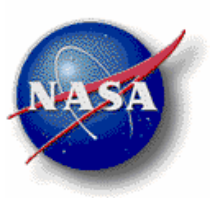
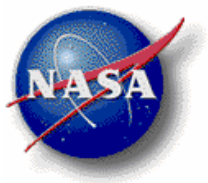


Space Telecommunications Radio Architecture (STRS)

Technical Overview

IDGA 4th Software Defined Radio Conference
February 2006



Abstract

A software defined radio (SDR) architecture used in space-based platforms proposes to standardize certain aspects of radio development such as interface definitions, functional control and execution, and application software and firmware development. NASA has charted a team to develop an open software defined radio hardware and software architecture to support NASA missions and determine the viability of an Agency-wide Standard. A draft concept of the proposed standard has been released and discussed among organizations in the SDR community. Appropriate leveraging of the JTRS SCA, OMG's SWRadio Architecture and other aspects are considered.

A standard radio architecture offers potential value by employing common waveform software instantiation, operation, testing and software maintenance. While software defined radios offer greater flexibility, they also poses challenges to the radio development for the space environment in terms of size, mass and power consumption and available technology. An SDR architecture for space must recognize and address the constraints of space flight hardware, and systems along with flight heritage and culture.

NASA is actively participating in the development of technology and standards related to software defined radios. As NASA considers a standard radio architecture for space communications, input and coordination from government agencies, the industry, academia, and standards bodies is key to a successful architecture. The unique aspects of space require thorough investigation of relevant terrestrial technologies properly adapted to space. The talk will describe NASA's current effort to investigate SDR applications to space missions and a brief overview of a candidate architecture under consideration for space based platforms.

FYI Only: not part of actual presentation

Space Telecommunications Radio System

Goal

- Improve capability of developing and operating NASA's space and ground transceiver communication assets

Objectives

- Systematic approach to analyze NASA communication requirements to apply and evaluate open architecture standards, concepts, and radio technologies for space
- Develop a transceiver system open architecture for space and ground communications
- Leverage the DoD's investment in software defined radio technology and software communications architecture as appropriate

STRS, Why now?

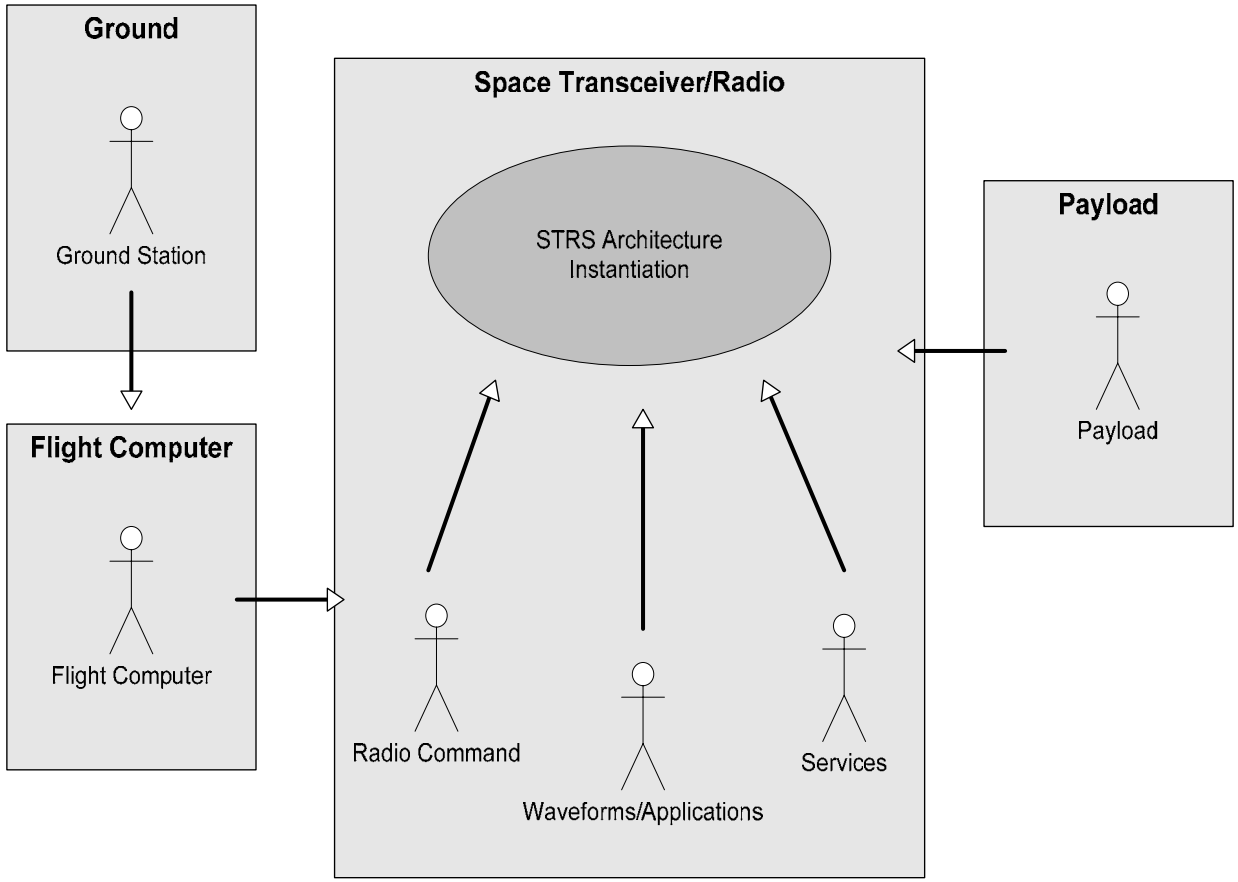
- Technology advancements in signal processing hardware make it possible for space based reconfigurable platforms
- Technology allows use of more software intensive systems for communication and navigation functions.
- Standard architectures for SDR emerging
  - JTRS, OMG, SDR Forum, etc.…
- SDR's being used in missions and technology demos (e.g. MRO, STS-107) without benefit of commonality or leverage from one development to another
- Architecture serves as a framework for commonality/interoperability among suppliers to get Agency wide benefit among future radio developments
- SDR's offer opportunity to reduce life cycle costs through software and firmware updates as opposed to hardware fixes due to problems or requirement creep.

Recent SDR Activities Within NASA

- Champ, Grace, Jason
  - Blackjack
  - L-band Receivers
  - Reconfigurabel/Reprogrammable
  - Waveforms
    - GPS
- STS-107, CANDOS Experiment
  - Low Power Transceiver
  - S-band Transceiver, L-band Receiver
  - Reconfigurabel/Reprogrammable
  - Waveforms
    - TDRSS single access and multiple access
    - GPS
- Mars Reconnaissance Orbiter
  - Electra Radio
  - UHF Transceiver
  - Reconfigurable/Reprogrammable
  - Waveforms
    - Proximity-1 protocol

