A Software-Defined GPS and Galileo Receiver: Single-Frequency Approach

Kai Borre, *Aalborg University*
Dennis Akos, *University of Colorado*

BIOGRAPHIES

Kai Borre is a Professor of Geodesy at the Aalborg University since 1976. His recent software developments include a large collection of MATLAB files for postprocessing of GPS observations. In 1997 he coauthored the book *Linear Algebra, Geodesy, and GPS* with Gilbert Strang, Professor of Mathematics at the Massachusetts Institute of Technology. His current interests focus on filters for RTK applications. The present development is one in a series of future applications.

Dennis M. Akos completed the Ph.D. degree in Electrical Engineering at Ohio University within the Avionics Engineering Center. He has since served as a faculty member with Luleå Technical University, Sweden, and then as a researcher with the GPS Laboratory at Stanford University. Currently he is a faculty member with the Aerospace Engineering Science Department at University of Colorado at Boulder.

ABSTRACT

We discuss GPS receiver architectures based on software defined radio techniques. The reason for doing this is to obtain a reconfigurable receiver with a wide range of applications. There is a need for a unified platform that will allow receiver development and testing for various applications; this speeds the design process and reduces the costs.

With the current functionality of the GPS constellation and the promise of the complete Galileo constellation, efforts have been focused on the 1575.42 MHz L1 signals for the software receiver implementation. These single frequency navigation signals, particularly when coupled with space based augmentation such as WAAS or EGNOS are likely to fulfill the navigation needs of most users.

In order to develop and test the software algorithms, a complete L1-band antenna and RF front-end has been designed capable of processing the wider bandwidth necessary for the Galileo L1 BOC(1,1) signal. This front end provides digital samples to the host computer for the software receiver implementation.

In addition, a GNSS signal generator was designed and implemented in Simulink to be used for algorithm development and testing to offer the user the option of using the front-end to collect and process actual GPS data or generate simulated GPS signals. The GPS software receiver was implemented in Matlab and is capable of performing GPS satellite acquisition and tracking on both real GPS data and simulated GPS data with extreme properties.

A complete GNSS software receiver was implemented and the receiver is able to perform acquisition, code and carrier tracking, navigation bit extraction, navigation data decoding, pseudorange calculations, and position calculations.

We have created a pedagogical tool for the GPS newcomer: A front-end which obtains actual GPS data via the USB port, a complete Matlab implementation that provides a real-time GPS and Galileo (as defined by the current ICD) software receiver, and finally a textbook, Borre et al. (2006), that describes the whole thing.

INTRODUCTION

Software-defined radios (SDR) have been around for more than a decade. The first Global Positioning System (GPS) related application was described by Akos (1997). Since then several research groups have presented their contributions.

SDR is a rapidly evolving technology that is getting enormous recognition and is generating widespread interest in the receiver industry. SDR technology aims at an open architecture, flexible receiver. This helps in building reconfigurable SDRs where dynamic selection of parameters for individual modules is possible.

The receiver employs a wideband Analog to Digital (A/D) converter that captures all of the channels of the software

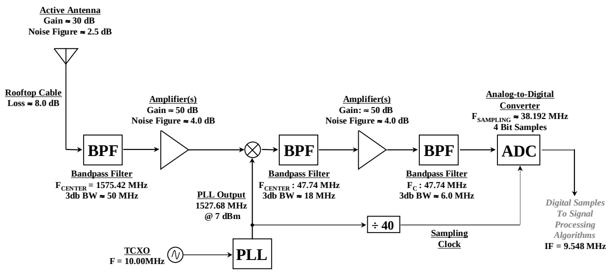

Figure 1: GNSS L1 front end

radio node. The receiver then extracts, downconverts, and demodulates the channel waveform using software on a general purpose processor. The idea is to get a wide band A/D converter as close to the antenna as is convenient, get the samples into something we can programme, and then grind on them in software. A SDR is an ideal platform for development, testing of algorithms, and possible integration of other devices. We have chosen MATLAB® (version 7) as coding language. Matlab is the de facto programming environment at technical universities, it is a flexible language, and easy to learn. Additionally it has good facilities for presentation of graphical results.

The concepts of the present project goes back more than ten years. At that time the technology was not mature for a realization; however, today we have been able to develop all pieces and enable them to work together:

- A front-end which eventually converts analog signals to digital signals that are imported into Matlab via the USB (version 2) port and working on a PC
- A forthcoming book, Borre et al. (2006), which describes a SDR
- A collection of Matlab code and data samples that comes with the book.

With all three parts you have an established GNSS software receiver capable of all the processing through the position solution.

