International Global Navigation Satellite Systems Society
IGNSS Symposium 2007

The University of New South Wales, Sydney, Australia
4 – 6 December, 2007

"Namuru-GPL", open source software for the Namuru FPGA-based GNSS receiver

Peter J Mumford
School of Surveying and Spatial Information Systems, UNSW, Australia
Ph +612 93854189, Fax +612 93137493, p.mumford@unsw.edu.au

Yong Heo
School of Surveying and Spatial Information Systems, UNSW, Australia
Ph +612 93854174, Fax +612 93137493, yong.heo@unsw.edu.au

ABSTRACT

The "Namuru" field programmable gate array (FPGA) based receiver platform has been in development at the University of New South Wales since 2004. In late 2006 a project to port open-source GPS software to the platform began. This project is the focus of the paper.

The concept behind Namuru is to develop a fully open-source GNSS receiver platform to support research, development and teaching across a wide range of topics. While the baseband processor and circuit board are already released as open-source, the current position solution software (a port of the GPS Architect) is available to licence holders only. This presents a problem as Zarlink, the current 'owners' of the GPS Architect, do not support it any more, and are not interested in issuing licences. A natural progression was to take an appropriate version of the open-source GPS software originally developed by Cliff Kelly for Zarlink based receivers and port it to Namuru. The version chosen was Andrew Greenburg's port to the Signav MG5001 receiver.

In this paper, an overview of open-source GNSS software is provided, followed by a brief look at the Namuru hardware platform. The details of the porting process are then presented, along with outcomes, results of testing, problems found along the way and future activities

KEYWORDS: Namuru, GNSS, GPS, eCos, open-source.by commas

1. INTRODUCTION

This paper provides the details and results of a project to port open source GPS software to the Namuru hardware platform. An overview of the GPL-GPS open source software along with a brief history is given followed by an introduction to the Namuru hardware. The porting process is then discussed before delving into the details. Finally results are outlined, problems identified and conclusions drawn.

2. GPL-GPS, open source software for the GNSS world

2.1 Brief history

Dr. Clifford Kelly began working on open source GPS software around 1995. He called the project OpenSource GPS (OSGPS). He hacked into a commercial GPS receiver board that used the Mitel (now Zarlink) GP2021 correlator chip, bypassing the receiver's microprocessor and connecting the GP2021 directly to a 486 PC using an ISA prototyping board. Figure 1 shows an example of a hand built card. He then developed software on the host PC that eventually became OSGPS. In 2002, Kelly’s work was published; this was the first open source GPS receiver software written, see Kelly (2002). Development of this software continues on SourceForge (2007), and a new PCI based receiver platform is available from GPS Creations (see GPS Creations (2007)). Figure 2 provides an image of a GPS Creations PCI board. There is also an active Linux port of the OSGPS.

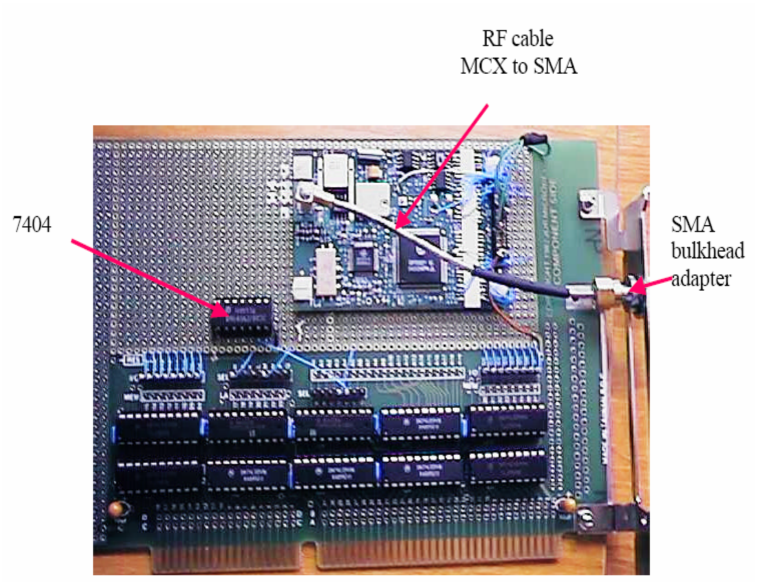


Figure 1. ‘Hacked’ GPS receiver card.