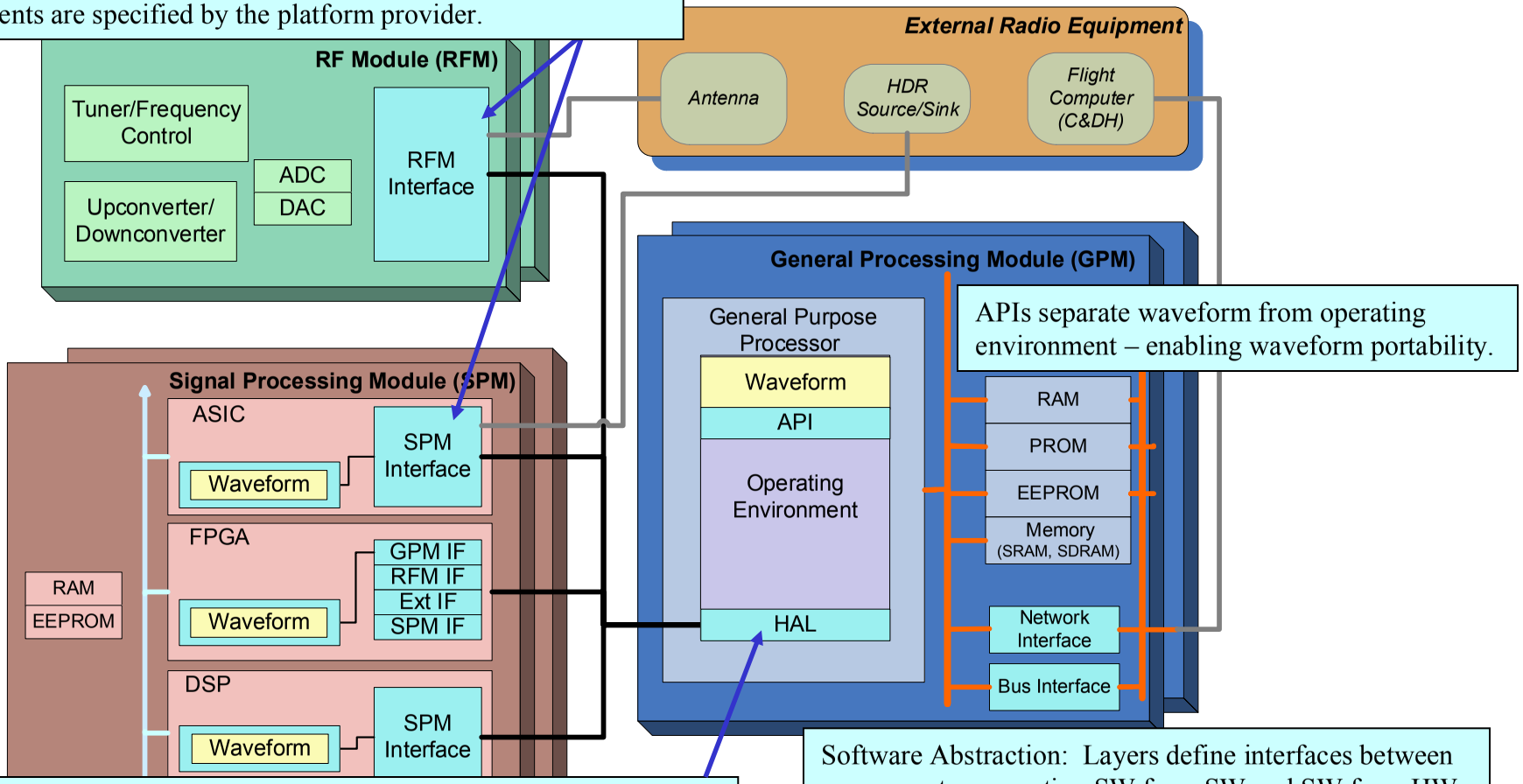
STRS Context Diagram

- Ground controls spacecraft assets
- Flight computer
  - Commands
  - Telemetry
- STRS Radio consists of:
  - Waveforms
  - Applications
  - Services
    - WF Instantiation
    - Communications
- Payload provides data to STRS Radio



Interface Descriptions Functional Diagram

Module Interfaces abstract and define the module functionality for data flow to waveform components. Enables multiple vendors to provide different modules or add modules to existing radios. Electrical interfaces, connector requirements, and physical requirements are specified by the platform provider.



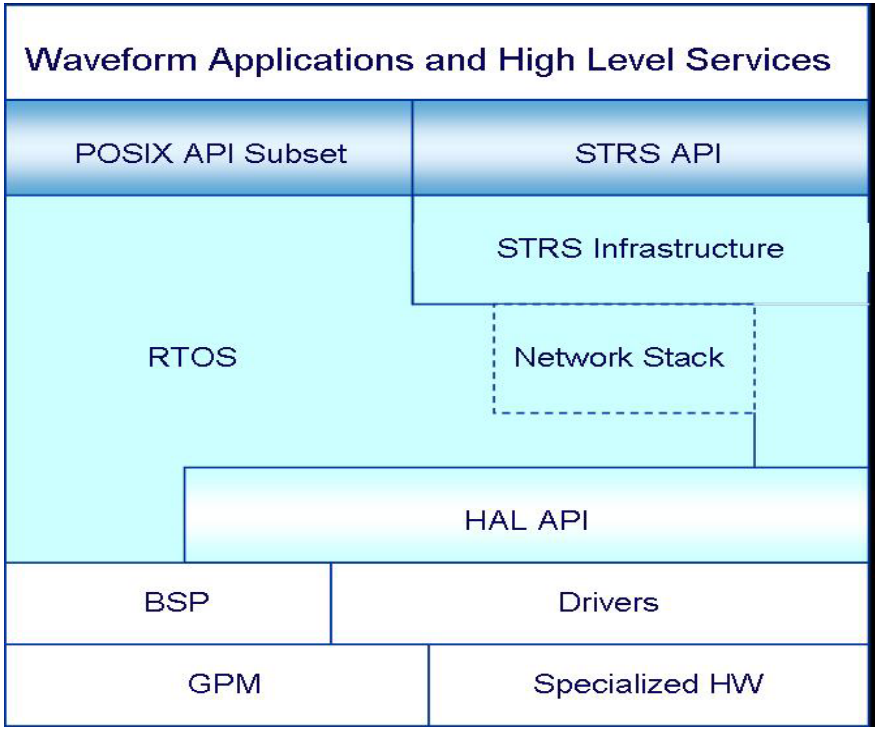
APIs separate waveform from operating environment – enabling waveform portability.

Software Abstraction: Layers define interfaces between components, separating SW from SW and SW from HW.

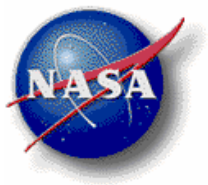
HAL API is published so that specialized HW developed by multiple vendors can be integrated with another vendor's STRS infrastructure.

STRS Software Structure

- **Waveform Applications**
  - Uses POSIX APIs to access RTOS
  - Uses STRS APIs to access STRS infrastructure
- **High Level Services**
  - Control waveform
  - Monitor waveform (QoS)
- **STRS Infrastructure**
  - System management
  - Device control
  - Data transfer
- **RTOS**
  - Real-Time Operating
  - System implementing a subset of POSIX
- **BSP**
  - Hardware Abstraction Layer (HAL)
  - Drivers
  - Hardware Interface (HID)
- **HW**
  - Hardware communications equipment (e.g. FPGA, DSP, A to D, and D to A)
  - Hardware Interface Definition



The above diagram shows relationship of the various software layers.



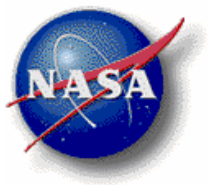
STRS API Overview

STRS API Categories

- System Management
  - API provides the functionality to control and operate waveforms
  - This includes APIs that must be supplied by the customer's waveform and services, supplied by the architecture to the waveform and services, and the operating services
- Inter-process Communications
  - Allows applications to communicate via messages between one another
  - As the ability for waveforms to communicate with other STRS applications is crucial for the operation of radio services, STRS specifically defines the API
- Device Control
  - Interfaces between the waveform and the endpoint communication algorithm block.
  - Set of service functions to provide basic interface services to the devices via the HAL drivers
- Control and Telemetry
  - A set of common radio services that provide a consistent set of command and control interface to system designers (Interfaces are the externally accessible interfaces)
  - Data capacity and telemetry data format provided by a STRS Platform

POSIX Subset

- Used for C/C++ language support for mathematical computation, string and character handling, I/O, and threads
- Determining appropriate subset of POSIX APIs for STRS



