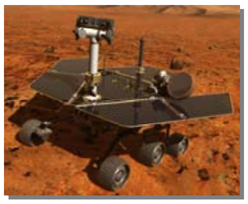
The Mars Rover Spirit FLASH Anomaly

Glenn Reeves
Tracy Neilson
Jet Propulsion Laboratory (JPL)
Pasadena, CA 91109
818-393-1051
Glenn.E.Reeves@jpl.nasa.gov, Tracy.A.Neilson@jpl.nasa.gov

Abstract—The Mars Exploration Rover "Spirit" suffered a debilitating anomaly that prevented communication with Earth for several anxious days. With the eyes of the world upon us, the anomaly team used each scrap of information, our knowledge of the system, and sheer determination to analyze and fix the problem, then return the vehicle to normal operation. This paper will discuss the Spirit FLASH anomaly, including the drama of the investigation, the root cause and the lessons learned from the experience.[1]

TABLE OF CONTENTS

1. INTRODUCTION



NASA's Mars Exploration Rovers (MER) project landed two rovers, Spirit and Opportunity, on Mars on January 4 and January 25, 2004, respectively, where they continue to operate in extended mission mode as of October 2004. An anomaly occurred on the MER-A vehicle, Spirit, on Sol[2] 18, or January 21, 2004, which precluded normal operation of the vehicle. The anomaly is now well understood and changes have been made to both the operational procedures and the flight software (FSW) to address the root cause. However, during the anomaly event period, the team was faced with the daunting challenge of doing detective work at interplanetary distances. Individual challenges included very little data to work with and a vehicle that appeared to behave differently than designed.

The vehicle remained in this uncontrolled condition until Sol 21 when the vehicle was commanded into a degraded, but usable, configuration. Subsequent diagnostic activities occurred over the next 11 sols. The vehicle was finally restored to its normal operational state on Sol 32, and nominal science activities resumed on Sol 33.

The root cause of the anomaly was a design error in the software module that provides file system services. In addition, two significant configuration errors detrimentally affected the overall behavior of the system once the root cause error occurred.

In addition to the technical factors related to the anomaly, there are also programmatic contributors and other lessons learned that became apparent as the anomaly investigation progressed.

[1] 0-7803-8155-6/04/$17.00©2004 IEEE

[2] A sol is a Martian day, 24 hours 37 minutes long

2. SEQUENCE OF EVENTS

By January 21, the Spirit rover team was euphoric. The rover had been safely landed, all the complex releases and deployments went off without a hitch, and we had driven her off the Lander and onto the Martian surface. Every science instrument checked out fine and every camera worked great. We had just settled into the fast turn-around of nominal mission operations.

A nominal sol consisted of two UHF communication windows[3] with the overhead orbiters (either Odyssey or Mars Global Surveyor), early in morning and in the afternoon. Direct-to-earth (DTE) X-band communication windows around 09:00 local solar time (LST) included data downlinks and sequence uplinks to the rover for the daily activities. The rovers are designed to "sleep" through the night and wakeup autonomously for these communication windows. During sleep, the rover electronics, receiver and processor are turned off, but the mission clock and battery control board (BCB) remain on. The BCB is responsible for turning the processor back on and regulating charging the batteries.

Sol 18

Sol 18 Plan

The plan for Sol 18 was for the rover to wake up for an early morning UHF communication window with the Odyssey orbiter to downlink yestersol's data. It would then return to sleep until 8:30 LST, when it would wake up again to warm up the actuators on the high gain antenna (HGA) for the DTE X-band communication window with Canberra. During the DTE window, a sequence would be executed to characterize the actuator that controls a mirror in the Miniature Thermal Emission Spectrometer (MTES). The purpose of this activity was to gather calibration data for operating the mechanism at the cold morning temperatures. The rest of the sol would be spent brushing a rock with the Rock Abrasion Tool (RAT). Afternoon DTE and UHF windows were scheduled onboard to downlink the engineering and science data.

Sol 18 Observations

The early morning UHF window telemetry showed no problems and the 9:00 LST DTE window started right on time, initially indicating that the vehicle was healthy. The operations team began the uplink of the sequences. Eleven minutes into the window, telemetry showed that uplink errors were detected onboard. The downlink became spotty. At approximately 9:16 LST, the signal dropped out completely, 14 minutes earlier than scheduled. At this point, without any more information, poor weather at the ground station was blamed for the signal dropout.

We expected a beep[4] at either 10:00 LST to indicate the new master sequence was onboard and running, or at 10:10 LST to tell us the old sequence was still running. We did not detect the carrier signal for either beep. Again, we blamed the weather at Canberra for the lack of any signal.

At 11:20 LST, we commanded a 30-minute high priority HGA communication window, but no signal was received. Again, the DSN station was the prime suspect, but we were beginning to suspect antenna pointing problems or telecom hardware failures.

At 12:45 LST, we commanded a beep and this one was received as predicted, both in start time and duration. This told us that the vehicle was commandable and that none of the onboard system level fault protection responses, that change the uplink rate, had executed.

We then commanded the afternoon master sequence to start and we received the beep embedded in the sequence per predict. The team breathed a sigh of relief, but it was short-lived.

Just a half hour later, the team waited for the planned 14:00 LST HGA DTE communication window, but no carrier signal was detected. Because the beeps (which worked previously) use the LGA (which does not have to be pointed by the rover), the speculation was that we could have an HGA pointing problem. Since we still hadn't received modulated data, the problem could still be the hardware interface board to the radio or possibly a FSW problem.

During the next two hours, we tested the sequences on the highest fidelity test beds and checked out a few theories before the next scheduled UHF window. This was to be the first UHF window since the trouble started, but unfortunately the Odyssey orbiter detected no signal from Spirit. The testing revealed no suspects either.

Since UHF communication uses a completely different path than the X-band DTE communication, the idea that a HGA pointing problem was to blame was thrown out

[3] The MER vehicles have pre-commanded communication windows. The ground specifies the start time, configuration, and duration; the vehicle will autonomously power on (if necessary), prepare the telemetry, configure the telecommunication hardware, and initiate transmission.

[4] A beep is a sequenced short duration communication window that sends a DTE carrier-only signal through the low gain antenna (LGA). No data are downlinked to the ground. Beep sequences can be activated either by an immediate command from the operations team or by sequence, usually to tell the team that the onboard sequence is running.

because there is no pointing of the UHF antenna. Panic started to set in for the operations team.

Sol 18 Conclusions

By the end of this sol, we knew the rover had not send modulated data to the ground, but it did communicate via carrier signal by command. Potential candidates for the observed behavior at this time were a low power situation, an overheat condition, a FSW communication behavior problem, a problem with the interface card to the radios or FSW-initiated resets. At this point, we formed an anomaly team and our now grim project manager, Pete Theisinger, reported to the press that "a very serious anomaly" had occurred.

