Model-based Autonomy for the Next Generation of Robotic Spacecraft

Michel Ingham
Lorraine Fesq, John Van Eepoel, Brian Williams
Model-based Embedded and Robotic Systems Group
MIT Space Systems Laboratory

Michael Pekala, David Watson
JHU Applied Physics Laboratory

October 18th, 2002

Objectives

- Define "model-based autonomy"
- Describe model-based executive technology (Titan)
- Describe application to representative space mission (ST7-Autonomy concept study)

Objectives

- Define "model-based autonomy"
- Describe model-based executive technology (Titan)
- Describe application to representative space mission (ST7-Autonomy concept study)

Model-based Autonomy

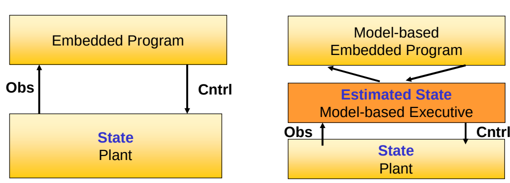
Creation of embedded & robotic systems that manage interactions automatically, by reasoning from models of themselves and their environment.

- Enabling technology for highly robust spacecraft.

Model-based Autonomy

Creation of embedded & robotic systems that manage interactions automatically, by reasoning from models of themselves and their environment.

- Enabling technology for highly robust spacecraft.
- Adopts notion of model-based programming.



Model-based Autonomy

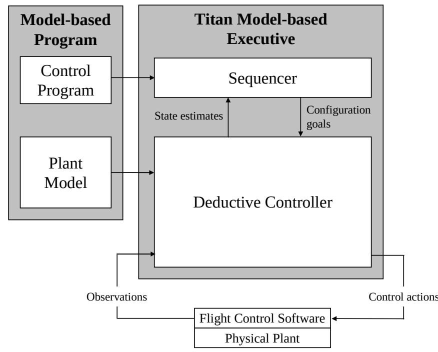
Creation of embedded & robotic systems that manage interactions automatically, by reasoning from models of themselves and their environment.

- Enabling technology for highly robust spacecraft.
- Adopts notion of model-based programming.
- Automates onboard sequence execution by tightly integrating goal-driven commanding, fault detection, diagnosis and recovery.

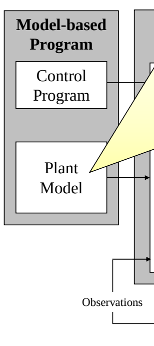
Objectives

- Define “model-based autonomy”
- Describe model-based executive technology (Titan)
- Describe application to representative space mission (ST7-Autonomy concept study)

Architecture For Model-based Execution

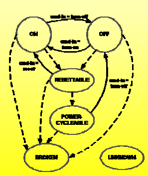


Model-based Execution

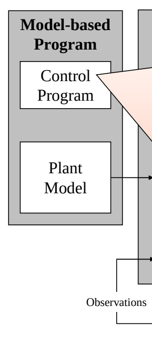


Plant model describes behavior of each component:

– nominal and off-nominal modes
– qualitative constraints
– probabilistic transitions
– costs/rewards



Model-based Execution

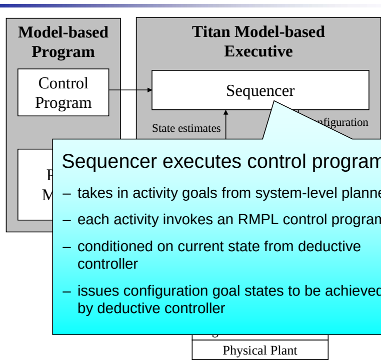


RMPL Control program specifies desired state trajectories:

– encode reactive control sequences
– refer to hidden states as if they were directly observable & controllable
– include constructs for expressing:
  - Concurrent Task Management
  - Contingency Handling
  - Iteration
  - Preemption

```
...
{when (ground-command = start-playback) donext
  {sequence
    {do
      {always (transmitter-mode = downlinking-data)}
     watching (ssr-playback = complete)}
    (transmitter-mode = idle)
    (downlink-status = succeeded)}}
...
```

