Advancing Autonomous Operations for Deep Space Vehicles

Angie T. Haddock [1]
angie.haddock@nasa.gov
NASA, Marshall Space Flight Center, Huntsville, AL 35812, US

Howard K. Stetson [2]
howard.k.stetson@nasa.gov
Teledyne Brown Engineering/NASA, Marshall Space Flight Center, Huntsville, AL 35812, US

Abstract— **Starting in Jan 2012, the Advanced Exploration Systems (AES) Autonomous Mission Operations (AMO) Project began to investigate the ability to create and execute "single button" crew initiated autonomous activities [1]. NASA Marshall Space Flight Center (MSFC) designed and built a fluid transfer hardware test-bed to use as a sub-system target for the investigations of intelligent procedures that would command and control a fluid transfer test-bed, would perform self-monitoring during fluid transfers, detect anomalies and faults, isolate the fault and recover the procedures function that was being executed, all without operator intervention. In addition to the development of intelligent procedures, the team is also exploring various methods for autonomous activity execution where a planned timeline of activities are executed autonomously and also the initial analysis of crew procedure development. This paper will detail the development of intelligent procedures for the NASA MSFC Autonomous Fluid Transfer System (AFTS) as well as the autonomous plan execution capabilities being investigated.**

I. Introduction

Manned deep space missions, with extreme communication delays with Earth based assets, presents significant challenges for what the on-board procedure content will encompass as well as the planned execution of the procedures. Manned deep space missions are envisioned as minimal crew with a currently unspecified amount of crew autonomy due to the light time delays in communication. Questions arise as to the number of crew, crew make-up and skills set, and how to move mission control center based operations to on-board for such a mission. Some of the answers to these questions encompass how autonomous the vehicle is, and the depth and scope of the procedure system placed on-board for crew use.

The AFTS consists of two fluid tanks, one for a source of supply connected to one for multi-use. There are two command-able transfer legs, a command-able return transfer leg, and one manual leg for pure manual operations. Each tank contains a pressure sensor at the bottom of the tank, a temperature sensor and a fluid heater that is interfaced with each tank at approximately the ¾ level up from the bottom. Each command-able transfer leg contains a fluid pump, pressure sensors before and after each pump and a flow meter after the pump. An additional third flow meter is placed just before the Multi-Tank for fault tolerance of either transfer leg. The return leg which transfers fluid from the Multi-Tank to the Supply Tank has a single command-able pump which is used simply to return fluid back to the supply tank. The return leg is semi-automated since there are no flow meters or pressure sensors in use on this transfer leg. This system architecture allows a single fault tolerance as to transfer leg pump failures and flow meter failures. The manual transfer leg then adds an additional fault tolerance when both the primary and backup transfer legs have been failed.

Once the AFTS hardware was built and the Arduino Controller command and telemetry interfaces were developed, an operations concept was needed to determine the requirements for autonomous operations of the system. Today it is common practice for the crew to initiate activities such as fluid transfer, after which they are monitored continuously by the ground. We desired the crew to operate the AFTS with "Single Button" functions in a "fire and forget" fashion. The functions must perform the activity, monitor the system during the activity, detect any failures of the sub-system, isolate failures and then recover the original function that was requested by the crew

[1] Computer Engineer, Space Systems Operations/Mission Operations Lab, NASA, Marshall Space Flight Center.
[2] Computer Scientist, Space Systems Operations/Mission Operations Lab, TBE/NASA, Marshall Space Flight Center.

through to completion. The intelligence of each function or activity was to be embedded within the transfer procedures to be developed. Since this was a fluid transfer sub-system, the functions would involve the transference of fluid in pre-selectable quantities and optional crew input quantities. The operational activities selected were ¼ tank, ½ tank, full tank and crew selectable X number of gallons transfers over the primary, backup and return transfer legs of the AFTS. Since the AFTS contains fluid heaters, a "Set Temperature" function was also envisioned for the development. During nominal space flight operations, flight rules, payload regulations and safety rules are continually monitored during operations; today, these rules are monitored by ground, and potential or actual problems are detected and responded to by the ground. It was desired that an autonomous monitoring system be employed for further enhancement and demonstration of ground operations movement to on-board capabilities. Further safety controls were desired such as "Single Button" AFTS Safing, where only one crew action is required to safe the complete test-bed. With the concept defined, the Software Requirements Specification was developed and work began.

