Software Defined Radios (SDR) for NASA Spaceflight Applications

James Lux, P.E.
Jet Propulsion Laboratory, California Institute of Technology
SDR Task Manager, CoNNeCT project Co-Investigator
james.p.lux@jpl.nasa.gov
This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

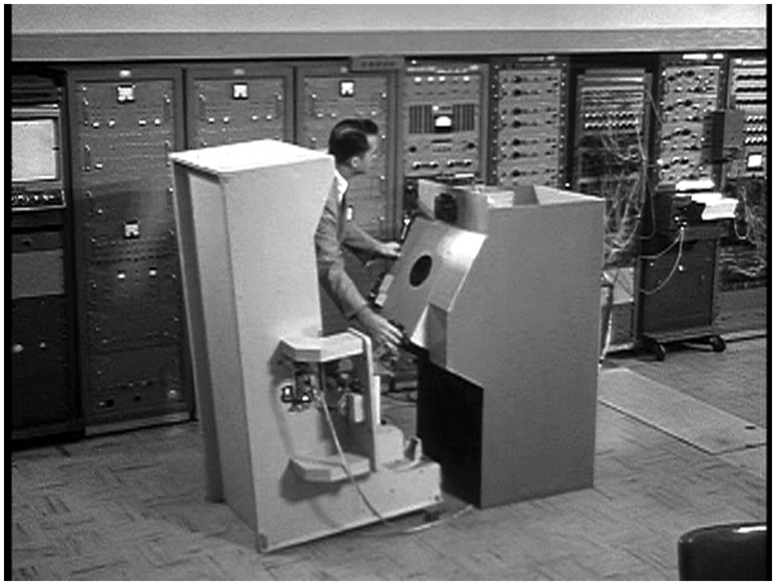
Outline

- NASA's adoption of Software Defined Radio
  - NASA's unique needs
  - NASA Space Telecommunications Radio System (STRS) Architecture Standard
  - CoNNeCT project and the SCaN Testbed on ISS
- JPL's SDRs
  - Electra on MRO, MSL, etc.
  - JPL-SDR for CoNNecT
- JPL-SDR system architecture
  - Hardware platform
  - Software Infrastructure (the STRS Operating Environment)
  - Waveforms and Applications
- Development Approach
  - Testbeds, configuration management, build tool-chain
- Future Plans
  - CoNNeCT Experiments

What’s a Software Defined Radio

- A SDR is a radio where the function is substantially determined by software
  - *and where that function is changeable*
- There are several flavors of SDR
  - Software controls an analog radio
    - Microprocessor front panel to turn knobs and flip switches
    - Not really a SDR – more a “software controlled radio”
  - Fixed function radio that uses Digital Signal Processing
    - Replacing analog components with digital equivalents
  - Radios that are reprogrammable
    - Function can be changed by loading new software
  - Radios that program themselves
    - Cognitive Radios – a topic for the future

Radio + Computer not always = SDR

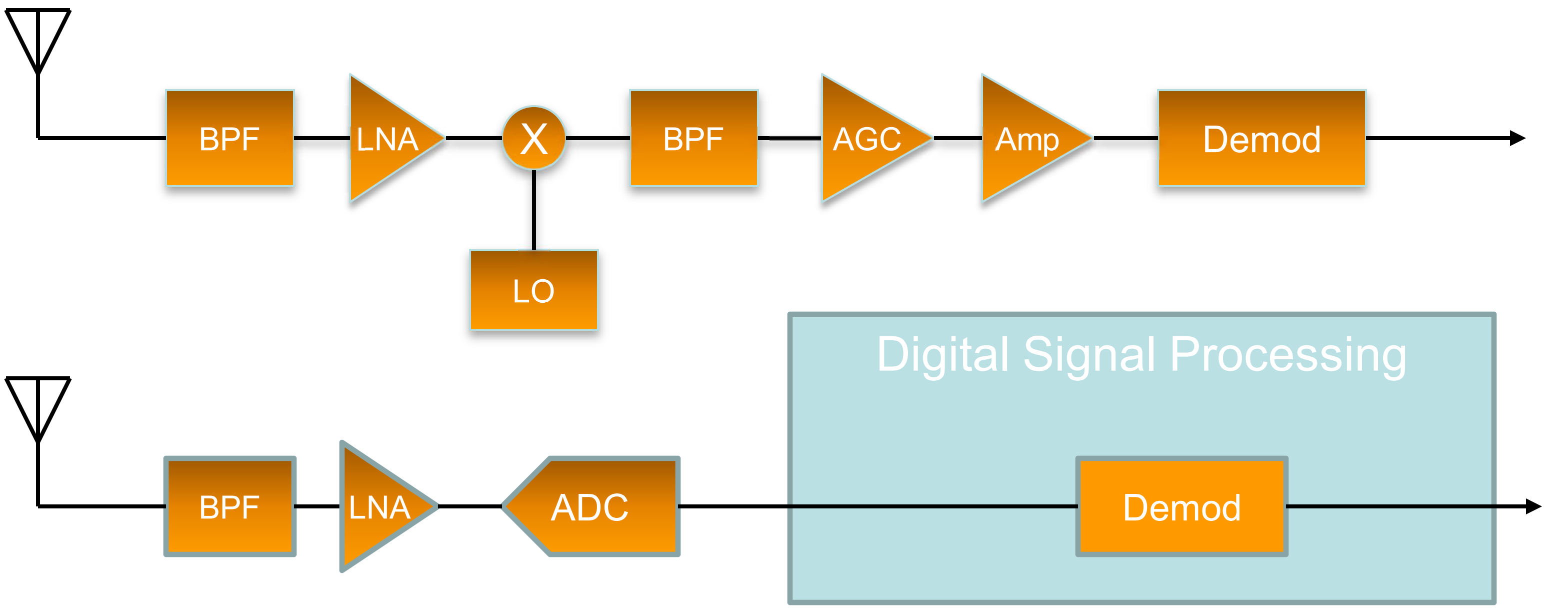


Analog computer, programmed with patch cables, for Project Mercury

JPL Computers from 1953, used to process telemetry data

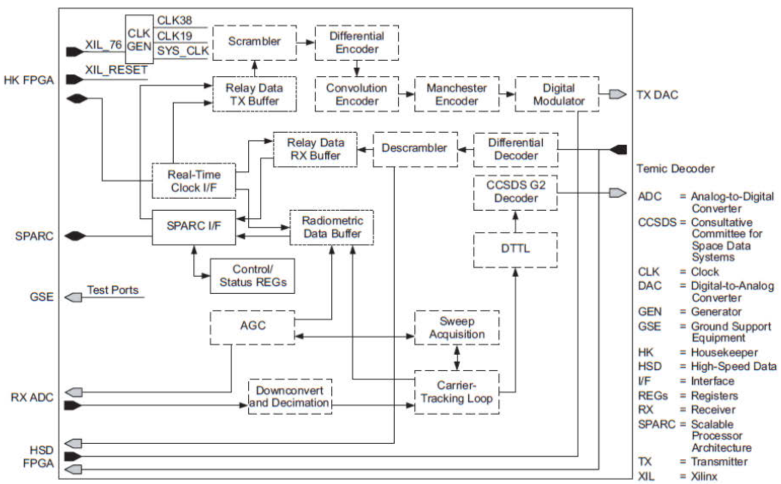
They're certainly reconfigurable, but SDR generally implies use of a generally reprogrammable resource with "loadable software"

Traditional vs Software Radio



- Today's radios tend to be a hybrid between these two
  - There aren't many 24 bit, 100 Gsample/second ADCs
  - There aren't multi-TeraFLOP processors, either
- But.. After downconversion and bandlimiting, most of the processing is reprogrammable and digital
  - And it's a whole lot more complex than the simple box "Demod" above
  - Just like there's a whole lot more to an LNA than a box that says "amplifier"

An example of what' s in the DSP box



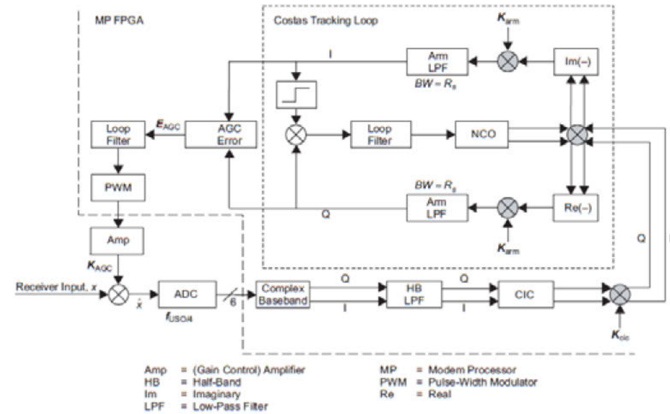


Fig. 2-2. AGC control loop.