Sol 19

Sol 19 Plan

Sol 19 plans were replaced by anomaly response activities. Our objectives for Sol 19 were to establish commandability and to establish modulation in an X-band communication window.

Sol 19 Observations

The next pre-programmed window was an early morning Mars Global Surveyor orbiter UHF communication window at 1:45 LST. This orbiter detected the signal from Spirit and the window started at the correct time, but no valid data were received. Two minutes and 20 seconds of repeating pseudo noise (PN) code were the only data received. This told us that the rover woke up for the communication window and turned on the UHF radio, but it did not pass valid data to it so the radio only transmitted PN codes.

No signal was detected during the scheduled 4:30 LST Odyssey window, or the pre-programmed 9:00 LST HGA DTE window.

At this point, the team started commanding beeps again, as well as listening for the sequenced beeps[5]. If the onboard fault protection response to a low power event had run, it would have added a communication window at 11:00 LST, but no signal was observed at this time. We finally received a beep, but it was one that we had commanded at the 7.8125 bps uplink rate, which is the data rate the onboard fault protection configures when a major fault has been detected. Note that a single processor reset will not result in this fault uplink rate. This meant that some new event had occurred on the vehicle that caused the behavior to change. The anomaly team was mystified.

The pre-scheduled afternoon UHF window failed to produce a signal, and our commanded DTE window on the LGA also failed. However, the commands were sent very close to Earthset and may not have reached the rover.

Sol 19 Conclusions

We had established commandability, but we had not yet received modulated telemetry.

We knew a system-level fault that changed the uplink rate had occurred sometime between Sol 18 13:30 LST and Sol 19 14:40 LST. The only autonomous responses that would change the uplink rate were responses to a low power event, a mission clock failure, an uplink loss fault or an ‘X-band fault’[6]. A mission clock failure was unlikely since the vehicle had tried to communicate at the correct time for the first morning UHF window, and an uplink loss response was unlikely because the timer wasn’t expected to expire until later the next sol. There were no indications of a power problem from the last received telemetry on the morning of Sol 18, and an ‘X-band fault’ should not affect the UHF communication.

FSW initiated resets seemed to be the only likely behavior for the lack of communication using the two separate communication paths, but the response to a reset does not automatically change the uplink data rate.

This was a puzzling sol for anomaly team. Pete Theisinger briefed the press that “the spacecraft thinks it's in the fault side of the tree somehow, for some reason. That would mean that we've got positive power, some elements of the software are working, … the X-band system is working, … the transponder; all that stuff is working, so that would be more information -- good news.”

Sol 20

Sol 20 Observations

None of the overnight UHF orbiters received any signal from the rover, and two of our attempts at commanded beeps in the 9:00 hour failed. However, we were relieved when we received an autonomous 10 bps signal at the scheduled 9:55 LST communication window. The carrier

[5] The main (master) sequence usually spanned two sols with the expectation that the second sol would have its own master sequence that would terminate the prior sol’s master sequence. The operations team embedded beeps in each master sequence that would indicate the new master sequence had taken over or, if the beep was received at a later time, that the old master sequence was still in control.

[6] An X-band fault can result when the FSW detects transponder failures, failure of a coax switch or the waveguide transfer switch, failure to turn on the amplifier, camera mast pointing problems, attitude estimation failures that drive the HGA beyond its allowable range, or HGA positioning problems during a communication window.

signal lasted only 10 minutes, but we did receive a single garbled telemetry packet. The FSW team attempted to decode it but quickly determined the data were garbage. However, the time of this attempted communication, and the data rate, convinced us that an onboard 'X-band fault' response had run.

The team commanded an LGA DTE communication window and we were overjoyed to receive the first set of telemetry, which unfortunately did not include any health information. Our hopes were quickly dashed when we discovered the next few packets contained exactly the same telemetry. The same information was sent over and over again. The signal then terminated suddenly, earlier than scheduled cut-off time.

At 11:50 LST, we commanded another LGA DTE communication window. This time, the signal and data for the full window duration were received, and the telemetry indicated many reset/reboot cycles had occurred. The multitude of resets explained much of the observed behavior.

When the FSW detects a severe error, it reacts by forcing a reset of the flight computer and a re-initialization of the FSW. If the FSW determines that a reset is necessary during the FSW initialization process, the FSW delays the actual reset until a minimum time period has passed. This process is referred to as a "delayed reset". The intent is to allow sufficient time for ground intervention.

When the delayed reset time expires, the reset is initiated. The system keeps track of the number of repetitive resets and modifies the delayed reset time interval. The boot logic part of the software also keeps track of the number of initialization attempts and is designed to load and start alternating copies of the FSW with each reset. By design, the time period between resets is set to a fifteen minute delay for the first severe error, fifteen minutes for the second, and then one hour for the third. This pattern then repeats (see Figure 1). Commands can be processed and communication windows can occur during the delayed reset period.

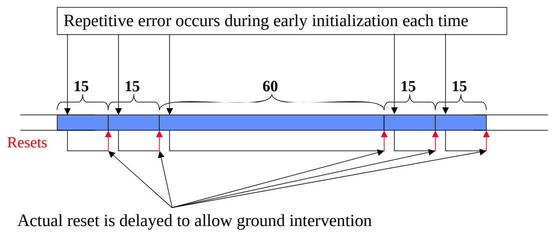

Repetitive error occurs during early initialization each time
15
15
60
15
15
Resets
Actual reset is delayed to allow ground intervention

Figure 1: Delayed reset behavior

At this point, the telemetry confirmed that the 'X-band fault' response had executed, and had changed the uplink data rate to 7.8125 bps. We quickly deduced this was caused by the fact that the FSW marks the encoder position knowledge as "unknown" during actuations (saving precise position knowledge would have required too many EEPROM writes) and one of the resets had occurred during an HGA move. When the FSW booted back up, it recovered the stored encoder positions from EEPROM. When the next scheduled HGA communication window started, the window failed because the encoder positions were "unknown" and the FSW could not point the antenna. The FSW declared an 'X-band fault', which aborted the window, changed the uplink rate to 7.8125 bps and converted subsequent X-band communication windows to 10 bps using the LGA. Our mystery of why the behavior changed between the Sol 18 afternoon and Sol 19 afternoon was solved.

But Spirit threw us another puzzle. The power and thermal telemetry indicated higher temperatures and a lower battery state-of-charge than predicted (given the nominal loads and nominal solar array current). This indicated that the rover electronics had been on much longer than expected and the vehicle had possibly not shut down overnight. The vehicle should have autonomously shut down the processor and electronics each afternoon around 15:30 LST to save power and then autonomously woken up at approximately 9:30 LST each morning via solar wakeup (when the solar array current reaches 2.0 amps for the first time each sol).

