This board of course is not suitable for delivery but was good enough for experiments with the dual band GNSS signals before V3.3 was fabricated.

Figure 5-12 shows the top level blocks in the Lego receiver. MAX2769 EVM was used for the L1/E1 band and MAX2112 which is primarily a DVB system EVM, was used for the L5/E5a band. MAX 2112 doesn't have the built-in ADC and hence a TI ADC EVM board was used to digitize the signals. The FPGA board is the Altera DE2-115 education board obtained through Altera University Program. Though several issues were faced in setting up this system due to the RF front-ends' incompatibility with PC configurations, finally the FPGA could successfully talk to the RF front-ends to configure and receive GNSS signals. This system was used mainly for the baseband verification purpose and before the firmware development for this board, UNSW received V3.3 and V3.1 boards and further developments were continued on the V3 boards.

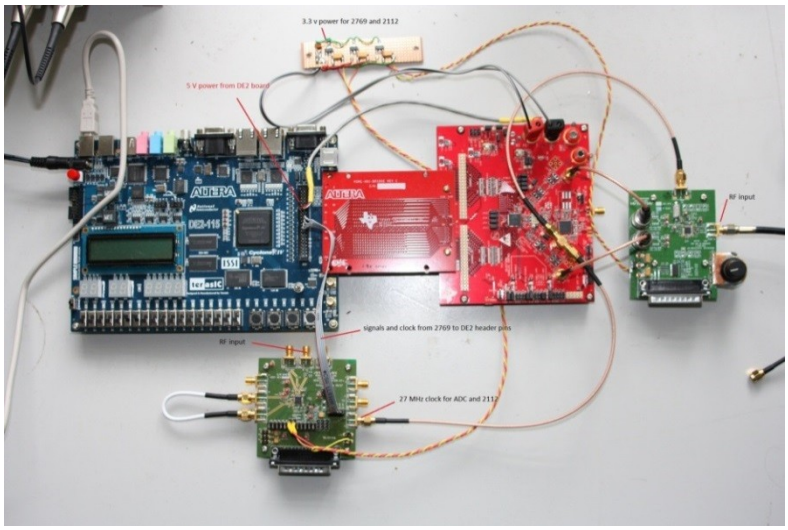


Figure 5-12 The Lego Receiver

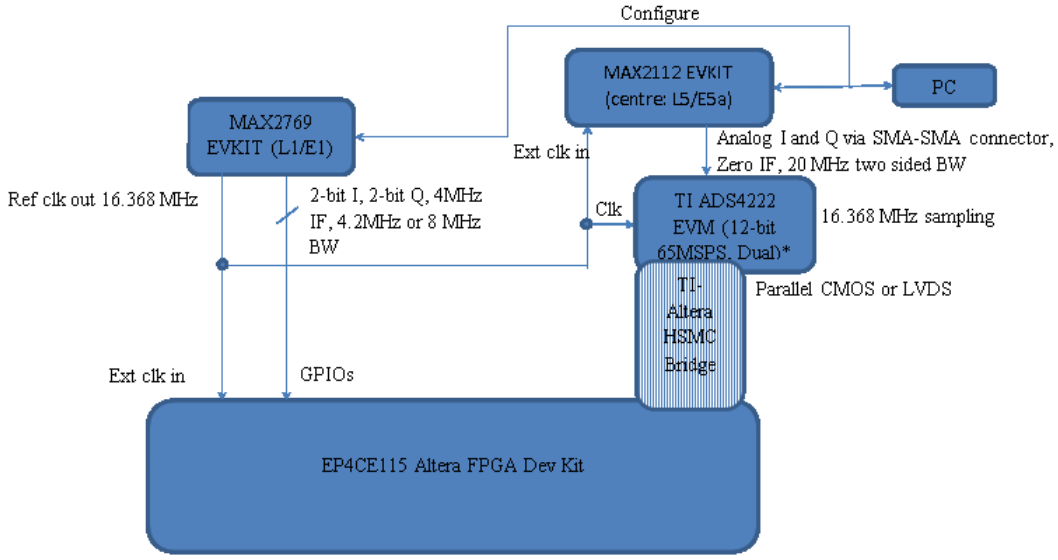


Figure 5-13 Top level block diagram of the Lego receiver

6 Space-certifiable receiver hardware design strategies used

6.1 Oscillator

The main oscillator for a GNSS receiver needs careful attention, often requiring short term stability of better than 2.0ppm. In space the oscillator will most likely need to maintain stability over a greater range of temperatures. The Namuru V3 receivers are equipped with a Temperature Controlled Crystal Oscillator (TCXO) with the option to pull the oscillator up to 5ppm either side of the nominal frequency. An option is also provided for an Oven Controlled Crystal Oscillator (OCXO) with the same frequency adjusting capability. A software algorithm drives the oscillator corrections while monitoring a precise temperature sensor.

6.2 Power supplies

All voltages are derived from high frequency (1MHz and above) high efficiency switch mode power supplies to minimise heat dissipation and reduce power consumption. In general each switch mode supply is capable of at least twice the required current to ensure low thermal dissipation especially

while recovering from latch-up conditions. Each supply also has the ability to switch over to cycle skip mode on light loads providing further power savings.

6.3 Latch-up protection and recovery

The only practical way to recover from a latch-up condition in semiconductor devices is to remove the power and restart. A special latch-up protection circuit is designed into the Namuru V3 power supply to remove power when the current drain exceeds a set threshold. After a short programmable time the supply is restored, after which the receiver will recover as if a power failure had occurred.

6.4 Construction

Traditional tin-lead paste alloy is used on the Namuru V3 space receiver because it is well understood and found to be reliable for component connections, as opposed to some of the recent lead free products which have not yet been proven for long term reliability (D. R. Frear, et al., 1994).
The multilayer printed circuit board is constructed with solid copper plating or plugs through via holes for electrical connection between layers. This gives greater reliability under variable temperature conditions and improved heat transfer to the inner layers where larger copper areas can dissipate heat.

6.5 Shielding

The RF front end section is shielded using a metal enclosure. Additional shielding between circuit sections is provided using careful printed circuit board layout by dividing large copper plane areas into segments. This ensures less coupling between low and high signal level areas. Some consideration is being given to the new PolyRAD shielding material being developed for NASA that is predicted to substantially reduce TID (NASA, 2002).

7 FPGA-based Multi-GNSS Baseband Logic Design

7.1 Baseband module overview

Figure 7-1 provides the functional details of the FPGA baseband module. The functionality is divided into three sub modules, the correlator, the correlator controller and the interface controller. The correlator module comprises of the carrier and code mixers and accumulators for all the signals GPS L1, Galileo E1b and E1c, GPS L5 and Galileo E5a. The correlator controller implements the carrier NCO, carrier generator, code NCO, code generator and timing control signals for the correlator and interface controller blocks. The interface controller manages the memory mapped interface to the processor.

The correlation computation block and the correlator controller block are instantiated for the required number of channels. The interface controller implements a memory-mapped interface to the processor and comprises of multiplexers/ de-multiplexers and the baseband wrapper for all the four signals.

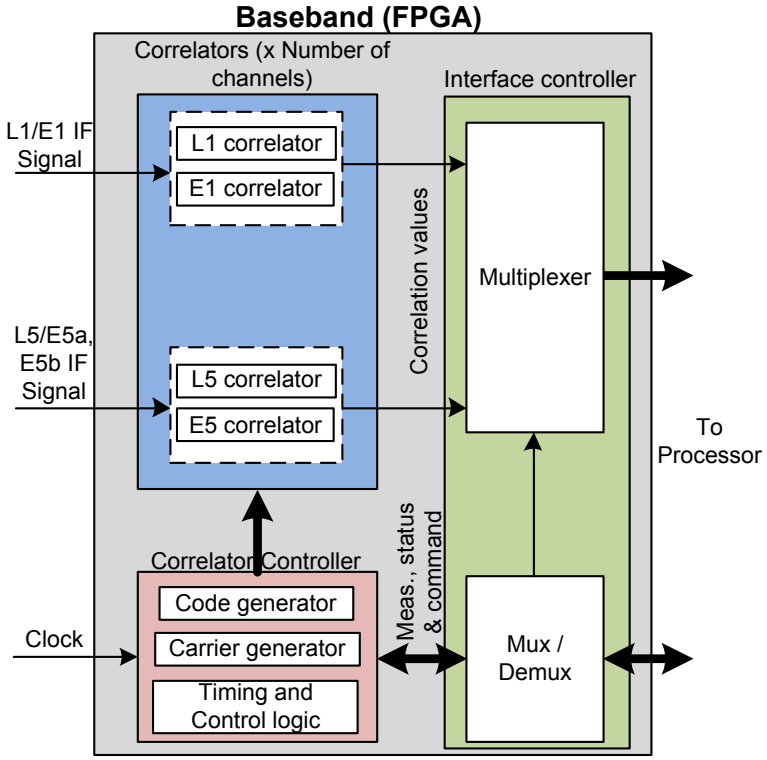


Figure 7-1 Functional details of the baseband module

The correlation computation block accepts the IF signal samples and the local reference signals as the input and produces the correlation values. The correlation computation block is again divided into two sub modules, the sample correlator and the accumulator as shown in Figure 7-2.

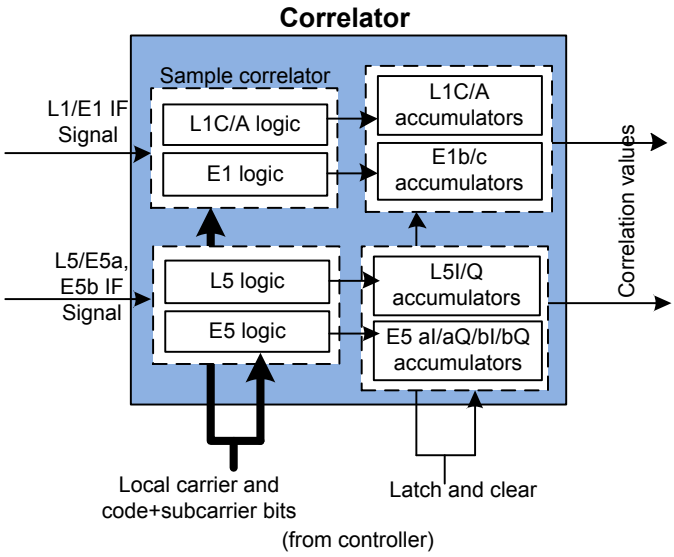


Figure 7-2 Details of the correlation computation block

Figure 7-3 shows the details of the correlator controller block for a single channel. The correlator controller module comprises of the carrier and code generation blocks and the control signals generation block. The processor programs the PRN number, carrier frequency, code delay and the integration duration for each channel of the baseband module. The correlator controller module provides the measurement output upon a measurement latch request from the processor.

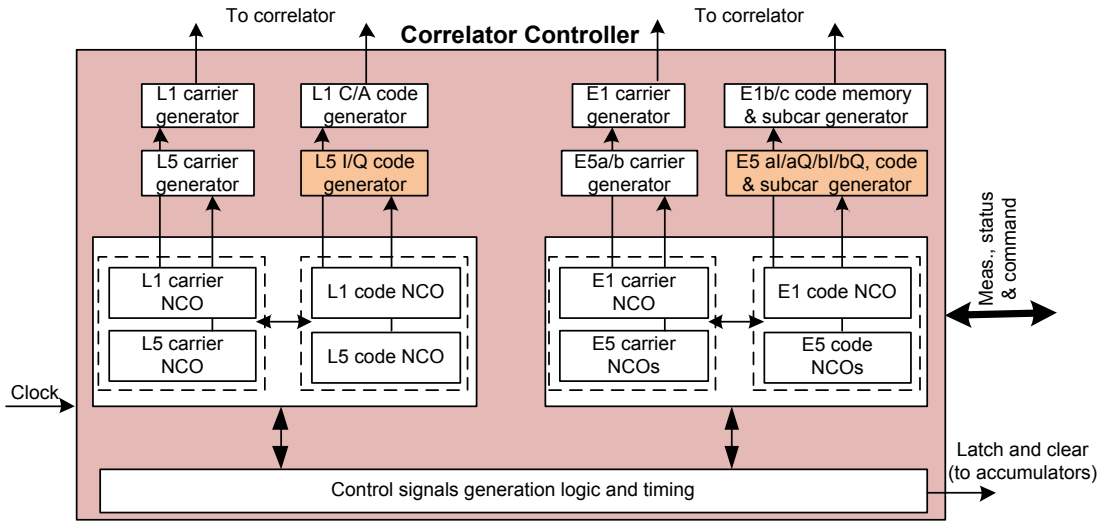


Figure 7-3 Details of the correlator controller module

7.2 Baseband and processor interface

Table 7-1 Top level memory map of the baseband shows the baseband module address map for the baseband module as seen from the processor. This is an extension of the existing Namuru receiver address map (Namuru data sheet, 2007). The offset address for all the signals have reserved bits to accommodate additional features and also allowing scalability to increase the number of code fingers for fast acquisition architectures in future.

Offset Address	Description
000-1FF	**GPS L1 C/A** (same as the existing Namuru legacy map)
200-5FF	**Galileo E1**
600-9FF	**GPS L5**
A00-DFF	**Galileo E5**
E00-FFF	**Reserved**

Table 7-1 Top level memory map of the baseband

7.3 Garada baseband signal processor features and Summary

Following is the list of salient features of the Garada multi-GNSS baseband:

- 2-bit IF (Sign/ Mag)
- Signals: GPS L1 C/A, GPS L5, Galileo E1, Galileo E5a
- GPS L1 C/A
  - 12 Channels, 3 code fingers per channel
  - 2 (I and Q) correlation values per finger
- Galileo E1
  - 12 Channels, 8 code fingers
  - 4 (I and Q, Pilot and Data) correlations per finger
- GPS L5
  - 12 Channels, 3 code fingers
  - 4 (I and Q, Pilot and Data) correlations per finger
- Galileo E5
  - 12 Channels, 8 code fingers
  - 4 (I and Q, Pilot and Data) correlations per finger
- Integration over one code period
- Code and carrier measurement for all channels
- Control and status registers to enable the processor firmware to command according to the acquisition and tracking algorithms and to read the status at any point in time
- Timing and control signals to schedule different computations within the correlation process
- Separately programmable for GPS and Galileo to enable independent acquisition, tracking of signals
- Default reference for the timing purposes across signals of both the systems is the GPS L1 C/A code.

Table 7-2 shows the summary of the important features. The firmware can extend the integration duration beyond the duration present in the baseband without affecting the correlation values or measurements.

Signal	# Channels	# Code fingers	# Accumulators per finger	Integration duration(ms)
GPS L1 C/A	12	3	2 (I, Q)	1
Galileo E1	12	8	4 (I,Q, Pilot,Data)	1 or 4
GPS L5	12	3	4 (I,Q, Pilot,Data)	1
Galileo E5	12	8	4 (I,Q, Pilot,Data)	1

Table 7-2 Garada baseband details for different signals

Figure 7-4 shows the screenshot of the baseband digital circuit obtained from the Altera FPGA design tool. The four-signal (L1/E1/L5/E5a), 12 channel core baseband design occupies about 80,000 Cyclone IV GX Logic Elements (LEs).

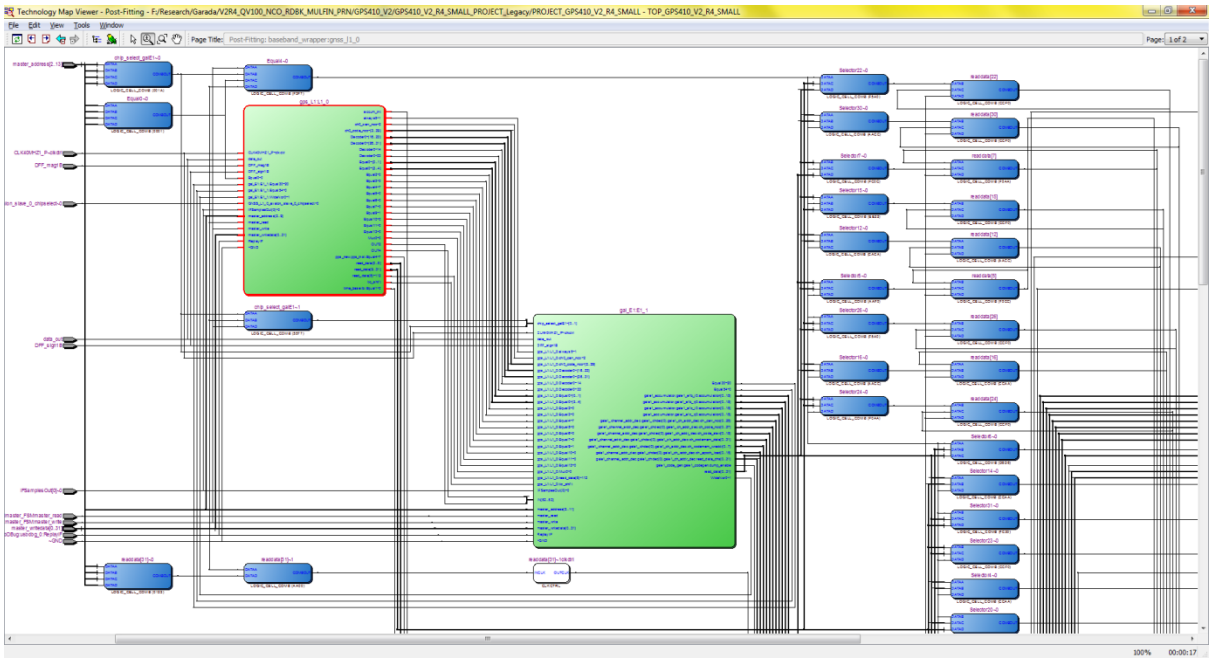


Figure 7-4 Screenshot of the baseband digital circuit from the FPGA design tool

8 Multi-GNSS signal processing and the development of Matlab based Receiver

This section is organized as follows: the development considerations are addressed in section 8.1; the software performance including GUI illustration, acquisition, tracking and positioning performances are shown in section PERFORMANCE ILLUSTRATIONS8.2.

8.1 Development considerations

8.1.1 PRN code generation

QZSS L1C/A and the future Galileo E1open service (OS) channels use pseudorandom codes sharing the same chipping rate as GPS L1C/A. The QZSS L1C/A signals could be treated as GPS L1C/A signals, except that it uses PRN numbers of 193-197. Therefore, the software upgrade for QZSS L1C/A code on the basis of GPS L1C/A receiver is relatively straight forward.

However, Galileo E1OS signals are quite different from the GPS L1 C/A signal. It uses a more complex modulation method called Binary Offset Carrier (BOC). To be more precise, it utilises Multiplex Binary Offset Carrier (MBOC) in wide band (i.e. double sided bandwidth larger than 14MHz). However, in a relatively narrow front-end bandwidth, the Galileo E1OS signal could be treated as BOC(1,1) which is a multiplex product of a subcarrier and a BPSK pseudorandom code, both waveforms having the same chipping rate as the GPS L1C/A pseudorandom code (i.e. 1.023MHz).

Unlike GPS L1C/A signal which uses Gold code as the PRN spreading code, the Galileo system adopts optimised memory codes. Although two of its prototype satellites (e.g. GIOVE-A and GIOVE-B) still

use Gold code, the future Galileo system will operate with optimised memory code sequences according to the published ICD [2]. Therefore, to generate the local replica for Galileo system, one should take account of not only its unique subcarrier modulation but also the usage of memory code requiring additional module for code storage and access.

Existing GLONASS L1C/A signal uses Frequency Division Multiple Access (FDMA) technique, although Russia has planned for the system modernisation with CDMA as the channel access method. Each GLONASS satellite transmits FDMA signal occupying different frequency by sharing the same BPSK PRN sequence with chipping rate of 0.511MHz. In the other words, a single 511-chip length of PRN code is shared by all the GLONASS satellites.

8.1.2 Integration time

The data rates for GPS L1 C/A, QZSS L1C/A and GLONSS L1C/A are all the same (=50bps), which allows the multi-system to use at least 1ms integration for the acquisition and tracking for three of those systems. On the other hand, the design for Galileo channels is relatively complex. Galileo E1B uses a longer pseudorandom code sequence.

Galileo E1 channel is a combination of data channel (E1B) and data-less/pilot channel (E1C). On the E1B channel, the PRN code has a period of 4ms (i.e. 4092 chips with 1.023MHz chipping rate), while the message data rate is also 4ms (i.e. 250bps). This fast data rate enhances the Galileo system communication throughput, but requires a more advanced mechanism to ensure the minimisation of bit error rate. Galileo system adopts forward error correction (FEC) decoders to enhance the effectiveness of channel decoding and encoding in digital signal processing. This FEC decoder is discussed in section 8.4.

Galileo currently has two in-orbit prototype satellites and two recently launched Galileo satellites. Recently launched Galileo satellites are still under test. The GIOVE-A and GIOVE-B as two prototype in-orbit Galileo satellites have been broadcasting the signal and the navigation message for a while. The signal plan and message data structure are described in the (GIOVE-A+B OS SIS ICD, 2008 [3]) . In this paper, real GIOVE-A and GIOVE-B raw intermediate frequency (IF) data is used for testing; while Spirent IF data is used for the simulation of Galileo satellites.

It should be pointed out that the E1C channel of GIOVE satellite uses 8184chip ,8ms PRN codes with 25chip secondary code modulation, resulting in 100ms overall code repetition interval. In the current software version, 8ms integration for GIOVE E1C and 4ms integration for Galileo E1C are implemented. By taking into account the modulation of the secondary code, the integration time can be extended to enhance the receiver sensitivity for weak signal detection. To facilitate the algorithm validation of acquisition and tracking sensitivity, in the current version, the implementation options of4ms and 8ms integration for the Galileo data channel and pilot channel acquisition and tracking are provided respectively.

In order to enhance the receiver sensitivity, integration time beyond 1ms for GPS signal processing is also desired. Therefore, options of integer multiple of 1ms integration time for GPS L1C/A code is included in this version of the software. The major parameter difference between GNSS systems at L1 channel that considered in the software version is listed in **Error! Reference source not found.**.

Table 8-1 Code structure comparison of GNSS in the L1 band

Parameter	Galileo E1 OS	GIOVE-A E1 OS	GIOVE-B E1 OS	GPS L1C/A	QZSS L1C/A	GLONASS L1C/A
Number of Codes combined to form the OS signal	2 (E1b and E1c)	2 (E1b and E1c)	2 (E1b and E1c)	1(L1 C/A)	1(L1 C/A)	1(L1 C/A)
Data modulation	Yes. Only on E1b	Yes. Only on E1b	Yes. Only on E1b	Yes	Yes	Yes
Data Rate	250bps	250bps	250bps	50bps	50bps	50bps for the first 1.7s 100bps for the 0.3s sync bits
Secondary Code Overlaid?	Yes. Only for E1c (25 chips length)	Yes. Only for E1c (25 chips length)	Yes. Only for E1c (25 chips length)	No	No	No
Code length (chips)	E1b – 4092 E1c – 4092	E1b – 4092 E1c – 8184	E1b – 4092 E1c – 8184	L1C/A-1023	L1C/A-1023	L1C/A-511
Chipping rate	1.023MHz	1.023MHz	1.023MHz	1.023MHz	1.023MHz	0.511MHz
Primary code period (chips)	E1b – 4 ms E1c – 4 ms	E1b – 4 ms E1c – 8 ms	E1b – 4 ms E1c – 8 ms	1ms	1ms	1ms
Number of shift registers	Not Applicable	2 for E1b, 2 for E1c, 4 in total	2 for E1b, 2 for E1c, 4 in total	2	2	1
Shift register length	Not Applicable (Memory code)	13 bits	13 bits	10 bits	10 bits	9bbits
PRN number	1-50	51	52	1-32	193-197	0-24
Code Modulation	BOC(1,1) *	BOC(1,1)	BOC(1,1) *	BPSK	BPSK	BPSK
Transmission mode	CDMA	CDMA	CDMA	CDMA	CDMA	FDMA
Carrier frequency	1575.42MHz	1575.42MHz	1575.42MHz	1575.42MHz	1575.42MHz	1602.00MHz +(CHN#)*x0.5625MHz

***Note:**

- (CHN#) represents channel number (i.e. PRN number)
- The exiting GLONSS is still using FDMA transmission although CDMA transmission is planned
- ICD of Chinese COMPASS (Beidou) is not yet published yet. The corresponding receiver upgrade is planned
- * Precisely speaking, it is Multiplex-BOC (MBOC) modulation in wideband

8.1.3 Message decoding

Galileo E1B use a higher data rate up to 250bps comparing to the 50bps of other GNSS (see also Table 1). To increase the efficiency of navigation message reception, FEC encoding as well as block interleaving is applied before satellite signal transmission. Within each 250 bits message page, there are 10 synchronisation bits and 240 navigation data bits. In this navigation message, before FEC, block interleaving algorithm (30 columns and 8 rows) is also adopted into each E1B data page allowing each interleave block a length of 240 bits. Each page includes also the12bit cyclic redundancy check (CRC) for GIOVE E1B and 24bit CRC for Galileo E1C. It is worth pointing out that the message structure of Galileo E1B is different from GIOVE E1B. Each page of GIOVE E1B message contains 240 bits before FEC decoding; while Galileo E1B has 480 bits lasting for 2 seconds. Due to the availability of real navigation data, only the message decoding for GIOVE-A and GIOVE-B is performed for the work in this paper.

To decode the GIOVE satellite message, a receiver has to go through following processing steps:
Achieving synchronisation to the page boundary by using he 10bits I/NAV synchronisation pattern;
De-interleaving the 240bit message data.
Applying Viterbi decoder (K=7) with a rate of R=½
CRC check to ensure the correctness of extracted message data.

The QZSS L1C/A signals can be treated as additional GPS L1C/A signals. Therefore, for the QZSS L1 C/A navigation data extraction, same message decoding method used for GPS L! C/A can be applied.

Since GLONASS is a FDMA signal, the message decoding algorithm of GLONASS is not the same as those GNSS using CDMA channel access method. Extra work is planned to implement the positioning integration of GLONASS. Current software development version for GLONASS system is only limited up to the acquisition and tracking stage for a single channel.

8.1.4 System time differences

The method of obtaining the pseudorange, satellite position and the user position for the new signals are similar to the GPS L1 C/A case. However, in particular, for Galileo system the following important points were observed during the software development.

The time difference between the Galileo system time and the GPS system time that is transmitted as a part of the Galileo navigation message should be incorporated into the Galileo satellite position computation to have a single time reference (referenced to the GPS system time) for both the GPS and GIOVE satellite positions.

The integration period of the GPS L1 C/A corresponding to the measurement instant and the integration period of the GIOVE/Galileo E1 corresponding to the same measurement instant are different; in the current software design it is 1 ms and 4 ms respectively. This difference needs to be accounted for, when the receiver latches the measurements.

The start point (in time) of the signal tracking and data demodulation for the GPS L1 C/A can differ to the start point (in time) of the signal tracking and data demodulation for the Galileo/GIOVE E1B/C. In addition, as mentioned in point 2 above the integration times differ. Therefore, the solution computation process should align the measurement instant to the nearest GPS/Galileo sub-frame boundary.

In the current software design, the position computation algorithm treats GIOVE A/B and QZSS PRN-193 satellites as additional satellites to the GPS constellation.

8.2 PERFORMANCE ILLUSTRATIONS

As mentioned in section 2, limited to the availability of real navigation data for Galileo satellites, and ICD for COMPASS system, Spirent simulator data is used to test the acquisition and tracking model for Galileo system; while GLONASS is recorded using USRP2 platform by turning the IF centre frequency to the GLONASS satellite channels (e.g. PRN1). On the other hand, for the L1 channel raw signals having IF centre frequency at 1575.42MHz are recorded using Nordnav wideband receiver, whose front-end has a double-sided bandwidth of 10MHz.

8.2.1 The GUI

Data format

The recorded digitised raw IF data of USRP2 is in complex format; while the data format of Nordnav receiver is 8bit real. To facilitate the compatibility to various type of front-end hardware setting, the updated software receiver has enabled the option to choose data type as well as data format.

Satellite PRN options

The current software version is mainly focus on positioning integration of GIOVE, QZSS and GPS. GLONASS as an extendable option is included; however, only one channel is dedicated in this current version. By turning the IF frequency, the chosen GLONASS satellite could be acquired and tracked. An example of GUI setting for a set of GLONASS IF data recorded using USRP2 is shown in Figure 8-1.

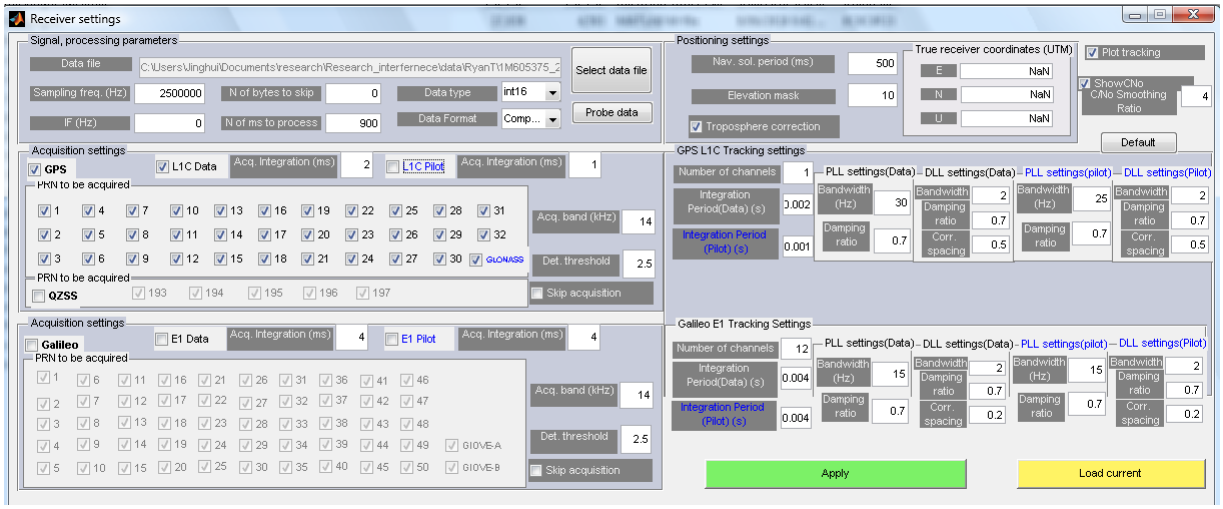


Figure 8-1. GUI interface for GLONASS PRN1, using USRP Front-end with sampling rate=2.5MHz, IF=0Hz

Check boxes for QZSS PRN 193-197, Galileo PRN 1-50 as well as GIOVE-A and GIOVE-B are added. L1C pilot for GPS section is currently unused and preserved for the future update.

Integration time

Integer number of integration time for acquisition section (in ms) could be filled in the gap on the left section of GUI; while tracking integration time (in ms) could be given on the right section. By default satellites included in GPS and QZSS sections use 1ms integration for acquisition and tracking.

Galileo satellite by defaulted is set to use 4ms integration at E1B channel and 4ms for E1C channel. If E1C channels of GIOVE-A or GIOVE-B are being acquiring and tracking, integration time should be set to be 8ms in this current software version.

C/No estimation

C/No estimation using two methods [4]: "average power" and "Narrowband against Wideband Power" are added and the plots could be chosen to be plotted from the GUI allowing different smoothing ratio (4 by default).

8.2.2 Acquisition module

In order to provide the acquisition plots for all of the GNSS except Galileo/GIOVE in one plot, the GLNASS is dedicated to PRN number 33 in the acquisition result plots shown in Figure 8-2. QZSS satellites are dedicated to PRN 34-38 in Figure 8-3.

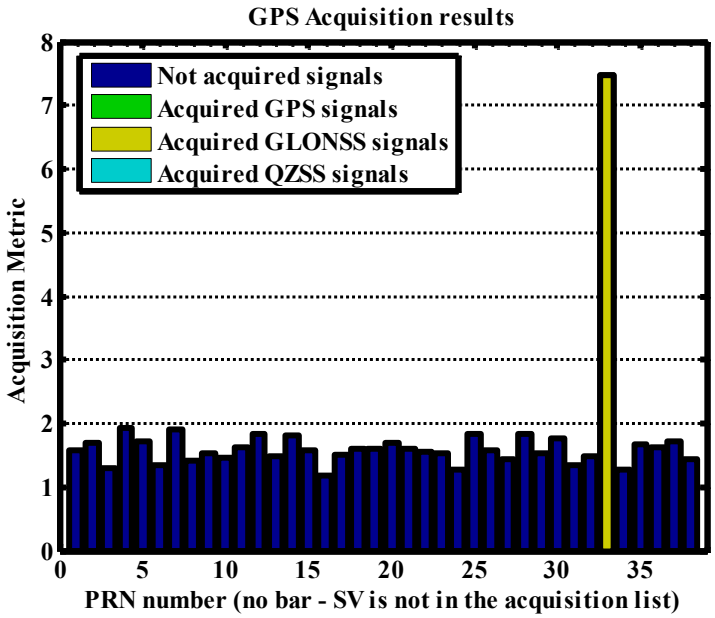


Figure 8-2. GLONASS Acquisition-recorded using USRP Front-end, Fs=2.5MHz, IF=0Hz

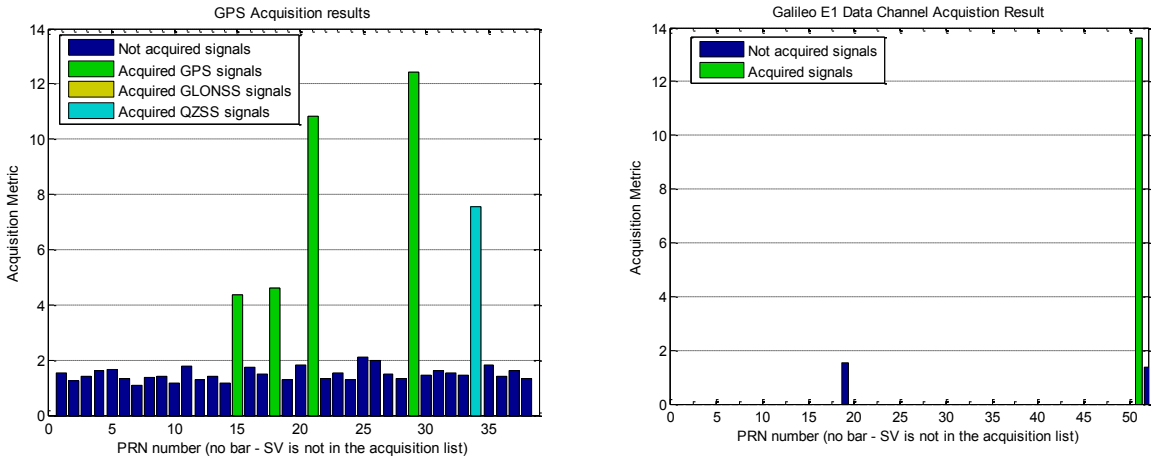


Figure 8-3. Acquisition results of GPS and QZSS (left), GIOVE-A (PRN 51) (right), recorded using Nordnav Front-end, with sampling rate 20Mhz, IF=5086667Hz

The acquisition results for real GIOVE-B satellite is shown in the right side of Figure 8-3. The Spirent simulated Galileo signals are also used for test. The acquisition results are shown in Figure 8-4.

