IOP Conf. Series: Materials Science and Engineering **152** (2016) 012033 doi:10.1088/1757-899X/152/1/012033

Low-cost approach for a software-defined radio based ground station receiver for CCSDS standard compliant S-band satellite communications

M A Boettcher[1*], B M Butt[2] and S Klinkner[1]

[1]University of Stuttgart, Institute of Space Systems, Germany
[2]University of Stuttgart, Institute of Telecommunications, Germany

[*]boettcher@irs.uni-stuttgart.de

Abstract. A major concern of a university satellite mission is to download the payload and the telemetry data from a satellite. While the ground station antennas are in general easy and with limited afford to procure, the receiving unit is most certainly not. The flexible and low-cost software-defined radio (SDR) transceiver "BladeRF" is used to receive the QPSK modulated and CCSDS compliant coded data of a satellite in the HAM radio S-band. The control software is based on the Open Source program GNU Radio, which also is used to perform CCSDS post processing of the binary bit stream. The test results show a good performance of the receiving system.

1. Introduction

A major concern of a university satellite mission is to download the payload and telemetry data from a satellite. While the ground station antennas are in general easy and with limited afford to procure, the receiving unit is most certainly not. As experience teaches, market prices of several ten-thousand USD should be considered for a commercial transceiver system. Nowadays, flexible and low-cost software-defined radio (SDR) transceivers are available and their performance is quite convincing. In particular, since most modern SDRs use the concept of direct conversion, there is no need of up/down converting from/to an intermediate frequency, which greatly supports a cost efficient solution.

Within the university project "Flying Laptop", a small satellite has been developed that uses a 2.4 GHz HAM-Radio transmitter to downlink the payload data (data downlink system, DDS). In order to receive the CCSDS-standard [1] coded and QPSK modulated data, the SDR "BladeRF" from NUAND has been chosen [2]. The main reasons for this choice are the low market price of a few hundred USD and a requirement of minimum of 20 MS/s in order to handle the satellite's payload stream at a data rate of 10 Mbit/s [3]. The BladeRF is a fully USB 3.0 bus powered device and it can be operated autonomously using a FPGA for signal processing. It offers a RF frequency range from 300 MHz up to 3.8 GHz and an independent RX/TX with 12-bit 40MSPS quadrature sampling. With its fully integrated RF transceiver LMS6002D, the SDR is capable of achieving full-duplex 28MHz channels. The signal post processing is done by an on-board 200MHz ARM9 with 512KB embedded SRAM and an on-board 40KLE or Altera Cyclone 4 E FPGA [2]. Other options, like the HackRF One by Great Scott Gadgets [4], would have been a cheaper solution. However, with a sample size of 8 bit only, a limited performance might have to be expected. Another suitable candidate is the USRP B200 by Ettus Research [5], which is offering similar features like BladeRF but with doubled bandwidth capability

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

and a roughly doubled price. Following the low-cost approach, this SDR has been considered as a back-up solution.

The implementation has been carried out according to the specification of DDS protocol, which can be seen in Figure 1. As a control program, open source software GNU Radio is used, including GNU Radio Companion [6] for a graphical user interface. All the functions and blocks are implemented in C++ and Python.

