An ARISSat-1 Overview

by Tony Monteiro, AA2TX, aa2tx@comcast.net
and Gould Smith, WA4SXM, wa4sxm@amsat.org

Introduction

On Monday October 4, 2010, Lou McFadin, W5DID, drove his Chevy Suburban to our Orlando Satellite Construction Facility (*also known as K4RSV George's garage*) picked up the completed ARISSat-1 satellite, and drove it to Essential Air Services where it was packed into a wooden crate. This humble trip marked the beginning of its long journey into space.

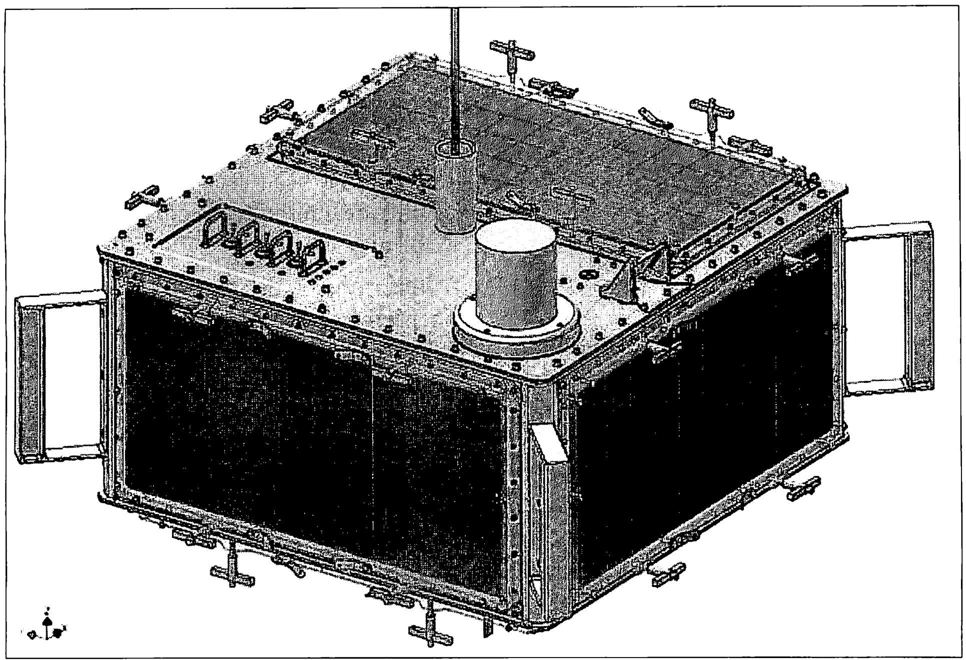


Figure 1: ARISSat-1 external view.

The satellite was shipped from Orlando to the NASA Johnson Space Center in Houston, TX. From there, it will travel to Moscow where the Kursk University science experiment will be added and final tests will be performed. And then it will be off to the Baikonur Cosmodrome where it is to be launched on a *Progress* spacecraft to the International Space Station (ISS.) Once on board, the flight battery will be installed and finally, it will be taken through the airlock by a pair of cosmonauts and released into earth orbit.

The completion of this satellite is the culmination of four years of effort and represents the work of over 50 AMSAT volunteers working in conjunction with NASA, RSC-Energia (the Russian space agency) and ARISS-International. This overview is the first in a series of articles planned for *The AMSAT Journal* that will describe the ARISSat-1 systems and capabilities.

ARISS and SuitSat

The Amateur Radio on the International Space Station (ARISS) program is an international effort staffed by volunteers. It is intended to promote space education and inspire an interest in science, technology, engineering and math (STEM) through Amateur Radio activities on the ISS.

The first ARISS space education satellite was SuitSat; deployed on February 3, 2006. SuitSat caused an international sensation and was perhaps the most unusual satellite ever devised. It consisted of a radio transmitter mounted to the helmet of an old (empty) Russian *Orlan* space suit. Pre-recorded voice greetings from students around the world were to be played through the transmitter on the 2 meter ham band. Unfortunately, the transmitted signal from SuitSat proved to be very weak and only a few sophisticated ground stations were able to receive it. After about two weeks in orbit, the batteries failed ending its operation.

In November of 2006, a new version of SuitSat was initiated. The intent was to again use an old Russian space suit as the satellite spaceframe but with significant technical improvements. The new satellite, dubbed SuitSat-2, was to be developed using modular, re-usable sub-systems and would provide a platform for flying student provided science experiments. It would also provide the opportunity to fly an improved radio communications system employing an experimental software defined transponder (SDX). To provide a longer life, solar panels and a power system would be used to re-charge its battery.

Alas, in July of 2009 with SuitSat-2 development well under way, the engineering team hit a major road block. Due to the need for more *space* on the ISS, the surplus Orlan space suit had to be discarded. RSC-Energia informed ARISS that they were still willing to provide the upmass launch on a Progress rocket to the ISS and the EVA (extra-vehicular activity *a/k/a/ a spacewalk*) to deploy it, but the satellite would need its own spaceframe. With this new development, the engineering team began in earnest to design and construct a new spaceframe along with re-working the previously designed modules and subsystems to go with it. The new satellite was named ARISSat-1.

The ARISSat-1 System

The new ARISSat-1 satellite consists of several subsystems and modules mounted to an aluminum spaceframe as shown in the drawings of Figures 1 and 2. The subsystems include six solar panels, a maximum power point tracker (MPPT) module, a battery, a control panel, four slow-scan television cameras, the Kursk science experiment, a 2 m whip antenna, a 70 cm whip antenna, an RF module and an Internal Housekeeping Unit (IHU.) These modules are interconnected as shown in Figure 3. Modules on the left side of the drawing are on the outside of the spaceframe structure while those on the right are inside.

The six solar panels were donated to the project from NASA. They were left over after the Small Explorer (SMEX) satellite program ended. Each panel measures 19-inches by 10.5-inches and consist of 50 cells. In full sunlight, each panel can produce about 50 volts open-circuit and over 19 watts of electrical power. A panel is mounted on each of the six surfaces of the spaceframe.

Each solar panel is connected to a maximum power point tracker circuit inside the MPPT module. These circuits are DC to DC power converters that are designed to draw the maximum power available from each solar panel to drive the satellite's +28V main

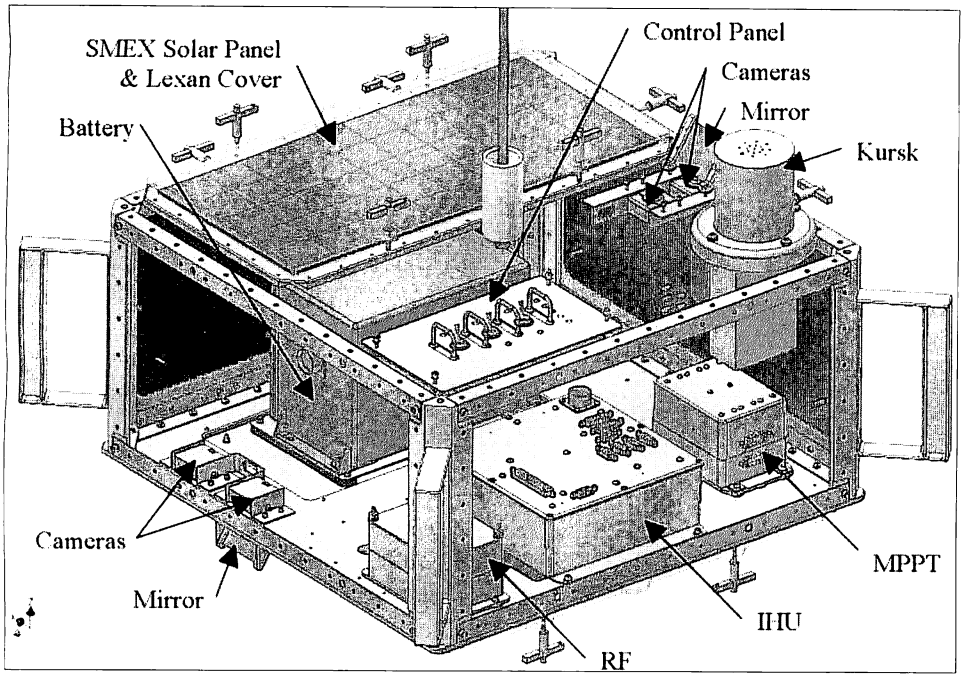


Figure 2: ARISSat-1 cutaway view.

power bus. The main power bus is directly connected to the battery. To prevent over-charging the battery, the maximum voltage from the MPPTs is limited to approximately 36 volts.

The battery was donated to the project by RSC-Energia. This is a type 825M3 and is the exact same type used to power the Russian Orlan space suit. It internally consists of eighteen, rechargeable silver-zinc (AgZn) cells and is specified for 14 ampere-hours at 28 volts. In sunlight, the solar panels and MPPT circuits will run the satellite and charge the battery and in eclipse, the battery provides the power to run the satellite.

To meet the ISS safety requirements, the control panel keeps the battery and solar panels from powering the satellite until it is ready for deployment. The control panel, shown in Figure 4, provides an interface that a cosmonaut can comfortably operate while wearing a space suit. It has three switches separated by finger guards so that only one switch at a time can be activated. All three switches must be turned on to power-up the satellite. A set of LEDs provide the satellite operational status. After all three switches are turned on, a safety timer circuit prevents any RF transmissions for 15 minutes. This is to prevent the RF transmitter on ARISSat-1 from causing a hazard by interfering with the space suit electronics.

While ARISSat-1 cannot transmit a signal for 15 minutes, it does actually power up and take several snapshots with its SSTV cameras soon after the switches are turned on. It saves these photos in its internal memory for transmission throughout its first day in orbit and it is hoped that it can catch a glimpse of the cosmonauts or the ISS as it is being deployed. There are a total of four cameras on ARISSat-1 with mirrors to set the field of view of two of the cameras. They point up, down, and to opposite sides of the satellite. A new set of snapshots are taken about every two minutes and saved in memory for transmission with very dark or blank images discarded. There are also several pre-recorded images that are sent when the satellite is in eclipse (i.e. darkness). One of these images showing the call sign (RS01S) can been seen in Figure 5. The pictures are transmitted in color using Robot 36 format.

To the right of the control panel on the top plate, as shown in Figure 2, is the Kursk science experiment. This experiment was developed by students at the Kursk State University in Russia and is intended to measure the vacuum of space. The experiment is started 30 minutes after deployment and will run once each day for a complete orbit. It should be possible to measure the differences in vacuum pressure as the satellite slowly descends into the upper part of the earth's atmosphere. A photo of the Kursk experiment can be seen in Figure 6.

Also mounted on the top plate, in the center, is the 2 m whip antenna. The base of this antenna can be seen in the drawing of Figure 1. ARISSat-1 has a 70 cm whip antenna as well that is mounted on the bottom plate though it is not visible in the drawing. These antennas are connected to the RF module.

The RF module has a 2 m communications transmitter and produces a total of 500 milliwatts of power. The input to the transmitter is a 10.7 MHz intermediate frequency (IF) signal that is generated in the IHU. The RF module also has a 70 cm band communications receiver and its output is a 10.7 MHz IF signal that is fed to the IHU. Additionally, the RF module includes a 70 cm command receiver which provides demodulated audio to the IHU. This receiver is only intended for ground control of the satellite.