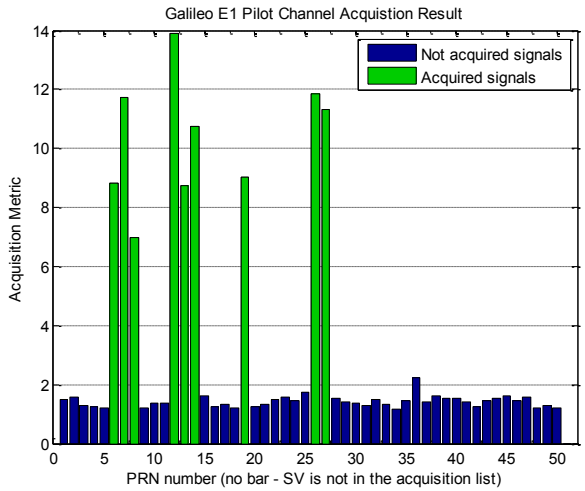


Figure 8-4. Acquisition results of Galileo signals simulated by Spirent, recorded using Nordnav Front-end, with sampling rate 20Mhz, IF=5086667Hz

8.2.3 Tracking module

The tracking results of the GLONSS PRN1 (refer to the acquisition result Figure 8-1) is given in Figure 8-5. Due to the higher noise floor of USRP2, the PLL tracking output is quite noisy; however, it still clearly shows the extracted navigation message bits. Two C/No estimation methods are used and the estimated C/No of about 43dBHz are shown on the bottom plots in Figure 8-5.

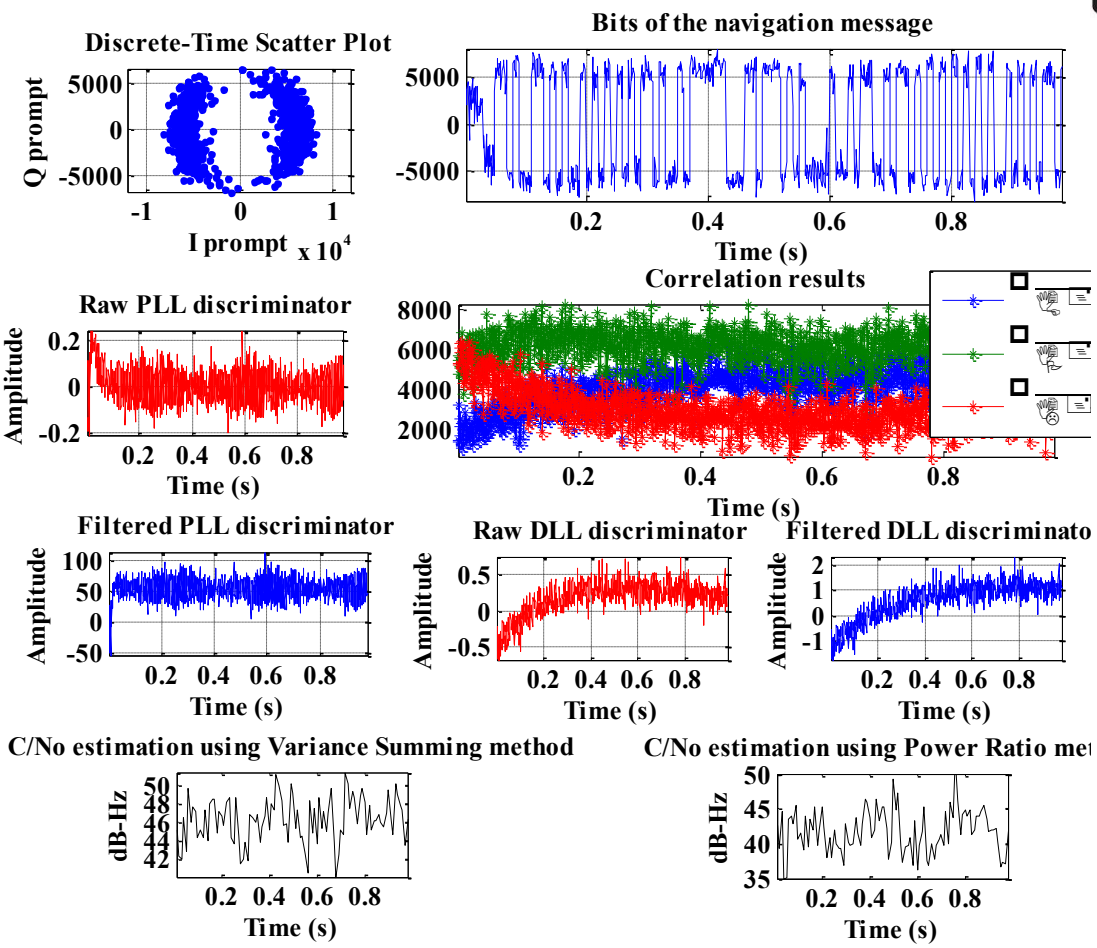
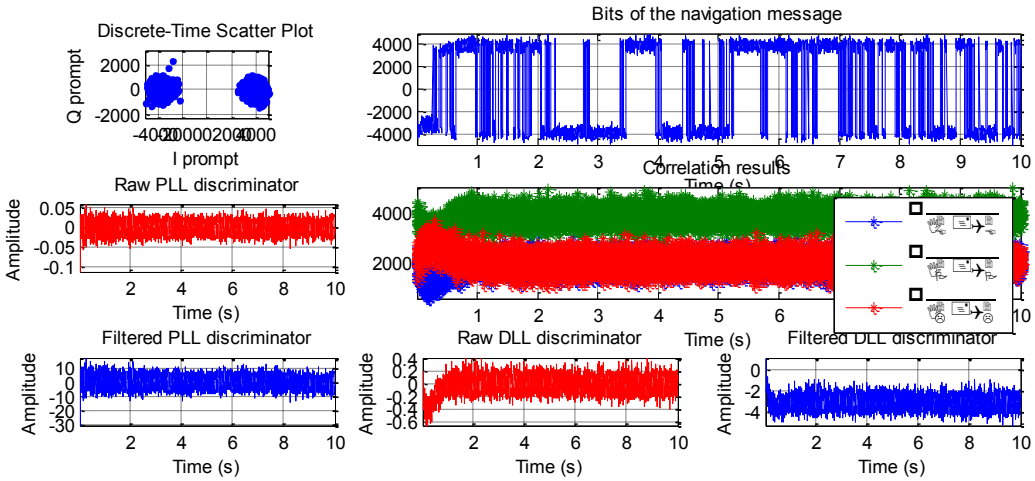
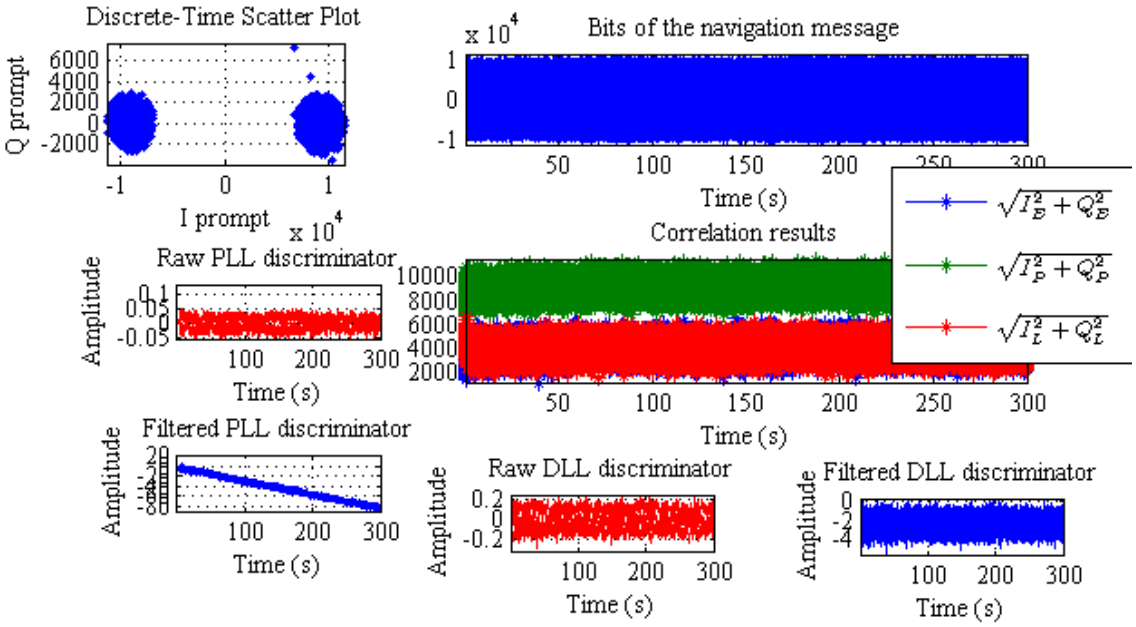


Figure 8-5. GLONASS Tracking -recorded using USRP Front-end, Fs=2.5MHz, IF=0Hz

The tracking results of QZSS PRN193 and GIOVE-A (refer to acquisitions results Figure 8-3) are shown in Figure 8-6b) (over view) and a) (zoom in version).



(a)



(b)

Figure 8-6. Tracking -recorded using Nordnav Front-end, Fs=2MHz, IF=5086667Hz

8.2.4 Positioning module

Using the IF data recorded from GPS, QZSS and GIOVE satellites in view, the positioning integration performance of three systems, GPS, QZSS and GIOVE-A/B were observed at the output of the navigation poisoning calculation module (using pseudorange measurements).

Though the GIOVE ICD does not recommend the use of GIOVE A and GIOVE B signals for position computation, an attempt was made to observe the positioning performance with combined GPS and GIOVE satellites. Nine types of position computation results are shown in this section with the help of a real-time dataset.

Dataset 1: Consists of 4 GPS satellites, the QZSS PRN193 and GIOVE-A satellite.
Scenario 1 (4GPS): 4 GPS satellites that are capable of producing the solution
Scenario 2 (4GPS+QZSS): 4 GPS satellites that are capable of producing the solution and the QZSS used as an additional satellite
Scenario 3 (4GPS+QZSS +GIOVEA): 4 GPS satellites with the GIOVE-A and QZSS used as two additional satellites

The following subsections provide the graphs depicting the result from the real signal. Each set of result contains the position spread along with the Skyplot.

Scenario 1

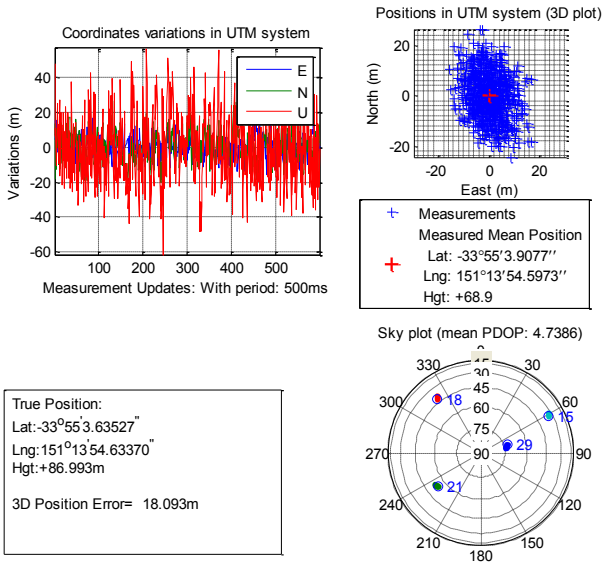


Figure 8-7. Scenario 3.1 (4GPS)

Scenario 2

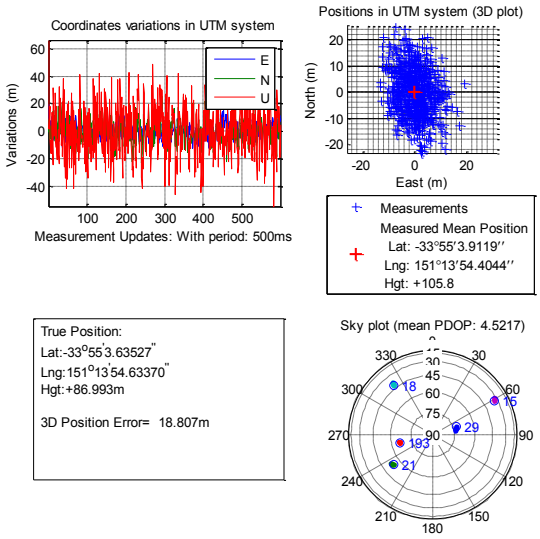


Figure 8-8 Scenario 3.2 (4GPS+QZSS)

Scenario 3

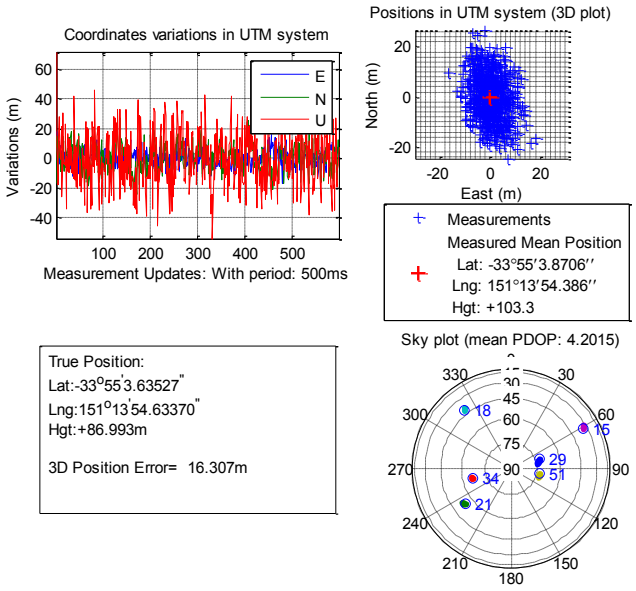


Figure 8-9. Scenario 3.3 (4GPS+QZSS +GIOVEA)

8.2.5 Inference

From the results, it is observed that the position solutions with GPS+GIOVE satellites or GPS+QZSS are degraded compared to the GPS only case. A part of this degradation is due to the increased DOP. It should also be noted that the GIOVE satellites are set "Unhealthy" and are not recommended to use in the position solution as per the GIOVE ICD. The combination of GPS+QZSS+GIOVE satellites provides improved signal availability and decreasing DOP. These results demonstrate the development of a multi-GNSS software receiver platform for the L1 band.

8.2.6 Real Galileo in Orbit Signal Processing

The Galileo Signal in Orbit has started transmitting test message for navigation purpose since March 2013. Raw IF data was recorded using Labsat receiver so that RF signal can be played back in analogy format. Signal was recorded again with Nordnav receiver. The Galileo only position result is shown in Fig 26.

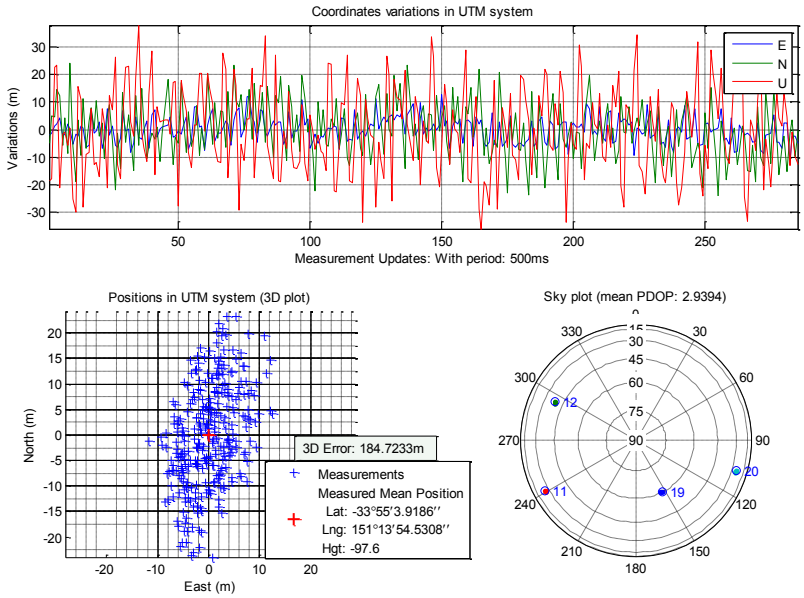


Figure 8-10 Real Galileo In Orbit Signal scenario (4Galieo: PRN 11 12 19 20)

8.3 GNSS Data Processing and Measurements Output in RINEX 3.0 Format

This section discusses the development of a Matlab-based multi-Global Navigation Satellite System (GNSS) software receiver. Compared to a real-time GNSS receiver, the major difference is in the implementation mechanism of latching and processing the measurements between the receiver channels of a satellite system as well as channels of different systems. Implementation challenges increase due to differences in the system time as well as the code repetition period. The signal processing modules, especially the tracking module of a conventional Matlab multi-GNSS software receiver, consume an enormous amount of time as IF data is read sequentially, requiring considerable file I/O operations. The proposed software receiver architecture uses the built-in parallel processing functions in Matlab to speed-up the execution by simultaneously processing multiple channels.

RTKLIB (version 2.4.1) is used to validate the pseudo-range and carrier measurements through position computation as it has the ability to process RINEX 3.00 format. However, this computed solution cannot be compared with the software receiver generated solution as the current version of RTKLIB cannot process GIOVE data. These results have been compared with the truth data which have been produced by the Spirent GNSS simulator.

Considering the computation burden and efficiency of processing multiple satellite signals, parallel processing has been implemented in the current software receiver. The aim of this development is to improve the computational efficiency and allow for the generation of carrier-phase measurement output in the latest version of the RINEX 3.0 format.

This section is organised as follows: the overall processing structure a GNSS software receiver is given in section 2; the functionality of the software modules (including acquisition, tracking and positioning) are explained in section 3 where the improved processing architecture is described. Receiver performance and efficiency improvements are presented in section 4.

8.3.1 Software Receiver Components

In this report, the testing platform for the 'Matlab-based software receiver' consists of three main components: Data collection, Matlab-based Software receiver, and Measurement outputs, as illustrated in Figure 8-11.

Data collection comprises three steps: RF simulator (with Spirent GS8000) configuration, RF raw digitised data recording (using Nordnav wideband receiver), and Digitised data format conversion to the Matlab readable format.

Software Receiver has the basic functionality of signal acquisition for various GNSS signals at L1 band, Multiple Channel tracking, and finally Navigation solution calculation for integrated multi-GNSS. This paper focuses on the integration solution of GPS and Galileo systems.

Measurement output including the pseudorange and carrier-phase are produced in RINEX 3.0 format by the software receiver. Integrated navigation solution is then verified using the receiver generated RINEX files via the post-processing software RTKLIB [5].

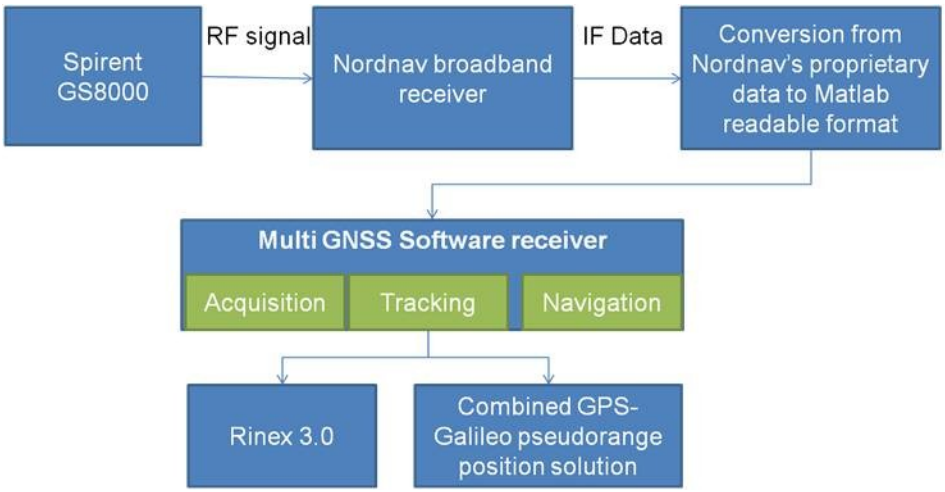


Figure 8-11: Receiver test setup

A multi-GNSS software receiver design has been described in last section. This software has enabled the functionality to process GPS, Galileo, QZSS as well as GLONASS signals at L1 band. The data processing flow for GPS and Galileo signal integration is shown in Figure 8-12. In the conventional software version, the acquisition and tracking for each GNSS at each channel are processed in a sequential fashion. This can be very computationally demanding for a standard PC. Therefore parallel processing strategy is used for current software receiver version.

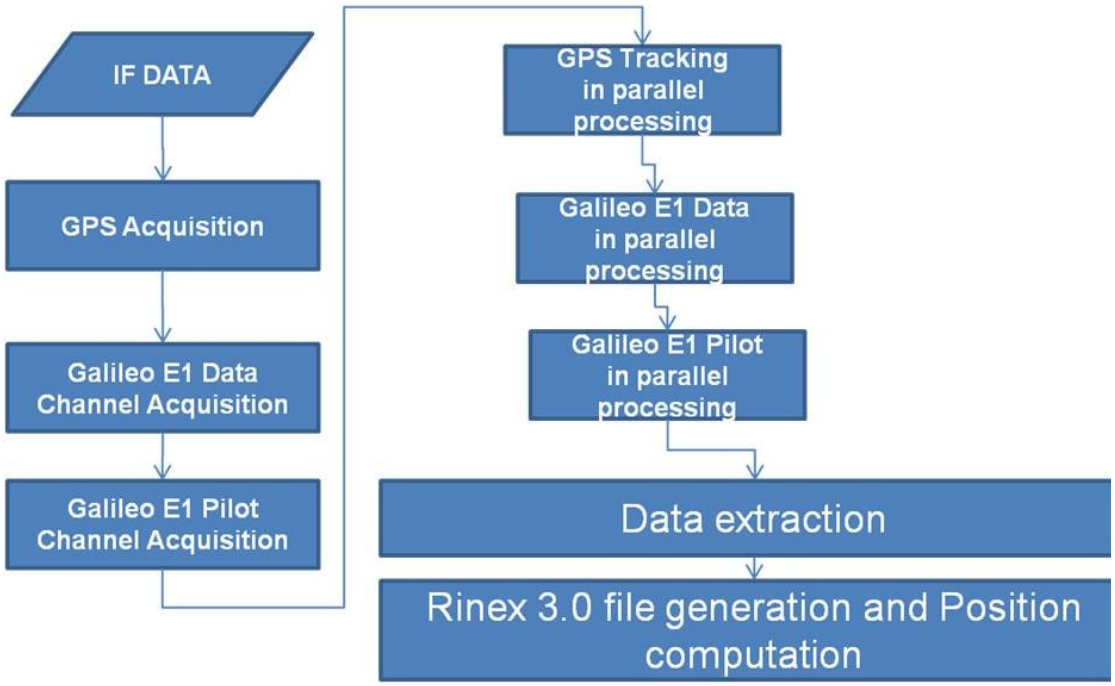


Figure 8-12: Multi-GNSS software receiver data flow

8.3.2 Test setup

To validate the software receiver several tests were conducted. Block diagram of the test setup is shown in Figure 8-13. The Spirent GNSS simulator (model GSS8800) is used to generate five minutes of RF signals of GPS and Galileo satellites, which were collected using the Nordnav wideband receiver. The simulator was configured to produce static data. This RF signal was converted to IF data, which are subsequently used by the software receiver for acquisition, tracking and position computation.

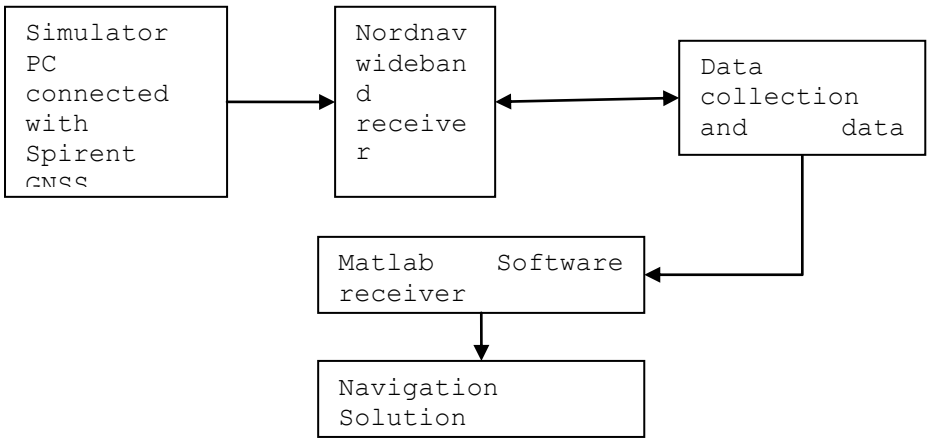


Figure 8-13: Block diagram of test setup

8.4 Results

8.4.1 Parallel processing

Parallel processing provides a significant advantage. Different tests during this study have confirmed that an average of 19±0.1 megabyte data is stored for per second of IF data, i.e. 1.2 gigabyte per minute. Such an enormous amount of data requires special attention if processing times are to be improved. A 'snapshot' of the 8 core processor resource use for both sequential and parallel processing are shown in Figure 8-14, where it can be observed that parallel processing improves the computer processer's usage level. Sequential processing process data sequentially which means at any given time all the computer processor is involved processing one set of data. On the other hand, parallel processing can be achieved through distributing data sets to individual processors. This parallel process maximizes the use of computer recourses which can also be observed by comparing Figure 8-14(a) and Figure 8-14(b). This improvement helps to reduce the processing times of IF data from 14 hours to 30 minutes for a 4 minute and 10 second long sequence of IF data. That is a 96% reduction in processing times.

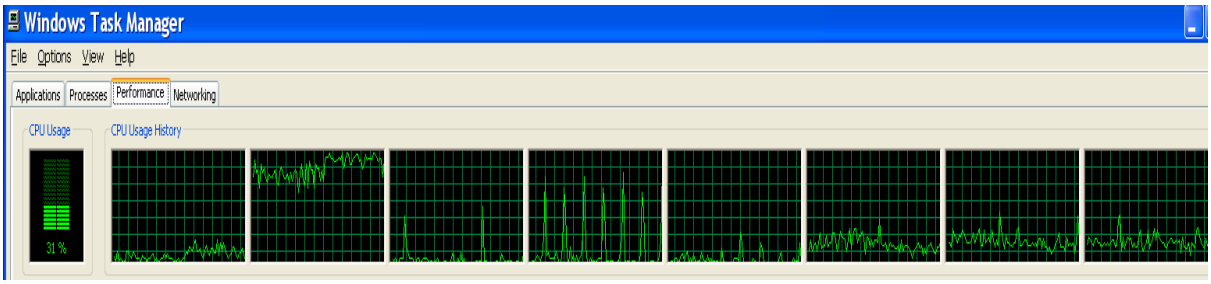


Figure 8-14 (a): Sequential processing resource use

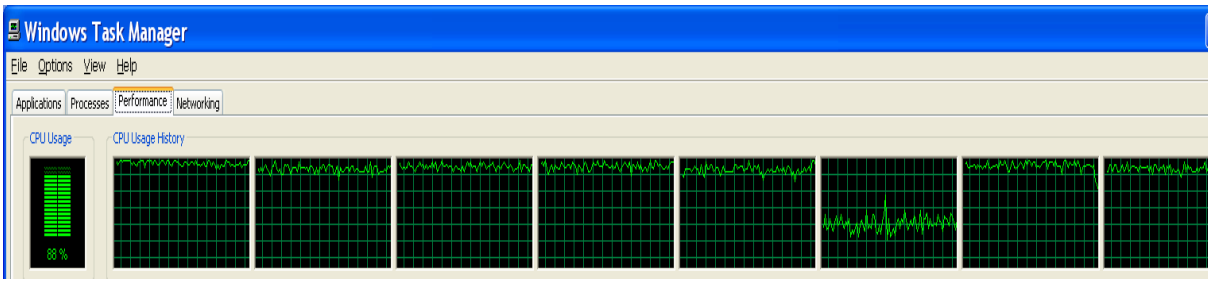


Figure 35 (b): Parallel processing resource use

8.4.2 Acquisition

Acquisition results for GPS and Galileo are presented in Figure 8-15and Figure 8-16. It can be observed that the Acquisition module acquires seven GPS satellites and eight Galileo satellites for this test scenario. The number of tracked satellites is sufficient for computing the position solution and provides a better than average Dilution of Precision (DOP).

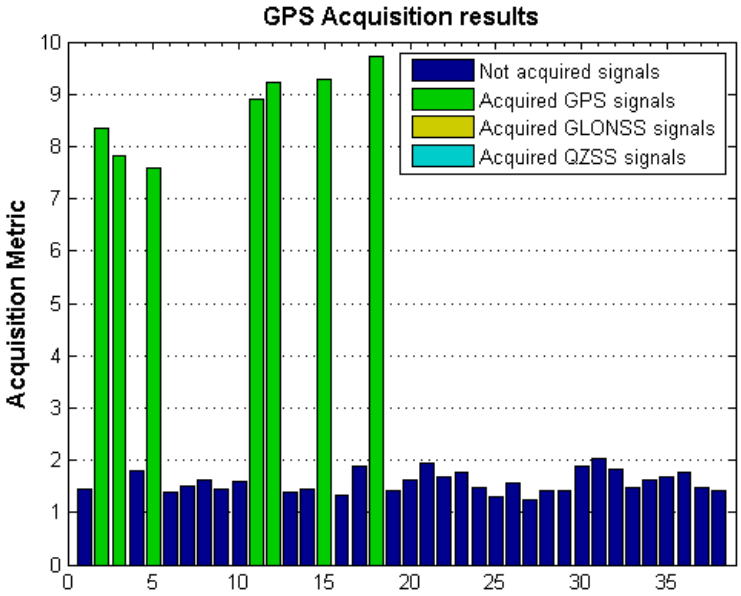


Figure 8-15GPS signal acquisition

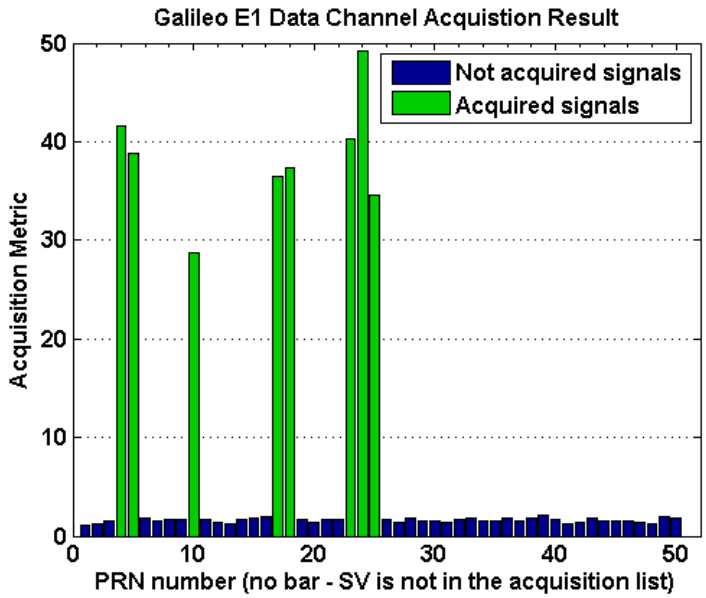


Figure 8-16Galileo signal acquisition

8.4.3 Tracking - Rinex 3.0 output

Output of the Tracking and Data Extraction is via files in the RINEX 3.0 format. Figure 8-17 shows a sample RINEX 3.0 output of pseudorange and carrier-phase measurements. Details of the RINEX 3.0 format can be found in [6].

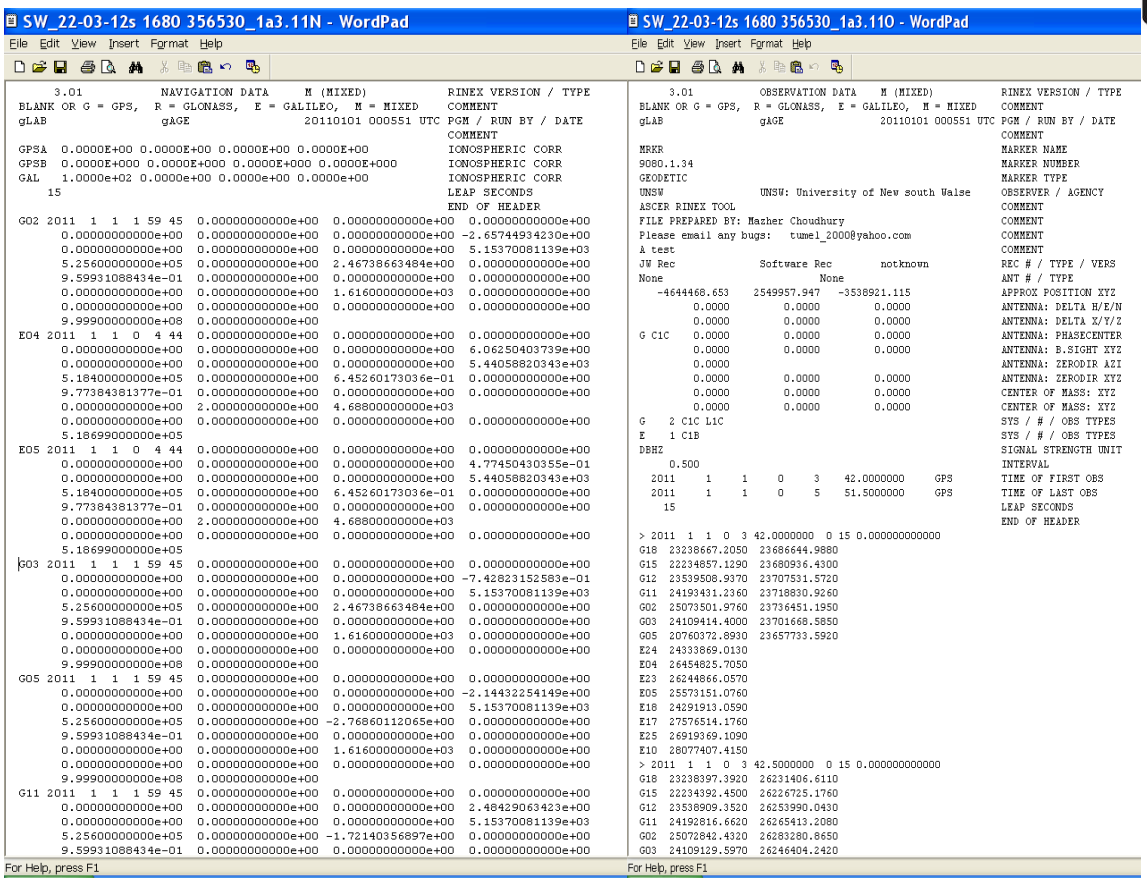


Figure 8-17 (a): Navigation data

Figure 38(b): Observation data

8.4.4 Position test

RTKLIB (version 2.4.1) was used to validate the pseudorange through position computation using the RINEX 3.0 measurement files. However, this computed solution cannot be compared with the software receiver generated solution as the current version of RTKLIB cannot process Galileo data. At this stage, RTKLIB can only process the output GPS and QZSS L1 C/A carrier-phase data.