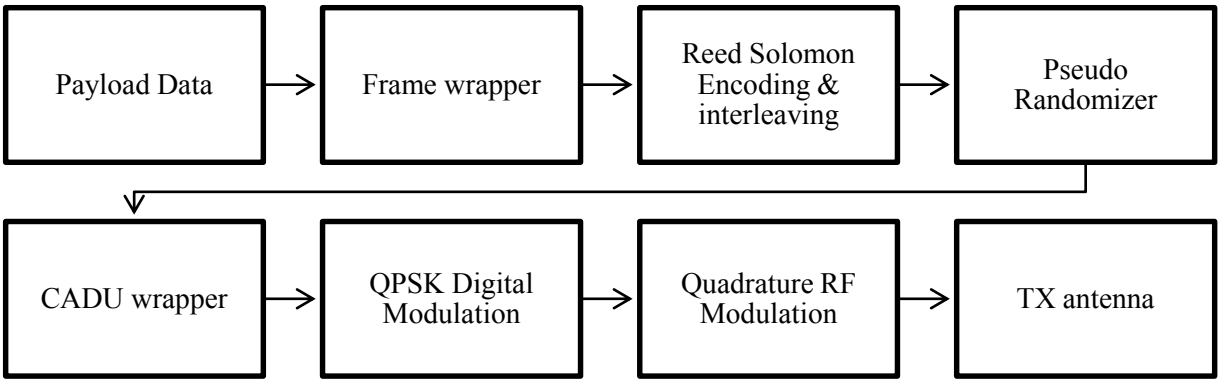


Figure 1. Signal processing chain of the payload transmitter DDS

2. Signal processing software chain

The GNU Radio Companion flow diagram is shown in Figure 2. Following this flow graph, all the steps for receiving and decoding of the DDS signal, given as an overview in Figure 3, can be taken sequentially. BladeRF offers a hardware interface via osmocom, by which the RF frequency and the filter settings can be chosen. Depending on the specific signal, the demodulation settings have to be inserted wisely. As can be seen in Figure 4, besides the wanted test signal from a signal generator, the FFT plot shows a large spike at 0 Hz. This DC offset called effect is a result from direct conversion receiving concept of the SDR. Following the solution recommended by the manufacturer NUAND [7], the DC offset can be avoided by the shifting frequency technique. As approximately 7 MHz bandwidth is occupied by the DDS signal and the BladeRF offers up to 20 MHz, the osmocom carrier frequency can be tuned at a frequency a slightly away from the original carrier frequency. In that case, as can be seen in Figure 4, the DC offset peak does not affect the input signal.