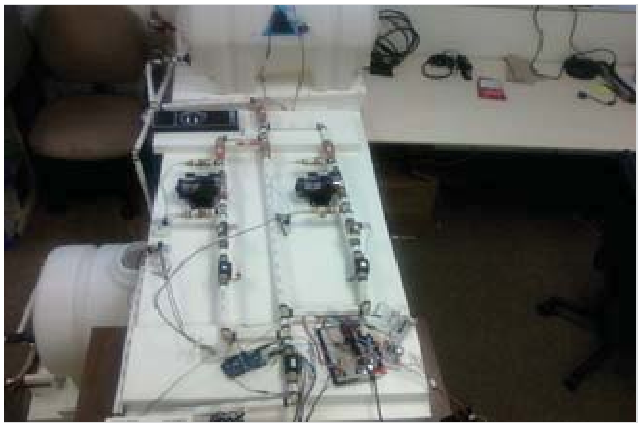


Figure 1: AFTS Test Bed Top View

II. AFTS Procedure System Architecture

The Marshall Space Flight team, with extensive experience with the Timeliner system on-board ISS [2], decided that the Timeliner-TLX automated procedure system met the procedural development requirements of the operations concept. The Timeliner-TLX system is both a procedure development environment and a procedure execution engine. Timeliner-TLX procedures are packaged as a file called a bundle and the procedures within the bundle are independently executed sequences that share global data within the bundle and also global data between all bundles for data exchanges between procedures. Another key feature of the Timeliner-TLX system is the built in sequence status telemetry available to all sequences, allowing a sequence's execution status to be available to all other sequences. A bundle is "installed" into the Timeliner-TLX engine making the sequences available for execution. The HAL9000 Space Operating System[3] is a prototype system that encompasses both procedure execution and the real-time planning and re-planning of procedures. The design of the execution component of the system has been utilized for ISS Payload Operations[1] successfully and could be adapted easily for AFTS. The execution component design, which requires use of Timeliner, includes the "Single Button" functionality requirement, and this design also provides for autonomous monitoring to support Flight Rule, Payload Regulation and Safety rules. In fact, the HAL 9000 Execution Component has Safety designed in as a real-time operator with command capability. The HAL 9000 Timeliner-TLX architecture divides operations into 9 auto-operators (HALMain, GN+C, Power, ECLSS, Comm, Propulsion, Safety, Robotics and Activity), where each auto-operator operates within a separate Timeliner-TLX Engine. Each auto-operator has a suite of bundles that contain the intelligent procedures required to operate the specific sub-system. For the AFTS test bed, only three of the auto-operators were carried over for the division of procedure responsibility, packaged into three bundles. The design then included 3 on-board operator's (Timeliner-TLX Bundles), based upon the HAL 9000 Execution component, HAL Main (the mission manager), Safety, and the Environmental Control and Life Support System (ECLSS) operators. HAL Main would be responsible for the startup and initialization of the AFTS as well as the Safety and ECLSS procedure installation. Safety would monitor all Flight Rules and other condition rules associated with safe operations of the AFTS and would also contain the single crew action system safe capability. ECLSS would own all the intelligent procedures that pertained to fluid transfers. This design allowed the complete AFTS initialization and procedure installation for operations to be activated with a single HAL Main install command. With the division of requirements allocated between the three automated-operators, it was time to develop the detailed procedures themselves.