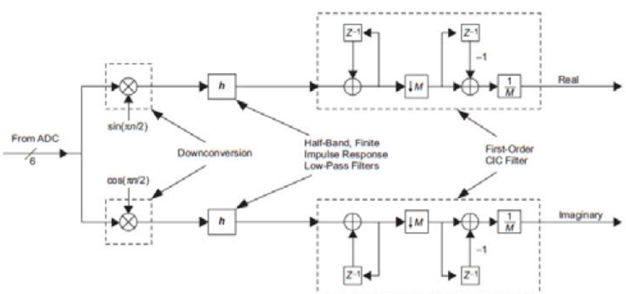


Fig. 2-3. Digital complex basebanding and decimation.

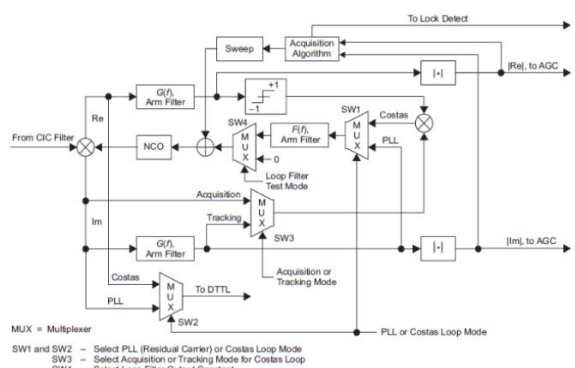


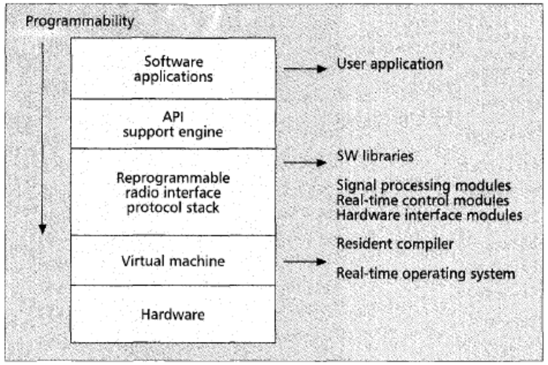
Fig. 2-4. Block diagram of the Electra carrier-tracking loop.

Figures from: Chapter 2 of Hamkins & Simon, *Autonomous Software-Defined Radio Receivers for Deep Space Applications,* DESCANSO Book Series Volume 9, JPL, 2006

And this doesn' t even get into "computer" kinds of things: file systems, file transfers, network protocols, etc.; All things expected of a modern SDR.

Bringing structure to an unruly world

- Early SDRs used ad-hoc software architectures
  - This reflects the evolution as various analog functions were migrated to DSP: the architecture was still the same
    - Down converters, demodulators, modulators, oscillators, mixers : All have analog equivalents & are often IP-cores in an FPGA
    - FPGA design is patterned after hardware design: tools developed for ASICs, design flow aimed at “tape-out”, etc.
    - Programmable DSP (TI, Analog Devices) more “software-y”; but still ad-hoc architectures and software design; much like embedded microcontrollers: one big loop, interrupt driven
  - Structure is “signal flow graph” driven
- Today’s radios need more complex functionality in their software
  - The radio is called upon to have varied functions:, many of which are not signal-flow-graph like: Networking functions, time transfer, etc.
  - Starts to look more like a computer that does radio, rather than a radio that happens to have a computer in it.
- Need something better, easier to maintain and generate
  - Maybe a layered architecture? Maybe dynamic late binding?
  - JTRS and the SCA



■ **Figure 7.** *The relationship between HW and SW in an SW radio context.*

Buracchini, “The Software Radio Concept”, IEEE Communications Magazine, Sept 2000

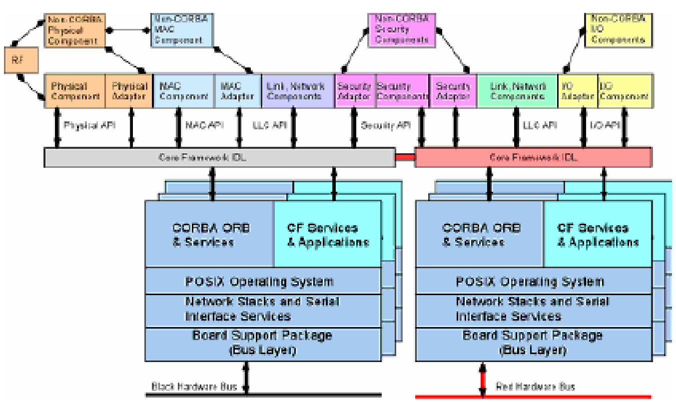


Figure 1 Original SCA Software Structure

Richardson, et al., “Evolution of the Software Communications Architecture Standard”, MILCOM 2009

And now, a word about "Flight Qualified"

- Very long development and use cycles compared to commercial world
  - Mission proposal to launch = 5-6 years
  - Hardware selection >3 years before launch
  - Cruise phase to outer planet = 5-6 years
  - Operating phase
- Small acquisition quantities
  - 1 Engineering Model, 1 Flight Model (FM)
  - Maybe 2 FMs and maybe a spare ("single string" is becoming the norm, EM as spare)
- Conservative design philosophy for parts selection
  - You can't bring it back to fix it
  - Use what has worked before, unless there's just no other way.
  - "Space Qualified" components preferred
    - Limited choices for packages (hermetic, ceramic)
    - Radiation test data available
    - Very long lead times (18-24 months)
    - Leads to limited performance, relative to commercial applications
- Conservative operating philosophy
  - No surprises! Everything rehearsed and tested on the ground
  - Don't "brick the box" with a software upgrade (it might be the only box)
  - Very skinny data pipe (bps to kbps) to the box for software upgrades, debug info, etc.

Not like your PC, a "space SDR" isn't going to have a big disk drive, a Gbps Ethernet connection, CORBA middleware, 5 layers of abstraction and dynamic name resolution with late binding and resource allocation.

