MICROTEL LLC

aerospace software and computer systems

SWTSv3

SpaceWire Test Set
Version 3

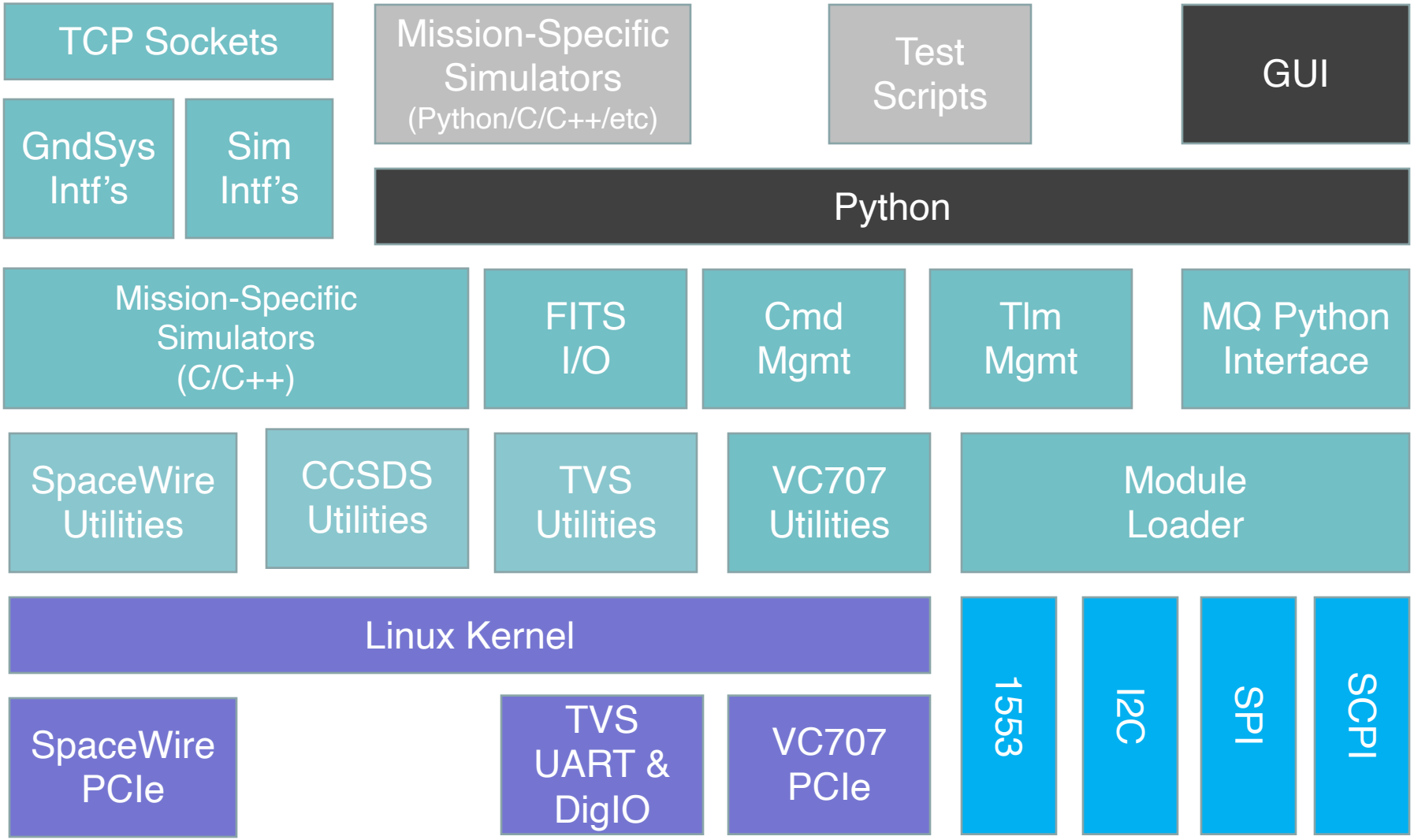
Background

- **SWTS = SpaceWire Test Set**
- **Originally developed as a SpaceWire Hardware Interface Tester**
- **Over time it has evolved into a full featured simulation and test environment**
  - Physical Interface Testing
  - Logical Interface Testing
  - High Fidelity Simulation of Flight Components
    - Instruments, C&DH Systems, Solid State Recorders, Comm Systems, etc
  - Integration with all major Ground Systems used at Goddard
    - ASIST, ITOS, ECLIPSE
- **Used extensively on nearly every mission developed at Goddard since 2003 for**
  - Flight Hardware development and testing
  - Flight Software development and testing
  - Box and System level I&T
  - Post-launch support
- **Flexible and Expandable for New/Custom Applications**
  - Multiple interfaces beyond SpaceWire supported
  - Multiple API's available for 3rd-party software/hardware expansion