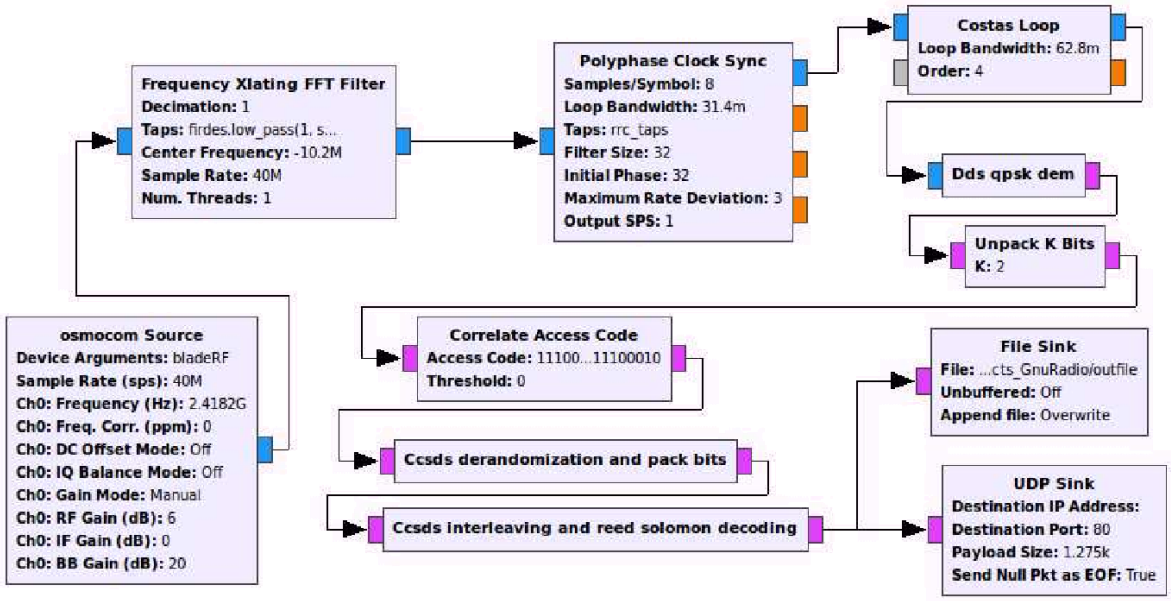


Figure 2. GRC flow graph for DDS receiver

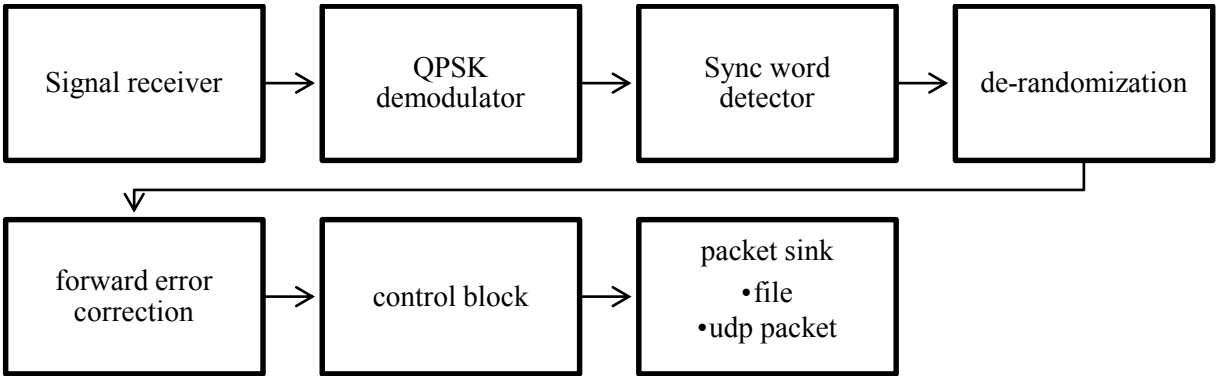


Figure 3. DDS receiver section

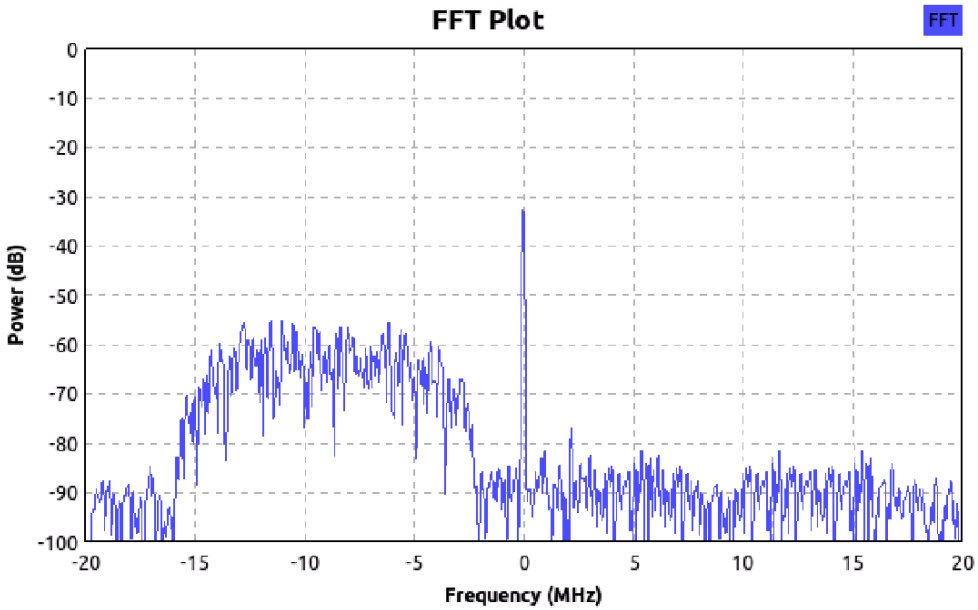


Figure 4. FFT plot of the received test signal (detuned by -10 MHz) including DC offset

In order to get only the wanted input signal, initial detuning of the SDR needs to be compensated. In GRC, a block named Frequency Xlating FFT has all the capabilities to execute two things at the same time. First, it shifts the center frequency of the signal according to the initial detune parameter, then it applies a low pass filter that helps to discard unwanted portions of the signal. Finally, the adjusted signal is handed over to the demodulation stage, consisting of clock synchronization, costas loop and QPSK demodulation. Moreover, the signal has to pass through the “Polyphase Clock Sync”, which recovers the timing by trying to find the best time to sample the incoming signal. This block is able to account for the SRRC filter settings of the DDS signal (excess bandwidth of 0.4) [8]. In order to compensate for possible phase and frequency offset, the “Costas Loop” block is used. The resulting signal is then handed over to the “DDS qpsk dem” block, where the input signal is being demodulated following the QPSK scheme. Input for this block are complex numbers from the former stage and the block calculates the distance from the expected position on consolation diagram. According to the CCSDS standard, Gray coding is used and this block generates two binary bits for one point such that the data rate will double from this point. Resulting constellation diagram of the FLP DDS transmitter is shifted by 45 degrees and can be seen in Figure 5. The binary output stream is handed over to the next stage, which is the channel decoding.

