A Flexible and Portable Multiband GNSS Front-end System

Alexander Ruegamer, Frank Foerster, Manuel Stahl, Guenter Rohmer
Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany

BIOGRAPHY

Alexander Ruegamer received his Dipl.-Ing. (FH) degree in Electrical Engineering from the University of Applied Sciences Wuerzburg-Schweinfurt, Germany in 2007. Since the same year he works at the Fraunhofer Institute for Integrated Circuits IIS in the field of GNSS front-end receiver development. He was promoted to Senior Engineer in February 2012. His main research interests focus on GNSS multi-band reception, integrated circuits and immunity to interference.

Frank Foerster has received his Dipl.-Ing. degree in Electrical Engineering from the University of Erlangen-Nuremberg, Germany, in 2003. Since then, he is at the Fraunhofer Institute for integrated circuits in Erlangen as a system design engineer. Currently he is involved in several navigation and communication projects where he is developing, implementing and testing the analog part.

Manuel Stahl received his Dipl.-Inf. degree in Computer Sciences from the University of Wuerzburg, Germany in 2009. Since the same year he works at the Fraunhofer Institute for Integrated Circuits IIS in the field of GNSS receiver software development for embedded systems and mobile robotics.

Guenter Rohmer received his Dipl.-Ing. degree in Electrical Engineering in 1988 and the PhD in 1995 from the University of Erlangen, Germany. Since 2001 he is head of a department at the Fraunhofer Institute for Integrated Circuits dealing with the development of components for satellite navigation receivers, indoor navigation and microwave localization systems.

ABSTRACT

In this paper, the latest version of the Fraunhofer USB front-end development, called the Flexiband, is presented. Thanks to its new modular concept, the Flexiband not only supports a set of pre-selected configurations but can also be set up for multi-antenna inputs, user selectable bandwidth, intermediate frequencies, and customized ADC sampling rates and resolutions. The Flexiband hardware and software components are described in detail. Different configurations are shown and one example of a recorded Galileo E1/E5a signal is analyzed.

INTRODUCTION

In a few years, at least four independent but interoperable GNSS - GPS, GLONASS, Galileo, and COMPASS - will be available on several frequency bands. Consequently, multiband GNSS reception is getting more and more popular both in the academic world and for commercial applications. The upcoming GNSS span over most of the L-band. For GPS, Galileo, and GLONASS alone, signals are broadcast on 8 different frequency bands (E5a/L5, G3, E5b, L2/L2C, G2, E6, E1/L1, G1) with center frequencies ranging from 1176 MHz (Galileo E5a, GPS L5) to 1609 MHz (GLONASS G1) and with main-lobe bandwidth varying from 2 MHz (GPS L1 C/A, L2C) to over 52 MHz (Galileo E5 AltBOC), see Figure 1. While the benefits of and demand for multiband GNSS reception and processing are obvious, a flexible, portable, and affordable front-end recording solution that can easily be adapted to the reception of all these bands has yet to be made available on the market.

Fraunhofer IIS previously developed a front-end, called the *L125 Triband* USB front-end, which allows a fixed frequency recording of L1/E1, L2 and L5/E5a via two USB 2.0 data streams with up to 40 MSPS sampling rate, a 2 or 4 bit analog-to-digital converter (ADC) resolution and one antenna input. This USB front-end has been used and validated in numerous scientific and industrial projects e.g. [1], [2], [3].

An upgraded version of this USB front-end was later developed to enable the simultaneous processing of the complete lower (1145-1310 MHz) and upper (1545-1630 MHz) frequency L-bands using two 410 MHz ADCs. A flexible FPGA signal conditioning enabled the selection of various GNSS signal combinations transmitted over three USB 2.0 channels. The wide reception bandwidths allowed many interesting and demanding applications using software GNSS receivers [4], [5]. Even though this front-end already enabled the reception of all GNSS signals of interest, the high

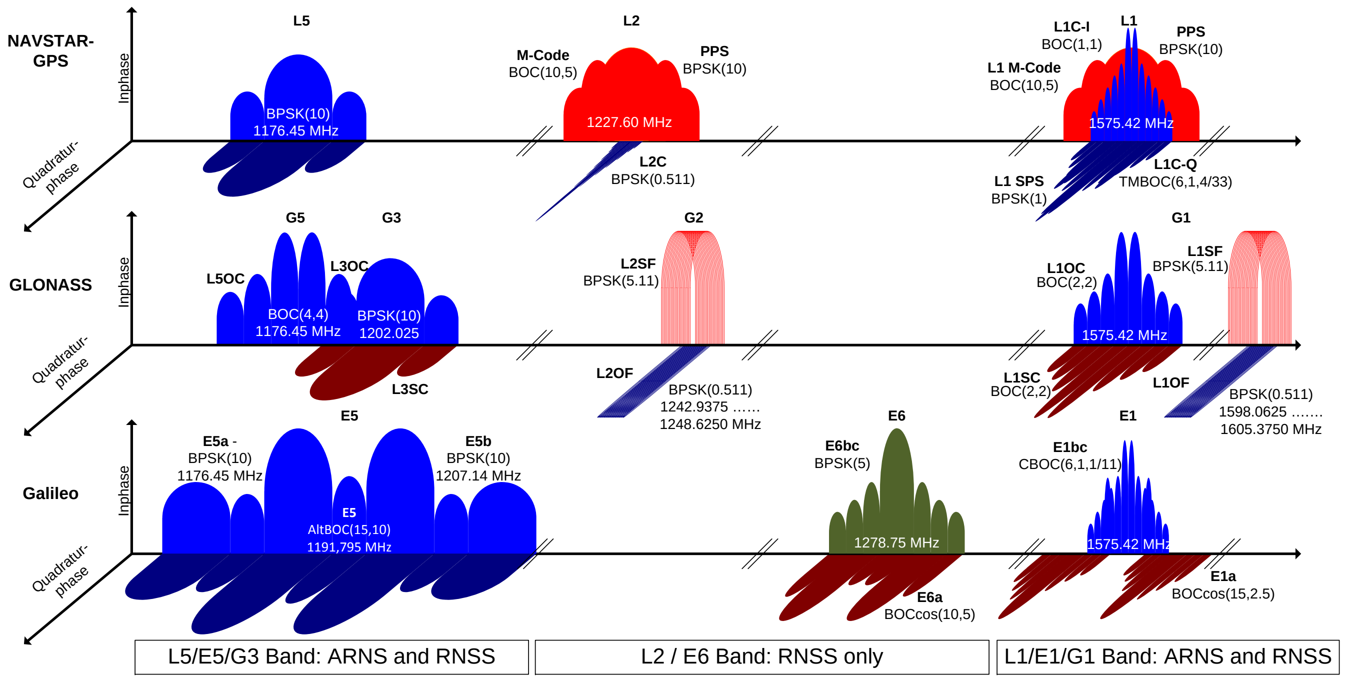


Figure 1. Current and upcoming GPS, GLONASS and Galileo signals

sampling rates of the required ADCs and high speed FPGA were expensive and the high speed digital data processing resulted in high power consumption. Therefore this version was not suited for users requiring a portable and affordable front-end solution.

Increasing customer requests for portable and flexible solutions concerning frequency band selection, adjustable sampling and intermediate frequencies, and multi-antenna support led to a complete redesign of this USB front-end concept.