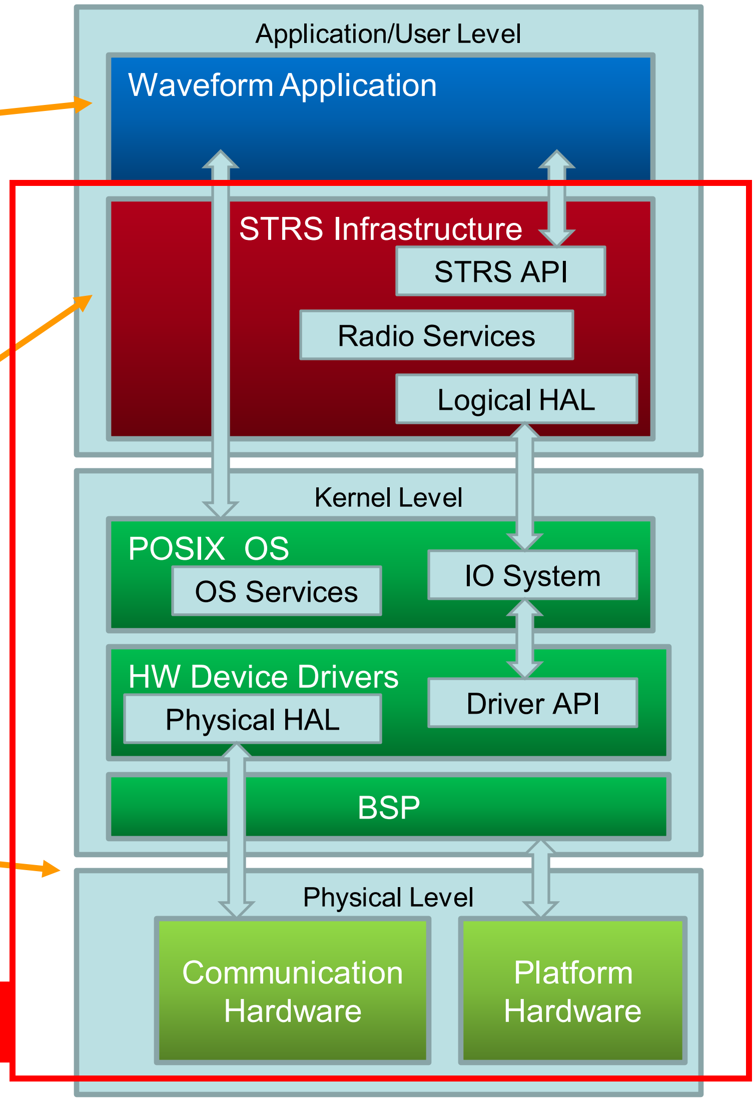
Space Telecommunications Radio System

- Over past few years NASA has promulgated a "lightweight" software radio architecture standard: STRS
  - Suited to more limited platform capability
  - Addresses more immediate needs first
- STRS has 5 goals
  - Evolvable Open Standard Architecture Specification
    - The standard provides a common, consistent framework to develop, qualify, operate and maintain complex reconfigurable and reprogrammable radio systems.
  - Standard Library of Hardware and Software Components
    - *e.g.* TDRSS modem
  - Design Reference Implementation & Specifications
  - Development Tools and Testbeds
    - JPL SDR, NASA/Glenn Research Center SDR3000
  - Flight Tests and Demonstrations
    - CoNNeCT

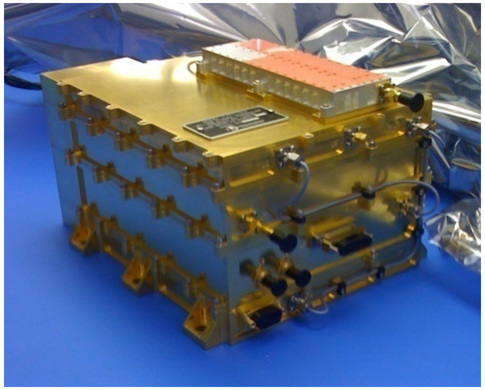
Architecture

- Waveforms (applications)
  - Covers from EM wave to data interface
  - Not just the modulation
    - Framing
    - IP routing
- Operating Environment
  - OS (and its services)
  - Infrastructure
    - helpful stuff the OS doesn't provide
- Hardware platform
  - The metal, the electrons, the photons

STRS Compliant Platform



The JPL Software Defined Radio



9 liters, 6.6 kg
(28.2 L x 20.6 W x15.5 H cm)

15 W Rx (typical)
+ 2 W (GPS)
+ 65W Tx S-band

Full Duplex S-band RF module
- Tx: 2.2-2.3 GHz, 5-10W output
  - 2 x 10bit 50 MSPS DAC (I/Q)
- Rx: 2.025-2.12 GHz, 11 MHz BW
  - 12 bit, 50 MSPS ADC

GPS Receiver
- L1,L2,and L5

Digital Processing
- 66 MHz SPARC V8
- 128 MByte SDRAM + 512 MByte Flash
- 2x Xilinx Virtex II 3Mgate FPGAs
- SDRAM and Flash on each FPGA

Control and Data Interfaces
- MIL-STD-1553B
- 2 SpaceWire Links

Other JPL SDRs in the family

- Electra – now flying around Mars on Mars Reconnaissance Orbiter
  - 17.2 x 21.9 x 14 cm
  - 4.9kg, 63 Watts @ 28V
  - UHF (390-450 MHz)
    - Tx 8.5 Watts
  - 1-2048 kbps BPSK
  - CCSDS Proximity-1 Space Link Protocol
  - 24 MHz SPARC V7
  - Xilinx XQVR 1000 FPGA
  - EEPROM

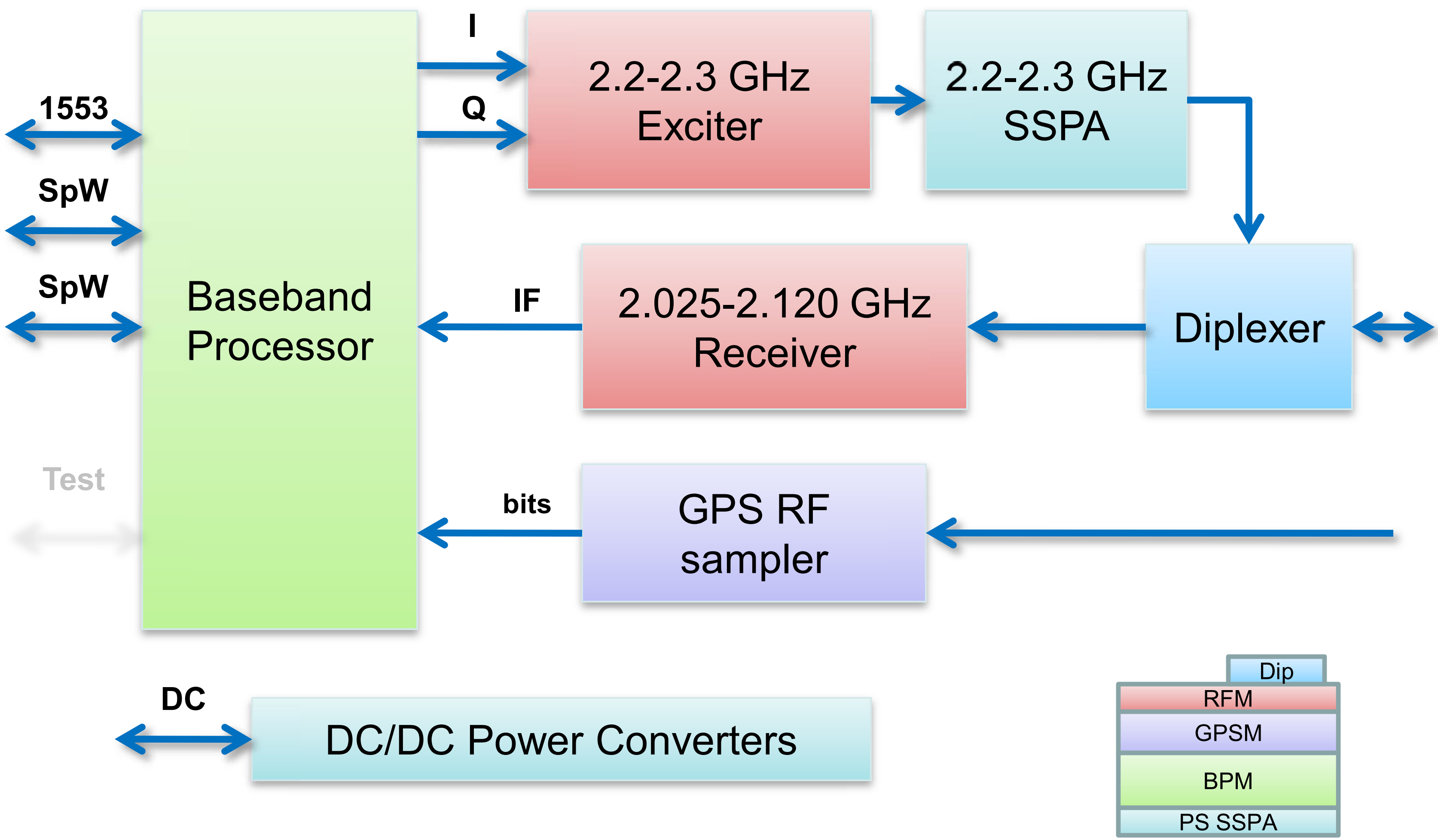
For more info read the paper:

Edwards, et al., "The Electra proximity link payload for Mars relay telecommunications and navigation", 54th Inter. Astro. Cong, Bremen Germany, 29 Sep 2003 at: http://hdl.handle.net/2014/7832