However, a PC based GPS receiver has limitations; a stand alone receiver is far more practical for real world applications. In 2004 Takuji Ebinuma took Kelley’s OSGPS code and ported it to the Signav MG5001 receiver. Tak started his project with OSGPS v1.15 in 2004. He broke down the OSGP’s main task into thread based tasks and used a simple real time operating system (RTOS) to manage these tasks. While Takuji’s code does works, it is unstable and has been abandoned. This work became the world’s first open source stand-alone GPS receiver.

Figure 2. GPS Creations PCI GPS hardware.

In 2004, Andrew Greenberg, at Portland State University, began a masters project to create open source GPS software for a commercial receiver. The outcome was the GPL-GPS software for the Signav MG5001 receiver. He took Tak's code and OSGPS v.1.17 as reference and used the eCos real time operating system to manage the software tasks. For information on eCos see Massa (2003). He restructured and replaced almost all of Tak's code to write the GPL-GPS code. The GPL-GPS's first position fix was reported on May 2nd, 2005. GPL-GPS runs on the Signav MG5001 receiver that has a Zarlink GP4020 chip, a 12-channel L1 C/A GPS receiver baseband processor with an integrated 32 bit ARM7TDMI microprocessor. Figure 3 reveals a Signav MG5001 receiver on a support card. Development of GPL-GPS appears to have pretty much stopped in 2006. Details about Andrews work can be found in his masters thesis, see Greenberg (2005).

Figure 3. Signav MG5001 receiver on support board.

In summary, there are in essence two flavours of open source GPS software, one is the active development of Cliff Kelly's code for a PC with PCI based hardware, the other is a dormant fork for the Signav MG5001 hardware. The later code was chosen for porting to the Namuru platform as it uses the eCos RTOS that is supported by the NIOS II soft-core processor that can be built on the Namuru's FPGA chip.

2.2 GPL-GPS overview

The code consists of a number of task threads, an interrupt routine and the eCos real time operating system that ties it all together. eCos stands for embedded, configurable operating system. eCos has been developed by Red Hat as a flexible and open source RTOS that supports a wide variety of processors. eCos has a configuration tool that enables a custom library to be built with a wide range of functionality for many different processors, including the Nios II soft-core processor developed by Altera. eCos supports all the common features of a complete RTOS such as mutex's and mail boxes.

Figure 4 provides an overview of the structure of the GPL-GPS code. The correlator block hardware (in the GP4020) is controlled by the software for channel allocation and signal tracking functions, and provides accumulation data for the tracking loops and measurement data for forming the pseudoranges used for the position, velocity, time (PVT) solution. The serial port hardware (also in the GP4020) is controlled by software to provide communication with the user.