In this paper, the latest version of the Fraunhofer USB front-end development, called the Flexiband, is presented. The paper is organized as follows: First the system architecture with its different hardware components is introduced. Then the Flexiband software is presented and its capability discussed. Finally a practical application setup with Galileo E1/E5a recording is given before conclusions are drawn.

SYSTEM ARCHITECTURE

The Flexiband's system architecture comprises four different blocks: The RF modules, a base and interface unit, and a software component. Each of these blocks will be described in detail in the following.

RF Module

A picture of the RF module is shown in Figure 2. Its block diagram is depicted in Figure 3. According to the Friis' Formula, the first low noise amplifier (LNA) is used to guarantee a low overall noise figure (NF). A three-way splitter can be used to distribute the amplified antenna input signal from one primary RF module to up to two other secondary RF modules. If this distribution is used (e.g. when only one wide-band antenna is used) the RF interface following the splitter is then the RF input for the secondary RF modules.

The RF bandpass filter attenuates possible out-of-band interferences. The RF bandpass filter is mounted on a special footprint adapter to enable the use of different filters, e.g. narrow-band SAW filter or wide-band ceramic filter with different amplitude and group delay behaviour.

A commercial off the shelf I/Q downconverter RF IC is used providing a zero-IF or low-IF architecture depending on its local oscillator (LO) setting. The LO (and therefore the resulting intermediate frequency (IF)), the amplifier gains, and the anti-aliasing low-pass filter bandwidth can be configured in a flexible way using SPI and I2C interfaces. These parameters can also be set by the user using the Flexiband software. The specific parameters of the filter used, the default configuration of the RF IC, and serial number of the RF module are stored in an EEPROM that can be read out by the Flexiband software.

The RF IC budget plan concerning its gain, noise figure (NF) and input intercept point 3rd order (IIP3) is given in Table 1 for one possible L1 configuration. Since the RF IC has a 75 Ω input impedance, a broadband resistive matching network is used for interconnection with the 50 Ω impedance filters, adding some additional implementation loss. As can be seen from the noise figure, the Flexiband front-end should be used either with an active antenna or with a passive antenna and an additional external LNA to

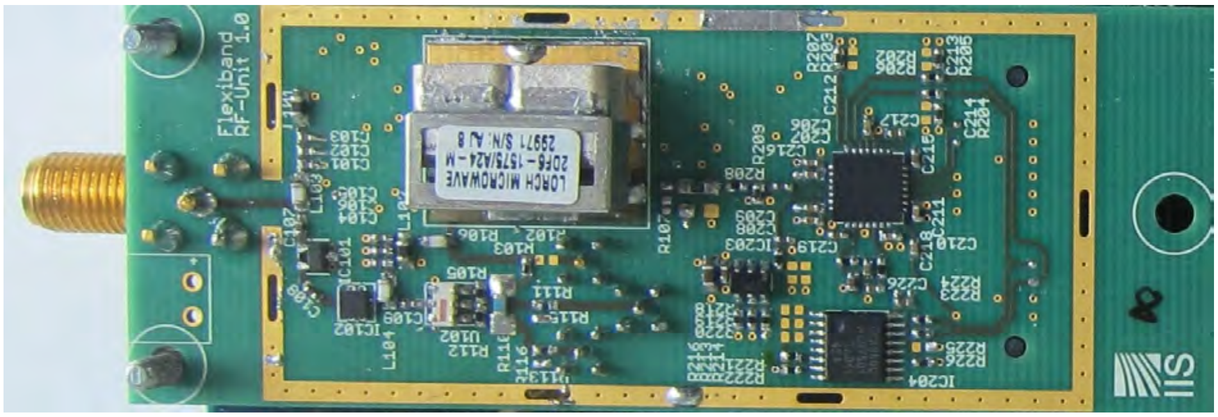


Figure 2. RF module

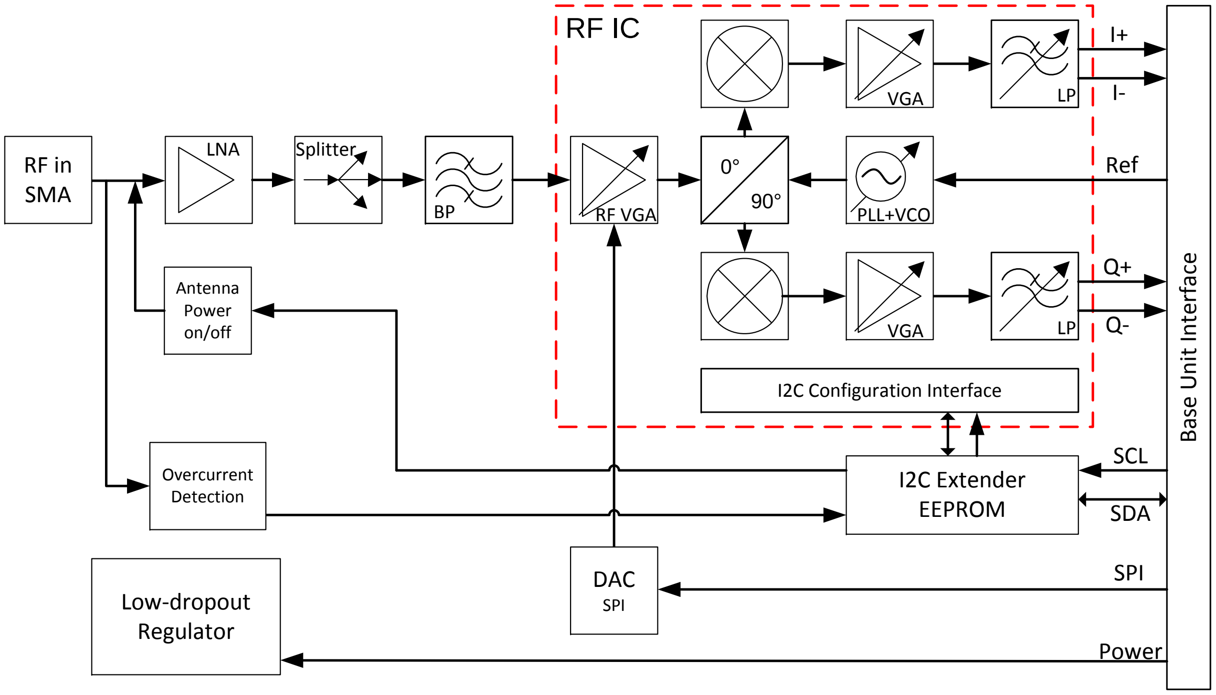


Figure 3. RF module block diagram

Table 1. Gain, noise figure and IIP3 budget plan of an exemplary L1 RF module

	LNA	Splitter	RF-Filter	$50 \to 75\,\Omega$	RF IC	**overall**
Gain [dB]	13	-6	-1.2	-6.3	75	**74.5**
NF [dB]	1.2	6	1.2	6.3	8.2	**9.4**
IIP3 [dBm]	25	50	50	50	-40	**-39.5**

lower the overall noise figure. The DC power supply for the active antenna (phantom power) can be switched on and off using the Flexiband software.