SWTS Evolution

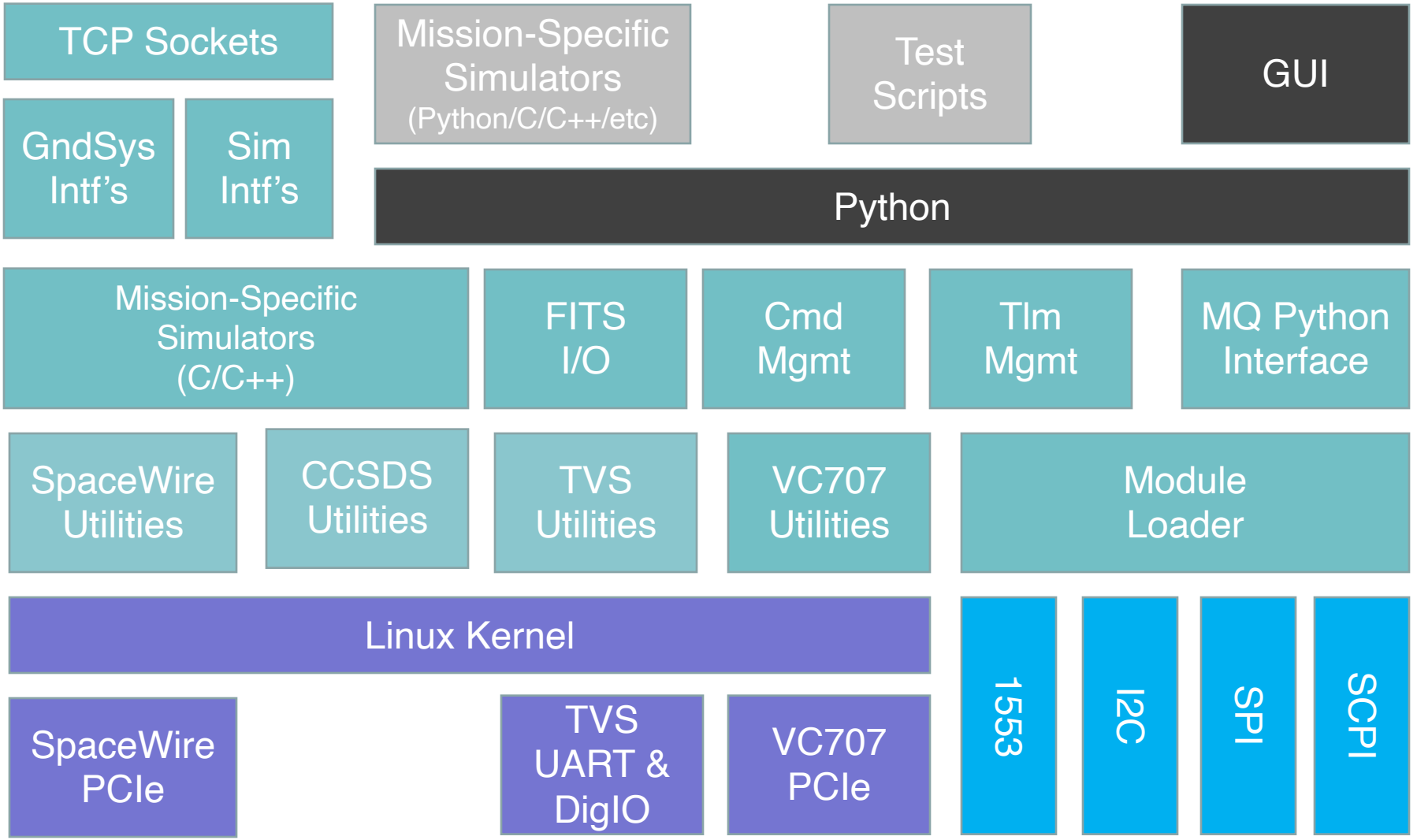
- **SWTSv1**
  - Developed: 2003
  - Status: Obsolete.
  - OS: Windows XP
  - SpW HW: COTS USB
  - Ground Systems: ECLIPSE
  - Missions: JWST
- **SWTSv2**
  - Developed: 2004
  - Status: End of Life
  - OS: Windows XP/7
  - Spw HW: Custom PCI Card
  - Ground Systems: ECLIPSE, ASIST
  - Missions:
    ASTRO-H, ATLAS, GPM, JPSS, JWST, LCRD, LRO, MMS, MOMA, and others
- **SWTSv3**
  - Developed: 2017
  - Status: Active Development
  - OS: Linux
  - Spw HW: Custom PCIe Card
  - Ground Systems: ECLIPSE, ASIST, ITOS
  - Missions:
    JWST, OCI, PACE, RESTORE-L, WFIRST, XRISM, and others
  - New Hardware Support:
    - 1553
    - Xilinx VC707
    - Analog I/O
    - SCPI
    - TVS
      - Discrete Digital I/O
      - UARTS
      - Pulse In/Out
      - I2C
      - SPI
      - And others
  - New Software APIs for customizing
    - Loadable Modules (C/C++)
    - POSIX MQ Interfaces (C/C++, Python, etc)

SWTSv3 Software Architecture



SWTSv3 Software Architecture

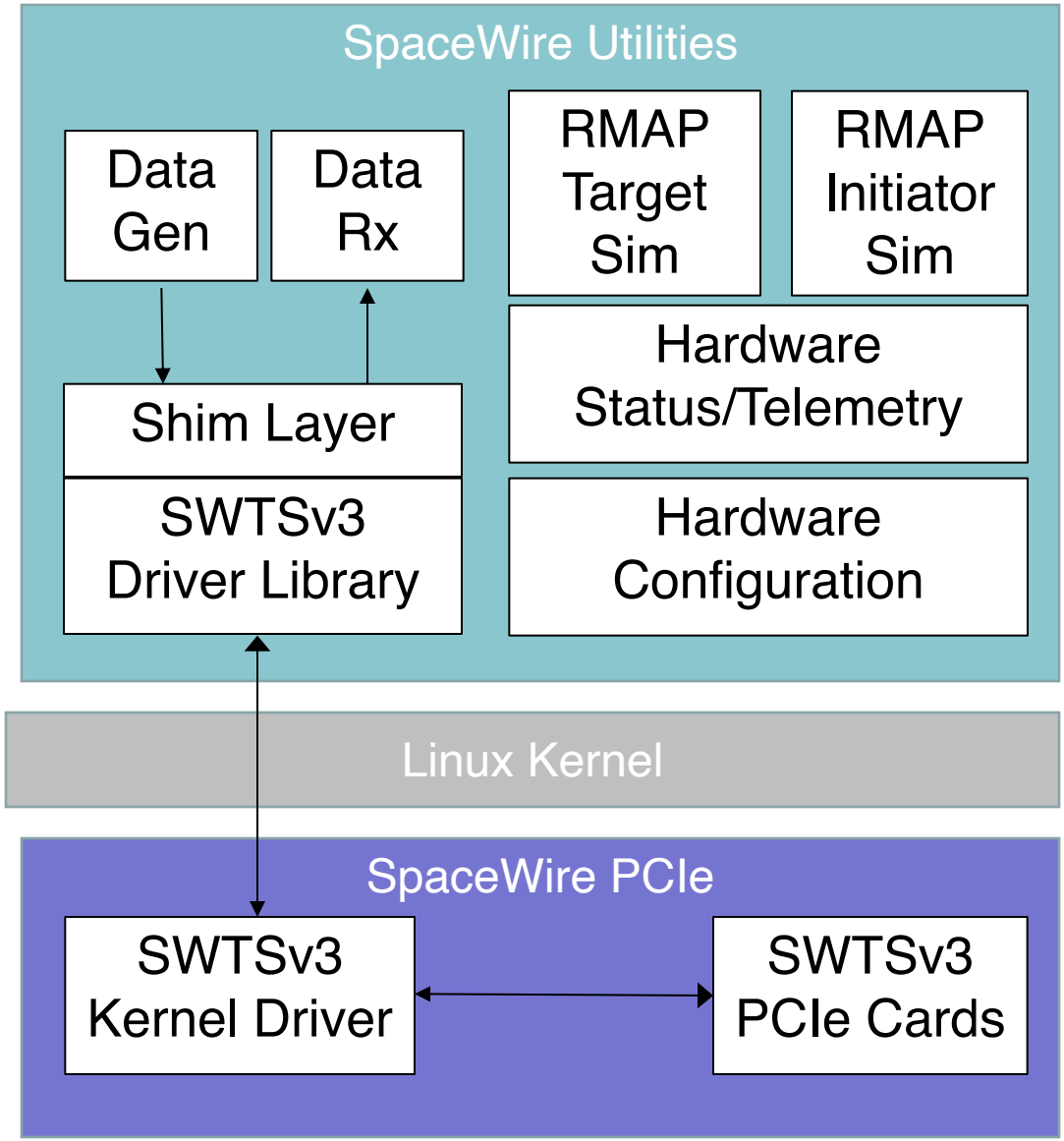
Deep Dive into one component of SWTS software as example…