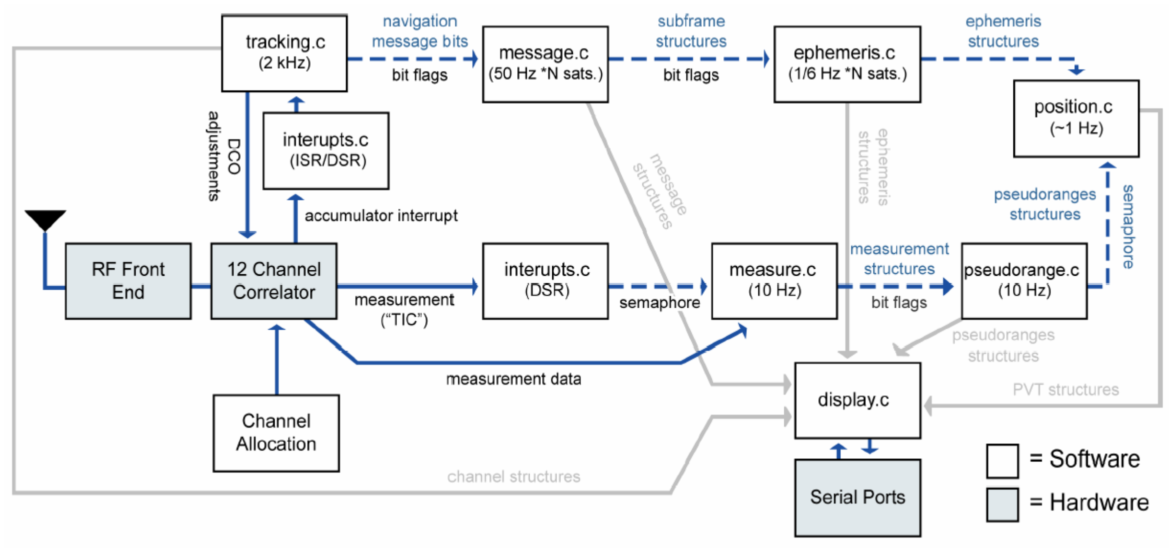


Figure 4. Structure of the GPL-GPS software.

There are seven tasks and an interrupt routine. Each task is contained in one file along with helper routines. The eCos framework provides task management code as well as common library functions and the hardware abstraction layer (HAL) for dealing with hardware.

2.3 Namuru hardware

The Namuru receiver hardware is based around an Altera field programable gate array (FPGA) chip; the Cyclone 2C50. In addition to the FPGA there is an L1 RF front end (using the Zarlink GP2015 chip), memory, various I/O including two serial ports, a real-time clock, JTAG conection a configuration controller on the board. Figure 5 shows version one of the Namuru board as used in this project. For more information on the Namuru project see Mumford (2006).

The correlator block (often called the baseband processor) and the Nios II soft-core processor, along with UART hardware and debugging logic reside in the FPGA chip. These logic blocks are designed with hardware design languages (mostly Verilog) using Altera's 'Quartus' development tool. The 'SOPC' builder in Quartus is used to build a custom Nios II soft-core processor. Software is developed using Altera's 'NiosII' IDE in the 'C' programming language. The eCos configuration tool is used to build a custom library based on the Nios core from the SOPC builder.

The Namuru board has successfully supported many designs including a port of the Mitel GPS Architect, a full featured GPS software application that provides PVT solutions with good quality and reliability. However, the GPS Architect is licensed software that is no longer supported by Zarlink, and therefore has limited use in source distribution. For this reason an open source solution was needed to support research – hence this project to port the GPL-GPS code to the Namuru platform.

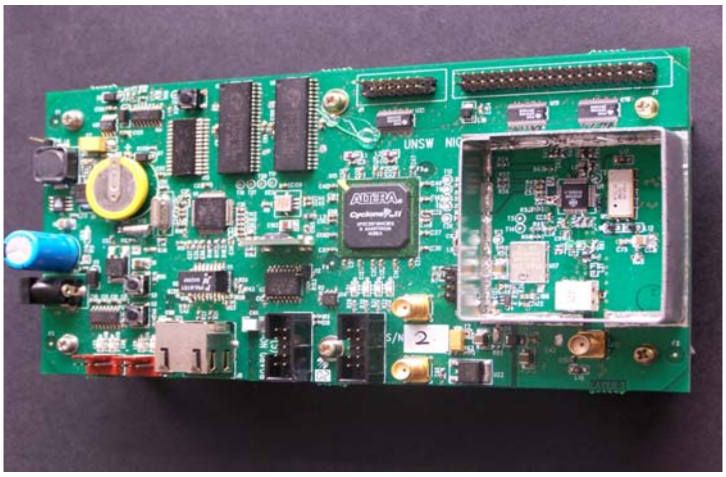


Figure 5. The Namuru V1 FPGA board.

3. The porting process

Porting the GPL-GPS software occurred in stages. There where considerations due to the different hardware and processor and to the different development tools. The main tasks in the porting process are described in this section.

To enable the porting effort, five primary areas were identified that required attention;
1) development tools
2) eCos library
3) correlator block hardware
4) UART hardware
5) debugging strategies
6) user interface

3.1 Development tools

The GPL-GPS code for the Signav MG5001 receiver was compiled for the ARM7 processor. This code base had to be brought into the Nios II integrated development environment, and any ARM specific code (assembler) discarded. A make file had to be developed to enable the compilation of ‘C’ files and to link them into the eCos library.