The internal Housekeeping Unit (IHU) is perhaps the most complex module in the system and includes several internal circuit cards with a total of five microprocessors. The IHU module diagram is shown in Figure

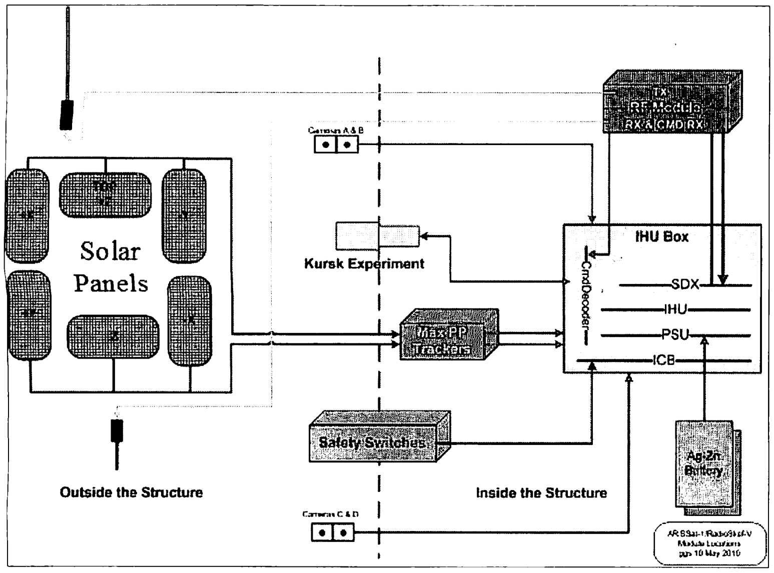


Figure 3: ARISSat-1 system diagram.

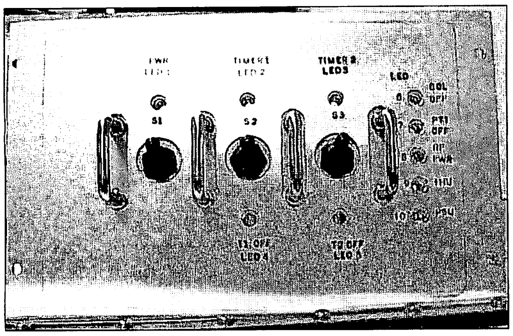


Figure 4: Control panel.

7. The Power Supply Unit (PSU) shown at the top of the drawing converts the +28 volts from the battery and MPPT module to the +5, +8, and +12 volts needed by the other IHU circuits. The main IHU processor is a PIC32MX as shown in the center of the diagram. This processor provides the overall command, control and telemetry for the satellite. It also does the video processing for the SSTV cameras and controls the Kursk experiment operation. A second PIC32MX processor performs the software defined transponder (SDX) functions. All of the radio signals transmitted from ARISSat-1 are generated in software.

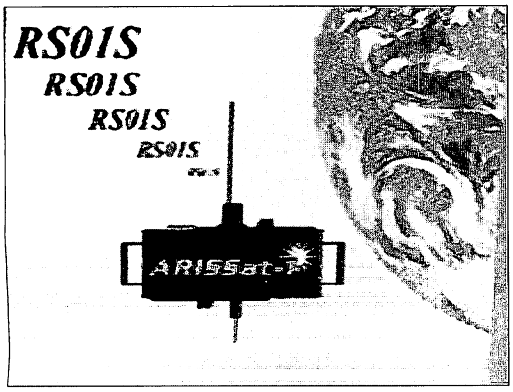


Figure 5: SSTV call sign.

These radio signals and their frequencies are shown in Figure 8. The 70 cm band is used for the uplink of a mode U/V (a/k/a mode B) transponder. The 2 m down-link signals include the transponder output, a digital BPSK Telemetry beacon, a CW beacon and FM audio. The transponder is a linear, inverting type and provides 16 kHz of bandwidth. It is very sensitive and should be workable with QRP equipment and omni-directional antennas. The digital BPSK beacon provides satellite telemetry and the Kursk experiment data. The beacon can operate in either legacy BPSK-400 mode or a new BPSK-1000 mode that was specifically developed for ARISSat-1. The new mode includes forward error correction and interleaving and should provide error-free reception using even simple antennas. A companion sound card based telemetry decoding program is currently being developed for PC and Mac computers. The CW beacon has the satellite call sign (RS01S,) telemetry data and the Amateur Radio call signs of instrumental ARISS and ARISSat-1 program contributors. There are over 200 call signs - see if you can collect them all! The FM audio signal has the satellite call sign, spoken telemetry, the SSTV image signals and voice greetings collected from children throughout the world. There are 24 different greetings in 15 languages and there is a special recording from Yuri Gagarin.

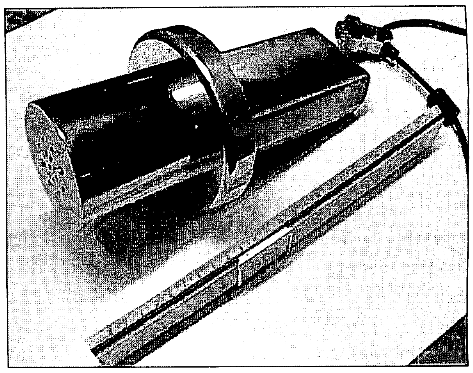


Figure 6: Photo of Kursk experiment.

Launch

ARISSat-1 is scheduled for deployment from the ISS on a *manned* spaceflight mission so it must meet very stringent safety requirements that go well beyond that normally required of a satellite. A NASA safety panel, with participation from RSC-Energia and AMSAT, has already conducted several reviews and has provided guidance on the requirements needed to pass the final review. The final safety review requires the results of testing that is to be done by RSC-Energia and so must be conducted *after* the satellite has been shipped to Moscow.

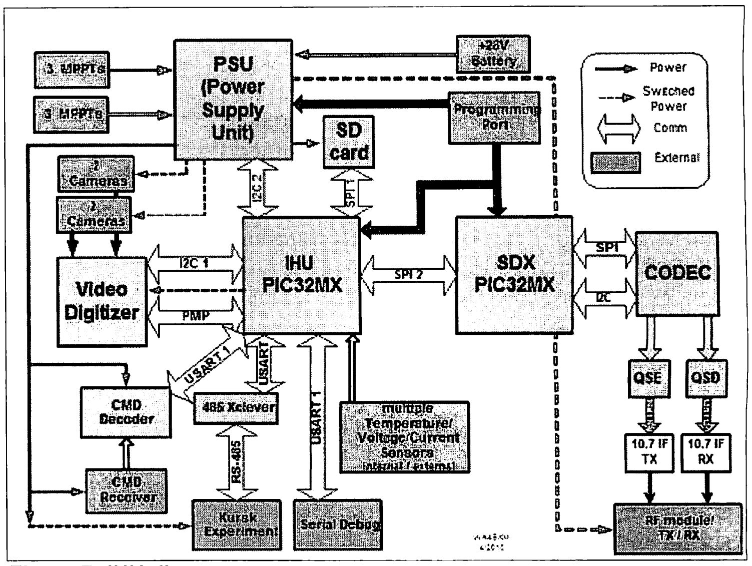


Figure 7: IHU diagram.

ARISSat-1 is currently planned for launch from the Baikonur Cosmodrome in Russia on Progress #41 in January 2011. It is scheduled for deployment from the ISS in February 2011 and should remain in orbit for about one year.

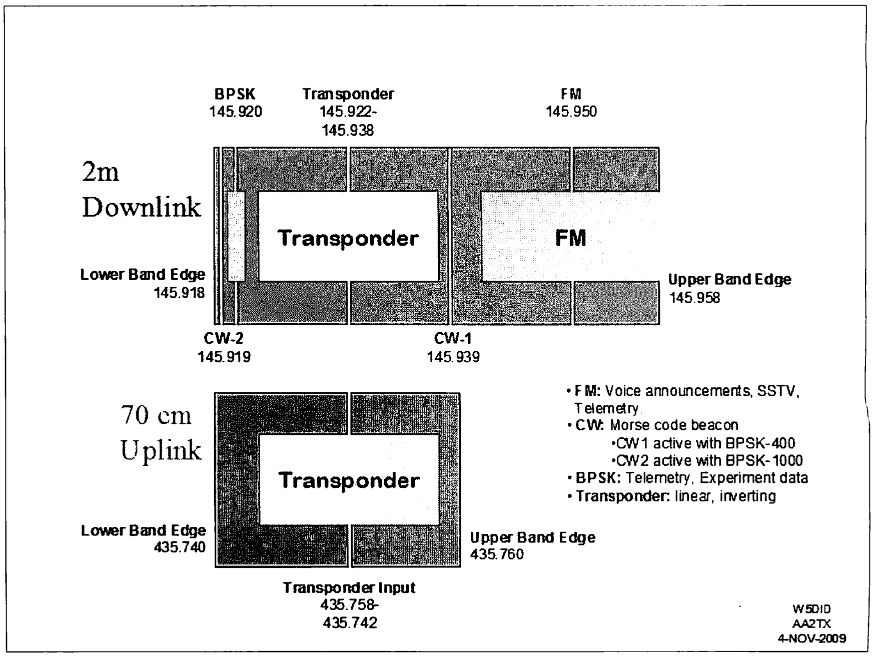


Figure 8: RF band plan.

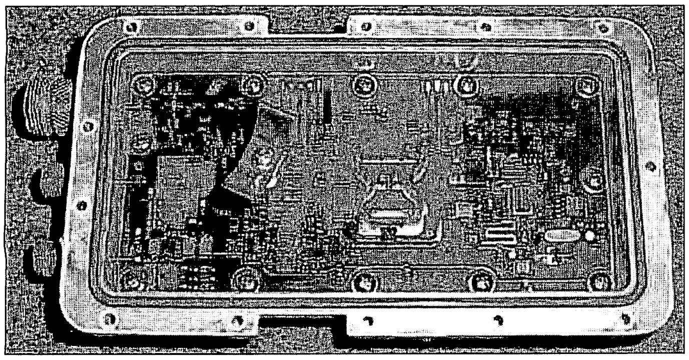


Photo B - Model 13001 Downconverter

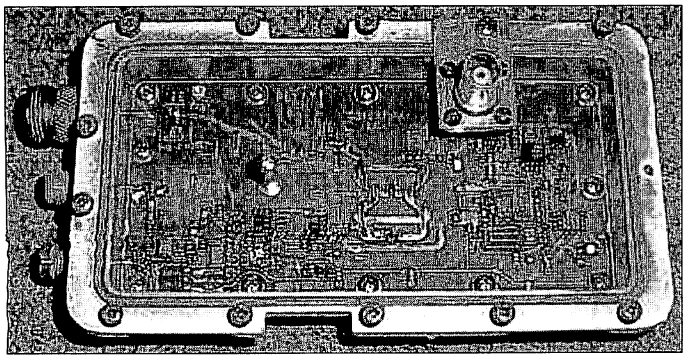


Photo C - Model 130002 Downconverter with Probe

enclosures that would be large enough to contain a PLL oscillator along with a 12V three-terminal voltage regulator. The internal cavity measures 2-7/8" x 3-7/16" and is 1" deep unless the internal 1/8" high mounting pads are used. I used the existing circuit board to size and layout the mounting hole pattern for drilling a 1/16" thick aluminum shelf plate. I cut the local oscillator section from the main printed circuit board of a model 130001 downconverter using a 4" hobby table saw. Adding an RF output cable to the printed circuit board was tricky. I used two U-shaped wires to ground the coax braid and physically restrain the cable. I was able to mount the local oscillator printed circuit board directly to the shelf plate with seven 4-40 screws, lock washers, and hex nuts using the original holes in the circuit board. I drilled a 1/2" diameter hole through the shelf plate under the crystal and a 3/8" diameter hole under the coax connection to prevent shorts and allow the circuit board to sit flush on the shelf. Model 130002 downconverters will require a larger cutout for the crystal under the circuit board. Unused circuit traces were cut and peeled from the circuit board using a razor blade and needle-nose pliers. With the new arrangement, the RF output power measured 10 dBm and the second and third harmonics are at least 20 dB below the fundamental. Hence, the local oscillator can be used to directly drive a double-balanced mixer with a little power to spare. The enclosure cover plate being on or off had no noticeable effects. Later, I fabricated and installed a three-terminal voltage regulator, 7812, on a short piece of angle aluminum. Photo D shows the local oscillator in the new enclosure without the 7812 and Photo E shows the complete assembly. In future conversions, I will use a #30 wire between the regulator and the oscillator circuit board to reduce the stress on the trace.