With the binary data string from QPSK demodulator section, valid DDS CADU packets (CCSDS) need to be detected by searching for the "Attached Synchronization Marker" (ASM). Each packet starts with a sync pattern and after that, packets of a fixed length of 100 bytes are sent. Furthermore,

the bits need to be pooled as bytes of data. Due to the fact that these bytes are scrambled before the transmission by the satellite, they need to be de-randomized for further processing. As can be seen in Figure 6, forward error correction can be implemented on the resulting bytes' string. First the data packet is de-interleaved and then the Reed Solomon (RS) decoding algorithm is applied to recover the error bytes in case of corrupted data. After recovering the original data blocks, payload information data is recovered according to the DDS protocol structure.

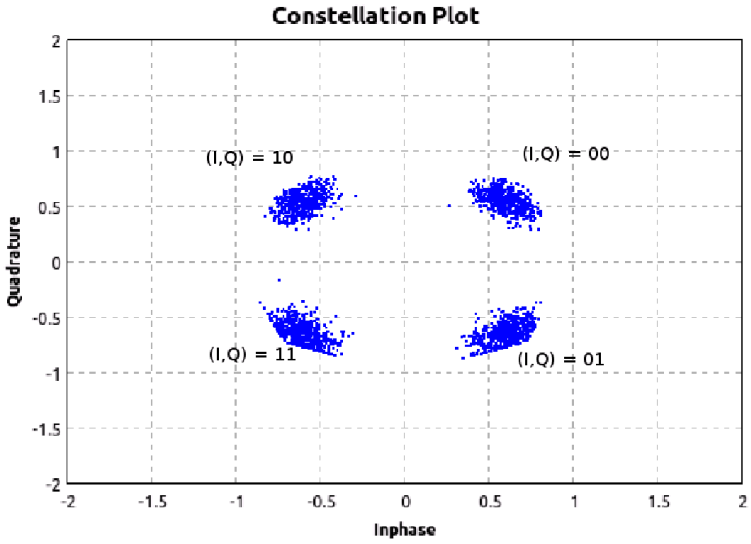


Figure 5. Constellation plot of the demodulated signal

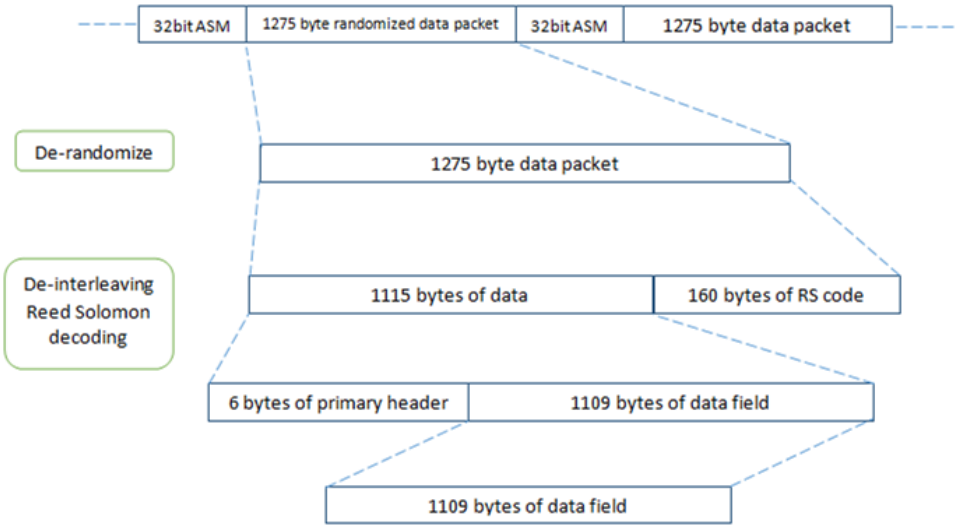


Figure 6. Example CADU packet according to CCSDS standard taken out of the binary string

2.1. Synchronization marker detection

The CCSDS Blue Book [9] recommends a 32-bit ASM to be used with the Reed Solomon and basic convolutional coding. The recommended 32-bit ASM is a hexadecimal number as 0x1ACFFC1D and the position varies as follows:

- If only Reed Solomon code is used including a bit randomizer implemented as optional, the ASM is attached at the start of each generated (randomized) code-word.
- If convolutional coding is used, ASM is attached to the start of data frame and convolutional encoding.

