Scotty Cowling, WA2DFI

P.O. Box 26843, Tempe, AZ 85285; **scotty@tonks.com**

Hands-On-SDR

This sixth installment revives the author's series.

It hardly seems possible that it has been three and a half years since my last installment. Time marches on, as they say, and we have exciting new advances in the SDR field to discuss! The last few years have seen a virtual explosion of new SDR hardware available to us: from SDRPlay **[1]** and RTL-DVB **[2]** dongle to Red Pitaya **[3]**, KiwiSDR **[4]**, Kerberos **[5]** and the several variants of LimeSDRs **[6]**. This list is hardly exhaustive, as there are so many from which to choose!

In spite of all the new SDRs on the market, it still seems difficult to find that one SDR that has all the features and capabilities that you want. Some are very inexpensive, but leave you wanting more performance. Some of the more expensive ones leave you wondering why they didn't offer some critical (to you, at least) feature, like a low phase-noise oscillator, for example. In this installment, I will introduce you to a new TAPR **[7]** project, called the TangerineSDR, that will attempt the impossible: to be all things to all users. We will try, at least!

The inspiration for TangerineSDR came from the HamSCI **[8]** group as a request for SDR hardware that could be used as a Personal Space Weather Station (PSWS). Their need resulted in the four-hour Sunday seminar at the TAPR Digital Communications Conference in Albuquerque, NM, on September 16, 2018. At TAPR, our first response was, "Let's find a commercial SDR that we can incorporate into a PSWS kit". Existing hardware would be our best bet for a quick solution. After some research and further consultation with the scientists at HamSCI, it became clear that there was no affordable (less than US $500) solution that met PSWS requirements. What are the requirements, you ask? Wait, not so fast…

What Do We Need to Get Started?

As with each of these columns, I always try to define what you need in the way of knowledge and equipment to get the most out of the "Hands On SDR experience".

I will assume a basic knowledge of direct sampling (DDC and DUC) SDR techniques, as well as some familiarity with the Single Board Computers (SBCs) of the day. For this initial introduction to the TangerineSDR, that is about all you need to bring to the party, except your interest! Future columns will dig deeper into the details. As the new hardware becomes available and firmware and software applications are written, it will be advantageous for you to obtain hardware. But for now, let's dig into the TangerineSDR features and see if it can deliver on our wild claims.

A Bit of History

Back in 2006, TAPR became involved in the production of boards for the openHPSDR project. As one of the participants in that project on both sides (TAPR as a facilitator and the openHPSDR group as a contributor), I think it was an unqualified success! For its day, openHPSDR was a state-of-the-art design, produced a working radio and contributed to a successful commercial SDR company. OpenHPSDR pre-dated FlexRadio System's highly successful direct sampling SDRs; it also pre-dated crowdfunding as a fund-raising method for building expensive hardware. How many of you reading this today sent your money to TAPR for an openHPSDR board before any hardware was even produced? TAPR thanks you for your trust! But I digress. TangerineSDR follows in the footsteps of openHPSDR. We hope to learn from the shortcomings of openHPSDR (yes, there were a few) and expand upon the good ideas that permeated that design.

TangerineSDR will reduce system complexity by using one central FPGA instead of multiple FPGAs distributed across several boards. TangerineSDR will increase performance by eliminating the narrow, slow, unterminated Atlas bus. TangerineSDR will provide high-bandwidth paths between ADC/DAC and the FPGA, as well as high-speed communications channels from the FPGA to the outside world. TangerineSDR will be modular, allowing you to customize the performance (and hence, the cost) of the hardware to a specific task, PSWS being only one of many use cases. Modularity is the key to being all things to all users! I am using this phrase so often, we should use an abbreviation for it: ATAU **[9]**.

Basic Requirements

The PSWS is the initial use case, but there are others that I will cover at the end of this installment. Many of the basic features will be useful to other users, and some will not. Here are the basic PSWS system requirements:

- Dual-channel, synchronous 14-bit direct sampling DDC receiver to 60 MHz
- Eight 192 kHz virtual receiver data streams
- Selectable attenuation and optional pluggable RF filtering to reduce overload
- Switchable on-board noise source for amplitude calibration

- Very accurate time stamping, to within 100 ns
- Accurate frequency accuracy, less than 30 ppb frequency error
- Gigabit Ethernet interface
- Networked receivers capable of secure communications with a global server
- Able to continuously store 24 hours of sample data history in a ring buffer
- 3-axis magnetometer with ~10 nT resolution at 1 sample/second.