I will be using a Clarke 167-J-2 FM receiver for an IF and was not overly concerned about the useable frequency range of the unmodified local oscillator. The Clarke, predecessor to Nems-Clarke, receiver has a tuning range of 55 to 260 MHz and I have a panadaptor for the 21.4 MHz IF frequency. The receiver IF bandwidth is 300 kHz, which is good for monitoring high speed telemetry. However, I did decide to try an assortment of six crystals from my junk box in the model 130002 test oscillator. The highest frequency crystals, 9.2250 and 9.1750 MHz, produced the same output frequency, 2,342 MHz, which is below the expected frequencies of 2,361.6 MHz and 2,348.8 MHz. So I believe the upper frequency limit of the unmodified VCO is 2,342 MHz. The other test crystals; 9.075, 8.975, 8.875, and 8.625 MHz, all produced the expected output frequencies and power. With no crystal, the output frequency was 2,167.0 MHz. The output power does begin to drop off a couple dB at the frequency limits. Based on the measurements, the absolute VCO range is 2,167 MHz to 2,342 MHz and the output bandpass filter is a little wider. The absolute VCO range implies a crystal range of >8.465 MHz to <9.148 MHz. Many amateurs have reported no problems with using an 8.8125 MHz crystal to get an IF frequency of 145 MH for an input frequency of 2,401 MHz. Information for ordering crystals from ICM is found on another Web site [8].

I removed the onboard crystal, the chip capacitors and trimmer connected to the crystal. I applied an external 0 dBm signal, terminated with a 47 Ohm load, through a 1000 pF series blocking capacitor to pin 1 (the crystal pin connected to the trimmer capacitor) of the FS4347E oscillator/phase detector integrated circuit. The shield from the reference source is grounded. There is no connection to pin 2. The VCO would lock for drive frequencies from 8.48 MHz to 9.13 MHz. The corresponding output frequencies are 2,171 MHz and 2,337 MHz, a range of +59 MHz to -107 MHz or a total range of 166 MHz. The lock limits of course do fall within the VCO low and high limits. Remember that this was with one specific circuit board, results may vary. But, if you do want increased stability, you can drive the PLL with an external oscillator or synthesizer.

The end result is a compact, stable, low power S-band source for little more than the cost of a crystal. For this effort, I have obviously drawn together the work of others, added some details of my own, and hope others will continue to add as they are able. This is in line with the spirit of Amateur Radio.

References

[1] Filby, Larry (K1LPS): "Drake 2880 Conversion Summary," http://www.nitehawk.com/rasmit/mds2880cv.html

[2] Moss, Adam (G0ORY): "California Amplifier 31732 Downconverter," http://www.qsl.net/n9zia/31732/index.html

[3] Flowers, Andy (K0SM): "An Easy Way to Receive S-Band Using the Cal-Amp 130016," http://rvhfg.org/articles/2/7

[4] Franke, John M., WA4WDL: "HP 8441A YIG Filter Spectrum Analyzer/Interface," Proceedings of the 12th Annual Southeastern VHF Society Conference, April 25-26, 2008, Orlando, Florida, pp. 143-145.

[5] Franke, John M., WA4WDL: "12.5 GHz and Counting," Proceedings of the 11th Annual Southeastern VHF Society Conference, April 27-28, 2007, Atlanta, Georgia, pp. 144-147.

[6] Cal Amp, http://www.calamp.com/home/pro_mmds.html

[7] http://www.cqham.ru/image2/cal-amp-dc_big.gif

[8] http://www.qsl.net/n9zia/wc0y/

Photos continue on next page ...

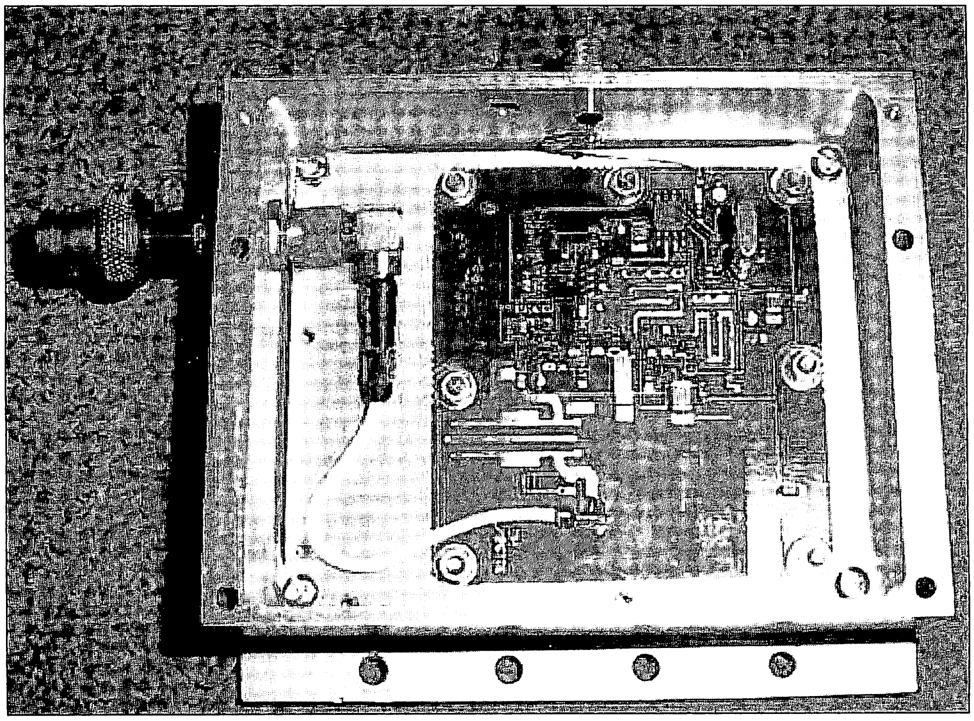


Photo D - Remounted PLL

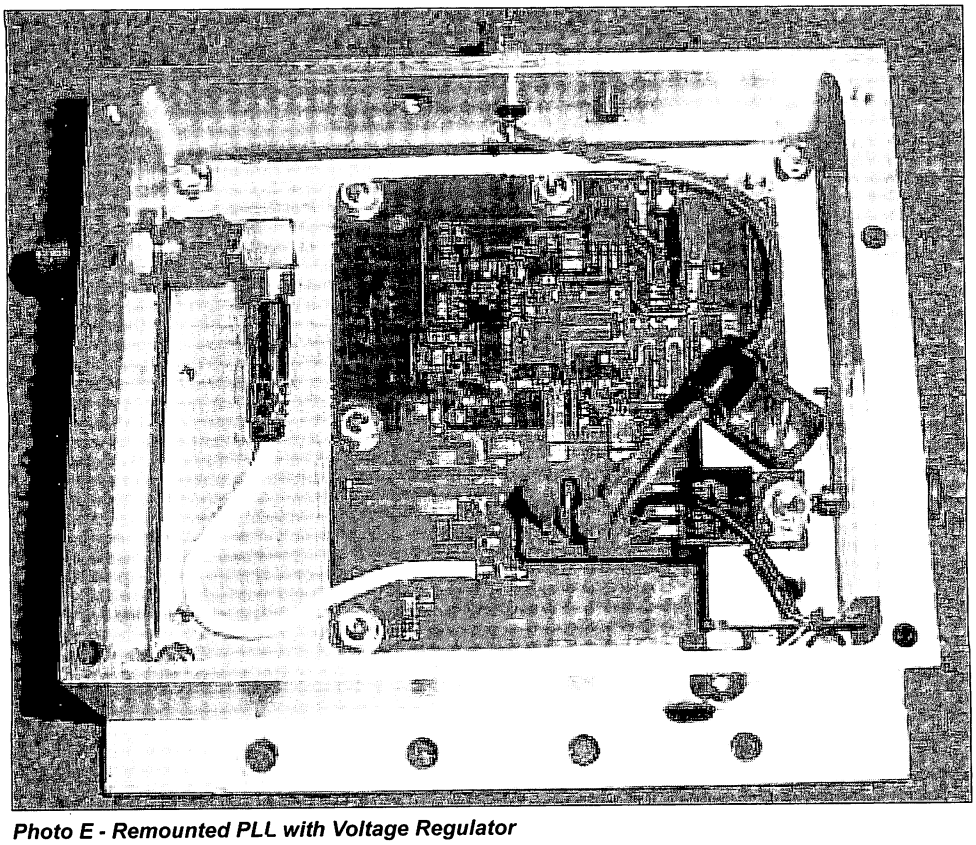


Photo E - Remounted PLL with Voltage Regulator

The BPSK1000 Telemetry Modem for ARISSat-1

Phil Karn, KA9Q, ka9q@amsat.org

Phil Karn's original paper was presented at the 2010 AMSAT Space Symposium held October, 2010 in Elk Grove Village, Illinois. This paper has been updated by the author. The original version is published in the Proceedings of the 2010 AMSAT Space Symposium.

Introduction

The ARISSat-1 (formerly SuitSat-2) spacecraft is planned for launch on a Russian *Progress* supply flight to the International Space Station (ISS) sometime in 2011 where it will be deployed as a free flying satellite.

ARISSat-1 will carry a new telemetry modulation and coding scheme, BPSK1000, designed to handle the severe fading often encountered with low orbit satellites without attitude control. Its performance and the link budgets for the ARISSat-1 spacecraft are such that reliable reception should require only a simple whip or ground plane antenna, a conventional 2 m SSB receiver and a reasonably modern personal computer with audio A/D input.

BPSK1000 uses differential binary phase shift keying (DBPSK) at a channel symbol rate of 1 kHz in a SSB bandwidth. With constraint length 7, rate ½ forward error correction (FEC), the user data rate is about 500 bits/sec. HDLC framing provides application flexibility (including the ability to carry AX.25 in other applications) and a deep (16 second) convolutional interleaver provides strong protection against fading.

I hope that the demonstrated performance of BPSK1000 on ARISSat-1 will encourage other amateur satellite projects with similar needs to consider it. At the very least, I hope to convince other satellite designers that forward error correction (FEC) should be a standard feature of *every* amateur satellite data link!

Software to demodulate and decode the ARISSat-1 BPSK1000 beacon will be available for Microsoft Windows, Mac OSX and Linux. I wrote the reference BPSK1000 demodulator and decoder in C under Mac OSX and Linux as libraries and command line tools and I am distributing them under the terms of the GNU General Public License (GPL). I strongly encourage everyone to study, modify, enhance, test, break, fix, improve, experiment with and use what I have written.

My reference decoder is incorporated into the Windows program by Doug Quagliana, KA2UPW, and the Mac OSX program by Gilbert Mackall, N3RZN, both of whom have turned a set of command line tools into complete, user-friendly applications.

My software can use, but does not require, multiple CPUs and the Altivec (Power PC) or SSE (Intel/AMD) vector instruction sets to improve performance, in some cases rather dramatically. Multiple CPUs are used through the POSIX Threads (pthreads) library, and vector instructions are invoked with gcc C complier intrinsics; there is no actual assembly code.

The rest of this paper discusses the physical layer modulation, interleaving, coding and framing elements of the generic BPSK1000 air interface. Its specific use by ARISSat-1 is documented separately.

Requirements

Engineering is all about making tradeoffs and the design of a digital modem is no exception. Every engineering project should start with a clear list of requirements. For a modem, making and meeting those requirements requires a good understanding of the channel on which it is to operate.

Since classroom educational demonstrations are a prime goal of ARISSat-1, my goal was to design a signal format that, while as robust and efficient as possible, should not require any special hardware beyond a general purpose personal computer and a 2 m receiver or transceiver. Nor should it require any special knowledge, training or practice beyond ordinary ham radio skills. I especially did not want to assume that the operator was already an experienced amateur satellite operator. I want to make it possible for *any* ham with the necessary equipment, not necessarily one already familiar with amateur satellites, to do a successful classroom demonstration. My hope is that this may entice more kids to become hams, and more hams to join AMSAT.