RTKLIB's processing capability can validate the RINEX 3.0 output formatting. In this test, a 4 minute and 10 second long period of IF data was processed. Figure 8-18 shows RTKLIB's pseudorange-based position solution.

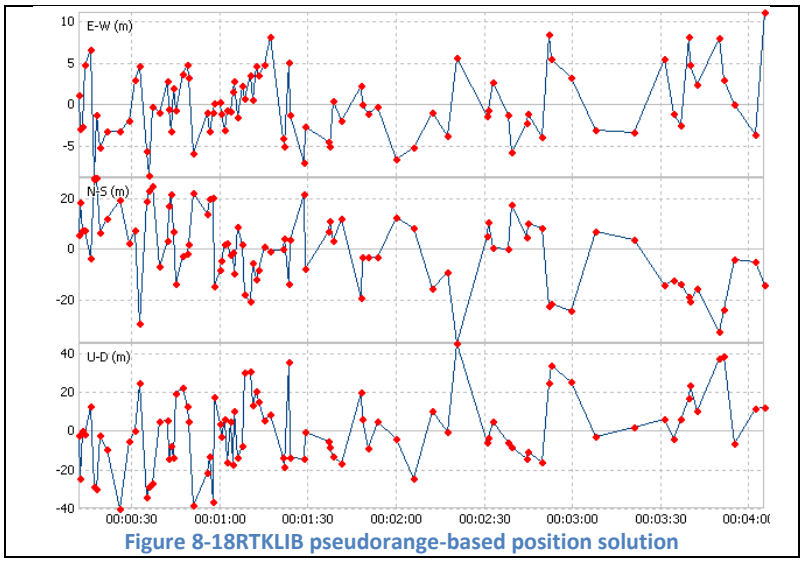


Figure 8-18RTKLIB pseudorange-based position solution

Figure 8-19 shows the difference between the truth and GPS-only position solution as well as difference between the truth and GPS+Galileo combined solution. Table 8-2 is a comparison between the two position solutions. It can be observed that the software receiver is capable of providing position solution at an accuracy of ±10m for a static scenario. However, the GPS-only position solution provides better accuracy than either the Galileo-only or the combined solution. On the other hand the combined solution is more precise. It is to be noted that in the combined solution inter-GNSS system bias or receiver bias was not accounted for. At the time of writing this paper the statistical validation process was still under development. Statistical validation is expected to improve position solution by rejecting data outliers.

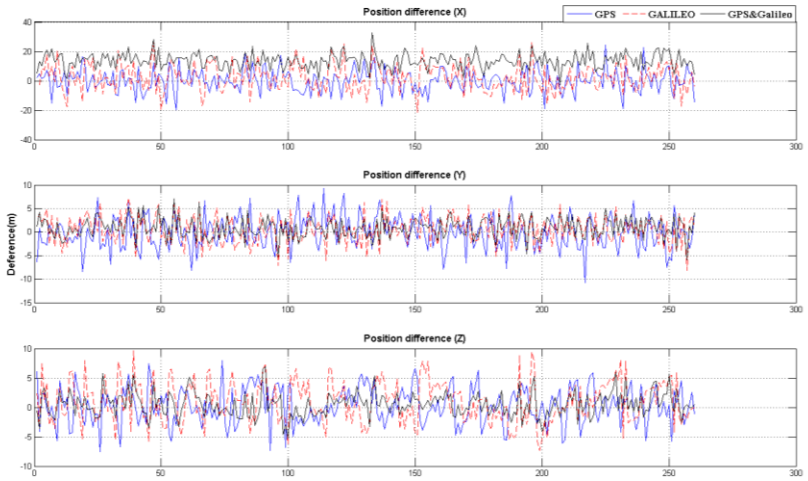


Figure 8-19. Position solution difference

Figure 8-20 shows the comparison between the positions generated by RTKLIB and Matlab-based software receiver. It can be observed that position solutions do not match. As Matlab-based software receiver does not employ any statistical validation process, position solution might have biases.

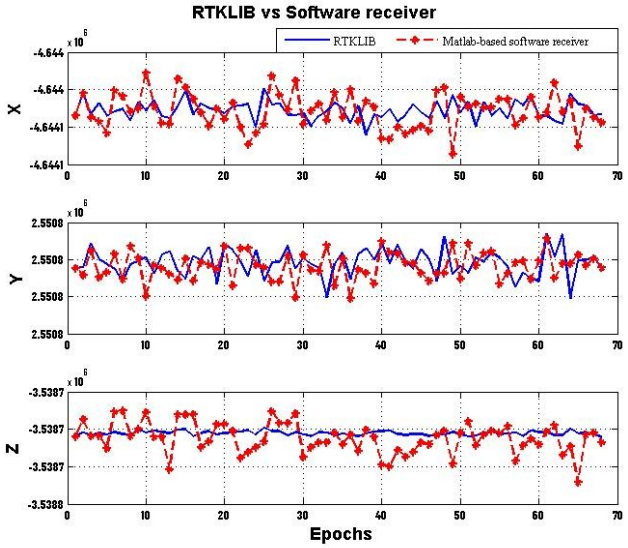


Figure 8-20Position solution comparison

Table 8-2 Position solution comparison

	GPS-only(m)	Galileo-only(m)	GPS+Galileo(m)
X	0.15±8	2±9	1.4±5
Y	-0.09±4	0.5±3	0.4±2
Z	0.32±3	1±4	0.37±2
3D	1.85±3	9±5	5±3
PDOP	1.96	2.3	1.39
HDOP	1.84	2.1	1.3
VDOP	0.67	0.8	0.49

8.5 Concluding remarks

This section described a Matlab Based L1/E1 multi-GNSS software receiver development for the algorithm verification of future multi-GNSS navigation system in the firmware development for GARADA project. An attempt was made to compute the position solution from the real data collected when the GIOVE satellites and QZSS were visible along with the GPS satellites. Position solution computation with the combined GPS, Galileo and QZSS pseudorange measurements was demonstrated. This study focused on performance verification improvement, in terms of GNSS signal processing, data processing and output format, of a multi-GNSS receiver at the L1 frequency band. Parallel processing has been implemented in the software receiver to improve computational efficiency and to make it feasible to produce carrier-phase as well as pseudorange measurement output in the RINEX 3.0 format. Results show that data processing performance is improved by 96%. In addition, the considered test case exhibits a 3D accuracy of 5±3m with the combined GPS and Galileo solution. Future improvement involves integration of statistical validation of pseudorange measurements, outlier detection in the Position module, and taking into account inter-GNSS biases.

9 Firmware development and debugging

9.1 Galileo E1 Firmware Architecture Overview

The Galileo E1 firmware design follows the fashion of the “Aquarius” GPS receiver software structure, designed for Biarri project. Regarding the software functionality, the software includes 7 major sections: Tracking Loop, Navigation Data Decoding, Navigation Engine, Satellite Database, Timing, User Interface, and RTOS. The firmware development efforts for Galileo E1 mainly focuses on the design and testing as described in section 9.1- **Error! Reference source not found.** and 9.6. An overview of the Galileo E1 Firmware Structure and Report/Debug Interfaces is shown in **Error! Reference source not found.**Figure 9-1.

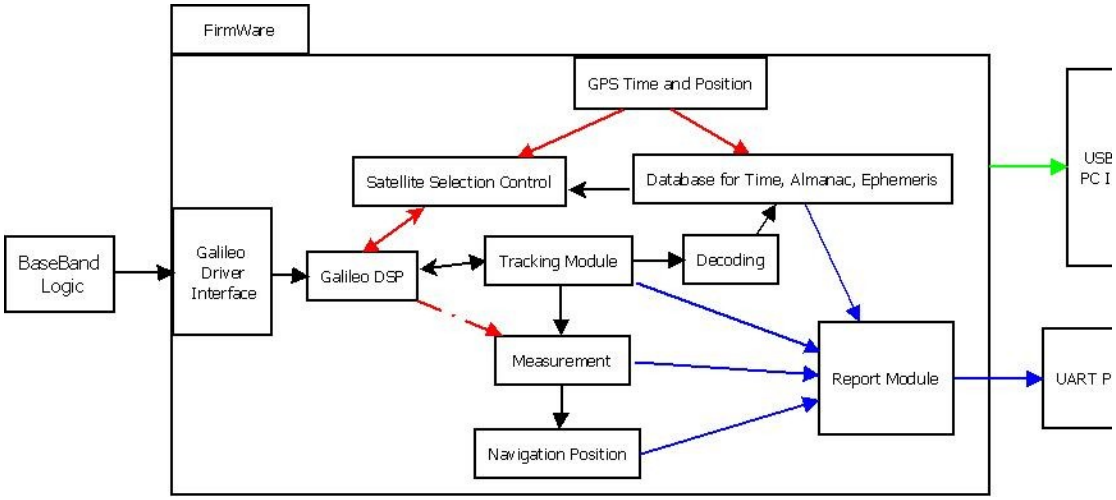


Figure 9-1 Garada Namuru Receiver E1 Firmware Structure and Report/Debug Interfaces

The Tracking Loop of the firmware section includes major modifications and requires significant effort during the Galileo Integration development process. It includes,

1) The Galileo Driver interface that matches to the baseband address map given in section 7: Baseband Logic Design.

2) The Galileo PreDSP obtaining the coherent integration dumps sample from the defined address of baseband. It also performs high-priority process controlling the running of multiple Galileo Channel assignment so that the Tracking Module can perform closed loop PLL/DLL or open-loop FFT trackers. Moreover, it also plays a high level controlling role in determining the latch events by comparing the accumulated Galileo code phase and carrier phase software epoch counter during tracking process with the hardware epoch counter accumulated since start of receiver operation, so as to synchronise the receiver clock with the tracked signal for measurement generation.

3) Galileo Tracking Loop Module performs most of the GNSS signal processing algorithm of signal detection, tracking as described in section 8 of this report. The “Search Mode”, “Capture Mode” and “Track Mode” has been designed following the fashion of GPS tracking module. However, Since Galileo has both pilot channel and data channel. The closed loop tracking can be performed with either data or pilot signals controlling by a Marco selection in this firmware. In the current version, it has been chosen to use data channel as default option for closed loop tracking while open loop has been used for pilot channel tracking so that it could aid the identification of 100ms secondary code boundary for resolving the measurement ambiguity.

Beyond the need of major change in Navigation Data Decoding section of the firmware, the Galileo signal processing requires a unique database for the storage of Almanac and Ephemeris. Obtaining these two structures of navigation message is critical for a robust and speedy visible satellite selection and Doppler prediction, especially in space mode due to high vehicle dynamic.

Integrated Navigation Position calculation can then be performed by taking the pseudorange, carrier measurements from the tracking module as well as the stored ephemeris messages from the Galileo database. On the other hand, the calculated vehicle position can work out the vehicle orbit so that it could predict the visible GNSS satellite together with the GPS time and Galileo Almanac/Ephemeris.

There were two types of PC interface for the receiver performance report and debugging. The selected results including the channel assignment and tracking status, observable measurements, satellite selection and position calculations can all be streamed out using UART interface defined in section 0: Namuru Receiver Command and Reports.

However, when it comes to debugging during the development phase, UART streaming is too slow for showing the real-time processing behaviour. USB debug described in section 9.13: Debugging Methodologies has been used for this purpose capturing the change of variable status in real time in a relatively high volume at a high capturing frequency.

9.2 Software Version Control

During the Garada firmware development, two Namuru FPGA hardware versions have been used in different phase of the firmware development. Different hardware version requires different hardware interface configurations for the "Aquarius" GNSS software. Therefore version control on the software is necessary so that a unified generic firmware version can be applied on either Namuru V2.4 or Namuru3.3. At the current stage, the Namuru 3.3 version does not support the USB debugging PC interface. Namuru 3.3 uses the same processor as the Namuru 2.4 except that Namuru 3.3 can support both L1 and L5 front-end. Also Namuru 3.3 has wider L1/E1 front-end bandwidth for the need of Galileo E1(OS) signal, since it requires at least 4Mhz double-sided bandwidth to accommodate the main power of the BOC(1,1) signal. However, we have been found that, with Namuru 2.4, despite there is about 3dB signal lose due to having narrow front-end bandwidth (i.e. 2MHz), the detected signal to noise ratio is still sufficient strong for the Galileo receiver development. Therefore, without loss of generality, unless it has been specially pointed out, the report on firmware development to the "subset" functionality design for the firmware development using Galileo receiver in Namuru 2.4 is also applicable to Namuru 3.3.

9.3 Galileo Firmware-Baseband Interface

The firmware interfaces with the baseband processes via a memory map interface as described in section 7**.** This interfacing is handled by the firmware via the GPS and Galileo driver module. It updates and transfers bits from the baseband memory map into firmware's data structures and vice versa.

This firmware interface is customised for various basebands. Hence, the GPS and the Galileo system each have its own driver module. There will also be a different driver module customised for various hardware platforms. In the course of this project, customised driver modules have been developed for Namuru V2.4 and Namuru V3.3 for both the GPS and the Galileo system.

9.4 Galileo Signal Processing Module

The signal processing module is used to acquire and track the GPS and Galileo signal such that the signals remains continually demodulated after the signal has been successfully detected. Figure 9-2 illustrates the simplified hierarchy of the firmware tasks. The blue shade indicates firmware tasks, while the red shade indicates processes residing in the baseband. Note that lower level tasks are given higher priority. The signal processing relevant tasks are the Tracking Loops task and the Signal Conditioning task.

<table>
<tr><td colspan="3">Higher Level tasks: Satellite Selection, Positioning Engine, etc.</td></tr>
<tr><td>GPS Tracking Loops</td><td rowspan="2">GPS + Galileo Measurement Task</td><td>Galileo Tracking Loops</td></tr>
<tr><td>GPS Signal Conditioning</td><td>Galileo Signal Conditioning</td></tr>
<tr><td colspan="2">GPS Driver</td><td>Galileo Driver</td></tr>
<tr><td colspan="2">L1 Baseband</td><td>E1 Baseband</td></tr>
</table>

Figure 9-2. Firmware Hierarchy of various tasks shown in ascending priority and descending level from top to bottom.

The Signal Conditioning task is used to coherently or non-coherently accumulate the baseband-processed samples to improve tracking sensitivity or adapt to signal dynamics by increasing or decreasing the integration period, respectively. In the current implementation, the integration period used for Galileo and GPS is 4 milliseconds and 2 milliseconds, respectively. This has been shown to be appropriate for on-orbit scenarios which have significant signal dynamics.

The normalised correlation amplitude of Galileo E1 signal as observed using Namuru V2.4 is shown in Figure 9-3**.** This correlation shape is theoretically expected for Galileo E1 and is the first indicator of a functional Galileo E1 baseband. Despite of having 3dB signal loss due to using 2Mhz double sided front-end bandwidth, the correlated signal components forms a strong peak at zero chip shift while uncorrelated signal components such as Gaussian white noise produce significantly lower correlation amplitudes beyond +/-1 chip.

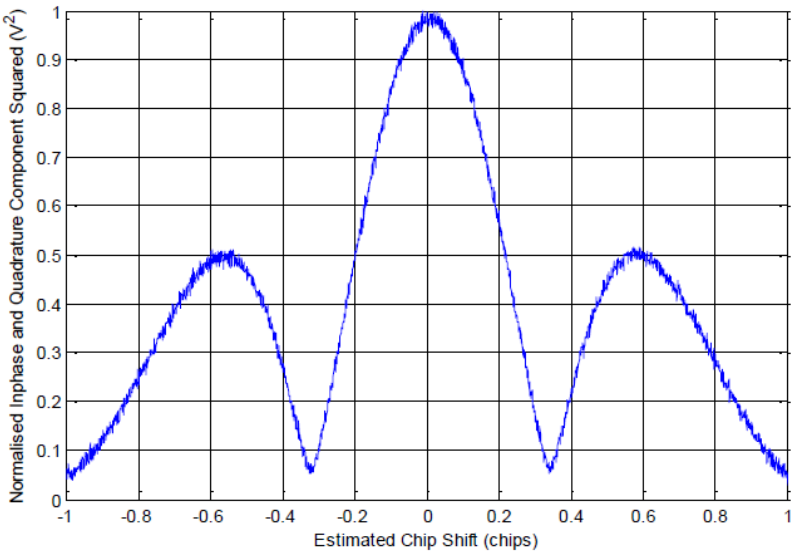


Figure 9-3. The Normalised Correlation Values collected from USB Debug

In signal acquisition for a particular navigation satellite's (SV) signal, the correlation amplitude of all possible code phases (i.e. chip shifts) and Doppler frequencies are compared against a threshold to determine if a signal has been detected. From the signal acquisition perspective, Galileo E1 differs from GPS L1 mainly by the length of its PRN code and the resolution of the code search. The Galileo E1-B and E1-C signal has four times longer code length and requires twice the resolution of the L1 code search. Thus, this acquisition process can be eight times longer than a typical GPS L1 signal if the same three-finger Early-Prompt-Late (EPL) correlator structure is used for Galileo E1. This significantly lengthened acquisition time can jeopardise the ability of the receiver to acquire the Galileo E1 in on-orbit scenarios where SVs in-view appear and disappear rapidly. To overcome this

problem, an eight-finger correlator structure has been implemented for Galileo E1 signal acquisition. This has proven to yield successful Galileo E1 acquisition in on-orbit scenarios.

Immediately after a successful acquisition, a following capture procedure is executed to smoothly transition into tracking mode. In the capture process, the firmware attempts to align the prompt finger of the correlator such that it is tracking the peak correlation amplitude and a fine frequency search is done to minimise the occurrence of false frequency locks and tracking loop loss of lock.

Figure 9-4 shows the time-domain plots of various indicators for a successful acquisition of six Galileo SV channels in on-orbit scenario. The different rows represent different SVs while the columns are indicators of various parameters. The Acquisition/Tracking column shows the firmware transitioning the channels from acquisition mode (indicated by 1) to capture mode (indicated by 2) and then to tracking mode (indicated by 3). The successful tracking of the signals are verified by the relatively higher Prompt correlation values that are sustained as the signal remained tracked.

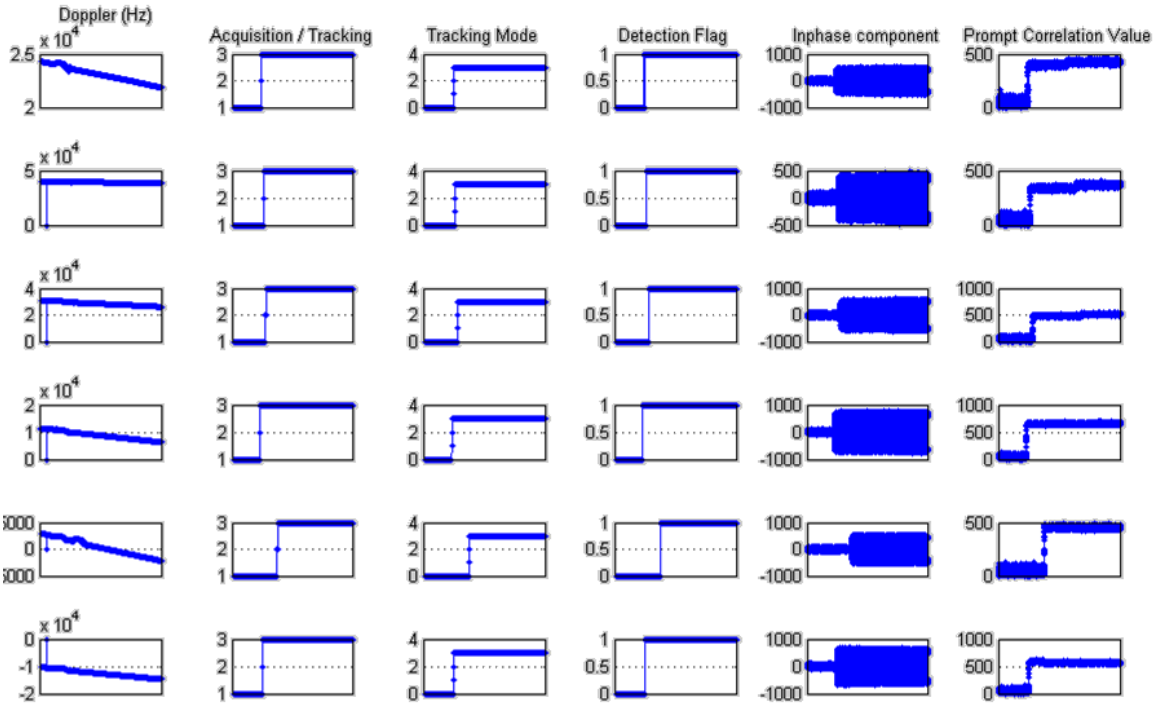


Figure 9-4. USB Debugger output showing successful acquisition and tracking.

There are three types of tracking loops implemented in Aquarius. At the beginning of tracking, the tracking loop operates as a first-order Frequency Locked Loop (FLL) and slowly transitions into a hybrid second-order Phase Locked Loop (PLL) first-order FLL and finally into a hybrid third-order PLL second-order FLL.

Recall that the E1-B (i.e. the Data channel) signal contains the navigation data bits while its orthogonal counterpart E1-C (i.e. the Pilot channel) signal contains the secondary code. This implementation allows the tracking loop to be fed back according to E1-B or E1-C correlation values. However, using either of the signals will produce the same result because both E1-B and E1-C are synchronous for a given SV.

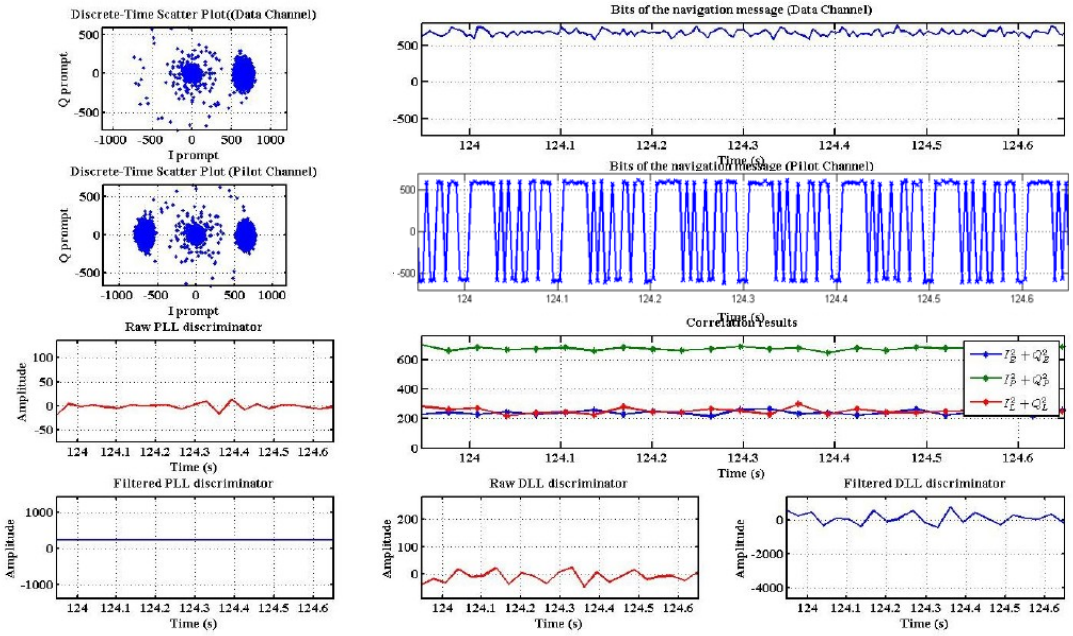
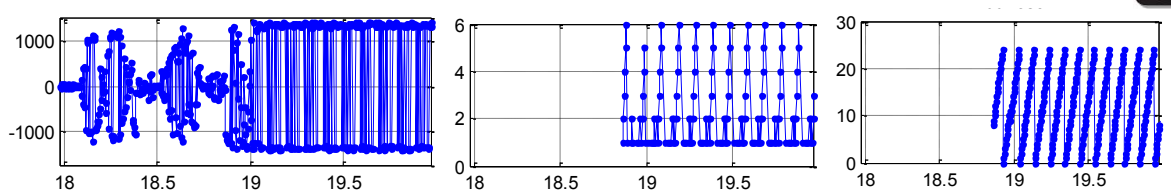


Figure 9-5. Zoom-in Version of Data and Pilot Chanel Tracking Outputs (A single channel mode)

9.5 Galileo E1-C Secondary Code Tracking

A dedicated counter that resides in the baseband and a software counter are used in tandem to keep track of the secondary code. Galileo Pilot Chanel has modulated the 4ms pseudorandom repeating code with a navigation liked secondary code. The difference between the navigation data at data channel bit and the secondary cod at the pilot channel is that, the secondary code has a repeating rate of 100ms. It has a fixed pattern of "0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 1 0". Unlike GPS' navigation message bits which have a bit rate of 50bps, each bit has a period of 20ms, the Galileo navigation message as well as the secondary code has a bit rate of 250bps., each bit has a period of 4ms. With this fixed secondary code, even without the need of proper decoding and message synchronisation, the 100ms boundary of the Galileo signal can be found right after tracking of secondary code which can help solving the ambiguity of pseudorange measurement 5 times faster literarily.

In order to find the alignment of 100ms long secondary code sequence, the firmware software does not implement with a normal correlation for synchronisation. Instead, a more efficient and simple matching detector has been developed to utilise the unique pattern of the secondary code sequence (the continuous seven zero digits from the 6th bit-12th bit. Figure 9-6(a) shows the inphase component of the pilot channel. From this figure, the repetitive pattern of the secondary code is evident after 19 seconds where the signal has been successfully tracked. Figure 9-6 (b) is the indicator of the secondary code tracking where 1 to 6 will indicate that a match has been detected and no value indicates that there is no match detected. Figure 9-6 (c) shows the implied phase of the secondary code phase maintained by the PreDsp which linearly increases over time and rolls over at the 25th chip. Subsequent to the successful tracking of the secondary code, an unambiguous pseudorange can be deduced from both the secondary code and primary code phase.



(a) (b) (c)

Figure 9-6. Secondary Code Synchronisation on the Pilot Channel (E1-c)

As a sidenote, several features have been added into the signal processing module for enhanced reliability. A Tong detector is implemented at the acquisition stage to reduce the occurrence of false detection. An indicator formulated based on the phase of the secondary code sequence has been used to detect false frequency locks. In the original "Aquarius" GPS receiver, the false frequency tracking is detected by applying a FIR smooth filter on the extracted navigation message bits. Each GPS navigation bit can last for 20 samples if the 1ms integration dump is used. Whenever there is phase rotation on the navigation bit, the FIR smooth filter can smooth out this occasion, resulting constant zero signal throughout the stabilised tracking.

However for Galileo, both navigation message and the secondary code bit last for 4ms resulting only 1sample for each 4ms integration dump. It means that the navigation bits for each integration dump can vary a lot. Applying the original FIR smooth filter on this extracted Galileo navigation message no longer has valid meaning regarding to the indication of frequency tracking behaviour. In the Galileo tracking module, the 180 degree phase rotation (i.e. sign) of the whole secondary code sequence is then used as the input to the FIR smooth filter. If the secondary code sequence can repeat its pattern continuously without sign change, it is a good indication of good frequency tracking.

9.6 Galileo Measurement Processing Module

This section describes the operation of the module that computes the pseudorange and carrier phase measurements that are necessary for position computation.

9.6.1 Pseudorange Measurement

In order to infer the pseudorange from a tracked Galileo E1 signal, the phase of the primary code and the secondary code at a particular time instance is simultaneously latched to be processed. The same is done for GPS L1 primary code. Recall that GPS L1 does not have a secondary code.

Galileo E1's primary and secondary code combined is 100ms long. Thus, there is 100ms worth of unambiguous ranging code which equates to approximately 30,000km of unambiguous range. Unlike GPS L1 which requires data decoding to compute an unambiguous pseudorange, Galileo E1 does not require any data decoding to produce an unambiguous pseudorange due to the adoption of a secondary code.

9.6.2 Carrier Phase Measurement

The firmware performs the accumulation of carrier phase as required. However, various adjustments to the accumulations and consistency checks are performed in order to ensure smooth and reliable carrier phase outputs.

Carrier phase measurements obtained from the baseband are ambiguous to 180° phase rotation at the beginning of signal tracking. This is resolved using the Preamble bits in the case of GPS L1. In the case of Galileo E1, the secondary code is used to disambiguate it instead.

9.7 Satellite selection

In order to assign the satellite PRN number into the available Galileo processing channels, the satellite selection module play the key roles in determining the suitable search mode so that the right satellite PRN code can be predicted and assigned to each channel for fast acquisition and then tracking so as to generate position measurement. The firmware currently supports six search modes. Out of those six, four are the mostly commonly used mode while the receiver is on flight. The rest of two modes are designed for manual searching enabling the debugging feature during development. The most commonly used search modes include:

- Initial searching mode is used during the receiver start up. It clears out the entire SV satellite selection and thereby effectively shuts down the receiver. The associated acquisition assistance is updated with this zeroed out selection.

- Sky search mode is used when the receiver is in cold start where neither the GPS/GNSS time, user position nor the complete almanac/ephemeris is available for visible satellite prediction. The sky search process involves searching for the entire satellite constellation by means of a blind search. All of the hardware channels are loaded up with a different satellite and the search parameters set so as to cover a wide Doppler range of -10 kHz to +10 kHz. When the searches terminate, the tracker task signals the satellite selection indicating the channel that has concluded its search without having found any satellite. That satellite is then removed from the current set and the next available satellite substituted in its place. Currently, the search range of Galileo PRN number is from 1-50, although the defined hardware memory PRN code for Galileo is only 1-30, the actual entire Galileo satellite constellation only content PRN code 1-27.

- Sky Search with Almanac/Ephemeris mode is used when the either ephemeris or almanac is available for warm/hot start. The different between warm and hot start is only determined by how long a receiver tracking channel was interrupted before re-acquisition. After a very short term of interruption (i.e. 6-8second typically),the Doppler of a tracking channel would drift away too much for performing re-acquisition process in hot start to search the code phase and carrier phase around from the last recorded tracking states. At this circumstance, warm start has to be performing with the acquisition assistant message obtained from this satellite selection mode.

Acquisition assistance needs to be continually maintained for the each of the satellites being tracked because once positioning, the presence of the acquisition assistance can assist in speeding up satellite acquisition. When the receiver is working on the space orbit mode, the acquisition assistance is not only need to predict the visible satellite in high speed, but also required to predict the space parameters including code and carrier phase uncertainty for the need of warm and hot start. The search parameters for the search are chosen to be sufficiently large so as to ensure that unknown satellite Doppler due to user velocity and TCXO drift, as well as unknown code phase due to unknown user position is properly accounted for.

The hardware channels are required to indicate to the satellite selection when the pre-assigned search space parameters have been completely searched. Therefore a signal task has been assigned at the end of searching completion for a particular channel. For GPS satellite signal processing, this task signal is triggered only when it is used in the sky search mode. Once almanac/ephemeris is downloaded, each receiver channel will be assigned to search for the predicted satellite parameters continuously until new assisted message available to alter the parameter selections. This is needed for satellite selection which needs to know when to move onto the next satellite. In the new firmware developed for Galileo signal tracking, this feature has been upgraded for Galileo channel assignment so that once a channel has failed to acquire the predicted satellite, no matter it is in sky search mode or sky mode with the availability of ephemeris/almanac, the completed searching signal can always be triggered so that the satellite selection module can update the visible satellite parameters prediction more often to suite the need of Galileo signal acquisition which take much long searching period than GPS.

During the sky search operation, it is desirable to ensure that when the transition to normal mode occurs that all of the sets required by normal mode are setup correctly. This means that the channel assignment states and the list of predicted visible satellites need to be properly setup as part of the sky search process. One way to improve the probability of finding at least one visible satellite early in the sky search process is to select satellites based on prior knowledge of the satellites orbits. For example, rather than simply searching through the set in sequential order, satellites may be selected based on the constellation at a particular time, where we first select satellites that all lie in the same orbital plane. This should result in a spread of satellites from a positional point of view resulting in at least one being detected. However at the current firmware version, this optimisation sky search strategy has not yet implemented. Hence, the Galileo channel assignment is based on a sequential order.

9.8 Satellite Visibility Prediction for Galileo Channels

To incorporate different GNSS's, the processes applicable to the GPS constellations have been applied to the SBAS, QZSS as well as Galileo constellation. Marco control has been applied for the GNSS system selection to allow completely switching off all the GNSS system except GPS as a reference.

Since Galileo has a unique ephemeris and almanac structure comparing to GPS and QZSS which share almost the same navigation message format, separate store structures have to be defined for Galileo.

Due to the limited capability of the RTOS being used and a desire to keep the software as simple as possible, a shared memory inter-thread communications process has been adopted from the “Aquarius” GPS receiver firmware. This can be implemented through the use of signals to indicate the presence of new data and shared memory to transfer that data. The data can be protected by a semaphore if necessary.

A conceptual block diagram of this communication process with the shared memory implementation has been shown in Figure 9-7.

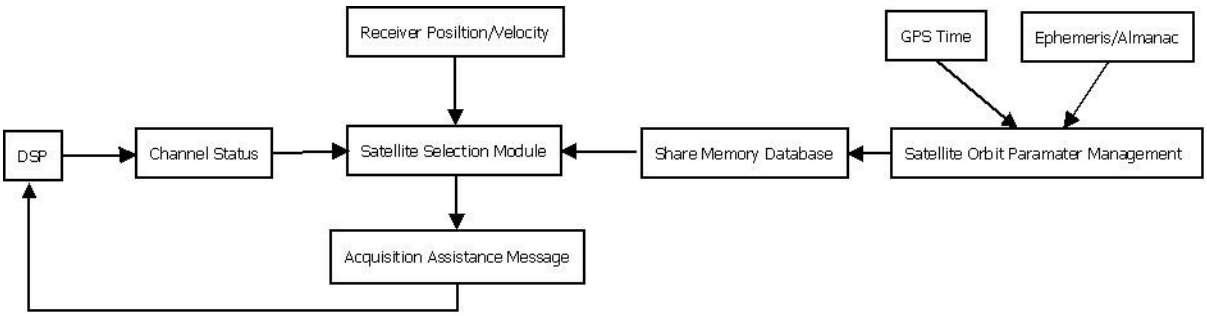


Figure 9-7. A shared memory inter-thread communications process implementation for satellite visibility prediction and selection for the channel assignment through the acquisition assistant message.

The transferring of the acquisition assistance to the DSP task is done using the channel status record, i.e. 'GALUpdateTracker()' function of the 'Tracking module, i.e. GALtracker.c'. This causes the acquisition assistance for a particular satellite & correlator channel to be copied into the correlator data-structures.

One consequence of this implementation with tracking status record is that the signal processing maintains its very own copies of the required acquisition assistance. This is because those pieces of data get used during the acquisition and reacquisition processes. In addition, it is undesirable to have redundant tables for recording the same information. Therefore, satellite need to be added or subtracted from the satellite sets either 'on-the-fly' (without knowing whether a 'better' set of satellites is present) or the calculation to determine the acquisition assistance can be done twice.

The satellite selection for Galileo has to get through two stages.

- At the first stage, satellite visibility is determined can calculated in the satellite orbit parameter management module so as to select the right satellite set and the associated hardware correlator channel number.

- In the second stage, the full acquisition assistance is recalculates and the required functions are called to perform the update through DSP module.