The Flexiband housing can host up to three RF modules that can receive any L-band signal with up to 80 MHz RF bandwidth. Intermediate frequencies, RF- and anti-aliasing filter bandwidths, as well as the number of antenna inputs can be customized. The three RF-modules can be fed from a single antenna using the described splitter approach or have separate antenna inputs, making the Flexiband very attractive for applications such as reflectometry or for antennas' comparison in real-world measurement campaigns.

Base Unit

The base unit board can carry up to three RF modules. A picture of the base unit is shown in Figure 4 and its block diagram is depicted in Figure 5.

A microcontroller is used to initialize the RF modules' RF IC using an I2C protocol, to control the variable gain amplifiers using SPI, and to read out the RF modules' EEPROM content. The microcontroller's firmware can be upgraded using the Flexiband software, enabling the implementation of corrections and/or new features if necessary.

The base unit provides three dual-channel ADCs featuring up to 100 MSPS sampling rate and up to 10 bit resolution. In the default configuration the three RF unit outputs are simultaneously and coherently sampled using a dual-channel I/Q ADC with a 80 MSPS sampling rate and a 10 bit resolution.

The ADC outputs are directly connected to an FPGA (Xilinx Spartan 3) where the data streams can be mixed, filtered, down-sampled, bit width reduced, and finally multiplexed including an error detection protocol. All parameters, including the reception bandwidth, the sampling rate, and the intermediate frequency can be configured as a firmware feature. Moreover the FPGA can be used to implement features such as automatic gain control (AGC) or pulse detection and blanking. An SDK for the FPGA will be available as an additional option.

One possible L1, L2, L5 configuration (Triple band configuration #6 in Table 4) of the signal conditioning is shown in the block diagram in Figure 5. The FIR filters limit the signal's baseband bandwidth to approx. 9 MHz (one-sided) so that the signal can be down-sampled from 80 to 20 MSPS. The L1 and L2 signal streams are reduced to 2 bit I/Q and multiplexed to one 8 bit@20 MSPS data stream. The L5 signal stream is reduced to 4 bit I/Q and fed to a second 8 bit@20 MSPS data stream, as depicted in Figure 6. The interface board connector features a synchronous 16 bit interface. For USB 3.0 the data is streamed via a single USB isochronous endpoint while for USB 2.0 two separate channels are used to respectively stream the L1/L2 and L5 data.

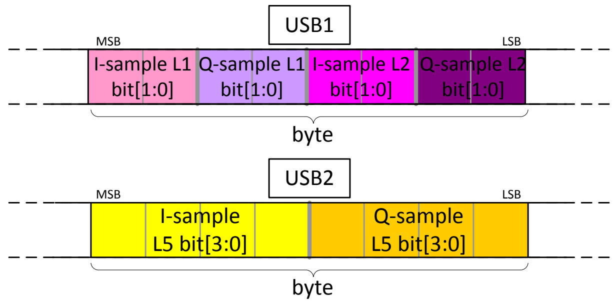


Figure 6. Exemplary data stream multiplex for L125 configuration

A dedicated clock generation and distribution chip is used to coherently derive all the frequencies required for the RF modules, the ADCs, and the FPGA from one single source. By default, an internal 10 MHz temperature controlled crystal oscillator (TCXO) is used as the reference. A buffered output of this clock is also available at the Flexiband housing as an output signal for synchronizing other equipment. Optionally, a switch on the Flexiband housing also allows the use of an external reference source instead of the internal one.

The base unit board can be powered either by a power socket on the housing or from the interface board.

Error Detection Protocol

For a GNSS receiver it is essential to have a continuous data stream. Any lost sample will disturb the acquisition and tracking algorithms. Samples can be lost either inside the USB chip's FIFO (e.g. when USB-packets cannot be fetched soon enough by the user's PC, the FIFO will overflow) or during the USB-ISO-transfer (since there is no retransmission on errors, corrupt packets are completely lost).

Since USB is a packet oriented and asynchronous interface and since the high speed transfer mode provides only low level error detection, a mechanism is implemented to detect and compensate data loss and to maintain a continuous data stream.

The USB-interface transfers 1 kByte data packets to the PC. In order to detect a packet loss, a signature is embedded in every packet. This signature consists of a 4 byte preamble and a 2 byte counter so that it overrides 6 byte of data which has a negligible effect on GNSS signals. In principle, it has the same effect as pulse blanking which is known to be negligible when short enough. For every new packet the counter is incremented by one. Using a transfer rate of approx. 20 MByte/s the counter will overflow every 3.4 seconds.

For USB 3.0 the same data stream is used. Since the 16 bit data is transferred over a single stream, this results in a duplication of each byte of the preamble and counter, see Table 3.

