# Operational Modification of the Mars Exploration Rovers Flight Software

**Martin E. Greco**
Mars Exploration Rover Project
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Martin.E.Greco@jpl.nasa.gov

**Joseph F. Snyder**
Mars Exploration Rover Project
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Joseph.F.Snyder@jpl.nasa.gov

**Abstract -** *The Mars Exploration Rovers (MER) Flight Software (FSW) was designed from the outset to be modified during operations. Two principal methods were envisioned: modifying (Patch) the existing image, or entirely replacing (Load) the existing image with a new version. In fact, both methods have been successfully used since the Rovers landed in January of 2004. This paper discusses the content of the uplink products that are sent to the Rover, the planning of the Patch and Load activities, the testing of the products and procedures, and the actual operations themselves. Lessons Learned and application to future missions are also discussed.*

**Keywords:** Flight software, modification, load, patch, operations, operational procedure.

## 1 Introduction

In the summer of 2003, NASA's Mars Exploration Rovers (MER) project, managed by the Jet Propulsion Laboratory, launched the twin rovers Spirit and Opportunity on June 10th and July 7th, respectively. After the conclusion of a six month interplanetary cruise, both rovers successfully landed on the surface of Mars: January 3rd 2004 for Spirit and January 24th 2004 for Opportunity. After successful completion of their 90 Sol primary mission (approximately 92 days), both rovers continue to operate in extended mission mode as of this writing.

The MER project was designed and built under a compressed development schedule. By early 2003 it was recognized that the flight software necessary to support Entry Descent and Landing (EDL) and initial Surface Operations would not be ready prior to launch. Accordingly, plans were made to perform a major update of the flight software late in the cruise phase. This update, a complete replacement of the flight software used since Launch, was completed in December of 2003, approximately a month prior to landing on Mars.

Similarly, during final development of this first flight software update, a number of improvements and enhancements specific to Surface Operations were identified and assigned to a potential further update of the flight software to be performed after landing and initial operations. This second flight software update eventually included changes and fixes resulting from the first few months of actual operations on the surface. This update, the first performed on the Martian surface, was a complete replacement of the flight software used since just prior to EDL. It was completed in April of 2004, roughly coinciding with the end of the prime mission.

In late 2004, the operational team, having now over 600 Sols of combined experience on both rovers, had collected a set of new and enhanced mobility related capabilities. These eventually formed the heart of a third flight software update. Unlike the first two updates, this update was a "patch" of the flight software loaded at the end of the prime mission. It was completed in February of 2005.

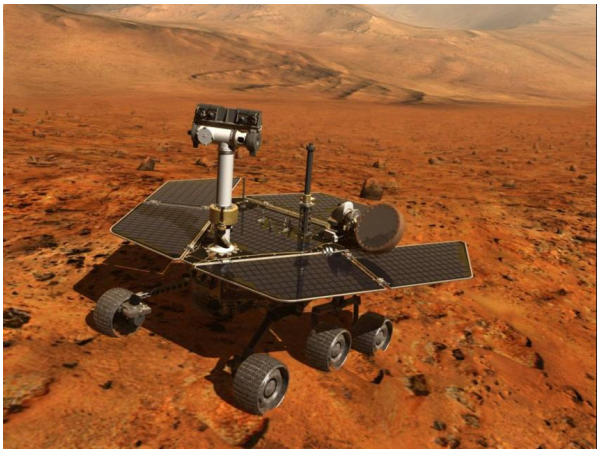

Figure 1 – Artist Rendition of MER rover.

## 2 MER Flight Software Images

The key avionics components related to storing and executing the MER flight software are resident in the Rover Electronics Module (REM). The RAD6K card contains the Central Processing Unit (CPU), 128 Megabytes of volatile Dynamic Random Access Memory (DRAM), and one 3 Megabyte bank of non-volatile Electronically Erasable Programmable Read Only Memory (EEPROM). The Non-Volatile Memory (NVM) card contains two 4 Megabyte banks of non-volatile EEPROM, and 256 Megabytes of non-volatile FLASH memory.