9.8.1 Expanding the memory database for the Galileo satellite set selection

The satellite selection module in the original “Aquarius” software was created solely for GPS satellite selection. This design is reflected in the implementation of this software module, whereby satellite sets are implemented by way of bit-masks, where each bit within a 32-bit unsigned word refers to a particular GPS satellite.

Implementing set operations using such a construct is both simple and efficient in terms of memory, but limits the sizes of the sets to no more than 32-elements. Since Galileo PRN can allow selection up

to 50. Clearly the original design using 32-elements to representing all satellite PRN number has a significant limitation for Galileo as well as other the capability of handling multiple GNSS systems.

Currently "Aquarius" expands the processing capability to GNSS by considering the constellation for each GNSS separately in parallel. Each system is allocated with adequate memory for the satellite set recording. This has the advantage of allowing the existing code base to simply be duplicated to handle each constellation in turn, but using the same algorithms in each case. However, support for such a scheme also encourages a constraint on the lower level processes such as satellite tracking and channel allocation. This follows because the structure naturally lends itself to the allocation of GPS satellites to GPS hardware channels, SBAS satellites to SBAS channels, QZSS satellites to QZSS channels, etc.

In order to accommodate the Galileo PRN number selection from 1-50 by utilising the existing 32-bit masking mechanism, a vector with two 32-bit elements has been introduced for Galileo satellite selection. Address zero is associated to PRN 1-32, while the Address one is associated to PRN 33-50. Ideally this vector implementation can allow the PRN selection up to 64. The remaining space (i.e. 14 bits) can be left for future GNSS development if it is required.

9.9 Satellite Visibility Prediction Results for GPS and Galileo Channels

In order to confirm the correctness of visible satellite prediction for Galileo channels in real-time, a real-time test using Spirent simulator with ground mode circular motion scenario had been performed. The firmware was flashed onto the Namuru 2.4 to perform cold start testing while Serial port and USB debug interface had been activated. Once the receiver position and timing is available due to GPS tracking and the almanac for Galileo has been uploaded properly on the receiver database, an almanac assisted sky search mode is activated. After the update of acquisition assistance message (showing on the Serial Port Terminal), within 1-2 min, all the Galileo Channel are corrected assigned and track properly. A snap shot of the satellite prediction for GPS and Galileo channels streaming from the USB debug interface has been shown in Figure 9-9 and Figure 9-10. While a screen shot (refer to Figure 9-8) of the Spirent simulator while testing was also captured to show that the visible GPS and Galileo satellites when the prediction occurs. From Figure 9-8, we can see that the visible GPS PRN was sorted according to the elevation angles as: 30, 22, 16, 31, 3, 6, 14, 18, 24, 11, 32, 19. While the visible Galileo PRN was sorted at the same manner as: 17, 22, 16, 21, 18, 19, 2, 3, 1. By checking the USB debug streaming, we could see that the receiver software predicted and sorted the visible GPS satellite number in the order of assigning the available 8 channels as, 30,16,22,31,3,6,14,32. While the Galileo satellite number assigning to the 5 available 4 channels as, 17,16,22,21,18.

In concluding, the satellite selection module can now correctly predict the visible Galileo Satellite as long as the receiver position/velocity, time and Almanac/Ephemeris are available for the prediction calculation. Otherwise, the sky search mode will be used for blind search which might take quite a long time for the Galileo satellite searching, although optimisation by carefully structuring the order of Galileo constellation in database might improve the speed of searching.

Figure 9-8. Snap-shot of the Spirent Simulator Screen showing all Satellite in view

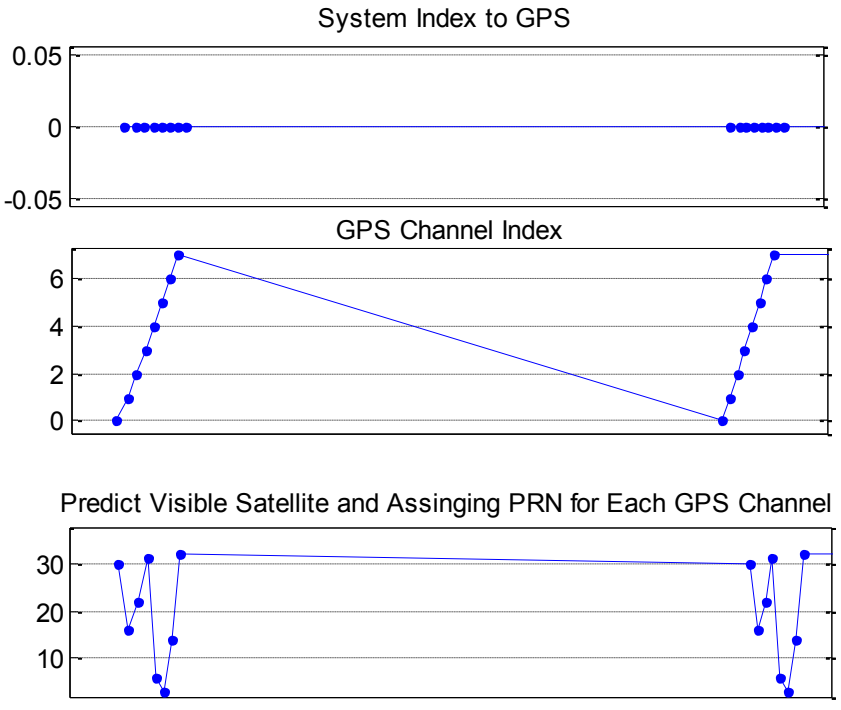


Figure 9-9. Predicted Visible GPS satellites for each available GPS channels

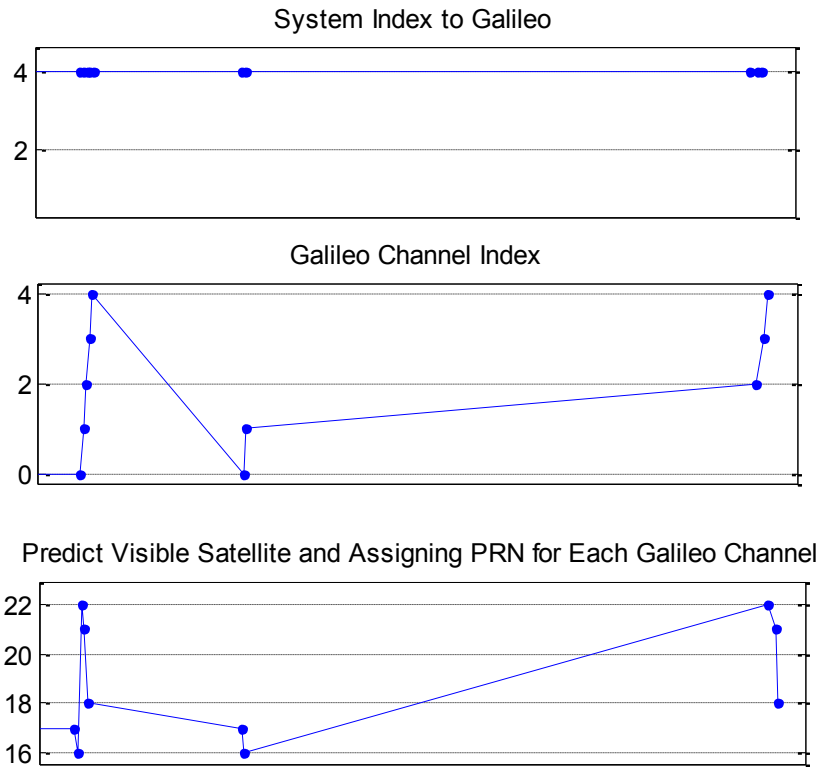


Figure 9-10. Predicted Visible Galileo satellites for each available Galileo channels

9.10 EPH and ALM data base update

In order to allow fast acquisition, a pre-set almanac or ephemeris message could allow the receiver working in warm start mode. Two UART command lines has been implemented so that the pre-set Galileo almanac and ephemeris message can be uploaded whenever it is necessary for debugging or warm start processing. The same mechanism can also expand for GPS and other GNSS system.

These two command lines are:

Uploading the pre-set ephemeris message:

$GPGPQ,UEUD,U

Uploading the pre-set almanac message:

$GPGPQ,UALM,A

9.11 Integrated Position Calculation

Once the tracking status is ok, synchronisation for observation measure is performing well, the satellite orbit parameters are healthy and available through checking the availability of Galileo Ephemeris, and the integrated position can now be obtained in the current version of firmware to allow integration solution calculation by using GPS, QZSS and Galileo signals.

The position calculation module is implemented in the Navigation Engine section of the firmware. It checks the health status of all the tracking channels and uses the entire health observation messages

for position calculation. If any of the satellite orbit is unhealthy, that particular satellite and channel would be excluded from the calculation. The satellite channels are picked according to their signal quality in terms of observation condition including whether tracking, synchronisation and availability of visible satellite orbital information.

9.12 Report update- through UART PC interface

In order to standardise the output, the Galileo signal processing statuses has been implemented in the same fashion as the original "Aquarius" GPS-only firmware. Extra data message has been streamed out to indicate the status of GNSS signal processing, such as GNSS integrated position calculation. When the GNSS systems are used, they are represented by G for GPS; E for Galileo; Q for QZSS.

e.g. $PNSWR,GPGSA is the extension of exiting $GPGSA message, except that it allows the indication of which system of which satellite PRN has been used in the integrated solution calculation. An example of the $GPGPSA and $PNSWR,GPGSA has been shown below, where only GPS (G) satellite has been used for the position calculation,

$GPGSA,A,3,3,6,16,14,32,22,30,31,,,,,2.0,1.2,1.7*32

$PNSWR,GPGSA,A,3,G,3,G,6,G,16,G,14,G,32,G,22,G,30,G,31,,,,,2.0,1.2,1.7*56

Another example is the streaming of observation message, the a GPS observation and Galileo observation examples are listed below respectively,

$PNSWR,OBS,13,2,G,1,3,0,03f,409146.924941048,22145458.328,-299.828,-3004059.473,27,357,46,337,32.9*07

$PNSWR,OBS,13,13,E,1,22,11,03b,409146.916761586,24597599.312,305.508,3066241.183,1078,000,47,337,32.9*35

9.13 Debugging Methodologies

A suite of debugging tools needs to be established in order to diagnose the firmware processes that produce erroneous pseudorange and carrier phase measurements during the development stage. This section describes the various combinations of freely available and proprietary in-house developed tools that are used in the testing process to identify and correct inappropriate firmware codes (also loosely known as bugs).

9.13.1 USB Interface for Firmware Debugging

The use of in-house developed proprietary RTOS with Altera Tools does not permit step debugging. To circumvent this restriction, a USB interface that is available on the Namuru V2.4 is customised into a proprietary peripheral that allows firmware developers to stream multiple time-tagged variables over the high-speed USB interface to the host computer. This tool is loosely called the USB Debugger. Unlike step debugging, this method of debugging allows firmware developers to visually identify anomalous behaviour in the firmware without stopping the RTOS.

One of the most useful applications of this tool is its use to perform high rate debugging whereby signal processing tasks are looked into. Thus, tasks such as tracking loop calculations that are

executed at 1kHz rate can be looked into over a long period of time to ensure reliability. On the other hand, the USB debugger can also be used to stream variables in low rate tasks such as carrier phase and pseudorange measurement generation. It can also stream high rate diagnostics conditionally. For example, to identify the correctness of an SV change process over the entire coarse of operation, the Doppler, mode of operation and correlation amplitude can be streamed only when a SV change is pertinent, as shown in Figure 9-11.

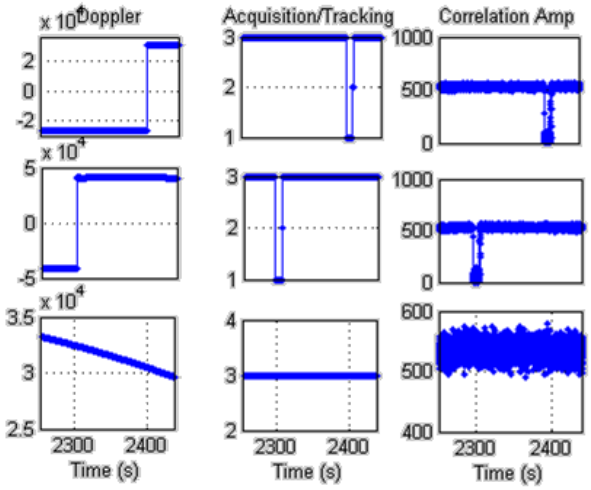


Figure 9-11. USB Debugger output used to identify glitches during SV transitions as two of the three channels loses lock of the SV going out-of-view and acquires a new SV coming in-view. Rows of the sub-figures belong to different channels while columns of the sub-figures belong to the indicated parameter at the top of the first row of sub-figures.

Various diagnostics can also be inferred from the RS232 UART stream that contains NMEA messages and proprietary NMEA-inspired messages (c.f. section 0) that delivers results of absolute positioning, pseudorange data, carrier phase data, timing data and other various data.

9.14 MATLAB Scripts for Pseudorange and Carrier Phase Analysis

Unfortunately, the UART stream is a text stream hence visibility of any glitches will be very low. Therefore, a suite of MATLAB analysis scripts has been developed to methodically detect anomalies in the UART stream and visually display the results. The scripts have been used to successfully detect undesired and sudden resets in the carrier phase accumulation process, pseudorange biases and glitches and many others.

9.14.1 RtkLib for Carrier Phase Analysis

RtkLib is a carrier phase post-processing software suite that is used provide centimeter level accurate relative positioning. RtkLib provides Trace files which show various diagnostics and residuals that can also be inferred as debugging data.

It is obvious that firmware glitches can instantaneously cause errors in the calculated position. Hence, the team has found that the most effective way of debugging is to align time series of the positioning results with USB debugger diagnostics. Alternatively the MATLAB analysis scripts can also be added into the mix if the glitches in the positioning results are not sufficiently obvious.

As a whole, whenever a new feature or change is implemented, a chain of testing process will entail. **Error! Reference source not found.** shows the lifecycle of one feature implementation. The unit testing phase refers to the testing of the feature within an isolated task whereby its operation will

not interfere with other tasks that were originally in place. In this phase, the behaviour of the feature is monitored to ensure that it behaves as intended. The RTOS integration testing allows the feature to interact with other tasks and is scrutinised using high rate USB debugging. The final verification stage is the full scale testing where its impact is observed throughout the entire intended duration of operation of the receiver.

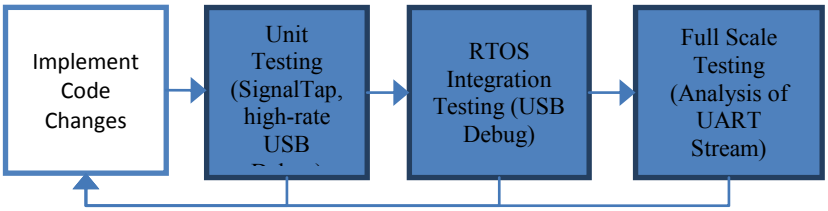


Figure 9-12. Firmware feature development and debugging lifecycle

10 Test Result

10.1.1 Namuru V3.2 GPS space-borne receiver Experiment Result

10.1.1.1 CASE 1: Error-free position testing along with PPS test

10.1.1.1.1 Absolute positions:

Absolute positions (i.e. Navigation solutions) are provided by the Namuru V3.2 receiver and are output via the serial port using National Marine Electronics Association (NMEA) messages types. Results were provided in the WGS84 XYZ coordinate system, which were compared with the truth data obtained from Spirent simulator. It was observed that the accuracy of the navigation and velocity solution from the base as well as rover case scenario was in centimetre level and precision was in metre level. Figure 10-1 and Figure 10-2 show the difference for base and rover scenario. Figure 10-3 and Figure 10-4 shows the velocity solution with respect to the truth for both cases. Table 10-1: Statistical properties of two scenarios lists the statistics of both position solution and velocity solutions.

Table 10-1: Statistical properties of two scenarios

	Base case scenario						Rover case scenario					
	X	Y	Z	Vx	Vy	Vz	X	Y	Z	Vx	Vy	Vz
Mean	0.3	-0.06	-0.44	0.02	0.09	0.001	-0.1	-0.9	0.3	-0.01	-0.1	-0.2
Std(1σ)	1.3	2.9	2.8	0.2	0.5	0.5	0.8	2.3	2.9	-0.1	0.5	0.4

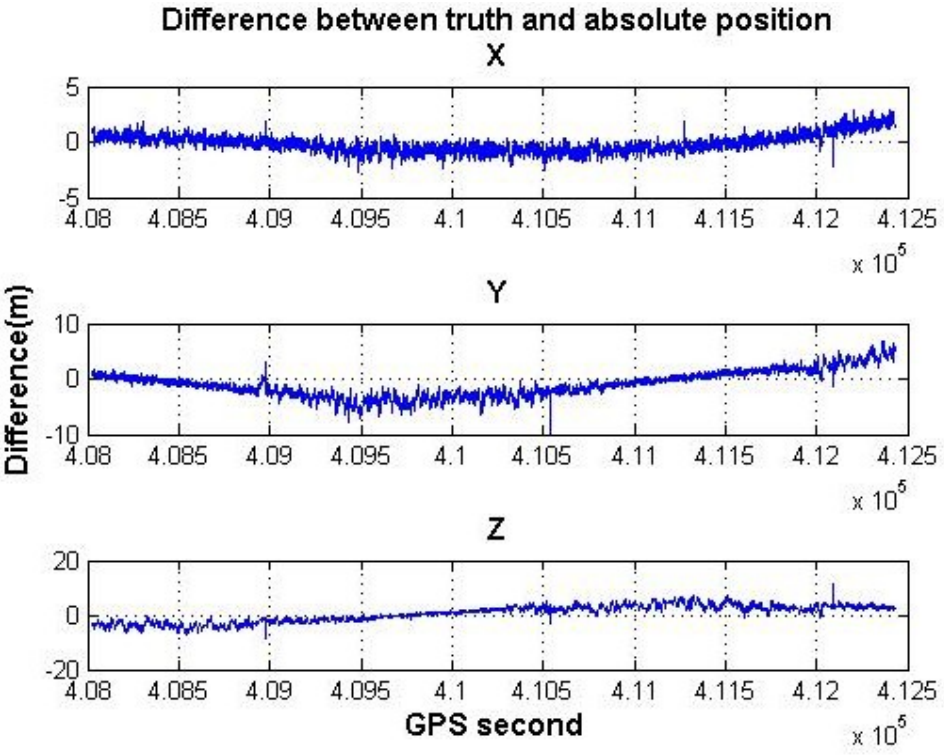


Figure 10-1: coordinate difference(base case)

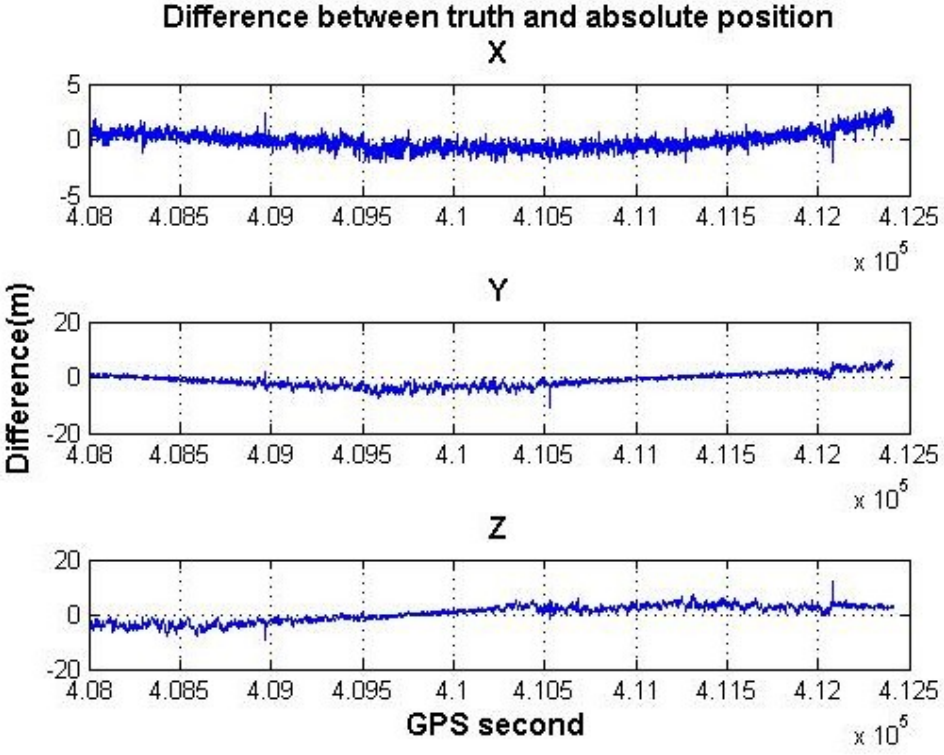


Figure 10-2:Coordinate difference(rover case)

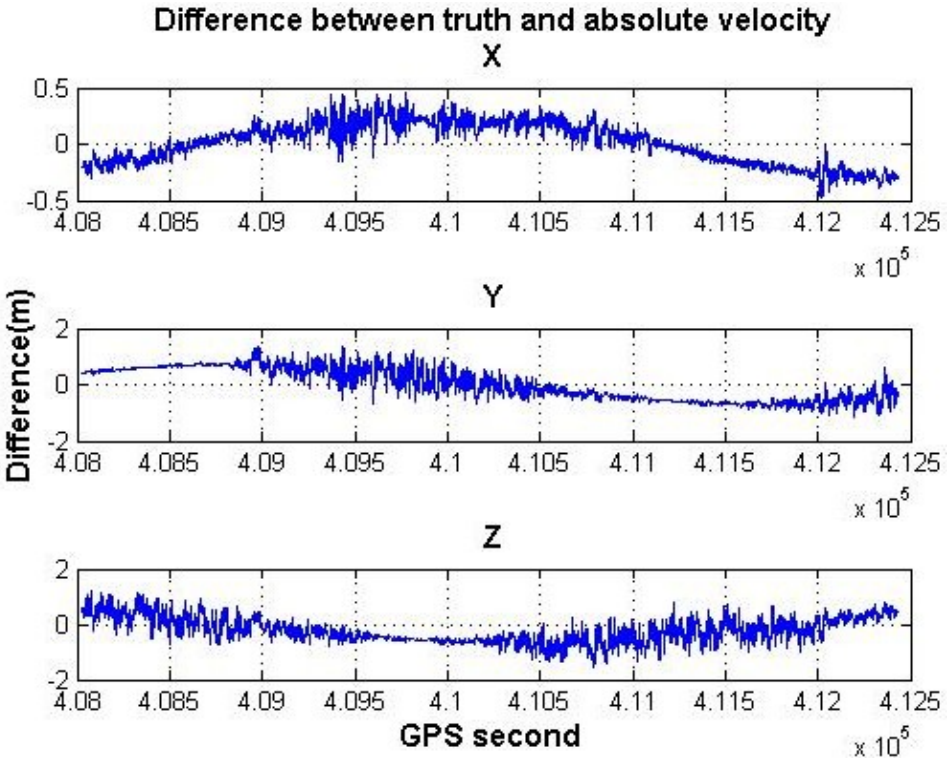


Figure 10-3:Velocity difference(base case)

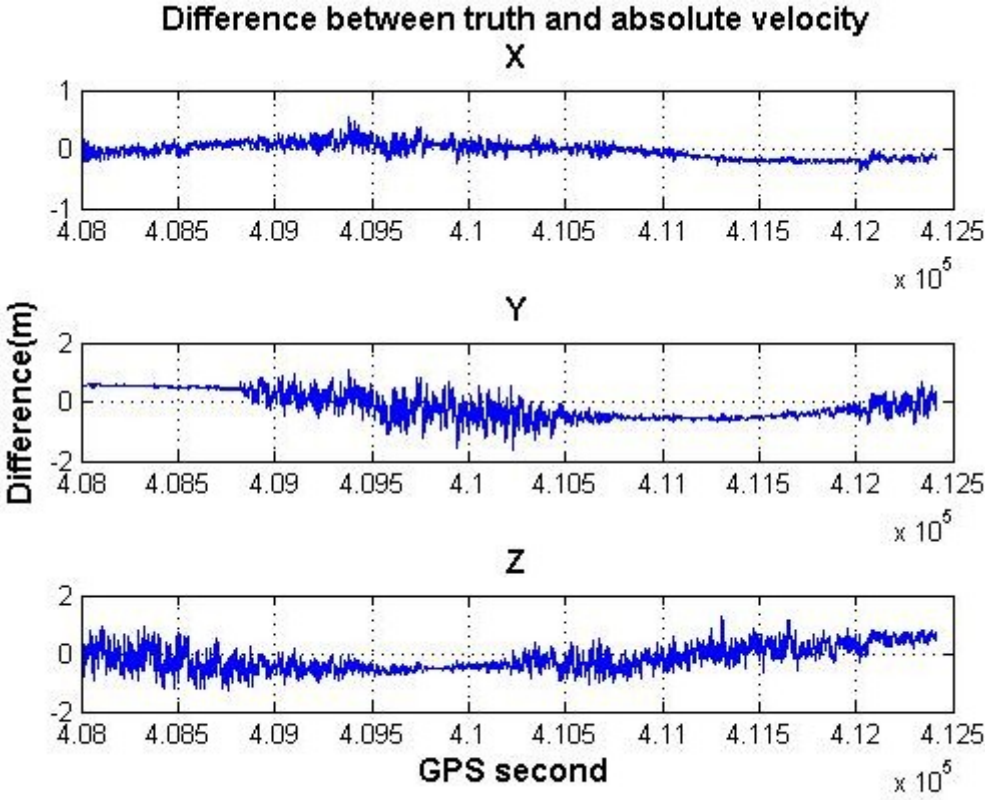


Figure 10-4:Velocity difference(rover case)

10.1.1.1.2 Carrier phase position

Baseline solutions generated using carrier phase observations were calculated using the RTKLib (version 2.4.1) software, an open source package for positioning using GNSS **(**Takasu and Yasuda, 2009**)**.

Results show that the accuracy of the differential carrier phase-based navigation solution is 0.001±0.02m, -0.001±0.02m and 0.001±0.05m for the E(East), N(North) and U(up) components, respectively. Figure 10-5 shows the baseline solution, whereas Figure 10-6 shows the difference between the truth and the baseline solution. Figure 10-7 shows the scatter plot of the test where it can be observed that results are precise. Lastly, relationship between accuracy and the number of visible satellite is shown in Figure 10-8.

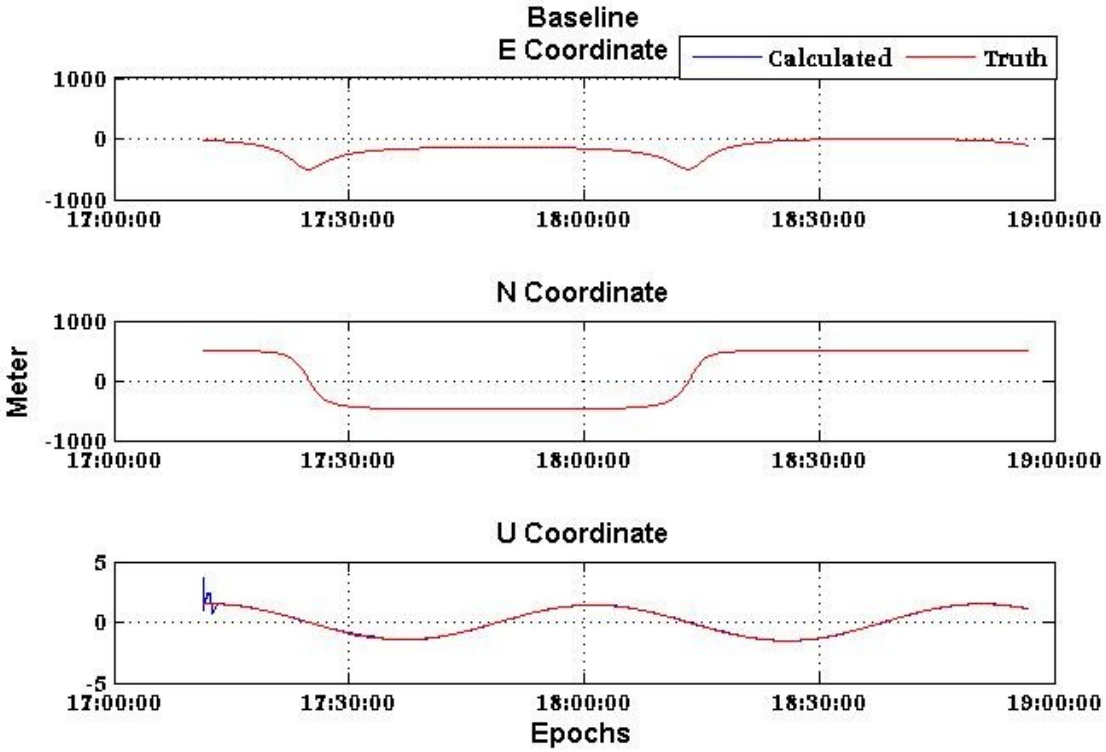


Figure 10-5: Baseline Solution

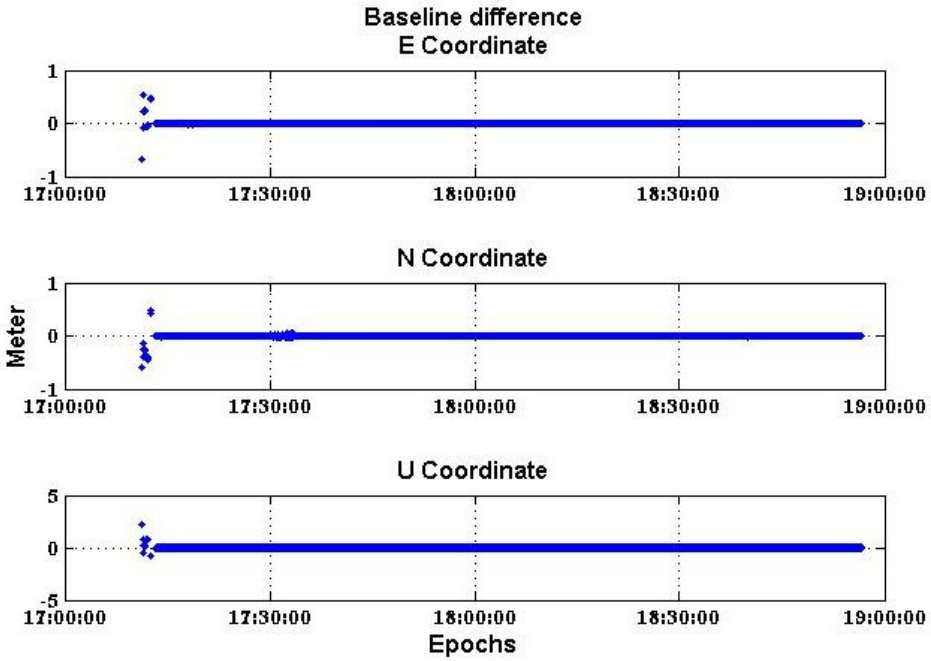


Figure 10-6: Baseline difference

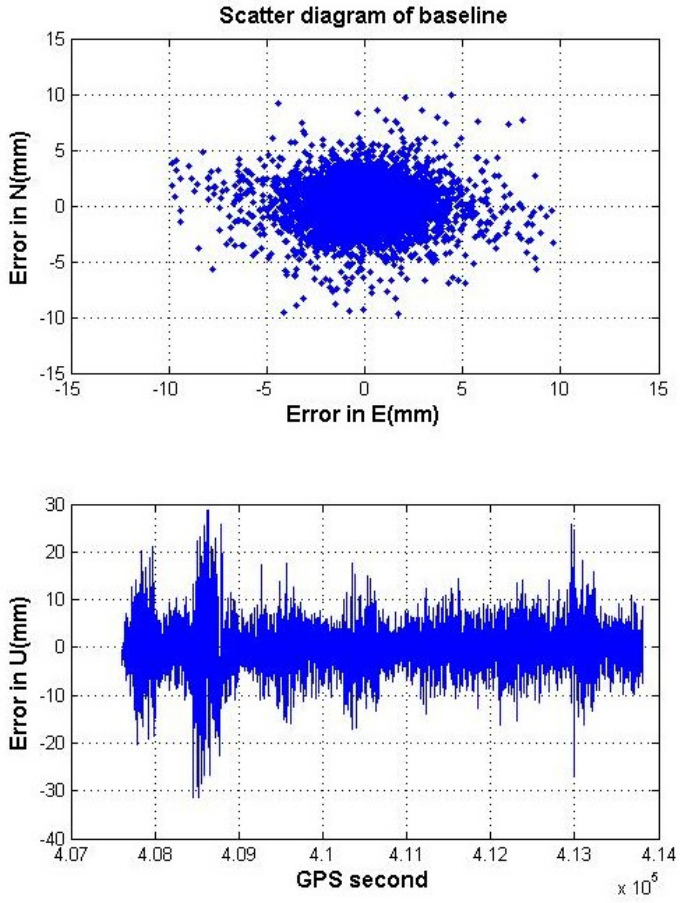


Figure 10-7:Baseline Scatter Plot

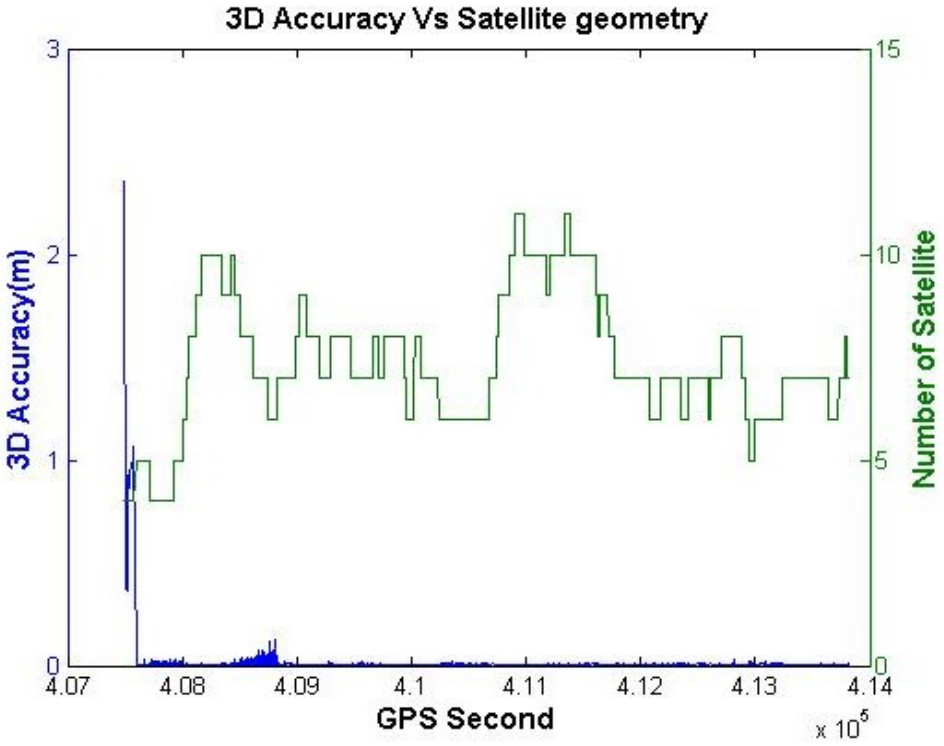


Figure 10-8: Relationship between number of satellite and baseline errors

10.1.1.1.3 PPS test

Results showed that PPS accuracy was ±60ns.