No one gave me a formal list of requirements beyond "it must work" and "we must have it before such and such date", so I made my own:

Requirement #1: No special RF equipment needed.

It must be possible to receive the ARISSat-1 telemetry with a stock 2 m SSB Amateur Radio transceiver. As nice as it would be to use a software defined radio capable of wideband operation, that would be more appropriate for a future mission, not one intended primarily for school demonstrations and general education.

Requirement #2: Good power efficiency.

DC power is at an absolute premium on any spacecraft and downlink transmitters invariably take the lion's share of that power. The downlink mode must be as power-efficient as possible.

Additionally, it should be possible to reliably receive ARISSat-1 telemetry with a small and simple antenna such as an ordinary whip or ground plane. Steered directional antennas should not be required.

Although it would be nice to work with FM-only transceivers, FM is just too inefficient for satellite use. Implementing an efficient modulation mode in software requires a linear transceiver and that currently means SSB. This leads to the requirement that the signal fit within the typical SSB voice filter response of 300 to 2700 Hz, a bandwidth of 2400 Hz.

A prime personal goal of this project – as with my previous effort that put an experimental FEC format on AO-40 – is to demonstrate that there exist far more power-efficient modulation and coding methods for amateur satellite telemetry links than are generally flown. Such improvements can either cut the cost of constructing a satellite by reducing the amount of DC power it must generate, and/or it can make amateur satellites accessible to more hams by reducing the size, cost and complexity of the ground antennas. Many hams live in homes with antenna restrictions where the traditional 2 m/70 cm Yagi combination simply isn't an option.

The AFSK/FM mode used by terrestrial amateur packet radio is *particularly* inefficient. Quite frankly I am at a loss to understand why anyone would fly it on a spacecraft.

Because forward error correction (FEC) codes can provide a significant "coding gain" that reduces the transmitted energy per bit, this requirement alone dictates the use of FEC, even without the additional dramatic improvements it provides in fading (see below). It has long been easy to achieve coding gains of up to 7 dB on channels impaired by only Gaussian (thermal) noise, and the latest codes can provide gains of up to about 10 dB.

Requirement #3: Fade tolerance.

Closely related to the previous requirement is the need to tolerate the unpredictable and often deep fades often encountered with amateur satellites in low earth orbit. Although the line of sight from satellite to ground station may not be interrupted, amateur spacecraft (such as ARISSat-1) are typically uncontrolled or partly controlled in attitude (e.g., with bar magnets) with the result that antenna nulls can sweep across the ground station and produce deep nulls. Multipath fading can also occur when omnidirectional ground antennas pick up reflections of the satellite signal from the ground or nearby buildings.

This requirement is closely related to #2 because an inefficient link (i.e., one without FEC) requires a lot of excess power to ride through a fade. An alternative is to keep retransmitting until the data gets through, but all those unsuccessful transmissions are also wasted energy. Fading represents errors that can be corrected by FEC, but not all FEC codes can tolerate the long bursts of errors associated with slow fading. Therefore, we need either FEC suited for burst error correction or an interleaver to break up error bursts into a long series of single bit errors that can be corrected more easily.

To see the huge benefit of FEC on fading channels, consider that without coding you need enough link margin – excess transmitter power – to ride through the usual fades; i.e., it must be designed for the worst case. Since the channel is faded for only a small fraction of the time, most of the time all that excess power goes to waste. On the other hand, a well designed FEC scheme for a fading channel will operate as long as the *average* signal-to-noise ratio is high enough. Brief fades, even when infinitely deep, can be reconstructed by the decoder from redundant information received when the channel is not faded. This difference between average and worst case SNR becomes the effective coding gain of the FEC. When the fades become very deep the coding gains can become truly dramatic. The link simply won't work without it.

Requirement #4: The average amateur should be able to tune the signal by ear without specialized skill or an excessive amount of practice.

Doppler shift from a LEO satellite on 2 m is about +/- 3.3 kHz, so tuning of a SSB receiver is obviously required throughout the pass. As desirable as computer controlled tuning may be, I did not want to require it.

I probably spent more time thinking about how best to meet this requirement than all the others combined. The problem is that efficient digital modes tend to sound like band limited white noise, and white noise is notoriously hard to tune to an exact frequency! [1]

Fortunately, an elegant and effective solution presented itself. The ARISSat-1 downlink also includes a CW beacon and I recommended that it simply be moved to the lower null of the digital telemetry signal. Here it will not interfere with the beacon and vice versa, but it can easily be heard and tuned by a human operator in such a way that it automatically tunes the telemetry signal as well.

Requirement #5: the transmitter must be simple.

Although ARISSat-1 carries a lot of processing power by satellite standards, CPU cycles and memory are both still quite limited in comparison to a typical personal computer. The IHU and DSP have many other things to do besides encode telemetry. Even when there's plenty of real time, you still want to minimize the number of executed instructions because every one of them requires some amount of energy to execute.

The only element of signal generation that created any concerns was the BPSK modulation. Not the modulation per se, but the filtering associated with it. The FIR filter is 454 taps long, but not every tap needs to be computed on every sample. By signaling with impulses most of the delay line will contain zeros, allowing those multiplications to be skipped.

Fortunately, generating BPSK1000 is relatively simple – certainly far simpler than demodulating and decoding it!

Signal Design

My job here is to carry packets or frames of arbitrary data from the ARISSat-1 experiments and onboard computer (IHU) to a ground station where they can be decoded and displayed by a computer running special software. This paper does not define or describe the actual data to be carried on this link, so my description here starts with the framing layer and works down to the modulation used on the physical channel. Consult the other papers on ARISSat-1 for information on the contents of the frames actually transmitted during flight.

BPSK1000 encoding can be succinctly described as follows:

- HDLC framing with a 32-bit CRC;
- constraint length 7, rate ½ convolutional forward error correction (FEC) coding;
- 128-way convolutional interleaving with "bit reversed" delay line ordering; and
- DBPSK (Differential Binary Phase Shift Keying) modulation.

These elements are all "off the shelf", well documented in the textbooks and used commercially for many years. What follows is a more detailed description and my rationale for selecting each one for this project.

Framing

When I started work on this format, the telemetry data formats and sizes were not yet established. Rather than pick a frame size that would almost certainly turn out to be wrong, I chose to support variable length framing for its versatility.

One of the best known and most widely used methods of variable length framing on synchronous bitstreams is HDLC, familiar to most hams from its use in the AX.25 packet radio protocol. Stripped to its bare essentials, an HDLC frame in BPSK1000 is shown in Figure 1.

The special 'flag' sequence 01111110 represents the end of one frame and the start of another. (Two frames may share a common flag or there may be extra flags between frames - the decoder will work either way.) As called for in the HDLC standard, user data is transmitted least

01111110	**Data (variable)**	**CRC-32**	**01111110**

Figure 1: HDLC frame.

significant bit first between the opening flag and the CRC.

To ensure transparency, the HDLC encoder inserts a '0' bit whenever five contiguous '1' bits occur in the data stream. When the receiver sees five '1' bits followed by a '0', it automatically removes the '0'. Although bit stuffing can consume up to 16.7% of the channel capacity in overhead, this occurs only for a data stream of all 1's. The overhead for bit stuffing on random data is much less.

I made only one significant change to standard HDLC framing as used in BPSK1000: I replaced the usual 16-bit CRC with a 32-bit CRC with the generator polynomial: [2]

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$

The stronger CRC greatly reduces the chance that a random bit stream will decode as a valid frame.

BPSK1000 treats the contents of the "data" field as opaque, i.e., it does not care about their contents. Although ARISSat-1 doesn't send AX.25 headers, nothing prevents another BPSK1000 application from doing so. I.e., BPSK1000 could easily carry standard AX.25 packet radio data, although for reasons discussed later it is not well suited to the half duplex, shared channel mode of operation of most amateur packet radio channels.

The fact that BPSK1000 does not assume the presence of AX.25 headers is one reason I decided to enlarge the CRC. Spurious HDLC frames are common in packet radio but they are easy to detect by their absence of an AX.25 header. Because I did not want to make any assumptions about the contents of ARISSat-1 HDLC frames, I chose a longer CRC that all but eliminates spurious frames in the first place.

In sum, HDLC converts an asynchronous stream of arbitrary length frames into a continuous 500 bps bit stream for the FEC encoder. If there's nothing to send, the HDLC encoder must emit an idle flag stream. The HDLC decoder at the receiver regenerates the original series of frames with the 32-bit CRC providing strong error detection. Although BPSK1000 includes forward error correction (FEC), the HDLC CRC is necessary because a Viterbi decoder still makes errors when its error correction ability is exceeded and it cannot reliably signal when they occur.

Convolutional Error Correction Coding

To allow for the correction of a limited number of channel errors, the HDLC-encoded bit stream is convolutionally encoded with the CCSDS standard constraint length 7, rate ½ convolutional code using connection vectors G1=1111001 binary (171 octal) and G2=1011011 binary (133 octal): [3] (See Figure 2) (http://public.ccsds.org/publications/archive/131x0b1.pdf)

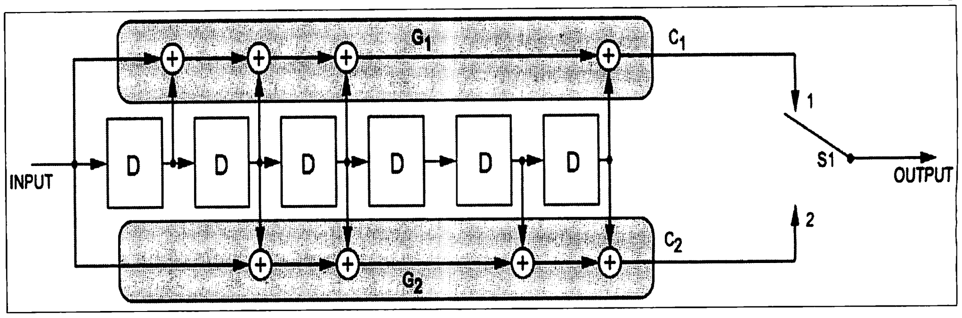


Figure 2: HDLC encoder.

For each data bit from the HDLC encoder, the convolutional encoder generates two symbols, C1 and then C2. I use the CCSDS convention for the ordering of the two encoded symbols but I do not invert either symbol.

The CCSDS and JPL versions of this encoder both invert the output of the 133 (octal) XOR ladder (shown here as symbol C2). Their reason for doing so is to limit the maximum possible number of contiguous 0's or 1's that can come out of the encoder regardless of the data so as to guarantee a minimum symbol transition density for demodulator tracking. That is unnecessary here because HDLC bit-stuffing ensures that no more than five consecutive '1' bits can ever occur in user data, and an HDLC flag consists of six consecutive '1' bits. An HDLC abort, if ever generated, need (and should) consist of only seven '1' bits, followed by an idle flag stream until more data is available for transmission. [4]

Although HDLC does not limit runs of '0' bits, this is not necessary here because of the use of differential encoding before BPSK modulation that turns every '0' bit into a 180° carrier phase transition. It is only necessary to break up consecutive '1' bits as differential encoding sends them as no change in carrier phase.

Convolutional Interleaving

Viterbi decoders tolerate burst errors poorly. Since a major requirement for ARISSat-1 is to mitigate the deep fading common with unstabilized satellites in low earth orbit, an interleaver is necessary.

Interleaving scatters the encoded symbols in time before transmission and rearranges them into their original sequence at the receiver before decoding. This has the effect of breaking up a burst of errors into a much longer sequence of single bit errors that are much more easily corrected.

There are two major classes of interleavers: block and convolutional. Block interleaving is straightforward. The encoded symbols are written into an array by rows and read out by columns for transmission. The receiver reverses the process by writing the array by columns and reading it out by rows.