3.2 The eCos library

The eCos configuration tool is used to design and build an eCos library based around the target hardware and the users needs. Altera provide a wrapper for the configuration tool that sets up a template for the tool based on the NiosII soft-processor design from the SOPC builder. The configuration tool is then used to add features, setup the HAL, resolve conflicts and finally, build the library. The FPGA system design, including the correlator block, NiosII soft-core processor, Uart and other supporting logic for the Namuru board is a reference design developed for the GPS Architect software in 2005.

3.3 Correlator block hardware

The Namuru reference correlator block design was written in Verilog and although modelled on the Zarlink GP2021 it has significant differences that must be addressed in the porting process. There are differences in the address map, data width and the use of three correlator 'fingers' instead of two.

The register setup in the reference design is much less complicated and more logically set out than in the GP2021/4020. The document describing the design, with the address map is available on the web, see Mumford (2006). The file that defined the correlator block register locations had to be completely replaced. The GPL-GPS code had to be searched for all occurrences of correlator block access and lines of code changed to reflect the new hardware. This task was fairly mechanical, apart from code that contained the tracking loops and channel allocation. The GP2021/4020 has two correlator fingers per channel for 'tracking' and 'prompt', separated by half a chip. The 'tracking' finger can be controlled to work in different ways such as early-late. In the Namuru reference design, there are three fingers; early, prompt and late, each separated by half a chip. The tracking loop code had to be revised to use the three correlator fingers instead of the two in the GP2021/4020.

3.4 The UART hardware

The Namuru reference design implements UART's differently from the GP4020. Previous experience has shown that a micro processor running GPS software can spend a lot of cycles servicing interrupt routines for serial data output. The UART's in the Namuru reference design make use of first in, first out (FIFO) buffering to take some load off the processor for communication. The GPL-GPS code associated with UART activity was rewritten to suit the Namuru UART hardware. A double buffering scheme was implemented, with a circular buffer in software, and as mentioned, a FIFO buffer in hardware. The benefits of flexible hardware design in FPGA technology was really helpful at an early stage of the porting process, as UART hardware could be designed to circumvent problems in the code relating to UART activity clogging up the processor.

3.5 Debugging the beast

Debugging real time systems is always challenging, but additional problems where encountered on this project due to the relatively small SRAM on the Namuru board. The problem was that the debug version of the code (including eCos) was too big to fit in the available memory, therefore other debug strategies had to be used. The first strategy was to use the on-board LED's and I/O pins to provide status information. This was very useful to ensure interrupts where functioning properly (and with the correct timing) and that all tasks where getting a chance to run. The second strategy was to use two serial ports, one for the normal communication implemented in the GPL-GPS code, the other for debug activities. As mentioned in the previous section, the UART hardware was modified to allow large data streams to be sent without placing too much added burden on the processor. These strategies proved to be quite successful, and probably better than using the debugger as the thread of execution does not have to stop to view the data of interest.