10.1.2 Receiver Performance

10.1.2.1 Cold start TTFF

Results show that 'cold start' requires 541±209s from 75 tests, with the maximum and minimum times measured as 1017s and 122s, respectively. The satellite geometry was unique in each case. The worst case occurs when the tracked satellites disappear from view before reaching the minimum required to obtain a reliable fix. In this type of situation the cold start requires more time than the typical terrestrial cold start. Figure 10-9 presents the cold start performance.

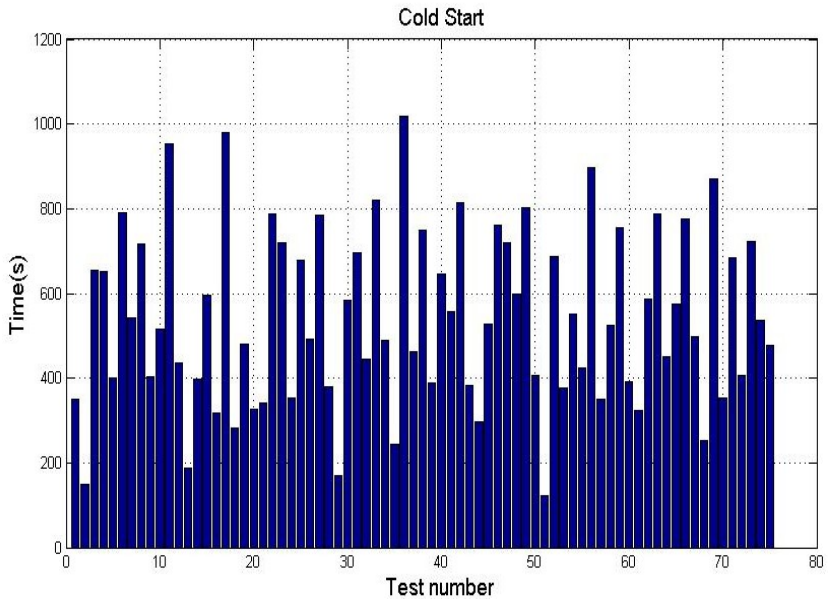


Figure 10-9: Cold start performance

10.1.2.2 Warm start TTFF

Results show that 'warm start' requires 47±13s from 97 tests, with the maximum and minimum times measured as 112s and 30s, respectively. The satellite geometry was different in each time. Figure 10-10 presents the warm start performance along. It can be seen that there is a repetition pattern exists. This is due to the satellite geometry (I.e. Position dilution of precision (PDOP)).

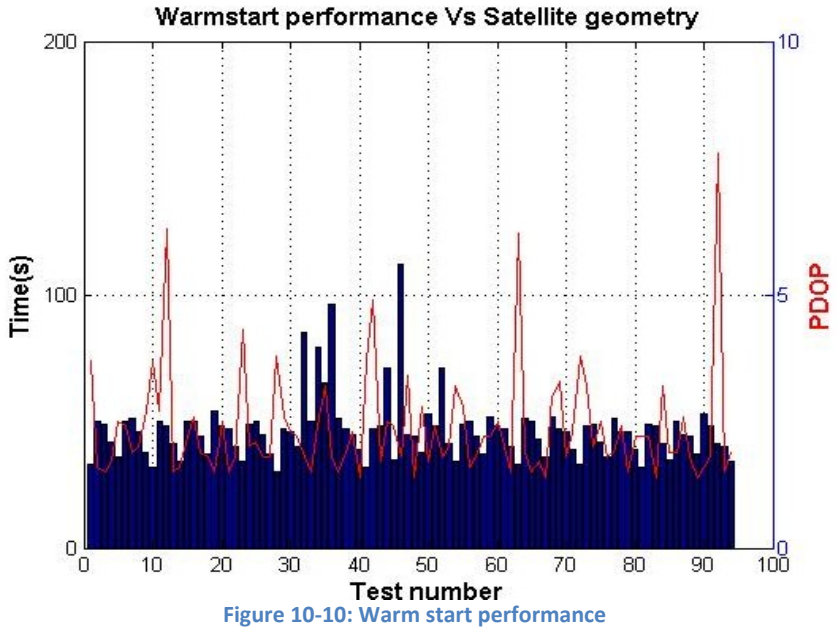


Figure 10-10: Warm start performance

10.1.2.3 Hot start TTFF

'Hot start' requires 44±23s from 80 tests, with the maximum and minimum times measured as 146s and 23s, respectively. The satellite geometry was different in each time. Figure 10-11presents the

hot start performance along. It can be seen that there is a repetition pattern exists same as warm start. This is due to the PDOP.

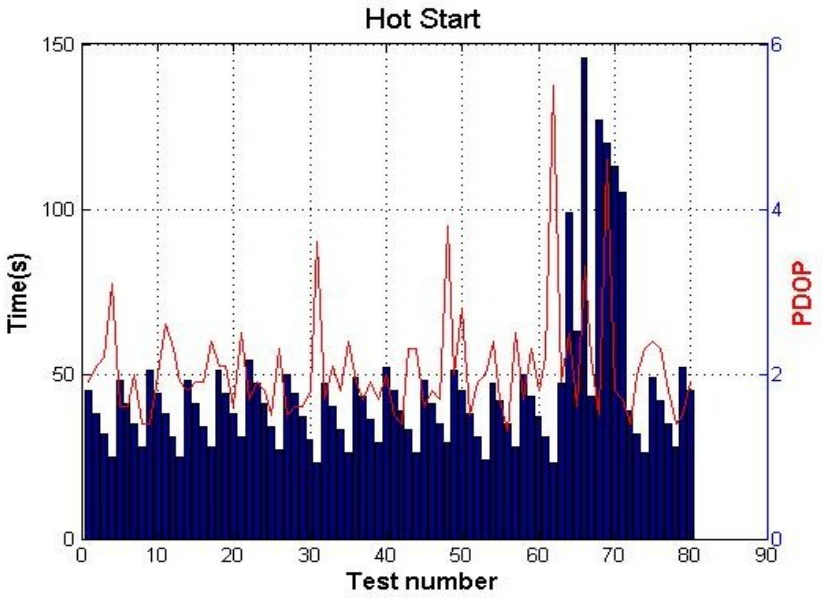


Figure 10-11: Hot start performance

10.1.2.4 Cold start TTL

From the cold start tests, it has been observed that cold starts' TTL was 152±78 seconds.

10.1.2.5 Warm start TTL

From the warm start tests, it has been observed that Warm starts' TTL was 40±12 seconds.

10.1.2.6 Hot start TTL

TTL for Hot starts is identified as 37±12 seconds.

10.1.2.7 Acquisition Sensitivity

Acquisition sensitivity tests showed that average signal power should be -118±0.7 dBm for satellite acquisition.

10.1.2.8 Tracking Sensitivity

Tracking sensitivity tests identified that to maintain a position solution minimum signal power is required is -128±0.7 dBm.

10.1.3 Namuru V2.4 dual-GNSS space-borne receiver Experiment Result

In this section, Namuru V2.4 has been used as a dual-GNSS space-borne receiver. V2.4 has the ability to process dual GNSS but with some resource limitations. This V2.4 is programmed for eight GPS channels and four Galileo channels. However, receiver's performance testing and PPS test were not conducted as this receiver was used for all initial development. Functionally, it was assumed that if any successfully receiver firmware update will be easily portable to V3.3 Garada receiver.

10.1.3.1 CASE 1: Error-free position testing along with PPS test

10.1.3.1.1 Absolute position

Galileo ephemeris decoding is still under development at the time of writing this report. As a result, GNSS observations are post processed for generating navigation solution. Two eight-hour tests (named as base and rover case) were conducted using an in-orbit scenario. Results were compared with the truth data obtained from Spirent simulator. It was observed that the accuracy of the navigation solution from the first 8-hour scenario was 0.001±1m, -0.01+1m and -0.04±2m for the X, Y and Z components, respectively, while second test exhibited results of 0.01±2m, -0.01+1m and -0.05±2m for the X, Y and Z components, respectively. Post-processing results are expected to be better than real time processing. As soon as targeted receivers (i.e. Namuru V3.3) have the capability of generating navigation solution using both GPS and Galileo observations, these tests will be conducted again.

Figure 10-12 and Figure 10-13 show the navigation solutions obtained from the RTKLib_Pro for both tests. Figure 10-14 &Figure 10-15 show the difference for 8-hour scenario. Figure 10-16presents the total number GNSS (i.e. GPS and Galileo) satellite verses number GPS and Galileo satellite used in 'Base case'.

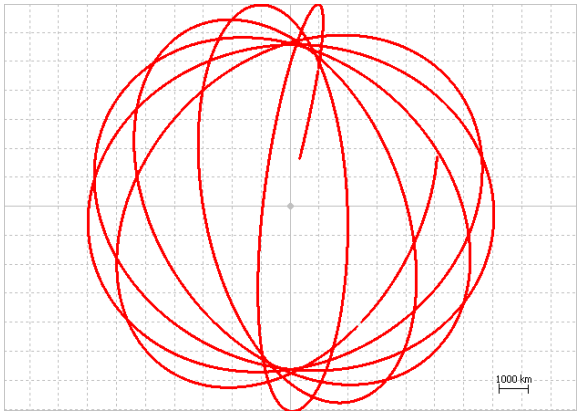


Figure 10-12: Base case

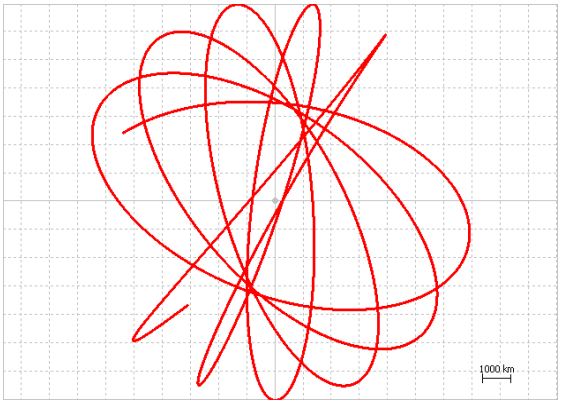


Figure 10-13:. Rover case

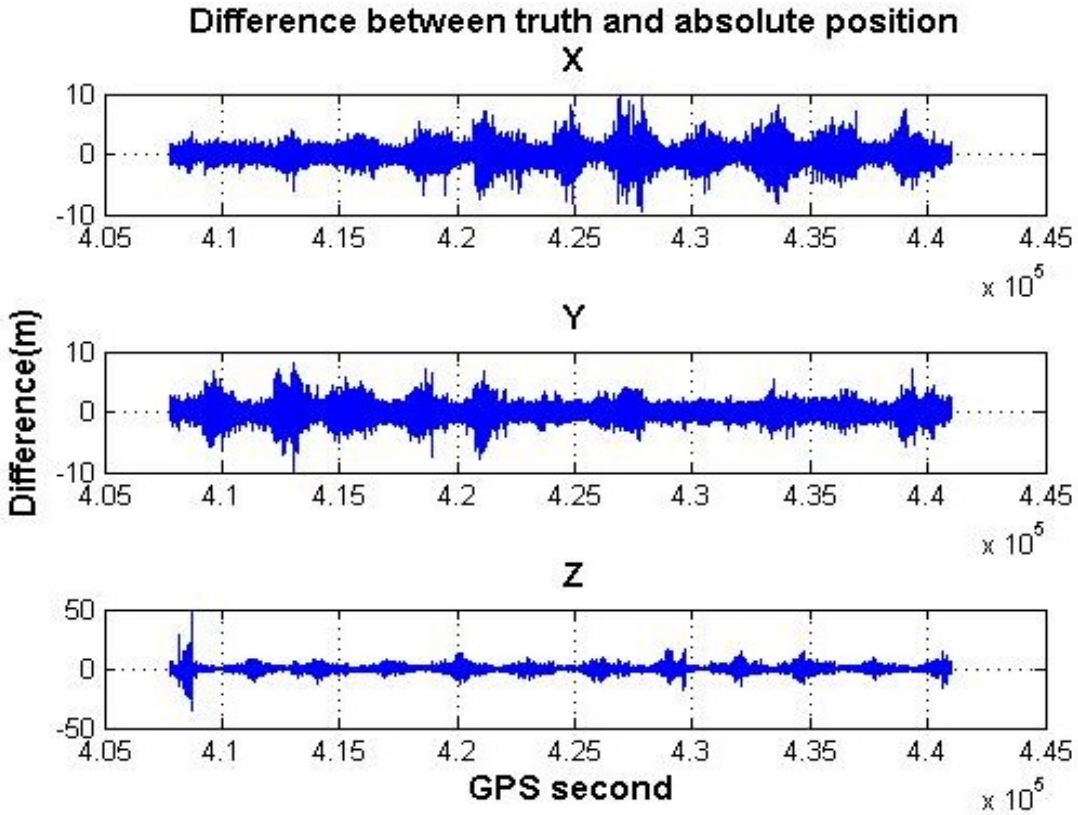


Figure 10-14: **Base case (difference from truth)**

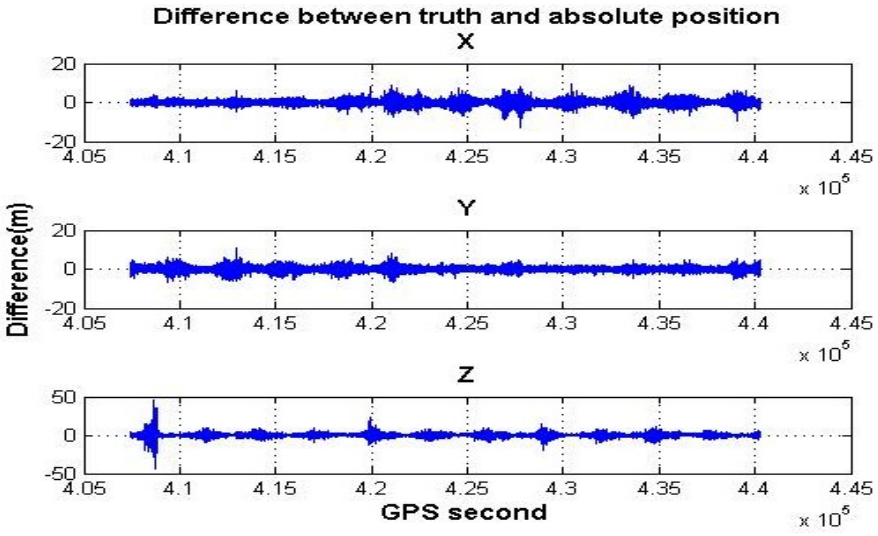


Figure 10-15: **Rover case (difference from truth)**

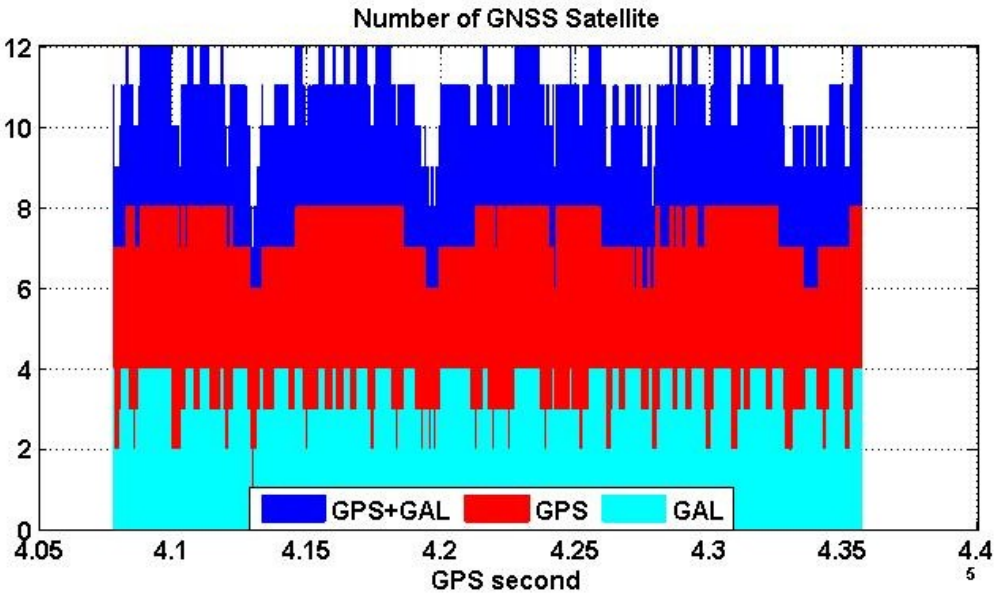


Figure 10-16: Number of GNSS satellite

10.1.3.1.2 Carrier phase positioning

Both 'Base_case' and 'Rover_case' were used for identifying the accuracy of the baseline using relative positioning. Carrier phase observations were used to calculate the relative position using RTKLib_Pro software. Results show that the accuracy of the differential carrier-phase based relative solution is 1±2mm, 1±2mm and -1±5mm for the E(East), N(North) and U(up) components, respectively. Figure 10-17 shows the baseline solution, whereas Figure 10-18 shows the scatter plot of the test where it can be observed that the errors are very close to zero.

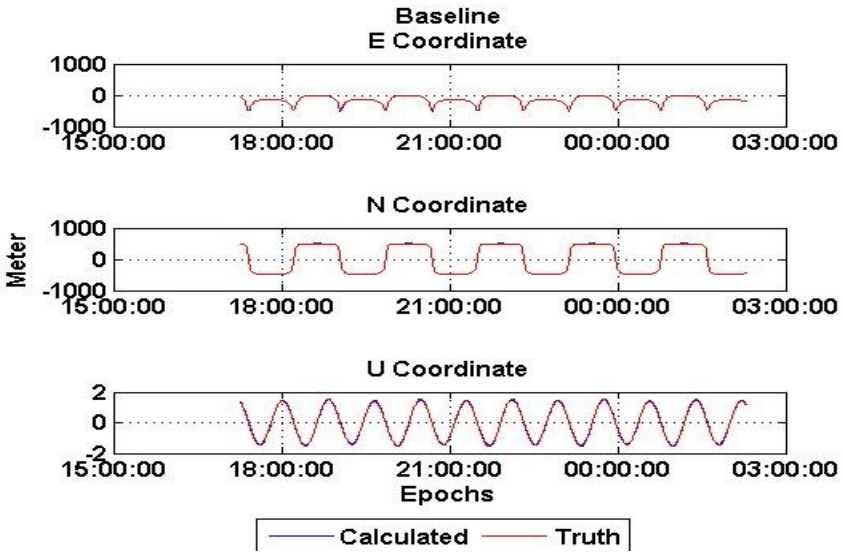


Figure 10-17:Baseline Solution

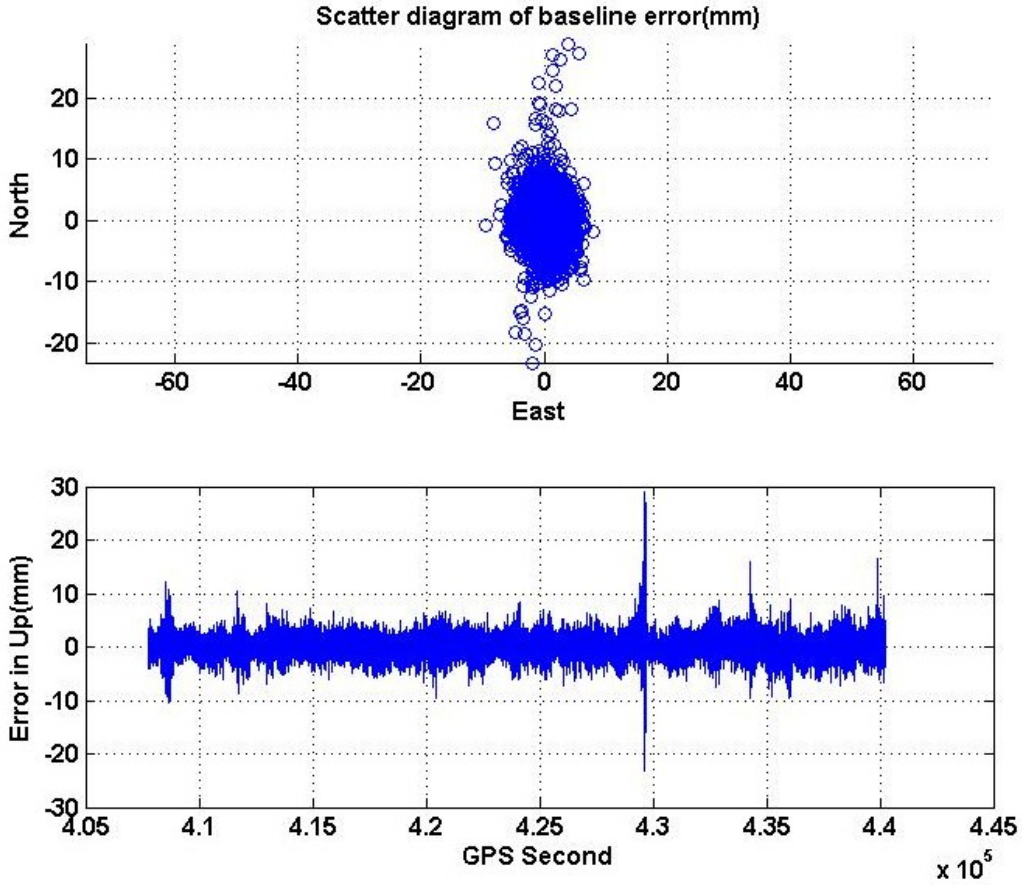


Figure 10-18: Baseline Scatter Plot

10.1.4 Namuru V3.3 dual-GNSS space-borne receiver's experiment result

In this section, Namuru V3.3 has been used as a dual-GNSS space-borne receiver which is an updated version of both V2.4 and V3.2. This V3.3 has the capability of processing muliti GNSS signals (i.e. L1, L5, E1 and E5). Initial absolute and carrier phase position tests (i.e. case 1) were conducted. However, receiver's performance testing and PPS test were not conducted as this receiver is still under development.

10.1.4.1 CASE 1: Error-free position testing along with PPS test

10.1.4.1.1

10.1.4.1.2 Absolute position

GNSS observations are post processed for generating navigation solution. Two eight-hour tests (names as base case and rover case) were conducted using an in-orbit scenario with 1 metre base line. It was observed that the accuracy of the navigation solution from the base case scenario was 0.001±1m, -0.01+1m and -0.04±2m for the X, Y and Z components, respectively, while rover case test exhibited results of 0.05±3m, -0.01+2m and -0.1±5m for the X, Y and Z components, respectively. Post-processing results are expected to be better than real time processing.

Figure 10-19 and Figure 10-20 show the navigation solutions obtained from the RTKLib_Pro for both tests. Figure 10-21 & Figure 10-22 show the difference for both cases. Figure 10-23 **and** Figure 10-24 present the total number GNSS (i.e. GPS and Galileo) satellite verses number GPS and Galileo satellite used for both cases.

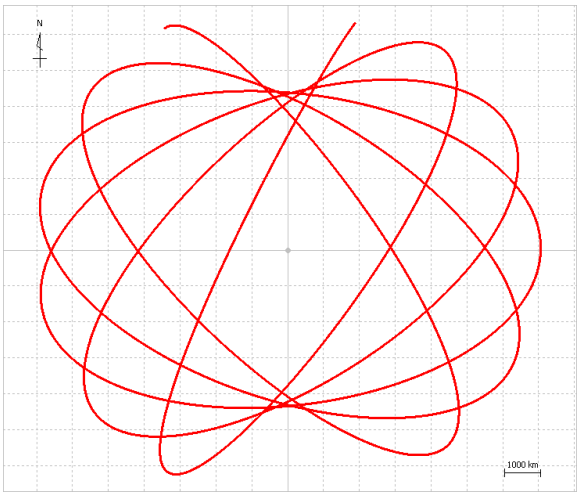


Figure 10-19: Base case

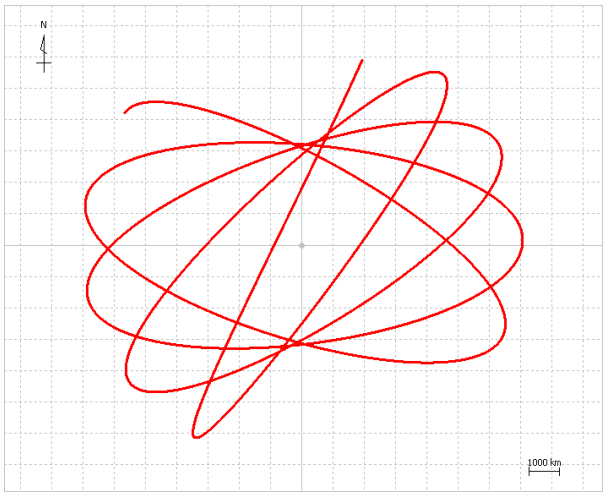


Figure 10-20: Rover case

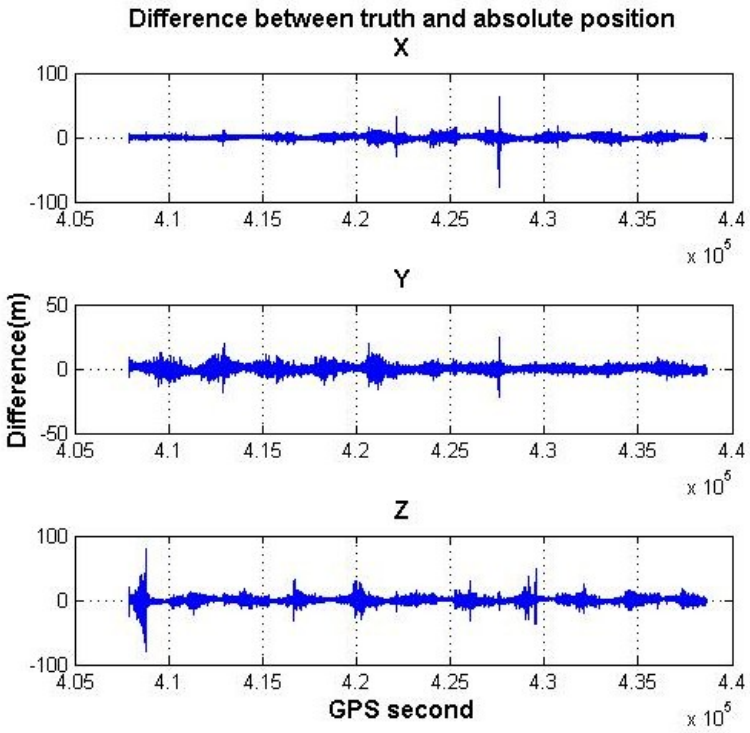


Figure 10-21: Base case (difference from truth)

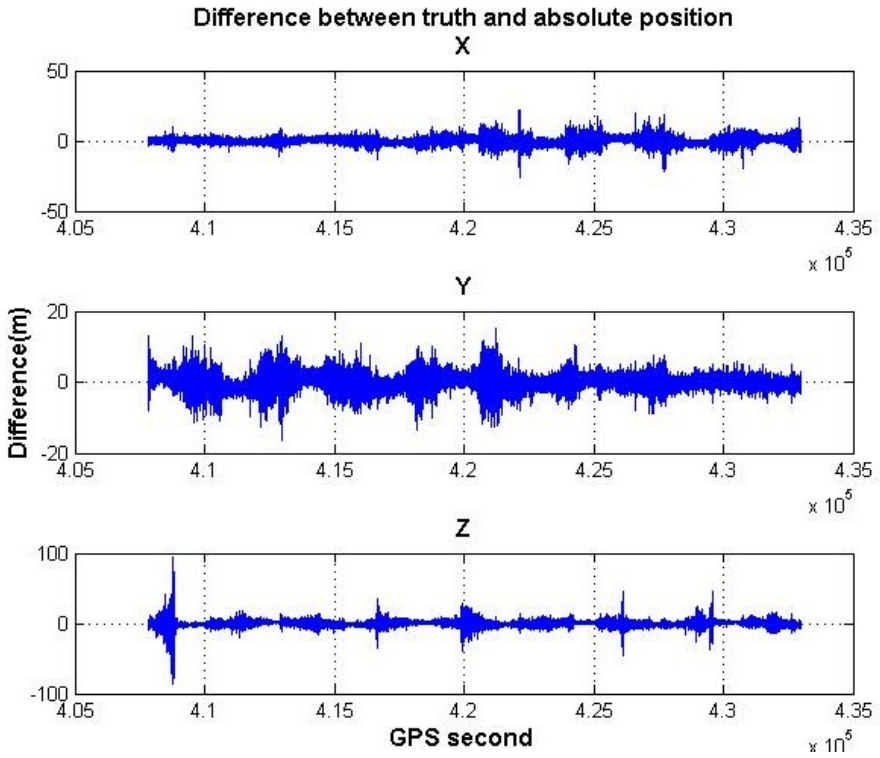


Figure 10-22: **Rover case (difference from truth)**

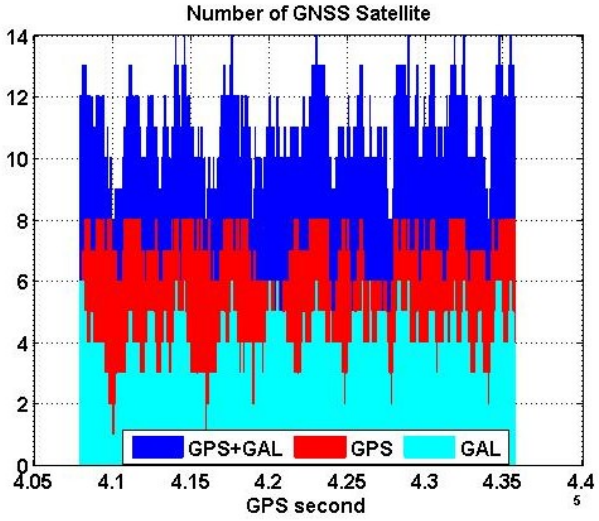


Figure 10-23: Number of GNSS satellite (Base case)

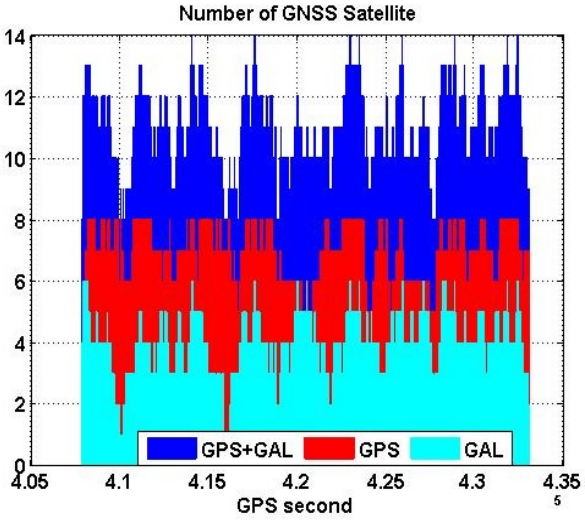


Figure 10-24: Number of GNSS satellite (Rover case)

10.1.4.1.3 Carrier phase positioning

Both 'Base_case' and 'Rover_case' were used for identifying the accuracy of the baseline using relative positioning. Carrier phase observations were used to calculate the relative position using RTKLib_Pro software. Results show that the accuracy of the differential carrier-phase based relative solution was 3±12mm, 7±8mm and -23±346mm for the E(East), N(North) and U(up) components,

respectively. Figure 10-25 shows the baseline solution, whereas Figure 10-26shows the scatter plot of the test where it can be observed that the errors are very close to zero after removing outliers.

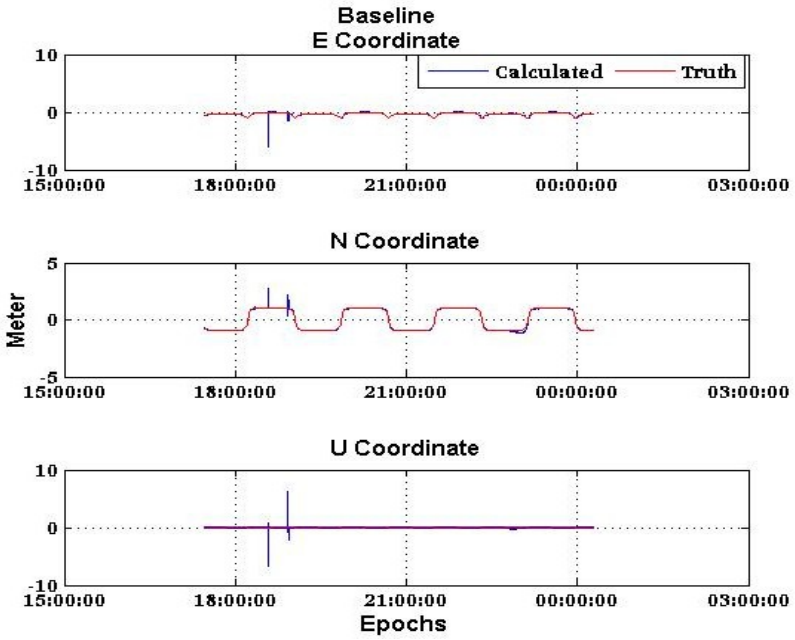


Figure 10-25: Baseline Solution

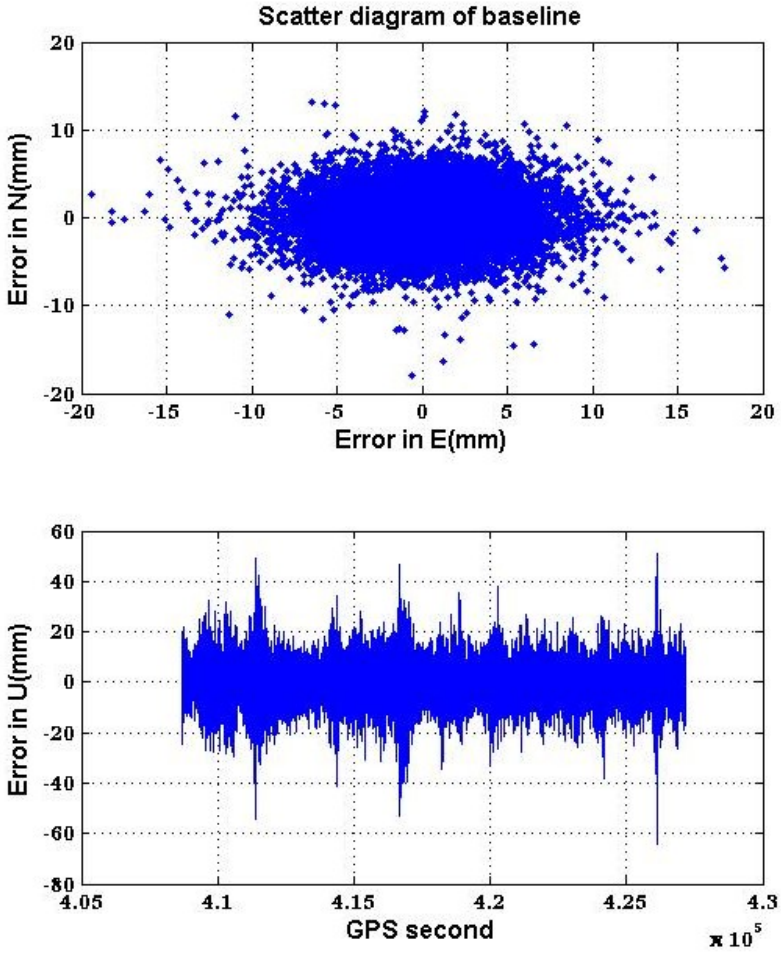


Figure 10-26: Baseline Scatter Plot

11 GNSS receiver operation and Environmental testing at DLR

11.1 Outline

A visit to the DLR space centre in Germany to compare the Namuru GNSS receiver with their Common Of The Shelf (COTS) based Pheonix receiver was carried out during the week of June 12th of June 2013. The Namuru GNSS receiver uses similar but more modern electronic components to the DLR receiver which is now obsolete. This visit was to identify which environmental testing procedures would be suitable to add to our existing Namuru receiver test procedures and to discover some operational failure experience.