The telemetry also indicated that the MTES characterization sequence on Sol 18 never completed. This information helped pinpoint the first reset event to the first Sol 18 HGA communication window.

At 14:00 LST on Sol 20, the rover performed another pre-scheduled DTE window at the fault configuration (10 bps downlink on the LGA), as predicted.

Our priority at this point was to shut the vehicle down early in the afternoon, in an effort to charge the batteries. We commanded a shutdown, which terminated the current communication window, and the loss of signal occurred at the predicted time. Fifty minutes later, we commanded a beep at 7.8125 bps to alert us if the shutdown command did not work, and much to our disappointment, the beep was received! This information affirmed our fear that the onboard shutdown process was not working.

At this point we scrambled to delete the next few scheduled UHF windows to conserve battery charge, but these commands did not reach Spirit successfully. The Earth had set below the horizon.

Quite surprisingly, the Odyssey communication window at 16:23 LST produced 73 Mbits of telemetry for the full duration. This was the first UHF window that lasted the predicted duration since the start of this anomaly. The telemetry provided additional indications that multiple resets were occurring and that some of the commands we sent during the afternoon were not received correctly (due to a delayed reset occurring in the middle of the transmission). However, only real-time data generated during the window were received. The majority of the telemetry was fill data and no recorded data were ever received.

At this point, the ops team requested the orbiters to delete their overnight communication windows to save rover power. The orbiters turned off their beacons so Spirit's transmitter would not attempt to transmit.

Sol 20 Conclusions

By the end of Sol 20, we knew FSW was in a continuous delayed reset loop. The first reset occurred during the Sol 18 morning DTE window coincident with the MTES actuator checkout. Both commanded and autonomous shutdowns were failing and the vehicle probably had not shut down in a while.

Memory corruption was unlikely because different FSW copies were being loaded and used. The lack of recorded data (which are stored in FLASH memory) indicated that the FSW was not accessing FLASH memory, so our prime suspects became the FLASH memory, the file system and the FSW that reads the data files and creates telemetry for downlink.

Our project manager upgraded Spirit's condition to "critical" at the press conference.

Sol 21

Sol 21 Plan

The plan, at Earthrise, was to send a hardware command to place the Spirit in a "crippled mode". This command tells the FSW "do not use the FLASH file system"[7].

Sol 21 Observations

Between 8:30 LST to 11:08 LST, we repetitively sent the CRIPPLED command. No signal was detected at 9:35 LST for the preplanned DTE communication window. This should have been an FSW-adjusted LGA DTE communication window at 10 bps. Loss of this window was not entirely unexpected given that the vehicle could not shutdown overnight and the batteries may have drained. This suspicion was confirmed at 11:00 LST when we received an LGA communication window that was instigated by the onboard 'low power fault' response.

Eight minutes later we forced a reset and commanded a LGA DTE communication window with a higher data rate. The telemetry indicated that the BCB hardware fault protection[8] had tripped between 4:28 and 6:15 LST. The rover woke up 8:57 LST via a solar wakeup (neither the receiver nor the processor were on until this time).

When the low power condition was detected by the FSW, it executed the 'low power fault' response. The fault response changed the communications configuration and attempted to shutdown the vehicle. Since this shutdown process failed too, the hardware fault protection embedded in the BCB took the batteries off the bus and the electronics were finally turned off. Note that the mission clock is powered directly by the batteries, so the system did not lose time. Upon solar wakeup, the onboard FSW fault response eliminated the preplanned communication windows and created "fault windows" at 11:00 LST.

At 12:20 LST, we commanded the vehicle to shutdown for 24 hours. Thirty minutes later we commanded a beep and received no response. Another beep was commanded but not received, which confirmed that the shutdown was successful!

Sol 21 Conclusions

The rover had browned out overnight and was now in low power mode. We could prevent continuous resets by issuing the CRIPPLED command and we could shut the vehicle down.

We knew the vehicle would still boot up into the error state in the morning because the CRIPPLED command only sets volatile registers, which revert to the nominal state after the electronics are powered off. The Spirit rover required manual intervention each sol to restart into the "CRIPPLED" mode until the FLASH file system could be repaired. If we had to continue to operate in this mode, the mission return would be severely impacted due to an inability to make use of the FLASH file system to buffer science data products. It would also be impossible to support the early morning UHF relay windows.

However, at this point, we had regained control of Spirit, which was extremely important because our other rover "Opportunity" was due to land on Mars in the next 12 hours! Our Project Manager reported to the media that he had upgraded Spirit's condition from "critical" to just "serious".

[7] The CRIPPLED mode command is one of a few commands that are processed entirely by the command decode hardware. The command only sets individual bits in a software accessible register that the FSW interrogates during initialization to ascertain if any off-nominal initialization actions are required.

[8] The BCB is responsible for protecting the batteries from completely draining and it performed perfectly on this day.

The Next 11 Sols

For the next 11 sols, we attempted to diagnose the precise cause of the problem, recover science and engineering data still in the FLASH memory, and return the vehicle to a science-taking machine. During this time, the press started to refer to us as us “space-age surgeons”. The timeline of the anomaly investigation and recovery is shown in Figure 2.

We were limited by the fact that the vehicle would not wakeup until at 9:00 LST each sol (on solar wakeup) and it was only commandable until 15:30 LST when the Earth set. We could use the afternoon Odyssey orbiter UHF window, but morning UHF windows would fail because the vehicle would reboot after 15 minutes. Since all transmissions drain energy from the batteries, we had to limit the time the rover could send us data.

An additional challenge was accessing any of the diagnostic data or the FLASH file system when the system was in CRIPPLED mode. The team had to invent new techniques to capture the error cause and make these diagnostic data available in CRIPPLED mode.