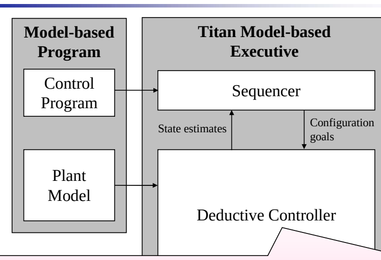
Model-based Execution



Sequencer executes control program:

– takes in activity goals from system-level planner
– each activity invokes an RMPL control program
– conditioned on current state from deductive controller
– issues configuration goal states to be achieved by deductive controller

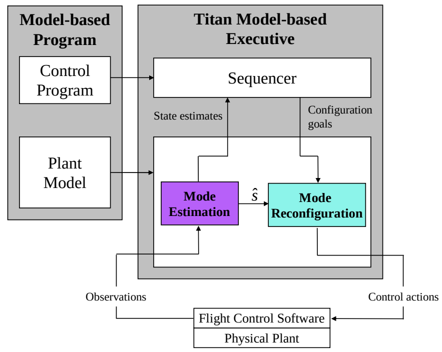
Model-based Execution



Deductive Controller maps from states to sensors/actuators:

– Mode Estimation uses plant model and observations to deduce current state (mode tracking, fault diagnosis)
– Mode Reconfiguration takes in configuration goals, reasons through plant model to compute commands that achieve goal (nominal, fault recovery)

Model-based Execution



Objectives

- Define "model-based autonomy"
- Describe model-based executive technology (Titan)
- Describe application to representative space mission (ST7-Autonomy concept study)

New Millennium ST7 Autonomy Mission Concept

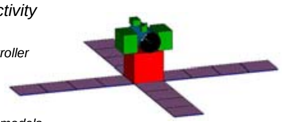
- 6-month concept definition phase ended January 2002
- autonomy-ready s/c design based on primarily off-the-shelf components
- mission design highlighting:
  - onboard execution of activities normally commanded from ground
  - science-driven execution
- software testbed demonstrations of component technologies

Scenario One : Downlink Data Block Activity

- *Demonstrate control sequencer operation*
- *Demonstrate interaction with deductive controller*

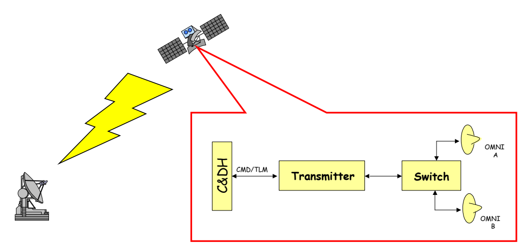
Scenario Two : Bus Controller Failure

- *Demonstrate mode estimation and mode reconfiguration on more sophisticated plant models*

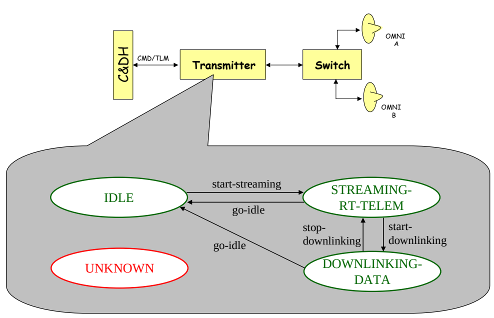


Scenario 1: Downlink Data Block Activity

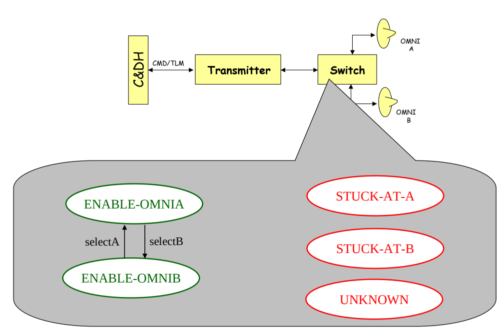
Objective: downlink a block of data from onboard storage to a specified ground station



Downlink Data Block Plant Model Overview



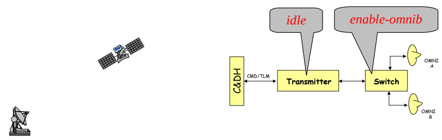
Downlink Data Block Plant Model Overview