Challenges to Common Radio Standard

- Architecture Development
  - Definition (open/closed)
  - Consensus
  - Reference Implementation
- Operations Requirements
  - Reliable Software
  - Radiation Environment
  - Risk
- Component Technology
  - FPGA
  - Memory
  - A/D Converters
- Compliance Verifications
  - Software Library
  - Tools and Development Kits
- Management
  - Space radio market
  - Business case analysis

Call for Industry Participation and Review Opportunity

- Comments on STRS architecture standard
  - Architecture Description Documents and high level Functional Requirements (other document (e.g. Use Cases, Requirements, STRS Specification) to follow later in '06/'07)
  - Provide written comments through RFI and SDR Forum
- Provide individual feedback through RFI
  - http://www.grc.nasa.gov/WWW/Procure/home.htm
  - Responses due to GRC by March 15th, 2006
  - Participate in SDR Forum Space Applications Work Group Review – Meeting in April, 2006
- Typical questions or considerations
  - Comments on architecture and approach (too much, too little - **what's missing**?)
  - Feasibility of producing architecture compliant radios
  - **Industry solutions to hw and sw interfaces to include in the STRS standard**
    - **API's or standard hardware interfaces (HID) used that should become standard for NASA radios**
  - **Recommendations on standards for software/firmware development for space**
  - Cost to implement architecture
    - Both from a performance (e.g. architecture overhead) and a financial (e.g. estimated NRE and recurring costs) perspective
  - **Examples of statistics or metrics when using SCA (magnitude of the savings)**
  - Available tools to model and characterize architecture
    - Resources (e.g. power consumption, performance, latency, cost)
  - **Life cycle/business case advantages/disadvantages of open architecture (development, operations, maintenance)**
  - Industry's role in future development activities

Phase I
STRS Architecture Review and Update Plan

- Introduce STRS Architecture
  - Released STRS Architecture Description Document via Request for Information/Comments (RFI
    - (**http://www.grc.nasa.gov/WWW/Procure/home.htm)**
  - Software Defined Radio Forum, January 2006
  - IDGA Software Radio Conference, February 2006
  - IEEE/AIAA Aerospace Conference, Big Sky (Space Exploration Communications and Navigation Panel) March 2006

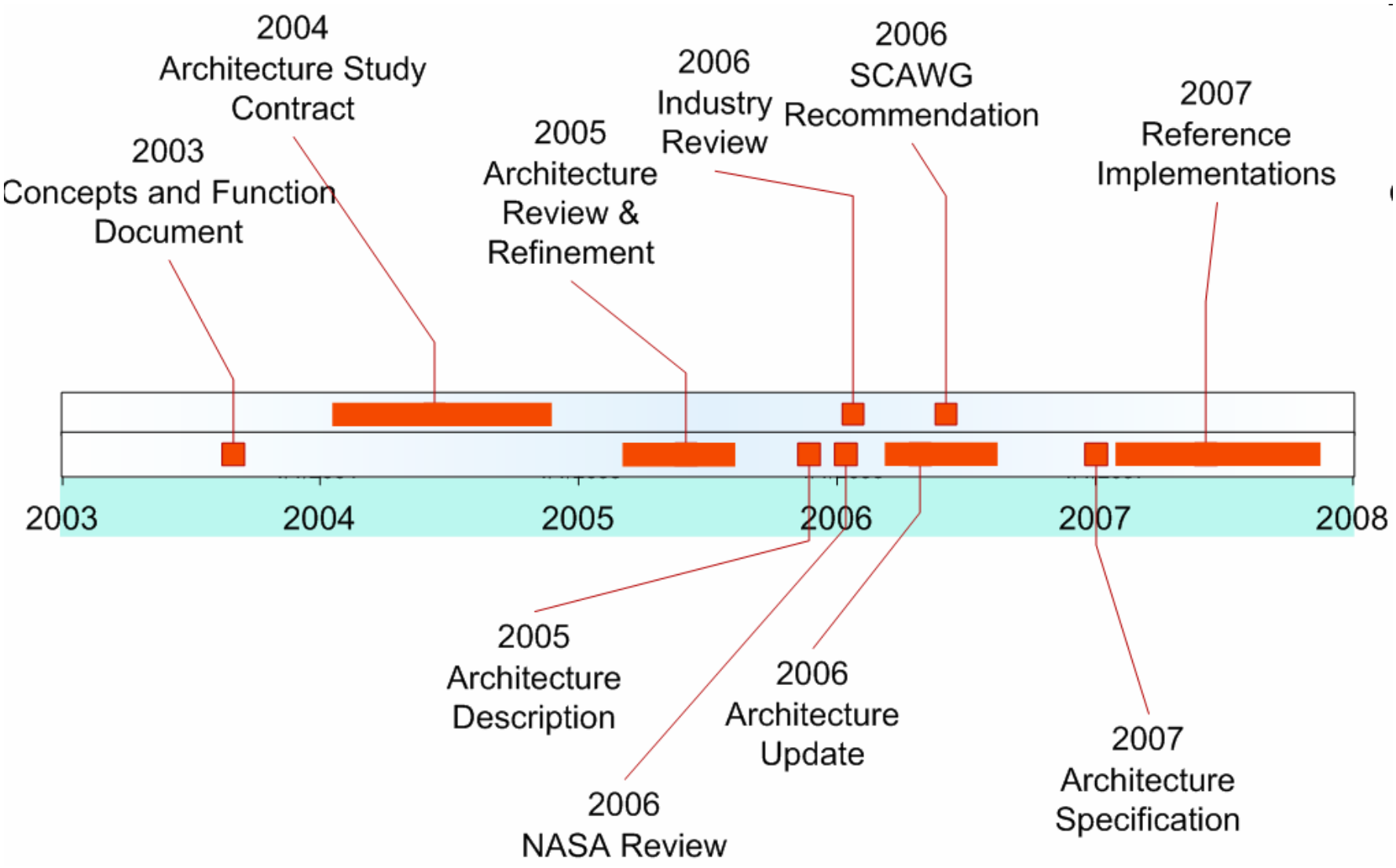
- Solicit comments from individual companies through RFI
  - Individual written response by March 15th, 2006
  - SDR Forum response by ~ April 30th , 2006 (Coordinated by Karen Perry/Prism Technology Corp.)
    - SDR Forum, Space Applications Study Group Meeting on April 10-13, 2006

- Update STRS Architecture by May 2006 based upon:
  - Comments from industry
  - NASA internal reviews
  - Completion of various trade studies

- Determine Architecture Standard Recommendation

- Develop reference implementations based upon architecture
  - Characterize architecture by through reference implementations of key hw & sw aspects
  - Update architecture based upon implementation findings
  - Develop prototypes based upon refined architecture

STRS Architecture Notional Development Process



JTRS/SCA Applicability

- Leverage JTRS/SCA
  - Most JTRS program goals applicable or similar to NASA's
    - Philosophy similar to NASA approach (e.g. open architecture, industry participation, technology insertion, networking, software reuse, portability)
  - Alternative to NASA developing a unique SDR standard
    - Need to define NASA's role in architecture standard's bodies
  - Networking and security are integral to JTRS design, although level of security may exceed NASA's security requirements (TBD)
  - Can use previous cluster experience for waveform definition and development
  - Resource (e.g. memory) consumption and complexity of SCA exceeds typical spacecraft communication resources.

