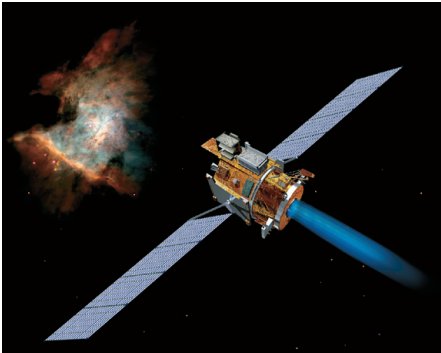


## NASA Uses Stateflow and Simulink Coder to Generate Fault-Protection Code for Deep Space 1



The Deep Space 1 spacecraft. Image courtesy of NASA/JPL/Caltech.

On October 24, 1998, NASA launched a Delta II rocket from Cape Canaveral in Florida. On board was a spacecraft named Deep Space 1 (DS1).

The DS1 mission was part of NASA's New Millennium Program, a series of deep-space and earth-orbiting missions designed to test and validate 12 high-risk new technologies. The ultimate goal, according to Dr. Wesley Huntress, NASA's Associate Administrator for Space Science, was to "enable these technologies to be used with greater confidence on scientific missions of the early 21st century."

DS1 was also designed to test an unofficial "13th technology": the model-based code generation of the spacecraft's system-level fault-protection (FP) software. This was accomplished using MATLAB®, Stateflow® and Simulink Coder™.

### The Challenge

A robotic spacecraft like DS1 must have the intelligence and autonomy to monitor and control itself at a great distance from Earth throughout its useful life. The ever-increasing distance between the spacecraft and Earth limits the ground teams' ability to respond quickly to changes in spacecraft conditions. FP algorithms ensure either that a mishap can be circumvented or that contact with Earth can be reestablished if it is interrupted.

Before the DS1 project, FP system software was typically hand coded by a team of software developers. The constraints of the DS1 mission design and development cycle, however, together with limited budget and resources, dictated a radical departure from

past practices. A further complication was the requirement that the FP system direct postlaunch activities, a task usually handled with sequences.

DS1 was operated by the Jet Propulsion Laboratory (JPL), a division of NASA managed by the California Institute of Technology. JPL had just over one year to design and code the spacecraft's FP system. A shortage of software engineers meant delays if traditional design-to-code methods were used. JPL had to find a faster, more efficient way—a "13th technology."

### Searching for a Better Method

Previously, the monitors for the DS1 Remote Agent Experiment had been successfully created using automatic code generation. This seemed a good possibility for the FP system, but it would be necessary to shift the design and development of the system-level FP software from lower-level, C constructs to higher-level requirements, issues, strategy, and tradeoffs.

JPL also needed a design notation that was clear enough to allow several people to follow an analysis review and concise enough to fit the tight schedule.

### The Solution

JPL decided to use state charts and automatic code generation. This approach would allow them to separate design and implementation concerns.

### Choosing the Design Tools

After evaluating several tools, JPL chose the "one-stop" solution provided by Stateflow. Of particular importance to the JPL team was the fact that, like all MathWorks

### The Challenge

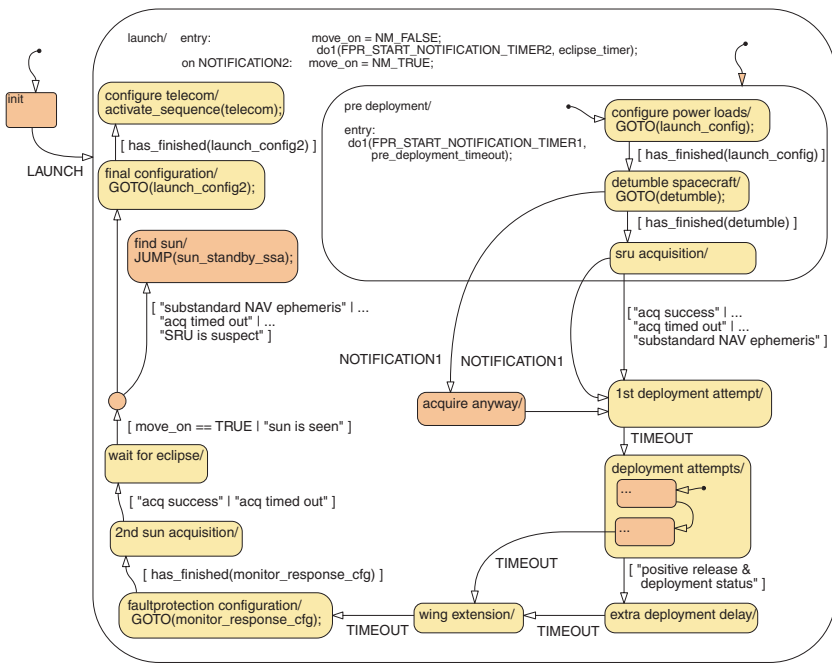
Design and code the fault-protection system of a robotic spacecraft within strict time and budgetary limits

### The Solution

Use Stateflow and Simulink Coder to create the logic flow and automatically generate code for the fault-protection system

### The Results

- Prolific generation of reliable code
- Record-breaking operational lifespan
- Rock-solid performance