A SDR not only solves the same tasks as a traditional hardware receiver, it also opens up for introducing new concepts. An illustration of the uniqueness of a GPS software receiver is for example that the powerful concept of a vector delay lock loop, which was first suggested by James Spilker, can be implemented. The recent paper Pany & Kaniuth & Eissfeller (2005) describes how integration between signal processing and navigation processor in a GNSS receiver may be realized if the navigation processor is used to control the code and carrier NCO instead of the tracking loops. In this mode, tracking loops become obsolete and the navigation solution is computed from the output from code and frequency discriminators. Once a good position, velocity, and time (PVT) solution is available (i.e. once at least 4 satellites are tracked), all other tracking loops remain in lock even under weak signal conditions. Short signal outages may be bridged if the navigation processor can predict the user movement. In this case, the receiver is in open loop mode—already known from GPS radio occultation experiments.

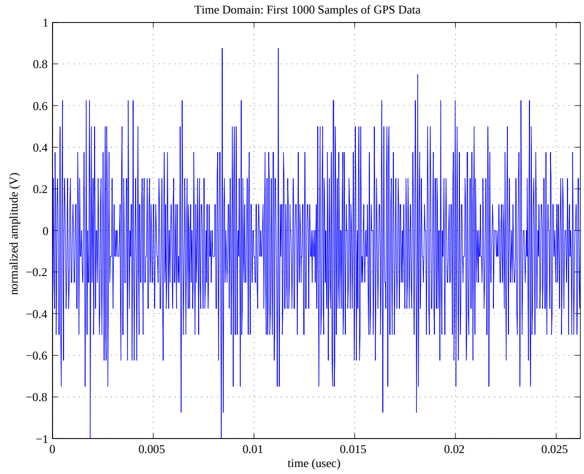

Figure 2: Time domain representation of the collected digital data samples

It is likely that we are facing an unbelievable and new technology for receiver development.

GNSS ANTENNAS AND FRONT ENDS

It is worthwhile to highlight the *resulting data set* that has been collected from the front end design depicted in Figure 1 and has been included on the media with the book.

The important parameters for the signal processing are:

- Sampling Frequency: 38.192 MHz
- Intermediate Frequency: 9.548 MHz, and
- Four bit samples represented as eight bit signed char values.

The above parameters provide all the necessary information for the operation of the signal processing algorithms.

What can be done is to show the resulting digital samples in typical representations. Thus in Figures 2, 3, and 4 a time domain, histogram, and frequency domain depiction of the collected data are illustrated, respectively.

In the time domain depiction, no discernable structure is visible despite the 9.548 MHz IF for the collected GPS data.

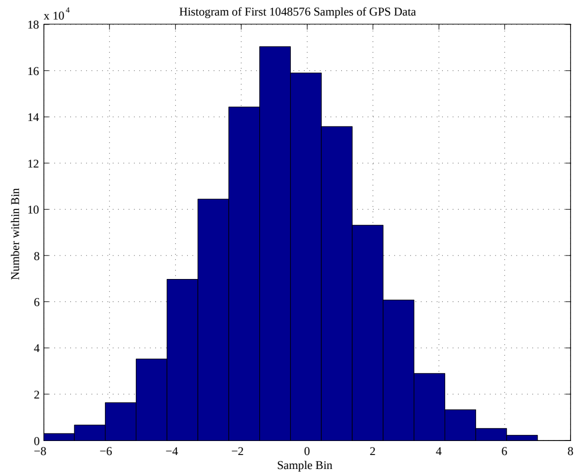

Figure 3: Histogram of the collected digital data samples

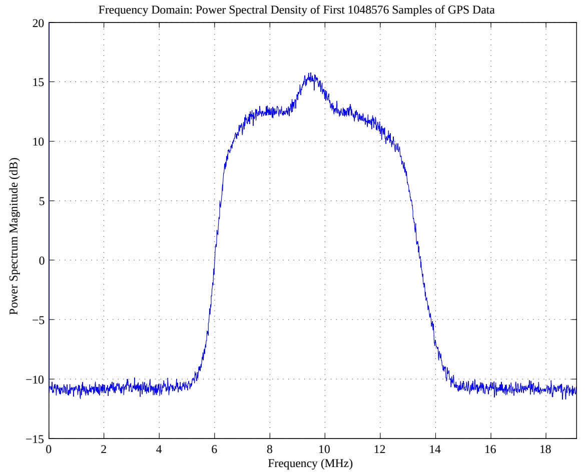

Figure 4: Frequency domain representation of collected data

In the histogram, it is obvious that all four bits of the ADC are being triggered based on the 16 levels present within the histogram. Also the histogram bears a strong resemblance to the probability density function for a Gaussian random variable, which would be expected for white thermal noise.