- Status of Space Application
  - NASA and JTRS have MOU to maintain collaborative partnership
  - SCA is being used for SATCOM appls (i.e. wideband ground based radios)
  - Optimized SCA implementations may apply to space-based radios

Backup

NASA Architecture Development Approach

- Assemble experts from different Centers to achieve a NASA SDR architecture standard
- Leverage previous and ongoing SDR work from each participating Center
  - Architecture Definition
  - Mission Applications
  - Architecture and Radio Technology Developments
  - Flight Radio Developments
- Identify applications of SDR technology across NASA missions
- Develop an open SDR HW & SW architecture and recommendation for the Space Communications Architecture Working Group (SCAWG) by May 06
  - NASA's new Space Communications and Navigation Architecture enabling NASA's Exploration and Science programs with operational flight dates from 2012-2030
  - Supports the SCAWG 5-year incremental communications architectures
- Determine the viability of an Agency-wide SDR standard

Assumptions and Questions

ASSUMPTIONS

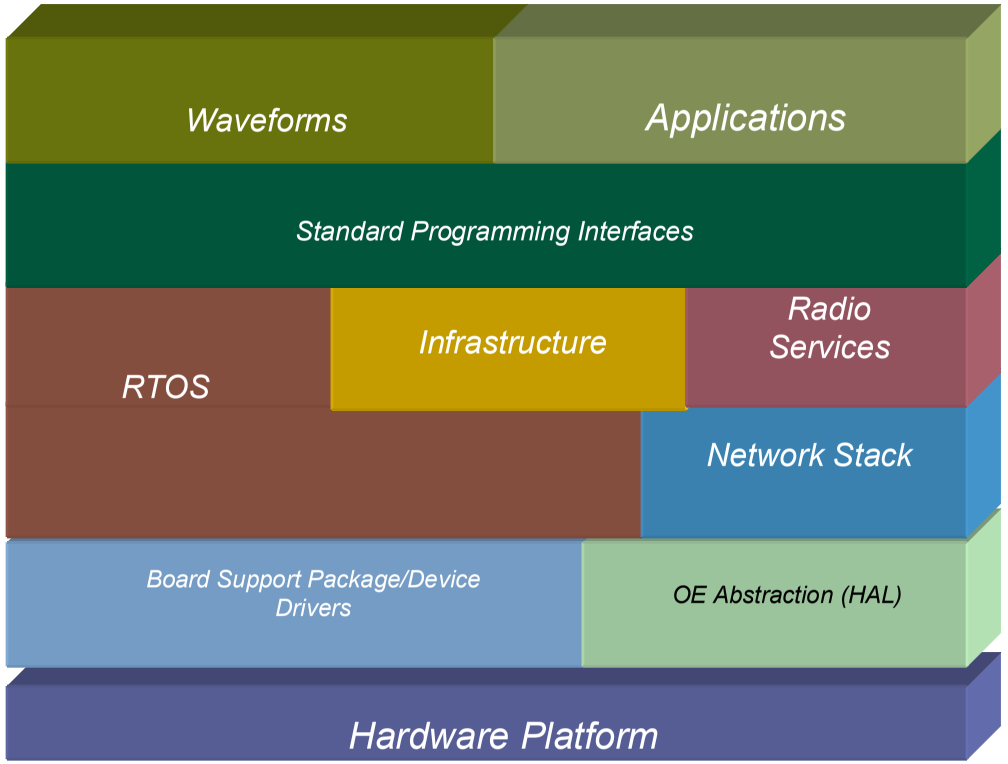
- Future radios will employ more advanced reconfigurable hardware
- Vendor unique SDRs prevent/hinder the sharing of software across platforms or pose high porting cost
  - Waveform typically tied to platform
- Standardized architecture is applicable to SDR's among most missions
- Architecture should not specify implementation details
  - limit innovative design and tech infusion
- The SAT architecture framework will be evolvable and adaptable.

KEY QUESTIONS

- What type of missions will benefit most?
- Ho do we apply lessons learned form other efforts?
- Is an architecture cost effective for the number of missions using SDRs?
- Should NASA maintain its own architecture definition, or align with a larger user base?
- How much processing overhead is acceptable to benefit from an open architecture?
- How does waveform processing in an SDR affect the architecture?

STRS Layered Software Model

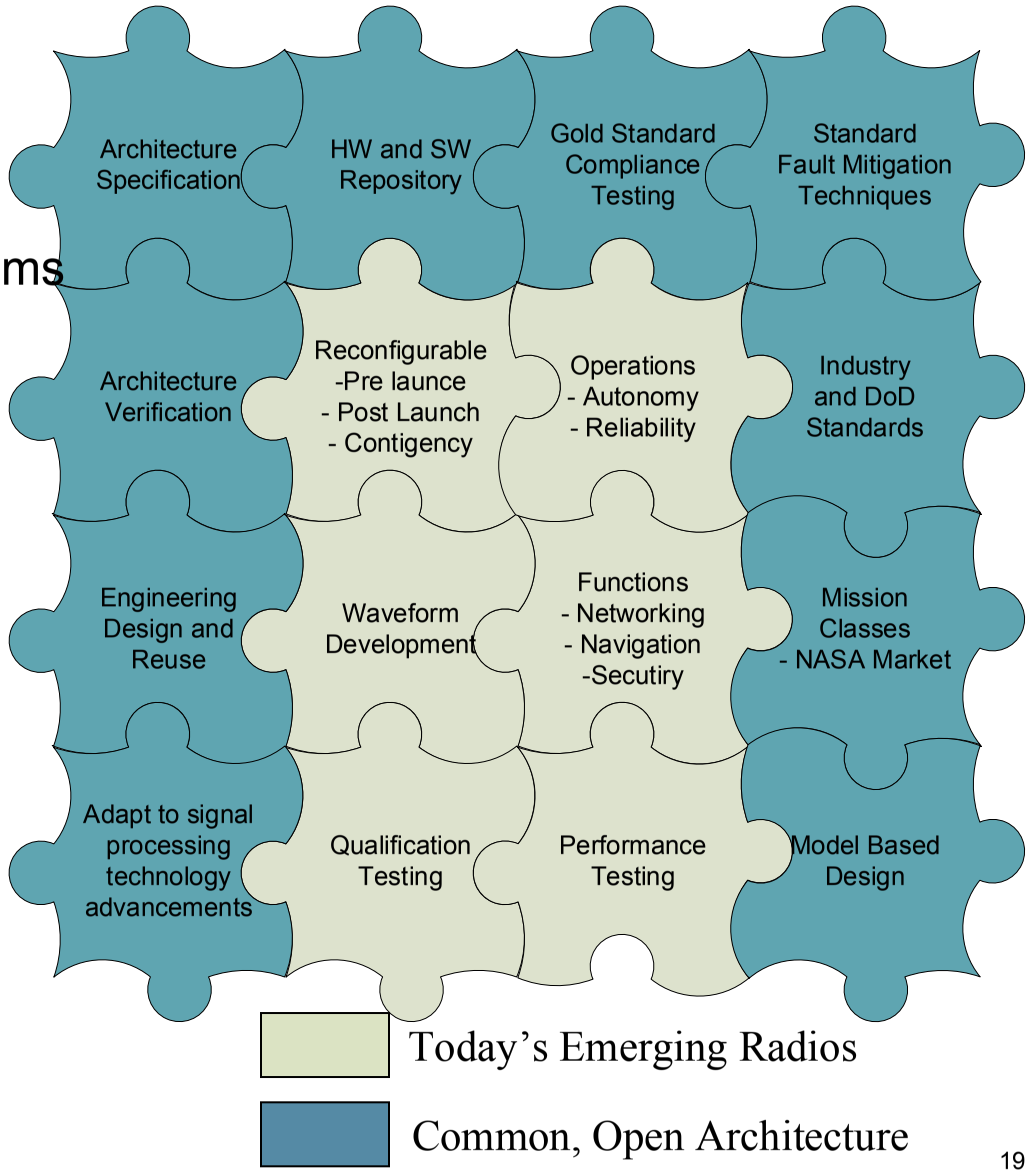
- **Waveform Applications**
  - Uses POSIX APIs to access RTOS
  - Uses STRS APIs to access STRS infrastructure
- **High Level Services**
  - Control waveform
  - Monitor waveform (QoS)
- **STRS Infrastructure**
  - System management
  - Device control
  - Data transfer
- **RTOS**
  - Real-Time Operating
  - System implementing a subset of POSIX
- **BSP (Board Support Package)**
  - Hardware Abstraction Layer (HAL)
  - Drivers
  - Hardware Interface (HID)
- **HW**
  - Hardware communications equipment (e.g. FPGA, DSP, A to D, and D to A)



This diagram illustrates the layered programming approach to STRS. Software is extensible through layered programming to provide an abstracted user interface and development environment.