11.2 Operational Issues

Some examples of space missions experienced by the team at DLR were discussed in detail to understand the types of Single Event Upset (SEU) failures and frequency of occurrence that could be expected using a COTS receiver.

Mission	Altitude	Memory size	Fault	Interval

Probo 2	~700Km	Large	SEU	2 days
Prisma	~650Km	Small	SEU	10 days
Probo V	~800Km	Small	SEU	10 days

The table above shows that the greater the size and density of the SRAM in the receiver, the greater the frequency of SEU faults. The nature of these faults is that the receiver ceases to function with correctly often with no further data output. Although it is not possible to study the exact nature of the faults in space, it is almost certain that they come from a latch-up of the SRAM in the receiver. This appears to be non-destructive because a simple power cycle reboot of the receiver restores normal operation.

DLR advised that the ESA radiation department is currently studying this problem using a range of SRAM devices from a range of different suppliers.

11.3 Flight model tests

Each flight model candidate receiver is put through tests in the following order. Note that multiple functional tests are performed throughout this sequence to verify basic operational performance.

11.3.1 Visual inspection

Careful visual inspection using magnifying equipment is performed to verify that there is no transit damage.

11.3.2 Functional test

A basic functional test is carried out to verify basic operation using a common antenna. This allows a position fix to be generated and a number of satellites to be tracked.

11.3.3 Vibration test

Stand alone shock and vibration tests are carried out in accordance with the specifications adopted by DLR in their document number TN04-02.

11.3.4 Functional re-test

A basic functional test is carried out to verify basic operation using a common antenna. This allows a position fix to be generated and a number of satellites to be tracked.

11.3.5 Thermal vacuum test

The receiver is tested in a thermal vacuum chamber while in operation according to the specifications adopted by DLR in their document number TN04-07.

11.3.6 Functional re-test

A basic functional test is carried out to verify basic operation using a common antenna. This allows a position fix to be generated and a number of satellites to be tracked.

11.3.7 EMC test

EMC testing is carried out with the receiver operating to determine the emission levels that may affect other satellite payload components.

11.3.8 Functional re-test

A basic functional test is carried out to verify basic operation using a common antenna. This allows a position fix to be generated and a number of satellites to be tracked.

11.3.9 Radiation test

An operating receiver is tested according to the procedures adopted by DLR in their document number TN04-01. SEU and Total Ionisation Dose (TID) characteristics are studied and summarized in the DLR report titled “DLR GPS Receiver Radiation Test Report”, Dated 08/02/2011 Issue 1 Rev 0.

11.3.10 Functional re-test

A basic functional test is carried out to verify basic operation using a common antenna. This allows a position fix to be generated and a number of satellites to be tracked.

11.3.11 Hardware-in-the-loop test

After integration into the satellite, a series of space simulation functionality tests may be carried out to different levels by the satellite integrator to verify total payload operation. During these tests some functional testing, as required by the integrator, may be varied out with the GNSS receiver. Tests will depend on the required GNSS payload specification.

11.4 Our tests

From the tests described above, it will be important for the Namuru receiver testing procedures to adopt both the thermal vacuum test and the radiation test. Both of these tests are to be carried out with a fully operating receiver in the testing environment.

From the radiation tests it will be essential to perform the SEU test because of the amount of SRAM used on the Namuru receiver. However, the TID test will be optional because this is not seen as critical for short missions in low earth orbit where the altitude and time period in flight will not expose the Namuru receiver to sufficient radiation to be at risk.

11.5 Concluding remarks

From discussions with the DLR team and a review of their documents there will be significant advantage in using their standards and procedures to improve and complete our testing of the Namuru receiver. Of particular interest is the offer of the DLR team to carry out the selected tests on the Namuru receivers as and when we require.

The DLR team can also arrange a TID test to be carried out on one Namuru receiver to gain an indication of performance.

12 Namuru Receiver Command and Reports

12.1 Serial Port Communications

The Namuru receiver / Aquarius GNSS firmware communicates with the user using serial port commands and reports. The settings used by the serial port are fixed at compile time and are currently set at a speed of 115,200 bits per second with a setting of eight data bits, no parity, one stop bit and no flow control; this being a typical speed and setup for PC based RS232 serial port communications.

It is desirable to use as high a transmission speed as possible due to the large volume of data that may be transmitted by the receiver during normal operation. For example, the generation of RINEX files requires that the output of GPS/GNSS measurements and GPS/GNSS ephemeris, ionospheric corrections and UTC corrections be enabled.

At present, two versions of Aquarius firmware are available. One is dedicated for Namuru V3.2 GPS only receiver and another one are for both V2.4 & V3.3 which are capable of both GPS and Galileo GNSS.

12.2 Reports

GNSS reports output by Aquarius are all event triggered. This means that unlike some receivers where the reports may simply be triggered at a regular and user selectable rate, the reports are triggered by some event that occurs within the GNSS firmware. However, although each report is triggered by a call to an appropriate routine within the software, the user has control over which reports are transmitted by the receiver. This allows the user to avoid cluttering the serial port output with messages that may not be required. User selection of the output serial reports is performed using the GPGPQ command and the settings chosen by the user will be stored in non-volatile memory, provided such a memory is available. The available output reports are shown in the table below (NMEA in blue typeface)

12.3 NMEA Standard

Many GPS receivers support the National Marine and Electronics Association (NMEA) standard 0183 with respect to the serial port commands and reports that are used to interact with the receiver. This standard has found widespread use for such receivers and many of the software applications that are used to interact with GPS receivers support this standard as well. It is for this reason that the Aquarius command and report set is based on this standard as well, although the software does not adhere strictly to the requirements of NMEA 0183.

The Aquarius firmware implements a number of predefined NMEA sentences, including the ALM, GGA, GPQ, GRS, GSA, GSV, VTG and ZDA sentences, although the GPQ sentence has been modified slightly to allow sentences to be turned on or turned off. The other remaining sentences have been implemented as NMEA proprietary sentences, although deviate from the standard in that the maximum 80-character sentence length constraint has been disregarded. This was found to be necessary in order to ensure that sufficient space within each sentence was available to include the quantities of interest.

NMEA sentences all start with a '$' character and end with a '*', followed by a two hex digit checksum and a carriage return/linefeed character pair. The checksum is calculated as the exclusive-or of the characters between the '$' and the '*'. All of the sentence characters are standard printable ASCII characters. Aquarius includes NMEA checksums on all of the output sentences, but does not require checksums on the input sentences. However, if a checksum is included on the input it is required to pass otherwise the sentence will be disregarded. This feature has been included to allow the user to easily interact with the receiver without needing to calculate a checksum for each message or forcing the user to use a particular application that performs the insertion of the required checksum each time.

Following table presents the applicable commands to different receiver versions.

String ID	Name	V2.4	V3.2	V3.3
AAM	Acquisition assistance	√	√	√
ACK	Command acknowledge	√	√	√
ALM	GPS almanac	√	√	√
ALM	GNSS almanac	√	×	√
DBG	DBG navigation failure	√	√	×
CFG	Receiver configuration	√	√	√
CHN	Channel debug report	√	√	√
CHN	Channel debug report for GNSS	√	×	√
CLN	Clear non-volatile memory	√	√	√
ECI	ECI position/velocity	√	√	√
EPH/EUD	GPS Ephemeris	√	√	×
EPH/EUD	GNSS Ephemeris	√	×	√
EPL	Request all ephemeris	√	√	×
FWV	Firmware version	√	√	√
GGA	Lat, long & height	√	√	√
GPQ	Query receiver	√	√	√
GRS	Measurement residuals	√	√	√
GSA	GPS Satellites and DOP	√	√	√

GSA	GAL Satellites and DOP	√	×	√
GSV	Satellites in View	√	√	√
ION	Ionospheric corrections	√	√	√
MEM	DBG memory examination	√	√	√
NAV	Navigation message	√	√	×
OBS	Raw GPS measurement	√	√	×
OBS	Raw GNSS measurement	√	×	√
PPS	Pulse per second	√	√	×
RTC	Real time clock	√	√	√
SEL	Satellite selection	√	√	√
SPT	Serial port settings	√	√	×
STK	RTOS stack usage	√	√	×
TCO	Monitor TCXO offset	√	√	√
TIM	Time pulse setup & trigger	√	√	×
TSK	RTOS task state	√	√	√
TST	Automatic testing	√	√	×
UTC	UTC corrections	√	√	√
VTG	Speed/course over ground	√	√	√
XYZ	XYZ position/velocity	√	√	√
ZDA	UTC time	√	√	√
ZZZ	Debug and event trigger	√	√	√
NAV	Navigation message	√	√	√
UEUD	Upload Galileo ephemeris	√	×	√
UALM	Upload Galileo almanac	√	×	√

12.4 Description of the commands

12.4.1 AAM Command and Report

Description

Acquisition assistance message

$PNSWR,AAM,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14*

The sentence is triggered whenever the acquisition assistance controlling the receiver channel allocation is re-calculated. Logic has been included in order to avoid resending previously sent acquisition assistance. The output rate is variable, although typically a new set of sentences will be output whenever the satellite selection task is run.

Fields

Number	Name	Description
F0	$PNSWR,AAM	Sentence header
F1	s	System, 0=GPS, 1=SBAS, 2=QZSS, …
F2	c	Channel
F3	x	SV number
F4	w	Reference Time-of-Receipt week
F5	t.t	Reference Time-of-Receipt time-of-week (s)
F6	t.t	Satellite time-of-flight, with SV clock corrections included (s)
F7	x	Code phase windows (chips)
F8	x.x	Satellite Doppler frequency (Hz)
F9	x	Doppler window (Hz)
F10		RESERVED
F11	x	SV elevation (degrees)
F12	x	SV azimuth (degrees)
F13	H	Debug flags (hex)
F14	H	Debug flags (hex)

Remarks

1. The sentence may be used by the user to supply acquisition-assistance to the receiver. This was included to allow support for assisted GPS projects, although this is currently not really supported by the receiver, which doesn't have the necessary sensitivity.
2. SBAS is currently not supported, although QZSS has been implemented.
3. The reference time-of-receipt time-of-week is printed out to 'ms' resolution. However, the time is exact and the digits that follow after the 3rd decimal place should be considered to be 0s.

4. The time-of-receipt minus the time-of-flight allows the satellite time-of-transmission to be calculated, from which the code phase, etc are easily obtained.

12.4.2 ACK Report

Description

Acknowledge report

$PNSWR,ACK,*F1*

The sentence is triggered whenever command acknowledgement has been enabled via the CFG command and when a command is successfully processed by the receiver.

Fields

Number	Name	Description
F0	$PNSWR,ACK	Sentence header
F1	s	Command string being acknowledged

Remarks

1. To enable this feature, it is necessary to configure the CFG command with the 'm' option. Selecting 'M' disables the feature.
2. The reports are issued following receipt and processing of the input command. In some cases, the acknowledgment will not be transmitted if the transmitted command is rejected (for some reason).

12.4.3 ALM Command and Report

Description

Almanac messages

$GPALM,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15*

The sentence is triggered whenever a new almanac element superseding an existing almanac element is extracted from the transmitted navigation-message. The output rate is variable, although typically a new almanac will be output following a non-volatile memory clearing event until the almanac is up to date.

Fields

Number	Name	Description
F0	$GPALM	Sentence header
F1	x	Total number of GPALM sentences
F2	x	Number of this GPALM sentence
F3	x	SV PRN number
F4	HHH	Almanac Week (10 bit hex), (weeks)
F5	HH	Almanac health (8 bit hex)
F6	HHHH	Eccentricity (16 bit hex), unsigned, LSB sf=2^{-21}
F7	HH	Toa (8 bit hex) : (sec), unsigned, LSB sf=2^{12}
F8	HHHH	δ_i : (sc), 2s complement, LSB sf=2^{-19}
F9	HHHH	OmegaDot : (sc/sec), 2s complement, LSB sf=2^{-38}
F10	HHHHHH	(A) $^{1/2}$: ($\sqrt{m}$), unsigned, LSB sf=2^{-11}
F11	HHHHHH	ω : (sc), 2s complement, LSB sf=2^{-23}
F12	HHHHHH	$(OMEGA)_0$: (sc), 2s complement, LSB sf=2^{-19}
F13	HHHHHH	Mo : (sc), 2s complement, LSB sf=2^{-23}
F14	HHH	Af0 : (sec), 2s complement, LSB sf=2^{-20}
F15	HHH	Af1 : (sec/sec), 2s complement, LSB sf=2^{-38}

Remarks

1. The sentence may be used by the user to re-instate the almanac after the non-volatile memory has been cleared.
2. As a non-standard enhancement, the almanac transmitted by the QZSS satellite is also output, but with an SV number of 193.
3. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E.
4. When ALM reports are requested, the receiver will first output those elements that have been recently received. If requested again, the full set will be output.
5. Sending almanac via the serial always replaces existing almanac irrespective of whether it is out of date or unusable

12.4.4 ALM Command and Report for GNSS

Description

Almanac messages

$GPALM,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16*

The sentence is triggered whenever a new almanac element superseding an existing almanac element is extracted from the transmitted navigation-message. The output rate is variable, although typically a new almanac will be output following a non-volatile memory clearing event until the almanac is up to date.

Fields

Number	Name	Description
F0	$GPALM	Sentence header
F1	x	Total number of GPALM sentences
F2	x	Number of this GPALM sentence
F3	x	SV PRN number
F4	HHH	Almanac Week (10 bit hex), (weeks)
F5	HH	Almanac health (8 bit hex)
F6	HHHH	Eccentricity (16 bit hex), unsigned, LSB sf=2^{-21}
F7	HH	Toa (8 bit hex) : (sec), unsigned, LSB sf=2^{12}
F8	HHHH	δ_i : (sc), 2s complement, LSB sf=2^{-19}
F9	HHHH	OmegaDot : (sc/sec), 2s complement, LSB sf=2^{-38}
F10	HHHHHH	(A) $^{1/2}$: ($\sqrt{m}$), unsigned, LSB sf=2^{-11}
F11	HHHHHH	ω : (sc), 2s complement, LSB sf=2^{-23}
F12	HHHHHH	$(OMEGA)_0$: (sc), 2s complement, LSB sf=2^{-19}
F13	HHHHHH	Mo : (sc), 2s complement, LSB sf=2^{-23}
F14	HHH	Af0 : (sec), 2s complement, LSB sf=2^{-20}
F15	HHH	Af1 : (sec/sec), 2s complement, LSB sf=2^{-38}
F16	X	G or E. G for GPS and E for Galileo.

Remarks

1. The sentence may be used by the user to re-instate the almanac after the non-volatile memory has been cleared.
2. As a non-standard enhancement, the almanac transmitted by the QZSS satellite is also output, but with an SV number of 193.
3. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E.
4. When ALM reports are requested, the receiver will first output those elements that have been recently received. If requested again, the full set will be output.

5. Sending almanac via the serial always replaces existing almanac irrespective of whether it is out of date or unusable

12.4.5 CFG Command and Report

Description

Receiver configuration and tuning

$PNSWR,CFG,*F1,F2,F3,F4,F5,F6,F7*

The sentence is only output if requested.

Fields

Number	Name	Description
F0	$PNSWR,CFG	Sentence header
F1	x.x	SV visibility selection mask angle (deg)
F2	x	Required solution mode: 2=2D,3=3D,4=Auto
F3	xxxxxxxxx	Flags: U/u=Disable/Enable Measurement Upload I/i=Disable/Enable Ionospheric Correction T/t=Disable/Enable Tropospheric Correction E/e=Disable/Enable Restoration of ephemeris A/a = Disable/Enable Restoration of almanac F/f = Disable/Enable TCXO frequency compensation M/m = Disable/Enable command acknowledgement K/k = Disable / Enable PVT Kalman filter V/v = Disable / Enable VC-TCXO control
F4	x	Tuning Parameter: 1=Stationary, 2=Land Mobile, 3=Air, 4=Space
F5	x.x	HDOP Limit (for Auto mode switching)
F6	x.x	PDOP Limit (for Auto mode switching)
F7	x.x	Carrier smoothing time constant τ (sec)

Remarks

For operation in low earth orbit, it is essential to select 'Space' mode. This has the effect of increasing the Doppler search space and reducing the sensitivity in order to ensure that the satellites can be acquired.

1. If TCXO frequency compensation is applied, then the receiver will apply software corrections to the local clock. This has the benefit of ensuring that errors do not accumulate in the local clock thereby reducing the need to apply 'clock resets'; these being the application of step corrections to the local clock in order to prevent the pseudoranges from becoming significantly different from the actual ranges. Note that a side effect of this is that it is necessary to also apply consistent corrections to the carrier phase and pseudorange-rate terms in order to ensure that all the observations are consistent. Unfortunately the application of these corrections have not been properly validated and for this reason, it is recommended that for carrier phase work, the

compensation should be disabled (i.e. set the flag to 'F'). Programs such as RTKLIB seem to be able to handle the existence of such clock resets, which are apparently routinely applied in other receivers that are on the market.

2. The carrier smoothing time constant determines the level of carrier smoothing that is applied. This can have a significant effect on the precision that is delivered by the receiver because applying more smoothing has the effect of reducing the noise level on the pseudoranges. If the smoothing time constant is set to a value of less than 0.5 seconds then the smoothing will be reset every one second. This means that the pseudoranges (and by association the 'averaged' time of transmission) that are output every one second are completely independent. However, if the smoothing level exceeds 0.5 seconds then smoothing across successive one-second measurement intervals is permitted. This introduces the danger that a glitch or larger pseudorange error will persist longer than a single measurement and could cause difficulties. The higher level of averaging may also cause difficulties for navigation where the receiver is subject to high dynamics or is moving rapidly. Note that the time constant has been chosen to reflect the normal meaning of a time constant, namely the time to reach approximately 63% of the final value. The default value for τ is therefore set to 0.3 seconds reflecting the fact that the smoothing is reset every 1 second.
3. After sending a CFG command, the user may request a CFG report in order to confirm that the specified values have been properly accepted.
4. The receiver can be forced to cold, warm or hot start with appropriate configuration of the flags in field F3. Preventing restoration of almanac and ephemeris will ensure a cold start, while preventing restoration of ephemeris will ensure a warm start (assuming that position & time & TCXO offset are correct). Allowing everything to be restored should allow a hot start to occur.

12.4.6 CHN Report

Description

Receiver channel status

$PNSWR,CHN,*F1,F2,F3,F4,F5,F6,F7*

The sentence is only output if requested and is included for diagnostic purposes only.

Fields

Number		Name	Description
F0		$PNSWR,CHN	Sentence header
F1		x	Sentence number
F2		x	Total number of sentences
F1=1	*F3*	x	Current time of week (ms)
F1≠1	*F3*	x	Hardware channel index
	F4	x	SV allocated to hardware channel
	F5	x	Mode: 0=Idle,1=Search,2=Capture,3=Track
	F6	x	Tracking Mode: 0=Idle,1=CodeLock,2=FreqLock,3=PhaseLock
	F7	x	Noise level (either measured or hardcoded)
	F8	x	Measured C/No (dBHz)
	F9	x	Mean Doppler freq (Hz)
	F10	H	Sync status bits (hex): Bit 0=BitSync, Bit1=FrameSync, Bit2=TimeSync

Remarks

1. The noise level is an amplitude with arbitrary units

Example

```
$PNSWR,CHN,1,13,352701728*01
$PNSWR,CHN,2,13,0,5,3,3,442,51,-3044,3*3F
$PNSWR,CHN,3,13,1,12,3,3,442,49,-3978,3*06
$PNSWR,CHN,4,13,2,15,1,0,442,20,0,0*10
$PNSWR,CHN,5,13,3,18,3,3,442,50,3467,3*2E
...
$PNSWR,CHN,10,13,8,30,3,3,442,49,1972,3*18
$PNSWR,CHN,11,13,9,0,0,0,0,0,0,0*2A
$PNSWR,CHN,12,13,10,193,3,3,442,51,841,3*22
$PNSWR,CHN,13,13,11,0,0,0,0,0,0,0*11
```

12.4.7 CHN Report for GNSS

Description

Receiver channel status

$PNSWR,CHN,*F1,F2,F3,F4,F5,F6,F7*

The sentence is only output if requested and is included for diagnostic purposes only.

Fields

Number		Name	Description
F0		$PNSWR,CHN	Sentence header
F1		x	Sentence number
F2		x	Total number of sentences
F1=1	*F3*	x	Current time of week (ms)
F1≠1	*F3*	x	GNSS system. GPS as 'G' and Galileo as 'E'.
	F3	x	Hardware channel index
	F4	x	SV allocated to hardware channel
	F5	x	Mode: 0=Idle,1=Search,2=Capture,3=Track
	F6	x	Tracking Mode: 0=Idle,1=CodeLock,2=FreqLock,3=PhaseLock
	F7	x	Noise level (either measured or hardcoded)
	F8	x	Measured C/No (dBHz)
	F9	x	Mean Doppler freq (Hz)
	F10	H	Sync status bits (hex): Bit 0=BitSync, Bit1=FrameSync, Bit2=TimeSync

Remarks

2. The noise level is an amplitude with arbitrary units

Example

```
$PNSWR,CHN,9,1,412722997*32
$PNSWR,CHN,G,9,2,0,8,1,0,442,6,0,0*4C
$PNSWR,CHN,G,9,3,1,9,1,0,442,25,0,0*7C
...
$PNSWR,CHN,G,9,8,6,30,1,0,442,27,0,0*48
$PNSWR,CHN,G,9,9,7,31,1,0,442,25,0,0*4B
$PNSWR,CHN,5,1,412723005*3D
$PNSWR,CHN,E,5,2,0,1,1,0,442,16,102,6f*19
...
$PNSWR,CHN,E,5,5,3,4,1,0,442,16,102,6f*18
```

12.4.8 CLN Command

Description

Clear non-volatile memory and real-time-clock

$PNSWR,CLN,*F1*,*F2*

The command is used to clear the non-volatile memory and the real-time-clock

Fields

Number	Name	Description
F0	$PNSWR,CLN	Sentence header
F1	X	A = clear all C = only clear real-time-clock time N = only clear non-volatile memory
F2	X	R = perform a software reset following the operation null field (i.e field is omitted) causes no software reset

Remarks

1. This command is used to clear the receiver non-volatile memory and to clear the time stored in the receiver real-time-clock.

12.4.9 DBG Report

Description

Position failure DBG

Requesting a DBG report causes in a number of sentences to be output. These sentences may help identify the cause of a bad position fix, although the sentences are only available if the firmware has been compiled with the PVTDBG feature enabled.

Fields

Number	Name	Description
F0	$PNSWR,SPV	Sentence header
F1	X	SV number
F2	H	Status word
F3	X	Time tag Week number
F4	x.x	Time tag Time of Week (s)
F5	x.x	Satellite x position coordinate (m)
F6	x.x	Satellite y position coordinate (m)
F7	x.x	Satellite z position coordinate (m)
F8	x.x	Satellite x velocity coordinate(m/s)
F9	x.x	Satellite y velocity coordinate (m/s)
F10	x.x	Satellite z velocity coordinate (m/s)

Remarks

1. The DBG group of sentences was added in order to assist in determining the cause of bad positioning. If the feature is enabled, the receiver keeps a short history of measurements, ephemeris, calculated SV positions, calculated results, etc. In the event of a failure, the input data to the navigation process can be examined and the cause of the bad position identified. This will typically be found to be a measurement blunder of some kind.
2. OBS, EPH, GGA, GRS, GSA sentences are also output.
3. This report is not available on Namuru V32 (Biarri) boards.

12.4.10 ECI Command and Report

Description

Input and output time-tag, position and velocity in ECI coordinates

$PNSWR,ECI,*F1,F2,F3,F4,F5,F6,F7,F8*

The report is triggered following each navigation solution, which occurs after each measurement upload.

Fields

Number	Name	Description
F0	$PNSWR,ECI	Sentence header
F1	X	GPS fix quality (same as GGA report)
F2	X	Modified Julian Date
F3	x.x	ECI x position coordinate (km)
F4	x.x	ECI y position coordinate (km)
F5	x.x	ECI z position coordinate (km)
F6	x.x	ECI x velocity coordinate (km/s)
F7	x.x	ECI y velocity coordinate (km/s)
F8	x.x	ECI z velocity coordinate (km/s)

Remarks

1. This command has been added to allow the GPS receiver to be provided with an initial position and velocity in Earth Centered Inertial (ECI) coordinates using a Modified Julian Date time tag. Such information is typically returned from SGP4 orbit estimators that use NORAD two line elements to calculate the position and velocity of a spacecraft. Given that such an orbit estimator may be found within many satellite flight computers, the use of this command can allow the GPS receiver to avoid performing a cold start when the receiver is powered up. This can substantially reduce the time-to-first-fix (TTFF), which is rather slow for low earth orbit based scenarios.
2. This command has similar fields to those contained within a Colony 2 Bus telemetry message. Converting from a C2B telemetry message to this command should be trivial.
3. Requesting an ECI report results in the GPS receiver reporting its position and time using this coordinate system, using a Modified Julian Date time-tag. If a recent and valid fix is not available, the receiver will report the previously saved ECI position and velocity data, although the Quality flag will be changed to 6 to indicate this. The user can only validate correct receipt of the information by sending the information and requesting a report when the receiver is not performing navigation solutions. This can be ensured by unplugging the antenna.
4. Note that internal to the receiver, the MJD time tags are converted to GPS time, which requires the availability of UTC corrections. Conversion of the time-tags will be incorrect if UTC corrections are not available.
5. Commands must set the Quality field to a non-zero value otherwise the data will be rejected.

Example

$PNSWR,ECI,1,56427.177233785,203.90312,5308.15851,-3518.52708,-0.38711,0.01496,-0.00029*16

12.4.11 EUD/EPL Command and Report

Description

Ephemeris messages

$PNSWR,EUD,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12*

The sentences are triggered whenever a new ephemeris element superseding an existing ephemeris element is extracted from the navigation-message. The output rate is variable, although typically a new ephemeris will be output following a non-volatile memory clearing event until the ephemeris is up to date, at startup if the previously saved ephemeris is out of date or has not been restored or every two hours when new ephemeris becomes available. Three sentences are transmitted for each satellite ephemeris

Fields

Number		Name	Description
F0		$PNSWR,EUD	Sentence header
F1		x	Sentence number (1,2,3)
F2		x	SV PRN number
F1=1	*F3*	HHH	10 bit week (hex)
	F4	H	URA (hex)
	F5	HH	Health
	F6	HH	Tgd : (sec), 2s complement, LSB sf=2^{-31}
	F7	HH	IODC
	F8	HH	Toc (16 bit hex) : (sec), LSB sf=2^{4}
	F9	HH	Af2 : (sec/sec^2), 2s complement, LSB sf=2^{-55}
	F10	HHHH	Af1 : (sec/sec), 2s complement, LSB sf=2^{-43}
	F11	HHHHHH	Af0 : (sec), 2s complement, LSB sf=2^{-31}
F1=2	*F3*	HH	IODE
	F4	HHHH	Crs : (m), 2s complement, LSB sf=2^{-5}
	F5	HHHH	dn : (sc/sec), 2s complement, LSB sf=2^{-43}
	F6	HHHHHHHH	Mo : (sc), 2s complement, LSB sf=2^{-31}
	F7	HHHH	Cuc : (rad), 2s complement, LSB sf=2^{-29}
	F8	HHHHHHHH	e : (dimensionless), unsigned, LSB sf=2^{-33}
	F9	HHHH	Cus : (rad), 2s complement, LSB sf=2^{-29}
	F10	HHHHHHHH	(A) $^{1/2}$: ($\sqrt{m}$), unsigned, LSB sf=2^{-19}
	F11	HHHH	Toe : (sec), unsigned, LSB sf=2^{4}
	F12	x	FitInterval

F1=3	*F3*	HHHH	Cic : (rad), 2s complement, LSB sf=2^{-29}
	F4	HHHHHHHH	$(OMEGA)_0$: (sc), 2s complement, LSB sf=2^{-31}
	F5	HHHH	Cis : (rad), 2s complement, LSB sf=2^{-29}
	F6	HHHHHHHH	Io : (sc), 2s complement, LSB sf=2^{-31}
	F7	HHHH	Crc : (m), 2s complement, LSB sf=2^{-5}
	F8	HHHHHHHH	ω : (sc), 2s complement, LSB sf=2^{-31}
	F9	HHHHHH	OmegaDot : (sc/sec), 2s complement, LSB sf=2^{-43}
	F10	HHHH	IDot : (sc/sec), 2s complement, LSB sf=2^{-43}

Remarks

1. Previously received EUD reports may be re-transmitted back to the receiver in order to restore or aid the receiver if the receiver does not have ephemeris.
2. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E
3. Requesting an EPL report using the GPGPQ sentence causes all of the available ephemeris elements to be transmitted to the user.
4. Sending ephemeris via the serial always replaces existing ephemeris irrespective of whether it is out of date or unusable.

Examples

$PNSWR,EUD,1,26,268,0,00,F3,03E,57E4,00,FF9B,3DF931*14
$PNSWR,EUD,2,26,3E,1466,279D,895557AB,1183,0A7844F3,0AAF,A10D4D02,57E4,0*4E
$PNSWR,EUD,3,26,006F,25BAA3EA,FFA8,2838CA00,24A0,2EFCE30E,FFADA8,0402*03

$PNSWR,EUD,1,193,268,0,3F,F6,0B1,5703,01,FFFC,3FFB54*29
$PNSWR,EUD,2,193,B1,F121,27EF,F4F6ED67,F78D,26DDE2F5,FFCC,CAE81F56,5703,0*0D
$PNSWR,EUD,3,193,FF7E,CF1AAE73,0000,1D014A68,13BE,BFD3E8BA,FFD5B5,3B3D*38

$GPGPQ,EPL
This causes the receiver to transmit its entire list of downloaded ephemeris elements

12.4.12 EUD Command and Report for GNSS

Description

Ephemeris messages

$PNSWR,EUD,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12*

The sentences are triggered whenever a new ephemeris element superseding an existing ephemeris element is extracted from the navigation-message. The output rate is variable, although typically a new ephemeris will be output following a non-volatile memory clearing event until the ephemeris is up to date, at startup if the previously saved ephemeris is out of date or has not been restored or every two hours when new ephemeris becomes available. Three sentences are transmitted for each satellite ephemeris

Fields

Number		Name	Description
F0		$PNSWR,EUD	Sentence header
F1		x	Sentence number (1,2,3)
F2		x	SV PRN number
F1=1	*F3*	HHH	10 bit week (hex)
	F4	H	URA (hex)
	F5	HH	Health
	F6	HH	Tgd : (sec), 2s complement, LSB sf=2^{-31}
	F7	HH	IODC
	F8	HH	Toc (16 bit hex) : (sec), LSB sf=2^{4}
	F9	HH	Af2 : (sec/sec^2), 2s complement, LSB sf=2^{-55}
	F10	HHHH	Af1 : (sec/sec), 2s complement, LSB sf=2^{-43}
	F11	HHHHHH	Af0 : (sec), 2s complement, LSB sf=2^{-31}
	F12	G or E	G= GPS and E=Galileo
F1=2	*F3*	HH	IODE
	F4	HHHH	Crs : (m), 2s complement, LSB sf=2^{-5}
	F5	HHHH	dn : (sc/sec), 2s complement, LSB sf=2^{-43}
	F6	HHHHHHHH	Mo : (sc), 2s complement, LSB sf=2^{-31}
	F7	HHHH	Cuc : (rad), 2s complement, LSB sf=2^{-29}
	F8	HHHHHHHH	e : (dimensionless), unsigned, LSB sf=2^{-33}
	F9	HHHH	Cus : (rad), 2s complement, LSB sf=2^{-29}
	F10	HHHHHHHH	(A) ½ : ($\sqrt{m}$), unsigned, LSB sf=2^{-19}
	F11	HHHH	Toe : (sec), unsigned, LSB sf=2^{4}

	F12	x	FitInterval
F1=3	*F3*	HHHH	Cic : (rad), 2s complement, LSB sf=2^{-29}
	F4	HHHHHHHH	$(OMEGA)_0$: (sc), 2s complement, LSB sf=2^{-31}
	F5	HHHH	Cis : (rad), 2s complement, LSB sf=2^{-29}
	F6	HHHHHHHH	Io : (sc), 2s complement, LSB sf=2^{-31}
	F7	HHHH	Crc : (m), 2s complement, LSB sf=2^{-5}
	F8	HHHHHHHH	ω : (sc), 2s complement, LSB sf=2^{-31}
	F9	HHHHHH	OmegaDot : (sc/sec), 2s complement, LSB sf=2^{-43}
	F10	HHHH	IDot : (sc/sec), 2s complement, LSB sf=2^{-43}

Remarks

5. Previously received EUD reports may be re-transmitted back to the receiver in order to restore or aid the receiver if the receiver does not have ephemeris.
6. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E
7. Requesting an EPL report using the GPGPQ sentence causes all of the available ephemeris elements to be transmitted to the user.
8. Sending ephemeris via the serial always replaces existing ephemeris irrespective of whether it is out of date or unusable.

Examples

$PNSWR,EUD,1,31,272,0,00,00,002,6432,00,0020,02901C,G*09
$PNSWR,EUD,2,31,02,0000,0000,E2554785,0000,03F00000,0000,A10CE300,6432,0*48
$PNSWR,EUD,3,31,0000,1AB62EC7,0000,27EA6763,0000,D9015A13,FFA9F0,0000*07
$PNSWR,EUD,1,32,272,0,00,00,002,6432,00,FFC0,37BFC8,G*79
$PNSWR,EUD,2,32,02,0000,0000,FEA64E5C,0000,063BA000,0000,A10D0400,6432,0*47
$PNSWR,EUD,3,32,0000,C86B1BF9,0000,26ED9613,0000,DFC6F40B,FFA7A1,0000*78
$PNSWR,EUD,1,26,272,7,00,00,000,6351,00,0000,000000,E*71
$PNSWR,EUD,2,26,00,0000,0000,041E3EEC,0000,00000000,0000,AA04B4A4,6351,0*30
$PNSWR,EUD,3,26,0000,1E25C458,0000,27D27D28,0000,00000000,000000,0000*04
$PNSWR,EUD,1,27,272,7,00,00,000,6351,00,0000,000000,E*70
$PNSWR,EUD,2,27,00,0000,0000,20900608,0000,00000000,0000,AA04B4A4,6351,0*34
$PNSWR,EUD,3,27,0000,1E25C458,0000,27D27D28,0000,00000000,000000,0000*05

12.4.13 FWV Report

Description

Provide details on the firmware version

$PNSWR,FWV,F1,F2,F3

Fields

Number	Name	Description
F0	$PNSWR,FWV	Sentence header
F1	x.x	Firmware string descriptor (BIARRI, AQUARIUS, …
F2	x.x	Firmware version number
F3	Xxx	Correlator hardware version number
F4	yyyymmdd_hhmmss	Compile date and time string

Remarks

1. This sentence can be used to determine the version number of the firmware and the correlator hardware
2. This sentence can also be used to determine the date and time the firmware was built (assuming that the last build was a 'clean' build).
3. Normally the firmware string descriptor will be set to BIARRI or AQUARIUS. Other strings indicate experimental firmware versions under test.