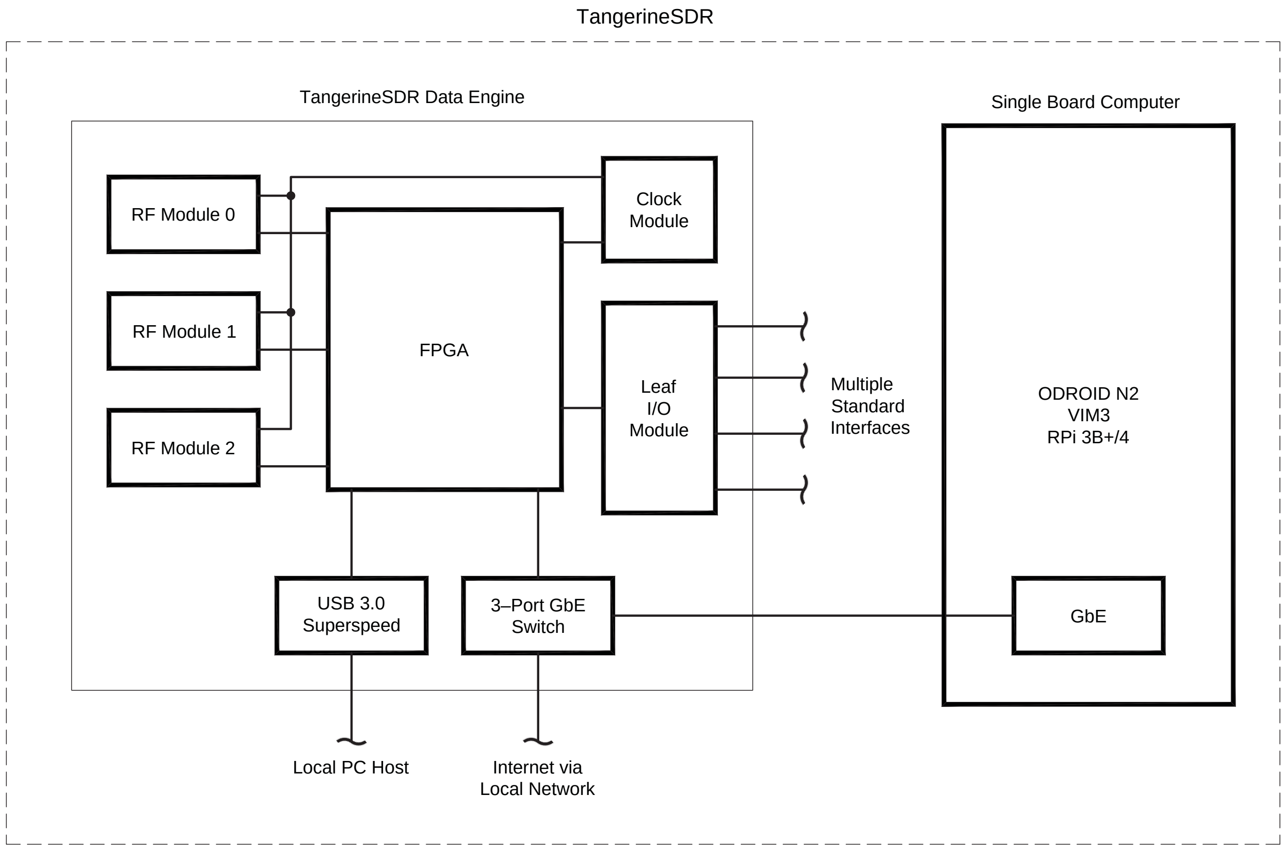
We can meet most of these requirements with a USRP **[10]** system, but the cost is high (typically more than US $2000). Two things to note in the above list are the accurate time stamping, which will require a GPS Disciplined Oscillator (GPSDO), and a magnetometer with enough sensitivity to be useful. Neither of these requirements can be met with inexpensive off-the-shelf components. Our solution will be to "roll our own" in an effort to reduce cost and obtain just the right cost/performance ratio that we require.