SWTSv3 Software Architecture

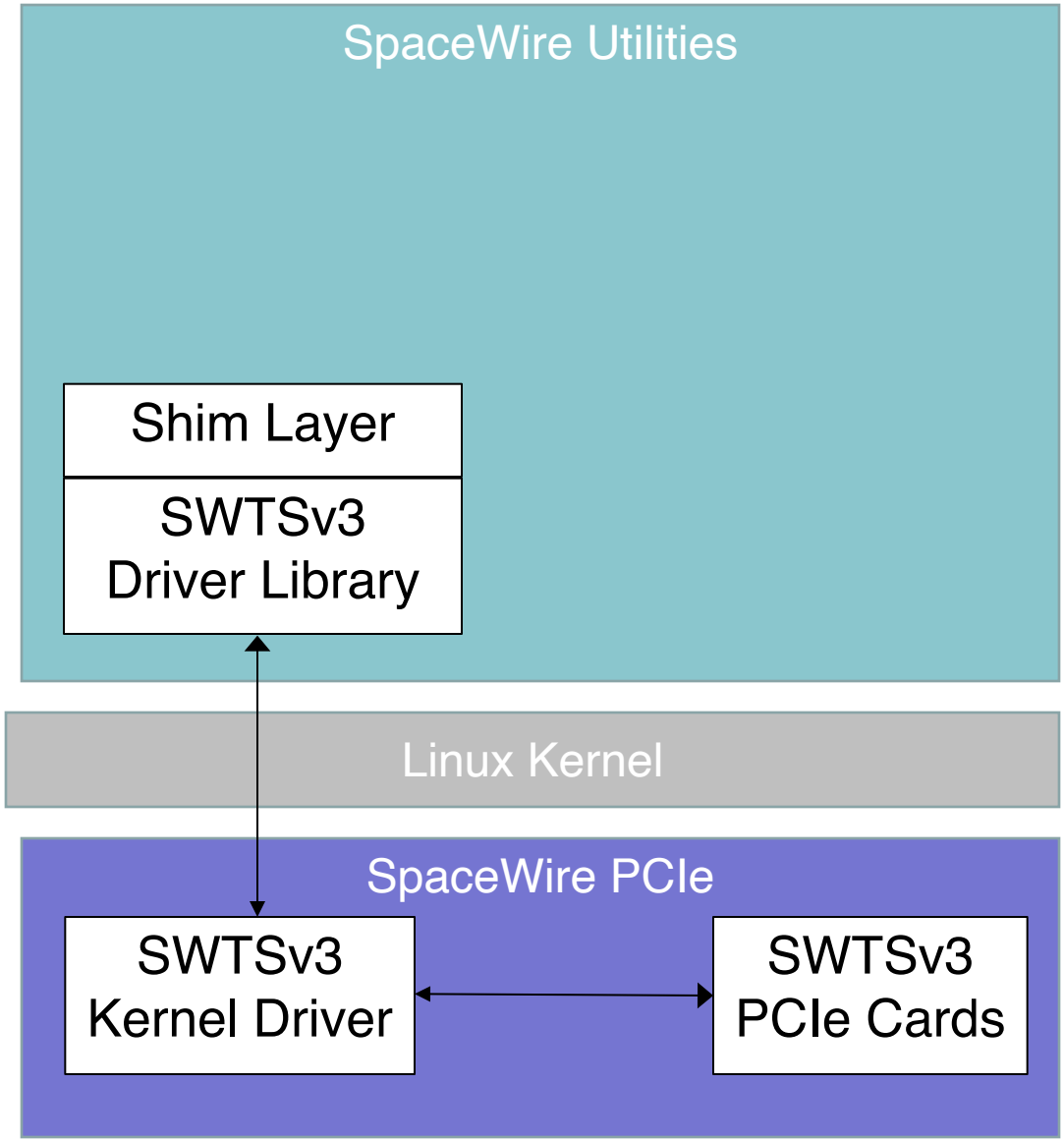
Deep Dive into one component of SWTS software as example…