In the DDS protocol structure, at first the payload data are Reed Solomon coded with interleaving and an enabled pseudo bit randomizer implementation, having the ASM attached at the start of every DDS packet. The ASM is never randomized.

2.1.1. Correlation and threshold. Data frames or packets can be identified by searching for the ASM. The correlation and threshold techniques can be used to perform ASM search from the bit string in order to identify a start of packet. Here, the sync word ASM (0x1ACFFC1D) will be correlated with incoming bit stream. As bit errors may occur, which can also corrupt the ASM, a certain threshold is necessary to compensate few error bits in this sync sequence. As shown in the Figure 7, the ASM sync word is been correlated with the input stream. It generates a high peak when it is correlated with same bits that are above the threshold. Hereby, it can be assumed that an ASM is being detected in the bit stream and the start of a packet is present.

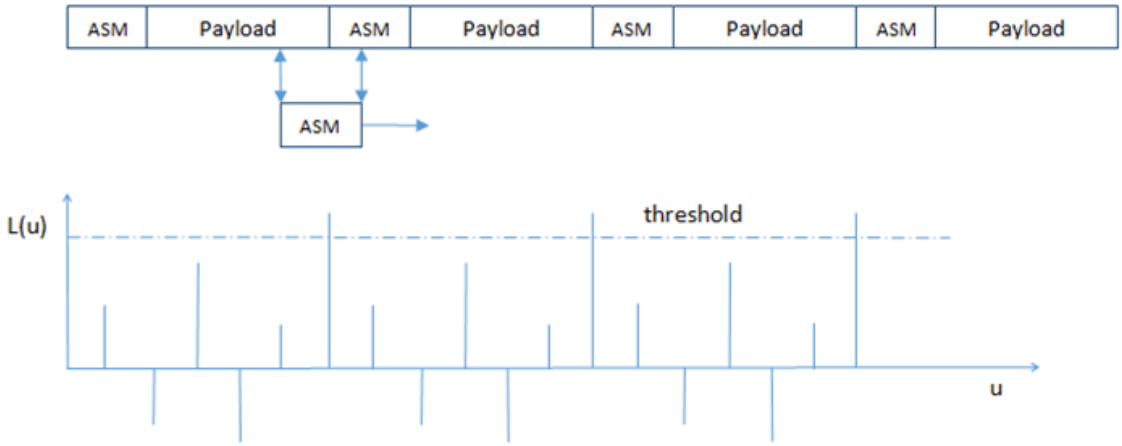


Figure 7. Correlation of ASM with the received data

2.1.2. Sync word packet detection. In GNU-radio companion flow graph, ASM detection is performed by a block named 'Correlate Access Code'. Input arguments for the block access code and threshold are needed. For access code, sync bits ASM 0x1ACFFC1D is used and for the threshold, the number of relaxation bits which will define the threshold to compare with the correlation result is inserted. Input samples for the block Correlate Access Code is the bit stream from the QPSK demodulator. It continuously correlates the input bit stream with the local argument ASM bits and compares result with the threshold. In this case, it compares how many bits are mismatched. If number of mismatched bits is less than the input argument threshold, the presence of ASM sync word in the bit stream is indicated. The immediate next bit will be marked as the start of a frame. This marker will be used by next block in GRC flow graph. Start of frame is required to perform de-randomization and further processing.

2.1.3. Pseudo randomizer. With valid detection of the start of the frame and fixed DDS packet length to 1279 bytes, 4 bytes out of 1279 are used for ASM and the remaining 1275 are the actual payload data of interest. In order to process those pseudo randomized bytes, a de-randomization is needed before further processing. In most communication systems, data is being randomized to distribute the density of 0 and 1 equally across the data string. This procedure is required especially for the receiver phased lock loop (PLL) synchronization that ensures sufficient transitions between 0s and 1s in the data bit stream. If long string of 0 or 1 is present in the data stream, the symbol recovery loop will face a difficulty to lock. CCSDS recommends a pseudo randomizer with the generator polynomial in Equation 1 [4], which is implemented in DDS transmitter.

$$h(x) = x8 + x7 + x5 + x3 + 1 \quad (1)$$

The period of this random sequence is $2^{(m-1)}$ where m is the degree of the generator polynomial. The sequence repeats itself after$2^{(m-1)}$ and therefore, the randomizer has a length (with m is equal to 8) of 255. At the start of each transfer frame, all the liner feedback shift registers are recommended in the CCSDS Blue Book [34] to be predefined as 1s. This is shown in Figure 8.