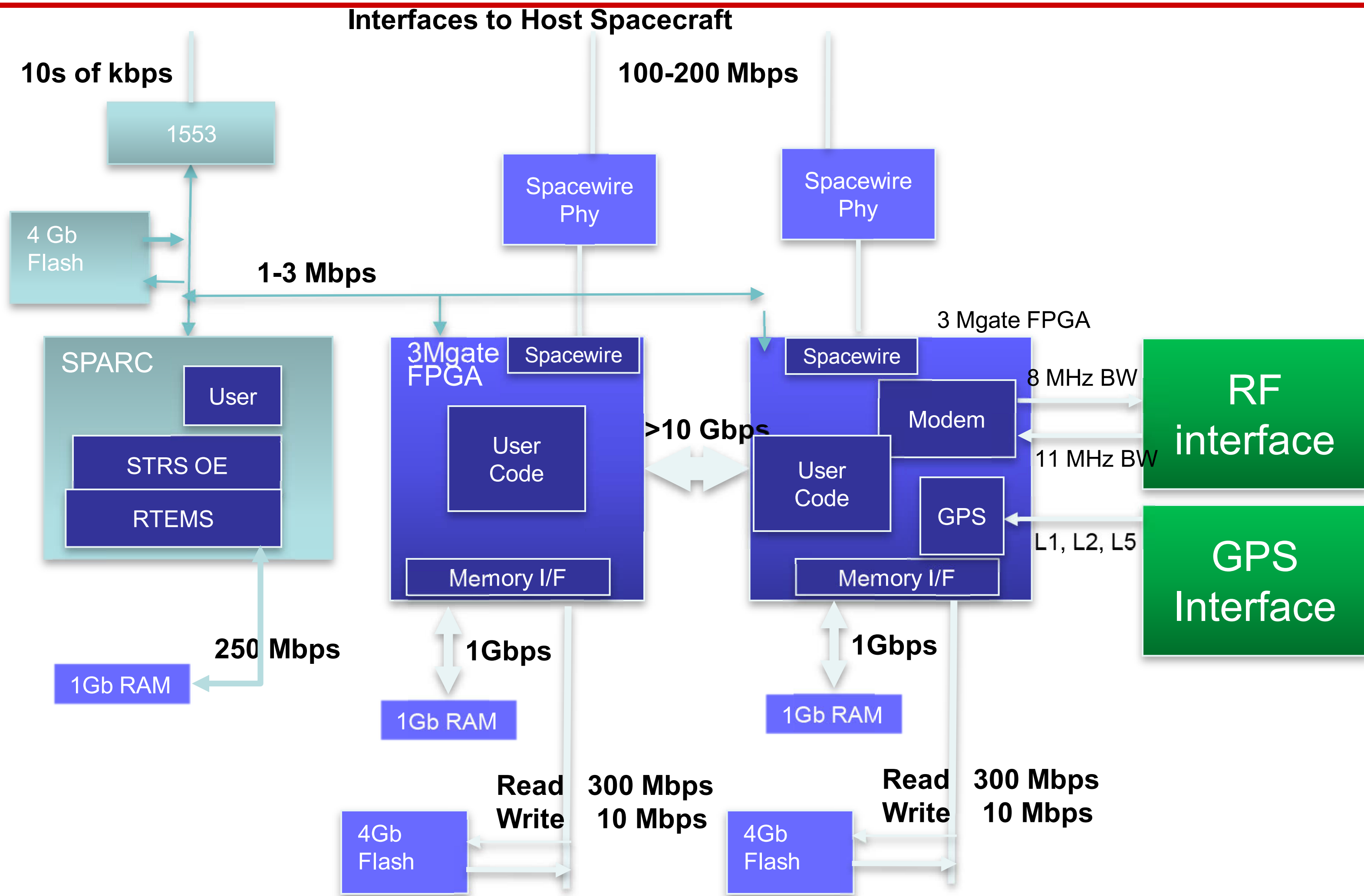
and the book:

Autonomous Software-Defined Radio Receivers for Deep Space Application edited by Jon Hamkins & Marvin K. Simon at: http://descanso.jpl.nasa.gov/Monograph/mono.cfm

