# Mars Exploration Rovers' Flight Software

**Glenn E. Reeves, Joseph F. Snyder,**
**Khaled S. Ali**

***Jet Propulsion Laboratory***
***California Institute of Technology***
***Pasadena, California***

# Outline



- **Introduction**
- **MER Mission Requirements**
  - **FSW Responsibilities**
  - **Autonomous Activities**
- **Flight Software Structure and Implementation**
- **Software Execution Model**
  - **Tasks**
  - **Inter-Process Communication**
  - **Task Priorities**
- **Conclusion**

# Introduction



- **Mars Exploration Rovers**
  - **Spirit: Landed January 4, 2004**
  - **Opportunity: Landed January 25, 2004**

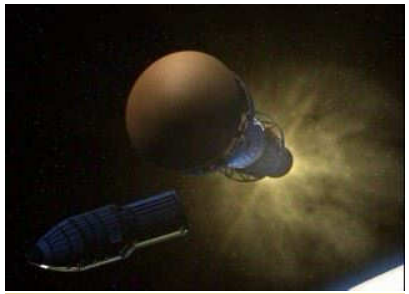
- **Flight Software developed by a team beginning July 2000**
  - **Launch Release**
  - **EDL Release**
  - **Extended Mission Release**
  - **Enhanced Mobility Release**
  - **Technology Infusion Release**

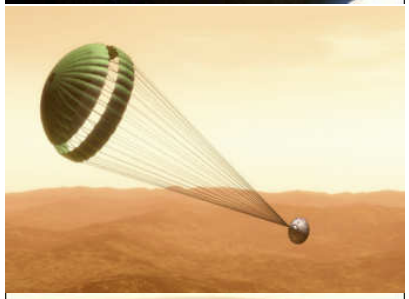
- **MER Flight Software Inheritance Path**
  - **Mars Pathfinder**
  - **Also Deep Space 1, Mars 98, Mars Odyssey**

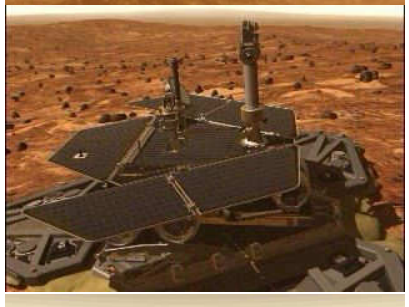
# MER Mission Requirements



- **Mission Phases**
  - **Launch/Cruise**
    - **Countdown to Cruise Stage Separation**
  - **Entry, Descent & Landing (EDL)**
    - **Mars Approach to Landing**
  - **Deployment/Egress**
    - **Articulation To "Six Wheels on Surface"**
  - **Surface Operations**
    - **Nominal science operations**







Images From Mission Animation by Dan Maas

# FSW Responsibilities 1



- **Infrastructure**
  - **Processor control and management**
  - **Spacecraft Time Management**
  - **Power Switch Control**
  - **1553 Bus and Device control and management**
  - **Command processing**
  - **Sequence execution**
  - **Engineering Data Collection**
  - **Data Compression**
  - **Telemetry Processing**
    - **Data Product model used to manage science and most engineering data as files with associated metadata in the onboard FLASH file system**
  - **Communication**
    - **Direct To Earth and Direct From Earth X-Band**
    - **EDL frequency based semaphores**
    - **Simplex and Duplex Orbital Relay via UHF**
  - **Celestial Body Estimation**

# FSW Responsibilities 2



- Infrastructure
- Launch/Cruise
  - Launch Vehicle separation detection
  - Attitude determination and control
  - Axial and lateral trajectory change maneuvers
  - Star identification
  - Inertial Measurement Unit – only attitude knowledge propagation
  - Closed loop thermal control