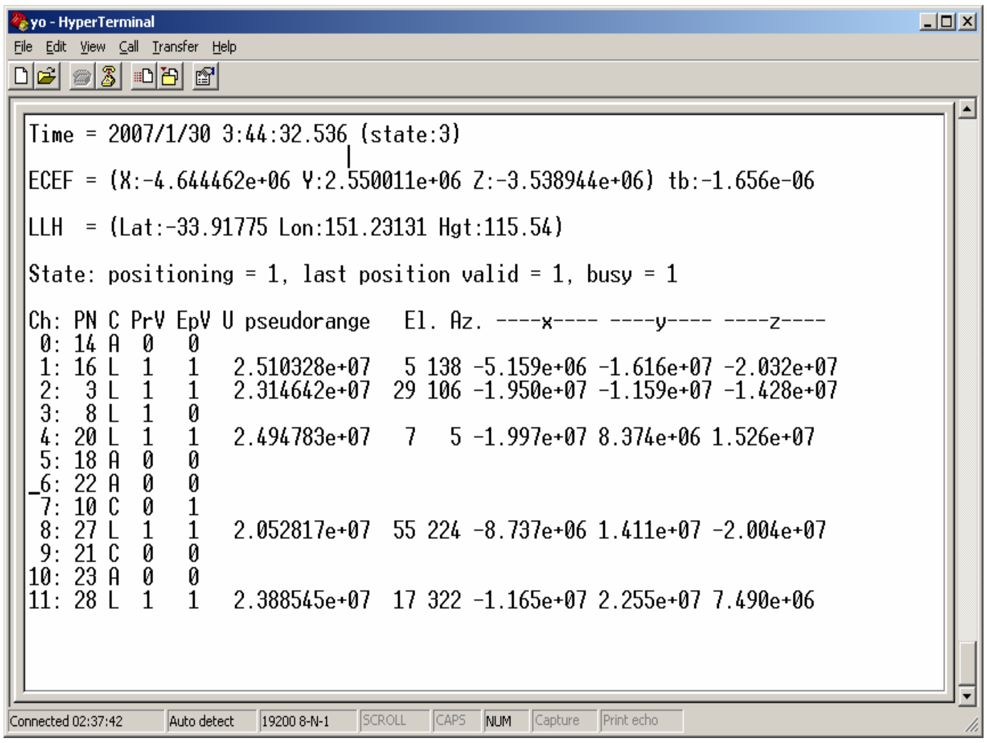
3.6 User interface

The RS-232 serial connection is used for communication between the display console computer and the Namuru board. The Hyper-terminal application is used to provide a visual user interface for monitoring and controlling receiver activity. The display is refreshed every second. Figure 6 reveals a screen shot of the interface.

GPL-GPS software supports several display modes that can be controlled from Hyper-terminal. Tracking mode is the default display, but other displays can be selected using letters on the keyboard. For example typing the 'e' character will change the display from tracking to ephemeris mode and the Log data display can be activated with the 'l' character.

4. RESULTS FROM TESTING

Before beginning the porting process, the GPL-GPS code was installed on a Signav MG5001 receiver. During testing it was found that satellite acquisition was very slow, and rarely more than five satellites would be tracked. Satellites where 'dropped' for no apparent reason. It was clear that the code had many bugs and required significant enhancements and improvements to really be effective GPS software. Due to these tests, the expectation was that the performance would be no better on the Namuru platform.



```
Time = 2007/1/30 3:44:32.536 (state:3)

ECEF = (X:-4.644462e+06 Y:2.550011e+06 Z:-3.538944e+06) tb:-1.656e-06

LLH  = (Lat:-33.91775 Lon:151.23131 Hgt:115.54)

State: positioning = 1, last position valid = 1, busy = 1

Ch: PN C PrV EpV U pseudorange   El. Az. ----x---- ----y---- ----z----
 0: 14 A  0   0
 1: 16 L  1   1   2.510328e+07   5 138 -5.159e+06 -1.616e+07 -2.032e+07
 2:  3 L  1   1   2.314642e+07  29 106 -1.950e+07 -1.159e+07 -1.428e+07
 3:  8 L  1   0
 4: 20 L  1   1   2.494783e+07   7   5 -1.997e+07 8.374e+06 1.526e+07
 5: 18 A  0   0
_6: 22 A  0   0
 7: 10 C  0   1
 8: 27 L  1   1   2.052817e+07  55 224 -8.737e+06 1.411e+07 -2.004e+07
 9: 21 C  0   0
10: 23 A  0   0
11: 28 L  1   1   2.388545e+07  17 322 -1.165e+07 2.255e+07 7.490e+06
```

Figure 6. Screen shot of the Namuru-GPL software showing the correct position and time.

Figure 6 shows a screen shot of the output display from the Namuru-GPL software. The position solution is correct to the tens of meter level and the time is right. Five satellites are tracked, but as in the GPL-GPS software, acquisition is slow, and satellites are dropped from time to time for no apparent reason.

A scatter plot of horizontal errors is shown in figure 7, and a vertical error time series plot is shown in figure 8. It can be seen that the position solution fails on several occasions (flat lines in figure 8) and the position errors become very large on occasion. It is clear that improvements to the code are required, particularly in acquisition and tracking as well as in the position solution.