There are two complete flight software images, referred to as the "A" and "B" images, stored in non-volatile memory. A flight software image is conceptually a

single binary block containing two independent components: a boot loader, and the flight software itself. The boot loader is essentially a small independent set of instructions that specifies the image order from which to fetch the flight software, i.e., the operating system and application code. If the selected flight software fails to successfully boot and initialize, the boot loader will go on to its next selection.

The "A" flight software image is stored partly in the NVM card Bank A EEPROM and partly in the NVM card FLASH memory. Similarly, the "B" flight software image is stored partly in the NVM card Bank B EEPROM and partly in the NVM card FLASH memory. The two separate images of the flight software provide redundancy in the event of several different fault scenarios. There is no requirement that the two flight software images be identical. In fact, for most of the surface mission these images have contained different versions of the MER flight software.

Selection of the flight software is determined at boot time. Following a "cold boot", which occurs when power is applied to the main REM electronics, the initial flight software image is selected via a ground commandable Critical Relay Control (CRC) - a latching relay. The boot loader associated with this image is executed, which in turn fetches the flight software from the first image specified. If the selected flight software image fails to load and initialize, the boot loader fetches the flight software from the second image specified, and so on. In the event of a warm boot (RAM remains powered) the boot order is selected by FSW. Figure 2 depicts the configuration of the "A" and "B" images after applying the R9.1 Patch in February 2005.

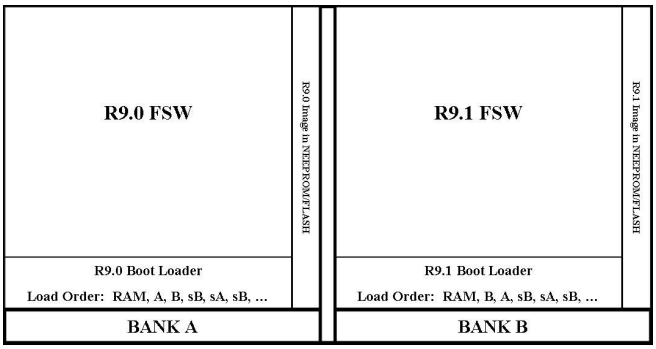


Figure 2 – Example MER FSW Image Configuration

# 3 Overview of MER FSW Modification

There are two methods of modifying a MER flight software image, Patch or Full Load. A Patch is a selective replacement of portions of an existing flight software image. A Full Load is the complete replacement of an existing flight software image including the boot loader. The choice between a Patch and a Full Load is generally dictated by the volume of the products that must be uplinked to the rover. The volume for a Full Load is about 8 Megabytes, and varies for a Patch, increasing with the number and complexity of the software changes. For the Patch performed in February of 2005 the volume was approximately 2 Megabytes.

The generation of the uplink products begins when the flight software development team releases a new version of the flight software. Releasing a new version of flight software includes generation of a new flight software image. As part of this process, the flight software development team can optionally produce a set of Full Load products or a set of Patch products. Full Load products are generated by splitting the new flight software image into a set sequential data files and producing a set of associated control files. The control files provide the ordering instruction for constructing the new flight software image directly from the data files. The control files also contain checksum and identification data used to uniquely identify and validate the reconstructed flight software image.

Patch products are generated by comparing the new flight software image to a flight software image containing an earlier version of the flight software. Differences are identified and a set of sequential data files are generated containing the changed portions of the earlier flight software image. In addition, a set of control files are produced which contain the instructions and locations for applying the changes contained in the data files to the earlier flight software image. The control files also contain checksum and identification data used to uniquely identify and validate the patched flight software image.

The MER flight software modification procedure consists of two main parts: 1) A set of one or more "Uplink Days" reserved for radiation of the Patch or Full Load products to the rover, and 2) a "Build Day" reserved for the building (creation), validation, and saving to non-volatile memory of the new flight software image. Uplink Days are planned to minimize all other activities running in parallel to the actual receipt of the modification products. This allows optimal use of the available uplink bandwidth and reduces the likelihood of unforeseen side effects. Similarly, the Build Day is planned such that *no* other activities are running in parallel.