SpaceWire Architecture

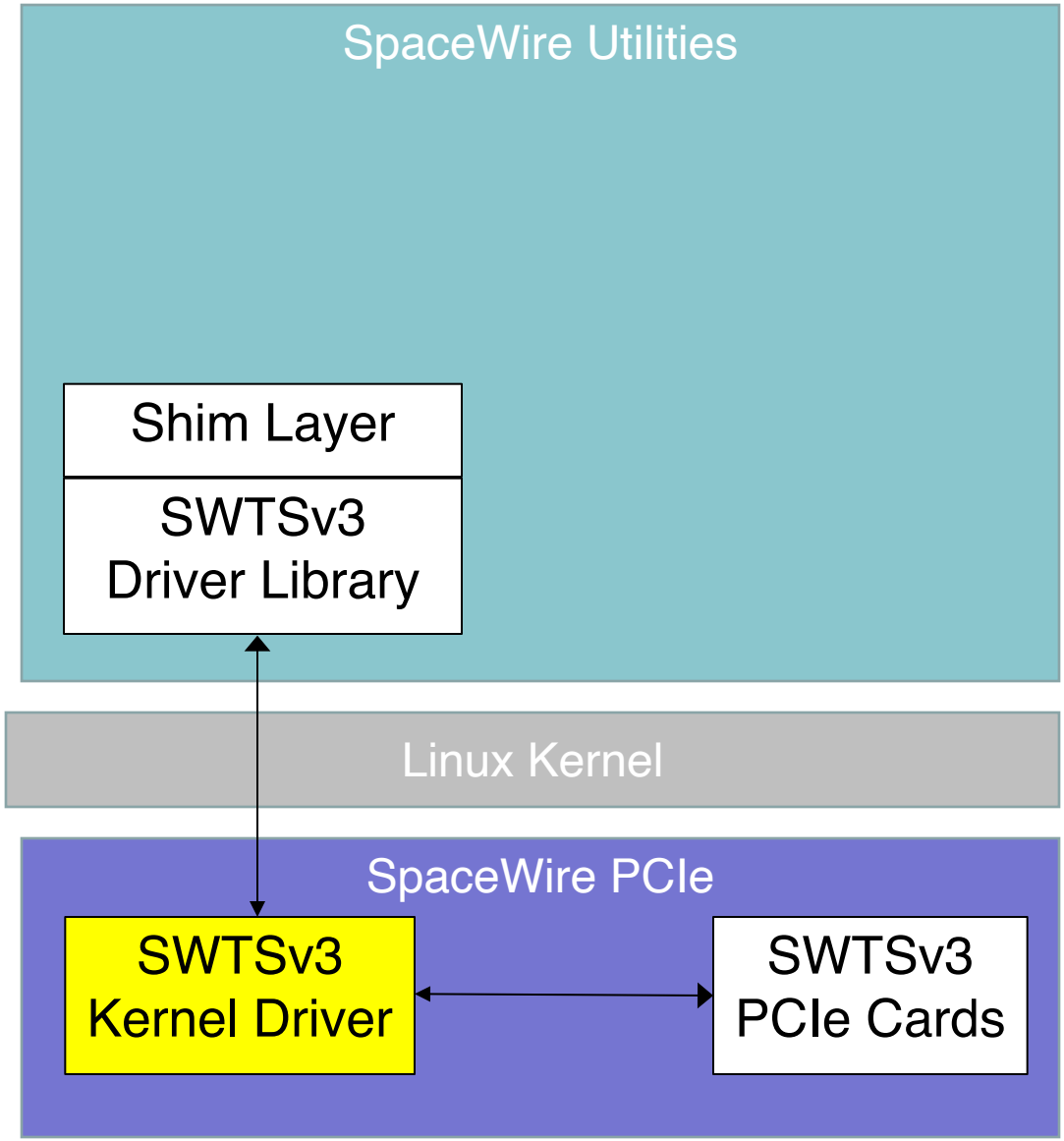


- RMAP Target Sim:
  - Simulates RMAP Targets
- RMAP Initiator Sim:
  - Simulates RMAP Initiators
- Hardware Status/Telemetry:
  - Retrieves Status from Driver Lib
  - Maps to Telemetry
  - Generates Hardware Events
- Hardware Configuration:
  - Actuates User Initiated Changes
- Data Gen:
  - Generates TX Packets/Buffers
    - Play from File with ns timing
    - Send from GUI
    - Generate Test Pattern
    - Forward from Simulator
- Data Rx:
  - Processes RX Packets/Buffers
    - Record to File with ns timing
    - Display to GUI
    - Compare w/ Test Pattern
    - Forward to Simulator
- Shim Layer
  - Moves Pkts/Buffers to/from Queues
  - Normalizes Driver Calls
- SWTSv3 Library APIs:
  - Hardware Init/Config/Status
  - Packet/Buffer TX/RX (Intr/DMA)
- SWTSv3 Kernel Driver:
  - PCI Discovery/Configuration
  - Register/Memory I/O Mapping
  - DMA Buffer Allocation and Mapping
  - Kernel-Level Interrupt Handling
- SWTSv3 PCIe Cards (Up to 3):
  - Xilinx Artix-7 FPGA
  - 8 SpaceWire Ports (or 4 redundant)
  - 4 GB SDRAM
  - DMA Controller
  - VCXO, Temp/Volt Sensors, etc.

SpaceWire Driver and Shim



SpaceWire Driver and Shim



SWTSv3 PCIe Kernel Driver

- **swts_init – called by OS on startup or module load.**
  - Initializes data structures
  - Calls ***alloc_chrdev_region*** to allocate character driver number
  - Calls ***cdev_add*** to add character driver to system
  - Calls ***pci_register_driver*** to inform OS which cards this module drives
- **swts_exit – called by OS on shutdown or driver unload**
  - Calls ***cdev_del*** to remove character driver from system
  - Calls ***unregister_chrdev_region*** to free character driver number
  - Calls ***pci_unregister_driver*** to inform OS this driver is no longer present

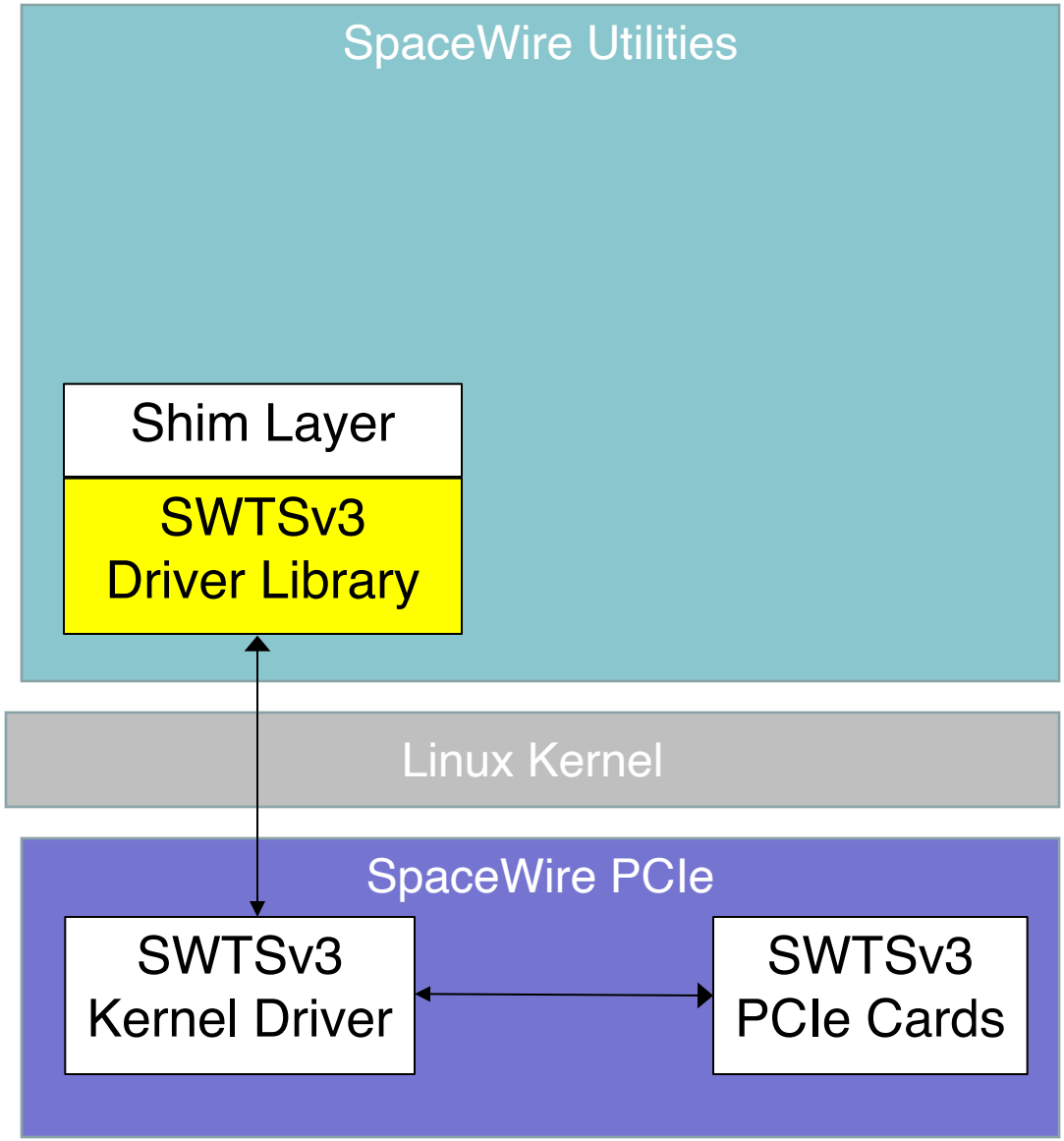
SWTSv3 PCIe Kernel Driver (cont.)