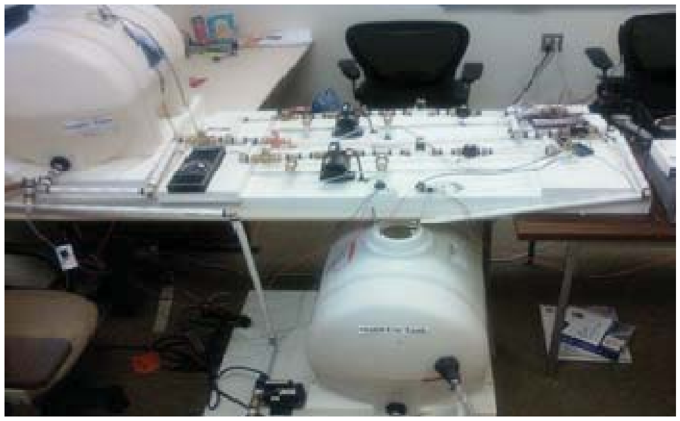


Figure 2: AFTS Test Bed Side View

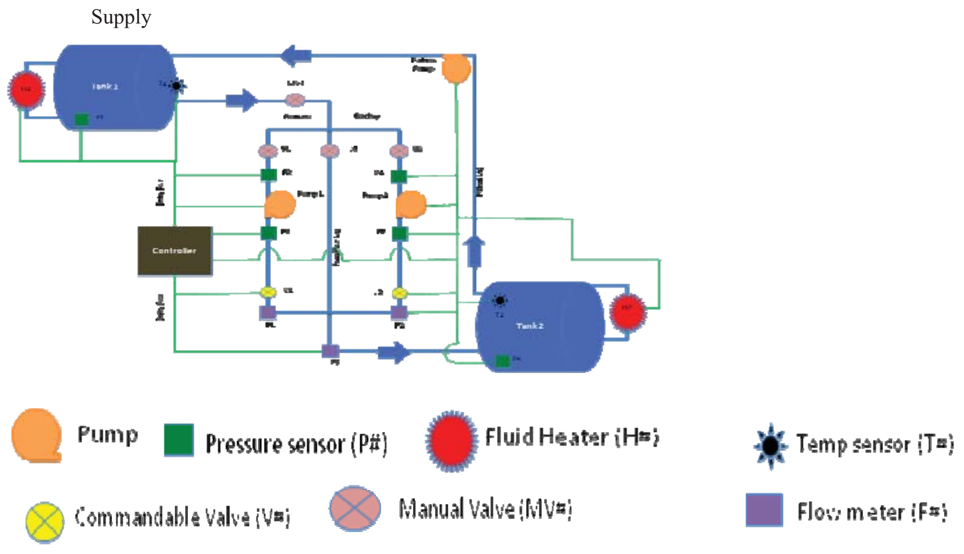


Figure 3: AFTS Context Diagram

III. AFTS Procedure Design

The procedure architecture allocated 3 operators, HALMain, Safety and ECLSS. The HAL Main Timeliner-TLX initialization procedure becomes active automatically upon installation. The first action taken is the Safety procedures are installed and safety monitoring becomes active automatically. The crew is then inquired of the status of the manual valves for confirmation of a "Ready for Operations" state. Once HAL Main determines it is safe to operate the AFTS, it then installs the ECLSS procedures for crew availability. This is inherently safe as the transfer procedures are not available for activation unless the AFTS is deemed operational. Once power is removed from the AFTS, the HAL Main monitoring procedures remove the ECLSS auto-procedures which inhibit inadvertent commanding to the sub-system by procedure execution as they are simply not available to be activated. This autonomous software installation and removal functionality is carried over from the initial ISS Timeliner HAL3 [2] design. It also frees memory allocated to the Timeliner-TLX Engine which executes the procedures, always minimizing the amount of memory needed for procedures based upon sub-systems being powered and un-powered.

The Safety Timeliner-TLX procedures become active automatically upon installation and begin monitoring all safety rules employed. The following rules are enforced:

Maximum Temperature of the Supply Tank Fluid as 75 degrees F
Maximum Temperature of the Multi-Use Tank Fluid as 75 degrees F
Maximum Fluid Level of 27 gallons in the Supply Tank
Maximum Fluid Level of 27 Gallons in the Multi-Use Tank
Supply Tank Heater must be off when fluid level is below the heater interface
Multi-Use Tank Heater must be off when fluid level is below the heater interface
No two pumps in the ON state simultaneously

Fluid temperatures were considered less critical than fluid levels as it pertained to the heater interfaces due to the nature of the fluid heaters (re-circulating water) and maintaining a fluid level where the heaters could actually intake fluid. It was decided that whenever the fluid levels dropped below the heater interfaces, Safety would command the heater off autonomously, demonstrating real-time autonomous Safety implementation. A problem with maintaining fluid temperature arose due to the location of the temperature sensors (bottom of the tank), and the location of the fluid heaters (3/4 full capacity), where only the top layer of fluid was being heated and a non-accurate temperature was obtained. This design has been augmented with fluid recirculation pumps that will pump the cold fluid from the bottom of the tanks to the top of the tanks allowing for some temperature control. An artificial "Bottom" and "Top" of the tanks were specified so that fluid would never go below a pump interface (Bottom), or overflow out the top opening. The artificial bottom and tops are specified in pressure units, the gauge used for fluid transfers.