So each sol, a number of activities were performed in parallel, including collecting telemetry, planning the next sol’s commands, exploring theories in the test bed, validating commands in the test bed, communicating with the management and the media and trying to get some sleep

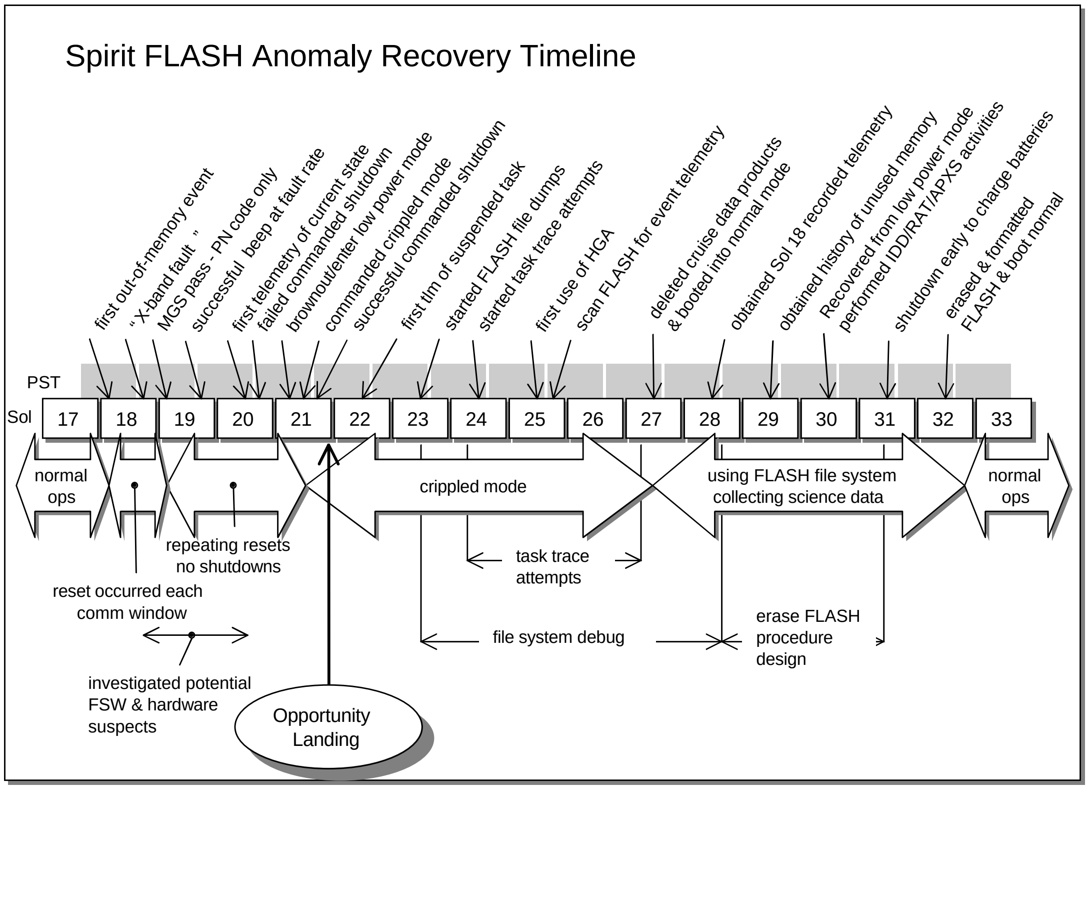

Spirit FLASH Anomaly Recovery Timeline
first out-of-memory event
“X-band fault ”
MGS pass - PN code only
successful beep at fault rate
first telemetry of current state
failed commanded shutdown
brownout/enter low power mode
commanded crippled mode
successful commanded shutdown
first tlm of suspended task
started FLASH file dumps
started task trace attempts
first use of HGA
scan FLASH for event telemetry
deleted cruise data products
& booted into normal mode
obtained Sol 18 recorded telemetry
obtained history of unused memory
Recovered from low power mode
performed IDD/RAT/APXS activities
shutdown early to charge batteries
erased & formatted
FLASH & boot normal
PST
Sol
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
normal ops
crippled mode
using FLASH file system collecting science data
normal ops
repeating resets no shutdowns
reset occurred each comm window
task trace attempts
file system debug
erase FLASH procedure design
investigated potential FSW & hardware suspects
Opportunity Landing

Figure 2: Anomaly events and recovery timeline

.On Sol 25, we calibrated the encoders and regained the use of the HGA, which increased our uplink and downlink capability. We also took two hazcam images of the science instruments still on the rock Adirondack. All of the science instruments and cameras were checked out in the subsequent sols.

On Sol 27, we deleted a large number of obsolete data products and their subdirectories left over from the launch FSW load (called the R7 data products), booted up in nominal mode and verified that the FSW was no longer autonomously rebooting. However, we observed indications that the FLASH file system was still corrupted and we decided to go ahead with the FLASH erase procedure.

On Sol 31, we shut down the vehicle early to fully charge the batteries and to cool down the electronics for the next sol, where Spirit need to be awake for nine hours. Early on Sol 32 at 6:30 LST, we manually booted the rover into CRIPPLED mode and copied the FSW image into RAM (for insurance). We then ran a sequence that erased small chunks of FLASH at a time, and then checked both FSW images (to ensure we didn't inadvertently corrupt or delete an image) after each chunk. This conditional sequence would have stopped if any command failed or if any FSW image corruption was detected. We chose this conservative strategy because there was a small fear that the root cause may still be in the FLASH devices or interface hardware.

After we verified the sequence completion and checksums, we commanded a FLASH file system format and booted back into normal mode. The FLASH memory was formatted as the system booted back up. The format was successful and the FLASH file system was ready to go.

At this point, the anomaly team declared victory and make plans for a work-free weekend!

3. THE ROOT CAUSE

DOS File System Usage

The MER FSW uses a commercial operating system and capitalizes upon many of the bundled functions and libraries that are included. Among these functions is a bundled file system library, which supports a DOS file system structure (referred to as the "DOS Library"). The file system model provides a very convenient and intuitive interface for many onboard functions. It is also straightforward to support in a test and simulation environment when real hardware is not available.

There are multiple file systems onboard the vehicle with partitioning to accommodate both differences in use and differences in the underlying storage media. There are no traditional disk drives on the vehicle. Each file system has RAM, EEPROM, or FLASH as the underlying storage media. Table 1 describes the onboard file systems.

The DOS Library module uses the operating system I/O services to interact with the physical memory devices via a device driver layer. Each memory type has its own device driver. In the case of the FLASH file system, an additional service layer provides management for erasure, write leveling, reading and writing.

Table 1: Onboard file systems

Name	Underlying Memory Type	Size	Purpose
Ram File System	RAM	4 Mbytes	Temporary storage of uplinked files
Temporary File System	RAM	2 Mbytes	Temporary data product store
FLASH File System	FLASH	224 Mbytes	Science and engineering data products
Primary Sequence	EEPROM	700 Kbytes	Sequence storage
Secondary Sequence	EEPROM	700 Kbytes	Redundant Sequence storage
Primary DPT	EEPROM	50 Kbytes	Downlink Table Storage
Secondary DPT	EEPROM	50 Kbytes	Redundant Downlink Table Storage

The DOS file system use on MER includes the use of many subdirectories as an organizing mechanism. Every file is stored in a subdirectory (no regular files are stored in the "root" directory). The primary telemetry mechanism is through the use of "data products" (see Ref. [2]). Data products are effectively files that contain the engineering or science data of interest plus a metadata file containing ancillary information (time of collection, vehicle orientation, etc.).

DOS Library Design Flaw

A design flaw exists in the DOS Library. The error is in how the DOS Library logic represents deleted files internally and how this impacts the size of the interlinked data structures held in the DOS Library private memory area.