# FSW Responsibilities 3



- Infrastructure
- Launch/Cruise
- **Entry, Descent & Landing**
  - **Pre-entry trajectory change maneuvers**
  - **Turn-to-entry attitude changes**
  - **Heat rejection system venting**
  - **Cruise stage jettison**
  - **Thermal battery initiation**
  - **Parachute deployment**
  - **Heatshield separation**
  - **Altitude & Velocity determination using radar data**
  - **Airbag deployment**
  - **Retro-rocket initiation**
  - **Horizontal velocity determination using optical data**
  - **Bridle cut and landing event detection**

# FSW Responsibilities 4



- **Infrastructure**
- **Launch/Cruise**
- **Entry, Descent & Landing**
- **Deployments and Egress**
  - **Petal deployment**
  - **Airbag retraction**
  - **Solar Array deployment**
  - **Mast deployment**
  - **High Gain Antenna (HGA) deployment**
  - **Rover standup**
  - **Mobility deployment**
  - **Robotic Arm (Instrument Deployment Device, IDD) unlatch**
  - **Rover-Lander separation**

# FSW Responsibilities 5



- Infrastructure
- Launch/Cruise
- Entry, Descent & Landing
- Deployments and Egress
- Surface Operations
  - Rover attitude and position determination
  - Pancam Mast pointing
  - HGA pointing
  - Rover mobility
    - Compensating for position estimation error (visual odometry)
    - Hazard Avoidance
    - Steering and driving
  - IDD motion and instrument placement
  - Science Instrument Operation
    - Configuration control and data capture
    - Instrument specific data processing, including data compression
  - Activity Coordination
    - Coordinate commanded activities, autonomous activities, system fault protection
    - Arbitrate/manage activities requiring same resources or coordinated interaction
  - Wakeup and Shutdown
  - Fault Protection