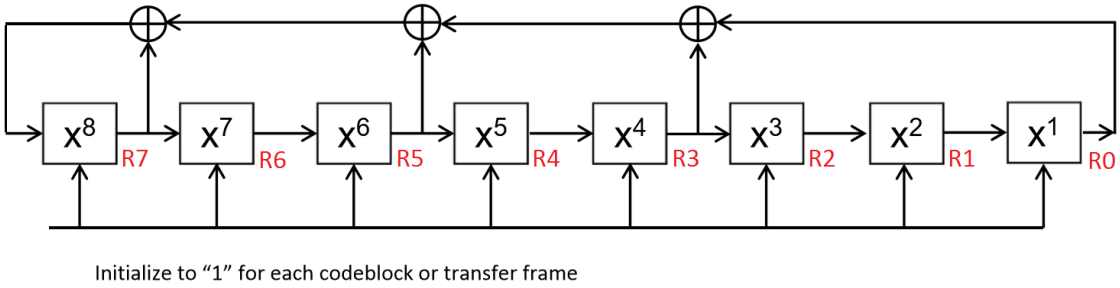


Figure 8. CCSDS driven pseudo randomizer block diagram

The output bits of the shift registers are combined to form the bytes R7, R6, R5, R4, R3, R2, R1, R0. This is the output for the pseudo random sequence bytes, which is implemented in DDS protocol. On every clock cycle, an XOR is executed between the corresponding bits from the output of the shift registers and its content is right shifted. XOR is done between the data byte and the pseudo random sequence bytes, and the output is the de-randomized transfer frames. Random sequence generation bytes are independent and constant, and they are especially independent from the input data stream. The bytes remain constant and repeat again after 255 bytes. So it is more feasible to construct a lookup table, which will optimize the speed, because bit shifting and the polynomial equation solving will not be required in real time.

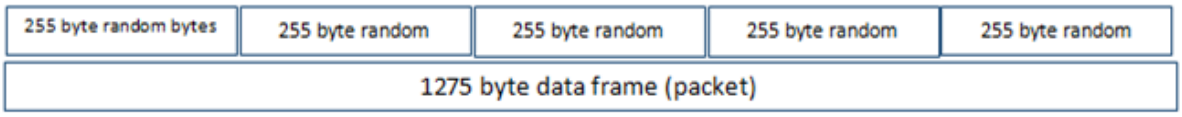


Figure 9. 255 bytes of random pattern repeats five time for one data frame

The output delivered by the Correlate Access Code is in the form of bits. It is more suitable to pack the bits into bytes by performing a packaging of bits after detection of the marker. After that, the bytes are ready for the de-randomized process. During the de-randomizing of a complete DDS frame, 255 random bytes sequence will repeat for five times within every frame. In GNU-radio companion flow graph, de-randomization is performed by a block called 'CCSD de-randomization'. The packaging of the bits is implemented as a dedicated block for the specific DDS packets. In this block, the detection marker from the previous block is needed. This marker indicates the start of the DDS frame. After detecting the frame's beginning, the bits needs to be packed to bytes and the DDS frame has fixed predefined length of 1275 bytes. These bytes are then XOR processed with pseudo random sequence bytes stored in the lookup table. In the end, a marker is placed at the beginning of the de-randomized frame to enable the next block to easily detect the start of this frame.

2.2. Forward error correction (Reed Solomon decoding)

According to the DDS protocol, the transmitter has implemented a Reed Solomon (RS) encoding with interleaving for forward error correction. Forward error correcting (Reed Solomon) is a technique of digital signal processing and it enhances the reliability of received data on the receiver side. In this technique, redundant data (overhead) is introduced and transmitted along with the original data. In the Blue Book [9], two systematic Reed Solomon codes (255,223) and (255,239) are recommended. The first one has 32 symbols of overhead and is capable for error correcting performance of 16 symbols. In

the DDS protocol, the first one (255,223) is implemented by the transmitter and therefore, the receiver has a capability of correcting 16 symbols (bytes) per Reed Solomon decoding per frame.