The DOS Library module creates an interlinked collection of data structures[9] in the flight computer memory to represent the file system structure (directories, sub-directories, number of files in directories, etc.). The initial memory space for this data structure collection is allocated out of the free memory space at initialization time. The DOS Library logic replies on the services of another bundled OS service ("Mem Library") to manage this private space.

When a file or a sub-directory is created, this internal representation of the file system structure is updated (as is the file system itself in FLASH). As the file system structure expands, one or more additional memory allocations, from the private area, will be required to provide the space for new elements of the data structure collection.

A DOS subdirectory is nothing more than a file itself except that it is specially structured as an array[10] that contains the information about the files and the subdirectories. The information contained within an array element includes the file name, attributes, date/time, file size, and a pointer to the starting cluster of the file (a cluster is the smallest unit of space that can be allocated to a file). When a subdirectory is first created, all of the array elements are initialized to the unused state (all zero values). As files are created in the subdirectory, the array is searched in a linear fashion until the first unused (or the first deleted) entry is found; the information for the new file is then placed in that entry. Never-used entries and entries for deleted files are treated differently. When a file is deleted, the only action performed is to change the value of the first letter in the file name to a special value (the infamous hex byte code value E5h). This is a special tag that tells the system "this file has been deleted".

The DOS Library logic manages both the actual file system (on the FLASH) and the private, RAM resident, internal data structure collection in parallel. The internal data structure collection is created when the file system is "mounted" during initialization and then updated as file interactions occur. This data structure collection is not retained when the vehicle is turned off but it is always recreated when the processor is turned back on. When the file system mount action occurs during initialization, the entirety of the file system structure is traversed in order to build the initial representation. Each subdirectory is read from the "disk" and traversed in order to create its internal data structure collection. Directory list entries that have never been used do not require any representation in the data structure collection (since they never represented a file) and no memory space is required for them. However, deleted file entries are represented. The premise being that this is necessary to maintain the symmetry between the internal representation and the actual file system structure.

[9] This was done to optimize performance.

[10] This is the simplistic description. The detailed description can be found in Ref [1].

The unfortunate consequence of this feature is that when a file is deleted, the space in the file system (on FLASH) is freed but the internal data structure space is still required. In fact, the space is never released.

Effectively, the "high water mark" or the maximum number of files that ever existed in the subdirectory is the direct contributor to the amount of memory space that will be needed to represent the file system structure.

The ramification of this is twofold: simply deleting files does not reduce the amount of memory needed to represent the file system structure, and the total size of the file system structure (from the perspective of the total memory space required to represent it) is not expressed by the number of current files but by the sum of the maximum number of files that had ever existed in each subdirectory.

Configuration Errors

Two configuration errors conspired to place the system into a condition where it would reset repeatedly and also prevented the vehicle from autonomously shutting itself off to save power. A configuration error in the DOS Library module allowed the size of the private memory area to expand by allocating additional space from the free system space[11]. A configuration error in the Mem Library module silently resulted in a suspended task when the request for additional memory could not be satisfied.

DOS Library Configuration Error

As described above, the DOS Library module creates an interlinked collection of data structures in the flight computer memory to represent the file system structure. The initial amount of memory that was allocated to this private area is one of the DOS Library configuration parameters. In the MER configuration, this initial size is 256K bytes. When new elements need to be added to the internal data structure collection, a block of free space from this area is used for the new element(s). The DOS Library contains logic that allows for the expansion of the

[11] The free system memory is > 4 Mbytes normally

private memory space by requesting additional space from the free system memory area. However, the MER development guidelines do not permit dynamic memory allocation. Unfortunately, this configuration parameter was not set correctly; expansion was permitted. In this erroneous configuration, when the initial space was exhausted, the DOS Library logic made allocation requests in increments of 256K bytes from the free system memory. This new space is added to the DOS Library private area. Additionally, a second parameter can be set to limit the growth to a maximum amount of space. This second parameter was also in error; no limit was specified. These configuration errors allowed all of the free system memory space to be consumed.

Mem Library Configuration Error

Another configuration error existed in the Mem Library module. The Mem Library module logic provides a simple malloc/free service[12] and operates using memory space previously provided to it. When an application program (or library) requests a buffer of a given size, the Mem Library provides a pointer to a buffer of that size. The available space shrinks accordingly. The actual process involves the management of a linked list of data structures that represent both the allocated (in use) space and the unallocated (free) space. In the MER architecture, the free system memory is managed using this service.

By convention, applications are allowed to request memory allocations only during initialization. This space is never returned to the free system memory space. No dynamic allocation and release of system space is permitted (by convention).

The DOS Library consumed the free system memory space until no further allocations could be satisfied. Ideally, when this occurred, the system should have reacted by failing the file system activity in progress (such as an file open, file creation, or file write). Instead, again due to a configuration error, the task performing the file system activity was silently suspended. The Mem Library module can be configured to react in various ways when certain usage errors occur. In the erroneous configuration, the MEM Library was configured to suspend any task attempting to allocate space when there is insufficient free space to satisfy the allocation request.

The suspension of a task is a severe error and is never supposed to occur. The MER architecture includes a "software health" function that continuously checks for suspended tasks. When the health monitor function identifies a suspended task, it forces a reset of the avionics electronics, including the flight computer, and the re-initialization of the FSW.

[12] Actually it provides a far more general memory management service.

Repetitive Reset

As described above, each time the software initializes, the FLASH file system is mounted. This caused the re-creation of the RAM resident data structures, which, in turn, consumed all of the available memory. Subsequently, when the FSW attempted to create a new file in the FLASH file system, insufficient space existed in the private area (the private area has been expanded many times already). The DOS Library attempted to allocate additional space from the free system memory, but the request could not be satisfied. The task that attempted to add the file was suspended. The FSW "health check" mechanism detected the suspended task and signaled a severe error, resulting in a reset (after the delay period).

The cycle then repeated.

Unfortunate Side Effects

An unfortunate side effect of the task suspension was that the file systems became inaccessible to the other FSW tasks. The DOS Library logic uses semaphores to protect critical regions. Unfortunately, the memory allocation failure occurred within one of these critical regions. With the task suspended, the access control semaphore was never released and no other tasks could gain access. This side effect prevented recorded telemetry from being produced (the task which reads files from the file system was blocked attempting access), prevented the shutdown (the task which controls the shutdown actions could not idle the file system), and was the cause of the repeating telemetry (the task which pushes new telemetry data into the hardware was also blocked).

4. HINDSIGHT IS 20-20

Now that the root cause of the problem is understood and the overall behavior of the system can be traced, it is possible to correlate the observed behavior and data with the actual events that occurred on the vehicle.