# Autonomous Activities 1

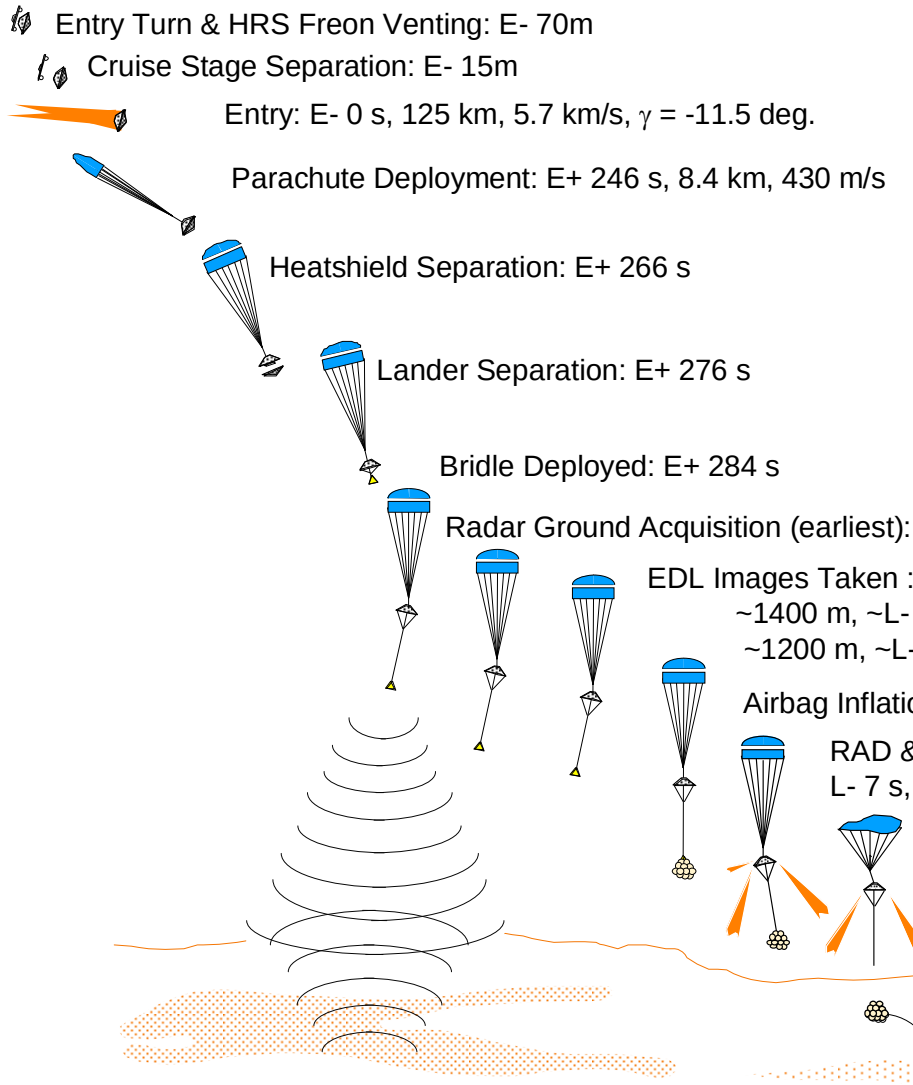


- **Autonomous handling of Critical Mission Events was a key driver of the FSW architectural and functional focus**
  - **Two main options for implementing these "behaviors"**
    - **Encapsulate all behavioral logic in sequences**
    - **Embed required behavior in FSW logic**

- **Selected Option: Embed key autonomous behaviors in FSW**
  - **Both nominal and off-nominal variants**
  - **Behaviors initiated by Operations Team with "high level" commands**
  - **FSW completes activity even in the presence of faults**
  - **Forced system design to mature early in order to fit within FSW schedule**
  - **Allowed for simplified sequence execution mechanism**