Pairing with Single Board Computers

The TangerineSDR block diagram is shown in **Figure 1**. One of the unique features of this SDR is the pairing of the SDR-specific hardware (the Data Engine, or DE, and its associated boards) with a computer. We considered using an SoC FPGA in place of the logic-only FPGA and a general-purpose Single Board Computer (SBC), such as a Raspberry Pi or Odroid N2. The SoC FPGA contains a dual-core ARM CPU alongside the FPGA logic on the same silicon die. There are a few problems with the SoC approach. First it is much more expensive, adding more than US $100 to the cost of the DE. A RPi 4 SBC costs US $35-$55, and easily out-performs the dual-core ARM system on the SoC FPGA. The SoC approach also flies in the face of our ability to match required performance to modular hardware. The SoC-based DE might work very well in a simple system with a light CPU load, but if you need more CPU power, you will be locked into the SoC CPU limitations. On the other hand, if we pair with an external computer, we can pick a low-cost, low-performance SBC for "easy" tasks or substitute a full-size i9 desktop system when massive computational power is needed. Using the external computer will also allow us to take advantage of the constant upward trend in capability and downward trend of size, cost and power consumption of tomorrow's SBCs.

High-speed Sample Paths

A closer look at Figure 1 shows the high-speed paths between the three RF Modules along the left side of the DE and the FPGA in the center of the DE. These paths consist of three Low Voltage Differential Signaling (LVDS) channels for each RF Module: two 17-bit input ports to the FPGA and one 14-bit output port for RF Modules 0 and 1, and a more restricted set of pins for RF Module 2. (RF Module 2 will be used mostly as a debug port, so we will wait until



QX2001-Cowling01

Figure 1 — TangerineSDR system block diagram.

a future installment to describe its use.) The two input ports are used for digitized receive data from an Analog-to-Digital Converter (ADC) and the output port is used to send transmit data to a Digital-to-Analog Converter (DAC). Both ADC and DAC circuits will reside on the RF Modules. Each of these three data paths is capable of nearly 500 MByte/s transfer rate for each RF Module. None save the most expensive SDR models provide this data rate between the antenna and the FPGA. Also keep in mind that all of the signals in the three paths connect to programmable pins on the FPGA. This means that any pins that are not used for data communications between the FPGA and an ADC or DAC can be re-programmed and used for other purposes, including single-ended connections. Since there are 96 of these signals (2 buses X 2 pins/pair X 17 pairs, plus 1 bus X 2 pins/pair X 14 pairs = 96 pins) per module, or 192 pins total, there are plenty of pins available between the FPGA and the RF Modules.

High-speed Communications Interfaces

Okay, so we have buckets of bytes arriving at the input to the FPGA, what are we going to do with all this data? The typical approach is to reduce the data rate using a mathematical process known as decimation. By performing decimation on a stream of data we do, in fact, reduce the data rate. But at the same time we reduce the information content of the stream. It may seem obvious that (for example) a 122.88 Msps data stream contains more data than a 192 ksps data stream, but how we derive the lower-rate stream from the higher-rate stream is not so obvious. We will leave this derivation to a later installment, but we will discuss the reason why we need to reduce the data rate from the relatively speedy data streams that we took so much care to create into the FPGA.

Consider the above example of 16-bit sample data arriving at the FPGA at 122.88 Msps. That is two bytes (16 bits) per sample or 245.75 Mbyte/s. At 8 bits per byte, this is a bit rate of nearly 2 Gbit/s (1.96608 Gbit/s, to be precise). Comparing this bit rate to a few common communications interfaces, Gigabit Ethernet (at 1 Gbit/s), Super-speed USB 3.0 (1 lane at 5 Gbit/s) and High-speed USB 2.0 (at 480 Mbit/s) we can see why the communications interfaces are just as important as the sample paths to/from the FPGA on the RF side. Note that even GbE will require some reduction of the input data rate. And here is a thought to ponder: it is not likely that our SBC will be able to keep up with the demodulation and filtering (DSP tasks traditionally performed by the PC in an SDR system) at full Gigabit Ethernet speeds. The characteristics of Ethernet networking will allow the FPGA to split the high-speed input stream into many smaller (decimated) output streams that can be directed at multiple SBCs (or more powerful PCs). The higher the communications speed, the more streams we can accommodate. Remember, there are two very high-speed input streams from each RF Module. You have heard of Multiple Input, Multiple Output, or MIMO **[11]** for transmitters and receivers? Well, this is MIMO for data streams. Most SDRs today provide only one communications interface, often times at a much lower throughput than on the RF side, limiting the width or number of output streams. TangerineSDR will provide both GbE and SuperSpeed USB 3.0 at 5 Gbit/s communications interfaces. Future DE implementations will likely migrate to 10 Gigabit Ethernet (10GE) or 40 Gigabit Ethernet (40GE) or future higher-speed versions of USB. So, while we cannot eliminate the restriction of the output data rate, we can at least minimize it by implementing multiple high-speed communications channels to the outside world.