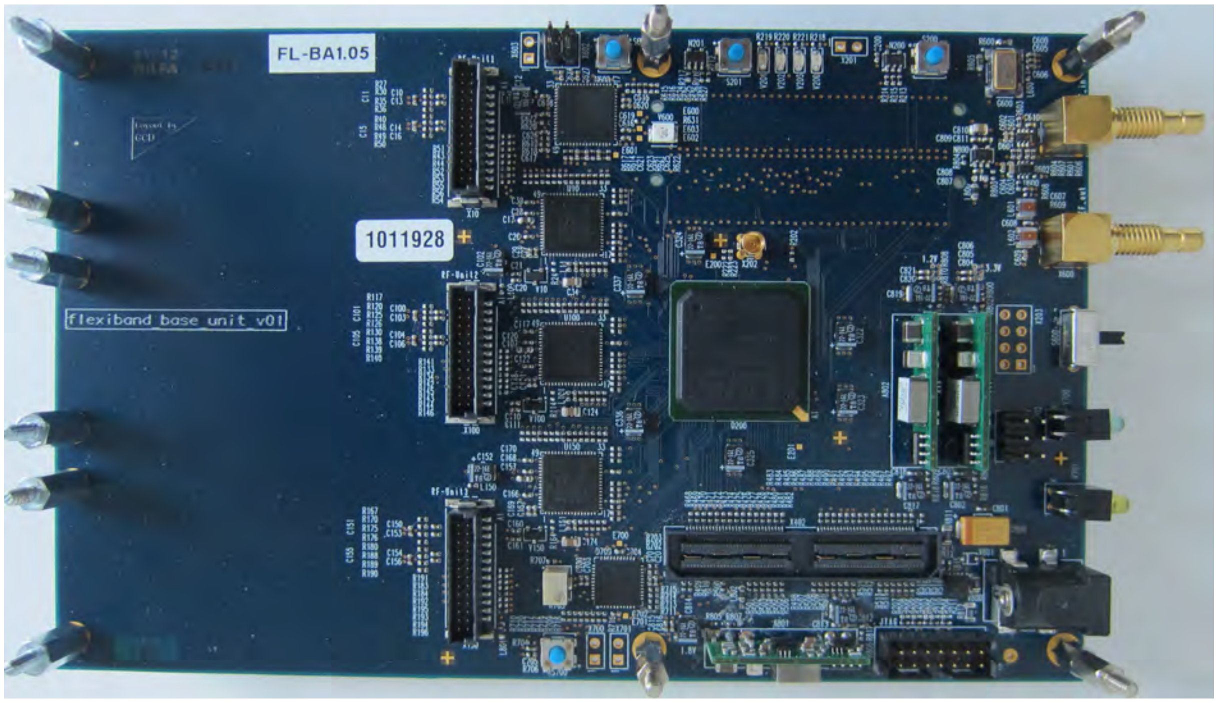


Figure 4. Base unit

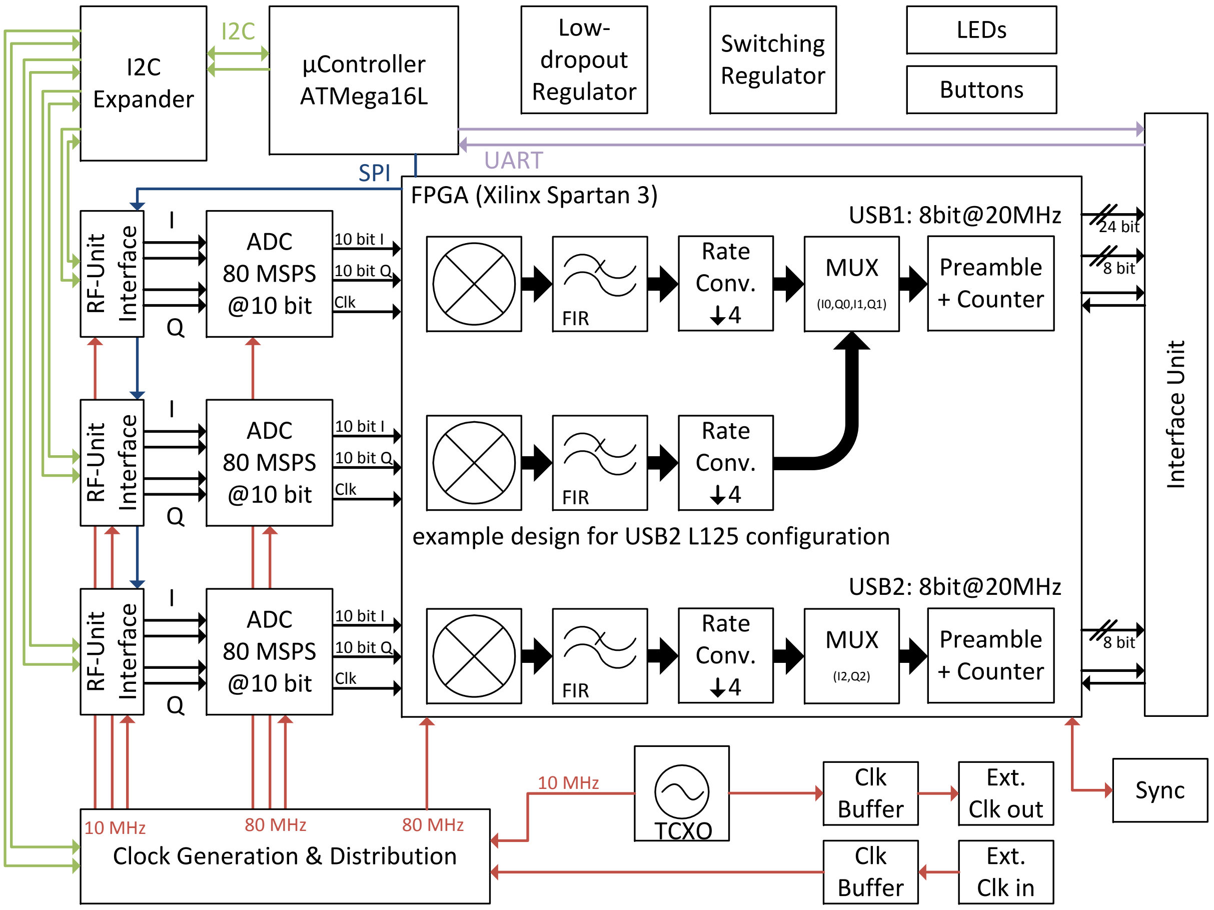


Figure 5. Base unit block diagram

Table 2. Format of the embedded error detection signature (USB 2.0)

Preamble				Counter	
0xAA	0xBB	0xCC	0xDD	0xUU	0xLL

Table 3. Format of the embedded error detection signature (USB 3.0)

Preamble				Counter	
0xAAAA	0xBBBB	0xCCCC	0xDDDD	0xUUUU	0xLLLL

While retrieving the USB-data from the USB-driver's application programming interface (API) the user can check the counter value of every incoming packet. If the expected counter value is located within the raw data stream then it can be assumed that everything was transferred correctly. If not, the complete 1024 bytes are searched for the preamble and its associated counter value to determine and report the number of missing bytes. These procedures are done in real time using the Flexiband API. Detected errors can be visualized in the Flexiband graphical user interface (GUI). In post-processing the missing packages can e.g. be filled up with zeros to maintain proper time synchronisation.

When the error detection and correction is completed, the preamble and counter value can be removed in post-processing and replaced by zeros to preserve the time reference.

Interface Unit

The interface unit can connect the FPGA output to either a computer, a GNSS digital baseband receiver, or a data recorder. Currently both USB 2.0 and a USB 3.0 interface boards are available, and both feature a parallel data output port that can be used, e.g. together with a digital signal recording device. But the interface unit is not limited to USB. Instead of using a USB connection the interface board can be replaced by other interface units to provide different physical interfaces such as a direct connection via PCIe, ExpressCard, Ethernet or other standards.

USB 2.0 Interface Unit

In the default USB 2.0 configuration the interface board transfers the baseband output data to a standard PC via two high speed USB 2.0 Cypress FX2 microcontrollers. The USB 2.0 interface board and its block diagram are depicted in Figure 7(a) and 7(c), respectively.

Keeping in mind that data integrity is critical (e.g. to ensure that no sample gets lost), 40 Mbyte/s was selected as the fastest rate that would still works reliably in practice. Moreover this data rate is split using two USB 2.0 connections. The reason is twofold. First, one USB 2.0 connection can supply a power of 5 V at only 500 mA. With two parallel connections the Flexiband can be completely be powered via USB, circumventing the need for an external power supply. This is not covered by the USB specifications but is a common practice for external hard drives. There is a protection circuit included on the interface board that guarantees that the base unit is only powered when both USB cables are plugged in. Second, one USB-ISO-transfer is limited to 24 Mbyte/s. Using only 20 Mbyte/s per channel includes a safety margin and should be achievable on any recent PC.

The first USB controller also ensures the communication from the Flexiband software to the Flexiband hardware (microcontroller and FPGA) using specific vendor requests. This enables updates of the firmware of the USB controllers themselves, of the base unit microcontroller via its bootloader and of the FPGA design using a JTAG generator on the FX2 firmware.

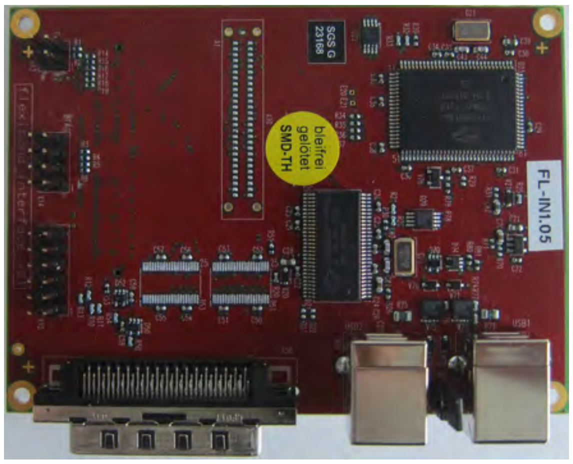
Since the GNSS data transfer requires both real-time capability and high data rates, the USB isochronous transfer mode is the only appropriate mode. Only this mode allows a device to reserve a defined amount of bandwidth (max 24 Mbyte/s) with guaranteed latency [6]. This USB mode uses an internal CRC16 checksum to report transmission errors. In isochronous data transfer mode a retransmission is not foreseen. That is the reason why additional information, namely the preamble and counters, are embedded in the USB-stream as previously explained.