- **probe – called by OS once for each PCI card found in chassis**
  - Calls ***pci_resource_start, pci_resource_len, pci_resource_flags, request_mem_region, ioremap_nocache, pci_ioremap_bar, request_region, and ioremap_nocache*** to configure memory windows for each BAR on each card
  - Calls ***kmalloc, pci_map_single, pci_set_dma_mask*** to allocate and map DMA buffers
  - Calls ***pci_enable_device, pci_set_master*** to enable card
  - Calls ***pci_enable_msi, request_irq*** to configure and enable interrupts for card
- **remove – called by OS once for each PCI on shutdown (or card remove in hot-swap)**
  - Calls ***disable_irq, free_irq, pci_disable_msi*** to disable and free interrupts for card
  - Calls ***iounmap, release_mem_region*** to free memory windows for each BAR on card
  - Calls ***pci_unmap_single, kfree*** to unmap and deallocate DMA buffers for card
  - Calls ***pci_disable_device*** to shutdown device

SWTSv3 PCIe Kernel Driver (cont.)

- **swts_ioctl – primary interface between library and kernel driver**
  - READ
    - Uses BAR information to Map Card address to System Address
    - Calls ***readl*** or ***readq***
  - WRITE
    - Uses BAR information to Map Card address to System Address
    - Calls ***writel*** or ***writeq***
  - GET_DMA_BUS_ADDRS _IO
    - Calls ***upper_32_bits, lower_32_bits*** to map DMA buffer addresses to PCI Bus Addresses (Note: this information is used by the application library to program the DMA controller)
  - GET_RXDMA_DATA
    - Calls ***copy_to_user*** to move data from DMA buffer to user space (Note: this is used by the application library to get the data after a RX DMA is completed)
  - SET_TXDMA_DATA
    - Calls ***copy_from_user*** to move data from user space to DMA buffer (Note: this is used by the application library to set the data before a TX DMA is started)
- **swts_read**
  - Used by application library interrupt service thread
  - Blocks until interrupt is received, and returns contents of interrupt status register

SpaceWire Driver and Shim



- Shim Layer
  - Moves Pkts/Buffers to/from Queues
  - Normalizes Driver Calls
- SWTSv3 Library:
  - Hardware Init/Config/Status calls
  - Packet/Buffer TX/RX (DMA)
- SWTSv3 Kernel Driver:
  - PCI Discovery/Configuration
  - Register/Memory I/O
  - DMA Buffer Allocation
  - Basic Interrupt Support
- SWTSv3 PCIe Cards (Up to 3):
  - 8 SpaceWire Ports
  - 1 GB SDRAM
  - DMA Controller
  - VCXO, Temp/Volt Sensors, etc.