The ECLSS Timeliner-TLX procedures provide the "Single Button" intelligent crew activities. Each transfer function is required to verify the crew activity before beginning operations. This includes verifying that enough fluid, based upon the specific activity, is available within the supply tank and that enough space is available in the Multi-Use Tank to receive the fluid transfer. Since each activity contains embedded Fault Detection Isolation and Recovery (FDIR), the procedures did not have to determine if the primary and back up transfer legs already contained a fault, such as a failed pump, as during the procedure initialization phase, any failure is detected and automatically fails over to the opposite transfer leg. For example: if the primary pump is failed and the crew requests a ¼ tank transfer over the primary leg, the procedure detects the failed pump and performs the operation on the backup transfer leg automatically. The procedure messages the crew on all failure detections and actions being taken by the procedure and also directs maintenance when failures are encountered. Once the transfer procedure passes initialization (verifies valve and pump successful operations), the procedure monitors the pressures in the tanks for obtaining the desired target pressure and also monitors the specific flow meters during the transfer for detection of pump failures as the valves used on the AFTS fail to an open state. If a failure occurs during the transfer, the transfer procedure commands the active transfer leg to a safe state and verifies the state. Once the active leg that has failed is safed, the transfer procedure activates the opposite transfer leg and completes the remaining fluid transfer quantity of the original transfer amount. During activation of the opposite transfer leg, fault detection again takes place and if a fault is detected, the failover transfer is aborted and the crew messaged for maintenance of both transfer legs. If no failures occur on the opposite transfer leg, once the target fluid pressure is reached, the active transfer leg is safed during which fault detection continues. The list of autonomous activities that the ECLSS operator has available is as follows:

¼ Tank Primary Transfer
¼ Tank Backup Transfer
½ Tank Primary Transfer
½ Tank Backup Transfer
Full Tank Primary Transfer
Full Tank Backup Transfer
X Gallon Primary Transfer
X Gallon Backup Transfer
Supply Tank Heater On
Supply Tank Heater Off
Multi-Use Tank Heater On
Multi-Use Tank Heater Off

Set Supply Tank Temperature
Set Multi-Use Tank Temperature
¼ Tank Return Transfer
½ Tank Return Transfer
Full Tank Return Transfer
X Gallon Return Transfer

Table 1 ECLSS Intelligent Functions

The “Transfer_Remaining” recovery sequence which is initiated by the each activity’s fault detection logic, is not available to the crew directly for stand-alone initiation as this sequence is dependent upon a transfer in progress that has failed. Each single button activity contains the specific FDIR logic required for the fluid transfer being requested. The result is an operations paradigm of an auto-operator that is employed only when needed and when desired, and only a single crew action is required. This capability is a great advancement to flight operations for crew and or ground operator work reduction. Imagine a manual fluid transfer procedure being worked by the crew that requires a command interface and a specific telemetry display where the crew must command the valves and pumps, perform the monitoring of the pressures for completion of the transfer as well as the monitoring of the valves, pumps and flow meters for failures. The crew, upon detection of a failure, would then be required to pull up off nominal operations procedures for the fluid transfer sub-system and command the safing of the transfer leg, monitor the safing, calculate the remaining amount needed to be transferred after the failure, command the recovery via the opposite transfer leg and perform the monitoring of the remaining transfer in progress as well as the final safing of the active transfer leg. This current procedure operations paradigm would require continued crew attention which unnecessarily increases the requirement for crew time and function.

IV. Procedure Format

During development of the ECLSS fluid transfer procedures, the development team noticed a consistent “format” could be employed for each procedure, essentially segmenting the procedure code. If a consistent format could become a standard for wholly encompassed autonomous procedures, could there be a user interface that could be developed to assist the crew for on-board crew authored procedures of this type? Future analysis and investigation in this arena should be conducted for determination of not only how crew authored procedures can be developed, but also how they can be verified, validated and executed during the mission. The architecture of each of the transfer procedures divided the procedure into 7 segments as follows:

Header Segment
Declaration Segment
Validation Segment
Initialization Segment
Monitoring Segment
FDIR Segment 1
FDIR Segment 2

The Header Segment is a commented information section describing the functional capability, authorship and version control information for the source. This segment has always been part of the Timeliner development coding standard used for ISS Timeliner operations [2].

The Declaration Segment defines internal variables used within the procedure such as the result of any calculations from telemetry the procedure will perform. An example from the AFTS system is the “Target Pressure” variable calculated based upon the amount of fluid to be transferred and the current pressure.

The Validation Segment defines the checks to be performed prior to actual command execution of the activity which insures the activity can be accomplished. In the AFTS examples, these checks included verifying the fluid quantities availability and capacity within the tanks.

The Initialization Segment defines the command and command end item checks that are performed to begin the function being requested. For the AFTS, the Initialization segment contained the specific valve open and pump on commands and their associated end-item verification.

The Monitoring Segment defines the parameters to monitor and rate of monitoring while the function is being performed. This segment determines whether the function or activity has completed. This segment also has fault detection implemented once the function has completed and the sub-system is being safed. An AFTS example is where the fluid transfer has completed and the pump is commanded off and the valve commanded close and anomalies occur during this safing.