2.2.1. De-interleaving. According to the DDS protocol, the 1275 bytes per frame contain five Reed Solomon encoded chunks. Each chunk size is 255 bytes. As RS(255,223)-coding is implemented in the DDS transmitter, 223 bytes are actual data and the rest of 32 bytes are overheads for error correction. In the DDS frame, these five RS chunks are merged with each other with a technique called interleaving. The objective of interleaving is to make the forward error correction more robust against bust error. For a whole DDS frame, up to 80 errors can be corrected because of 5 RS encoded chunks.

2.2.2. Reed Solomon error correction realization. In the GNU-radio companion flow graph, de-interleaving and Reed Solomon decoding is performed by a block named 'CCSD interleaving and Reed Solomon decoding' that is implemented as a dedicated block for specific DDS packet. It is essential for this block to use the detection maker from previous block, which indicates the start of the de-randomized DDS frame. Knowing the frame's beginning, the bytes can be de-interleaved and five different RS encoded chunks can be created. These chunks are then further fed separately to the Reed Solomon decode algorithm [10] and processed. At the end, these data chunks are organized in such a way that all the data part (first 223 bytes) of all five of the chunks placed together in the beginning, which leads to 1115 bytes of data and the RC codes for each chunk (remaining 160 bytes) placed together. The number of error corrected for each chunk is also be placed along with the data, as this information might be helpful as a signal quality indicator.

2.3. Control block and file sink
From previous stage, there are 1275 RS corrected bytes, including 1115 actual data bytes and 160RC overheads bytes. Depending on the different user needs, either all the bytes (including RS overheads) can be send to the next stage, e.g. for debugging purpose, or the overhead can be discarded and only the payload data is forwarded. The marker is detected first to ensure the start of the frame, which was marked in the previous stage. Then the data is separated or manipulated as required by using different blocks. In the last stage, the payload data stream can be transferred to a file or transmitted via UDP.
File sink requires the filename as input parameter where the samples will be stored in real time. In the case of UDP sink, the destination port, destination IP address and number of bytes per packet of UDP that is also called as payload size of UDP packet are required.

3. Test results

The tests have shown a very convincing capability of the SDR receiver. At first, the DDS channel decoding section was tested separately, showing that the whole software processing chain including the DDS protocol was working as expected. The assembly was able to decode the data package from the DDS protocol, fed by an input file. It was also successfully able to save all the data in a file and to send the packets through the network via the "UDP sink". By adding artificial bursts and bit errors of different numbers to this input file, the forward error correction section could also be successfully tested. As expected, up to 16 bit errors within one block could be detected and corrected.

In the next step, a complete system was tested to verify its functionality including RF processing chain. Therefore, the BladeRF has been connected to the FLP satellite within the clean room facilities at Institute of Space Systems in Stuttgart, Germany. In order to actually measure the radio interface, the SDR was not directly connected to the satellite's DDS transmitter. Instead, an EGSE antenna cap was attached to the FLP payload horn antenna and connected to the BladeRF. Besides, a computer connected thought an USB 3.0 interface has been used to run the GRC software.

Figure 10 shows the frequency spectrum, which was recovered from the output of the first block in GRC, the osmocom source. It can be seen that the receiver is able to receive and process the data transmitted by the satellite at a data rate of 10 Mbit/s with a corresponding bandwidth of 7 MHz. The carrier frequency is to be found around 2.408 GHz, the nominal DDS center frequency.