The communications and GPIO data paths are shown in **Figure 2**. The bold lines are high-speed (LVDS and LVCMOS parallel, typically greater than 10 MHz or high-speed communications) paths, and the thin lines are low-speed paths (GPIO, I2C, SPI and UART serial interfaces, typically up to a few MHz). Although it might not be obvious from the figure, there are two GbE ports on the DE. They are bridged by a three-port GbE switch to the FPGA. This arrangement allows the DE and SBC to communicate with the external network, as well as with each other, over the same Ethernet wire. The protocol is designed to

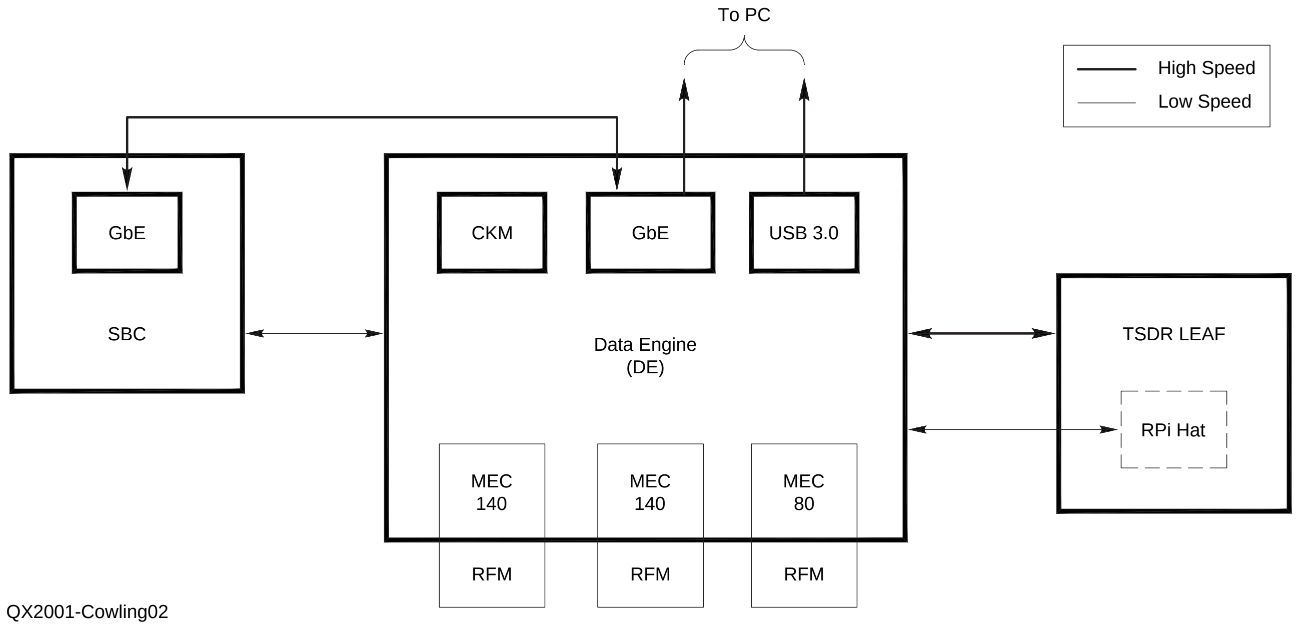


Figure 2 —TangerineSDR communications and GPIO data paths.

protect the DE to SBC path from outside traffic. Authentication is performed by the SBC, simplifying the FPGA system design.