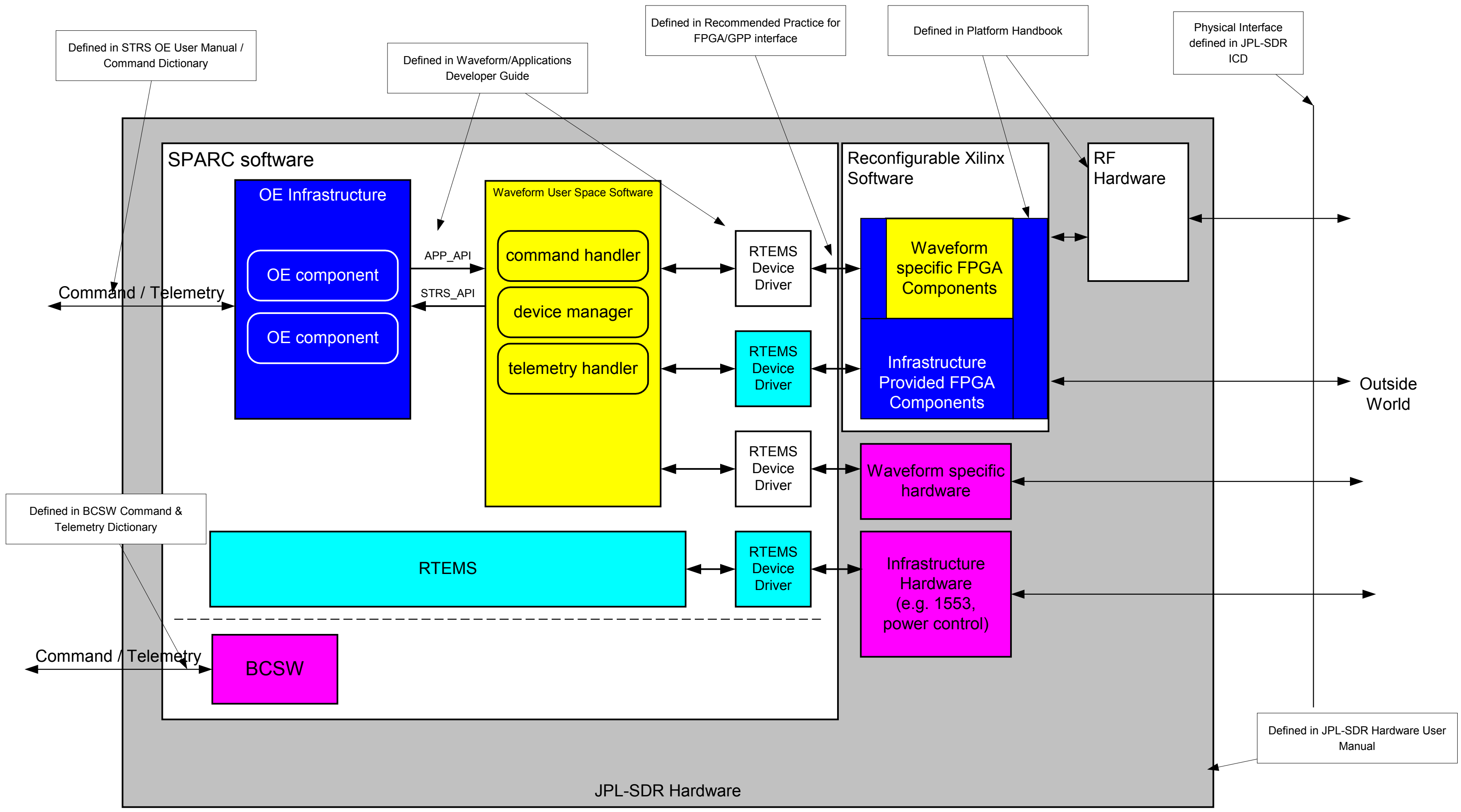
JPL-SDR Hardware Overview



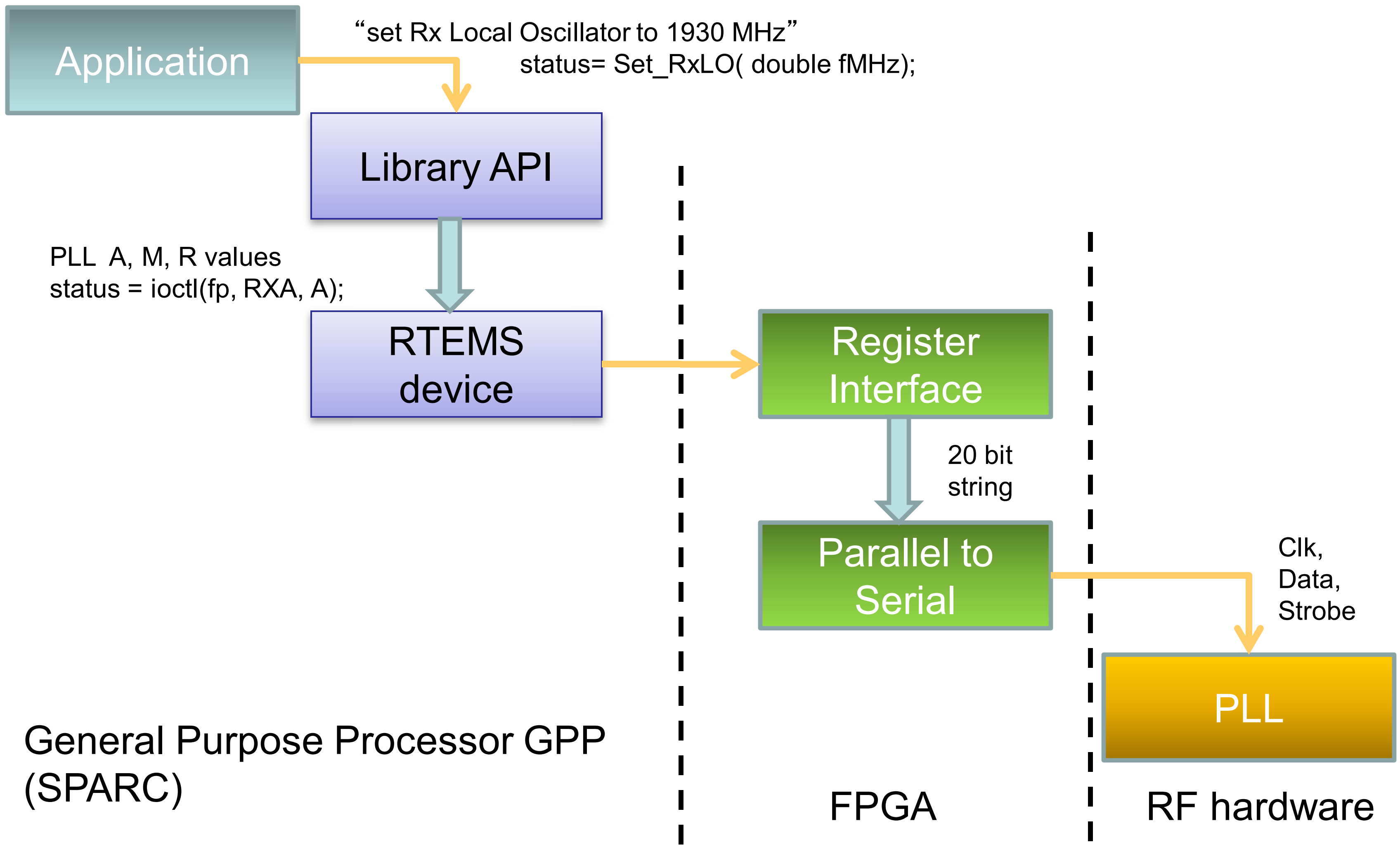
Baseband Processor



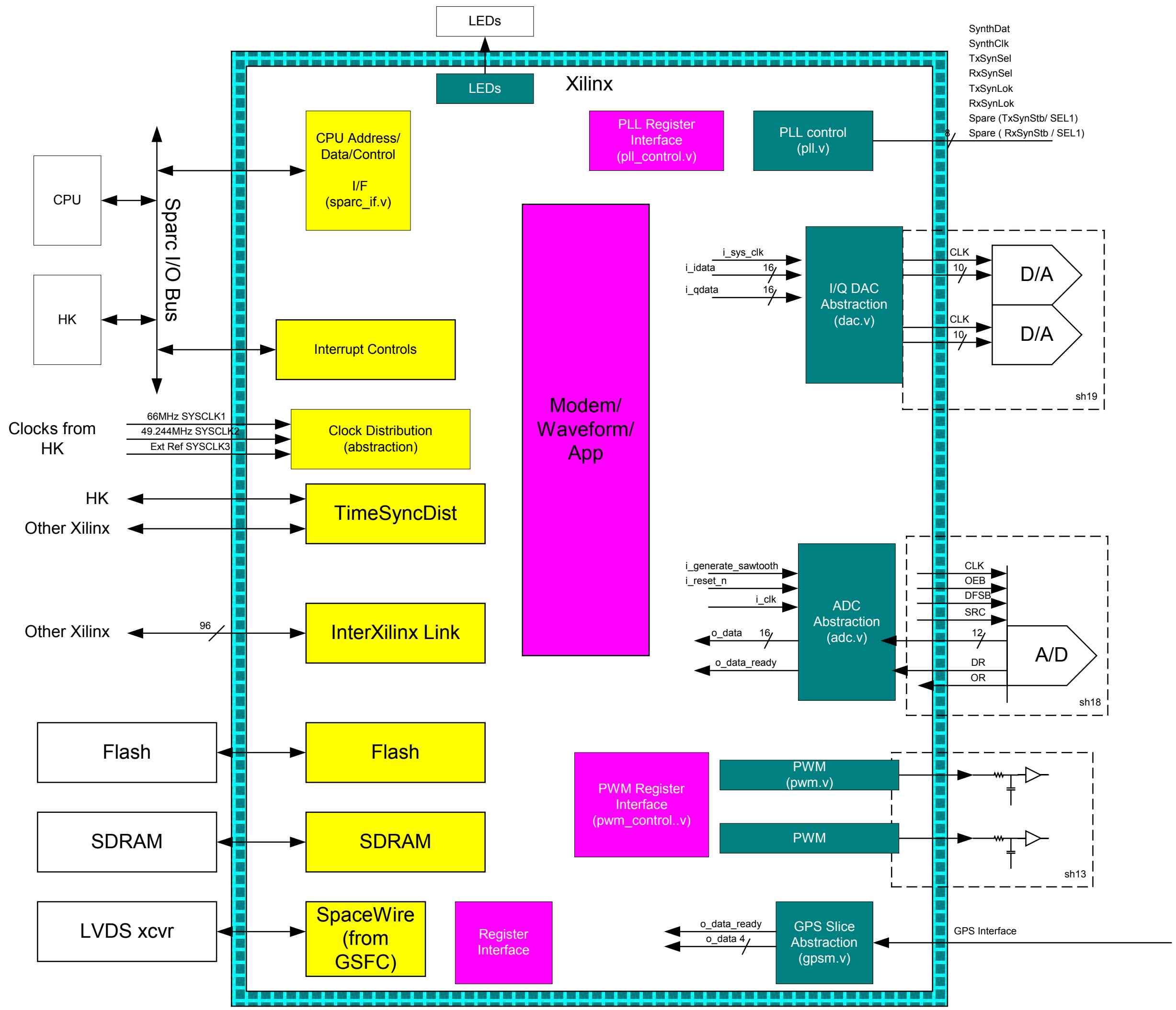
Hardware:Infrastructure:Waveform



Abstractions of Hardware



FPGA interface abstractions

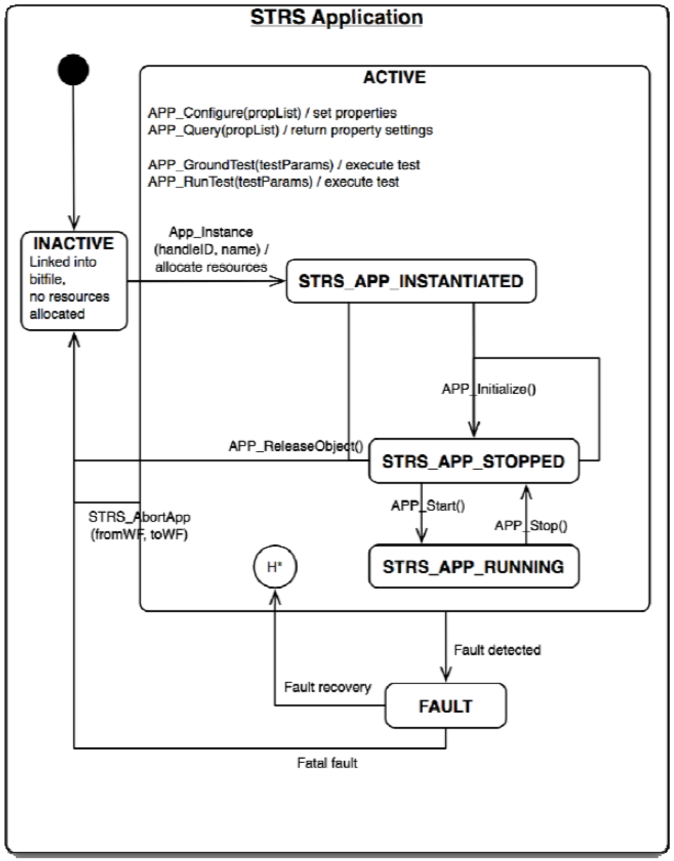


GPP Software Infrastructure

- GPP = General Purpose Processor (distinguishes it from specialized signal processing)
- Command ingest and dispatch
- Heartbeat telemetry
- File Transfers
  - Transfer from host to radio
  - Transfer from radio to host
- Utilities
  - Debug (memory dumps)
  - File system (directory, delete, copy, etc.)
- Hardware control
  - Power on/off for GPS slice and Solid State Power Amp (SSPA)
  - Reference oscillator configuration
- Flash management
  - Copy from InMemory File System to/from Flash
- Application management
  - Initialize, start, stop, configure, query