The used FX2 USB 2.0 microcontroller features an 8 bit data interface. Thus the base unit FPGA prepares two 8 bit data streams in each of which the aforementioned preamble and counter values are embedded. These two data streams can be then transfered in an asynchronous way via USB. Since a signature is embedded in each USB-stream the time relation between these streams can also be verified in post-processing. Packets with the same counter value are sample-synchronous and relate to the same time slice.

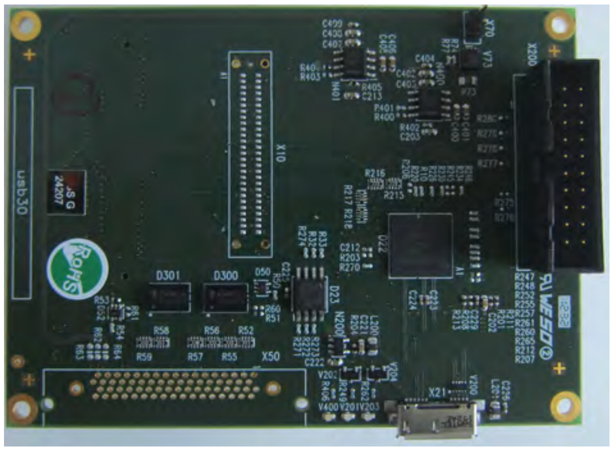
Most frequently it is the hard disk speed that is the bottleneck and leads to packet losses when the output data of the FX2 buffer cannot be polled fast enough. An additional ring-buffer in the FPGA could possibly reduce the hard timing requirements. Nevertheless using RAID arrays or solid-state drives (SSDs) for data recording has shown satisfying results.

USB 3.0 Interface Unit

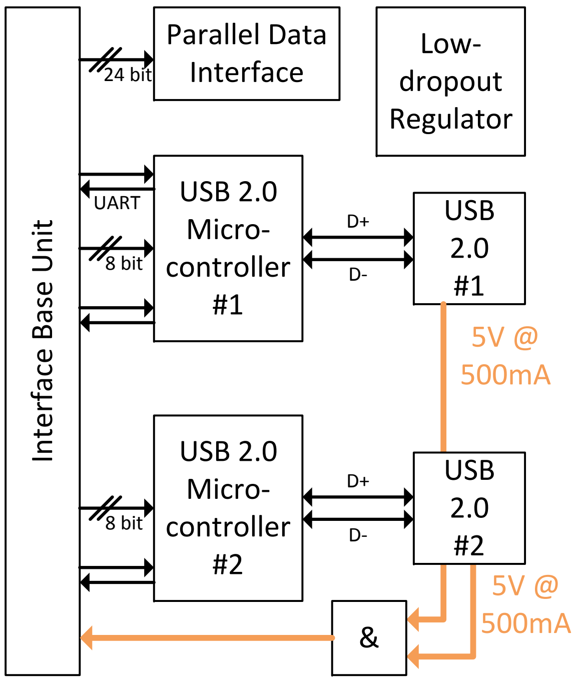
An alternate interface board option is the USB 3.0 interface. It features a nominal data rate of 5 Gbit/s and a power supply of 5 V at 900 mA. Thus one USB 3.0 port is sufficient regarding the data rate and can supply power for the



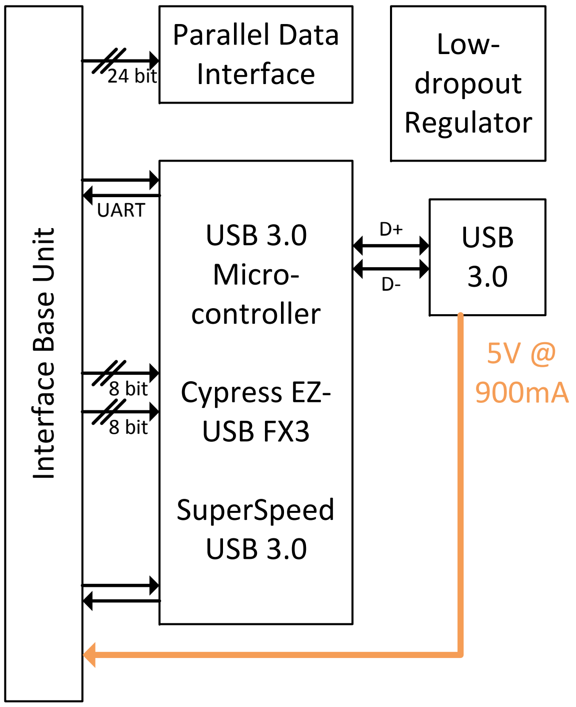
(a) Block diagram of USB 2.0 interface board



(b) USB 3.0 interface board



(c) Block diagram of USB 2.0 interface board



(d) Block diagram of USB 3.0 interface board

Figure 7. USB interface units

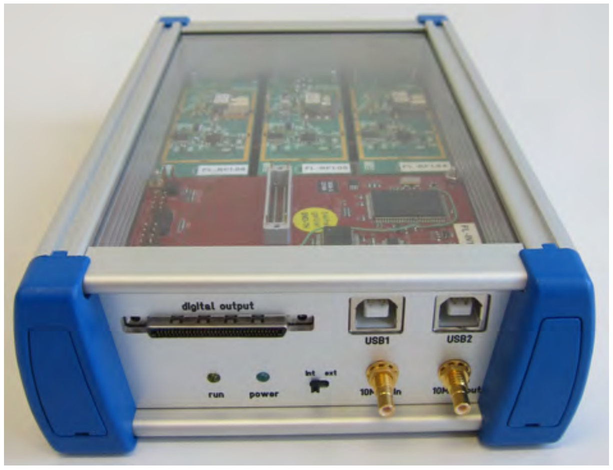


Figure 8. Flexiband housing with USB 2.0

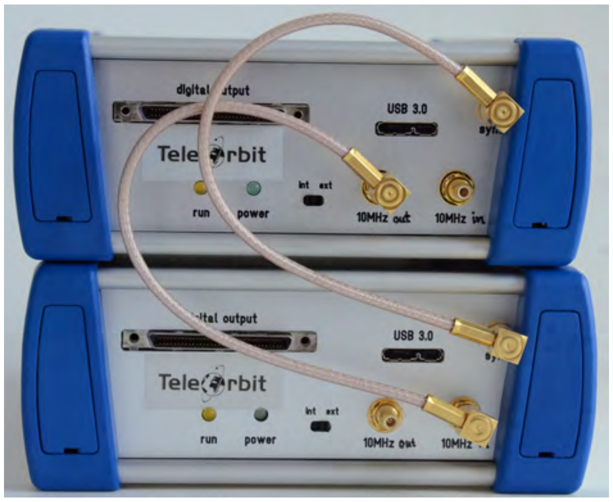


Figure 9. Synchronization of two Flexiband USB 3.0 units

Flexiband and one active antenna. Since more and more PCs are equipped with USB 3.0 ports and use SSDs instead of hard disks the USB 3.0 interface unit is likely to become the new Flexiband default connection standard soon.