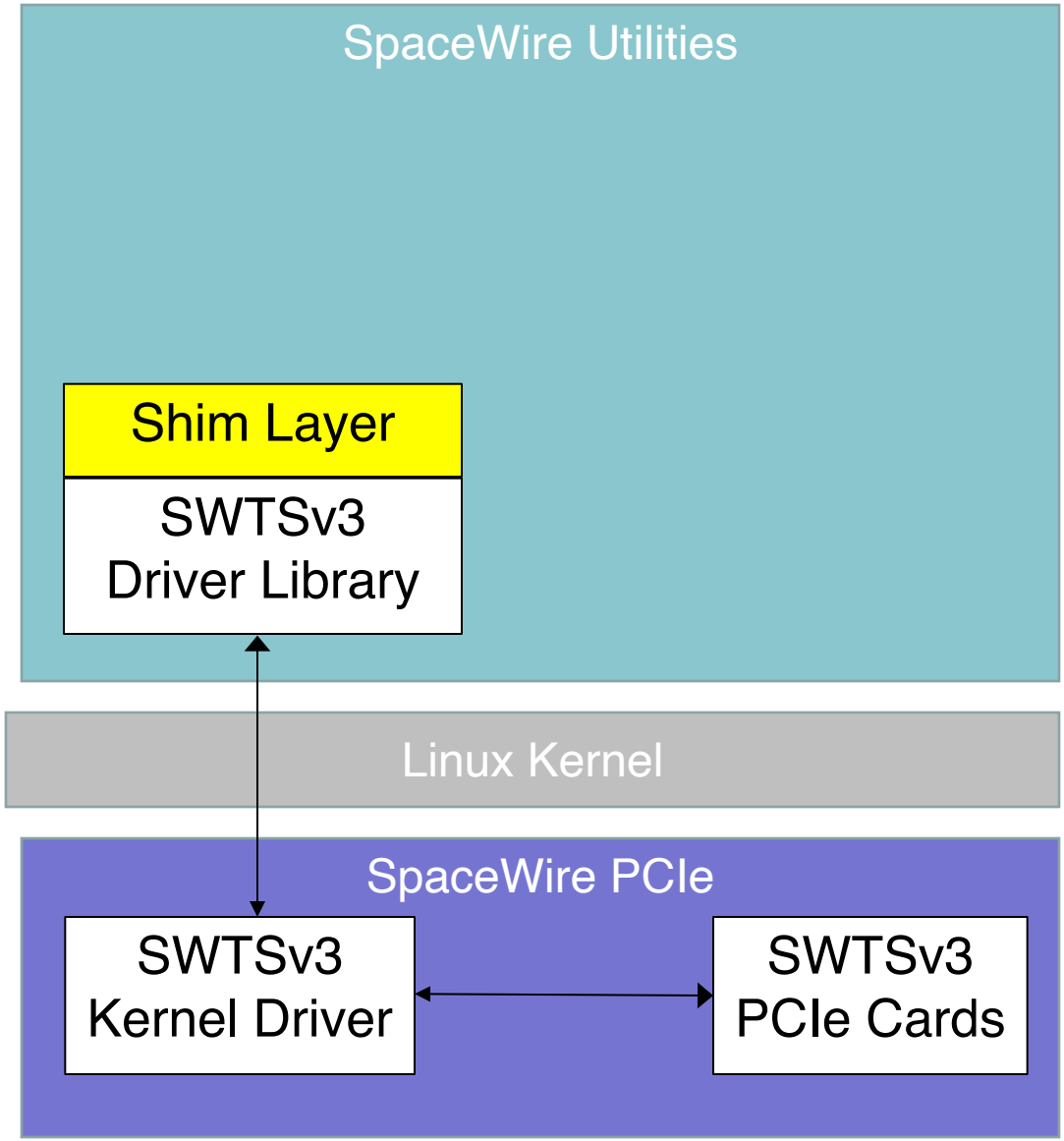
SWTSv3 Application Library

- **swts_lib_init**
  - Allocates memory, semaphores, pthreads, etc.
  - Initializes memory/data-structures
  - Initializes hardware via ioctl WRITEs
  - Gets DMA buffer PCI addresses via ioctl GET_DMA_BUS_ADDRS _IO
- **swts_lib_start**
  - starts swts lib receive threads
  - starts swts lib interrupt service threads
- **swts_lib_receive_thread**
  - Periodically polls swts hardware buffers for received data via ioctl READs
  - Reads data into user-space buffer from card via DMA using ioctl READs, WRITEs, and GET_RXDMA_DATA
  - Calls application-provided callback function to announce receipt of data
- **swts_lib_interrupt_thread**
  - blocks on device “read” until interrupt received
  - Uses contents of returned interrupt status register to announce interrupt to other library components (primarily DMA completion).

SWTSv3 Application Library (cont.)

- **swts_lib_tx_space_available**
  - Returns amount of space available on card for TX via ioctl READs
- **swts_lib_tx_buffer**
  - Transfers data from user-space buffer to card via DMA using ioctl READs, WRITEs, and SET_TXDMA_DATA
- **swts_lib_config_board**
  - Passes in data structure of board-level configuration information
- **swts_lib_conf_port**
  - Passes in data structure of port-level configuration information
- **swts_lib_status**
  - Fills out data structure with complete board status (counters, link state, etc.)
- **swts_lib_write_pkt_hdr/ftr**
  - Used by application to write packets to buffer
- **swts_lib_extract_pkt**
  - Used by application to extract packet from buffer

SpaceWire Driver and Shim

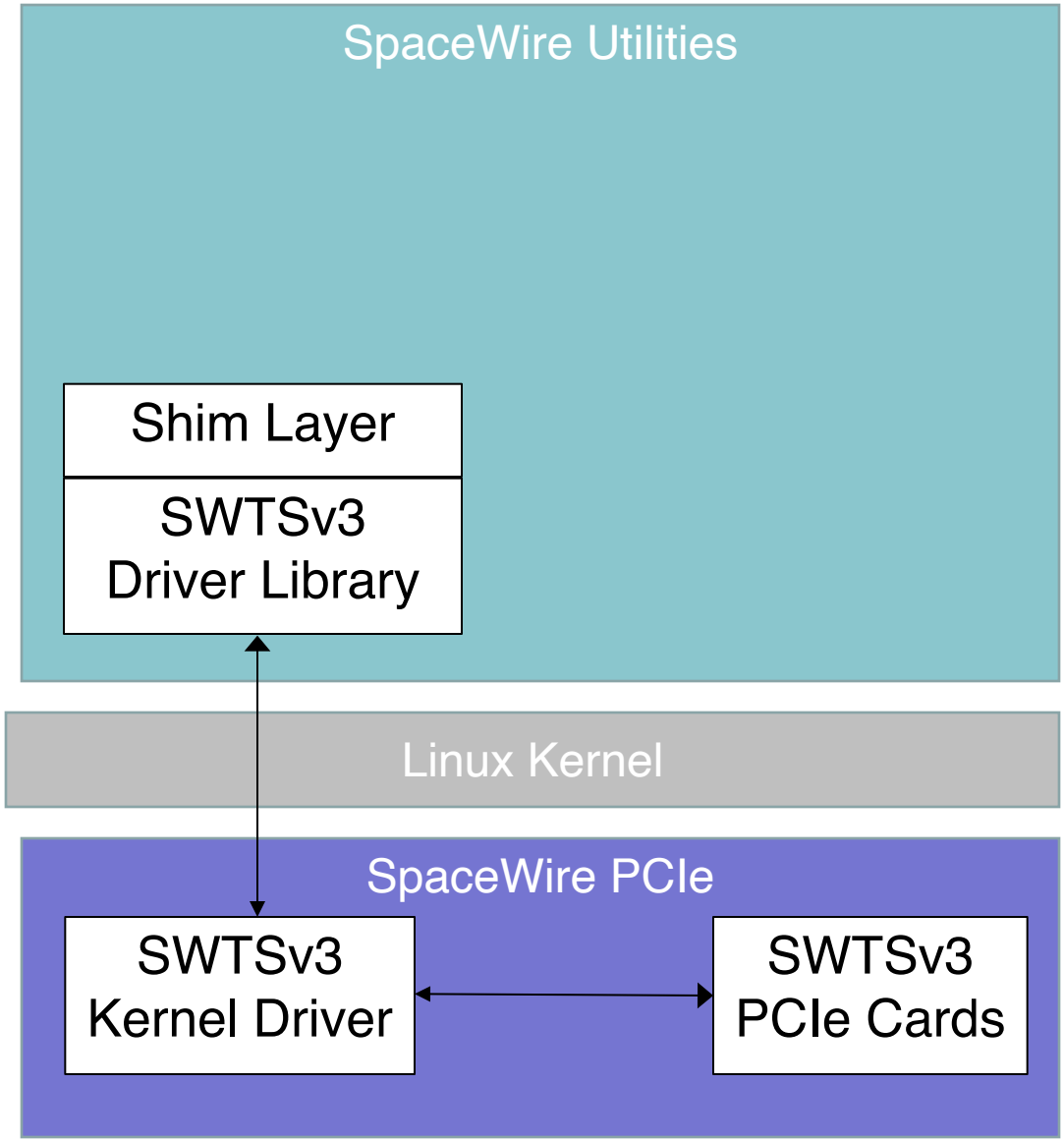


- Shim Layer
  - Moves Pkts/Buffers to/from Queues
  - Normalizes Driver Calls
- SWTSv3 Library:
  - Hardware Init/Config/Status calls
  - Packet/Buffer TX/RX (DMA)
- SWTSv3 Kernel Driver:
  - PCI Discovery/Configuration
  - Register/Memory I/O
  - DMA Buffer Allocation
  - Basic Interrupt Support
- SWTSv3 PCIe Cards (Up to 3):
  - 8 SpaceWire Ports
  - 1 GB SDRAM
  - DMA Controller
  - VCXO, Temp/Volt Sensors, etc.