STRS Application State Diagram

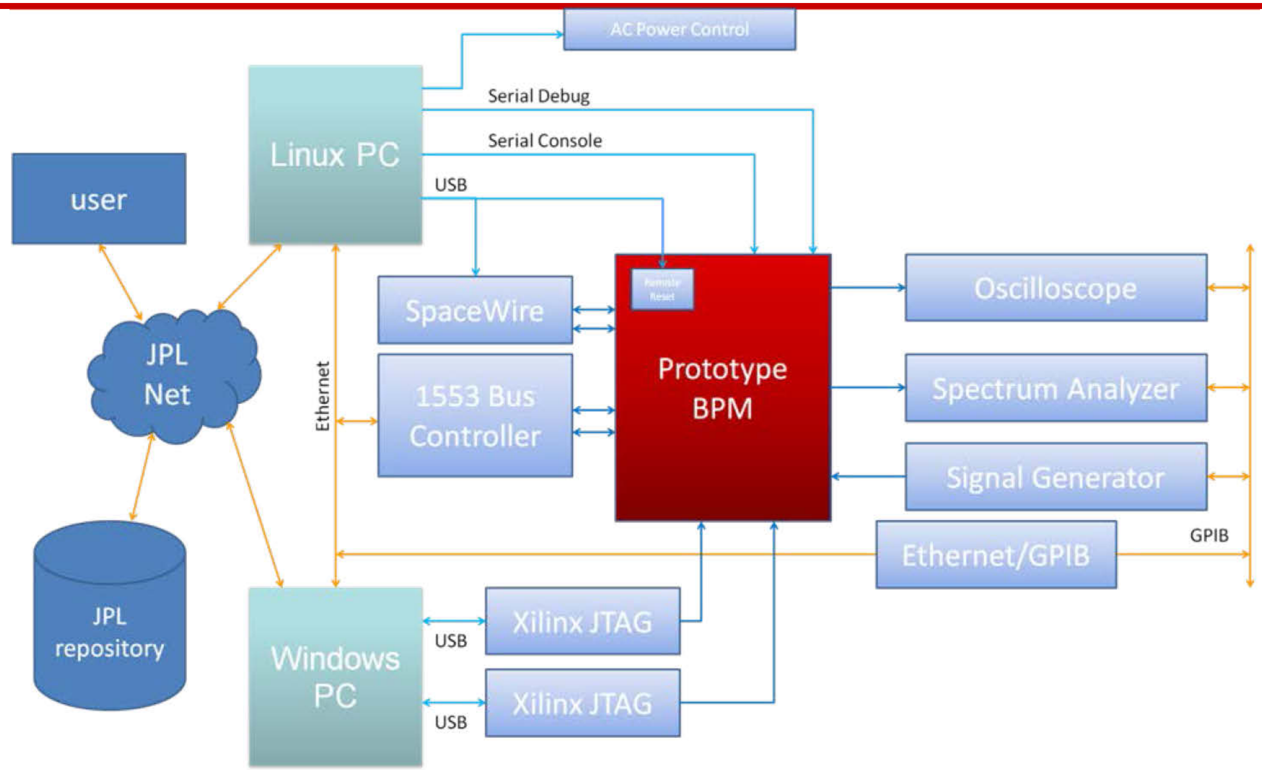
- All STRS applications (waveforms) have the same basic state diagram for interaction
  - OE talks to App via APP_API
  - App uses OE by using STRS_API
- Applications receive “configure” commands
  - Text based commands
    - configure sband 1 txFreq 2250
    - configure sband 1 bitRate 2048
  - Translated by infrastructure to
    - APP_Configure(handle, …)
- Applications return results from “query” commands
  - query sband 1 txPLLlock
  - query connecthw 1 SSPAtemp



Development process

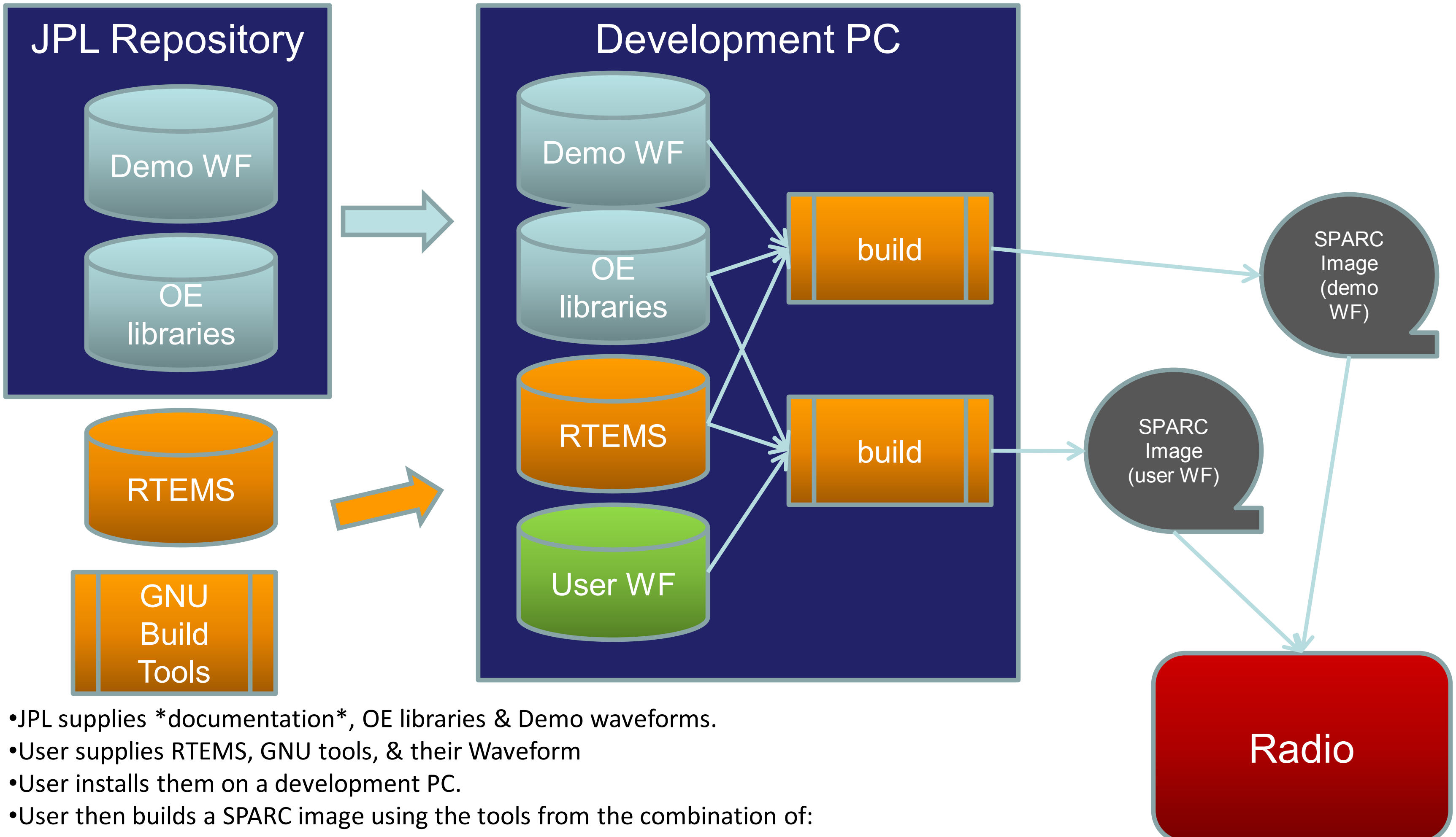
- Standard Toolchain
  - Standard gnu tools for cross development
  - Hosted on a Linux workstation, targeted to SPARC
- JPL STRS OE
  - Libraries of utilities and infrastructure (device drivers)
  - Standard RTEMS 4.9.3 for the RTOS
  - Libraries for FPGA modules
- Build process
  - Build for the target
  - Test on prototype BPM
  - Test on flight unit in Ground Testbed
  - Schedule for upload to spacecraft
  - Test in flight on ISS