Aspects of SDR and Architectures
How the pieces fit…

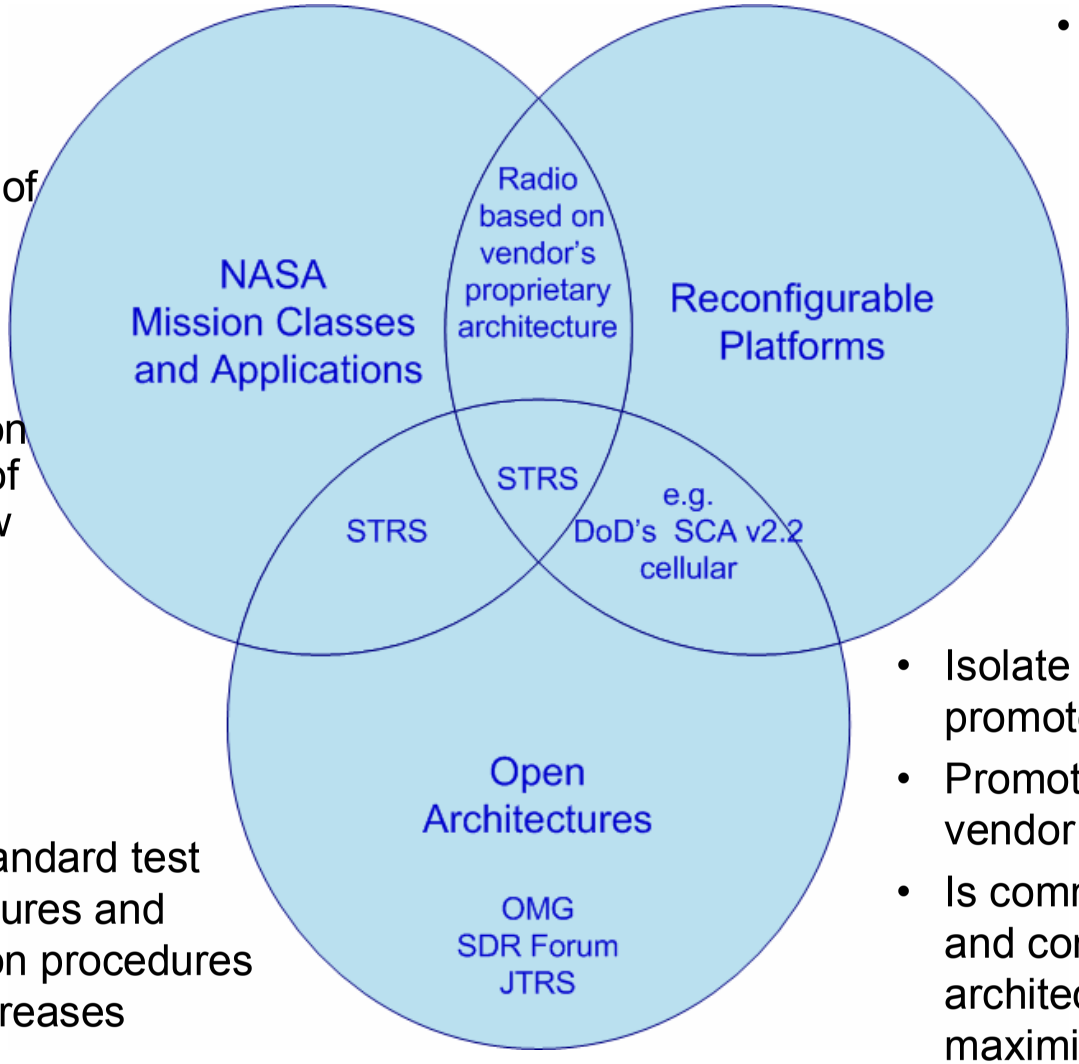
- "SDR" means more than just "reconfigurable"
- Today's radio technology offers new development and operational paradigms
  - Upgradable
  - Flexible
  - Industry trends to SW based systems
- Architecture adds framework to reconfigurable (SDR) transceiver technology
  - Multi-vendor participation
  - Common Interfaces
  - Technology Insertion
  - Common Specification
  - Reuse
  - Software verification - Developing a qualification approach suitable for missions



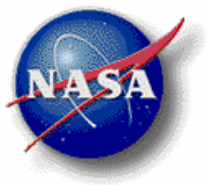
SAT Overview

Applying SDR and Standard Architectures to NASA

- SDR's will provide more mission flexibility
- Space qualification of software/firmware intensive systems may add costs to missions
- Simplify new mission adoption by reuse of standard hw and sw components
- Need to verify on-orbit software reconfigurations (risk)
- Provides path to standard test qualification procedures and validation/verification procedures as software use increases



- Potential for increased processing (i.e. power) requirements and larger form factors vs. predominately hardware based solutions (e.g., ASIC)
- Isolate SW from HW to promotes reuse
- Promotes/enables multi-vendor participation
- Is common NASA, DoD, and commercial architecture needed to maximize market for space radios to reduce cost?



Assumptions

1. Industry will pursue unique, proprietary SDR developments and technologies for space to reduce their production costs and increase marketability regardless of NASA mission specific needs.

2. A clear boundary should be identified between the open and proprietary portions of the standard architecture to encourage industry participation and better align industry development and NASA requirements.

3. Specifying implementation details as opposed to architecture will limit innovative design and technology infusion limiting the usefulness of the architecture. Need to keep architecture aspects separate from implementation. The SAT will need to specify a boundary between architecture and implementation.

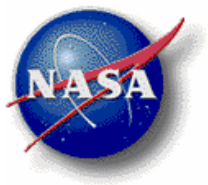
4. A standard open architecture would provide for easier and less costly waveform porting from one proprietary SDR approach to another.

Assumptions (cont)

5. A standard open architecture would likely:
   - Increase the number of industry participants who could provide software or hardware components.
   - Enables upgradeability as requirements and technology mature.
   - Enables common hardware qualification test procedures and common software validation and verification procedures.
   - Enables common interfaces and common commands for both development and operations (e.g., common command and telemetry interfaces).
   - Provide standardized, command formats to enhance software development.

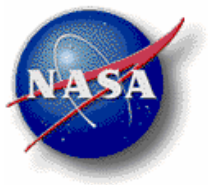
6. Less dependence on specific vendor's implementation may reduce cost and risk to develop, augment, and maintain reconfigurable systems. Multiple vendor participation in radio development will reduce costs (e.g., separate platform developer and waveform developer)

7. Software and hardware design and module reuse from a software and hardware repository or library will reduce radio development cost, offers potential to reduce qualification cost (e.g. qualification procedure reuse, reduction).
   - Waveform model, design, source code reuse.
     - Use of COTS software products (could include tailoring for space).
   - Hardware design/module reuse.



Questions

1. What characteristics make a good architecture?
   - What quantitative metrics can be used to evaluate an architecture
2. How much overhead in resources or processing is reasonable to expect to get the benefits of open architecture?
3. What are the lessons learned from JTRS/SCA that should be considered during the architecture definition.
4. Are the benefits of open architecture enough for NASA to maintain its own definition or do we need to align with another radio user or larger customer base (e.g. DoD/SCA, SDRF/SCA).
5. What's the best way to get scalability across smaller applications (e.g. EVA and rover)? How do you scale an architecture, yet comply to a single standard?
6. What portion of navigation and networking functions belong in SDR?
7. What is the appropriate set of API's and hw interfaces to maximize waveform portability, but minimize resources?
8. Are there hardware aspects that should be standardized (e.g. hardware slices , interconnected backplane for modules or slices) implementation left to vendors?