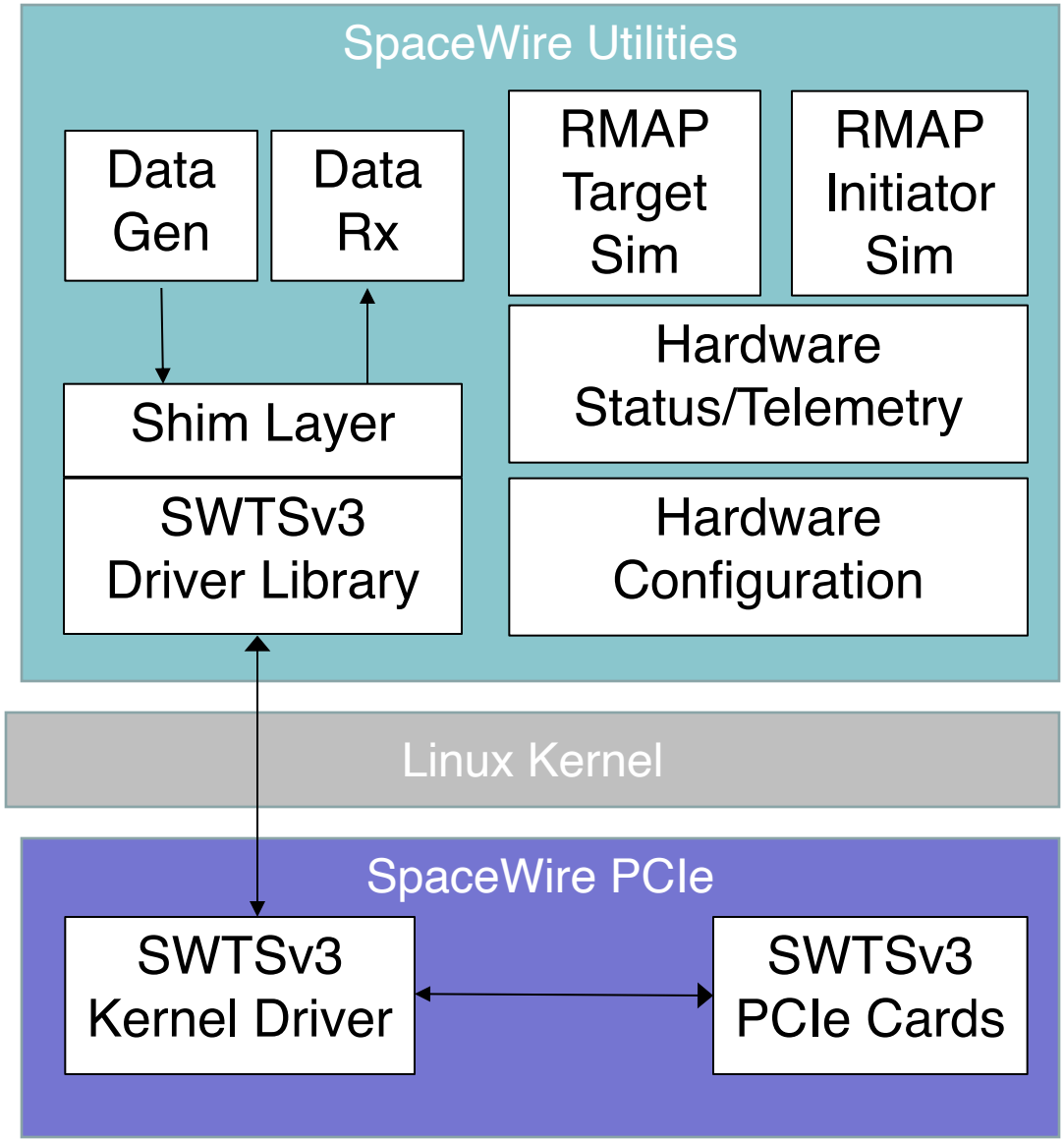
Shim Layer

- **Shim layer is “software connector” that simplifies future additional SpW hardware**
  - Implemented with jump table structure for
    - Tx_space_available
    - Tx_buffer
    - Config_board
    - Config_port
    - Status
- **swts_shim_tx_thread**
  - Monitors system TX queues for outgoing traffic
  - Monitors Hardware queues for space via tx_space_available
  - When there’s a match, transmit buffer via tx_buffer
- **Swts_shim_rx**
  - Callback provided to swts_lib_receive_thread
  - Forwards received buffers into RX queue