Delay can be a problem with block interleaving, especially when a large block is required to tolerate deep fades. Nearly the entire block has to be written before transmission can start, and then the entire block has to be received before decoding can even begin. Block interleaving usually requires that a single block size be selected in advance, though some systems use several sizes. However, this doesn't necessarily restrict the system to a fixed data frame size; HDLC or something similar could support variable length data frames just as it does here.

These drawbacks are not necessarily serious. For example, in its FEC mode the AO-40 IHU generated fixed sized 256-byte data blocks more or less instantly, so the delay from block generation in software to the decoded block being available on the ground was equal to just one block transmission time (13 sec) plus the times needed for the IHU to encode it and the ground CPU to decode it.

But when the source is a steady bit stream, or a series of small data frames spread out in time, the encoder must wait for the interleaver block to fill before it can even begin to transmit it. The last data frame in the block will not see any difference, but this can nearly double the time between the generation of the first (small) data frame in the block and when it is available on the ground. [5]

An alternative to block interleaving is convolutional interleaving, [6] and I chose it for BPSK1000 to help reduce latency when small HDLC frames are carried. Just as a convolutional encoder differs from a block encoder by operating on a continuous data stream rather than fixed-sized blocks, a convolutional interleaver also operates on a continuous stream. This is well suited to the continuous stream of encoded symbols from the convolutional encoder.

Here is a block diagram of a small (order 4) convolutional interleaver (Figure 3):

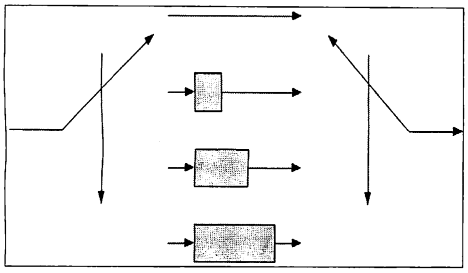


Figure 3

The symbol stream to be transmitted is fed in on the left to the 1:4 rotary switch that deposits each symbol in turn into one of four delay lines whose lengths range from 0 in the first row (no delay) to 3 time units in the last row. Their outputs are collected by the 4:1 rotary switch operating synchronously with the input switch. The delay lines clock only when selected by the rotary switches, which operate synchronously, and the output of the right hand rotary switch goes to the transmitter.

At the receiver, the incoming symbol stream passes through the deinterleaver, the mirror image of the interleaver (Figure 4):

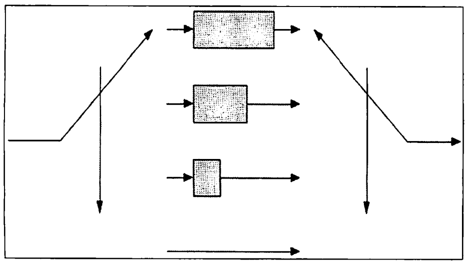


Figure 4

The essential feature of convolutional interleaving is that in each row, the length of the interleaver delay line plus the length of the deinterleaver delay line is a constant, N. In this example, $N = 3$. So the overall effect of the interleaver and deinterleaver combined is to reproduce the original input stream, in its original sequence, delayed by a number of time units equal to N times the number of rows R; in this example, that would be 12 time units.

N can actually be any number you choose, though it can't be less than R-1 if each row is to have a delay on each side of the channel different from all the other rows. Otherwise some input symbols will be sent in their original order, and that defeats the point of interleaving. And if you make N bigger than this, you're just adding unnecessary delay. So for all practical purposes, $N = R$-1.

The main advantage of convolutional interleaving over block interleaving is that less memory and less end-to-end delay are required for a given amount of fade protection. One can think of convolutional interleaving as being something like block interleaving but with a single block split between transmitter and receiver.

Convolutional interleaving also has its drawbacks. Because there's no well defined start or finish to the data stream as in block interleaving, it is not well suited to short, bursty transmissions like those on a shared, multiple access channel (e.g., conventional packet radio). Conversely, a system that uses block interleaving could key up, send a single block and stop transmitting without having to allow any time to "prime" the de-interleaver or to flush the interleaver.

But that doesn't matter for our application because it's a one-way continuous broadcast. The ground station will still need to prime its de-interleaver at the beginning of the pass. At first this might seem a disadvantage relative to block interleaving, but it's not. Satellite AOS can occur at any time, and were it to occur in the middle of a block in a block-interleaved system you'd have to wait up to one block time for the first full block to start. With convolutional interleaving, priming of the de-interleaver takes the same amount of time no matter when you start.

Bit-reversal convolutional interleaving

In most convolutional interleavers, the delay in each row is monotonically increasing (and decreasing at the deinterleaver, or vice versa). But this isn't mandatory. Just as the rows and columns of a block interleaver can be written in any order (even random) so long as the receiver mirrors the transmitter and puts everything back in its original place, anything goes.

This got me thinking about playing with some alternative orderings. One common trick in block interleaving is to access the rows and columns in "bit reversed" order. Take the integers from 0-7 and write them in binary:

000, 001, 010, 011, 100, 101, 110, 111

Now reverse them left-right and convert back to decimal:

0, 4, 2, 6, 1, 5, 3, 7

Now access the rows or columns in that permuted order.

Why do this? Well, it helps to spread out adjacent input symbols during an error burst (e.g., a fade). The first errored symbols in the burst are as far apart as they can get in the memory available. Additional errors are evenly sprinkled between the previous errors, though of course if the burst lasts long enough it will eventually have to fill in all the gaps and decoding failures will occur.

Bit reversal interleaving requires that the number of rows be a power of two, so I chose an interleaving order of 128. This implies that each row contains a total delay of 128 bit times divided between transmitter and receiver for an end-to-end interleaving/deinterleaving delay of 16,384 symbols. [7]

Modulation BPSK? QPSK?

The power efficiency requirement pretty much mandates either BPSK or QPSK. In theory, QPSK (quadrature phase shift keying) requires the same power as BPSK and only half the bandwidth. In practice it is difficult to make QPSK perform as well as BPSK especially when the data rate is low compared to the frequency uncertainty, as is the case here, because QPSK has much tighter receiver requirements than BPSK for carrier frequency and phase tracking. So despite the bandwidth limitations of SSB filters I decided to stick with BPSK. It has a very long history on Amateur Radio satellites and satellites in general. [8]

The requirement that the signal fit a SSB receiver means that the modulated bandwidth cannot exceed a typical SSB voice filter, usually about 2400 Hz. It should be somewhat narrower to allow for errors in tuning and to stay away from the nonlinear group delay distortion typically found at the edges of crystal filters designed for voice operation where such delay isn't a serious problem.

Signaling rate

The Nyquist bandwidth of BPSK is 1 bit/

sec/Hz. That is, I could in principle fit 2,000 bits/sec into 2 kHz of a SSB passband while leaving 200 Hz on either side. But achieving 1 bit/sec/Hz with BPSK requires very tight filtering with very little tolerance of symbol timing errors and group delay distortion. The FIR filter used at both transmitter and receiver must be quite long to achieve the necessary response, and processing time in the onboard DSP is limited.

Tightly filtered BPSK also generates a fair bit of overshoot in the RF envelope. This rules out efficient constant-envelope amplification, although that's not a problem in ARISSat-1 because it already gives us a linear amplifier for the entire 2 m downlink including the telemetry beacon.

So to be conservative, I selected a BPSK symbol rate of only 1 kHz (i.e., 1000 symbols/sec) with a raised-cosine filter having 100% excess bandwidth, i.e., 1/2 bits/sec/Hz. This is achieved with a finite-impulse-response (FIR) filter with 454 taps running at a sample rate of 48 kHz; the same filter taps are used as the matched filter in the demodulator on the ground. Although the resulting bandwidth is still 2 kHz, instead of being flat the signal spectrum rolls off from a peak at the nominal center frequency to nulls at +/- 1 kHz, with relatively little energy near the nulls.

Manual tuning with the CW beacon

As mentioned in the requirements section, the CW beacon is placed in the lower null of the telemetry beacon spectrum, 1 kHz below the nominal carrier frequency of the BPSK signal. All the operator need do is tune the receiver in SUB mode so that the CW beacon comes out at 500 Hz, and this will automatically center the BPSK signal in a typical SSB receiver bandwidth; i.e., the signal will extend from the lower null at 500 Hz to an upper null at 2500 Hz with the carrier at 1500 Hz, the nominal center of the SSB passband.

Why not coherent BPSK?

Because our spacecraft will transmit on 2 m from low earth orbit, path losses will be relatively low and the signal will, on average, be fairly strong. But experience has shown that unstabilized spacecraft in low earth orbit often exhibit deep, unpredictable fading. That's the reason for the deep interleaver.

Fading (or its absence) is also a major consideration in the design of BPSK modems as it can be implemented in two very different ways depending on which channel impairment is more important: thermal (Gaussian) noise or fading.

Satellite links are traditionally designed to minimize thermal noise, so most use a form of PSK that requires the receiver to form a clean local copy of the unmodulated signal carrier to serve as a phase reference for the receive signal. This is coherent demodulation.

This can be done in one of two ways. The first is for the transmitter to send a residual carrier along with the data so that the receiver can track it with a conventional phase lock loop (PLL). The second is for the receiver to reconstruct the carrier entirely from the data by stripping off the modulation.

Although residual carriers are still used in some deep space links, particularly at the extremely low data rates sent by a spacecraft in "safe mode", this takes power away from the data. It is much more common to suppress the carrier entirely at the transmitter so that all of the power can go into the data. The receiver recovers the carrier from the data with one of two mathematically equivalent methods: the squaring loop and the Costas loop.

Especially when strong FEC is used, these loops can suffer significantly from "squaring loss", an effect that reduces the signal-to-noise ratio of the recovered carrier. This happens because the raw symbols (before FEC decoding) are very weak, so when the loop strips off the modulation it does so with a fairly high error rate that degrades the recovered carrier.

Squaring loss can be overcome by merely narrowing the loop filter, effectively averaging out the noise over a longer interval that may span hundreds or even thousands of channel symbols. This is fine if the signal is very stable, as on most deep space links. But it can become completely unworkable when the signal is subject to rapid fading, e.g., on ionospheric or land mobile channels, especially at the low symbol rates we're using here.

We also have fading problems with our low earth orbiting spacecraft. Although we may have continuous line of sight, the spacecraft may tumble unpredictably and sweep antenna nulls past the receiver, causing deep fades accompanied by a sudden reversal of signal phase that would defeat a conventional coherent BPSK demodulator of the Costas or squaring loop type.

Differential BPSK

For this reason I have chosen a form of BPSK for ARISSat-1 that doesn't require the receiver to form a carrier phase reference. This is DBPSK, differential binary phase shift keying. It is generated just like ordinary BPSK with one crucial exception: before it modulates the transmitter, the data stream is differentially encoded so that a binary zero is transmitted as a *180° change* in the phase of the transmitted carrier while a binary one is transmitted as no change from its previous state. This compares with conventional BPSK where a binary zero corresponds to the RF carrier with a 0 degree phase shift and a binary one corresponds to the RF carrier shifted by 180°.

The advantage of DBPSK for our application can be seen in how it is demodulated. The receiver can simply compare adjacent incoming symbols, looking for sudden 180° changes in phase - whatever it may be.

This is extremely easy to implement. In software (the only way to build modems nowadays!), the standard way to demodulate DBPSK is to first convert the input signal to complex baseband at zero frequency. That is, we generate a complex sinusoid (a matching sine and cosine wave) at the estimated signal frequency and multiply it by the input signal (which can be either complex or real) to produce a complex baseband signal at (approximately) zero frequency. If our frequency estimate is correct, an unmodulated carrier becomes a stationary phaser at some arbitrary phase angle (since we don't know or care about its phase). Should the signal flip phase by 180°, this phaser will rotate 180°. By computing the dot product of pairs of complex symbols we simultaneously demodulate the data and remove the differential encoding; this works because the dot product of a vector with itself is a positive number while the dot product of a vector with itself flipped 180° is a negative number.