All onboard activities related to the flight software modification are planned to tolerate and recover from all credible anomalies. Power consumption, Telecom link margin predictions, and procedure timing all assume a worst case scenario. Margin is included to allow for re-radiation of missing or corrupted products, ground

evaluation and confirmation of all critical steps, and execution of contingency commands in the event of off-nominal events.

# 4 Activity Planning and Command Generation

The flight software Full Load and Patch activities required early planning and allocation of resources. Deep Space Network (DSN) station coverage must be negotiated in advance, requiring the layout of a straw-man plan. This plan includes the estimated size of the uplink products as well as the estimated duration of the Build Day. For Build Days a 70-meter antenna is desired for higher downlink rates and for added margin in case of an anomaly.

The MER rovers possess two telecom systems, an X-Band system and a UHF system. For the Full Load performed prior to landing, only the cruise X-Band system was available for uplink. For the Full Load and Patch performed during surface operations, the uplink could have been achieved via UHF relay via the Odyssey orbiter. However, the necessary extra overhead on the uplink files, the brief pass durations (approximately 15 minutes), and the low uplink rate to the orbiting asset of approximately 1000 bits-per-second (bps) more than offset the higher relay uplink rates. Because of this, X-Band uplink was used exclusively.

During surface operations the rovers have two available X-Band antennas, a monopole Low Gain Antenna (LGA) and an articulated High Gain Antenna (HGA). The HGA was selected because it supports higher uplink rates.


Figure 3 – Spirit's HGA.

Selection of the HGA (see Figure 3) involves additional constraints. Using the HGA for long passes requires that the rovers be oriented such that there will be no mechanical occlusions (from the Pan-Cam Mast Assembly (PMA) or from the rovers' deck), no contact with any of the HGA hard-stops, and no possibility of performing a "flop". A "flop" is a 180 degree rotation of the azimuth drive and a reflection in elevation which takes about one minute, resulting in a loss of communications [1]. Another constraint that must be considered is potential shadowing of the HGA by the PMA. This reduces the temperature margins on the warm-up times for the HGA motors and gears. If the associated temperature limits are violated it could result in a premature stall of the HGA motors and a complete loss of the communications pass.

Detailed power predictions were generated for the duration of the Full Load and Patch activities to verify that the rovers will be maintained in an energy neutral state. Figure 4 shows an example of the predicted power profile for the Opportunity Patch activities. During the surface Full Load activities it was discovered that HGA shadowing on the solar arrays was not accounted for by the power predictions. Fortunately, the power loss incurred by the shadowing was less than the allocated margin and the rovers remained in a power positive state.

Detailed thermal predictions to verify the rovers would not overheat were not necessary due to the dates the Full Load and Patch activities were executed. Winter was approaching during the Full Load and winter was ending during the Patch.

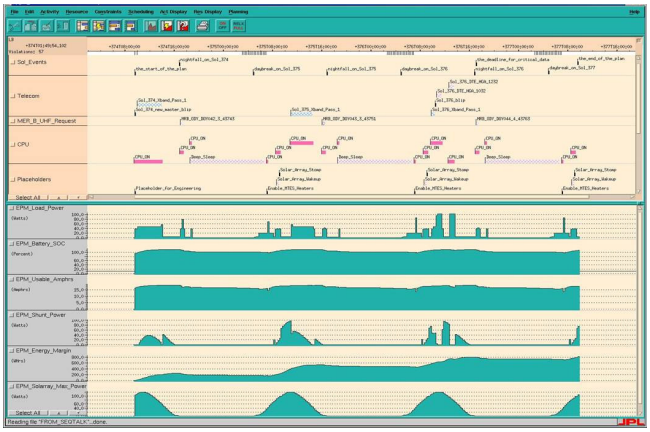

Figure 4 - JPL Mapgen Power Modeling.