# Autonomous Activities 2



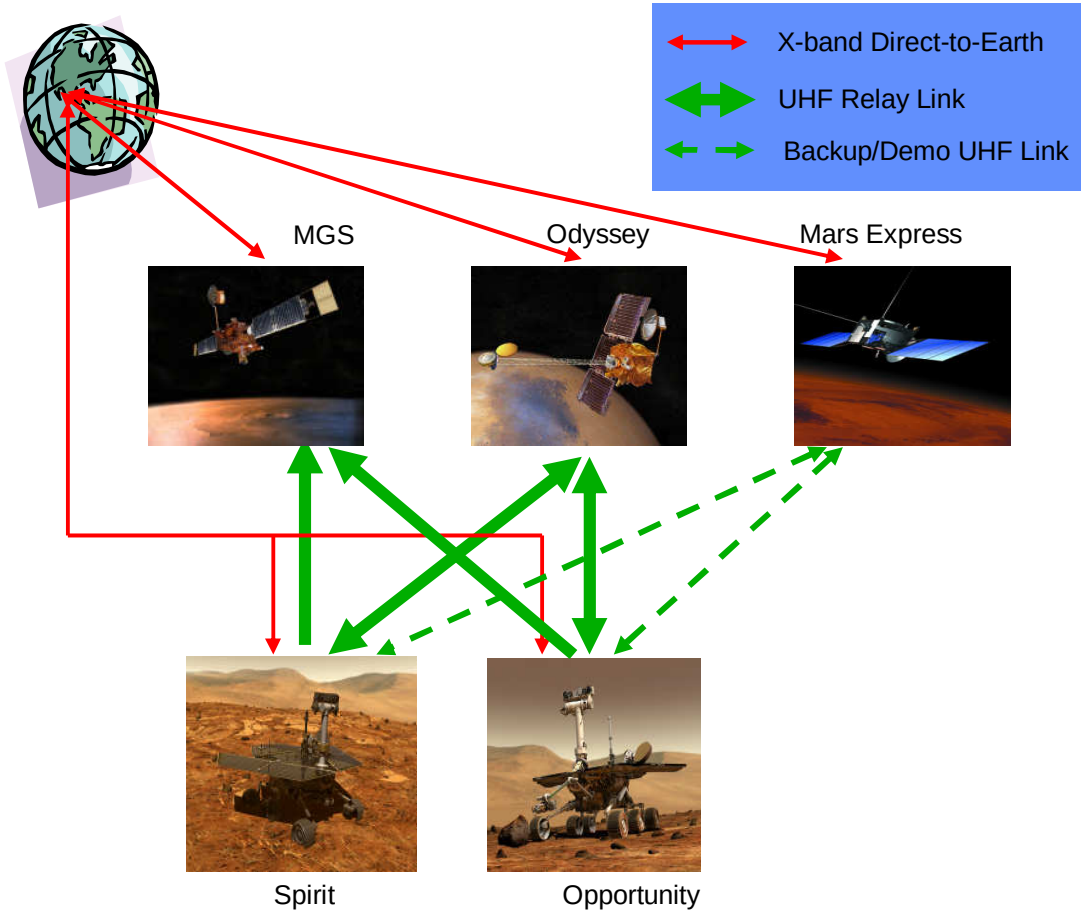



- **EDL Behavior**
  - **Manages entire series of actions from Landing minus 5 days through completion of Solar Array Deployment on surface**
- **Critical Deployments Behavior**
  - **Manages Airbag retraction, petal deployment and solar array deployment**

# Autonomous Activities 3



- **Communications Behavior**
  - **Coordinate all communication**
    - **X-Band (direct to/from Earth)**
    - **UHF (relay)**
  - **Start/Stop session**
  - **Configure telecom hardware**
    - **Turn on warm-up heaters**
    - **Request attitude estimate**
    - **Initiate HGA Earth tracking**
  - **Initiate data preparation**
  - **Fault detection and responses**

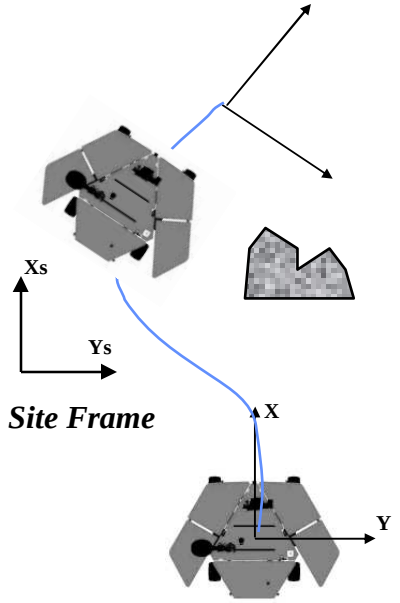


# Autonomous Activities 4



- **Driving Behavior**
  - **Manages repetitive cycle:**
    - **Capture stereo imagery (from any stereo camera pair)**
    - **Compensate for position estimation error (visual odometry)**
    - **Convert imagery to range and obstacle data**
    - **Determine safe path toward goal while avoiding obstacles**
    - **Coordinate motor control for steering and driving**

**Autonomous traverse toward a commanded waypoint with on-board hazard detection using stereo vision - Closed-loop around position and heading estimate**



# Autonomous Activities 5



- **Other Autonomous Behaviors**
  - **Launch Vehicle Separation detection and Communication initiation**
  - **Inertial Pointing for Critical Turn-To-Entry and Emergency Sun Pointing**
    - **Uses multiple combinations of sensors and thruster branches**
  - **Axial and Lateral Burn Control**
    - **Continues in the presence of sensor failure, thruster failure, valve failure or reset**
  - **Standup**
    - **Coordinates mechanism and motor control for Rover standup and mobility deployment**
  - **Attitude Determination with Sun Search**
    - **Mast azimuth and elevation control, imaging, and attitude determination**
  - **System Fault Protection**
    - **Command Loss, Low Power, Overheat**
  - **Wakeup Shutdown behavior**
    - **Autonomously shutdown when Solar Array power below a threshold and no directed activity in progress**
    - **Wakeup via Alarm Clock for Communication sessions or other directed activity**