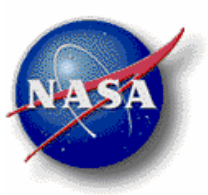
Architecture Evaluation Criteria

- Openness
  - Non-proprietary set of rules and interfaces
  - Scalable, Adaptable
- HW/SW Isolation
  - Enables technology insertion
- Radio Performance
  - Does it meet operational requirements (e.g. latency, bootup)
  - Reliability
- Cost
  - NRE (Radio development based on architecture)
  - Recurring (Evolve architecture)
- Resource Consumption
  - Power and memory consumption
  - Logic equivalents, number of operations
  - Size, mass
- Software Portability
  - Software architecture should not require any specific vendor, hardware component or software vendor for implementation
- Complexity
  - Measure of capability
- Design Reuse
  - Flexible tool use across required hardware platforms

JTRS/SCA Applicability

- Leverage JTRS/SCA
  - Most JTRS program goals applicable or similar to NASA's
    - Philosophy similar to NASA approach (e.g. open architecture, industry participation, technology insertion, networking, software reuse, portability)
  - Alternative to NASA developing a unique SDR standard
    - Need to define NASA's role in architecture standard's bodies
  - Networking and security are integral to JTRS design, although level of security may exceed NASA's security requirements (TBD)
  - Can use previous cluster experience for waveform definition and development
  - Resource (e.g. memory) consumption and complexity of SCA exceeds typical spacecraft communication resources.

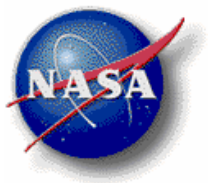
- Status of Space Application
  - NASA and JTRS have MOU to maintain collaborative partnership
  - SCA is being used for SATCOM appls (i.e. wideband ground based radios)
  - Optimized SCA implementations may apply to space-based radios



Architecture Requirements, Mission Drivers, GPS Architecture Example

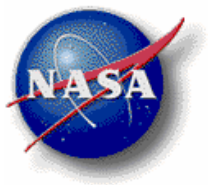
Sample High Level Architecture Requirements

- SDR HW and SW architecture shall be open and provide published interfaces and description of allowable operations and information exchanged among components across the defined interfaces.
- SDR architecture shall enable software portability and abstraction layers while minimizing required resources (e.g. power, mass).
- SDR architecture shall minimize specific application and/or implementation approaches to provide flexibility for different mission classes and available resources.
- SDR architecture shall define minimum set of hardware and software interface definitions to enable technology infusion over time.
- Be HW and SW configurable for multiple mission requirements and/ or accommodation of new requirements (pre-launch and post-launch; when operational and non-operational).



Sample High Level Architecture Requirements (Cont'd)

- Recover the pre-existing operational configuration when a run-time exception fault is detected during reconfiguration of the SDR
- Enable incremental uploads (at module / shared library level) of both software and firmware
- Scale to accommodate different transceiver form factors
- Enable fully autonomous operations, if its software/firmware load is capable of such operations
- Enable scaleable system networking and network services
- Enable the transceiver from operating multiple channels, in multiple frequency bands



Sample of STRS Architecture Drivers

- Resources available to space based radios significantly less than terrestrial based radios
  - Need architecture scalable to different mission resources
- Minimize architecture specifications (“keep it simple”)
  - Abstract waveform from platform
  - Let provider implement/optimize OE that supports standard SW interface
- Define APIs up front – (need path to standardization – work with SCA WG)
  - Leverage SCA, SDRF SCA
  - Commercial
- Divide mission radios up into platform classes (initial cut provided in document-warrants discussion)
- Emphasis on waveform implementations on specialized hardware , not GPP
- Wanted to address hardware architecture aspects (HID) to enable more interoperability and reduce costs of qualification processes
- Less or no emphasis for dynamic deployment and hardware discovery

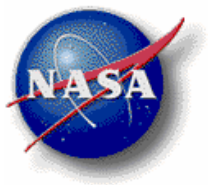
Sample of STRS Architecture TBD's

Technical

- Hardware Interface Descriptions – what's publishable and what remains proprietary?
- List of waveforms
  - Not yet formalized, but…
  - SNUG, GNUG, DSN
  - Exploration waveforms TBD
- Security requirements TBD

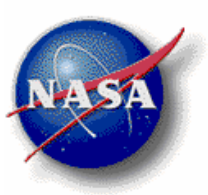
Non-Technical

- Technical Advisory Group
  - Recommendations on how to implement
- Unique aspects of space business case
  - Different market volume than DoD, Commercial, Public Safety
  - Software model or hardware model
- Openness of architecture – NASA may have to address ITAR issues in the actual STRS Specification Document(s) as did JTRS.

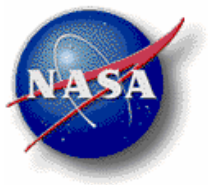


Platform Class Characteristics

- Receive
  - Frequency Bands
  - Number of Simultaneous Signals
  - Maximum signal bandwidth
- Transmit
  - Maximum Number of Frequency Bands
  - Maximum Transmit Bandwidth
  - Maximum Transmit Data Rate
  - Transmit Power Level
- Coding
  - Convolutional
  - Turbo
  - Reed Solomon
  - Low Parity Density Codes
- Network
  - Low Rate Port
  - IP Network Routing
  - High Rate Port
- Security
  - Command Authentication
  - Transmit Encryption
  - Network Security
- Spacecraft Input/Output
  - RS422
  - 1553/1773
  - Ethernet
  - Spacewire
  - Firewire



STRS Architecture Description Overview



STRS Hardware Modules

- **General Processing Module (GPM)** – Consists of the general purpose processor, appropriate memory, spacecraft bus (e.g. MILSTD-1553), interconnection bus (e.g. PCI), and the components to support the radio configuration.