The USB 3.0 interface board and its block diagram are depicted in Figure 7(b) and 7(d), respectively. As can be seen, it uses one super speed USB 3.0 Cypress FX3 micro-controller.

The FX3 chip uses a 16 bit data interface via a direct memory access (DMA) from the FPGA to the FX3 and from the FX3 to the PC. An embedded ARM core on the FX3 is in charge of the required memory and buffer management, and of the setup of the DMA channels. The data transfers are handled by a configurable state machine inside the FX3 with very few CPU interactions. The previously described error detection protocol with preamble and counters is also used here.

Flexiband Housing

The Flexiband housing is both compact (approx. 188x-125x50 mm^3) and lightweight. Thanks to its energy efficient design the Flexiband front-end can be powered by two USB 2.0 or one USB 3.0 interface making it perfectly suited for mobile recording campaigns with e.g. a notebook as a data recording device.

Synchronization of Several Flexibands

Several Flexiband front-ends can be synchronized with each other. For the reference frequency coupling it is possible to use either the internal reference oscillator or a high quality external reference such as the reference clock from an IMU or from a reference trajectory system.

When synchronizing two Flexiband units, a dedicated synchronisation link is needed, coupling the asynchronous reset of the base unit FPGAs. The synchronization link works as a wired OR connection. As long as each Flexiband unit holds the sync line in reset state, all Flexiband units stay in reset mode. Using a special splitter cable it is possible to use an arbitrary number of Flexibands in parallel. With the help of the preamble and counter it is guaranteed that the data streams are synchronized.

Systematic delays between the sampling points of the connected Flexiband front-ends can be calibrated out using e.g. fractional delay filters in the FPGA signal conditioning design.

FLEXIBAND SOFTWARE

The Flexiband software is threefold. It consists of a USB driver, an API, and a visualization and recording software.

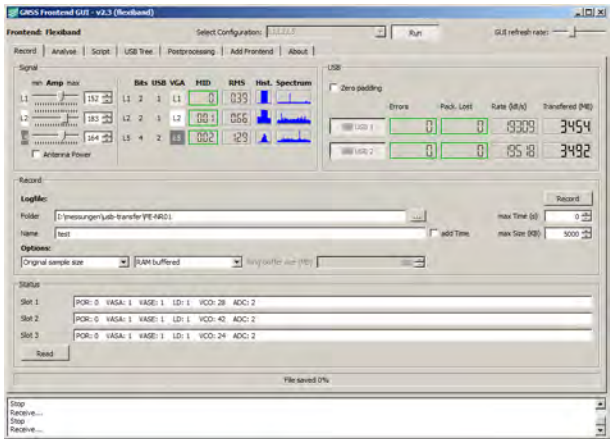
For front-end operation, a USB-driver that manages the high speed isochronous data streams is necessary. The driver delivered with the Flexiband is recommended for Windows. On Linux the open source library libusb-1.0 can be used. When using a USB 2.0 interface unit, the installation process on the PC has to be run once for each USB interface. The USB 2.0 devices then appear as ”GTEC RFFE USB 1” and ”GTEC RFFE USB 2” in the Windows device manager. When using a USB 3.0 interface unit the FX3 first connects in bootloader mode. The supplied driver can then load the FX3 firmware, reconnect the FX3 and handle the device operation.

Flexiband API

The API supplied for both Windows and Linux operating systems gives the user the possibility to realize a complete, real-time software-defined GNSS receiver with multiband and multisystem capability. Figure 10 shows a UML class diagram of the API.

Flexiband GUI

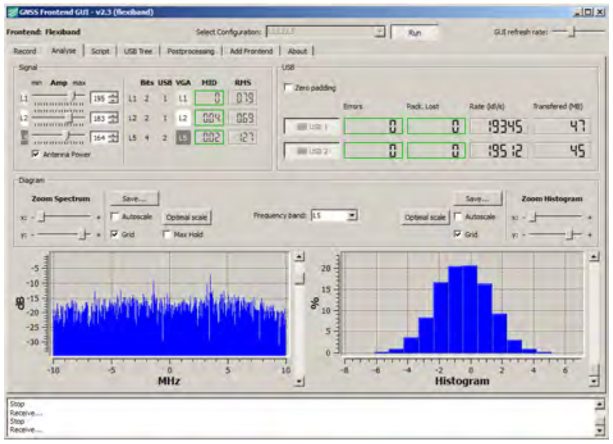
The graphical user interface (GUI) is implemented with the platform independent Qt library. Therefore versions



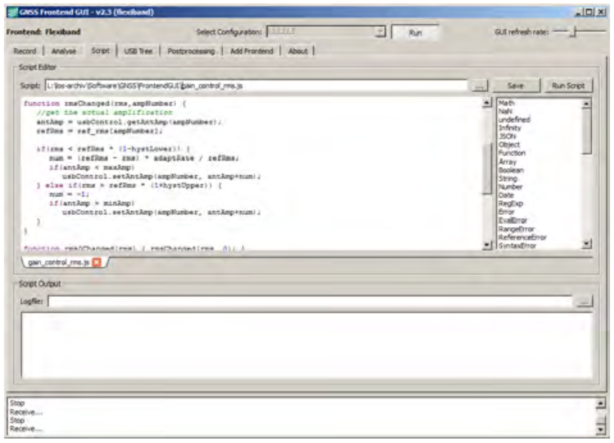
```
Slot 1  POR: 0  VASA: 1  VASE: 1  LD: 1  VCO: 28  ADC: 2
Slot 2  POR: 0  VASA: 1  VASE: 1  LD: 1  VCO: 42  ADC: 2
Slot 3  POR: 0  VASA: 1  VASE: 1  LD: 1  VCO: 24  ADC: 2
```

```
Stop
Receive....
Stop
Receive....
```

(a) Record tab



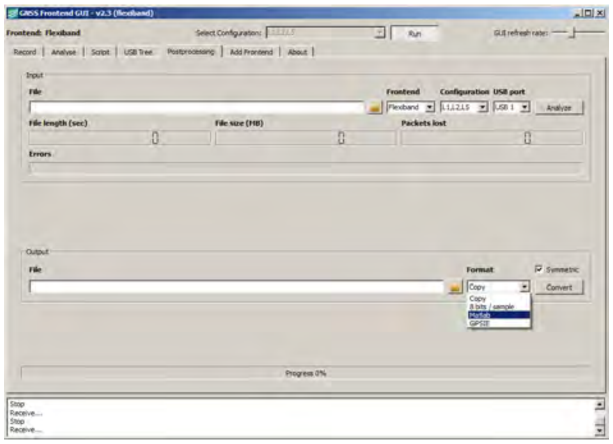
(b) Analyze tab



```
function rmsChanged(rms,ampNumber) {
   //get the actual amplification
   antAmp = usbControl.getAntAmp(ampNumber);
   refRms = ref_rms[ampNumber];

   if(rms < refRms * (1-hystLower)) {
      num = (refRms - rms) * adaptRate / refRms;
      if(antAmp < maxAmp)
         usbControl.setAntAmp(ampNumber, antAmp+num);
   } else if(rms > refRms * (1+hystUpper)) {
      num = -1;
      if(antAmp > minAmp)
         usbControl.setAntAmp(ampNumber, antAmp+num);
   }
}
```