Modularity

Referring again to Figure 2, the GbE and USB 3.0 circuitry are part of the DE, while the Clock Module (CKM) and the three RF Modules (RFMs) are pluggable options. Even though the LEAF Module is shown alongside the DE, it is also a pluggable module.

The LEAF Module is somewhat unique in its design, and part of the ATAU philosophy. LEAF stands for Low-speed Expansion Adapter Fixture, along the lines of the Raspberry Pi Hat (Hardware Attached on Top), the Arduino Shield (arbitrary name appropriation as far as I can tell) and the Beagle Bone Cape (due to its cape-like shape). To fit in with the crowd, we need a catchy acronym, and believe it or not, it fits all of the terms in our description: it has Low-speed I/O connections, it Expands these connections to other modules (ironically, to Shield and Cape modules), Adapts I/O to other connectors (like Click **[12]** and Ultra96 Boards **[13]**) and acts as a Fixture for adding other components. You can even use a standard Raspberry Pi Hat board in place of the LEAF Module (see **Figure 3**.) What distinguishes the LEAF Module from the Hat is the additional high-speed I/O expansion connector on the edge opposite the RPi low-speed expansion connector. It is a high-speed M.2 connector (formerly NGFF, or Next Generation Form Factor) of which you may already be

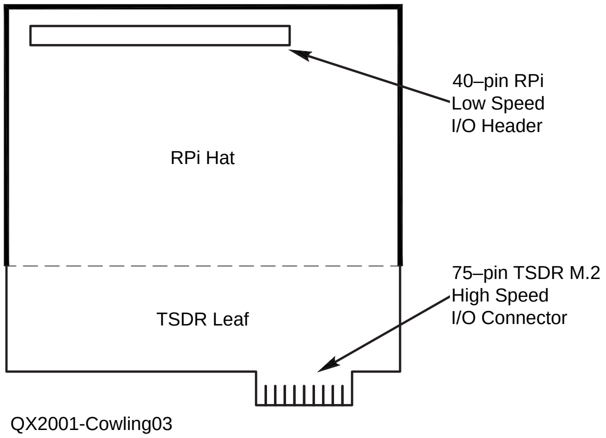


Figure 3 — TangerineSDR LEAF *vs*. RPi HAT I/O expansion boards.