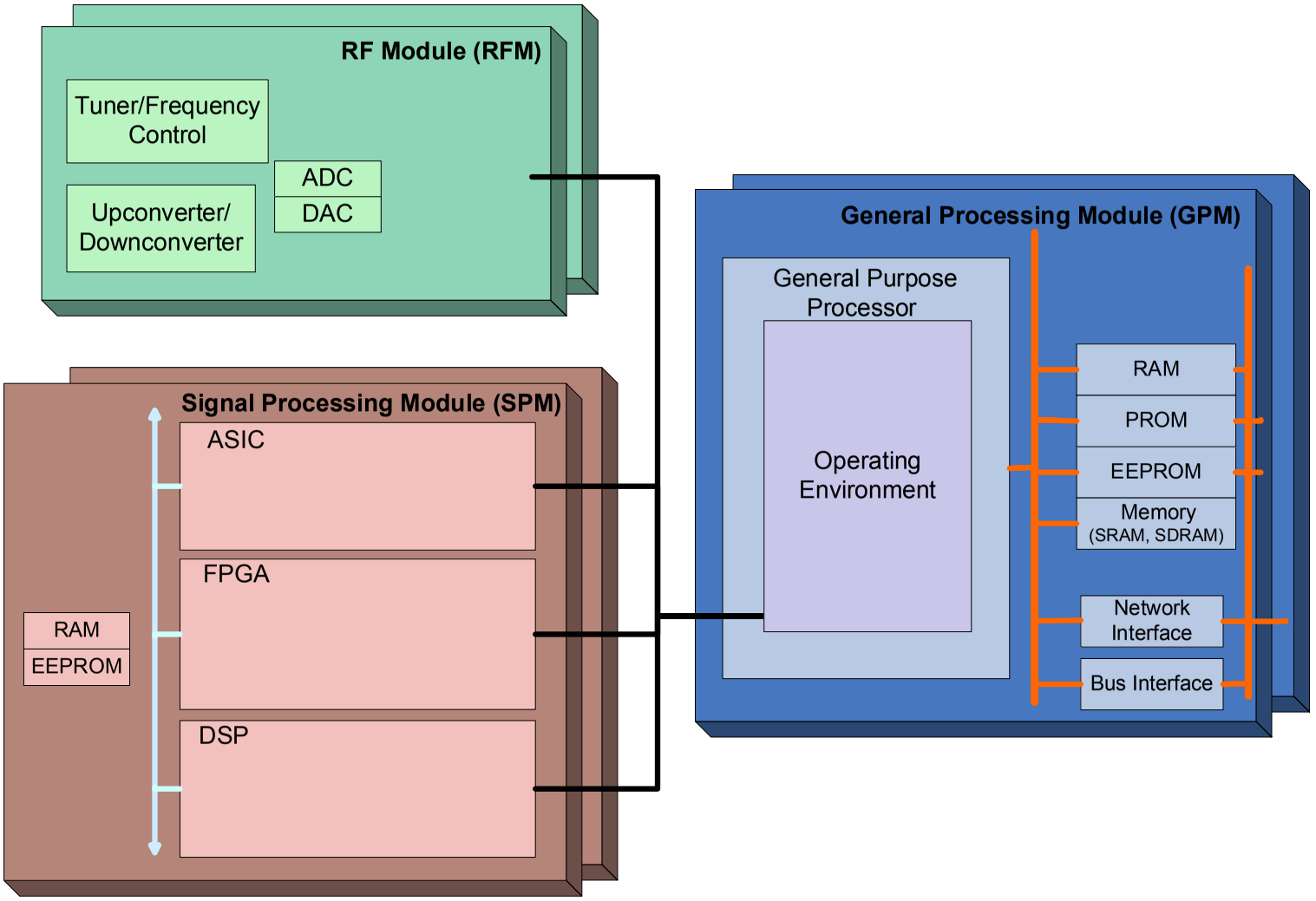
- **Signal Processing Module (SPM)** –Contains the implementations of the signal processing used to handle the transformation of received digital signals into data packets and/or the conversion of data packets into digital signals to be transmitted. Components include ASIC's, FPGA's, DSP's, memory, and connection fabric/bus (e.g. PCI, flex-fabric)

- **RF Module (RFM)** –Handles the RF functionality to provide the SPM with the receive digital signal and transmit the output digital signal. Its associated components include filters, RF switches, diplexer, LNAs, power amplifiers, and data converters. This module handles the interfaces that control the final stage of transmission or first stage of reception of the wireless signals, including antennas.

- **Network Module (NM) –** The architecture supports Consultative Committee for Space Data Systems (CCSDS) and Internet Protocols (IP) and networking functions.

- **Security Module (SEC)** –The detail of this module are TBD.

Hardware Modules Functional Diagram



STRS HW Rule Set Sample

General Purpose Module

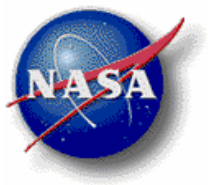
- Provides programmable GPIO to support
  - Interrupt source/sink
  - Application data transfer
- Provides control/configuration interfaces
- Provides spacecraft bus Interface
- The GPM configuration file must describe the hardware environment for the STRS architecture. It will identify the existence of the different hardware modules and their associated configuration file that will allow the architecture to instantiate drivers and test applications.

Signal Processing Module

- Integrates with GPM bus for
  - FPGA/DSP program transfer (Waveform Instantiation)
  - bus for signal data transfer
  - GPIO to generate and receive interrupts
- Direct IO for high rate mission data (bypass GPM)
- The SPM configuration file provides the STRS infrastructure with information on what resources are provide by the module as well as the resources that must exist on the GPM for access to the devices.

Hardware Interface Definition

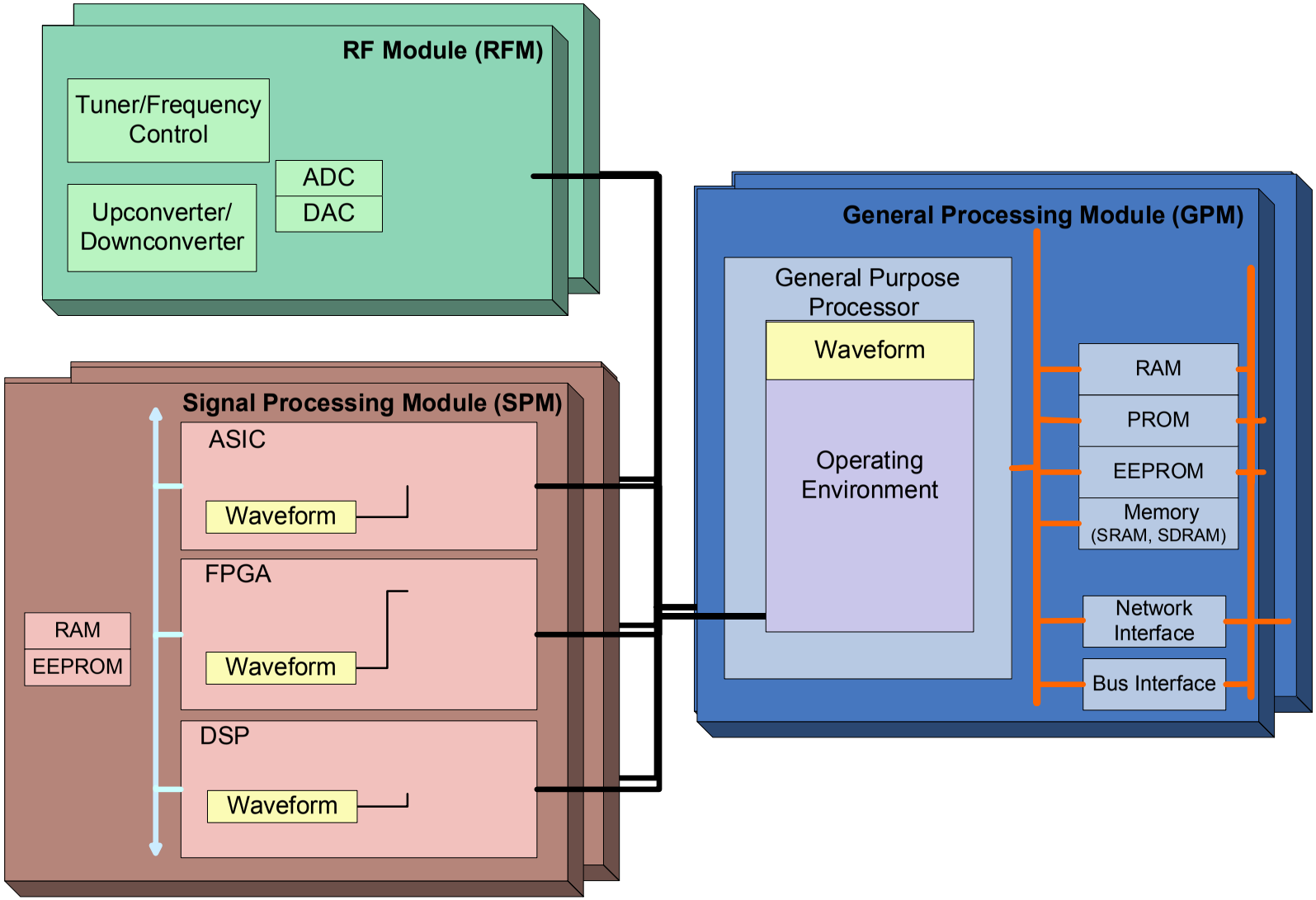
- HID shall be published for each module so 3rd party developers have the structure under which they can develop new modules
- The HID will specify bus configurations and IO pin assignments for the backplane.