Development Workstation



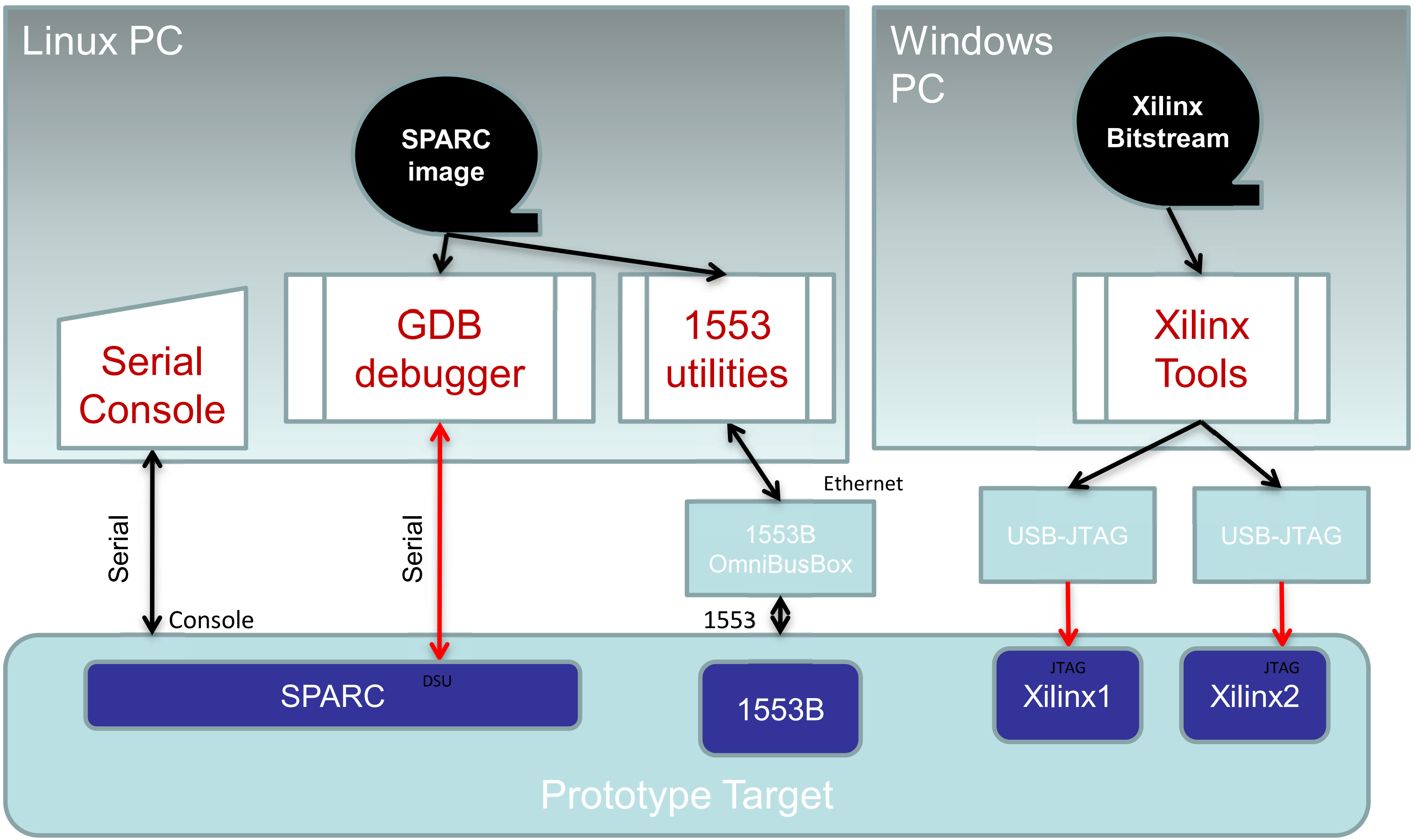
- Uses a prototype of the Baseband Processor Module
  - the ADC/DAC/DSP and SPARC
  - RF sections simulated with test equipment
- Remotely accessible
- Similar versions with the actual radio are possible
- Simulated processors are available, but… the simulation doesn't have the timing fidelity and it's hard to do signal processing.

High level flow

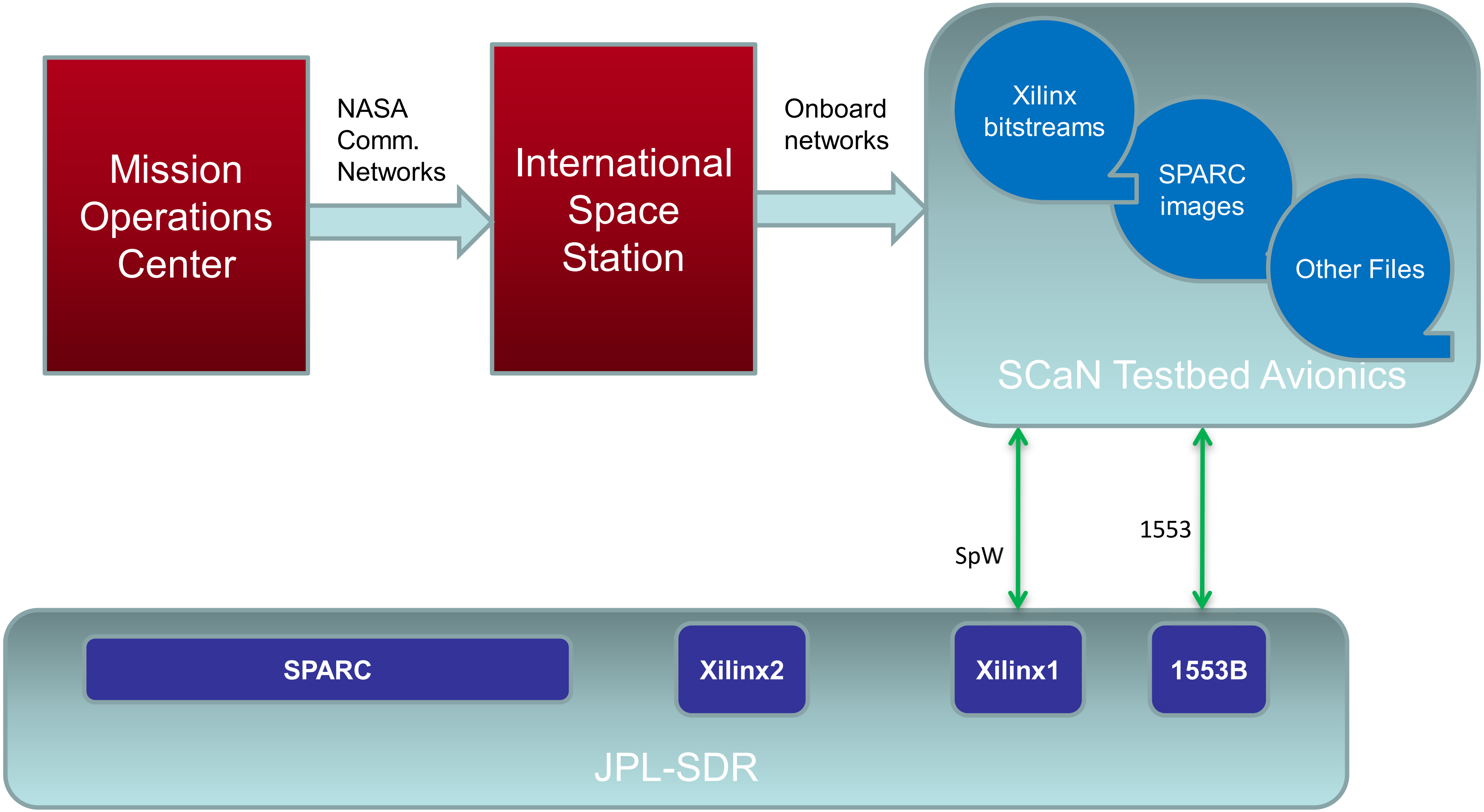


- JPL supplies *documentation*, OE libraries & Demo waveforms.
- User supplies RTEMS, GNU tools, & their Waveform
- User installs them on a development PC.
- User then builds a SPARC image using the tools from the combination of:
  - OE libraries, RTEMS and one or more applications.
- The loadable image is then loaded into the radio and run.
- Similar flow for FPGA: JPL libraries in Verilog and VHDL, you use the synthesis tool of your choice