# FSW Structure and Implementation 1

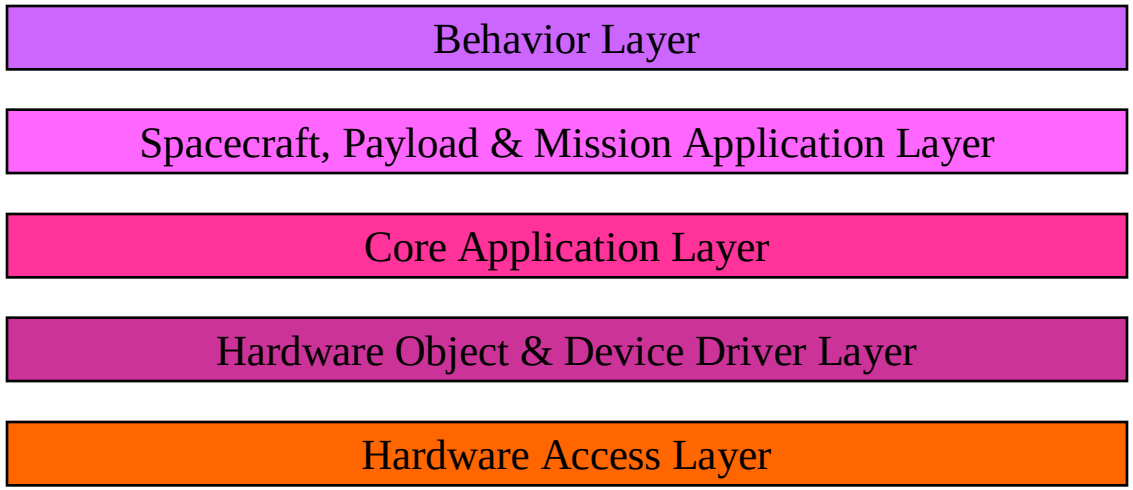


- **Architecture/Design of FSW emphasizes interfaces, encapsulation, and modularity**
  - **System is decomposed into modules**
    - **Module is a collection of source code files and any related ancillary files that are logically managed as a single entity**
  - **Module patterns**
    - **Modules resulting from object oriented analysis to bring together objects with significant collaboration (most of these were inherited via MPF)**
    - **Modules resulting from functional decomposition of the overall system (most of these have minimal MPF heritage)**
    - **Modules that reflect a one-to-one mapping to the execution model or exist solely to support the software development working model of one primary developer per module**

# FSW Structure and Implementation 2



- **Module Layering**
  - **FSW modules are loosely arranged into a layered model (without enforcement)**
  - **Layers represent progression from simplest functions to those functions that are necessary for all (or many) spacecraft to those that are increasingly mission unique or autonomous**



# FSW Structure and Implementation 3



- **Implementation Language and Tool Chain**
  - **FSW coded primarily in ANSI C; some targeted assembly code; some C++**
    - **Limited use of C++ features (inheritance, polymorphism)**
  - **Green Hills compiler, linker, assembler for flight product**
    - **305,000 SLOC (using "real_lines" tool, not including OS)**
      - **Includes 35,000 lines auto-generated code**
    - **11 MB Image size, including OS**

# Software Execution Model 1



- **MER FSW executes on single processor controlled by a Real Time Operating System**
  - **VxWorks RTOS supplies basic outline for FSW operation**
    - **Supports multiple pre-emptive prioritized tasks**
    - **Provides other basic facilities**
      - **Timers, math libraries, I/O, file systems support, logging, message queues**
  - **RTOS receives a Real Time Interrupt (RTI) to drive the system clock**
    - **Supplied by MER Telecom Service Board**

# Software Execution Model 2



- **FSW Tasks**
  - **Relationship between tasks and modules not 1-to-1**
    - **Some modules have no tasks ("library" modules)**
    - **Some tasks share common source code (sequence engines)**
    - **Some modules have multiple tasks (fault behavior module has 13 response tasks)**
  - **Tasks assigned priority commensurate with its required response time**
    - **Generally fixed at initialization**
  - **Tasks are cyclic (time driven) or sporadic (event driven)**
    - **Execution is driven by the arrival of messages via the Inter-process Communication system**
    - **Interrupt Service Routines do the minimum action necessary and then generate messages that wake the appropriate task that completes**