The actual on-board steps of the Full Load and Patch activities were performed by a combination of real-time and sequenced commands. A sequence is a specialized file containing one or more commands that execute serially. MER sequences allow for conditional commanding. This capability allows ground-in-the-loop interactions to be kept to a minimum. The key reason to minimize ground-in-the-

loop interactions is that each of these interactions absorbs a significant portion of the total available time for operations. The total available time for operations is a function of the solar energy available, initial battery state for charge, and expected power dissipation. During the Martian winter the available energy is especially limited. Ground-in-the-loop interactions are costly due to the time lag between events occurring on Mars and on Earth. For a rover on Mars, the one-way light time is roughly between 3 and 23 minutes, with an average of 10 minutes. Therefore each "event → telemetry → decision → command → event" cycle takes at least 30 minutes for the average one-way light time.

The following is a snippet from the key Build Day sequence that demonstrates how conditional commanding allows events to proceed without ground-in-the-loop interactions between each step.

*Sample Sequence*
1. CMD Load_Validate Prom1
2. IF Last Command ≠ Success
3. THEN Terminate Sequence
4. CMD Load_Validate Prom2

In the example the sequence first issues a command to validate the "A" flight software image. A load validation verifies that the image has not been corrupted. If the commanded load validate fails, (i.e. the image is corrupted) the rest of the sequence will not execute. Otherwise, if the commanded load validate succeeds, the sequence continues by issuing the next command – in this case a command to validate the "B" flight software image.

When planning the flight software Full Load and Patch activities it is necessary to deal with multiple time systems, including Mars Local Solar Time (LST), Coordinated Universal Time (UTC); and Pacific Standard Time (PDT). Since there is at least one ground-in-the-loop interaction built into the activities, a synchronized timeline that can be updated as actual events occur and will propagate the remaining events is critical. Care must also be taken to ensure that the activities do not contain any overly tight timing requirements. Thus, wherever possible, extra time, up to several minutes, was inserted between all key events. This is particularly applicable to events that change the rover telecom configuration, as this requires a corresponding reconfiguration of the DSN. For the Build Day activities there are approximately six different DSN configurations with only one or two having a predefined absolute time.

Once the nominal activities were finalized, all credible anomalies that could occur at each step of the activity were investigated and contingency plans were generated to deal with them. Some of the specific anomalies prepared for include: 1) Failure of the commands to create the new flight software image or copy it to non-volatile memory, 2) Loss of communications due to HGA errors, 3) Loss of communications due to anomalies unrelated to the HGA, 4) Unexpected power profiles, and 5) Loss of communications or data due to DSN issues.

Other anomalies not specific to the flight software modification activities are handled as they would at any other time by the Flight Operations Team.

# 5 Testing

Testing of the uplink products and the Full Load and Patch procedures followed the JPL philosophy of *Test as you fly and fly as you test*. All files, sequences, and immediate commands that will or, in the case of an anomaly, might be radiated to or executed on the rovers must first be tested. The MER project possesses three hardware rich Testbeds: 1) The Flight Software Testbed (FSWTB), where most of the development level testing occurred; 2) The Cruise Entry Descent and Landing Testbed (CETB), where, as the name implies, almost all of the Cruise and EDL tests were performed; and 3) the Surface System Testbed (SSTB), for mobility testing. In addition to these high fidelity test resources, the MER project has several lower fidelity (i.e., no hardware in the loop) Flight-Like Test Sets (FLTS), where simple commands, sequences, or other uplink files can be verified. For example, the uplink files for Spirit were tested using the Testbeds, but the corresponding files destined for Opportunity were tested using an FLTS. All of the tests of the Build Day activities, especially those involving communication passes, the creation of the new flight software image, and the copy of the image to non-volatile memory, were performed on the CETB due to its high fidelity.

The Full Load and Patch activities were tested with respect to the following requirements:

- All files, sequences and immediate commands designated for transmission or potential transmission were used in their flight-like form; this ensures that the flight software properly accepts and processes the files or commands.
    - The uplink products were presented at the expected uplink rate.
    - Uplink durations were verified.
- All X-Band and UHF communications passes were executed at the appropriate times and for the actual durations.
- Nominal sequences and immediate commands were executed in flight-like order with the appropriate timing.

- Contingency sequences and immediate commands were executed in as credible a flight-like manner as possible.