Examples

$PNSWR,FWV,BiarriAqrs,1.2,1.06,20120903_113134*54

$PNSWR,FWV,BiarriAqrs_NewGpsDrv,1.4.3,2.1.3,20130124_125555*6B

12.4.14 GPQ Command

Description

Poll for particular reports and enable/disable triggered report output

$GPGPQ,F1,F2

Fields

Number	Name	Description
F0	$GPGPQ	Sentence header
F1	XXX	3 character report identifier
F2	X	D=disable, E=enable C=Clear automatic output of reports R=Restore previously cleared output reports

Remarks

1. This sentence is used for polling a particular output report. Requesting a report will generally result in that report being output regardless of whether automatic output reports have been enabled via the E option.
2. Only one level of history is recorded when using the C/R feature. If a change to the set of automatically output commands and reports is made and a second C command is sent, the state prior to receiving the C command will be restored. Note that for the C/R feature to be used, field F1 should be left blank.

Example

1. Request a GGA sentence

$GPGPQ,GGA

2. Enable output of GGA reports after each navigation solution

$GPGPQ,GGA,E

3. Switch off output of GGA reports after each navigation

$GPGPQ,GGA,D

4. Switch off all automatically output reports

$GPGPQ,,C

5. Restore previously switched off output of automatically output reports

$GPGPQ,,R

12.4.15 GGA Command and Report

Description

Report GPS solution quality, solution time, solution position (latitude, longitude and altitude), dilution of precision (DOP) and satellite count.

$GPGGA,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12*

The sentence is triggered at the end of each navigation solution step, regardless of whether a solution is actually performed (typically 1 Hz).

Fields

Number	Name	Description
F0	$GPGGA	Sentence header
F1	hhmmss.ssssss	GPS time of solution in hours, minutes and seconds
F2	llmm.mmmmm	Latitude in degrees, minutes and fractional minutes
F3	C	Hemisphere descriptor (N=North, S=South)
F4	lllmm.mmmmm	Longitude in degrees, minutes and fractional minutes
F5	C	Hemisphere descriptor (E=East, W=West)
F6	C	Solution quality (0=no solution, 1=good solution)
F7	C	Number of satellites in the solution
F8	x.x	HDOP of solution
F9	x.x	Altitude
F10	M	Units of altitude (m)
F11	0	
F12	M	Units of geoid/spheroid separation (m)

Remarks

1. This implementation outputs GPS time instead of UTC time in field F1
2. This implementation does not include a model for geoid/spheroid separation. As such, it means that the altitude included in field F10 is a spheroidal altitude rather than the geoidal altitude, with the geoid/spheroid separation in field F12 set to 0.
3. If the quality field F6 is 0, the time will stop updating.
4. If the position update stops, the time that is output is the time of the last position solution.
5. At startup, the report shows the position being used for the initial position by the satellite selection (when not in space mode).

Example

$GPGGA,025356.000000,3341.7880,S,15056.2436,E,1,08,0.9,71.4,M,0,M,,*6D

12.4.16 GRS and RRR Reports

Description

Report GPS solution measurement residuals

$GPGRS,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15*

The sentence is triggered at the end of each navigation solution step, regardless of whether a solution is actually performed (typically 1 Hz).

Fields

Number	Name	Description
F0	$GPGRS	Sentence header
F1	hhmmss.ss	GPS time of solution
F2	0	
F3,F4,F5,F6 *F7,F8,F9,F10* *F11,F12,F13,F14* *F15*	x.x	Satellite measurement residuals in (m), where the order corresponds to the same order as given by the GSA report. As a non-standard addition, values that exceed 999 m are converted to km and output as xxxxE3.

Remarks

1. If a GPS satellite residual is not defined then the field will be output as a NULL field.
2. The number of non-null residuals displayed in the GRS sentence will be consistent with the number of satellites used in the navigation solution and the number of satellites contained within the corresponding GSA sentence.
3. Requesting an RRR report results in output of a similar sentence, except that instead of the $GPGRS header, a header of $PNSWR,RRR is used and the data that is output are range-rate residuals (m/s) rather than range residuals (m).

Example

$GPGSA,A,3,28,26,17,15,11,8,7,,,,,,2.2,1.2,1.8*3B

$GPGSV,3,1,9,01,18,091,18,07,35,072,47,08,63,115,47,11,22,110,47*44
$GPGSV,3,2,9,15,13,222,42,17,46,348,51,24,31,069,44,26,45,240,44*44
$GPGSV,3,3,9,28,65,207,50*79

$GPGRS,073622.00,0,-2.4,+2.8,+1.0,+2.1,-0.4,-0.1,+1.9,,,,,,*61

12.4.17 GSA Report

Description

Report satellites used and Dilution of Precision (DOP).

$GPGSA,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17*

The sentence is triggered at the end of each navigation solution step, regardless of whether a solution is actually performed (typically 1 Hz).

Fields

Number	Name	Description
F0	$GPGSA	Sentence header
F1	A	
F2	X	2 for 2D solution, 3 for 3D solution
F3, F4,F5,F6,F7 F8,F9,F10,F11 F12,F13,F14	X	Satellite numbers, NULL fields for unused slots
F15	x.x	PDOP
F16	x.x	HDOP
F17	x.x	VDOP

Remarks

1. If the GPS quality is zero and the DOP is not defined, then three NULL fields will be output instead.

12.4.18 GSA Report for Galileo only

Description

Report Galileo only satellites used and Dilution of Precision (DOP). A preparatory command used by Namuru V3.3 and V2.4.

$PNSWR,GPGSA,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17*

The sentence is triggered at the end of each navigation solution step, regardless of whether a solution is actually performed (typically 1 Hz).

Fields

Number	Name	Description
F0	$PNSWR,GNSA	Sentence header
F1	A	
F2	X	2 for 2D solution, 3 for 3D solution
F3, F4,F5,F6,F7 F8,F9,F10,F11 F12,F13,F14	X	Satellite numbers, NULL fields for unused slots
F15	x.x	PDOP
F16	x.x	HDOP
F17	x.x	VDOP

Remarks

2. If the GNSA quality is zero and the DOP is not defined, then three NULL fields will be output instead.

12.4.19 GSV Report

Description

Report GPS visible satellites, including elevation, azimuth and signal strength.

$GPGSV,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17,F18*

The sentence is triggered at the end of each navigation solution step, regardless of whether a solution is actually performed (typically 1 Hz).

Fields

Number	Name	Description
F0	$GPGSV	Sentence header
F1	X	Total number of sentences
F2	X	Sentence number
F3	x	Total number of satellites visible
F4,F5,F6,F7 F8,F9,F10,F11, F12,F13,F14,F15, F16,F17,F18,F19	x,x,x,x	Group of 4 values, where each group includes a: 1) Satellite number 2) Satellite elevation (degrees) 3) Satellite azimuth (degrees) 4) Satellite signal strength (dBHz)

Remarks

1. If the satellite azimuth and elevation are unknown, then NULL fields are output. This can occur at startup before a navigation solution has taken place or before the satellite almanac/ephemeris has been received.

12.4.20 ION Command and Report

Description

Report ionospheric correction coefficients transmitted in the GPS navigation message

$PNSWR,ION,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10*

The sentence is triggered whenever new ionospheric corrections are extracted from the GPS navigation message, although the user may poll for the current set of corrections by manually requesting an ION report.

Fields

Number	Name	Description
F0	$PNSWR,ION	Sentence header
F1	HHH	WNt field
F2	HH	Time of Ionospheric Corrections & 0xFF
F3	HH	$\alpha(0)$ & 0xFF : (sec), 2s complement, LSB sf=2^{-30}
F4	HH	$\alpha(1)$ & 0xFF : (sec/sc), 2s complement, LSB sf=2^{-27}
F5	HH	$\alpha(2)$ & 0xFF : (sec/sc^2), 2s complement, LSB sf=2^{-24}
F6	HH	$\alpha(3)$ & 0xFF: (sec/sc^3), 2s complement, LSB sf=2^{-24}
F7	HH	$\beta(0)$ & 0xFF : (sec), 2s complement, LSB sf=2^{11}
F8	HH	β (1) & 0xFF : (sec/sc), 2s complement, LSB sf=2^{14}
F9	HH	β (2) & 0xFF : (sec/sc^2), 2s complement, LSB sf=2^{16}
F10	HH	β (3) & 0xFF: (sec/sc^3), 2s complement, LSB sf=2^{16}

Remarks

1. The unsigned hexadecimal values output in this report exactly match the fields transmitted by the satellites in the navigation message.
2. Output of this report is necessary for the construction of RINEX navigation data files because the header portion of a RINEX navigation file includes the ionospheric correction terms.
3. Sending a previously received ION report allows the ionospheric corrections within the receiver to be restored.
4. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E.

12.4.21 NAV Report

Description

Report raw navigation report for examination by the user.

$PNSWR,NAV,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13*

The sentences are triggered at the end of each subframe when a new subframe is ready to be decoded, although only if support for the sentence has been switched on at compile time.

Fields

Number	Name	Description
F0	$PNSWR,NAV	Sentence header
F1	x	Channel number
F2	x	Space Vehicle (SV) number
F3	HHH	3 hex digit status word, where each bit indicates whether the associate word number was received with good parity (MSB refers to W1, LSB to W10)
F4, F5, F6, F7, F8, F9,F10,F11,F12,F13	HHHHHHHH	Hexadecimal navigation message data words, where F4 is word 1, F5 is word 2, …

Remarks

1. Support for this report must be switched on at compile time
2. The report has been included for information only and is not strictly necessary.
3. Adding this report consumes additional memory resources that would otherwise be available.

Example

$PNSWR,NAV,00,02,3FF,22C2C03E,88166B18,001402FB,FD8650B4,0013C986,90A11578,074EA4DE,996EA2A5,401752D1,7F411090*2C

$PNSWR,NAV,01,04,3FF,22C2C03E,88166B18,000A42E2,AD9EF621,400E7652,8E1A7DDC,0713C9D1,788FBE08,3FE9D1E0,0EBF4608*2B

$PNSWR,NAV,03,09,3FF,22C2C03E,88166B18,3FE2A2D5,57A64F1B,FFCC75C8,0767DF3E,897F908E,AE39658A,BFE9A837,F0C01E24*5B

$PNSWR,NAV,04,12,0,162C6B6F,D1DD5495,76F0DE39,711E1542,93387E78,167FC4C0,019F75B9,4FC726B1,484AE426,B2462B3E*20

$PNSWR,NAV,06,24,3FF,22C42D32,88166B18,0004A2E1,5F747C73,FFFE3634,05FA5335,76D7FEE7,C31693C4,3FE95943,EE002B3C*53

12.4.22 OBS Report

Description

Report raw GPS measurement data set; this comprising a time-of-reception sentence and a group of satellite observation sentences.

$PNSWR,OBS,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15*

The sentence is triggered whenever new GNSS measurement set is uploaded by the measurement task to the navigation solution task (typically 1 Hz).

Fields

Number		Name	Description
F0		$PNSWR,OBS	Sentence header
F1		X	Number of sentences
F2		X	Sentence number
F2=1	*F3*	X	Issue of data, incrementing measurement ID
	F4	x.x	GPS time of reception (TOR) (s), (seconds of the week)
	F5	X	GPS time of reception (week number)
	F6	X	Delta TOR – this field is non-zero if a step correction is applied to the TOR
	F7	X	Currently applied frequency offset (ppb)
	F8	X	Increment/adjustment to frequency offset (ppb)
	…	…	*Other fields may be appended in the future.*
F2≠1	*F3*	X	Frequency band: Fixed as L1 for Namuru V3.2
	F4	X	SV number
	F5	X	Channel number
	F6	H	Hex status word for observation
	F7	x.x	GPS time-of-transmission (TOT) (s),(seconds of the week)
	F8	x.x	Pseudorange (m)
	F9	x.x	Pseudorange-rate (m/s)
	F10	x.x	Carrier phase (cycles)
	F11	X	SV elevation (degrees)
	F12	X	SV azimuth (degrees)
	F13	X	C/N0 (dBHz)
	F14	HHH	Debugging hexadecimal flags indicating tracking and synchronization status
	F15	x.x	Pseudorange variance (m^2)

Remarks

1. OBS reports are required for the creation of RINEX observation files
2. An OBS report is output every one second at the second boundary
3. The measurement ID can be used to obtain a rough idea of the amount of time the receiver has been running for, since each second the receiver generates a new measurement and increments the count.
4. The 'Hex status word for observation' (field F5) is a useful field to examine when trying to determine why a receiver is not positioning when the observations appears to be present. For an observation to be acceptable for use within the navigation solution (least squares one-shot or Kalman filter), the observation has to have good tracking, have good synchronization and have orbital model parameters (indicated by the asterisk in the following table) that allow the receiver to calculate the position and velocity of each transmitting spacecraft.
   The table below provides the meanings associated with some of these flags (subject to change without notice):

Bit	Hex Mask	Interpretation
0 *	0x001	Observation OK
1 *	0x002	Synchronisation OK
2 *	0x004	Orbital parameters (ephemeris) available
3	0x008	PLL is phase locked
4	0x010	Parity is known (required for CPH)
5	0x020	Set when CPH is being accumulated
6	0x040	Set if a ½ cycle correction has been added to the CPH (as determined by the parity)
7	0x080	Set if loss of lock occurs, as required for RINEX
8	0x100	Possible cycle slip may have occurred.
9	0x200	dTsv clock corrections applied (bias, drift, aging, single frequency group delay, relativitistic)
10	0x400	Range blunder detected
11	0x800	Range-rate blunder detected

5. The 'Debugging hexadecimal flags indicating tracking and synchronization status' (field F13) are also useful flags to examine if the receiver is failing to navigate. Three 4-bit values are concatenated together in this field that indicate important tracking and synchronization information. Some of this information is also provided in the 'CHN' report.
   The first (most significant) nibble indicates the mode of the tracking and acquisition processor (0=idle, 1=searching/acquisition, 2=capture, 3=tracking), the second nibble indicates tracking mode when the processor is tracking (0=no-code-lock, 1=code lock, 2=frequency lock, 3=phase lock), while the last (least significant) nibble is a bit mask that indicates the synchronization status of the channel (0x001=bitsync, 0x002=time-sync, 0x004=frame-sync).

6. In the example that follows, it will be observed that when the channel is operating normally, the value 0x337 is output indicating that the channel is tracking, has phase-lock and full synchronization.

Example

$PNSWR,OBS,9,1,432,286581.997741325,1711,0,0,0,0,0*2E
$PNSWR,OBS,9,2,L1,7,1,07f,286581.924985039,21811785.719,130.781,196723.185,35,072,47,337,32.9*21
$PNSWR,OBS,9,3,L1,8,2,07f,286581.930410418,20185298.000,-163.805,-266137.730,63,115,47,337,32.9*2D
$PNSWR,OBS,9,4,L1,11,3,07f,286581.920130218,23267224.375,-188.438,-311672.999,22,110,47,337,32.9*14
$PNSWR,OBS,9,5,L1,15,4,07f,286581.918997379,23606841.047,-826.562,-834720.321,13,222,42,337,32.9*13
$PNSWR,OBS,9,6,L1,17,5,07f,286581.928906734,20636091.203,-824.742,-1306874.470,46,348,51,337,32.9*29
$PNSWR,OBS,9,7,L1,24,6,07f,286581.923396835,22287917.438,159.773,156634.739,31,069,44,337,32.9*1F
$PNSWR,OBS,9,8,L1,26,7,07f,286581.926569072,21336904.703,-607.484,-914043.338,45,240,44,337,32.9*1A
$PNSWR,OBS,9,9,L1,28,8,07f,286581.932899281,19439155.812,-259.523,-418035.578,65,207,50,337,32.9*16

12.4.23 OBS Report for GNSS

Description

Report raw GPS measurement data set; this comprising a time-of-reception sentence and a group of satellite observation sentences.

$PNSWR,OBS,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15*

The sentence is triggered whenever new GNSS measurement set is uploaded by the measurement task to the navigation solution task (typically 1 Hz).

Fields

Number		Name	Description
F0		$PNSWR,OBS	Sentence header
F1		X	Number of sentences
F2		X	Sentence number
F2=1	*F3*	X	Issue of data, incrementing measurement ID
	F4	x.x	GPS time of reception (TOR) (s), (seconds of the week)
	F5	X	GPS time of reception (week number)
	F6	X	Delta TOR – this field is non-zero if a step correction is applied to the TOR
	F7	X	Currently applied frequency offset (ppb)
	F8	X	Increment/adjustment to frequency offset (ppb)
	…	…	*Other fields may be appended in the future.*
F2≠1	*F3*	X	GNSS system. GPS as 'G' and Galileo as 'E'.
	F4	X	Frequency band: L1 as 1 and L5 as 5
	F5	X	SV number
	F6	X	Channel number
	F7	H	Hex status word for observation
	F8	x.x	GPS time-of-transmission (TOT) (s),(seconds of the week)
	F9	x.x	Pseudorange (m)
	F10	x.x	Pseudorange-rate (m/s)
	F11	x.x	Carrier phase (cycles)
	F12	X	SV elevation (degrees)
	F13	X	SV azimuth (degrees)
	F14	X	C/N0 (dBHz)
	F15	HHH	Debugging hexadecimal flags indicating tracking and synchronization status

	F16	x.x	Pseudorange variance (m^2)

Remarks

7. OBS reports are required for the creation of RINEX observation files
8. An OBS report is output every one second at the second boundary
9. The measurement ID can be used to obtain a rough idea of the amount of time the receiver has been running for, since each second the receiver generates a new measurement and increments the count.
10. The 'Hex status word for observation' (field F7) is a useful field to examine when trying to determine why a receiver is not positioning when the observations appears to be present. For an observation to be acceptable for use within the navigation solution (least squares one-shot or Kalman filter), the observation has to have good tracking, have good synchronization and have orbital model parameters (indicated by the asterisk in the following table) that allow the receiver to calculate the position and velocity of each transmitting spacecraft.
    The table below provides the meanings associated with some of these flags (subject to change without notice):

Bit	Hex Mask	Interpretation
0 *	0x001	Observation OK
1 *	0x002	Synchronisation OK
2 *	0x004	Orbital parameters (ephemeris) available
3	0x008	PLL is phase locked
4	0x010	Parity is known (required for CPH)
5	0x020	Set when CPH is being accumulated
6	0x040	Set if a ½ cycle correction has been added to the CPH (as determined by the parity)
7	0x080	Set if loss of lock occurs, as required for RINEX
8	0x100	Possible cycle slip may have occurred.
9	0x200	dTsv clock corrections applied (bias, drift, aging, single frequency group delay, relativitistic)
10	0x400	Range blunder detected
11	0x800	Range-rate blunder detected

11. The 'Debugging hexadecimal flags indicating tracking and synchronization status' (field F13) are also useful flags to examine if the receiver is failing to navigate. Three 4-bit values are concatenated together in this field that indicate important tracking and synchronization information. Some of this information is also provided in the 'CHN' report.
    The first (most significant) nibble indicates the mode of the tracking and acquisition processor (0=idle, 1=searching/acquisition, 2=capture, 3=tracking), the second nibble indicates tracking mode when the processor is tracking (0=no-code-lock, 1=code lock, 2=frequency lock, 3=phase lock), while the last (least significant) nibble is a bit mask that indicates the synchronization status of the channel (0x001=bitsync, 0x002=time-sync, 0x004=frame-sync).

12. In the example that follows, it will be observed that when the channel is operating normally, the value 0x337 is output indicating that the channel is tracking, has phase-lock and full synchronization.

Example

$PNSWR,OBS,12,1,1,406841.091000000,1650,0,0,0,0,0.0000*3C
$PNSWR,OBS,12,2,G,1,3,1,00d,29.672021942,-0000000.016,-577.562,0.000,25,354,46,331,225.0*01
$PNSWR,OBS,12,3,G,1,6,2,00d,29.671474975,-0000000.016,-508.375,0.000,28,009,46,331,225.0*0B
$PNSWR,OBS,12,4,G,1,16,3,084,29.657709476,-0000000.016,-777.656,0.000,59,348,12,100,2025.0*58
$PNSWR,OBS,12,5,G,1,22,4,084,29.650603101,-0000000.016,-47.836,0.000,48,050,06,100,2025.0*63
$PNSWR,OBS,12,6,G,1,30,5,084,29.643389037,-0000000.016,-313.938,0.000,78,104,18,100,2025.0*52
$PNSWR,OBS,12,7,G,1,31,6,01f,406841.015393027,22666400.453,89.148,0.000,27,176,46,337,33.5*0B
$PNSWR,OBS,12,8,G,1,32,7,01f,406841.007627164,24994547.484,-1015.219,0.000,22,207,46,337,33.5*2B
$PNSWR,OBS,12,9,E,1,5,8,094,406840.984916047,31803168.922,-307.008,0.000,00,673,18,107,2025.0*7A
$PNSWR,OBS,12,10,E,1,6,9,094,406840.964660302,37875688.625,-307.008,0.000,02,569,12,107,2025.0*44
$PNSWR,OBS,12,11,E,1,7,10,094,406840.962415002,38548812.500,-307.008,0.000,1078,000,18,107,2025.0*70
$PNSWR,OBS,12,12,E,1,8,11,094,406840.961064559,38953665.312,-307.008,0.000,1103,512,16,107,2025.0*7F

12.4.24 PPS Report

Description

Pulse per second report indicating current estimate of local clock corrections

$PNSWR,PPS,*F1,F2,F3,F4,F5*

The report is triggered by the firmware used to trigger the coarse 1 PPS timing pulse that drives one of the debugging LEDs on the board (1 Hz).

Fields

Number	Name	Description
F0	$PNSWR,PPS	Sentence header
F1	x	GPS week number
F2	x	GPS time of week (seconds)
F3	x	Clock correction time tag (seconds of week, ms)
F4	x	Clock correction offset (s)
F5	x	Clock correction drift (s/s)

Remarks

1. This report was added to allow a sentence describing the most recent pulse per second to be output, although the time-tag that is output in fields F1 and F2 is a coarse software maintained quantity.
2. The set of corrections in fields F3, F4 and F5 are the same corrections that are used to correct the local clock.
3. While writing this document and reviewing the code, it crosses ones mind that perhaps the value output in this sentence should be the TOR at the measurement instant, since this corresponds to the PPS in any case.

12.4.25 RTC Report

Description

Real time clock (RTC) report indicating the state of the RTC.

$PNSWR,RTC,*F1,F2,F3,F4,F5,F6,F7,F8*

The report is not triggered by any event, but has been added to allow the user to manually poll the receiver in order to determine the state of the RTC. The format of this report is dependent on the underlying hardware,

Fields

Number	Name	Description
F0	$PNSWR,RTC	Sentence header
F1	hhmmss.ss	
F2	dd	Day of month (1-31)
F3	mm	Month (1 – 12)
F4	yyyy	Year (1980+)
F5	w	Week number
F6	t	Time of week (ms)
F8	X	Raw RTC count

Remarks

1. This report was added to allow the user to determine the state of the RTC. The data output within this sentence provides information on the time stored within the RTC

Example

PNSWR,RTC,090643.55,03,09,2012,1704,119203550,1030698403*18

12.4.26 SEL Command and Report

Description

The SEL command is used to force or excluded the selection of a particular satellite set.

$PNSWR,SEL,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12*

The report is not triggered by any event, but has been added to allow the user to manually poll the receiver in order to determine the state of the SEL command.

Fields

Number	Name	Description
F0	$PNSWR,SEL	Sentence header
F1	x	Set type: F=Forced set, X=Excluded set
F2	x	Constellation type: G=GPS, Q=QZSS, S=SBAS, R=GLONASS … (Only G & Q are implemented)
F3	SV1	1st SV in set
F4	SV2	2nd SV in set
F5	SV3	3rd SV in set
F6	SV4	4th SV in set
F7	SV5	5th SV in set
F8	SV6	6th SV in set
F9	SV7	7th SV in set
F10	SV8	8th SV in set
F11	SV9	9th SV in set
F12	SV10	10th SV in set
…	…	…

Remarks

1. This command was added to allow a particular satellite set to be forced to be used or to ensure that particular satellites are excluded from use. This may be useful during development.
2. When requested as a report, two sentences are transmitted for each constellation (new version of FW only). One refers to the excluded set, while the other refers to the forced set.
3. The list is terminated by a NULL field. (ie. the last character in the sentence will be a comma).
4. Each constellation is considered independently of the rest, so excluded GPS satellites cannot be referred to in the same sentence as say excluded QZSS satellites.

5. Any number of satellites may be listed in this sentence, subject to not exceeding the maximum sentence length of 80 characters. This is denoted by the use of ellipsis (...) in the table above. Remember that the fields are simply separated by commas.

12.4.27 SPT Command and Report

Description

Serial port settings for the receiver

$PNSWR,SPT*F1,F2,F3,F4,F5,F6*

The report is not triggered by any event, but has been added to allow the user to change the serial port settings and view the current serial port settings.

Fields

Number	Name	Description
F0	$PNSWR,SPT	Sentence header
F1	x	UART number, 1 = RS422, 2=RS232
F2	x	Baud rate (bps), 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 23400
F3	x	Data bits
F4	x	Parity (N=no parity)
F5	x	Stop bits
F6	x	Flow control (0=no flow control)

Remarks

1. This report was added to allow the user to change the serial port speed. Other fields, although reported are not currently changeable.
2. Commands used to change the speed result in new settings being saved to non-volatile memory. The new results take effect following the next reset.
3. This command is not available on Namuru V3.2 Biarri receivers.

Example

$PNSWR,SPT,1,115200,8,N,1,0*72

$PNSWR,SPT,2,115200,8,N,1,0*71

12.4.28 STK Report

Description

Stack usage report.

$PNSWR,STK,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13,F14,F15,F16,F17*

The report is not triggered by any event, but has been added to allow the user to manually poll the receiver in order to determine the receiver stack usage.

Fields

Number	Name	Description
F0	$PNSWR,STK	Sentence header
F1	x.x	Remaining stack for 'TestTask1'
F2	x.x	Remaining stack for 'UartTask'
F3	x.x	Remaining stack for 'PreDspTask'
F4	x.x	Remaining stack for 'TrackerTask'
F5	x.x	Remaining stack for 'RtcTask'
F6	x.x	Remaining stack for 'WatchTask'
F7	x.x	Remaining stack for 'MsmtTask'
F8	x.x	Remaining stack for 'BitExtractTask'
F9	x.x	Remaining stack for 'ExtractTask'
F10	x.x	Remaining stack for 'CmdTask'
F11	x.x	Remaining stack for 'ReportTask'
F12	x.x	Remaining stack for the 'TLETask'
F13	x.x	Remaining stack for 'AutoTestTask'
F14	x.x	Remaining stack for 'TestTask3'
F15	x.x	Remaining stack for 'PVTTask'
F16	x.x	Remaining stack for 'SvDbTask
F17	x.x	Remaining stack for 'SvSlctTask'

Remarks

1. This report was added to allow the user to determine whether a stack overrun has occurred on any of the receiver tasks. The remaining stack size (units of bytes) for each task is listed in this message. Adjustments can be made if the margins are getting low.
2. An overrun on the stack is generally denoted by a large positive value (i.e. a 2's complement negative value printed out as an unsigned positive value).
3. A value of 0 may indicate that the task in question is not running and as such, no overrun has taken place. For example, the 'NamuruV2R4' hardware running Aquarius firmware and configured to use the dual-FIFO serial port will not have an entry for the 'UartTask' and this entry will be set to 0.
4. Remaining stack sizes are printed in decimal

12.4.29 TCO Command and Report

Description

Command to set the local oscillator or TCXO offset (in ppb) and to report an estimate of the current TCXO offset.

$PNSWR,TCO,*F1,F2*,

The report is not triggered by any event, but has been added to allow the user to manually poll the receiver in order to determine the state of the TCXO offset.

Fields

Number	Name	Description
F0	$PNSWR,TCO	Sentence header
F1	x.x	Local oscillator offset (ppb) as stored in the receivers internal database
F2	x.x	Estimate of the local oscillator offset (ppb) as calculated from the difference between the measured Doppler frequency and the estimated Doppler frequency given in the acquisition assistance.

Remarks

1. This report was added to allow the user to determine the state of the local oscillator offset. When the receiver is navigating successfully, the local oscillator offset can be noted in case a cold start is necessary and a faster TTFF is desired.
2. If the receiver is failing to acquire, it is often useful to examine the local oscillator offset in order to check whether the value being used by the software is sensible. Values that exceed the values expected given the drift characteristics of the TCXO device might indicate that a navigation error resulted in the correct offset being corrupted in some way.

Example

$PNSWR,TCO,+1554,01515*22

12.4.30 TIM Command and Report

Description

Command to configure the behavior of the hardware timing pulse and to trigger output of a timing pulse by serial port command when software triggering of timing pulses has been selected.

$PNSWR,TIM,*F1,F2,F3,F4,F5*

The report is not triggered by any event, but has been added to allow the user to manually poll the receiver in order to determine the state of the timing pulse setup.

Fields

Number	Name	Description
F0	$PNSWR,TIM	Sentence header
F1	x	Set to 1 to trigger a HW pulse using this sentence Set to 0 or leave field NULL to avoid triggering a HW timing pulse
F2	x	Number of hardware pulses to issue -1 = Output pulses forever 0, 1, 2, … = Number of pulses to output
F3	x	Pulse width in us (hard coded to 1000 us in Biarri)
F4	x	Hardware timing pulse trigger type: 0 = No PPS Enable (i.e. always output PPS pulse) 1 = Message PPS Enable (i.e. send a TIM command) 2 = Hardware PPS Enable (no implemented)
F5	x	Hardware timing pulse polarity (not currently supported with Biarri)

Remarks

1. This report was added to allow the user to determine the state hardware timing pulse configuration.
2. The number of timing pulses following a trigger can be set using field F2. For the BIARRI platform, this will default to a single pulse.
3. The triggering of pulses can be defined by the user rather than pulses being output automatically once the receiver is navigating.

Example

Output timing pulses automatically forever (once the receiver is navigating), each timing pulse is 1000 us in width
$PNSWR,TIM,,-1,+1000,0,P*4E
Trigger a hardware timing pulse by sending a software message

$PNSWR,TIM,1

12.4.31 TLE Commands and Reports

Description

Provide orbital Two Line Elements (TLE) to the receiver and perform testing of the TLE feature.

$PNSWR,TLE,*F1,F2,F3,F4,F5,F6,F7,F8,F9*

Fields

Number		Name		Description
F0		$PNSWR,TLE		Sentence header
F1		x		Sentence number
F1=1	*F2*	eeeec		NORAD (TLE) satellite number and classification string
	F3	x		Satellite international designator string
	F4	yyddd.d		Epoch year and (fractional) day
	F5	±.dddddddd		Mean motion derivative (NORAD TLE format)
	F6	±ddddd±d		Mean motion 2nd derivative plus exponent term (NORAD TLE format)
	F7	±ddddd±d		TLE B* drag term (NORAD TLE format)
	F8	…		TLE ???
	F9	ddddc		Element number and a single digit checksum
F1=2	*F2*	eeee		NORAD (TLE) satellite number
	F3	dd.d		Inclination (degrees)
	F4	dd.d		RA of ascending node (degrees)
	F5	ddddddd		Eccentricity (scaled by 10^7)
	F6	dd.d		Argument of perigee (degrees)
	F7	dd.d		Mean anomaly (degrees)
	F8	dd.dd…ddrrrrrc		Mean motion (degrees), Epoch revolutions and a single digit checksum
F1=3	*F2*	c		*Output coordinate system – FIELD IGNORED W=WGS84 ECEF, T=TEME ECI, J=J2000 ECI*
	F3	t		Time-tag type U=UTC, G=GPS, J=Julian Date, T=Two Line Element format, D=Offset from TLE
	F4	F3=U F3=G F3=J F3=T	yyyymmdd wwww ddddddddd.dd yymm.mmm	Time tag in the format defined by F4

		F3=D	t.t	
	F5	F3=U F3=G F3=J F3=T F3=D	hhmmss.s tttttttt - - -	
F1=4 *F1=5*	*F2*	Mode		Mode and error flag returned by 'init_sgdp4' and 'satpos_xyz'
	F3	ddddddddd		Julian Date time-stamp
	F4	X		X (m), ECI if F1=4 or WGS84 ECEF if F1=5
	F5	Y		Y (m), ECI if F1=4 or WGS84 ECEF if F1=5
	F6	Z		Z (m), ECI if F1=4 or WGS84 ECEF if F1=5
	F7	XDot		XDot (m/s), ECI if F1=4 or WGS84 ECEF if F1=5
	F8	YDot		YDot (m/s), ECI if F1=4 or WGS84 ECEF if F1=5
	F9	ZDot		ZDot (m/s), ECI if F1=4 or WGS84 ECEF if F1=5

Remarks

1. The fields in this report are derived directly from the TLE elements published by NORAD, where NMEA 'comma' delimiters substitute directly into the 'space' delimiters used by NORAD. This was done in order to make it as simple as possible to convert NORAD TLE elements into the required format. The conversion is best illustrated using an example.

   A set of two line lines extracted from the collection of test cases is shown immediately below. The first line is a comment line, denoted with a starting # character and the remaining two lines are the first and second elements of the NORAD two line elements. Immediately after these is the conversion of the published two line elements into two Aquarius TLE commands. It can be seen that each sentence may be constructed by replace 'space' delimiters with 'comma' delimiters and pre-pending the NMEA '$PNSWR,TLE' prefix.

   ```
   #DELTA 1 DEB          # Near Earth normal drag equ. (perigee = 377.26km, ...)
   1  06251U  62025A       06176.82412014  +.00008885  +00000-0  +12808-3  0  03985
   2  06251   058.0579   054.0425   0030035   139.1568   221.1854   15.56387291006774

   $PNSWR,TLE,1,06251U,62025A,06176.82412014,+.00008885,+00000-0,+12808-3,0,03985
   $PNSWR,TLE,2,06251,058.0579,054.0425,0030035,139.1568,221.1854,15.56387291006774
   ```

2. Documentation on the NORAD TLE format may be found at:
   http://celestrak.com/NORAD/documentation/tle-fmt
   http://celestrak.com/columns/v04n03/

3. Requesting a TLE report after sending TLE commands containing TLE elements results in the TLE elements being echoed back to the user. This allows the user to confirm correct reception of the elements.