So instead of generating a nice clean phase reference by averaging some hundreds of symbols, in DBPSK demodulation we just use each symbol as the phase reference for the next. The dot product yields a scalar (a number) that tells us how certain the symbol estimate is, just what we need for our soft-decision Viterbi decoder.

Effects of frequency errors on DBPSK demodulation

What is the effect of an error in the frequency estimate? If the local oscillator doesn't match the incoming signal frequency, then the baseband phaser will rotate clockwise or counterclockwise depending on whether the oscillator is above or below the actual signal frequency. But if the error is small, the vector will rotate very slowly. It may still rotate many times in a second, but because we're only comparing adjacent symbols – not trying to form a reference from hundreds of them – it only matters that it rotate no more than a few degrees during one symbol time.

Let's say the BPSK is at a symbol rate of 1 kHz and our frequency estimate is off by 100 Hz. The baseband phaser will rotate 100 times per second, but that's only 1/10 of a rotation (36°) from one symbol to the next. Because the magnitude of the dot product of two vectors is proportional to the cosine of the angle between them, a small frequency error decreases the magnitude of the dot product to cos(36°) = 0.81. This corresponds to a loss of $20*\log_{10}(0.81) = -1.84$ dB. If the frequency error can be kept to 50 Hz, then the loss is only $20 * \log_{10}(\cos(18°)) = -0.44$ dB, a tolerable figure.

Decoding BPSK1000

Nearly every digital format is easier to generate than it is to decode and BPSK1000 is no exception. This is because the receiver has to first determine various signal parameters before it can demodulate and decode the signal and this searching and tracking consumes most of the CPU time.

Input format

My BPSK1000 demodulator accepts a 16-bit linear PCM audio signal sampled at 48 kHz. The sample rate is a compile-time constant; although it can be changed, the structure of the code requires it to be an integer multiple of the symbol rate (1 kHz) so if you have a signal at some other sample rate (e.g., 44.1 kHz) it's easier to simply convert it with a tool like *sox*.

Chunking

My demodulator operates on blocks of PCM samples called "chunks". The chunk size, currently 512 symbols (512 ms), is chosen to contain enough energy for the carrier frequency and symbol timing search while being short enough that these parameters remain essentially constant within the chunk.

Carrier frequency and symbol timing search

The first step in decoding BPSK1000 is to determine the carrier frequency and symbol timing. Because it's DBPSK, an estimated frequency will do; as shown above, +/- 50 Hz will do, so I search in 100 Hz steps. I must also determine where each symbol starts and stops. So I run a series of trial demodulations with every possible combination of carrier frequency (in 100 Hz steps) and symbol time. Since the sample rate is 48 times the symbol rate, this means trying 48 different hypotheses for symbol timing.

No attempt to acquire data is made at this stage. Only the total demodulated energy is calculated and the carrier frequency and symbol timing parameters that yield the largest demodulated energy are taken as the best estimates. This step is both CPU intensive and parallelizable, so it's a natural for a machine with multiple CPU cores. But in practice this search is actually pretty fast except on very old and slow machines thanks to a fast FIR filter that uses vector instructions.

Carrier frequency refinement

After the initial estimate, the carrier frequency estimate is refined by simple linear extrapolation. As explained above, when a frequency error is present the baseband signal vector slowly rotates. This rotation can be detected by summing the magnitudes of the vector cross products of adjacent pairs of baseband phasors. When those vector cross products are zero (except for noise) the frequency error is zero. This step is actually overkill since, as stated, 50 Hz frequency errors have little effect on demodulation. I do it mainly to give the user a more precise frequency reading.

Tracking and demodulation

Once the carrier frequency and symbol timing are determined, there's no need to repeat the full-blown search unless the signal is lost. Changes in frequency and timing between each chunk are tracked by simply running a trial demodulation (and looking at the demodulated energy) at values bracketing the current estimates.

Then the data can finally be demodulated.

Deinterleaving and decoding

Convolutional interleaving, as described above, assumes that the rotary switches in the interleaver and deinterleaver are synchronized. If not, the deinterleaver produces garbage.

There are no sync patterns in the BPSK1000 format to indicate interleaver phasing. Correct interleaver phase has to be found by brute force and this automatically synchronizes the Viterbi decoder phase as well. Once the DBPSK demodulator has acquired and demodulated a symbol stream, the soft decision samples are scaled and fed to 128 virtual deinterleavers, Viterbi decoders and HDLC decoders, one for every possible interleaver phase. Eventually, one of them will produce a HDLC frame with good CRC. Now that we know the correct phasing, the other 127 virtual decoders can be shut down. They remain shut down as long as we continue to get valid HDLC frames. If too much time passes without a valid frame, then all 128 decoders are re-enabled until another frame is decoded.

This was my main motive for using a 32-bit CRC with HDLC; I was concerned that spurious frames would occur frequently with the 16-bit CRC and interfere with the determination of the correct deinterleaver phase.

This brute force approach isn't nearly as bad as it sounds; even when all 128 decoders are running, DBPSK demodulation still takes about half of the program's total CPU time. Demodulation and decoding run in separate threads, both for architectural convenience and to take advantage of extra CPU cores.

Concluding thoughts

After the launch of AMSAT-OSCAR-40, I designed and implemented an experimental FEC-coded telemetry format (**http://www.ka9q.net/ao40/**) designed to deal with the deep periodic fading characteristic of the S2 transmitting antenna at high squint angles.

This project was very successful, though it was eventually cut short by the complete failure of the spacecraft. I learned a great deal from that experience, and while the ARISSat project has a different set of design objectives and constraints, I was able to reuse many important design elements.

Yet ARISSat-1 will be a very different spacecraft than AO-40, if only because it will operate from LEO while AO-40 was in a high orbit. BPSK1000 is also rather different from my AO-40 FEC design, partly because the latter was constrained to work with the existing Phase III IHU hardware and software: the symbol rate had to be 400 Hz, the modulation had to be BPSK with Manchester (biphase) encoding, the data was fixed-length 256-byte blocks, and so on.

I had none of those constraints with ARISSat-1. In some ways that lack of constraints actually made BPSK1000 more difficult to design, but I think the result is better for it.

Although I've used mostly "off the shelf" modem design elements, as a system BPSK1000 is a new design that has never flown on an Amateur Radio satellite. I would have liked to implement a range of operating modes, e.g., data rates, bandwidths and interleaver depths, selectable by command so that the optimal mode for the actual conditions could be selected in flight. This wasn't possible so I designed a single mode to be as robust as possible, e.g., by picking a conservatively user low data rate. I very much look forward to seeing it operate from space so we can gain the experience necessary to design improved modems for future flights. With digital signal processing both on the ground and on the spacecraft, and with the new generation of software defined radios, the potential for new amateur satellite digital modes is extremely bright.

Footnotes

[1] Clarke's Law (after Arthur C. Clarke) states that any sufficiently advanced technology is indistinguishable from magic. My corollary to Clarke's Law states that any sufficiently advanced communication scheme is indistinguishable from white noise.

[2] For the 32-bit CRC polynomial I chose the one used in Ethernet, sometimes known as CRC-32.

[3] The polynomials and connection vectors for code generator polynomials can be written with either the most or least significant term first, often leading to considerable confusion. The convention here is taken from JPL Deep Space Network Handbook 810-005, 208A, page 12, in which the leftmost bit in the connection vector represents the taps encountered by the newest data bit entering the encoder. Some other references, plus my own software, use the opposite convention. This makes the vectors 1001111 and 1101101 binary; 117 and 155 octal, or 6D and 4F hex.

[4] The interleaver complicates this issue considerably. Even with HDLC bit stuffing (which I use) and alternate FEC symbol inversion (which I don't use) I can conceive of pathological cases where the interleaver might produce a very long string of 1's from a sequence with a bounded number of them. I could have added a scrambler, as I did for AO-40, but with the addition of HDLC bit stuffing I just didn't think the remaining problem was serious enough to warrant it here.

[5] As an analogy, consider a group of shuttle busses that run on demand. Each bus carries 50 people and the drivers like to fill all the seats. If you're part of a group of 50 people who arrive at the bus stop together, you can quickly board the first bus and leave right away. But if 50 people arrive individually over the space of 15 minutes, the first person to board the bus will have to wait 15 minutes for the bus to fill before it leaves.

[6] Not to be confused with convolutional *coding*, which I'm also using here!

[7] Strictly speaking the delay should be 127*128 = 16,256 symbols but it the code was easier to write with a little more delay.

[8] Some amateur satellites using BPSK include the Phase 3 series (Phase 3A and Oscar 10, 13 and 40) and the Pacsat series, e.g., Oscar 16.

References

HDLC framing is described in many places. Just a few include:

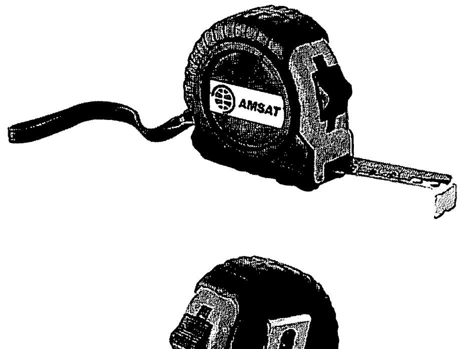
- AX.25 Amateur Packet Radio Link Layer Protocol Version 2.2, 1998, **http://www.tapr.org/pdf/AX25.2.2.pdf**
- International Telecommunications Union X.25, Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit, **http://www.itu.int/rec/T-REC-X.25/en**
- International Standard ISO/IEC 13239, "Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures", **http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37010**.

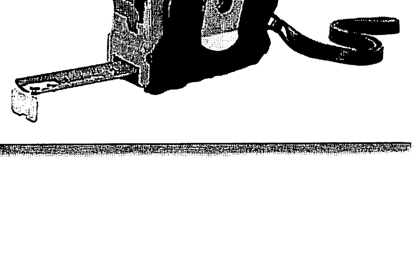
The k=7 r=1/2 convolutional code is described in many places including:

- DSN Telecommunications Link Design Handbook, 208, Rev. A Telemetry Data Decoding, May 18, 2009, **http://deepspace.jpl.nasa.gov/dsndocs/810-005/208/208A.pdf**
- CCSDS 131.0-B-1, TM Synchronization and Channel Coding, Blue Book Issue 1, September 2003. **http://public.ccsds.org/publications/archive/131x0b1.pdf**
- FEC Encoding for AO-40 Telemetry, including links to my published paper in the 2002 AMSAT Annual Meeting. (http://www.ka9q.net/ao40/)

The AMSAT Store (see www.amsat.org) is back online! Here is a new AMSAT product. A handsome and functional AMSAT 10' locking carpenter's tape with both inch and metric scales.

Minimum donation: $5





QST

DEVOTED ENTIRELY TO AMATEUR RADIO

February 2011 WWW.ARRL.ORG

ARISSat-1

SuitSat's Successor

Page 30

```
*****************AUTO**5-DIGIT 32640
0008361654 AA4FL F0 05/2011
JA1 QST_1 FEB11 D056      #222 #47142
JAY GARLITZ
22829 SE 63 PL
PO BOX 1333
HAWTHORNE FL   32640-1333
```

Official Journal of ARRL The national association for AMATEUR RADIO™

Get Ready for ARISSat-1

From space to classroom with the first Software Defined Amateur Radio transponder in orbit.

David Jordan, AA4KN, for the ARISSat-1 Team

Several years ago, an idea surfaced out of the Russian space program of turning a retired Russian Orlan spacesuit stored on board the International Space Station (ISS) into an Amateur Radio satellite. The suit would be filled with Amateur Radio equipment, connected to a battery and an antenna mounted on the helmet, and then released into orbit during an extra-vehicular activity (EVA). Sergey Samburov, RV3DR, of RSC-Energia introduced the idea at the 2004 AMSAT Symposium in Washington, DC. At this meeting, Lou McFadin, W5DID, of AMSAT-NA introduced the name *SuitSat* for the project and the name stuck.