A section of the launch state chart, showing sun acquisition and predeployment of the DS1 solar array panels.

```

/* Fault Protection Responses use three global variables
 * to access information about:
 * - monitor inputs
 (fpr_monitors_inputs)
 * - fault responses parameters
 (fpr_responses_parms)
 * - fault responses state variables (fpr_responses_state)
 */
extern fpr_monitors_inputs_t fpr_monitors_inputs;
extern fpr_responses_parms_t fpr_responses_parms;
extern fpr_responses_state_t fpr_responses_state;

/* Enumeration of all the events used in this machine */

typedef enum {
CALL_EVENT = 0 /* A general event for calling any chart */
,EVENT_LAUNCH = 3 /* Exported */
,EVENT_TIMEOUT = 2 /* Exported */
,EVENT_NOTIFICATION1 = 1 /* Exported */
}

```

A sample of the FP code generated by Simulink Coder.

products, Stateflow is an open system. This meant that they would be able to customize both the state charts and the automatic code generation to meet the demanding requirements of the DS1 project.

JPL also knew that Stateflow would let them reuse logic from another of their FP systems, that of the Mars Pathfinder. It would also enforce standard diagrammatic conventions for representing state charts and the resulting logic and allow systems engineers, rather than software engineers, to design and implement the system.

For code generation, JPL chose Simulink Coder, primarily because the MathWorks code-generation algorithm provides an open and customizable architecture. Two further benefits impressed them: the code is isomorphic to the topology of the state chart, and code generation is driven from Abstract Syntax Trees.

### Executing the Design Architecture

The JPL team based the concept of the DS1 FP system on the successful Mars Pathfinder FP system. This design follows a uniform architecture in which EVR\_FAULT events map to fault-response functions and the FP system executes fault-response functions, after which all faults are cleared.

The FP system has two levels of fault-response priorities: uninterruptible (a fault response runs until completion, no matter what else happens) and interruptible (a fault response can be put on hold at specific points).

Sensor monitors extract features from raw sensor data to detect symptoms of normal and abnormal behavior. Although symptom detection is hardware-specific, the overall FP architecture views sensor monitoring as a data-flow process, performing functional transformations of monitor state and sensor data.

In addition to the FP applications, JPL used automatic code generation to produce the flight software that guided DS1 through postlaunch separation and initial signal acquisition.

*“Until Deep Space 1, state charts and automatic code generation technology had not been used on large systems for spacecraft avionics software. MathWorks tools made this approach possible.” —DR. WESLEY*

*HUNTRESS, ASSOCIATE ADMINISTRATOR FOR SPACE SCIENCE, NASA*

### Controlling the Initiated Launch

The DS1 launch initiated control- and system-level activities following separation of the spacecraft from the Delta II rocket. The launch state chart is a three-level hierarchy. At the top level, there are two states: “init” and “launch.” The transition from the “init” state is predicated on the “launch” event. The FP system broadcasts this event after it receives an indication that the spacecraft has booted up and separated from the launch vehicle.

JPL used Stateflow diagrams to enable the FP system to send commands to several managers to configure the system for initial acquisition. The Stateflow diagrams also cause the FP system to command the attitude control system (ACS) to estimate the spacecraft’s attitude and point-to-sun and then wait for an acknowledgement from the ACS.

### Testing the Fault-Protection System

The FP system was tested both on the ground and in space. Ground testing was conducted in three phases: unit testing, when every branch in the FP system logic is tested in a standalone environment; test-bed testing, when only the most likely fault scenarios are tested; and system testing of behaviors that use extensive software-to-hardware interfaces.

During postlaunch separation and initial signal acquisition, the DS1 FP system prevented several potentially damaging malfunctions. For example, during the second sun acquisition state, the SRU (Stellar Reference Unit) processor began producing internal checksum errors and illegal software variable values. This produced a persistent Celestial

Inertial Reference Loss (CIRL) condition, and the CIRL monitor declared a fault.

The FP system then suspended the launch state chart and started the CIRL response, which power-cycled the SRU and corrected the problem after a few minutes. The SRU then reacquired the sun signal and began tracking. When the turn-to-sun was completed, the FP system reconfigured the heater states, turned on the power amplifier, and initiated the X-band downlink.

### The Results

**Prolific generation of reliable code.** Between June 1997 and January 1999 (when the FP code was last modified), JPL automatically generated over 230,000 total lines of code using Simulink Coder. This translates to approximately 180 lines of code per day.

**Record-breaking operational lifespan.** DS1 flew within 26 km of Asteroid Braille on July 29, 1999. New software systems and commands were built and installed when the spacecraft was more than 300 million km from Earth. Now that the hardy spacecraft has resumed normal operations, it has added yet another record to its extensive list of accomplishments: the longest operating time for an ion propulsion system in space.

**Rock-solid performance.** Having completed 100% of its original testing mission, NASA has extended DS1’s mission to include an encounter with a distant comet. The spacecraft has pushed well beyond the limits of what it was designed to do, and its fault protection system continues to work, day in and day out, exactly as intended.

### Industry

- Aerospace and defense

### Application Areas

- System design and simulation
- Embedded code generation
- Embedded systems
- Control systems

### Products Used

- MATLAB®
- MATLAB Coder™
- Simulink Coder™
- Stateflow®

### Learn More About JPL and Deep Space 1

[nmp.jpl.nasa.gov](http://nmp.jpl.nasa.gov)