FLOW DIAGRAM FOR THE GNSS SOFTWARE RECEIVER

The present version of the software receiver is planned for postprocessing mode. The data are available either as files created by the font-end or as files that come with the book. Given a data file, 1 ms of data is read for acquisition. This results in a set of acquired PRNs, their (raw) tracking frequencies, and code phases. The flow diagram is shown in Figure 5.

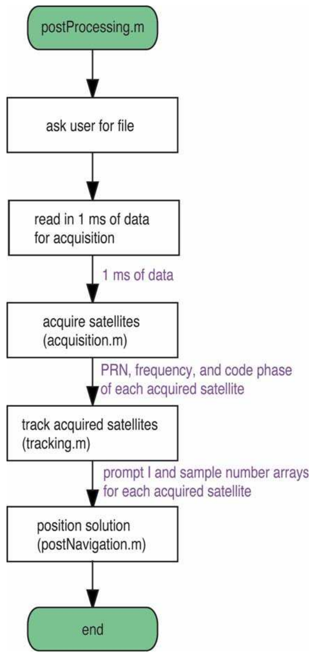

Figure 5: GNSS software receiver flow diagram

ACQUISITION

The acquisition is accomplished as a *parallel frequency space search* which uses a Fourier transform to perform a transformation from the time domain into the frequency domain. Figure 6 shows the flow diagram for the parallel frequency space search algorithm.

The incoming signal is multiplied by a locally generated PRN sequence, with a code corresponding to a specific satellite and a code phase between 0 and 1 023 chips. The resulting signal is transformed into the frequency domain by a Fourier transform. The Fourier transform could be implemented as a Discrete Fourier Transform (DFT) or a Fast Fourier transform (FFT). The FFT is the faster of the two but it requires an input sequence with a radix-2 length, that is 2^n, $n \in \{1, 2, 3, \ldots\}$.

With a perfectly aligned PRN code, the output of the Fourier transform will show a distinct peak in magnitude. The peak will be located at the frequency index corresponding to the frequency of the continuous wave signal and thereby the frequency of the carrier wave signal.

The accuracy of the determined frequency depends on the length of the DFT. The length corresponds to the number of samples in the analyzed data. If 1 ms of data is analyzed, the number of samples can be found as 1/1 000 of the sampling frequency. That is if the sampling frequency is $f_s = 10$ MHz, the number of samples is $N = 10\,000$.

With a DFT length of 10 000, the first $N/2$ output samples represent the frequencies from 0 to $\frac{f_s}{2}$ Hz. That is, the fre-

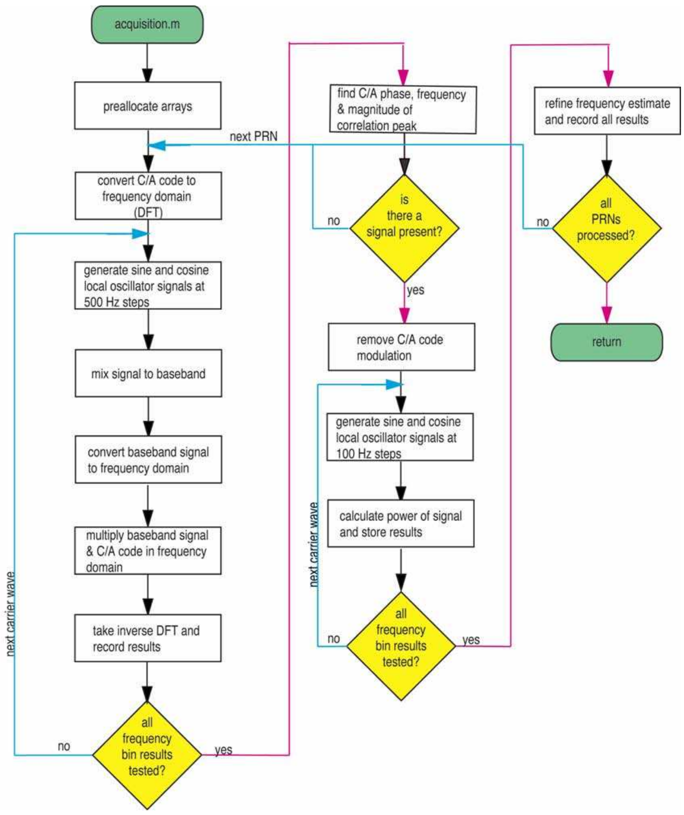

Figure 6: Flow diagram of the parallel frequency space search algorithm

quency resolution of the output is

$$\Delta f = \frac{f_s/2}{N/2} = \frac{f_s}{N}. \tag{1}$$

In case of $f_s = 10\,\text{MHz}$ the resulting frequency resolution is

$$\Delta f = \frac{10\,\text{MHz}}{10\,000} = 1\,\text{kHz}. \tag{2}$$

In this case, the accuracy of the estimated carrier frequency is 1 kHz.