Downlink Data Block Control Program Overview

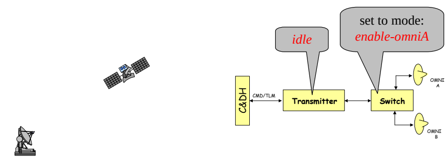
Initial State:

- omniA in view of ground station
- no ground command received
- omniA in nominal mode (remains so throughout)
- omniB in nominal mode (remains so throughout)
- transmitter in idle mode
- switch set to enable omniB



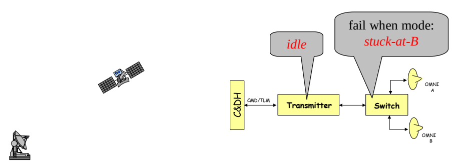
Downlink Data Block Control Program Overview

- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)



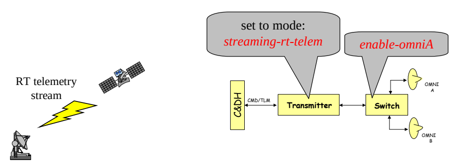
Downlink Data Block Control Program Overview

- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity



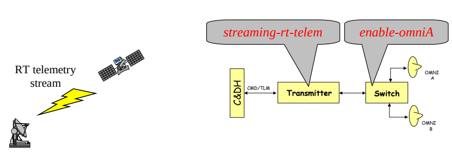
Downlink Data Block Control Program Overview

- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity
- Otherwise, set transmitter to start streaming real-time telemetry so that ground can establish communication link



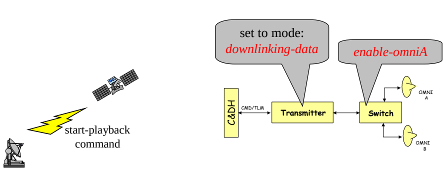
Downlink Data Block Control Program Overview

- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity
- Otherwise, set transmitter to start streaming real-time telemetry so that ground can establish communication link



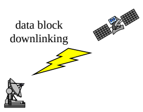
Downlink Data Block Control Program Overview

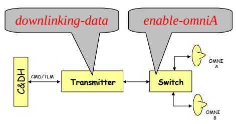
- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity
- Otherwise, set transmitter to start streaming real-time telemetry so that ground can establish communication link
- When start-playback command received from ground, start downlinking data from onboard storage



Downlink Data Block Control Program Overview

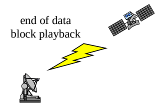
- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity
- Otherwise, set transmitter to start streaming real-time telemetry so that ground can establish communication link
- When start-playback command received from ground, start downlinking data from onboard storage

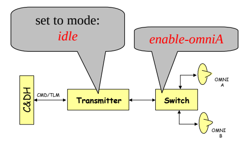




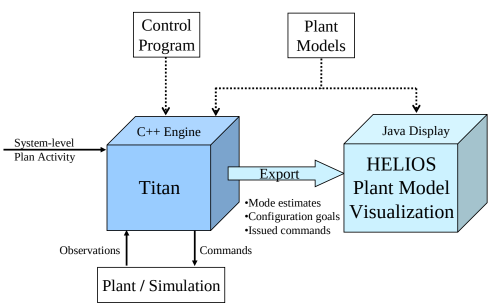
Downlink Data Block Control Program Overview

- Set switch to enable appropriate omnidirectional antenna for downlink of streaming real-time telemetry (based on omni-in-view info from ACS)
- If switch gets stuck in wrong position, fail DownlinkDataBlock activity
- Otherwise, set transmitter to start streaming real-time telemetry so that ground can establish communication link
- When start-playback command received from ground, start downlinking data from onboard storage
- When downlink finished, idle transmitter and report success of DownlinkDataBlock activity