4. To support testing of this feature, the user may send a TLE command containing an output format and a time-tag (in a variety of formats). The GPS receiver will then use the time-tag to calculate an orbital position using the TLE orbital parameters and the SGP4/SDP4 orbit model, which can be verified against a calculation performed elsewhere. When the receiver has calculated the satellite position using the orbit model, TLE sentences are output containing the calculated position and velocity vectors of the satellite.

5. Aquarius uses the Dundee SGP4 C code to perform these calculations. http://www.sat.dundee.ac.uk/~psc/sgp4.html

6. The output coordinate system (input command 3, field 2) is currently ignored.

7. For the TLE feature to be effective, the receiver needs to have been powered off for longer than 15 minutes, have current almanac, current time and have the TLE elements. The receiver must also be in 'Space' mode. The 15 minute requirement stems from a feature whereby the unit will propagate a previously saved position/velocity using a very simple orbit propagator if the previously saved position/velocity is not too old.

Example

1. Send TLE elements to the receiver, which in this case correspond to the test case for spacecraft 6251 described in the paper. paper "Revisiting Spacetrack Report #3" by David A. Vallado, Paul Crawford, Richard Hujsak and T. S. Kelso (AIAA 2006-6753).
   $PNSWR,TLE,1,06251U,62025A,06176.82412014,+.00008885,+00000-0,+12808-3,0,03985
   $PNSWR,TLE,2,06251,058.0579,054.0425,0030035,139.1568,221.1854,15.56387291006774
2. Request TLE elements for verification
   $GPGPQ,TLE
   and receive
   $PNSWR,TLE,1,06251U,62025A,06176.82412014,+.00008885,+00000-0,+12808-3,0,03980*29
   $PNSWR,TLE,2,06251,058.0579,054.0425,0030035,139.1568,221.1854,15.56387291006770*2B
3. Request calculation of satellite using TLE at TLE reference time plus 0 seconds and request (after a few seconds) a TLE report. Because the most recent TLE command was not a set of orbital elements, what is output is the result of using the TLE elements with the specified time to calculate an orbital position and velocity vector. Alternatively, if TLE reports are enabled before sending the calculation time then the outputs will be transmitted automatically. This avoids the need to wait for the calculation to complete.
   $PNSWR,TLE,3,W,D,0
   GPGPQ,TLE
4. Observe output
5. $PNSWR,TLE,4,3,2453912.32412014,3988310,5498967,901,-3290,2358,6497*18
   $PNSWR,TLE,5,3,2453912.32412014,-6226938,-2714865,901,1441,-3247,6497*16

   It can be seen that the ECI results contained within the first of these two sentence agrees with the test cases described in the above paper.

6251 xx

Min from epoch	position x km	position y km	position z km	vel km/s	vel km/s	vel km/s	year	mon	day	hr	min	sec
0.00000000	3988.31022699	5498.96657235	0.90055879	-3.290032738	2.357652820	6.496623475	2006	6	25	19	46	43.98??

12.4.32 UTC Command and Report

Description

Report UTC correction coefficients transmitted in the GPS navigation message

$PNSWR,UTC,*F1,F2,F3,F4,F5,F6,F7,F8*

The sentence is triggered whenever new UTC corrections are extracted from the GPS navigation message, although the user may poll for the current set of corrections by manually requesting an UTC report.

1.1.1 Fields

Number	Name	Description
F0	$PNSWR,UTC	Sentence header
F1	HHHHHH	A1 : (sec/sec), 2s complement, LSB sf=2^{24}
F2	HHHHHHHH	A0 : (sec), 2s complement, LSB sf=2^{-30}
F3	H	dTLS : (sec), 2s complement
F4	HH	Tot : (sec), 2s complement, LSB sf=2^{12}
F5	H	WNt : (weeks)
F6	H	WNLSF : (weeks)
F7	H	DN : (days)
F8	H	dTLSF : (sec), 2s complement

Remarks

1. The unsigned hexadecimal values output in this report exactly match the fields transmitted by the satellites in the navigation message.
2. Output of this report is necessary for the construction of RINEX navigation data files because the header portion of a RINEX navigation file includes the UTC correction terms.
3. Sending a previously received UTC report allows the UTC corrections within the receiver to be restored.
4. The hexadecimal output quantities are as contained within the corresponding navigation message and have the scaling described by IS-GPS-200-E.

12.4.33 VTG Report

Description

Course over ground NMEA sentence

$GPVTG,*F1,F2,F3,F4,F5,F6,F7,F8*

The report is triggered by each navigation solution update and outputs the receiver horizontal velocity vector

Fields

Number	Name	Description
F0	$GPVTG	Sentence header
F1	x.x	True course over ground/Heading (degrees)
F2	T	True north
F3		Magnetic course over ground/Heading (deg) Not implemented, NULL field
F4	M	Magnetic north
F5	x.x	Horizontal speed (knots)
F6	N	N=knots
F7	x.x	Horizontal speed (km/hr)
F8	K	K=km/hr

Remarks

1. This report implements the NMEA VTG report.
2. The magnetic course over ground has not been implemented, so field F3 is left NULL.

Example

$GPVTG,262.4,T,,M,00.1,N,00.1,K*62

12.4.34 XYZ Report

Description

Output time-tag, position and velocity in WGS84 XYZ coordinates

$PNSWR,XYZ,*F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11*

The report is triggered following each navigation solution, which occurs after each measurement upload.

Fields

Number	Name	Description
F0	$PNSWR,XYZ	Sentence header
F1	x	GPS fix quality (same as GGA report)
F2	x	GPS Week Number
F3	x.x	GPS Time of Week (seconds)
F4	x.x	WGS84 x position coordinate (m)
F5	x.x	WGS84 y position coordinate (m)
F6	x.x	WGS84 z position coordinate (m)
F7	x.x	WGS84 x velocity coordinate (m/s)
F8	x.x	WGS84 y velocity coordinate (m/s)
F9	x.x	WGS84 z velocity coordinate (m/s)
F10	x.x	Clock bias (m)
F11	x.x	Clock bias rate (m/s)

Remarks

3. The values returned for the clock bias terms depend on whether the receiver applies corrections for frequency drift. If TCXO frequency drift corrections are not applied (i.e. the CFG has an 'F' instead of an 'f') then the clock bias term will be large and will drift consistent with the rate term, which means that it will occasionally be necessary to apply a step correction to the local clock in order to ensure that the pseudoranges do not become significantly different to the actual ranges. However, if corrections are applied then the receiver tries to compensate for drift and the bias will be smaller and not drift. The advantage of the latter option is that clock resets will not occur since the receiver local clock will not accumulate excessive error.

Example

$GPGGA,080002.999999976,3341.0000,S,15056.0000,E,1,04,2.7,86.9,M,0,M,,*53

12.4.35 ZDA Command and Report

Description

NMEA UTC time and date report

$GPZDA,*F1,F2,F3,F4,F5,F6*

The report is triggered at each 1 second boundary, although this implementation actually reports GPS time (and not UTC time like it should), as does the command.

Fields

Number	Name	Description
F0	$GPZDA	Sentence header
F1	hhmmss.ssss	UTC time with hours, minutes, seconds and fractional seconds
F2	dd	Day of month (1-31)
F3	mm	Month of year (1-12)
F4	yyyy	Year
F5	zh	Local zone hours (+/-13)
F6	zm	Local zone minutes (same sign as hours)

Remarks

1. This implementation actually outputs GPS time, so the leap second and fractional second correction is NOT included.
2. Similarly, the command does not apply any UTC/GPS leap second correction and as such, the input time is actually GPS time expressed in *hhmmss.sss* format.
3. The time-zone correction is hard coded to 00:00.
4. Output of the sentence is not strictly aligned to the GPS 1 second boundary and neither is the first character of each sentence
5. Sending a ZDA sentence to the receiver may result in the real-time-clock of the GPS receiver being updated.

Example

$GPZDA,071723.509,16,06,2011,+00,00*72

12.4.36 UALM Command and Report

Description

This is commands only used for debugging purpose. This command will force the receiver to use Galileo almanac that is in hardcoded into the receiver's memory.

$GPGPQ,UALM,A

12.4.37 UEUD Command and Report

Description

This is commands only used for debugging purpose. This command will force the receiver to use Galileo ephemeris from 25th august,2011 at 17:00:00pm. This Galileo ephemeris is hardcoded into the receiver's memory.

$GPGPQ,UEUD,U

13 Field Upgrade of the Namuru Receiver Firmware and Logic

This section is only applicable to Namuru V3.2 GPS receiver.

13.1 Bootloader Requirements

The bootloader is required to satisfy the following requirements.

1. Allow the firmware to be upgraded
2. Allow the A3 FPGA logic to be reprogrammed
3. Conform with the requirements of the Colony 2 bus and the Biarri mission
4. Coexist with the existing Aquarius firmware on board the Namuru V32 GPS receiver

13.2 Bootloader Operation

The Biarri bootloader is the first piece of user firmware that is executed following a hardware or software reset. Only internal SmartFusion firmware that is hard coded by Actel into the device will precede it.

The bootloader interacts with the user via the same RS422 serial interface that is employed by the Aquarius firmware. Following a reset, the bootloader waits for a pre-determined period of time for the user to send a command that will force the device to stay in bootloader mode. Failure to receive this command will result in the firmware switching to one of two Aquarius firmware images that is stored within the SmartFusion flash memory. Each Aquarius firmware image is allocated a maximum size of 0x30000 hex bytes, with one starting at 0x10000 and the other at 0x50000. A memory addressing aliasing feature of the SmartFusion system is used to select between the two images, which means that a compiled image can be programmed into either of the two slots and still operate

correctly. The bootloader itself starts at address 0x00000 and is allocated a maximum of 0x10000 hex bytes.

Serial port commands similar to the NMEA serial port messages used by the Aquarius firmware are used to interact with the bootloader. The difference lies in the header and footer used by the bootloader, the use of a four character message identifier, as well as the type of checksum employed, with the bootloader using a 16 bit CRC rather than an 8-bit exclusive-or sum checksum. This should be more robust and enable failed messages that pass the checksum test accidentally to be far less likely.

A description of each message follows.

13.2.1 BOOT Command

Description

This command is used to force the bootloader to remain in bootloader mode following a reset.

~$PBTLD,BOOT*BLFT<CR><LF>~
~$PBTLD,BOOT*hhhh<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,BOOT	Sentence header

Remarks

5. Sending this command causes the receiver to remain in bootloader mode.
6. The command must be sent within the timeout window following a hardware or a software reset.
7. The length of time of the timeout window can be set using the CNFG command, which if changed will be saved in the receivers non-volatile serial flash.
8. When the receiver enters bootloader mode following a reset, the receiver will print out a welcome string indicating that the bootloader is active. The print out operation can be disabled using the CNFG command if this operation is not desired

Example

~$PBTLD,BOOT*BLFT
~

13.2.2 CNFG Command and Report

Description

This command is used to configure the bootloader behaviour.

~$PBTLD,CNFG,*F1*,*F2*,*F3**BLFT<CR><LF>~
~$PBTLD,CNFG,*F1*,*F2*,*F3***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,CNFG	Sentence header
F1	X	Selected firmware image to run 1 = Run Aquarius firmware image 1 by default 2 = Run Aquarius firmware image 2 by default
F2	X	Timeout value (supposed to be seconds, but isn't really) Valid values in the range of 10 to 120
F3	X	Flags a/A = enable/disable message acknowledgement w/W = enable/disable welcome message at reset x/X = enable/disable error reports

Remarks

9. Sending this command allows the default behaviour of the bootloader to be changed. In particular, the default firmware image (1 or 2) can be selected thereby allowing one of two firmware images to be selected for operation.
10. Sending this command causes the previous data that is stored in one of the serial flash sectors to be deleted and the data to be re-written. Care should be taken to avoid sending the message too frequently as the serial flash is limited to something like 100,000 write cycles per sector.
11. In the rare event that power is lost during the process of re-writing this sector, the unit will replace the data with default settings.

Example

```
~$PBTLD,CNFG,1,40,awx*BLFT<CR><LF>~
```

13.2.3 COPY Command

Description

This command is used to copy firmware images from external SRAM to SmartFusion Flash memory.

~$PBTLD,COPY,*F1*,*F2*,*F3**BLFT<CR><LF>~
~$PBTLD,COPY,*F1*,*F2*,*F3***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,COPY	Sentence header
F1	X	Destination memory ENVM
F2	x.x	Destination address (ASCII hex)
F3	X	Source memory ERAM
F4	x.x	Source address (ASCII hex)
F5	x.x	Length of copy (ASCII hex)

13.2.4 CRCI Command

Description

This command is used to reset the CRC that is maintained when sending data using the PROG commands

~$PBTLD,CRCI*BLFT<CR><LF>~
~$PBTLD,CRCI*_hhhh_<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,CRCI	Sentence header

Remarks

12. The bootloader maintains a 32-bit CRC when receiving or transmitting large blocks of data. This command is used to reset this CRC before commencing transmission of new data to the receiver.

Example

```
~$PBTLD,CRCI*BLFT<CR><LF>~
```

13.2.5 CRCP Command

Description

This command is used to provide the CRC for a firmware image that is then stored in the bootloader serial flash storage sector. The value is used when verifying that the firmware images stored in the SmartFusion flash memories are correct.

Command

~$PBTLD,CRCV,*F1*,*BLFT<CR><LF>~
~$PBTLD,CRCV,*F1*,**hhhh*<CR><LF>~

Report

~$PBTLD,CRCV,*F1*,*F2***BLFT*<CR><LF>~
~$PBTLD,CRCV,*F1*,*F2***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,CRCP	Sentence header
F1	X	Firmware image Valid values are 1 and 2.
F2	X	Image base address (ASCII hex)
F3	X	Image length (ASCII hex)
F4	X	Image CRC (ASCII hex)

Remarks

13. This command tells the GPS receiver the CRC value for a particular firmware image.

Example

```
~$PBTLD,CRCV,1,10000,25000,1234FEDC*BLFT<CR><LF>~
```

13.2.6 EXEC Command

Description

This command is used to cause the bootloader to transfer control to a particular firmware image.

~$PBTLD,EXEC,*F1*,*BLFT<CR><LF>~
~$PBTLD,EXEC,*F1*,**hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,EXEC	Sentence header
F1	X	Firmware image Valid values are 1 and 2.

Remarks

14. This command tells the GPS receiver bootloader to start executing a particular image immediately on receipt of the command.

Examples

```
~$PBTLD,EXEC,1*BLFT<CR><LF>~
~$PBTLD,EXEC,2*BLFT<CR><LF>~
```

13.2.7 FPGA Command

Description

This command is used to initiate an FPGA programming operation

~$PBTLD,FPGA,*F1**BLFT<CR><LF>~
~$PBTLD,FPGA,*F1***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,FPGA	Sentence header
F1	x.x	32-bit CRC in ASCIIized hex

Remarks

15. This command is used to initiate an FPGA programming operation.
16. The 32-bit CRC included in the command has been included in order to enforce a check on the integrity of the FPGA image. The value included in this command is required to match the value calculated by a READ of the FPGA image performed immediately prior to executing the write command.
17. An FPGA programming operation is not permitted to take place unless the READ of the memory has been performed.
18. The 'gen3' utility generates an appropriate command and places the command into the 'FpgaMsg.out' file whenever an FPGA image is converted from the '.dat' format that is output by the Actel/Micosemi Libero toolchain. The 'gen3' utility also generates a READ command, placing that output into the 'ReadA3.out' file.

Examples

```
~$PBTLD,FPGA,2EC61514*2A7D<CR><LF>~
```

13.2.8 PROG Command

Description

This command is used to send a page worth of data to the receiver for programming.

~$PBTLD,PROG,*F1*,*F2*,*F3*,*F4*,*F5**BLFT<CR><LF>~
~$PBTLD,PROG,*F1*,*F2*,*F3*,*F4*,*F5***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number		Name	Description
F0		$PBTLD,PROG	Sentence header
F1		x	Total number of sentences
F2		x	Sentence number
F2=1	*F3*	xxxx	Destination memory: ERAM, ENMV, SFLH
	F4	x.x	Address of page in destination memory (ASCII hex)
	F5	x.x	Length of data (ASCII hex)
	F6	NULL field	NULL field
F2≠1	*F3*	x	ASCII = A, Binary = B
	F4	x.x	Address within page (ASCII hex)
	F5	x.x	Length of data block within page If *F3* is B, then the F5 field should be delimited by a '# 'character rather than a ',' character
	F6	x.x	ASCIIized hex or binary data. If binary data, the data is required to conform with the Colony 2 Bus bit stuffing convention.

Remarks

19. The PROG command represents the way in which a large file is broken down into a sequence of individual page writes, where a page is either 128 bytes or 256 bytes in length. Each page of data requires the transmission of several sentences.
20. The first sentence defines the memory to be programmed, the address within that memory and the length of data to transmitted.
21. The remaining sentences break the page of data into sufficiently small packets that conform with the Colony 2 Bus messaging requirements.
22. The data block may be sent using either ASCII or binary, although ASCII requires roughly double the transmission time.

Examples

```
~$PBTLD,PROG,7,1,SFLH,500000,100,*BE21
~~$PBTLD,PROG,7,2,A,0,30,44657369676E65722031302E312E322E3120202020202020459C6616000000CFA15
```

```
306010001000100000000000000000*ED58
~~$PBTLD,PROG,7,3,A,30,30,00000000000000100FFFFFF030A0C16CC06890006017B00000009000000028400
0000DA000000035E01000004000000*30E4

~~$PBTLD,PROG,7,4,A,60,30,046201000004000000056601000030641600069665160004010000087FDA56FCCF4
CF873924499224499224499224499224*6F23
~~$PBTLD,PROG,7,5,A,90,30,244992244992244992244992244992244992244992244992244992244992244992
244992244992244992244992244992244992*1741
~~$PBTLD,PROG,7,6,A,C0,30,244992244992244992244992244992244992244992244992244992244992244992
244992244992244992244992244992244992*EB7F
~~$PBTLD,PROG,7,7,A,F0,10,24499224499224499224499224499224*9345
~
```

13.2.9 READ Command

Description

This command is used to read the contents of a GPS receiver memory.

~$PBTLD,READ,*F1*,*F2*,*F3*,*F4**BLFT<CR><LF>~
~$PBTLD,READ,*F1*,*F2*,*F3*,*F4***hhhh*<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,READ	Sentence header
F1	XXXX	Source memory: ERAM = external SRAM ENMV = SmartFusion Flash memory SFLH = Serial Flash
F2	x.x	Start address (ASCII hex)
F3	x.x	Length of read (ASCII hex)
F4	X	Output format A = ASCII B = Binary C = calculate CRC for the block only

Remarks

23. This command is used to read the contents of a memory that can be programmed using the bootloader.
24. When F3 is A or B, the output of the process is a sequence of commands that should be suitable to allow a programming operation to be performed later.
25. If binary output has been selected, the Colony 2 bit-stuffing convention is used.
26. The main use of this command is to validate an FPGA image before the programming of the FPGA is allowed to take place. However, for that use-case, the F4 field can be set to 'C'.

Examples

```
~$PBTLD,READ,ENVM,10000,25040,C*A6C3<CR><LF>~
~$PBTLD,READ,SFLH,500000,16669C,C*0419<CR><LF>~
```

13.2.10 TEST Command

Description

This command is used to trigger the execution of self-test routines

~$PBTLD,TEST*BLFT<CR><LF>~
~$PBTLD,TEST*_hhhh_<CR><LF>~

Here *hhhh* refers to the 16-bit CRC, <CR> a carriage-return character, <LF> a line-feed character and 'BLFT' is the 'cheating CRC', which can be used if interacting with the bootloader using a terminal and no CRC calculating routine is available.

Fields

Number	Name	Description
F0	~$PBTLD,EXEC	Sentence header

Remarks

27. This command tells the GPS receiver bootloader to run self-test routines. The current bootloader only supports two types of self-test. One is a test of the internal and external SRAM devices. The other is a verification that the CRCs of the firmware images match the values stored in the non-volatile serial flash.

Examples

```
~$PBTLD,TEST*BLFT<CR><LF>~
```

13.2.11 Reprogramming the A3 FPGA

The following procedure is used to reprogram the A3 FPGA.

1. Use the 'gen3' utility to convert the FPGA '.dat' file into a sequence of sentences used to reprogram the FPGA. This is done using by running one of the following two commands:
   ```
   gen3  -i GpsL1TopV2-1-2.dat -o AsciiGpsL1TopV2-1-2.out -a -t A3
   gen3  -i GpsL1TopV2-1-2.dat -o AsciiGpsL1TopV2-1-2.out -b -t A3
   ```
   where in this case, 'GpsL1TopV2-1-2.dat' is the input '.dat' file, 'AscciGpsL1TopV2-1-2.out' is the output file containing the sequences of sentences required to program the Namuru V3.2 serial flash with the FPGA image, the '-a' or '-b' option determines whether the output is entirely ASCII or binary, while the '-t A3' argument indicates that an A3 image is being generated. The binary file is smaller and may be preferred if it is important to minimize the transmission time.
2. Executing the above command will also generate several other files that are useful for the reprogramming process. The first is 'WipeA3.out', the second is 'ReadA3.out' and the third is 'FpgaMsg.out'. These are placed into the same directory as the other output from the utility.
3. After resetting the Namuru V3.2 GPS receiver, send the command:
   `~$PBTLD,BOOT*BLFT<CR><LF>~`
   where `<CR>` is a carriage return and `<LF>` is a line feed. This needs to be sent during the window of time that the bootloader is executing and has the effect of forcing the bootloader to remain in bootloader mode. The bootloader will then acknowledge this command (unless acknowledgements have been disabled), after which the bootloader will remain active until terminated by the user. Future versions of the bootloader may use a hardware line to achieve the same effect, although this option is not currently implemented.
4. The user is then required to send the commands contained within the 'WipeA3.out' file. This command causes the bootloader to clear sectors within the 64 Mb ST M25P64 serial flash memory where the FPGA image will be stored. Clearing the serial flash sectors before sending the file allows the flash to be programmed as each page of data is received. This is necessary because the operations to clear the serial flash memory sectors are slow, taking typically 1 second and up to a maximum of 3 seconds to complete a single 64 kB sector erase. The bootloader will send an acknowledgement when the operation has completed.
   `~$PBTLD,WIPE,SFLH,500000,16669C*A63B<CR><LF>~`
   An example of the WIPE command to clear the FPGA serial flash memory sectors is given above, where the 500000 parameter refers to the start address of the FPGA image within the serial flash and 16669C refers to the length of the block to be cleared.
5. The next step is to send the commands required to program the serial flash with the data to be programmed into the A3 FPGA. This represents approximately 1.4 MB of information, although once converted into the sentences required to perform the programming the size of the data that is actually transmitted will be further increased. These commands are contained within '.out' file generated by the command in step 1.
6. On completion of the transmission of the '.out' file, it is necessary to read the saved data in order to verify that the data stored in the serial flash matches the data transmitted to the unit. The user should confirm that the 32-bit CRC values displayed at the end of step 5 and calculated by step 6 all match.
   `~$PBTLD,READ,SFLH,500000,16669C,C*0419<CR><LF>~`

An example of the READ command is given above, where the 500000 parameter refers to the start address of the FPGA image within the serial flash, 16669C refers to the length of the file and C indicates that the command is only required to return the 32-bit CRC for the output.

7. The final step is to send the command to actually perform the programming operation. This command is contained within the 'FpgaMsg.out' file that is generated by 'gen3'. The programming operation takes approximately 5 minutes. The user should refrain from transmitting data to the receiver during this process. During the programming operation, one of the LEDs on the board will light up indicating that a programming operation is in progress. The unit will send an acknowledgement once the operation has completed.

13.3 Reprogramming the GPS Firmware

The procedure for programming the GPS firmware images is as follows:

1. Use the 'gen3' utility to convert the Aquarius '.bin' file into a sequence of sentences used to reprogram the SmartFusion flash memories. This is done using one of the following commands:

```
gen3 -i Aquarius.bin -o Bin1SramAquarius.out -b -t FW -f 1 -s
gen3 -i Aquarius.bin -o Bin2SramAquarius.out -b -t FW -f 2 -s
gen3 -i Aquarius.bin -o Ascii1SramAquarius.out -a -t FW -f 1 -s
gen3 -i Aquarius.bin -o Ascii2SramAquarius.out -a -t FW -f 2 -s
```

   where in this case, 'Aquarius.bin' is the input firmware binary image, 'Bin1SramAquarius.out' or 'Bin2SramAquarius.out' or 'Ascii1SramAquarius.out' or 'Ascii2SramAquarius.out' is the output file, the '-t FW' argument indicates that a firmware image is being generated, the '-f x' determines whether the image being generated is for slot 1 or slot 2, while the -s indicates that the external SRAM should be used to store a temporary copy of the firmware image prior to copying it to the SmartFusion flash memory.
2. After resetting the Namuru V3.2 GPS receiver, send the command:

```
~$PBTLD,BOOT*BLFT<CR><LF>~
```

   where <CR> is a carriage return and <LF> is a line feed. This needs to be sent during the window of time that the bootloader is executing and has the effect of forcing the bootloader to remain in bootloader mode. The bootloader will then acknowledge this command (unless acknowledgements have been disabled), after which the bootloader will remain active until terminated by the user. Future versions of the bootloader may use a hardware line to achieve the same effect, although this option is not currently implemented.
3. The next step is to send the commands required to program the SmartFusion flash. This is less than 192 kB of data, although once converted into the sentences required to perform the programming the amount of data to be transmitted will be increased. The sentences to be transmitted are in the output file generated in step 1. Unlike the FPGA programming operation, the data is first placed into the external SRAM and when fully received by the receiver, will be copied from the SRAM into the SmartFusion flash memory. As is the case for the FPGA programming, the LED is lit during the programming operation. Several different bootloader commands are sent for this part of the process. The first command is the CRCI command, which has the effect of clearing the 32-bit CRC. This is followed by a sequence of PROG commands that are used to define the contents of a single page of data, where a SmartFusion flash page is 128 bytes and a serial flash page is 256 bytes. Each set of

PROG commands defines the address of the particular page to be programmed. On completion of transmission of the entire block of data, a CRCV command is used to transmit a CRC for the data block, which may be used by the user to confirm that the data was correctly received. A CRCP command is used to store the CRC of the firmware image in the serial flash thereby allowing the self-test feature to verify that the firmware image is correct. The final step is to send a COPY command that causes the firmware image to be copied from SRAM to the SmartFusion flash memory.

4. It is important to avoid sending serial port commands during the COPY operation as loss of serial port data during a programming operation has been observed.
5. This procedure allows one of two firmware images to be programmed, where the slot to be programmed is determined by the -f parameter. This feature allows two different versions of firmware to be loaded on the board, with the bootloader selecting one of those two for execution depending on how the user has configured the device using the CNFG command.
6. On completion of the programming operation, the TEST command can be sent to confirm that the firmware image was programmed correctly.
   ~$PBTLD,TEST*BLFT<CR><LF>~
   Currently, the only tests supported by the self-test feature are tests on the internal and external SRAM memories
7. Control can be passed from the bootloader to a particular firmware image by using the EXEC command, as shown below:
   ~$PBTLD,EXEC,1*BLFT<CR><LF>~
   ~$PBTLD,EXEC,2*BLFT<CR><LF>~
   where the first command transfers control to firmware image 1, while the second transfers control to firmware image 2. This is useful to verify that the firmware programming exercise has been successful.

14 Conclusion and Future Work

This document covered all the activities carried out during the development of GNSS receiver prototype for the Garada and Biarri space mission projects. The details provided in this document make it quite clear that the work package 5 was the largest because of several stages and kinds of development involved. The design, development and testing of this high-precision positioning and timing receiver involved several revisions of the hardware, digital logic and the firmware.

The challenges in hardware were to add space-capability e.g. in terms of recovery during high radiation, to design a thermally and mechanically balanced receiver (for Colony-II in the case of Biarri) and high frequency high bandwidth RF to digital interface design (in the case of L5/E5a front-end in GD1030 and GD1040 chips). One indication of the hardware complexity is that it took General Dynamics six revisions to get the Namuru V3.3 board noise down to an acceptable limit for space applications.

The challenges in the signal processing digital logic and firmware development were not simple either. New implementation techniques were devised to cater to the very high platform dynamics of the proposed LEO spacecraft where the GNSS receiver will operate. Fast signal acquisition techniques were designed and implemented in digital logic and the firmware for the Galileo E1 signals and a highly improved high dynamics signal tracking and measurement generation algorithms were implemented for the carrier-phase measurement and processing. Kalman filter and related algorithms were tuned appropriately to produce reliable and improved position estimates.

It is worth to mention that the in-field firmware and FPGA logic upgrade feature in the Namuru V3.2 will go a long way in defining a new benchmark for space-capable GNSS receivers and in the final setup, this upgrade will be On-The-Air over the satellite link.

It should be noted that as part of this project UNSW team was involved in modifying the widely used open source GNSS post processing software, the RTKLib to accommodate new and modernized GNSS signals like Galileo, GLONASS, GPS L5, QZSS and named the package as RTKLib Pro. In addition, the RTKLib Pro implemented innovative techniques to bring down the data handling and post processing time.

Some of the multi-GNSS receiver features proposed as per the initial plan of this project, the GPS L5 and Galileo E5a integration to the Namuru receiver are still under testing and debugging phase at the time of writing this document. As mentioned in the report, the reasons were partially the effort made for the Biarri project but mainly from delays due to the many Namuru V3.3 hardware revisions.

The good side though is the successful integration of the Biarri platform into the final system, the development of four new Namuru platforms V3.1, V3.2, V3.3 and V3.4, the in-house development of three RF front-end ASICs BL2627, GD1030, GD1040 by General Dynamics, NZ. The development of tailored RF chips is a complex process and the unique ‘tuneable’ chips developed by the UNSW team will continue to have access to these chips for further developments.

The triple-RF-channel platforms, Namuru V3.3 and V3.4 after successful testing, will be very strong research platforms because they offer the GNSS receiver designer to process wideband GPS L1/E1

signal plus any two of the other GNSS signals including L1/E1. Some of the envisaged applications where these platforms (with minimal interface modifications if required) will be useful are GNSS base station receivers, applications that require triple frequency carrier phase solutions, attitude determination etc.

The UNSW and General Dynamics Namuru GNSS receiver roadmap has planned interesting platforms such as a small form factor dual-frequency GD1040 based receiver for GNSS-Reflectometry applications. It should be noted that several upcoming GNSS related research projects by UNSW (some of which are in the proposal stage) will benefit from the Namuru 3.x platforms.

As mentioned earlier in the report, the UNSW Namuru GPS receiver platforms Namuru V2.x series are widely used across the world for different kinds of research activities (listed at the end of this section). The flexibility of the Namuru platforms due to the usage of FPGA devices are being successfully utilised by several organisations and one example is DLR, Germany for their space-related and sounding rocket projects. It is envisaged that the clients of Namuru V2.x will upgrade to the triple-frequency Namuru V3.3 and V3.4 receivers and the Namuru V3.x series will find new internal / Australian and International customers such as stratospheric scientific ballooning projects of UNSW ADFA, NavSAS GNSS receiver research groups in Torino, Italy and Hanoi, Vietnam and several other research initiatives by ESA.

List of major customers for V2.x:

1. University of Westminster (Research)
2. UNSW (Research)
3. DLR (Space)
4. Wipro (Product development)
5. Korea Institute of Standards & Science (Research)
6. Tokyo Institute of Marine Science and Technology (Research)
7. Hong Kong Polytech (Research)
8. Ansaldo STS (Railway control systems)
9. University of Barcelona (Research)
10. ITT communications Systems (Product development)
11. Bentch Computer Corp.. Taipei
12. IMT/EPFL Neuchatel (Research)
13. Universitat Politecnica de Catalunya (Research)
14. Richcore Technologies USA
15. GFZ, POTSDAM (Research)
16. Wuhan University, China (Research)
17. Tyco Flow Control (HK) Ltd, Hong Kong (Research)
18. Xi'an Auttech Digital Technology Ltd, China (Research)
19. DELFT University of Technology, Netherlands (Research)
20. NTK International Corp., Japan (Product development)
21. Honeywell Tech Solutions Lab (Product development)
22. TNX Inc, CA, USA (Product development)
23. Vanguard Exim Pte Ltd, Singapore (Product development)
24. BDStar, China (Product development)

25. Shanghai Avionics Corporation (Product development)
26. Dubai Electronics Industry Ltd (Product development)
27. Ravenlake Ltd (Product development)
28. INTERNEX.CO.LTD (Product development)
29. Chung-Ang University (Research)

Additional list of potential customers for V3.x :

1. NASA (Space applications)
2. DLR (Space applications)
3. CubeSat50 customers (Space applications)
4. Ansaldo (Railway safety control systems)
5. Navicom Dynamics (Marine navigation)
6. DSTO Australia (Space and Research)
7. DTA New Zealand defence (Space and Research)

15 References

Parkinson, K.J. and Rizos, C., Prof. (2004) An Open GNSS Receiver Platform Architecture. in The 2004 International Symposium on GNSS/GPS (Sydney),

Frear, D.R., Burchett, S. and Morgan, H.S. (1994) The Mechanics of solder alloy interconnects. Springer.

www.garada.unsw.edu.au, Garada, 2010

Grillenberger, A. and Markgraf, M. (2011) Flight test results of a novel integrated GPS receiver for sounding rockets. in 20th ESA Symposium on European Rocket & Balloon (Hyeres, France),

Grillenberger, A., Rivas, R., Markgraf, M., Mumford, P.J. and Parkinson, K.J. (2008) THE NAMURU RECEIVER AS DEVELOPMENT PLATFORM

FOR SPACEBORNE GNSS APPLICATIONS. in Navitec 2008, Noorwijk, The Netherlands

Layton, D., Czajkowski, D., Marchall, J., Anthony, H. and Boss, R.(2010) Single Event Latchup Protection Of integrated Circuits, Maxwell Technologies, San Diego, CA

www.sbir.gov/sbirsearch/detail/218094, Nasa, 2002

Parkinson, K.J., Dempster, A.G., Prof, Mumford, P.J. and Rizos, C., Prof. (2006) Improving signal quality in FPGA based GPS

receiver designs. in IGNSS Symposium 2006 (Surfers Paradise, Australia), International Global Navigation Satellite Systems Society

www.sstl.co.uk, Surrey Satellite Technology Ltd, 2011

Zarlink Semiconductor Inc.(2007) GP2015

IS-GPS-705A (2010), “Navstar GPS Space Segment / User Segment L5 Interfaces”, Revision A, GPS Wing, USA, 8th June 2010.

OS SIS ICD 1.1 (2010), “Galileo Open Service Signal In Space Interface Control Document”, European Union, Issue 1, Revision 1, Sep 2010.

GIOVE-A+B OS SIS ICD (2008), “GIOVE-A+B (#102) Navigation Signal In Space Interface Control Document”, Galileo Project Office, 8th Aug, 51pp.

Namuru Datasheet (2007) Namuru FPGA GPS tracking module Issue 1.3 January 2007, http://www.dynamics.co.nz/media/Namuru_GPS_datasheet.pdf

GPS ICD (2010), “Navstar GPS Space Segment / Navigation User Interfaces”, IS-GPS-200 Revision E, GPS Wing, 8th June, 185pp.

Takasu, T., and Yasuda, A. (2009),"Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB", International Symposium on GPS/GNSS,Jeju, Korea,November 4-6.