SpaceWire Driver and Shim



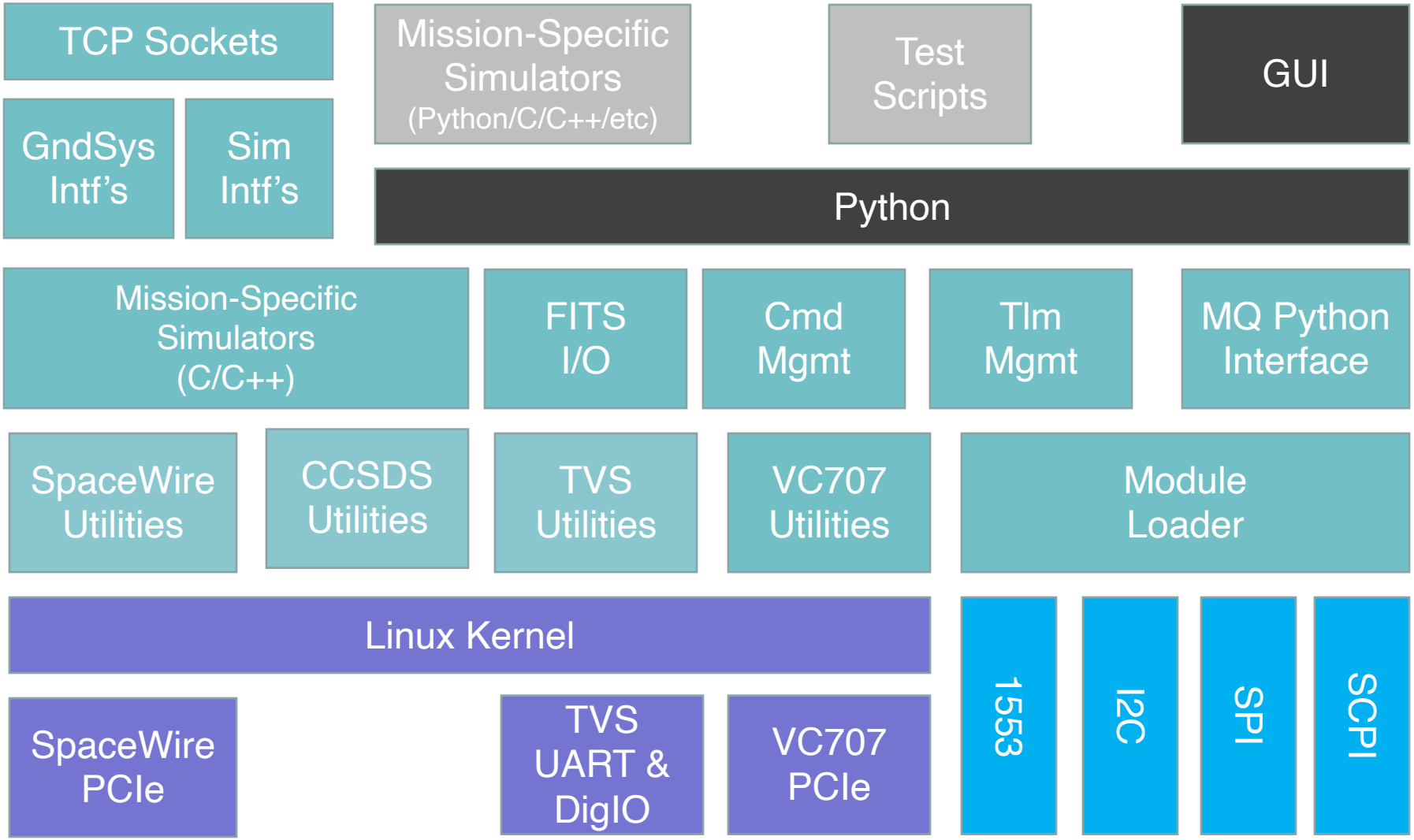
SpaceWire Architecture



- RMAP Target Sim:
  - Simulates RMAP Targets
- RMAP Initiator Sim:
  - Simulates RMAP Initiators
- Hardware Status/Telemetry:
  - Retrieves Status from Driver Lib
  - Maps to Telemetry
  - Generates Hardware Events
- Hardware Configuration:
  - Actuates User Initiated Changes
- Data Gen:
  - Generates TX Packets/Buffers
    - Play from File
    - Send from GUI
    - Generate Test Pattern
    - Forward from Simulator
- Data Rx:
  - Processes RX Packets/Buffers
    - Record to File
    - Display to GUI
    - Compare w/ Test Pattern
    - Forward to Simulator
- Shim Layer
  - Moves Pkts/Buffers to/from Queues
  - Normalizes Driver Calls
- SWTSv3 Library:
  - Hardware Init/Config/Status calls
  - Packet/Buffer TX/RX (DMA)
- SWTSv3 Kernel Driver:
  - PCI Discovery/Configuration
  - Register/Memory I/O
  - DMA Buffer Allocation
  - Basic Interrupt Support
- SWTSv3 PCIe Cards (Up to 3):
  - 8 SpaceWire Ports
  - 1 GB SDRAM
  - DMA Controller
  - VCXO, Temp/Volt Sensors, etc.

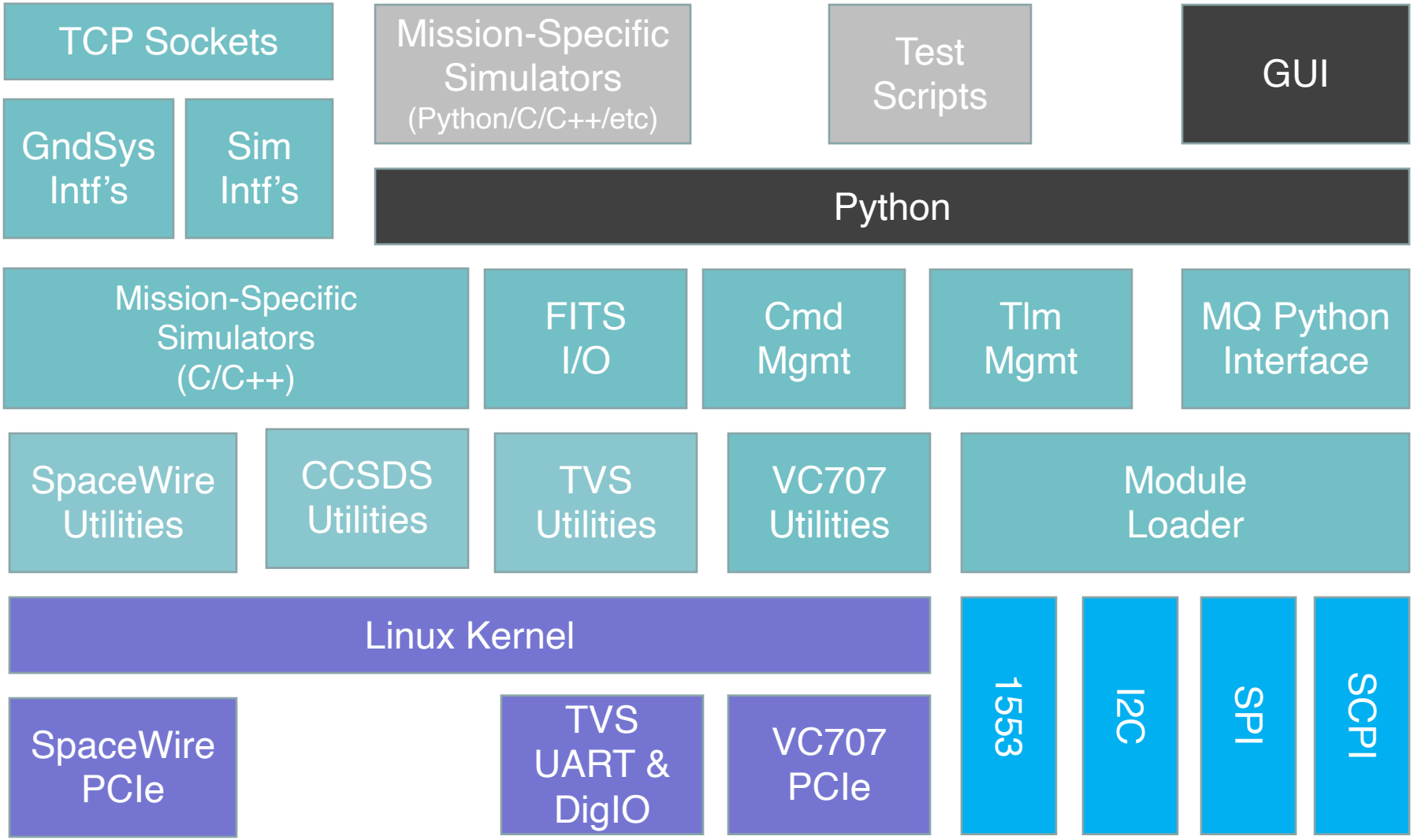
SWTSv3 Architecture

SWTSv3 Software Architecture



SWTSv3 Software Architecture

Note: can provide equally detailed breakdowns of other software components



Summary

- **SWTSv3 is the 3rd Generation SpaceWire Test Set**
  - Direct port of SWTSv2 code used on almost every mission at Goddard since 2003
    - ASTRO-H, ATLAS, GPM, JPSS, JWST, LCRD, LRO, MMS, MOMA, and others
  - SWTSv3 is currently deployed on almost every mission at Goddard today
    - JWST, OCI, PACE, RESTORE-L, WFIRST, and others
  - Active development on-going
  - Increased flexibility via new hardware and software APIs
  - Available for new missions/applications today