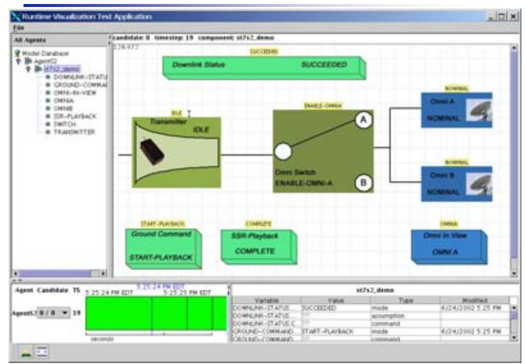




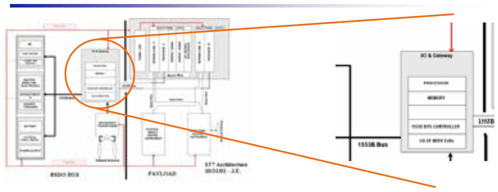
Titan Execution Visualization



Titan Execution Visualization



Scenario 2: Bus Controller Failure



- Bus Controller maintains power to the bus, making power distribution and communications possible.
- Cascading Failure
  - Reset Bus Controller
  - Power Cycle Bus Controller
  - Switch to Redundant Backup Bus Controller

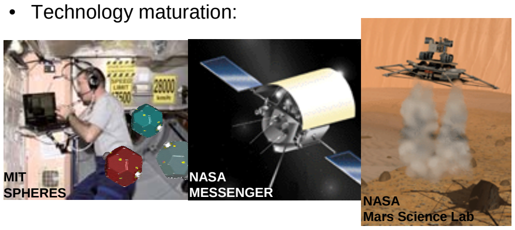
Concept Study Results

- Scenarios address a number of operational use cases
- Highlight desired features for autonomous spacecraft control

Nominal operations:	Fault operations:
• *accept high-level activity goals and decompose into sequence of configuration goal states* • *accept configuration goals and generate sequence of atomic plant commands*	• *diagnose both single and multiple faults* • *manage completely unanticipated faults* • *recovery by repairing faulty components* • *recovery by using physical or functional redundancy*

Conclusions

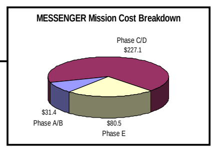
- Model-based execution bridges gap between system-level planning and real-time commanding
- Robustness in sense-decide-act loop
- Cost reduction / Risk reduction / Mission enabling
- Technology maturation:

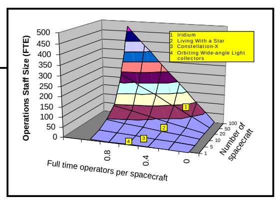


Backup Slides

The Case for Spacecraft Autonomy

COST Reduction

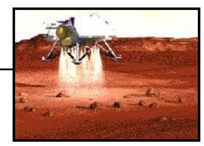


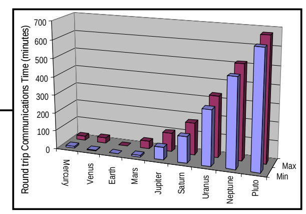


The Case for Spacecraft Autonomy

COST Reduction

RISK Reduction



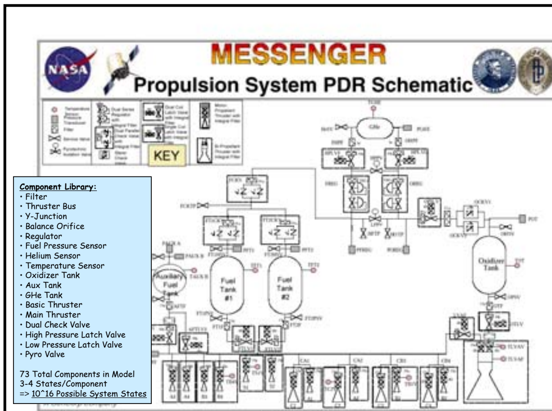


The Case for Spacecraft Autonomy

COST Reduction

RISK Reduction

Mission Enabling



"Autonomy Rules" in Current Application

5.1.2.2.6 If the supply current for OXIDIZER TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF OXIDIZER TANK PRIMARY HEATER.

5.1.2.2.7 If the supply current for HELIUM TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF HELIUM TANK PRIMARY HEATER.

5.1.2.2.8 If the supply current for STAR TRACKER A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER A.

5.1.2.2.9 If the supply current for STAR TRACKER B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER B.

5.1.2.2.10 If the supply current for IMU PPSM-A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-A.

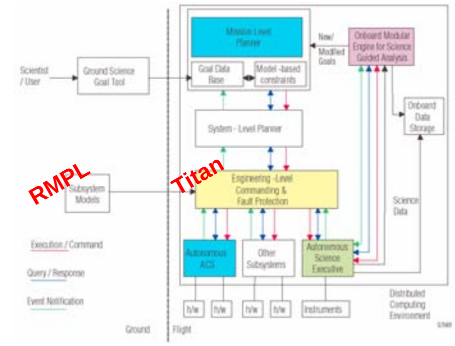
5.1.2.2.11 If the supply current for IMU PPSM-B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-B.