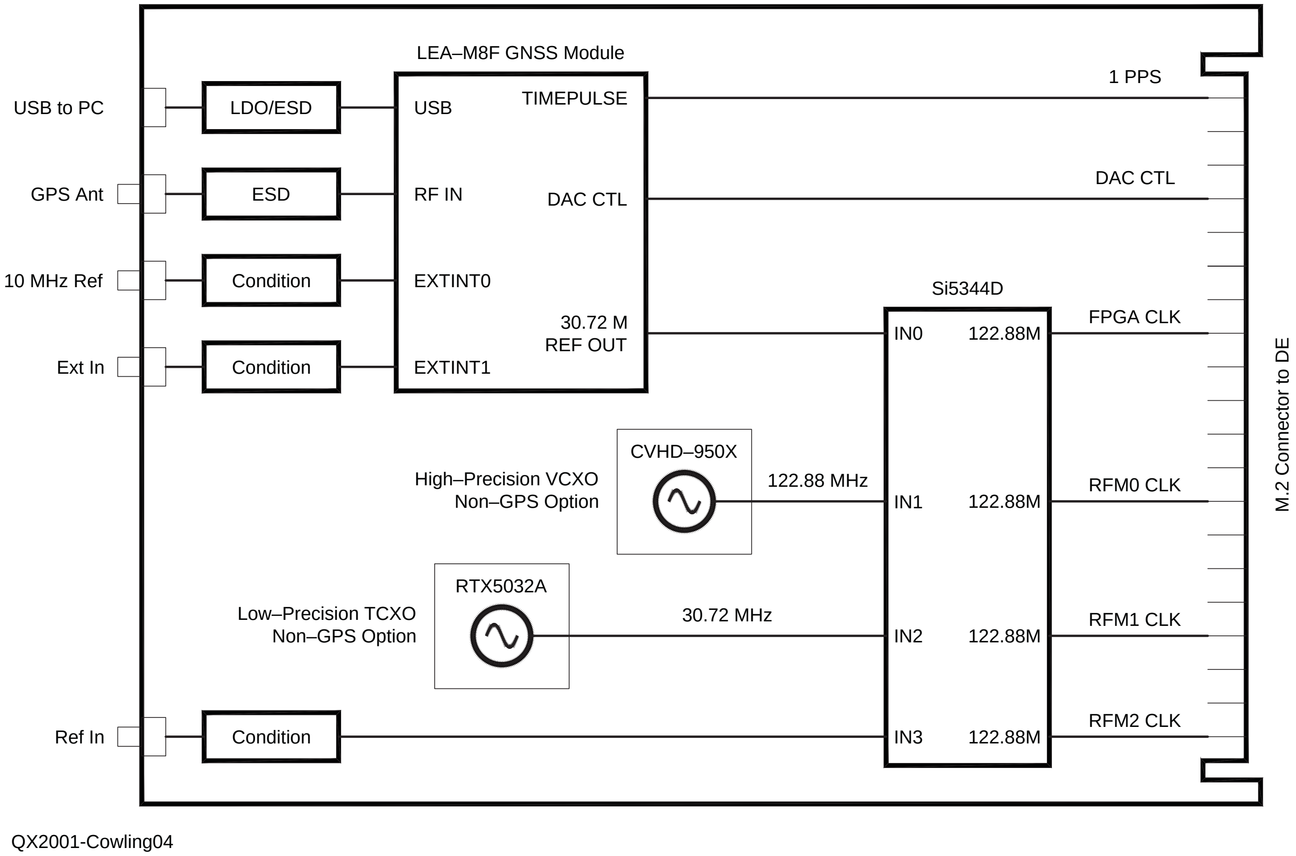


Figure 4 — TangerineSDR GPSDO clock module (CKM).

familiar: it is used for PCIe (PCI Express), NVMe (non-volatile memory express) and SATA (Serial ATA) solid state drives (SSDs). This allows LEAF Module designs to interface to higher speed devices, such as the Xilinx Ultra96 Board, which has a high-speed I/O expansion port. Many LEAF boards are planned, including a version that has Arduino Shield connectors, one with Cape connectors, one with a Click interface, and others.

Clocking

The Clock Module (along with the RF Modules, below) is probably the most performance-defining piece of the TangerineSDR design; see **Figure 4**. It will also have a significant cost impact in its higher-performance versions. In actuality, the lowest performance TangerineSDR will not have a Clock Module at all! The “standard” clock oscillator will be located on the DE board, and will be selected as a reasonable tradeoff between performance and cost. Higher performance (lower phase noise, higher frequency accuracy, oven stabilized, etc.) oscillators are under development even as we work on the DE and RF Module designs. The first TangerineSDR use case is a very demanding one: the PSWS. The PSWS requires very accurate frequency stability as well as a very accurate data arrival time stamp, down to 100ns (or at least as close as we can afford to get to that accuracy). Since this will require a GPS to stabilize, or discipline, the oscillator (hence the name GPS Disciplined Oscillator, or GPSDO), this Clock Module will be expensive. For others, who may not need the performance (and expense) of the GPSDO Clock Module, but desire better performance than the standard on-board DE oscillator, we will build an intermediate-performance Clock Module that will focus more on low phase-noise rather than highly accurate time stamping. Something similar to the Crystek CVHD-950 used on the openHPSDR Mercury receiver is a possibility.

RF Modules

While there can (and likely will) be many types of RF Modules, a preliminary block diagram of the first variant is shown in **Figure 5**. The details may change somewhat as we proceed down the design path, but the basic architecture will remain as shown. For example, the two 14-bit data paths from the ADCs to the 140-pin MEC connector may be differential, instead of the single-ended paths shown. The features of this RF Module are driven by the requirements of the PSWS. Note the on-board noise source