(c) Script tab



(d) Post processing tab

Figure 11. Flexiband front-end GUI

Table 4. Exemplary of different Flexiband configurations readily available

Configuration Nr.	Frequency band	Bandwidth [MHz]	Intermediate freq. [MHz]	Sampling rate [MSPS]	Sample bitwidth	USB data rate [Mbit/s]
Single band						
1	L1/E1	8	4	20	4	80
2	E5	50	0	80	2 I/Q	320
Dual band						
3	L1/E1	16	8	40	4	320
	L2/L2c	16	8	40	4	
4	L1/E1	8	4	20	4	160
	G1	10	5	20	4	
Triple band						
5	L1/E1	18	0	20	2 I/Q	320
	L5/E5a	18	0	20	2 I/Q	
	E5b	18	0	20	4 I/Q	
6	L1/E1	18	0	20	2 I/Q	320
	L2/L2c	18	0	20	2 I/Q	
	L5/E5b	18	0	20	4 I/Q	
7	L1/E1/G1	38	0	40	4 I/Q	960
	L1/E1/G1	38	0	40	4 I/Q	
	L1/E1/G1	38	0	40	4 I/Q	

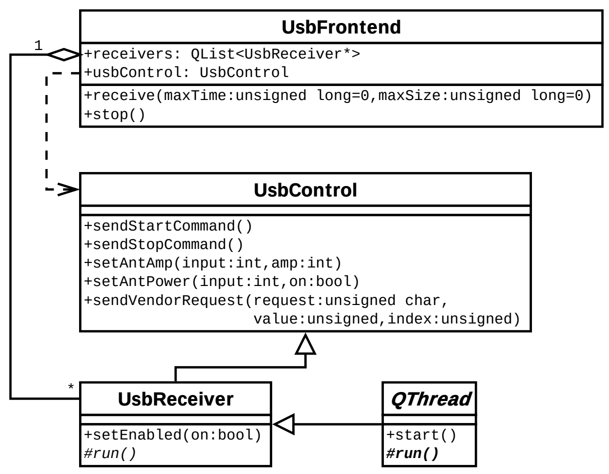

UsbFrontend
+receivers: QList<UsbReceiver*>
+usbControl: UsbControl
+receive(maxTime:unsigned long=0,maxSize:unsigned long=0)
+stop()
1
UsbControl
+sendStartCommand()
+sendStopCommand()
+setAntAmp(input:int,amp:int)
+setAntPower(input:int,on:bool)
+sendVendorRequest(request:unsigned char, value:unsigned,index:unsigned)
*
UsbReceiver
+setEnabled(on:bool)
#run()
QThread
+start()
#run()

Figure 10. Flexiband API class diagram

are available for Windows as well as for GNU/Linux. The Flexiband GUI uses the previously described API.

The four main tabs of the Flexiband software for a USB 2.0 interface unit with an L1, L2, L5 triple band configuration are depicted in Figure 11.

Flexiband GUI - Recording Settings

The *record* tab, depicted in Figure 11(a), can be used to set and monitor all the useful recording parameters. The sliders control the variable gain amplifiers (VGA) of the connected RF modules. The check box controls whether the antenna power is be on or off. The widget group gives some statistic information about the received frequency bands such ADC resolution, USB port used, arithmetic mean and root mean square of the received samples, histogram and spectrum view. Since the *Run* button is pushed, the USB group displays the number of USB errors and lost packets as well as the current transfer rate in kByte/s and the overall size of the transferred data in MBytes. If the *zero padding* check box were activated, missing packets in the data stream would automatically be replaced by zeros while recording. Since this can lead to a higher disk load, it is better left for post-processing. The location and the name of the recorded file can be set in the record sub window. If desired, the start time of the recording can be automatically prefixed to the name of the file. Finally, it is possible to specify either a maximum recording time in seconds or a maximum recording size in kBytes. A value of zero means ”no limit”. If one of the boundary is reached, the recording ends. The user can choose between three recording formats:

Original sample size For each USB port, the raw channel data (including preambles and counters) is saved in an individual file.

8 bits/sample For each frequency band, the samples are extracted and expanded to 8 bits/sample signed values and saved in an individual file.

Matlab The raw USB samples are converted as is done in the 8 bits/sample format but they are stored in individual MAT files.

The conversion between these formats can also be done in post-processing. The *post processing* tab, depicted in Figure 11(d), provides the flexibility to split and convert recorded raw data into the aforementioned formats. Furthermore the positions of transmission errors of the input file can be visualized.

Moreover, the user can select between three recording modes:

Direct recording In this mode, the data is written directly to the hard disk. The operating system has to take care of the write buffer. This mode has the advantage that an arbitrary amount of data can be recorded but the disadvantage that the recording performance depends on the writing speed of the hard disk (that is, errors may occur if the hard disk is too slow).

Ring buffered In this mode, a ring-buffer for simultaneous reading and writing is allocated in the RAM of the user’s PC. The raw data is first written to this fast RAM and then transfered to the hard disk using a file writer.

RAM buffered In this mode, the data is buffered in the RAM of the user’s PC and written on the hard disk when the recording is completed. The advantage with this mode is that the recording performance no longer limited by the hard disk’s writing speed. But the size of recording is limited by the available RAM.

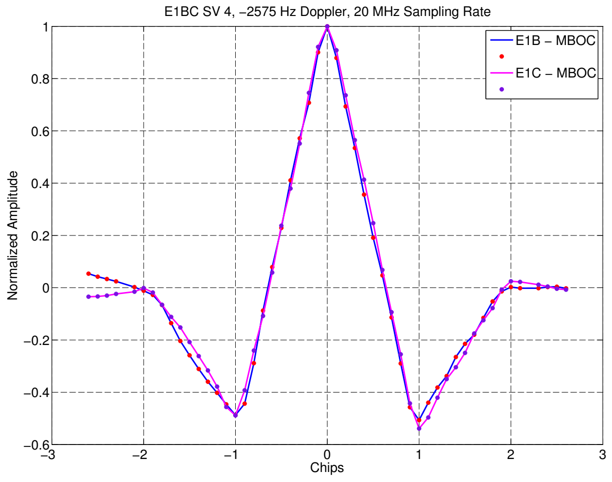
Finally, some status register settings of the RF modules can be read out, e.g. if the VCO/PLL is locked.