The FDIR1 Segment handles faults that occur during the procedure execution and is responsible for the actions that take place for safing and recovery. This monitoring is in effect after initialization has been successful and fluid is flowing. For the AFTS, this segment monitored the flow meters during the fluid transfers. If a fault occurred, this segment performed the safing of the transfer leg, the safe verification and also the start of the recovery procedure. A common recovery procedure (Transfer_Remaining), was developed that would recover any transfer and it too contained fault detection logic.

The FDIR2 Segment handles faults that occur during the procedures initialization. The main purpose here is to detect a fault when the function is being initialized and not yet in progress. The common recovery procedure would be started at this point. Further analysis of this code structure and a corresponding graphical user interface for code segment development by crew members could determine the feasibility of on-orbit crew developed intelligent procedures. Although there may be pro's and con's for and against the crew developing intelligent procedures, it remains a potential risk mitigation solution for permanent loss of communications and increases assured crew return.

V. Planning Autonomous Activities

The ECLSS fluid transfer procedures actually become wholly contained activities and as such, the activity duration and resource utilizations can be quantified upon the first execution during testing and hence can now also be easily planned. From the planning perspective, the results of a test execution can derive the time it required to transfer a fixed quantity, we know the quantity of the fluid resource desired, and we know the power requirements needed for the pump and for the activities duration. As these type activities are performed, resource utilizations can be updated for each such as the instances where a degraded pump performance is encountered, affecting the activity duration time and power requirements. The instance where a procedure detects a failure and fails over to the alternate transfer leg on the AFTS, becomes an autonomous unplanned resource utilization unless the Transfer_Remaining recovery activity is planned for the same time period. This then becomes an "overhead" booking of resources with the potential of not utilizing all that is planned for the activities, but does ensure the plan can be executed autonomously even when failures occur. The delta time that is encountered upon a failover becomes the time it takes to send the required safing commands and the time it takes to send the initialization commands in the recovery procedure, which is a small delta that can be included in an activities "slip" time.

VI. Autonomous Plan Execution

The AFTS intelligent procedures are activities that execute separately and require fairly fixed resources for both nominal and off nominal execution, which made generation of a plan of transfer activities relatively simple. The Marshall team decided to investigate how a plan of fluid transfers could be executed autonomously. Two concepts, the "Master Bundle" concept from early ISS payload operations capability and the HAL 9000 System "Auto-Mode" [3][4] execution concept were selected as potential candidates. The Master Bundle concept required an initial "On-Board" plan of activity records where Timeliner auto-procedures had a unique flag in the record indicating an auto-procedure. Software would scan the plan and produce Timeliner bundle/sequence source code that contained Install Bundle, Start Sequence, HALT Bundle and Remove Bundle commands based upon the planned activity start time and activity durations. These were absolute times and unconditional.

The HAL 9000 System has two different ways of executing activities autonomously. There is the plan of activities much the same as the ISS Onboard Short Term Plan (OSTP), but where all the activities are wholly contained intelligent procedures, and where the Planning Engine[4] would start each activity, and then there is the "Full Auto" mode design where the plan is contained in time ordered state code arrays for every device in the activity.

An AFTS Master Bundle was created manually, with two differences from the ISS payload concept. The AFTS Master Bundle was based upon relative time from the start of execution rather than absolute time and would also insure no two activities were in execution simultaneously. The master sequence first retrieves the current time and then starts each activity relative to the current time that was collected. The start of an activity is conditional as the master sequence ensures the previous activity has been completed. Each AFTS activity was given an activity duration of 15 minutes and 12 separate activities were planned. An example of the AFTS Master Bundle entry is as follows:

```
When Time >= Plan_Start+2:00 and ActInProgress=False
   Start ECLSS.HalfTankBackupTransfer
   When ECLSS.HalfTankBackupTransfer.SEQSTAT =
    SEQ_ACTIVE
       Set ActInProgress = True
    End When
 End When
```

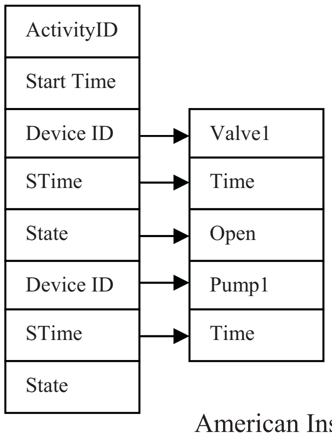
The master sequence continuously verifies the sequence's status to determine whether the activity has completed nominally or off nominally as follows:

```
When ECLSS.HalfTankBackupTransfer.SEQSTAT =
     SEQ_FINISHED Or
         ECLSS.HalfTankBackupTransfer.SEQSTAT =
            SEQ_STOPPEDBYERROR  Or
         ECLSS.HalfTankBackupTransfer.SEQSTAT =
            SEQ_STOPPEDBYCMD Then
        Set ActInProgress = False
End When
```