5.1.2.2.12 If the supply current for REACTION WHEEL 1, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF REACTION WHEEL 1. In addition, the corresponding rules that monitor power dissipation of the remaining three reaction wheels shall be disabled. REACTION WHEEL 1 shall be flagged as "unavailable" to the G&C task in the MP.

5.1.2.2.13 If the supply current for REACTION WHEEL 2, multiplie[...] greater than TBD watts, then the FPP shall issue a comma[...] REACTION WHEEL 2. In addition, the corresponding ru[...] dissipation of the remaining three reaction wheels shall be[...] WHEEL 2 shall be flagged as "unavailable" to the G&C t[...]

Example from MESSENGER Safing and Fault Protection Requirements Specification. (Flight Software Design to Support 1280 Rules)

ST7-A Autonomy Software Architecture



Downlink Data Block Control Program

- *Class Definition:*

```
(defclass comm_subsystem ()
   (public state transmitter-state transmitter-mode)
   (public state switch-state switch-mode)
   (public state goal-status downlink-status)
   (public state ground-cmd-status ground-command)
   (public state omni-view-status omni-in-view)
   (public state process-status ssr-playback)
```

- *Method Definition:*

```
(DownlinkDataBlock()
  (do-watching (or (downlink-status = failed) (downlink-status = succeeded))
     (if-thennext-elsenext (omni-in-view = omniA)
      (parallel
         (switch-mode = enable-omniA)
         (when-donext (or (switch-mode = enable-omniA) (switch-mode = stuck-at-A))
            (do-watching (ground-command = start-playback)
               (always (transmitter-mode = streaming-rt-telem))))
         (when-donext (switch-mode = stuck-at-B) (downlink-status = failed))
         (when-donext (ground-command = start-playback)
            (sequence
               (do-watching (ssr-playback = complete)
                  (always (transmitter-mode = downlinking-data)))
               (downlink-status = succeeded)))
      )
     ;; Similarly for the case where (omni-in-view = omniB)
```

Set switch to enable appropriate omnidirectional antenna (based on omni-in-view info from ACS)

Downlink Data Block Control Program

- *Class Definition:*

```
(defclass comm_subsystem ()
   (public state transmitter-state transmitter-mode)
   (public state switch-state switch-mode)
   (public state goal-status downlink-status)
   (public state ground-cmd-status ground-command)
   (public state omni-view-status omni-in-view)
   (public state process-status ssr-playback)
```

- *Method Definition:*

```
(DownlinkDataBlock()
  (do-watching (or (downlink-status = failed) (downlink-status = succeeded))
     (if-thennext-elsenext (omni-in-view = omniA)
      (parallel
         (switch-mode = enable-omniA)
         (when-donext (or (switch-mode = enable-omniA) (switch-mode = stuck-at-A))
            (do-watching (ground-command = start-playback)
               (always (transmitter-mode = streaming-rt-telem))))
         (when-donext (switch-mode = stuck-at-B) (downlink-status = failed))
         (when-donext (ground-command = start-playback)
            (sequence
               (do-watching (ssr-playback = complete)
                  (always (transmitter-mode = downlinking-data)))
               (downlink-status = succeeded)))
      )
     ;; Similarly for the case where (omni-in-view = omniB)
```


Set transmitter to start streaming real-time telemetry

Downlink Data Block Control Program

- *Class Definition:*

```
(defclass comm_subsystem ()
   (public state transmitter-state transmitter-mode)
   (public state switch-state switch-mode)
   (public state goal-status downlink-status)
   (public state ground-cmd-status ground-command)
   (public state omni-view-status omni-in-view)
   (public state process-status ssr-playback)
```