Flexiband GUI - Signal Analyzer

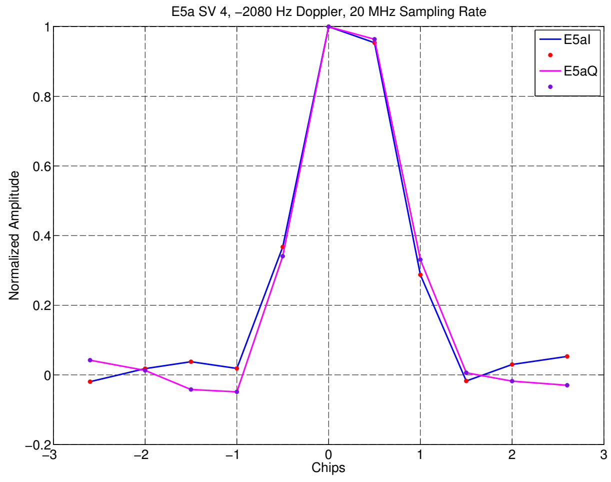
In the *analyse* tab, depicted in Figure 11(b), the statistical information provided in the *record* tab about the incoming raw signals is displayed again. The user can as well select the VGA settings of the RF modules. Depending on the chosen frequency band, the (complex) FFT spectrum of the raw samples is computed and visualised together with the histogram of the raw ADC values. Thus the front-end GUI software can be used as a simple real-time spectrum analyzer. This feature can not only be used to properly adjust the VGA gain but also to ensure that the antenna is working correctly and to instantly detect interference sources like continuous wave jammers.

Flexiband GUI - Scripting

The *script* tab, depicted in Figure 11(c), provides the user with the possibility to implement custom functions and tools



(a) Cross-correlation function of E1B/E1C MBOC with 20 MHz sampling rate



(b) Cross-correlation function of E5aI/Q with 20 MHz sampling rate

Figure 12. Flexiband Example

for data recording and analysis. E.g. the GUI automatic gain control (AGC) is realized using this scripting engine. The scripting editor is described in the Flexiband manual in detail and is part of the GUI. The scripting language is based on the ECMAScript standard and is similar to JavaScript.

A simple example for save the digital output data histogram to a file is given in the following:

```
function hist0Changed(hist,size) {
  buf=hist[0];
  for(i=1;i<size;i++)
    buf=buf+" "+hist[i];
  buf=buf+"\n"
  print(buf);
  appendToFile("test.txt",buf,buf.length);
}
statFilt0.histChanged.connect(hist0Changed);
```

EXAMPLES

Figure 13 shows one application setup of the Flexiband with a USB 2.0 interface unit connected to a notebook running the Flexiband GUI software and connected to the Fraunhofer IIS multi-band antenna [7].

As an example the Flexiband in configuration #6 (L1/E1, L2, and L5/E5a bands with 20 MSPS each) was used to record both Galileo E1 and E5a signals. More readily available default configurations are shown in Table 4.

Figure 12(a) and 12(b) depicts the cross correlation function (CCF) of the recorded signals and a local replica. The correlation length is always just one PRN-sequence long: 4 ms for Galileo E1B, E1C and 1 ms for Galileo E5a, respectively. The crosses on the CCF are the actual sample points of this snapshot acquisition. Figure 12(a) show the CCF of the recorded E1 signal with respectively the E1C (CBOC(6,1,1/11,'-')) and E1B (CBOC (6,1,1/11,'+')) local replica only.

Figure 12(b) show the CCF of the recorded E5a with its I and Q component processed individually. It can also be seen that the number of samples on the correlation peak is 10-times lower than on the E1 CCFs, which is in line with the fact that the chipping rates are 10-times faster on E5 than on E1 and also leads to a sharper correlation peak for E5 (note that the figures showing the E1 and E5 CCFs do not have the same scale on the x-axis).

CONCLUSION

Thanks to its new modular concept, the Flexiband front-end supports a future proof, upgradable, and flexible GNSS recording solution. Since it can also be exclusively powered through its USB connection the Flexiband is perfectly suited for mobile recording sessions. The possibility to change specific recording parameters on the fly, as well as its multi-antenna support, and the possibility to synchronize several Flexiband units with each other make the Flexiband a unique solution for numerous scientific and industrial projects.

The supplied Flexiband API and GUI give the user powerful tools to record IF GNSS samples and support the development and testing of real-time GNSS software receivers.

Future work will aim at integrating the Flexiband front-end into the GIPSIE environment [8] and at enabling digital-I/O devices to playback the recorded data later on. This will allow the exact replay of digital samples to reproduce deterministically the data collection conditions in a software receiver. It will also allow the replay of the recorded

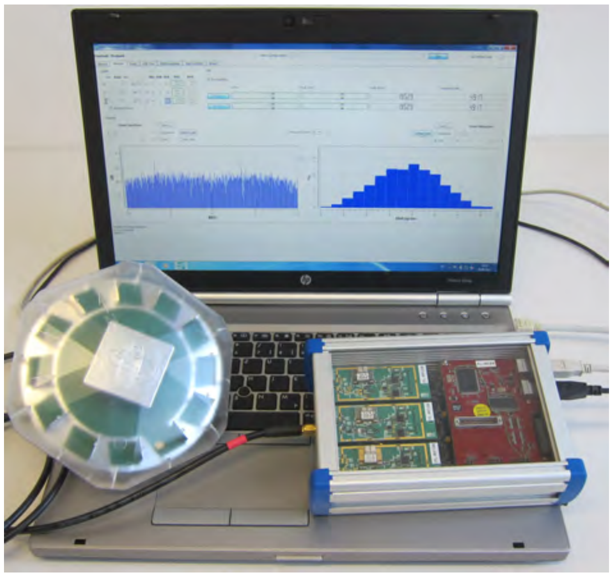


Figure 13. Flexiband setup with antenna and notebook running the front-end GUI software

digital data stream starting right at the desired time (e.g. at the beginning of a challenging episode) and enable a direct and fair comparison between different algorithms and/or parameter settings.

REFERENCES

[1] U. Haak, H.-G. Buesing, and P. Hecker, "A Multi-Purpose Software GNSS Receiver for Automotive Applications," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010), Portland, OR, September 2010, pp. 1869-1874*, 2010.

[2] A. Ayaz, R. Bauernfeind, J. Jang, I. Kraemer, D. Dtterbock, B. Ott, T. Pany, and B. Eissfeller, "Performance Evaluation of Single Antenna Interference Suppression Techniques on Galileo Signals using Real-time GNSS Software Receiver," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010), Portland, OR, September 2010, pp. 3330-3338*, 2010.

[3] H.-G. Busing, U. Haak, and P. Hecker, "Odometer-aided Instantaneous Signal Reacquisition for Automotive GNSS Receivers," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, OR, September 2011, pp. 954-959*, 2011.

[4] D. Doetterbock, C. Stoeber, F. Kneissl, and B. Eissfeller, "Tracking AltBOC with the ipexSR Software Receiver," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010), Portland, OR, September 2010, pp. 1896-1904*, 2010.

[5] J.-H. Won, B. Eissfeller, and T. Pany, "Implementation, Test and Validation of a Vector-Tracking-Loop with the ipex Software Receiver," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011), Portland, OR, September 2011, pp. 795-802*, 2011.

[6] C. Peacock, "USB in a Nutshell - Making Sense of the USB Standard, Second Release, 9th May 2002."

[7] A. Popugaev, R. Wansch, and S. Urquijo, "A Novel High Performance Antenna for GNSS Applications," in *Antennas and Propagation, 2007. EuCAP 2007. The Second European Conference on*, pp. 1 –5, Nov. 2007.

[8] A. Ruegamer, M. Overbeck, S. Koehler, G. Rohmer, P. Berglez, E. Wasle, and J. Seybold, "Digital GNSS Signal Recorder, Generator, and Simulator for Receiver Test, Qualification, and Certification," in *Proceedings of the 23th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2010, Portland, Oregon, September 20-24*, September 2010.