This idea came to fruition when, on February 3, 2006, cosmonaut Valery Tokarey quite literally shoved the spacesuit into orbit from the ISS with the parting words, "Goodbye...Mr Smith." And with that, SuitSat-1 began its journey, sailing on a slow descent as it orbited Earth. For a couple of weeks it transmitted a collection of stored messages and other data until its final transmission was heard on February 18, 2006.

Unfortunately, due to a still unexplained problem, SuitSat-1 was extremely hard to hear on the ground for all but the most sophisticated stations. But, nevertheless, the "successful failure" of SuitSat-1 piqued the interest of millions, not to mention energizing its innovative experimenters into building a follow-on project they tentatively called *SuitSat-II*.

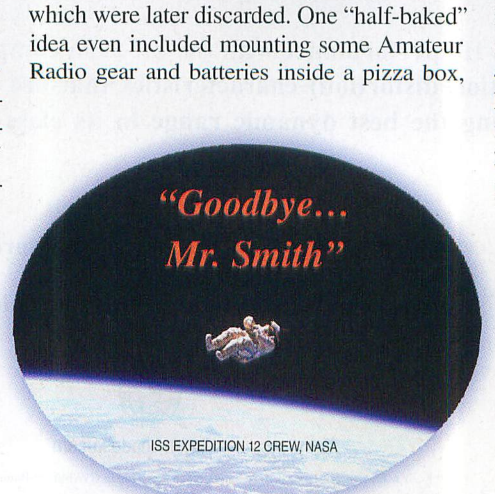
In November 2006, discussions began with the idea of creating a new and improved SuitSat. Again, the plan was to use a retired Orlan suit, but with some significant improvements. For example, SuitSat-1 contained only a battery for supplying power. This new "satellite in uniform" would have solar panels attached to its legs for recharging its onboard battery along with modular subsystems that, once designed, could be easily replicated for future SuitSats. And to really push the envelope, the spacesuit turned satellite would also contain a software defined transponder (SDX), a first for any ham satellite up to that time.

Things were going well and excitement was building until July 2009 when, out of necessity to make room on the ISS for some new orbital occupants, the Orlan suit tagged for SuitSat-II had to be discarded. The good news was that the Amateur Radio on the International Space Station (ARISS) team would still be given the opportunity to fly the radio gear up to the ISS and have it deployed during an EVA. But there was the lingering problem of what would house all the radio gear, solar panels and batteries.

For many years, AMSAT had been toying with the idea of building a small satellite that could be tossed overboard from the space shuttle. The idea took on many forms, most of which were later discarded. One "half-baked" idea even included mounting some Amateur Radio gear and batteries inside a pizza box, sticking a few antennas on the outside and tossing it overboard...an idea that, for obvious reasons, came to be unceremoniously dubbed "pizza sat."

So, once again the ARISS team drew on a similar concept and came to the conclusion that, since they no longer had a spacesuit available for the project, how about using a space frame instead? And, sure enough, almost, immediately after the announcement that an Orlan suit was unavailable, work began in earnest to reconfigure SuitSat-II's components to both fit and operate inside a new housing. But, there was another issue to resolve. Since there would no longer be a spacesuit involved, the name needed to change. ARISS-International later decided that the craft would now be called *ARISSat-1* with its full official name being *ARISSat-1/Radioskaf V*.

By the time you read this, ARISSat-1 will have been shipped to Russia for integration of the Kursk Student Experiment as well as testing with the Russian battery installed. It was to have then been shipped to Kazakhstan in December where it was due to be loaded onto the Russian Progress supply vehicle 41P and launched to the ISS in January 2011. Deployment of this Amateur Radio experiment to the ISS is scheduled for February 2011 during EVA #28 and, if all goes as planned the craft will be in full operation 15 minutes after its release.



So...What's in the ARISSat-1 Space Frame?

Well...quite a lot! ARISSat-1 is basically an aluminum frame with modules inside (Figure 1). The overall size of the unit was determined by the solar panels that measure 19 × 10.5 inches and are mounted on all four sides and the top and bottom plates of the craft. The modules contain the various subsystem circuits. The circuits interconnect allowing the satellite to carry out its on-orbit functions. Let's take a look at the subsystems on board.

First of all, there are a total of six solar panels — one on each of the four sides plus on the top and bottom. Each panel can generate 50 V and more than 19 W of power. The output of each panel connects to its own circuit in the Maximum Power Point Tracker module (or MPPT) where the power from each panel is optimized. Power from the solar panels is used to run the satellite and recharge its battery. Having a charged battery is especially important for supplying power to the spacecraft when it's in darkness ("eclipse"). The battery is the same type used on the Russian Orlan spacesuits and was donated by RSC-Energia. There is an RF module containing a 2 meter communications transmitter that connects to a whip antenna mounted on the satellite's top panel and a 70 cm receiver with a whip on the bottom panel. The module also houses an SSB/CW transponder that will (hopefully) be easily accessible in orbit by users running less than 5 W.

All Earth-orbiting satellites must have a means of being controlled from Earth as necessary. So, a 70 cm command receiver, always listening for commands from hams serving as ground control stations is also contained within the RF module.

As a safety precaution, it is important that ARISSat-1's transmitter and solar panel power system remain inactive until after its deployment. The control panel made up of three toggle switches is mounted on the top plate of the satellite. Just before ARISSat-1 is released into space, a space-walking crewmember will turn on all three switches. This starts a timing sequence that delays activation of the transmitter and generation of power by the spacecraft for 15 minutes to insure there will be no RF interference with the crew member's spacesuit electronics until ARISSat-1 is well on its way.

In addition to the radio gear carried aboard, the satellite also has a total of four cameras mounted on the top and bottom of the spacecraft. These cameras will receive power just prior to release and are designed to capture images of the deployment for later transmission. They will also continue to operate while in orbit, supplying views of the Earth and space via slow scan television (SSTV) to those stations so equipped.

Protruding from the top plate and resembling a silver colored "top hat" is the Kursk science experiment. This is an experiment developed by students at the Kursk State Technical University in Russia. Its purpose is to take periodic measurements of vacuum as ARISSat-1 continues its gradual descent into the Earth's atmosphere.

The Internal Housekeeping Unit (or IHU) is the processing center for the satellite. It is here where all analog and digital signals from the modules are routed and converted to a usable form to do particular tasks. The main "brains" of this unit is a PIC32MX processor that provides the overall control of the satellite's systems and generates telemetry to report the health of the spacecraft. A second PIC32MX is the first software-defined transponder (SDX) ever to fly on a ham satellite.

What Happens after ARISSat-1 is Switched On?

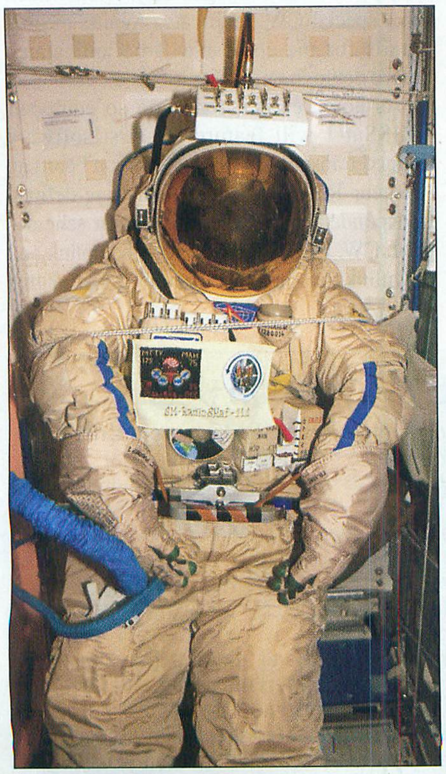
Just prior to ARISSat-1's EVA, clear Lexan solar panel covers will be carefully removed from all sides of the satellite frame and replaced by protective soft covers. These soft covers will be removed before deployment. One of the EVA crewmembers will slowly guide the craft through the open hatch and then engage the three control panel switches beginning from left to right. Flashing yellow LEDs on the panel will begin their slow cadence indicating that the 15 minute "countdown to power up" sequence has begun. The crewmember will then grasp the space frame's large corner handles, carefully angle the craft and push ARISSat-1 into a gradual rearward separation from the ISS.

ARISSat-1 is initially expected to orbit at a height around 350 km while exhibiting a very gradual decline in altitude over time. Hopefully, it will remain functional and in orbit for two to six months. The "best estimate" is approximately 3.5 months based upon an analysis conducted by NASA's ISS and Trajectory Planning Team as part of their review of how ARISSat-1 should be deployed by the International Space Station.

How Do I "Work" the New Bird?

ARISSat-1 is a unique Amateur Radio experiment and offers many new and exciting features. Among them, the on-board SDX system allows hams to speak with one another via satellite using CW and SSB. And unlike FM repeater satellites that can only support one conversation at a time, the

NASA



SuitSat-1 all dressed up for its flight into space.

TONY MONTEIRO, AA2TX

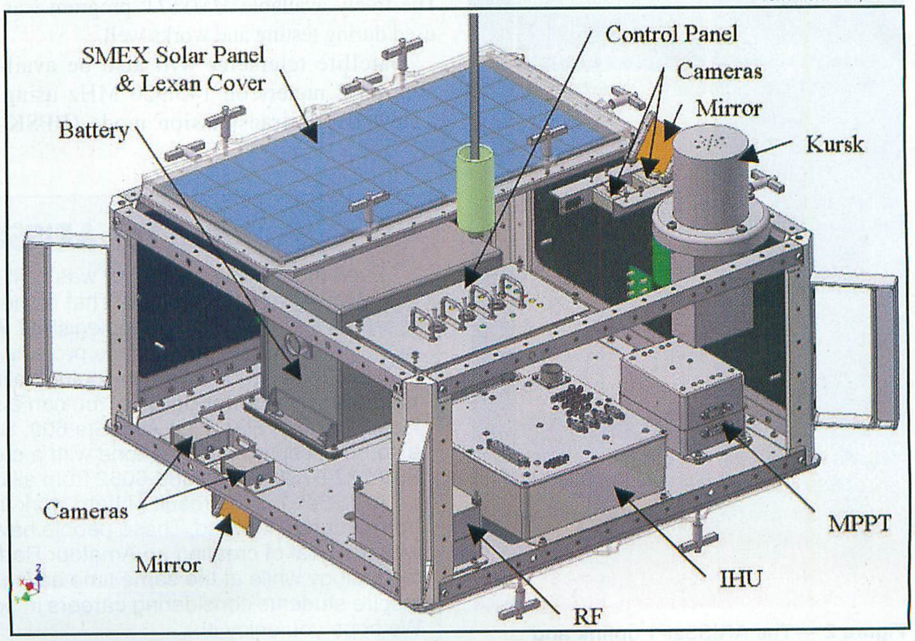


Figure 1 — An internal view of the ARISSat-1 modules housed in a space frame.