The Timeliner-TLX system allows any sequence the ability to monitor the status of any other sequence that is installed and is a built in feature of status variables that are predefined. Multiple plan executions were accomplished with the AFTS Master Bundle with failure injection during transfer operations and failover activations and the sequence FDIR did what it was programmed to do. After the first few runs, it became apparent that these type operations could be background activities and that the crew would not be required unless a fault occurred during safing operations that required manual intervention. We could force the plan to be out of synchronization with fluid quantities not being available, but the activities verified these and simply informed the crew and continued with the next activity when it was time.

VII. Alternate Way for Autonomous Execution

The HAL 9000 System planning engine design[4] plans the intelligent procedures, produces a time-ordered linked list activity plan for the specific sub-system and also produces data arrays for each device and state of the device within the planning engines sub-system. These "state code" arrays contain the time stamped state of devices corresponding to the execution of the intelligent activities that is planned for the sub-system. The HAL 9000 Execution Component Plan Monitor for the sub-system utilizes these states and time to determine if the real-time execution of activities is meeting the plan. The HAL 9000 Plan Monitor has the additional capability of executing the time tagged state codes in time order which is referred to as "Full Auto" mode. For the AFTS, the state code arrays for a ½ Tank Primary Fluid Transfer format and example content would be as follows:



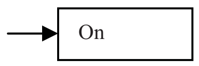


Figure 4: State Code Array Format

We wanted to compare the different methods of execution although the HAL 9000 “Full Auto” mode is dependent upon the Fault Detection, Diagnostics and Response (FDDR) Monitor for notification of failures, we could re-create the state code arrays by using a Timeliner-TLX sequence to load memory before starting the Plan Monitor. The Plan Monitor was then programmed to query the crew for “Full Auto” mode or “Monitor Only” mode. The state code array example above opens VALVE1 at the specified time before PUMP1 is turned on at its specified time. The issue in this execution is the timing during anomalies such as an AFTS failover from the primary transfer leg to the backup leg. To accommodate the pre-planned failover that may or may not be executed, the failover activity is also scheduled as a successor activity and contains a duplicate negative activity id denoting contingency. The full scheduling of the state code arrays then appears as follows:

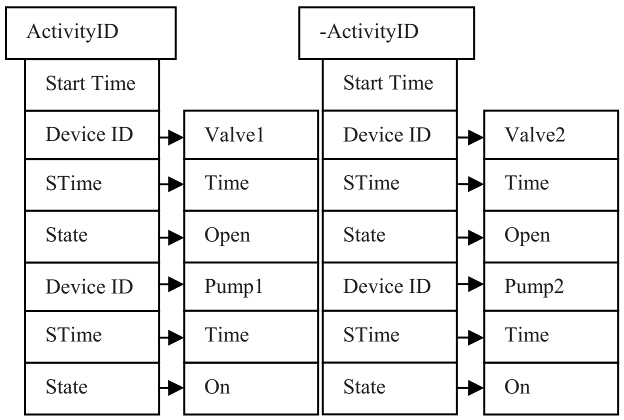


Figure 5: Contingency Plan Sate Code’s

The negative activity identifier correlates the contingency activity where Valve 2 and Pump 2 are commanded on. There is additional information in the state code arrays not covered here such as slip time and number of states as two examples not shown for simplicity and space savings.

The state code arrays are used to determine whether a specific activity is meeting the plan when the HAL 9000 Plan Monitor [3] sequence is not in the “Full Auto” mode. The Plan Monitor continuously verifies the state of each device in the plan and the time the device is supposed to be in this state while in Monitor Only mode. The HAL 9000 Plan Monitor for the specific sub-system is dependent upon the FDDR Monitor [3][4] to generate a fault notification during execution of the nominal state code array for determination of contingency activation. The contingency activity id for the HAL 9000 system is simply a duplicate activity id that is negative in value which results in one activity with one contingency. The HAL 9000 Plan Monitor for the specific sub-system, upon notification of the failure from the FDDR Monitor while in “Full Auto” mode, inserts the times into the contingency state code arrays based upon the current time and the duration of the predecessor activity’s execution derived from the current time – the start time. The HAL 9000 FDDR Monitor specific to the sub-system performs the safing of the primary leg in this example, allowing the Plan Monitor to simply execute the contingency state codes. It should also be noted that no re-plan was required as the contingency was already built into the plan. The differences in execution between wholly contained intelligent Timeliner-TLX procedures and the HAL 9000 Execution Component “Full Auto” mode lies within the embedded FDIR employed within the intelligent procedures, versus a separate monitor that only knows whether a device or sub-system has failed. The intelligent procedure has the specifics to the actual safing that is required at any specific point in the procedure, while the FDDR Monitor has the generic safing capability and will need to check all devices for a safe state and command the ones that are needed. Timeliner-TLX provides the identification and status of all installed procedures that are in execution as a built-in feature of this procedure system, which allows the FDDR Monitor more intelligence as it pertains to the higher level function that was being performed at the point of the failure. Essentially the FDDR Monitor can detect whether a Timeliner-TLX sequence is stopped by an error, stopped by command from the crew or in a finished condition. For the AFTS, no HAL9000 FDDR Monitor was employed and only Plan Monitor functionality developed for autonomous plan execution due to time constraints. In the HAL 9000 system, full-auto mode is selected at the planning engine interface versus the execution component such as in the AFTS testing. In this way the real-time planning engine has knowledge that an activity would be performed in this mode and hence knows the function that was to be performed and can perform the re-plan if an anomaly occurs.