Loading images on the target (debug)

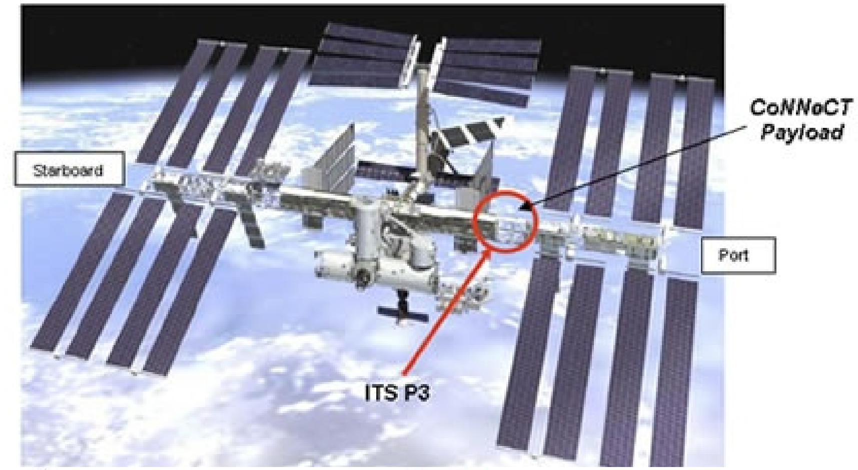


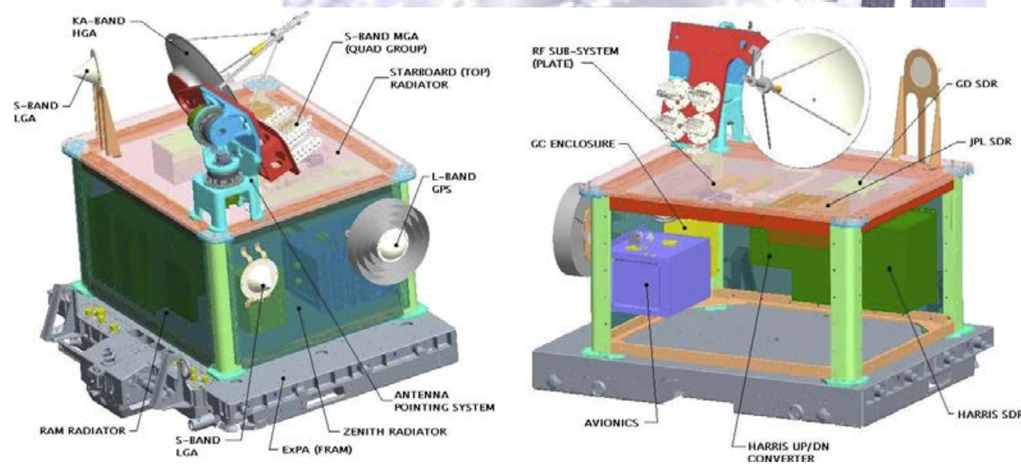
Loading new software in Flight



CoNNeCT

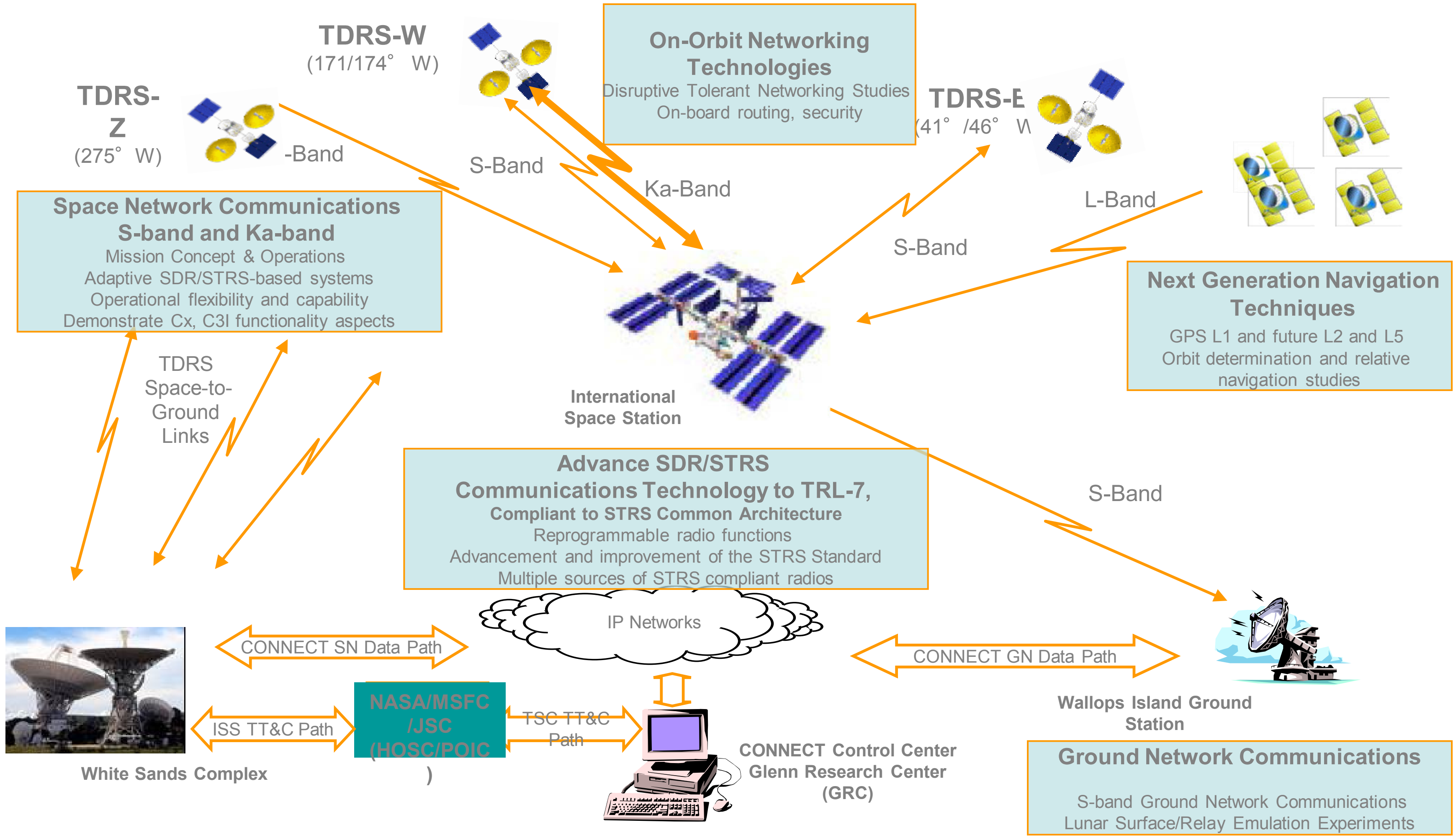
- Communications, Navigation, and Networking Reconfigurable Testbed
- 3 SDRs, antennas, and a flight computer on International Space Station
- Launch in Jan 2012, available for experimental operations 2012 and 2013
- Experiment Announcement of Opportunity soon





More info: http://spaceflightsystems.grc.nasa.gov/SpaceOps/CoNNeCT/

Another View of CoNNeCT



Future of space SDRs

- SDRs are here for most conventional needs
  - Many space radios are already software defined, but just not in-flight reconfigurable
- Work is in progress to add network stacks and other functions
  - *i.e.* Disruption Tolerant Networking (DTN) & GPS
  - Limited only by processing resources and creativity of developers
- SDRs are not a panacea
  - Highly specialized functions needing especially low power or mass and no network function
    - *e.g.* radio science coherent carrier only transponder
  - As technology advances, the "break even" line between specialized and general SDR will move
- SDRs are challenging to specify and develop requirements for
  - How do you specify generalized "platform" requirements when the ultimate communication or application requirement won't be known until after launch?