COMPLETE TRACKING BLOCK

Figure 7 shows an optimized version of the combined tracking loops. Here the I and Q inputs to the phase discriminator are the I_p and Q_p correlation from the code tracking loop.

Figure 8 illustrates the same activity; but in the form of a flow diagram.

NAVIGATION DATA DECODING

The final signal processing function of the receiver is to decode the 50 Hz navigation data stream. The bits are clearly visible in the inphase channel of the Costas loop.

Processing proceeds as follows

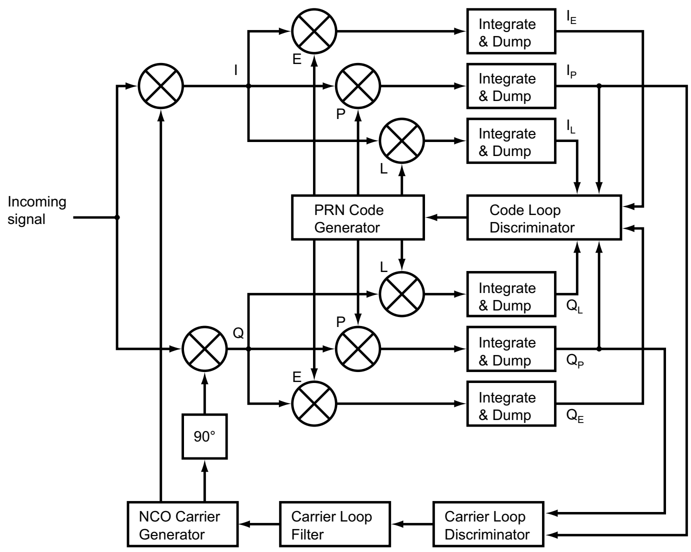

Figure 7: The block diagram of a complete tracking channel on the GPS receiver

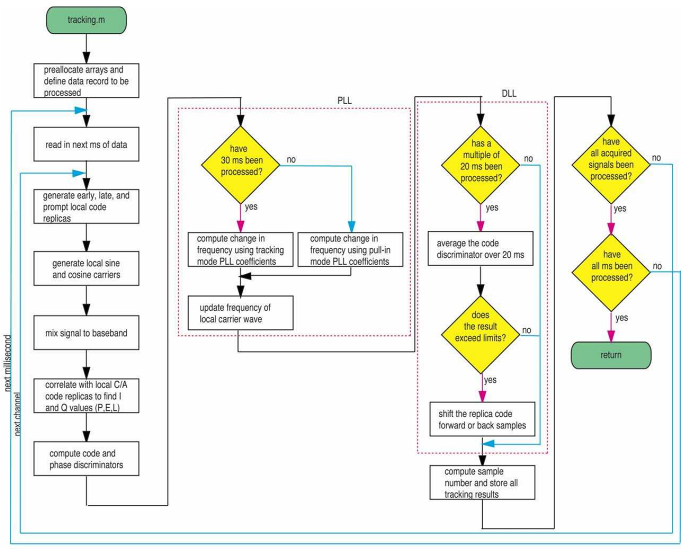

Figure 8: Flow diagram for a complete tracking channel on the GPS receiver

- Bit synchronization: determine the start/stop of each bit
- Frame synchronization: determine the start/stop of the navigation data frames
- Data decoding: extract the necessary parameters from the transmitted '1's and '0's in the first three subframes; they are required for the position solution.

The ICD-GPS-200 (1991) and SPS (1995) are outstanding references and describe in detail the structure of the navigation data message.