VIII. Execution Comparison

After plan execution with both methods, the Master Bundle concept appears to be the better operational implementation for the AFTS due to the lack of a full HAL 9000 suite of planning engines and FDDR monitors, but is certainly dependent upon having intelligent procedures as failure without recovery may impact the activities that follow in the plan. An AFTS example of this condition is when a pump failure is induced, without recovery capability, the safing would occur such as the valve commanded close and power off of the pump but since the full transfer amount did not occur, there may not be sufficient fluid or fluid capacity to perform the next activity in the plan. Intelligent procedures with full embedded FDIR increases the success percentage of Master Bundle execution as each activity eventually comes to completion and the system is in a safe state for the next activity.

The Full Auto Mode execution does not have procedures in operation, does not have knowledge of the higher level function being performed, and does not know the quantity of fluid being transferred. Essentially, the same operational effectiveness can be achieved via time-tagged command queue's, where commands are stored in a buffer on-board and issued when the time tagged to the command is encountered. Command chaining is another example of how Full Auto Mode can be useful as each device entry that has a duplicate time would be commanded in the order they are encountered in the chain. A key feature of the Plan Monitor executing the state code arrays is that each command is verified before the next command entry is performed where command queue's and command chains generally do not perform this verification. If a full HAL 9000 system had been employed, the FDDR monitor would have safed the system and then notified the Plan Monitor of the failure. The Planning engine for the specific system would then start the contingency activity and perform any real-time re-planning that was required. Since the HAL 9000 System would automatically re-plan upon notification of the failure, and a new updated plan produced, the distribution of actions would result in the same execution as the intelligent activity with the exception that a new plan would have been produced.

The HAL 9000 combination of intelligent procedures along with the corresponding state code data for the plan allows both the autonomous execution of the plan of activities and the autonomous monitoring of the plan when the Plan Monitor is in Monitor Only mode. An autonomous operation requires monitoring and the capability to track real-time execution with planned events programmatically, results in less crew dependency and faster notification when the plan is not met. In summary, intelligent procedures can reduce the amount of external monitoring and planning that is required since these functions are wholly encompassed within the procedure.

IX. Future Work

An increase in complexity of the AFTS system is desired to fully exercise and expand on the distributed operations realized by the procedure architecture. Additional pressure sensors are to be added to the tanks to increase fault tolerance and logic, and additional recirculation pumps and filters will be added that will not only increase the operations requirements but also make temperature control available. The filter recirculation will be controlled via 3-way valves to allow filter / non-filter recirculation. Adding complexity and additional flight rules will help further prove the procedure architecture and its understanding of how adaptable it may be for multiple systems to be operated. As it pertains to intelligent procedures, the AMO team will be developing these type procedures to operate an EXPRESS Payload rack on-board the ISS in the near future. The procedures will be proved on the ground before interfacing them on-board for eventual crew usage. The intent of the EXPRESS operations is to provide single action functions for operating the rack to include power, thermal control, smoke detection and payload configuration, with embedded FDIR wherever possible. An interesting consideration as we begin this development is whether a system that was built with minimal intent of automation can be fully operated autonomously. This we shall see.