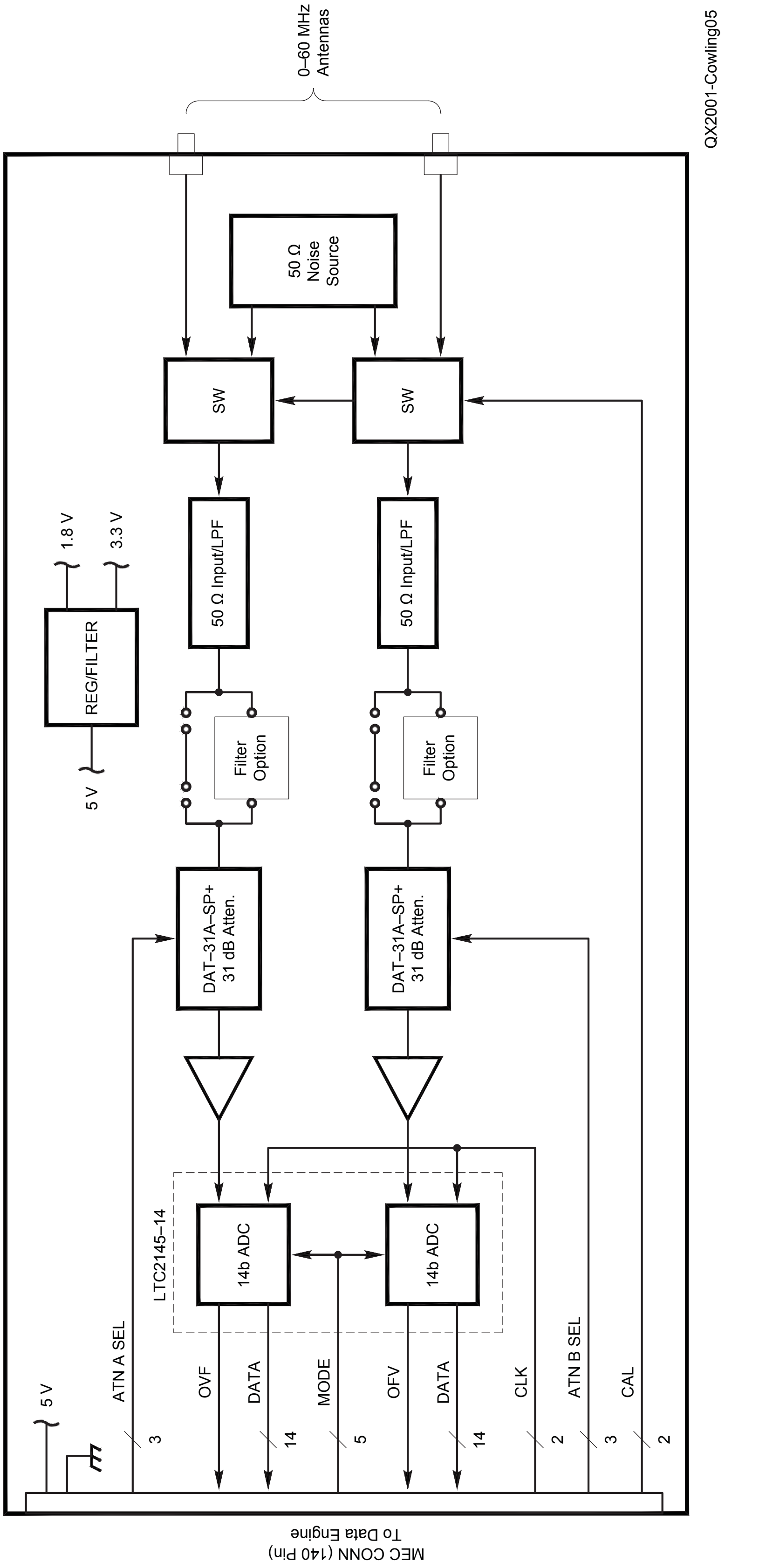


Figure 5 — TangerineSDR Dual-Channel 0 – 60 MHz RF Module (RFM).