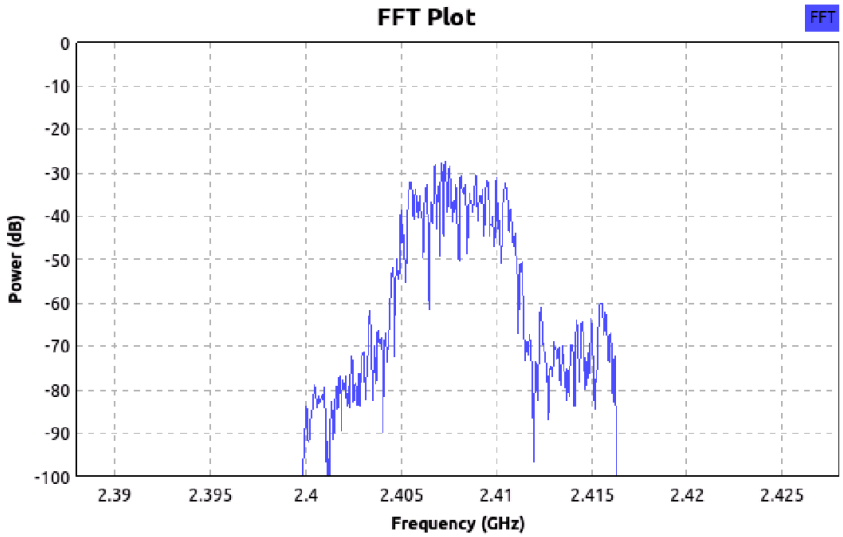


Figure 10. Received QPSK signal from satellite at 10 Mbps (7 MHz of occupied bandwidth)

During the test, most of DDS packets were received easily. Meanwhile, some of the packets were received with bit and burst errors. While most of those packets could be recovered due to the Reed Solomon coding, some contained more than 16 bit errors and were corrupted. Main reason for this behavior is to be found in actual measurement setup. Because the satellite is already fully assembled, it is not possible to get a direct access to the output port of the DDS transmitter. When using the EGSE antenna cap as a receiving antenna within the horn's near field, the SNR becomes smaller and bearing in mind that the center frequency is around 2.4 GHz, also the local Wi-Fi is accountable for some burst errors. Besides that, there were also missing packets because of instabilities in the DDS demodulation section. Displacements in the constellation plot point to the assumption that the polyphase filter bank clock recovery and SRRC filter are not working as reliable as expected, particularly in terms of the signal phase not being recovered continuously without errors. Investigation on this matter is still ongoing to improve the algorithm. Furthermore, it is intended to specify the receiver more in detail, especially to determine the sensitivity and Eb/N0 vs. bit error probability correlation.

4. Conclusion

In this work, low-cost approach for a software-defined radio based ground station receiver for CCSDS standard compliant S-band satellite communications has been shown. The SDR bladeRF by NUAND has been chosen to receive the data, which is QPSK modulated, SRRC filtered and can receive data at a rate of 10 Mbit/s and occupying a bandwidth of 7 MHz. By using this hardware, the cost could be decreased to about 10-20 % of the cost of a commercial satellite receiver. In order to configure the SDR and further process the data according to the CCSDS standard, open source software GNU Radio has been used, including GNU Radio Companion for a graphical user interface. Meanwhile, all the functions and blocks are implemented in C++ and Python. This software chain is able to demodulate the QPSK data according to the CCSDS given constellation diagram and wrap the bit stream into CCSDS packets by searching for the given synchronization marker and descrambling. Reed Solomon decoding and interleaving for forward error correction is then applied and finally, valid data packets sink via UDP over the network or into a file. The CCSDS software decoding chain has been tested successfully. During the test of data processing chain together with the hardware, the system showed very convincing results. Although most packets transmitted by the satellite were received correctly or could be recovered, some packets were found missing. Main reasons are suspected within the clock and phase recover stages because the constellation plot seemed to show some instabilities. Further investigations are still ongoing to correct this stage and to further improve the reliability of this SDR based ground station receiver. Besides, the RF parameters also need to be specified more in detail.

References

[1] *Consultative Committee for Space Data Systems* [Accessed online August 2016]
[2] *Blade RF Software defined Radio home page* [Accessed online August 2016]
[3] Thibaut C 2012 *DDS Transmitter Technical Notes and Specifications* (Stuttgart: IRS University of Stuttgart)
[4] *HackRF Software defined Radio* [Accessed online June 2016]
[5] *USRP Software defined Radio* [Accessed online June 2016]
[6] Blossom E et al. 2016 *Welcome to GNU Radio* [Accessed online June 2016]
[7] *DC offset and IQ Imbalance Correction for BladeRF* [Accessed online June 2016]
[8] Thibaut C and Boettcher M 2015 *FLP Data Downlink System Protocol Technical Note* (Stuttgart: IRS University of Stuttgart)
[9] Consultative Committee for Space Data Systems 2015 *TM Synchronization and Channel Coding. Recommended Standard CCSDS 132.0-B-2*
[10] Karn P 2016 *DSP and FEC Library* [Accessed online June 2016]