# 6 Surface Operations Description

As described earlier, the actual flight software modification activities are divided into two main parts: 1) one or more "Uplink Days", and 2) a "Build Day". Figure 5 shows the high level timeline for the two Uplink Days and one Build Day used for the flight software Patch on Opportunity in February, 2005.

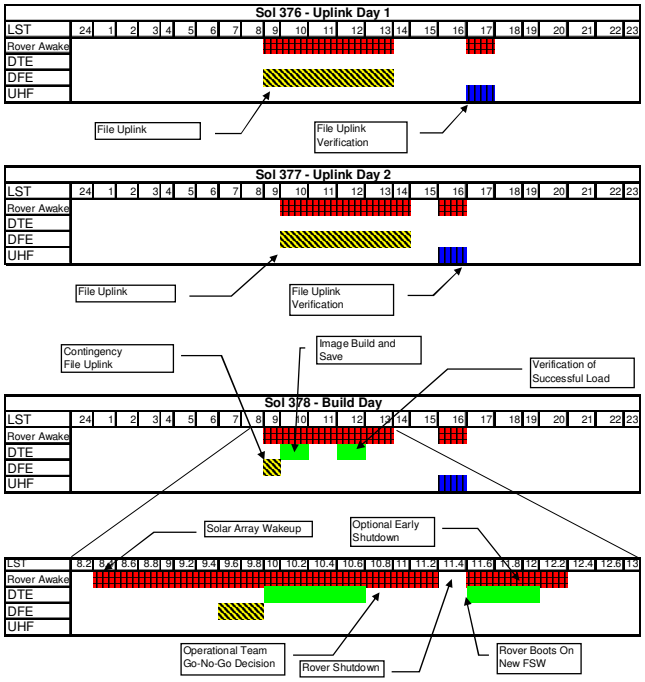


Figure 5 – High Level Timeline of Approximate Times of Opportunity Patch Activities.

Uplink Days are quiet days for the rover; there is no mobility or Instrument Deployment Device (IDD) activity allowed and only minimal remote science activities are scheduled in order to preserve power margin and to minimize possible side effects.

The first Uplink Day begins with the handover from the previous Sol's master sequence[1] to this Sol's master sequence. Its first action is to confirm the handover by issuing a "beep"; that is, is a modulation of the X-Band's sub-carrier to generate a single tone for five minutes. This is a convenient method often used on the MER project to confirm the occurrence of an event without expending the energy necessary for a full communications pass. After the beep is complete, any pre-planned engineering and/or science sequences are activated. The rover then begins a long Direct from Earth (DFE) receive-only X-Band communications pass. The pass is used to uplink as many Full Load or Patch files as possible. Although DFE allows long (3.5 to 4.5 hour) passes because of low energy consumption (not using the transmitter saves 50 Watts), there is no verification that the uplinks were successful until the UHF communication pass at the end of the Sol.

After the DFE pass ends there is another opportunity to activate any pre-planned engineering and/or science sequences. The next major event is the end of Sol UHF communication pass. Just before this pass, an uplink verification sequence is activated. This sequence generates directory listings and several other types of data needed to verify all planned activities during the Uplink Day were successful. After the pass, the rover shuts down for the night. This cycle is repeated for each Uplink Day.

The Build Day begins with a short 30-45 minute contingency DFE pass, available in the event that any of the previous uplink products need to be re-radiated. There is no ordering requirement, so any missing file can be uplinked during this pass. Once the contingency DFE window completes, the rover begins a Direct to Earth (DTE) two-way X-Band communications pass.

Shortly after DTE pass begins, the Build Day Master sequence activates the conditional sequence that will perform the actual flight software modification. Because this activity is proceeding in parallel with the DTE pass, the Flight Operations Team is able to monitor and verify events in near real-time. The sequence will perform the following principal steps. The conditionality constructs are used after each step; if successful, the sequence proceeds, otherwise it ends.

1. Verify that the current flight software images stored in non-volatile memory are not corrupted.
2. Build the new flight software image in DRAM. For a Full Load this is done directly from the uplinked data files. For a Patch, one of the existing flight software images is first copied into DRAM, and then modified.
3. Save the new flight software image to the designated location in non-volatile memory.