that can be switched onto the receive path for calibration of each receive channel. Also of note are the optional in-line filters, one for each channel. These filters are normally bypassed by jumpers, but you can remove the jumpers and install filter modules to adapt the front-end response to whatever local situation might arise. Live near a local high-power MW AM station? Add a notch filter to take its signal down to a manageable (i.e., non-ADC-saturating) level. Live in between two or more high-power MW AM stations? Add a high-pass filter to eliminate overload from anywhere within the whole MW broadcast band. We will provide power to the filter mounting headers, so you can even put active circuitry on the filter boards! The 31-dB step attenuator may be replaced by a three-step 0-10-20 dB passive attenuator in order to keep as few active components as possible ahead of the ADC. The clock for the ADCs comes from the DE, and its characteristics will depend on the type of Clock Module installed. This first RF Module will be designed to clock at the ever-popular frequency of 122.88 MHz. Just in case you are still wondering why we use such an odd frequency, it happens to be an integer multiple of 48 ksps (2,560 times 48,000 to be exact), which makes our common output sample rates of 48 ksps, 96 ksps, 192 ksps, 384 ksps and 768 ksps easier to generate.

Use Cases

Hopefully I have given you a reasonable introduction to the hardware in this installment. I fully intend to fill in the gaps in this description (there are many) in much more detail as the project unfolds. Since the project's inception, many individuals have come forward (some at my direct request, and others by word of mouth) with ideas for how they could put a TangerineSDR to use. I call these our use cases, and the idea is to make sure to include as many features as we can so as not to overlook any important ones. Some use cases are simple (listen to SSB on my favorite 40 m frequency), while others are more difficult (PSWS). Following is a partial list of TangerineSDR use cases. If you have others, please contact me. ATAU!

- PSWS (of course)
- Satellite Ground Station (requires new RF Modules)
- High Performance HF transceiver
- WSPRnet/RBN on multiple bands simultaneously
- HF noise sniffing/calibrated receiver
- Remotely controlled stations
- Radio Astronomy (Project Jove **[14]**, SARA **[15]** pulsar detection)
- Academic Learning.

What's Next?

Remember that the TangerineSDR project is open source, and there are many use cases for the hardware. There are opportunities for FPGA programmers, applications software developers, GUI designers, and yes, even hardware designers for the next generation. Each DE board variant will have an on-board FPGA and Verilog code to match. Each new RFM will need an RF expert to oversee its development. I will be covering the evolution of the hardware, firmware and software in subsequent columns. Hardware will start to become available this spring with the DE and HF RF Module, followed closely by the Clock Module. I expect a long and evolutionary (hopefully revolutionary, too!) lifetime for TangerineSDR. OpenHPSDR started in 2006, nearly 14 years ago now. It will be interesting to see where TangerineSDR development is in the year 2034!

As always, please drop me an email if you have any suggestions for topics you would like to see covered in future Hands-On-SDR columns or even just to let me know whether or not you found this discussion useful.

Notes

[1] SDRPlay: **sdrplay.com**.

[2] RTL-DVB Dongle: **rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles**.

[3] Red Pitaya: **redpitaya.com**.

[4] KiwiSDR: **kiwisdr.com**.

[5] Kerberos SDR: **othernet.is/products/kerberossdr-4x-coherent-rtl-sdr**.

[6] LimeSDR: **limesdr.com**.

[7] Tucson Amateur Packet Radio, or TAPR is the creator of the TNC1, TNC2 and numerous other advances in the art of digital radio. SDR is the logical continuation of TAPR's effort to advance the state of the art in digital radio. Visit **tapr.org** for more information.

[8] Ham Radio Science Citizen Investigation, or HamSCI, is a group of hams and scientists (many are both) dedicated to advancing scientific research through amateur radio activities. Visit **hamsci.org** for more information.

[9] A Google search turned up nothing for "ATAU" or "All things to All Users," so I will stake the claim of ownership to the abbreviation "ATAU."

[10] Universal Software Radio Peripheral: **ni.com/en-us/shop/select/usrp-software-defined-radio-device**.

[11] Multiple Input Multiple Output (MIMO) is a method for multiplying the capacity of a radio link using multiple transmission and receiving antennas: **en.wikipedia.org/wiki/MIMO**.

[12] Click is a small I/O module that uses mikroBUS™: **mikroe.com/mikrobus**.

[13] Ultra96 MPSoC FPGA development board: **96boards.org/product/ultra96**.

[14] Project Jove studies radio emissions from the planet Jupiter, the Sun and our galaxy: **radiojove.gsfc.nasa.gov**.

[15] Society of Amateur Radio Astronomers (SARA): **radio-astronomy.org**.