Appendix A
Acronym List

AES	Advanced Exploration Systems
AFTS	Autonomous Fluid Transfer System
AMO	Autonomous Mission Operations
COMM	Communications
ECLSS	Environmental Control and Life Support System
EXPRESS	EXpedite the PRocessing of Experiments for Space Station Racks
FDDR	Fault Detection, Diagnostics and Response
FDIR	Fault Detection Isolation and Recovery
GNC	Guidance , Navigation and Control
HAL	Higher Active Logic
ISS	International Space Station
MOL	Mission Operations Lab
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
OSTP	Onboard Short Term Plan
TBE	Teledyne Brown Engineering

Appendix B
Glossary

Autonomous Fluid Transfer System	A dual tank, computer controlled Test-Bed which mimics either a habitat water or a simplistic cryogenic fluid system.
HAL 9000 Space Operating System	HAL 9000 Space Operating System is a crew-integrated, autonomous command and control system designed specifically for fully automated, long-duration deep space vehicles.
Timeliner-TLX System	The Timeliner scripting language for expressing operational procedures. The TLX integration platform is used for developing and executing Timeliner applications. Developed and maintained by Draper Laboratory. Timeliner has been used in Space Shuttle simulation since 1982, on Space Shuttle since 1991, and on ISS since 1994; Timeliner-TLX was commercialized in 1997.

Acknowledgments

The authors would like to acknowledge Craig Cruzen, Mission Operations Laboratory, NASA Marshall Space Flight Center, Scott Akridge, Space Systems Department, NASA Marshall Space Flight Center, George Plattsmier, Space Systems Department, NASA Marshall Space Flight Center Shen Chang, Spacecraft & Vehicle Systems Department, NASA Marshall Space Flight Center, and Dr. Jeremy Frank, NASA Ames Research Center, for their support with the initial groundwork, development and testing of "single button" crew initiated autonomous activities using Timeliner-TLX in the Autonomous Fluid Transfer System.

References

[1] (Haddock et al., 2012) Haddock, A T., Stetson, H K., "Automated Operations Development for Advanced Exploration Systems", 2012 Space Operations Conference, Stockholm Sweden, June 2012

[2] (Stetson et al., 2007) Stetson, H K., Deitsch, DK., Cruzen, CA. , and Haddock, AT., "Autonomous Payload Operations On-Board the International Space Station", IEEE Aerospace Conference, paper #1066 Jan 8 2007

[3] (Stetson et al., 2011) Stetson, H K., Knickerbocker, GK., Cruzen, CA. , and Haddock, AT., "The HAL 9000 Space Operating System", IEEE Aerospace Conference, paper #1027 March 2011

[4] (Stetson et al., 2012) Stetson, H K., Watson, MD., and Shaughnessy, R., "The HAL 9000 Space Operating System Real-Time Planning Engine Design and Operations Requirements", 2012 Space Operations Conference, Stockholm Sweden, June 2011

Biography

Angie T. Haddock *is employed by NASA's Marshall Space Flight Center (MSFC) in Huntsville, Alabama, in the Mission Operations Lab. Currently, Ms. Haddock is the Lead of the MSFC Advanced Exploration Systems-Autonomous Mission Operations (AES-AMO). Prior to the AES projects, she was an analyst for the Space Launch Systems (SLS) Flight Operations. In the SLS position, she worked with the analysis and development of the SLS Flight Operations Specification and supported the analysis and documentation of requirements and Launch Commit Criteria derived from the vehicle system. Preceding the SLS and the AES projects, Ms. Haddock was the Operations Lead for the ISS PLMDM. In this role, she was responsible for leading the development of PLMDM operations, technical coordination for the development and update of PLMDM software. Prior to joining NASA in 2000, she worked for Teledyne Brown Engineering, where she trained to serve as a Command and Payload MDM Officer (CPO) for the ISS Payload Operations Integration Center. She holds a Bachelor of Science degree in Computer Science from Athens State University. Ms. Haddock, and her husband, Stacey, have three daughters; Stephanie, Mary Elizabeth & Amy.*

Howard K. Stetson *is a contractor for Marshall Space Flight Center, Space Systems Operations and is currently working as an analyst for the Advanced Exploration Systems-Autonomous Mission Operations project and has over 34 years of experience in software development and engineering, encompassing numeric intensive computing, parallel processing, computer graphics, simulation and modeling, computational fluid dynamics, real-time C&C operations, operations automation, and software integration and test. Preceding the AES projects, Mr. Stetson designed, developed and implemented the Higher Active Logic (HAL) autonomous system for ISS payloads. Mr. Stetson, an employee of Teledyne Brown Engineering, has produced three white papers for the Marshall Space Flight Center, the Higher Active Logic System (HAL) for the ISS payload operations computer system, the Automated Multi-Purpose Space Operating System (AMPSOS) and the HAL 9000 Space Operating System design. Mr. Stetson is also a member of and instructor for the United States Parachute Association and has over 3200 jumps to date.*