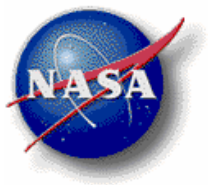


Waveform Component Allocation

- STRS waveforms are comprised of software components with signal processing algorithms necessary to transmit and receive messages.
- Waveform components allocated to processing (GPP) and specialized hardware (SPM).
- High level services used to control and monitor waveform (via GPP)
- Operating Environment provides access to processor and specialized hardware for waveform processing
  - OE implements the system management, device control, data transfer functions
  - Leverages commercially available products and functions
- Specialized Hardware (SPM) contains and executes firmware enabling higher rate processing within the Software Defined Radio

Waveform Component Allocation Functional Diagram





Interface Descriptions

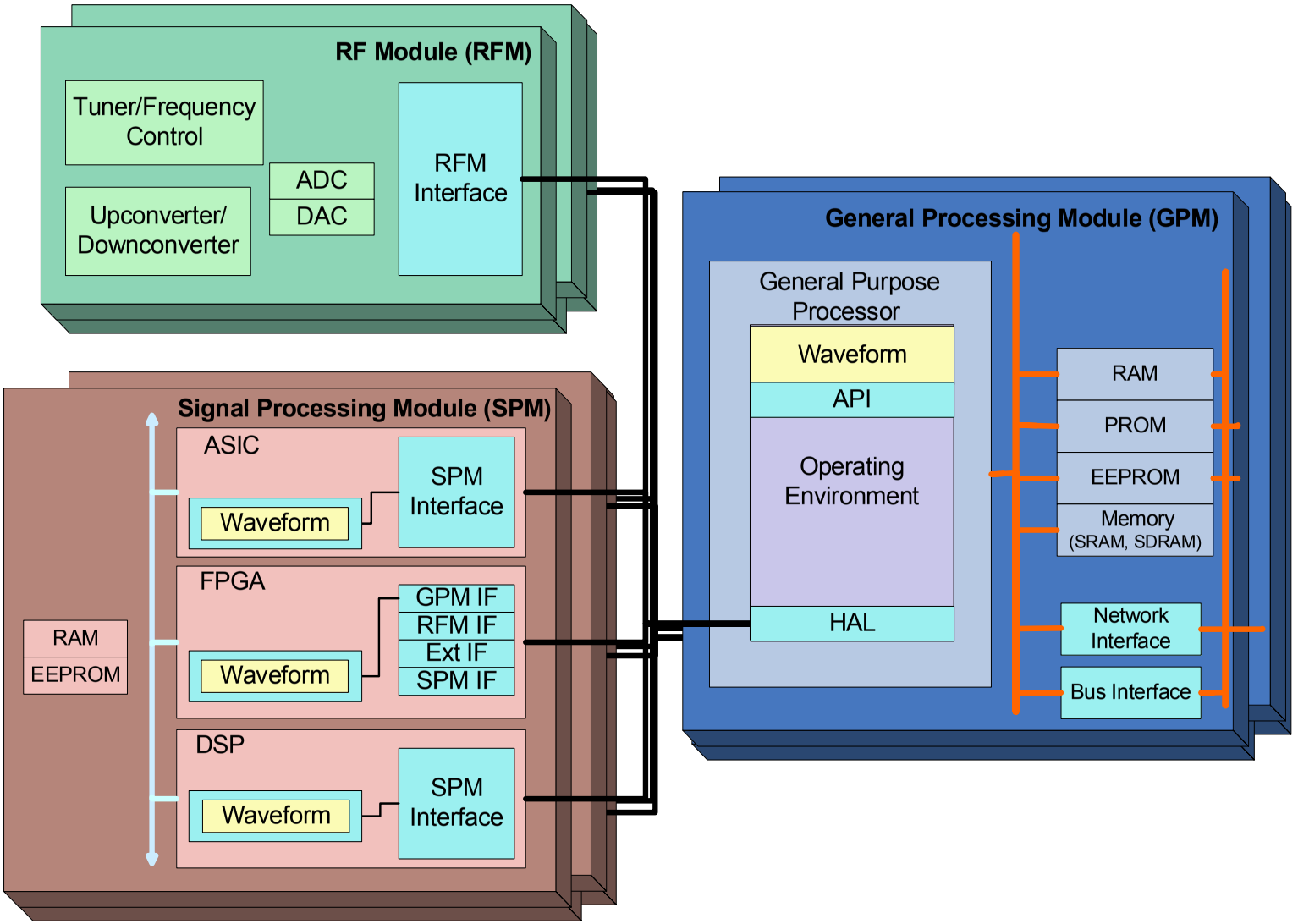
Hardware

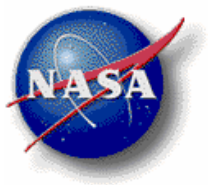
- Module interfaces abstracts and defines the module functionality for data flow to waveform components to enable multiple vendors to provide different modules or add modules to existing radios.
- The electrical interfaces, connector requirements, and physical requirements are specified by the platform provider.

Software

- Layers help define interfaces between components
- Certain layers separate SW from SW and other SW from HW
- APIs defined by the architecture separate waveform from operating environment for waveform portability
- The HAL API (Application Programmer's Interface) is published so that specialized hardware made by one company may be integrated with the STRS Infrastructure made by a different company
  - Includes a description of each method/function used, including its calling sequence, return values, an explanation of its functionality, any preconditions before using the method/function, and the status after using the method/function
  - Contains information about the underlying hardware such as address and data interfaces, interrupt input and output, power connections, as well as plus other control and data lines necessary to operate in the STRS platform environment (firmware code portability)
- The STRS Infrastructure will use the HAL information to initialize the hardware drivers such that the control and data messages will be appropriately delivered to the module.

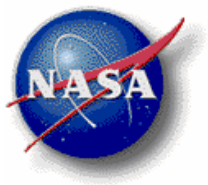
Waveform Component Allocation Functional Diagram





STRS SW Rule Set Sample

- All STRS waveform applications will use the STRS Application Programmer Interfaces (API) suite to instantiate and control the waveform functionality
    - The STRS API provides the interfaces that allow applications to be instantiated and use platform services. These APIs also enable communication between waveform and application components
- The API layer shall consist of published STRS APIs while leveraging a defined subset of industry standard POSIX APIs
- Operating environment consists of STRS infrastructure, real time OS, board support packages, hardware abstraction layers.
- The operating environment (via STRS infrastructure) shall implement the STRS APIs and system management, device control, data transfer functions
- STRS configuration files shall contain platform and waveform specific information to allow customization of installed waveforms. The configuration file contains the parametric values for configurable components as well as filenames for loadable devices (FPGA, DSP, etc.)



Layer Definitions

Layer	Description
Application / Services	This layer encompasses the waveform and application entities that use the services of the STRS infrastructure to perform the desired functionality.
STRS API	The STRS APIs provide a consistent interface for executing the applications and services.
STRS Infrastructure	The STRS infrastructure implements the behavior and functionality identified by the STRS APIs.
POSIX OS	The STRS defines a minimum POSIX AEP for the allowed OS services. This layer can either consist of a compliant POSIX RTOS, or by a POSIX AEP Abstraction in conjunction with a non-compliant RTOS.
Direct Service Support	This layer identifies the ability for the STRS infrastructure to have a direct interface to the hardware drivers on the platform.
HW Drivers	The hardware drivers provide the platform independence to the software and infrastructure by abstracting the physical hardware interfaces into a consistent Device Control API.
HAL/HID/BSP	The Hardware Abstraction Layer (HAL) works in conjunction with the HW drivers to provide a platform independent view for the infrastructure and applications. The Hardware Interface Description (HID) defines the physical interfaces that allow third party hardware developers to integrate their products with a specific STRS platform. The Board Support Package (BSP) provides the hardware abstraction of the GPM module for the POSIX-compliant Operating System.
Communication Equipment	Physical layer of the hardware modules existing on the STRS Platform.