Sol 18 Revisited

Three FSW resets occurred on Sol 18. The first was due to the creation of the large number of motion history data products that resulted in new files being created in the FLASH file system. This increase in the number of files required additional space in the DOS Library private memory area that led to additional memory allocations from the system memory pool. When one of these allocations failed, the Mirror Ram to FLASH (MRF) task (which creates data product files in the FLASH file system), was suspended. The health monitoring function detected the suspension and forced the first reset.

The next reset did not occur for another two hours. A scheduled communication window contained a HGA encoder calibration to ensure the HGA is pointed correctly. Unfortunately this calibration created more motion history report files, leading to another out-of-memory event and reset.

The afternoon HGA window failed because the encoders were left un-initialized following the reset. The on-board fault protection declared an 'X-band fault', aborted the window and set the communication rates to the fault configuration.

During the afternoon UHF window, another out-of-memory event occurred when the FSW autonomously created a data summary report file[13]. Earlier commanded beeps on Sol 18 succeeded because no data summary report file is generated for beeps.

Sol 19 Revisited

When the rover woke up for the early morning UHF window on Sol 19, another out-of-memory event occurred. The reset was delayed 15 minutes, so the FSW continued with the communication window, including turning on the UHF radio. The reset took place before any data was pushed to the radio and the FSW turned the radio off in the next initialization, so the orbiter only received PN code for only two minutes and 20 seconds.

This is the probable start of the continuous delayed resets (given the number of resets recorded in telemetry). Many of the future communication windows were interrupted because the reset would take place during the setup or transmission portion of the window.

Sol 20 Revisited

Our first telemetry on Sol 20 was received at the fault configuration rate and garbled. A FSW reset had occurred 10 minutes into the window. Due to the size and management of the hardware buffer space, no valid data were radiated before the radio was shut off after the reset. On the next commanded LGA session, 11 complete transfer frames were transmitted, but they were all identical. The reason for this repeating nature was the FSW task that packages frames was blocked from generating new frames. The radio's transmitter continued to pump out the last frame that was in its hardware buffer. A delayed reset terminated the session after approximately 20 minutes.

Our next commanded window occurred within the one hour delayed reset period, so it was not interrupted by a reset as the other windows had been. Only real-time data were received because recorded data is stored in FLASH memory and the FSW could not access it.

The vehicle would not shutdown on Sols 19 and 20 because the task that suspended during initialization held the semaphore that was needed by the shutdown logic to gracefully idle the FLASH system. Since the shutdown logic could not access the semaphore and there was no timeout or other logic to force the continuation of the shutdown actions, the shutdown process halted and the processor was never turned off.

Sol 21 Revisited

The CRIPPLED mode mechanism allowed us to regain control of the vehicle. This debug feature forces the FSW to create a file system in RAM with the same logical device name as the FLASH file system. The file system created in RAM is empty; there are no residual files and subdirectories. As result, the memory space necessary is considerably less, so the FSW was able to complete the initialization without error.

The system is not as capable in this configuration since the RAM based file system is 10 times smaller and volatile. However, all functionality is restored. All of the application and system software uses the logical file system name so the change in configuration is completely transparent to the rest of the software.

5. CONTRIBUTING FACTORS & LESSONS LEARNED

Although the anomaly has a precise technical cause, there were other contributing factors in which the cause and effect relationship is much more subjective. This section describes these other contributing factors and lessons that have been identified during and after the anomaly investigation.

Compressed Schedule

Many of the contributing factors are related to the compressed development schedule (three years from concept to launch). The MER project was always challenged by the tight schedule and the FSW development was tailored to address this reality. During

[13] This engineering telemetry data product tells operators which data products exist. The report is autonomously produced for each communication window.

the development process there was a continuous reprioritization of activities and focus. One impact of this dynamic process was that only the highest priority issues and problems could be addressed. Several of the contributing factors can be described as simply lower priority activities whose completion or completeness could not be realized.

Incomplete Development

The behavior of the DOS Library module, at least in regard to expanding the private memory area, was known by a subset of the FSW development team. This was identified as a potential problem area and it was recognized that the behavior of the system would need to be characterized in order to establish the flight configuration. The DOS Library configuration parameters were set with the expectation that over the course of the test activities, a pattern of use could be identified that would permit a final configuration to be identified. The intent was to run the system using the initial configuration, then gather data to:

- Investigate how often requests for additional memory occurred (some were expected). A significant difference from what was expected should have triggered additional investigation.
- Identify the maximum amount of space ever allocated.
- Identify any combination of activities or operations that resulted in the free system space being entirely consumed. This specific situation never occurred.

This investigation was never completed so the configuration remained unchanged. No errors were identified during testing and as result, there was no immediate need to finalize the configuration.

Lesson: Many lower priority development activities could not be completed in the development time frame. This was an accepted risk. There should have been a formal management of incomplete items that required resolution. This would have had the beneficial effect of creating a checklist of items requiring closure and thus would have insured that this specific investigation was completed.

Unanticipated Behavior

There was a belief among the FSW development team that the system would not exhibit the behavior that is the root cause of the anomaly (even though it was known to be a possibility). This posture affected the priority of many activities including the investigation of related problem reports, the analysis of test results, and the closeout of related investigations. This unfounded belief was based on a number of factors including:

- A recognition that the DOS Library implementation will request and release temporary buffer space from the free system memory during its normal operation. This is unrelated to the private memory area use. This is not an ideal characteristic, but the implementation is correct; no memory leaks occur. It had the unfortunate side effect of desensitizing the developers and testers to other similar, but incorrect behavior, as was seen in the anomaly.
- A belief that the only limitation regarding the use of the FLASH file system was the total file system capacity. At the time there was no indication or knowledge regarding the mismanagement of deleted files.
- A false impression that the number of files in the file system would be limited to the number of data product files that would be allowed to exist. The data product management software limits the number of data products to 8192 unique items.

Lesson: Understand the behavior of any Commercial Off-The-Shelf (COTS) software that is utilized. There should have been an effort to review the implementation of the DOS Library module (and others) with an emphasis on review of the basic logic and function. The vendor could have participated in these reviews and could have been tasked to brief the FSW development team on these functions.

Lesson: Enforce the design guidelines. There was a design guideline that prohibited allocation of space from the free system memory space after initialization was complete. This guideline was enforced for the JPL-developed software and it should have been enforced for the COTS software as well.

Lesson: Verify assumptions regarding the expected behavior. No developer was explicitly assigned to investigate or review the DOS Library. This allowed the module to be used without detailed review or scrutiny. There needed to be a responsible individual to address any design or test issues.

Inadequate Telemetry

The telemetry important to detecting the underlying problem was not a part of the normal telemetry process. The FSW does produce the required information, but it is telemetered in a format that does not feed into the normal data analysis tools. In fact, only a few FSW team members were even aware that the information existed.