- *Method Definition:*

```
(DownlinkDataBlock()
  (do-watching (or (downlink-status = failed) (downlink-status = succeeded))
     (if-thennext-elsenext (omni-in-view = omniA)
      (parallel
         (switch-mode = enable-omniA)
         (when-donext (or (switch-mode = enable-omniA) (switch-mode = stuck-at-A))
            (do-watching (ground-command = start-playback)
               (always (transmitter-mode = streaming-rt-telem))))
         (when-donext (switch-mode = stuck-at-B) (downlink-status = failed))
         (when-donext (ground-command = start-playback)
            (sequence
               (do-watching (ssr-playback = complete)
                  (always (transmitter-mode = downlinking-data)))
               (downlink-status = succeeded)))
      )
     ;; Similarly for the case where (omni-in-view = omniB)
```


If switch gets stuck in wrong position, fail DownlinkDataBlock activity

Downlink Data Block Control Program

- *Class Definition:*

```
(defclass comm_subsystem ()
   (public state transmitter-state transmitter-mode)
   (public state switch-state switch-mode)
   (public state goal-status downlink-status)
   (public state ground-cmd-status ground-command)
   (public state omni-view-status omni-in-view)
   (public state process-status ssr-playback)
```

- *Method Definition:*

```
(DownlinkDataBlock()
  (do-watching (or (downlink-status = failed) (downlink-status = succeeded))
     (if-thennext-elsenext (omni-in-view = omniA)
      (parallel
         (switch-mode = enable-omniA)
         (when-donext (or (switch-mode = enable-omniA) (switch-mode = stuck-at-A))
            (do-watching (ground-command = start-playback)
               (always (transmitter-mode = streaming-rt-telem))))
         (when-donext (switch-mode = stuck-at-B) (downlink-status = failed))
         (when-donext (ground-command = start-playback)
            (sequence
               (do-watching (ssr-playback = complete)
                  (always (transmitter-mode = downlinking-data)))
               (downlink-status = succeeded)))
      )
     ;; Similarly for the case where (omni-in-view = omniB)
```


When start-playback command received from ground, start playing back data from onboard storage

Downlink Data Block Control Program

- *Class Definition:*

```
(defclass comm_subsystem ()
   (public state transmitter-state transmitter-mode)
   (public state switch-state switch-mode)
   (public state goal-status downlink-status)
   (public state ground-cmd-status ground-command)
   (public state omni-view-status omni-in-view)
   (public state process-status ssr-playback)
```

- *Method Definition:*

```
(DownlinkDataBlock()
  (do-watching (or (downlink-status = failed) (downlink-status = succeeded))
     (if-thennext-elsenext (omni-in-view = omniA)
      (parallel
         (switch-mode = enable-omniA)
         (when-donext (or (switch-mode = enable-omniA) (switch-mode = stuck-at-A))
            (do-watching (ground-command = start-playback)
               (always (transmitter-mode = streaming-rt-telem))))
         (when-donext (switch-mode = stuck-at-B) (downlink-status = failed))
         (when-donext (ground-command = start-playback)
            (sequence
               (do-watching (ssr-playback = complete)
                  (always (transmitter-mode = downlinking-data)))
               (downlink-status = succeeded)))
      )
     ;; Similarly for the case where (omni-in-view = omniB)
```

When playback finished, report success of DownlinkDataBlock activity

Bus Controller Failure Scenario Descriptions

- *Assumptions:*
  - All devices have some feedback allowing detection of anomalous behavior (ex. Report of “no-comm”)
  - Two bus controller devices, BC A & B, where BC B is the backup for BC A.
- *Scenario A*
  - Initial State: BC_A = on, BC_B = off
  - Observe: Comm-status = NO-COMM!
  - Diagnosis: BC_A has a resettable failure
  - Recovery: Issue reset command to BC_A
  - Observe: Comm-status = COMM!
- *Scenario B*
  - Follow on to Scenario A where last observation is:
    - Observe: Comm-status = NO-COMM!
  - Diagnosis: BC_A has a power cycleable failure
  - Recovery: Issue cycle-power command to BC_A
  - Observe: Comm-status = COMM!
- *Scenario C*
  - Follow on to Scenario B where last observation is:
    - Observe: Comm-status = NO-COMM!
  - Diagnosis: BC_A is now broken
  - Recovery: Switch to backup bus controller.
  - Observe: Comm-status = COMM!