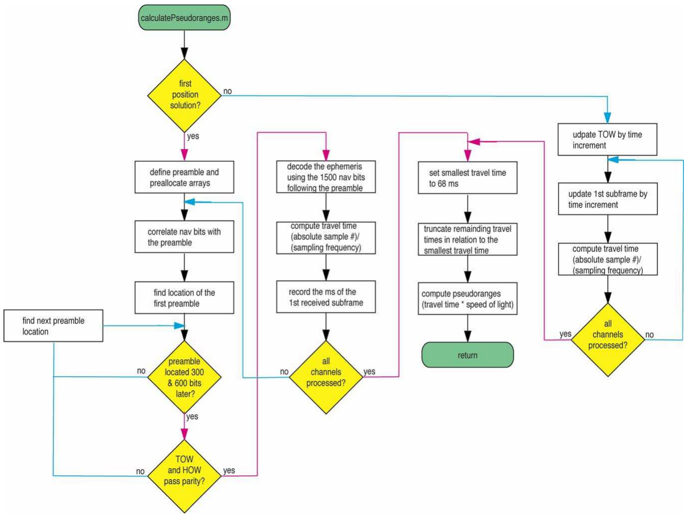

Figure 9: Flow diagram for computing pseudoranges

Navigation data is partitioned into five subframes, each six seconds long (300 bits). 30 seconds are required to obtain all five subframes. Subframes 4 & 5 are further sub partitioned across 25 frames (12.5 minutes).

As a result, the Matlab implementation processes data in 36 second blocks to ensure all navigation data parameters have been obtained to enable the position solution.

COMPUTING PSEUDORANGES

Pseudorange computations are based on selecting a “master” satellite and then assigning that satellite an arbitrary propagation time delay of 70 ms or the corresponding 20 985 472.06 m pseudorange. Then the other satellites which have been acquired and tracked within the data set have pseudorange set relative to this “master” satellite. The relative time offset is determined from decoding the navigation data stream associated with each satellite. Then, through the process of the least squares, a resulting absolute clock offset and position solution are determined.

COMPUTATION OF RECEIVER POSITION

The receiver positions are computed by use of the Easy-Suite software, see Borre (2003). The corresponding flow diagram is presented in Figure 10.

The result of the sample data is plotted in Figure 11. It shows that the repeatability within a short time of period is a few meters in the horizontal direction.

DEVELOPMENT AND OUTLOOK

The book, Borre et al. (2006), will be available early 2006 and should provide a basis for software GNSS receiver courses. It will establish a reference textbook and a complete GPS/Galileo GPL Matlab framework to be used for algorithm development and testing. Focus is on algorithm research and development in order to augment our knowledge concerning signals and algorithms.

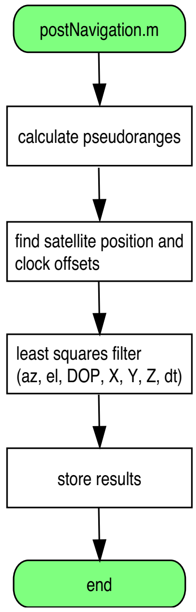

Figure 10: Flow diagram for the position computation

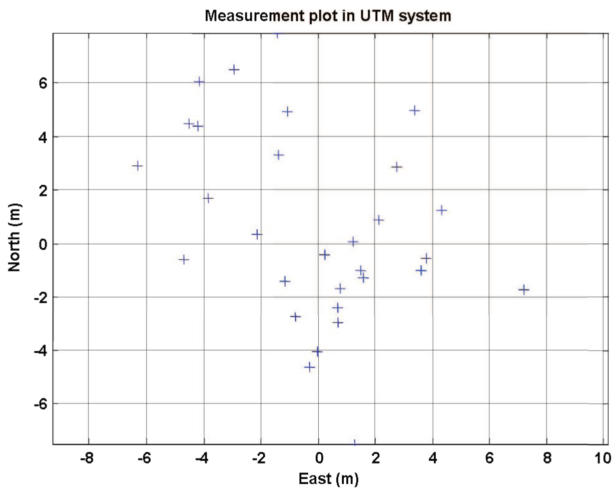

Figure 11: Positions computed for every second

The computation speed of the present version is approximately 6 times real-time.

The current receiver developments include

- Support for Galileo signals
- Support for EGNOS signals.

REFERENCES

Akos, Dennis (1997). *A Software Radio Approach to*

Global Navigation Satellite System Receiver Design. Ohio University, Athens, OH.

Borre, Kai (2003). The GPS easy suite—Matlab code for the GPS newcomer. *GPS Solutions*, 7:47–51.

Borre, Kai, Akos, Dennis, Bertelsen, Nicolaj, Rinder, Peter, & Jensen, Søren Holdt (2006). *A Software-Defined GPS and Galileo Receiver: Single-Frequency Approach*. Birkhäuser, Boston, MA.

ICD-GPS-200 (1991). Interface control document. ICD-GPS-200, Arinc Research Corporation, 11 770 Warner Ave., Suite 210, Fountain Valley, CA 92 708.

Pany, Thomas, Kaniuth, Roland, & Eissfeller, Bernd (2005). Deep integration of navigation solution and signal processing. In *18th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Long Beach, CA.

SPS (1995). Global positioning system standard positioning service signal specification. U. S. Department of Defence.