Figure 3 shows Spirit's in-flight telemetry of the free memory space from launch through recovery. Note we did not connect the segments on the figure because that would be misleading regarding the rate of decrease in the system memory space. Insufficient data exists to extrapolate when each request for new space occurred. This data was not examined until the anomaly occurred.

Lesson: The FSW should have included flight telemetry for resources (such as the free system memory space) so that the actual and expected behavior of the system can be compared.

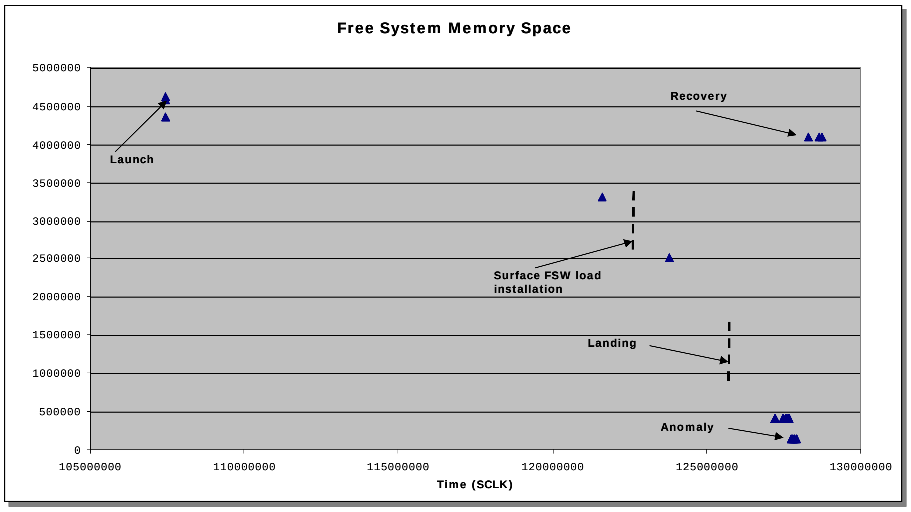


Figure 3: Flight telemetry of Spirit's free memory space

Test Program Design

The MER system and FSW test activities were extensive and comprehensive. The test program included unit, design, functional, system level, verification, and validation activities. These test activities were performed using test beds with various degrees of fidelity including multiple test beds with hardware in-the-loop configurations. The test activities were performed by multiple organizations within the MER project structure.

The project performed several realistic test exercises intended to demonstrate the performance of the system and personnel in a realistic flight-like manner. There were several project operational readiness tests (PORTs) where the system was exercised in a manner believed to be representative of the actual flight use. The anomaly was never seen during these (or any other) test activities. There are several factors that contributed to this result:

- The operational tests did not exercise the system fully in a flight-like way. This was because both the surface operation processes and the FSW were immature when the tests were done and this limited the activities that were performed during the test. As a result, the operational tests did not produce the diversity of data products or the volume of data products that were created during the surface mission.
- The number of files on the flight vehicle on Sol 1 was greater than the number of files on test bed during the operational tests. Although we attempted to reproduce the flight-like conditions by walking through launch, cruise, entry, descent, and landing, we did not reproduce every turn, maneuver and communication window performed in flight.

Observation: The operations team exercised the system in what was considered a "flight-like" manner during the operational tests. Once the rover reached the surface of Mars, the experience and training allowed the operations team to develop activities that exceeded the envelope of the test activities. It is not apparent that anything other than a longer duration operational test would have exposed the anomaly, but a longer operational test (over 11 sols) was impractical within the overall MER schedule.

Test Data Review

A post-anomaly review of one operational test showed that the memory leak was evident in the data. Figure 4 shows a portion of the telemetry of free system memory space and it indicates a memory consumption trend. The data set is sparse because the telemetry for this data was set to a low priority and was not consistently telemetered. Unfortunately, we did not analyze this data during the test execution or during the post-test data review.

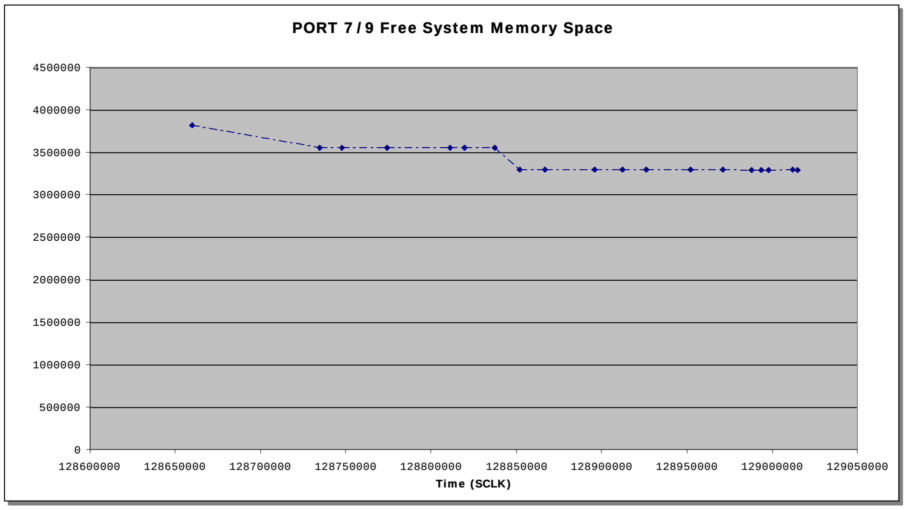


Figure 4: Project operational test 7/9 free system memory space

Observation: The analysis of this test data by the FSW team was a lower priority activity when compared to other activities occurring in the same timeframe.

Lesson: A suite of tests and automated analysis tools should have been created early in the development process. The FSW should have included support for reporting resource usage. The system (FSW, test scripts, ground support equipment) should have included (additional) support for collecting, archiving, and analyzing this information.

Build for the Unexpected

The idea for CRIPPLED mode originated on the Mars Pathfinder project. Without this capability, the MER mission may well have been lost.

Lesson: The system design should include the mechanisms to address both problems envisioned as well as the unforeseen and "unknown-unknowns". Contingency commands and similar mechanisms need to be included where they can be put to use to resolve both failures and design errors such as the Spirit anomaly scenario. Of course, the contingency commands are insufficient unless the system is robust enough to take the necessary actions to both maintain a safe configuration and to let the operations team know what is happening onboard.

Observation: The MER design placed the primary responsibility for initiating communication on the rover. This removed the ambiguity of a situation where no data or signal is received, which would force the operations team to blindly attempt to initiate contact. Just knowing that the rover was attempting to communicate at a particular time gave us clues as to what was happening onboard.

Lesson: Build in the ability to continue autonomous communication, even when onboard fault protection responses run. The onboard fault responses did initiate communication at unique times so that even if a full data set didn't get through, the time of the communication attempt described what's happening onboard.