As this sequence executes, the Flight Operations Team verifies the activity. If all is nominal, the new flight software image has been saved as either the A or B image. At this point, upon final confirmation from the Operations Team, the rover is ready to start using the new flight software.

[1] A master sequence is a special sequence that is in control of activating all other sequences that will execute that Sol; it is also in charge of the awake and sleep cycle of the rover.

If necessary, the appropriate CRCs are then set to ensure that upon the next cold boot the new flight software version will be executed. Finally, a shutdown is commanded to force a cold boot. The rover shuts down, and then wakes up 15 minutes later. The Build Day master sequence then initiates another DTE communications pass. During this pass the Flight Operations Team verifies that the new version of flight software has been successfully booted and is executing as expected. Following this verification, all that remains are cleanup activities, including deletion of the original uplink products saved in FLASH, and any necessary configuration related to using the new flight software. At this point, the rover is released for resumption of nominal surface operations.

# 7 Lessons Learned

The single biggest improvement to the MER flight software modification process would be to reduce the amount of time necessary to stand down from nominal surface operations. Currently, the most time consuming activity is uplinking the Full Load or Patch files. There are several options available to optimize this activity:

1. Use the maximum file size allowed by the uplink protocol, thus reducing the overall overhead per file.
2. Relax the rules for using UHF uplink, thus reducing or eliminating all UHF overhead.
3. Compress all Full Load or Patch files before converting them to uplink products.

For future rover missions, DFE for uplink products will likely be unavailable or impractical. All significant uplinks will be via relay assets. It is expected that future relay assets will have higher communication bandwidth, both between Earth and Mars, and between Mars orbit and the surface. With these increased capacities, it is envisioned that a Full Load could be completed in less than two Sols.

The next area of improvement is in automation of the on-board modification activities. Rather than rely on sequencing, Full Loads and Patches should be full fledged high level behaviors, intrinsic to the flight software. Improved visibility into the process should be included, with specialized high priority telemetry dedicated to the behavior. This has a number of important advantages, including enforced operational consistency, standardized prediction and testing, and improved performance and timing margins. This will be especially important for future rover missions where the surface lifetime requirements exceed one Martian year, and the need and/or desire to change flight software will be greater than ever.

The last area of improvement is in testability. As discussed earlier, planning and testing the uplink products and procedures are resource intensive. A large portion of test time was consumed by attempting to replicate the rover state at the time a particular command was to execute. Improving testbed state initialization and removing the reliance on sequencing would release more time for contingency planning and testing. Currently, most contingency testing occurs as a side effect of nominal testing. That is, because of the difficulty in executing a completely nominal test, many unplanned test anomalies occur. Even though the root cause of these anomalies are almost always due to improper test conditions, their symptoms often are the same as credible flight-like anomalies. Furthermore, recovering from them typically requires the use of flight-like contingency procedures. An interesting observation is that many countless hours were spent in the Testbeds to achieve one flawless nominal test. But on the actual rovers, the Full Load and Patch activities to date have all executed perfectly.

# 8 Conclusions

The MER project was developed under a highly compressed schedule, making in-flight update of the flight software a certainty well before launch. To date, two full replacements and one major modification of the executing flight software have been successfully accomplished. Updating the MER flight software remains a challenging and complex set of activities, including planning, development, test, and operational implementation. Alternative approaches to improve uplink efficiency, expand on-board autonomy, and simplify testing should be explored on future rover missions to reduce this complexity and its accompanying risk.

# 9 Acknowledgements

The following people have been essential to the successful flight software modifications performed to date. Tracy Neilson, Rover Behavior Specialist, for conceiving and testing all surface and cruise contingency plans; Edwin Odell, Flight Software Lead Integration Engineer, for initial contingency planning; Al Herrera, for activity power modeling; Tony Vanelli for invaluable Guidance Navigation and Control and HGA knowledge; and Hanry Hartounian for boot testing and CRC telemetry predictions.

# References

[1] Mars Exploration Rover Functional Design Description Volume 26: High Gain Antenna Gimbal (HGAG), Julie Townsend, Diana Darus, and Joel Krajewski, 2003 Jan 23, Internal JPL Document.