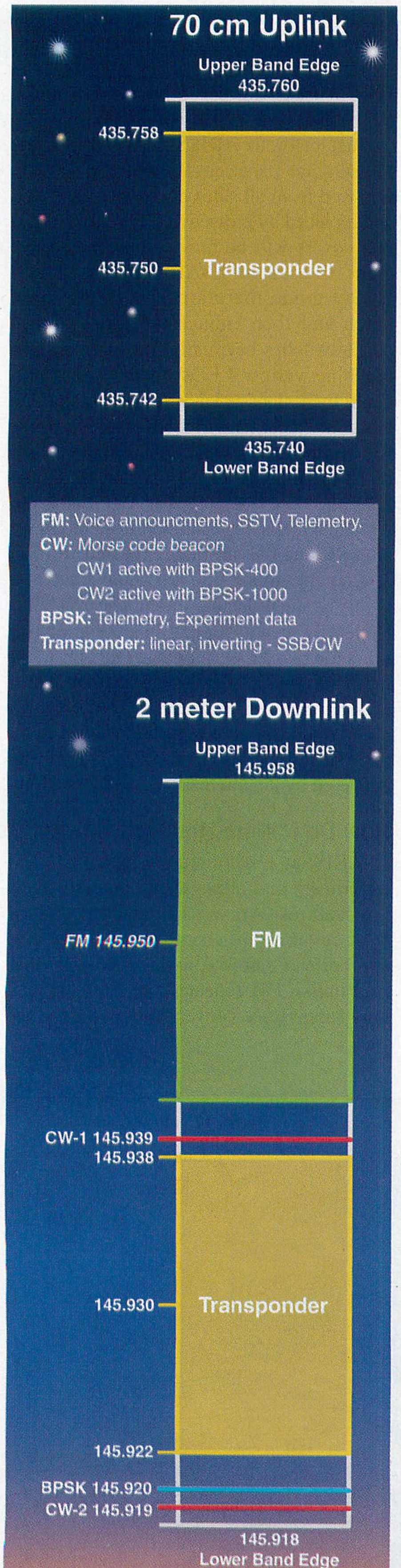


Figure 2 — The ARISSat-1 uplink and downlink bandplan.

Classroom Applications for ARISSat-1

Innovative teachers can find many ways to integrate communication with ARISSat into their Science, Technology, Engineering and Mathematics (STEM) curriculum. Topics might include the satellite's characteristics, capabilities and transmitted information. Whether it's constructing a classroom ground station to receive and track ARISSat-1, demonstrating RF frequency shift due to the Doppler effect, creating a project to track the daily changes in the health of the satellite by recording and decoding telemetry, or the opportunity to introduce students to Morse code through collecting call signs in the CW contest, the only limit is the teacher's imagination.

ARISSat-1 transponder can relay several conversations *simultaneously*.

Digital enthusiasts will enjoy the challenge of receiving BPSK-1000 telemetry. This will be an exciting application of a digital mode that is entirely new to Amateur Radio.

All transmissions from ARISSat-1 will be sent within the 2 meter satellite band using various modes of operation (see Figure 2). All stored voice transmissions will use FM at 145.950 MHz. This includes a female voice ID, female voice telemetry (subset) and 24 greeting messages in 15 different languages from students and other individuals from around the world. Many of these messages end with a "secret word." If listeners successfully collect and identify all the secret words, they can submit them and receive a special certificate. Details for the contest will be announced via the AMSAT and ARISSat-1 Web pages as well as the AMSAT News Service before deployment. To add even more variety, SSTV ID images and images acquired by the on-board cameras are slated to be part of the transmission sequence. The SSTV signals will be sent in Robot 36 protocol and can be displayed in real time on a computer using free downloadable software from the Internet. The freely available *MMSSTV* program was used during testing and works well.

Satellite telemetry will also be available on 2 meters on 145.920 MHz using a new BPSK transmission mode (BPSK-1000) specifically developed for ARISSat-1 by Phil Karn, KA9Q. There is a Phase 3, 400 bps BPSK telemetry downlink on board for use only as a backup system. Only the BPSK-1000 *system* is slated for activation. Even though the satellite will *not* be spin stabilized, the BPSK-1000 mode should provide an error-free signal allowing the use of simple antennas on the ground even during deep fades. Free decoding software for BPSK-1000 will also be made available before deployment. In addition, a downlink telemetry subset will be transmitted using CW (Morse code). If CW is transmitting on 145.919 MHz, then BPSK-1000 is active on 145.920 MHz. If CW is transmitting on 145.939 MHz, the BPSK-400 backup mode is active on 145.920 MHz.

To encourage an interest in using Morse code, the CW beacons will also host a CW contest, transmitting the call signs of hams that have contributed to Amateur Radio in space. Received call signs may be submitted for a special CW certificate. Again, specific details for the contest will be announced before deployment.

As noted earlier, for those so equipped, ARISSat-1 will feature the first software defined transponder (SDX) to fly on board an Amateur Radio satellite. A software-defined transponder creates the modulation schemes (FM, CW, USB, LSB) used by the uplink and downlinks in software through digital signal

Funding for ARISSat-1 — You Can Help!

Even though the spacecraft was built by volunteers, the ARISSat project has cost AMSAT about $180,000. That money has come from AMSAT's financial reserves that now must be replenished. AMSAT also needs funding to provide seed money for work on future space projects.

You can support AMSAT by going online and making a donation at **www.amsat-na.com/store/donation.php**. You can also send donations by *mail to AMSAT* Headquarters, 850 Sligo Ave, Ste 600, Silver Spring, MD 20910. Donations can also be accepted via telephone with a credit card from the USA and Canada at 888-322-6728 or 301-589-6062 from all other locations.

ARISSat-1 is the result of hard work from a large group of talented individuals from around the world. These people have freely volunteered their time and talents with the goal of creating an Amateur Radio satellite that demonstrates the latest technology while at the same time acting as an educational tool for teachers to inspire students considering careers in science, technology, math and engineering. We hope you enjoy it!

What Do I Need to Work or Hear ARISSat-1?

The only radio you need to receive the voice ID, voice telemetry, SSTV and voice messages is a 2 meter FM transceiver (even a handheld radio), or a scanner that covers 2 meters.

If you are fortunate enough to own a multiband or multimode 2 meter/70 cm rig such as the Yaseu FT-817, Kenwood TS-2000, ICOM IC-910 or an older Yaesu FT-847 or FT-736R, you already have what you need to monitor and/or access ARISSat-1. If this isn't the case, here are some options to consider.

Since ARISSat-1 is a multimode satellite, it would be best to have a single radio that covers 2 meters and can receive SSB, CW and FM modes. The Yaseu FT-817 is a good choice since it also features digital and packet modes and has positions for narrower CW filters to improve reception. This rig, while being a good performer, can be expensive if purchased new, however.

Another approach is to use one of several multiband/multimode scanners available that cover the 2 meter band. Pricing for these units starts at around $300 to $500. Hamfests (or online auction sites such as eBay) are also good sources for used radios such as the ICOM R-10 and Yaesu VR-500. With a little searching, these radios can often be found in good condition and at bargain prices.

If you only plan to listen to the FM audio, a common "scanner" should be adequate provided it covers the 2 meter band and you have a decent outside antenna. By that, I'm suggesting that you at least use a ¼ wave vertical or circularly polarized antenna. This is discussed in more detail in the section on antennas. Of course, you'll only be able to tap into the downlinked voice and SSTV signals with these rigs.

The RF output of ARISSat-1 varies according to the mode in use…

250 mW: FM audio (including SSTV)
100 mW: BPSK-1000 beacon
25 mW: CW beacon
125 mW: SDX transponder

This may not seem like much power, but consider the height of the antenna!

The multimode receivers I've discussed will also allow you to receive downlink signals from the SSB/CW SDX transponder. If you want to uplink to the transponder, however, you will need a 70 cm SSB transmitter.

Antennas

When it comes to antennas, a ¼ wave vertical antenna should work well even without a preamp as long as you minimize your coax cable loss by keeping its length less than 25 feet. If a longer cable run is necessary, you might want to move up to the lower loss, LMR-400 solid center conductor coax. A ⅝ wave or larger "gain" vertical will be worse since all the gain is toward the horizon. Because the orbiting satellite will be in a slow, random tumble, the listener will probably experience some fading caused by *spin modulation*.

To minimize this problem, you may want to consider building a simple circularly polarized antenna such as the K5OE "Texas Potato Masher." You'll find instructions on the Web at **victrolla.homeip.net/wo5s/junkpile/432/tpm2.pdf**. Another approach is to purchase and install the 2 meter Eggbeater, model EB144, from M^2 Antenna Systems (**www.m2inc.com**). The popular handheld Arrow beam antennas (**www.arrowantennas.com**) should also work well. Note that the 70 cm versions of all of the antennas I've mentioned here will also allow you to uplink to the SDX transponder. I recommend the 70 cm antenna be separated from the receive antenna by at least 8 feet to avoid crosstalk.

Computers and Software

As I've said, free software for demodulating and decoding the BPSK-1000 via a computer sound card will be made available before ARISSat-1's deployment. You'll also need software to receive and display the SSTV images from ARISSat-1. *MMSSTV* performed well for this purpose during the craft's development. *MMSSTV* can be downloaded free of charge at **mmhamsoft.amateur-radio.ca/pages/mmsstv.php**. Mac users have several options to decode SSTV signals. The Mac SSTV program *Multiscan* is available free at **web.me.com/kd6cji/MacSSTV/*MultiScan*.html**. Another option is to download *Coca Modem*, which is also a free at **homepage.mac.com/chen/w7ay/cocoaModem/index.html**.

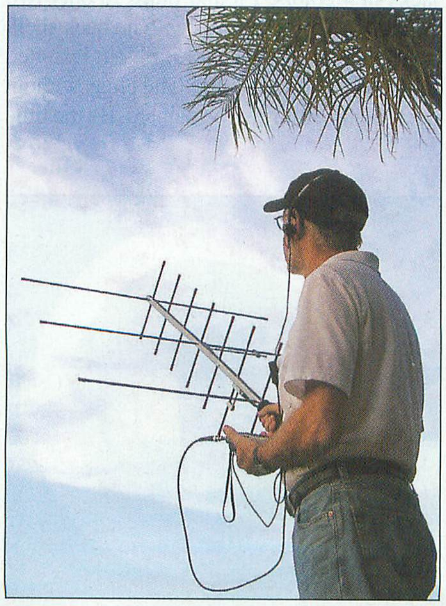
To know when ARISSat-1 passes over your location, you'll need to track it. You'll find tracking software such as *SATPC32* available from AMSAT at **www.amsat.org** (all proceeds from the sales are donated to AMSAT). The AMSAT Web site also hosts a satellite pass prediction program for your convenience (**www.amsat.org/amsat-new/tools/predict/**). Mac users can purchase the *MacDoppler* tracking program at **www.dogparksoftware.com/Macintosh_Amateur_Radio_Pr.html**.

processing rather than analog circuits. Using a linear, inverting SSB/CW transponder in U/V mode (UHF uplink/VHF downlink) sporting a 16 kHz bandwidth, hams will be able to chat with each other whenever the satellite is within range. The uplink window is from 435.742 MHz to 435.758 MHz with a downlink window of 145.922 MHz to 145.938 MHz.

Another exciting aspect of the ARISSat-1 mission is called "Fly a File." The ARISS team has accepted digitized submissions of space and science related images as well as information about various science projects from students worldwide. These have all been loaded onto a memory chip that was attached to the inside of the spacecraft prior to shipment and will now be flown as part of the mission. The submissions will not be accessible from space, but rather, are posted for viewing at the ARISS Europe Web site at **www.ariss-eu.org/arissat-1.htm**.

As you can see, ARISSat-1 is a satellite that offers something for everyone. It will offer various modes of operation for individual hams as well as providing many "hands-on" opportunities for educational applications.

DAVID JORDAN, AA4KN


A simple handheld setup like this will be more than sufficient to receive ARISSat-1.

David Jordan, AA4KN, is an Amateur Extra Class operator and has been a licensed ham for 37 years. He attended the University of Central Florida where he earned a BS in Engineering Technology and worked as a design engineer for a local Orlando firm. David spends much of his time promoting Amateur Radio in local schools and is heavily involved in both AMSAT and the ARISS program. Most recently, David served as a member of the ARISSat-1 Project Team playing a role in its development and integration. He also serves as Education Chair for the Lake Monroe Amateur Radio Society (LMARS). You can contact David at 825 Hickory Hill Ct, Orlando, FL 32828 or by e-mail at **aa4kn@amsat.org**.

VOTE **Did you enjoy this article? Cast your vote at:**
www.arrl.org/cover-plaque-poll