# Software Execution Model 3

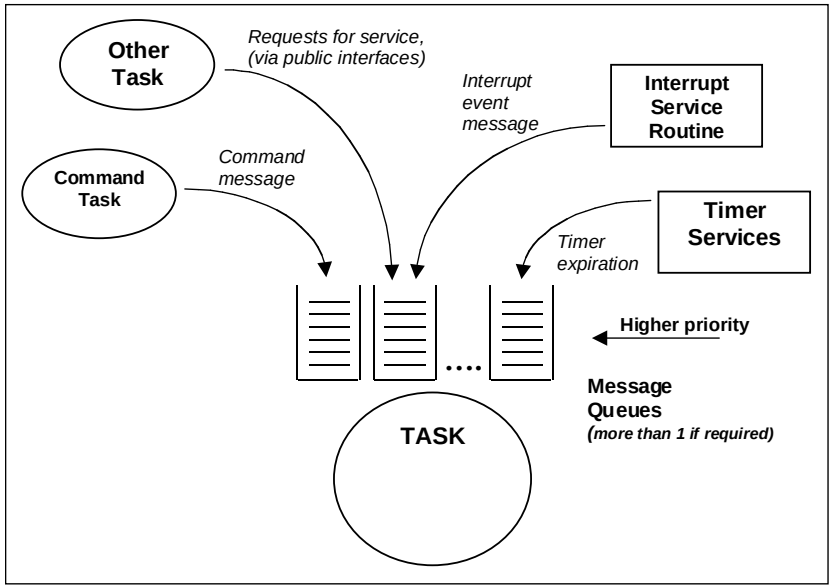


- **Inter-process Communication**
  - **FSW process communication is primarily message-based, provided by capabilities encapsulated in the *ipc* and *reply* modules**
    - **Messages are the principal method for Task-To-Task communication**
    - **Messages are also generated by interrupt handlers, timers, and spacecraft commands**
    - **Messages are point-to-point, not broadcast**
    - **Receiving (server) task provides functional interface that execute in the sender's (client) context**
      - **Results in message being placed in one of the receiving (server) task's input queues**
      - **Keeps details of message format and input queue selection local to the server module**
    - **Messaging architecture includes a "reply" pattern**
      - **Enables a message to be returned to the client after the server completes the action associated with the original message**
      - **Enables Event-based sequencing, spacecraft configuration management, and autonomous behaviors**

# Software Execution Model 4



- **Inter-process Communication (continued)**
  - **Message-based communication (continued)**
    - **Task Input Queues implemented as VxWorks "pipes"**
      - **Holds a series of messages in arrival time order**
      - **Tasks with multiple queues can process them according to specific priorities, or ignore queues altogether**
      - **Queues are of fixed length, holding a fixed maximum number of messages**
  - **Other methods of communication include limited instances of shared memory interaction**
    - **Semaphore controlled access**
    - **Double buffer access**



# Software Execution Model 5



- **Task priority is dictated by relative importance of primary function**
- **Task priorities can be shared, such that execution is time sliced**

- **Tasks with periodic execution requirements subscribe to a timer service**
  - **Service delivers messages to task at the subscribed rate**
  - **Relative task priority determines execution order when multiple tasks are delivered messages at the same rate**
- **Many tasks are entirely event driven; that is, they have no rate requirement**
  - **Event driven tasks usually at a lower set of priorities than Time driven tasks**

# Conclusion



- **MER FSW is a complex product, developed on a very tight schedule**
  - **Overall, the FSW has performed well throughout mission**
- **Design decomposition into modules reflects system and mission requirements**
- **Task assignment reflects separation and priority of the different and varied system activities**
- **Module and task encapsulation enforced by requiring module-to-module interaction only via public interfaces**
- **Software execution model enforces this encapsulation via the message based inter-process communication system**