Other Lessons

Lesson: A different file system type, or a more robust implementation, is required for future missions. The lack of compaction for deleted files in the directory structures is a fundamental flaw for long duration missions.

Deleted: Similarly, the in-memory representation of an active file system directory structure is an unnecessary complexity that reduces the confidence in the correctness of the existing implementation.

6. OPERATIONAL & FSW CHANGES

Operational Changes

The problem can be avoided by better oversight of the file system use, at the cost of a more conservative approach to gathering science data. Since the amount of memory consumed is directly related to the structure of the file system and the number of files, it is possible to monitor the free memory space and to terminate activities if the amount of free space drops below a designated limit.

The anomaly team recommended, and the project adopted, the following changes and guidelines. These apply to both rovers.

- Monitor the amount of free memory space in the system. If the remaining space drops to 800 Kbytes or less, then terminate all science and engineering activities.

- Aggressively command the deletion of received data products after they have been received on Earth. The data management team has reduced the latency for this action from 48 to 24 hours.

- Remove subdirectories when no more data product files exist. This action forces the release of memory used to represent the subdirectory and eliminates the space utilized for deleted file entries. This operational guideline became unnecessary once the FSW was updated in April.

- Permanently upgrade the priority of the data products that contain the telemetry generated during initialization that shows the free system memory space and the results of the autonomous file system check.

Onboard Post-Anomaly FSW Changes

Several FSW changes were included in the April 2004 FSW load to address the problems and issues discovered during the anomaly investigation. These changes included:

- Additional logic to remove the directory list structure entries for deleted files. This change "compacts" the directory list structure and thus, removes the "high water mark" effect. This compaction action runs during the initialization process before the FLASH file system is mounted.

- Additional logic to autonomously enter CRIPPLED mode when multiple resets have occurred. This autonomous action allows the system to initialize correctly if another similar anomaly should occur.

- Modifications to the shutdown logic to use the alarm clock hardware function as a secondary watchdog. This change forces a power cycle of the avionics electronics in cases where the FSW cannot perform all of the shutdown actions. If this occurs during the night, the vehicle will remain powered off until sunrise.

- The register set by the CRIPPLED command is volatile and the value is not retained across power cycles. The FSW was changed to also examine one of the spare non-volatile registers bits, so the operations team may permanently force the system to initialize in CRIPPLED mode, if necessary.

- Modified the shutdown logic to wait a limited amount of time for the FLASH file system to become idle. In the previous FSW version, the shutdown logic would wait indefinitely. This change addressed the semaphore/deadlock issue discovered during the anomaly investigation.

FSW Changes Considered but Not Included

In hindsight, the correct implementation would have been to limit the private memory area used by DOS Library to a fixed size. This strategy is consistent with the way all application memory space on MER is allocated and managed. In this configuration, the free system memory would not have been consumed and no out-of-memory event would have occurred.

However, this change was not incorporated into the new FSW. This was a considered decision that balanced the benefit versus risk of performing this optimal change. These tradeoffs included:

- Although a modification to limit the size of the DOS Library memory area is straightforward, the change might expose other, unexplored, behavior when the private memory space is exhausted. The test program to verify the overall behavior of the system in all operational modes would be challenging, complex, and time consuming.

- The vehicle lifetime is limited. A test program of this complexity would have significantly delayed the upload of the new FSW version. The new FSW version also contained many changes unrelated to the anomaly that increased the robustness of the system, optimized the collection and processing of science data, and enhanced the mobility capabilities of the vehicle

- The operational changes addressed the necessary steps to limit the number of active files in the file system.

7. Acknowledgements

The recovery of the vehicle was truly a team effort. Many members of the operations team, including the mission managers, the flight directors and the subsystem teams made extremely valuable contributions to the recovering from this anomaly.

There is a small group of individuals the authors would like to recognize for their extraordinary effort: Khaled Ali, Jeff Biesiadecki, Mike Deliman, Jim Donaldson, Ed Gamble Jr., David Hecox, Roger Klemm, Todd Litwin, Cindy Oda, Ed Odell, David Smyth and Joseph Snyder.

The rover design described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

8. Conclusion

In January 2004, Spirit suffered an anomaly the prevented nominal communication for several days. Given every clue the rover presented us, the anomaly team successfully diagnosed the problem and recovered this rover back to perfect health. These clues included when the rover communicated (and when it didn't), the rates it used to transmit (pre-planned rates vs. onboard fault response rates), the type of information it transmitted (real-time vs. recorded, repeating event reports, PN code), and the commands to which the rover would respond (as well as those to which it would not respond).

The root cause was a design error in the file services FSW module that resulted in an out-of-memory event, causing a processor reset. During re-initialization, when it would access the FLASH memory, another out-of-memory event would occur, so the rover experienced repetitive resets. This rippled into difficulties communicating with Earth, as well as the inability to shutdown the solar-powered rover's electronics at night to save power.

The predominant factor that led to this error was the compressed design schedule. Incomplete development, inadequate telemetry and limited testing were a direct result of the breakneck development pace. In the end, it was a dedicated, insightful team that had designed a system with built-in diagnostic tools, autonomous communication and robust fault protection that led to the recovery of Spirit.

9. REFERENCES

[1] Glenn Reeves, Tracy Neilson & Todd Litwin, "Mars Exploration Rover Spirit Vehicle Anomaly Report," Jet Propulsion Laboratory Document No. D-22919, July 5, 2004.

[2] Joseph F. Snyder, David E. Smyth, "Data Management for Mars Exploration Rovers", Jet Propulsion Laboratory Document No. D-30712, July 19, 2004.

10. BIOGRAPHY

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Glenn E. Reeves *is a Principal Engineer at the Jet Propulsion Laboratory. Most recently he was the Flight Software Architect and team leader for the Mars Exploration Rover project. He also performed a very similar role for the Mars Pathfinder mission. The flight software architecture he developed for the Mars Pathfinder mission has been adapted and used on several subsequent spacecraft. Glenn's professional passion is the development of evolutionary and revolutionary spacecraft autonomy. His current assignmennt is with the JPL Office of the Chief Engineer.*

Tracy Neilson *is a Senior Engineer in the flight systems engineering section at the JPL. Her role on the MER project was fault protection lead and she designed various onboard algorithms for the cruise and surface phases. Tracy's previous assignments included launch behavior specification and on-onboard fault protection design for the Deep Space 1 mission (a project that flight-tested new technologies such as ion propulsion in deep space and autonomous navigation) and the international Gravity and Climate Recovery Experiment (GRACE) mission. She was also the Attitude and Articulation Control Subsystem lead for the Galileo mission's Jupiter orbit insertion and probe relay, and for the first two years of orbital operations.*