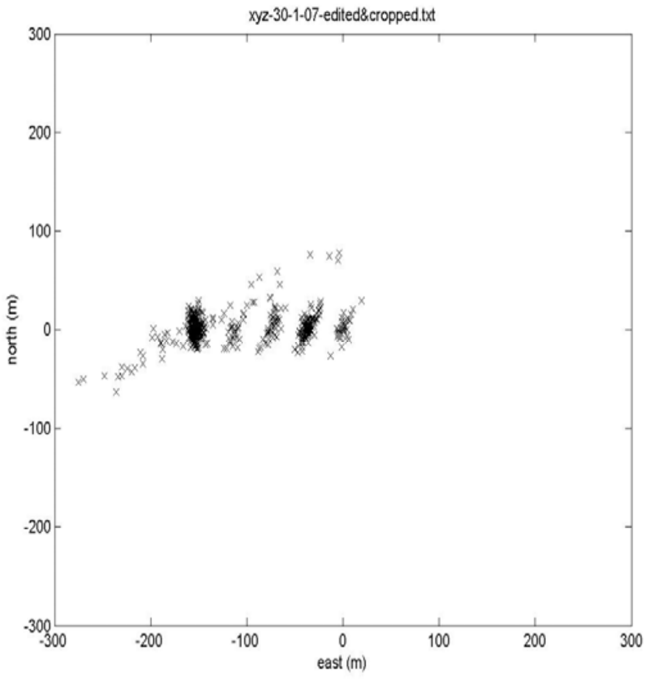


Figure 7. Horizontal error plot

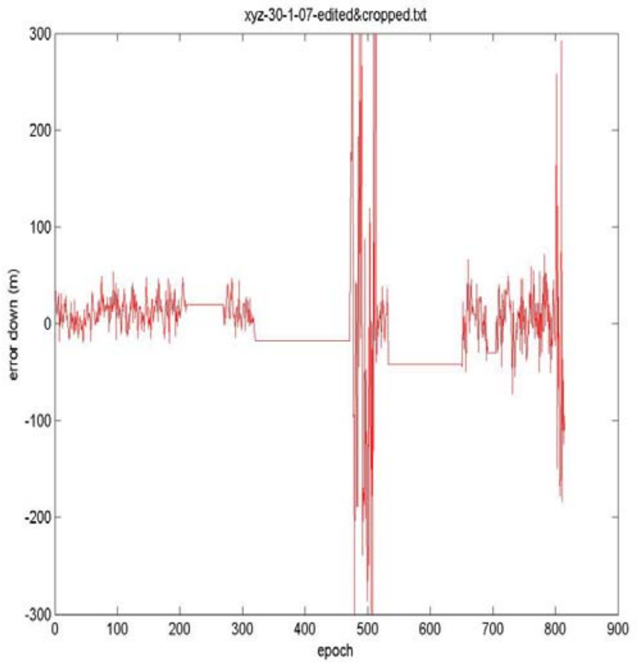


Figure 8. Vertical error plot

5. CONCLUSIONS

The primary objective of the project was to port the GPL-GPS software from the Signav MG5001 hardware to the Namuru platform. This has been successful and the software is stable and clean. The user interface works well, with a wide variety of information and control provided to the user. The acquisition and tracking functions are in need of refinement or redesign to improve the time to first fix and the number of satellites tracked. Different tracking loop strategies and filters could be implemented to improve signal tracking and provide carrier phase measurements. The position solution part of the code requires work to provide a more accurate position. In conclusion, the groundwork for open source software for the Namuru platform has been completed, but there is a lot more work required before it can be claimed to be truly useful.

REFERENCES

GPS Creations (2007) at http://www.gpscreations.com/

Andrew Greenberg, (2005) Open Source Software For Commercial Off-The-Shelf Gps Receivers Master degree Thesis Paper.

Kelley,C.,Cheng,J., & Barnes,J.,2002 Open source software for learning about GPS. 15th Int. Tech. Meeting of the Satellite Division of the U.S. Inst. of Navigation, Portland, Oregon, 24-27 September.

Massa A.J., (2003) Embedded Software Development with eCos, Prentice Hall, New Jersey.

Mumford P.J., (2007) Namuru FPGA GPS Tracking module data sheet, http://www.gmat.unsw.edu.au/namuru/documents/Namuru_GPS_datasheet.pdf

Mumford, P.J., Parkinson, K., & Dempster, A.G., 2006. The Namuru Open GNSS Research Receiver. 19th Int. Tech. Meeting of the Satellite Division of the U.S. Inst. of Navigation, Fort Worth, Texas, 26-29 September, 2847-2855.

Sourceforge, (2007) OSGPS on the Sourceforge code repository, http://sourceforge.net/projects/osgps/