SpaceOps-2025, ID #574

AIM Long-Term Operations: Commanding a Deaf Spacecraft

Authors: Stephanie Ruswick – Laboratory for Atmospheric and Space Physics, University of Colorado, stephanie.ruswick@lasp.coloardo.edu

David Welch – Laboratory for Atmospheric and Space Physics, University of Colorado, dave.welch@lasp.coloardo.edu

Sean Ryan – Laboratory for Atmospheric and Space Physics, University of Colorado, sean.ryan@lasp.coloardo.edu

Abstract

The Aeronomy of Ice in the Mesosphere (AIM) was a NASA Small Mission Explorer that carried three instruments designed to study noctilucent clouds. It launched on April 25, 2007 and successfully operated until March 2023. Nine days after launch, the mission operations center (MOC), located at the University of Colorado's Laboratory for Atmospheric and Space Physics, started having difficulty achieving lock on the uplink sub-carrier. They were unable to consistently achieve bitlock and would often go for extended periods of time without being able to command the spacecraft. The program made heroic efforts to fix the uplink receiver but, unfortunately, they were unable to resolve the issue. Bitlock remained intermittent for the rest of the mission, with the longest outage being 1,684 days (July 2012-March 2017).

In the first year after launch, the flight operations team (FOT) were able to develop and implement spacecraft autonomy that allowed the AIM mission to continue and complete its science objectives. The team also developed an alternate way to command the spacecraft without sub-carrier uplink. This alternate commanding is the focus of this paper.

The team created a process that involved modulating the radio frequency (RF) signal and using flight software (FSW) to detect the changes in RF signal. The FSW would look for specific changes in order to trigger additional on-board autonomy. This allowed the operators to execute pre-loaded command sequences. These sequences were primarily used for recovery operations but were also used to adjust nominal spacecraft operations (ex: battery charge control). There was also an option to use that same technique to load and execute a unique relative time sequence (RTS) generated on the ground. This gave the FOT the ability to send commands and perform activities that were not part of any of the pre-loaded sequences.

For nearly five years, the FOT had to rely exclusively on this alternate commanding to successfully fly AIM. With careful planning, the FOT was able execute complicated activities such as managing the battery, mitigating hardware anomalies, and preparing for full-sun operations where AIM did not see eclipse for several months. This paper will go into further detail on how the alternate commanding worked, its challenges, and how the FOT was able to utilize this unique commanding to operate the spacecraft for nearly 16 years.

1. INTRODUCTION

The Aeronomy of Ice in the Mesosphere (AIM) spacecraft was a NASA Small Explorer mission dedicated to the study of Polar Mesospheric Clouds (PMCs), also called noctilucent clouds. It launched on April 25, 2007 on a Pegasus XL rocket from Vandenberg Air Force Base in California, USA. AIM was built by Orbital Sciences Corporation (now part of Northrop Grumman) in Dulles, Virginia and operated by the University of Colorado's Laboratory for Atmospheric and Space Physics (LASP) in Boulder, Colorado. The mission worked in collaboration with multiple organizations including, Hampton University (HU), Virginia Polytechnic Institute and State University (VT), the Space Dynamics Laboratory (SDL) at Utah State University (USU), G and A Technical Services (GATS), the Naval Research Laboratory (NRL), George Mason University (GMU), and Norwegian University of Science and Technology (NTNU). The mission lasted for nearly 16 years before the battery degraded and mission and science operations were no longer possible. AIM re-entered Earth's atmosphere the morning of August 19, 2024.

Three instruments were flown on AIM and each were designed to gather data pertaining to PMCs. The Solar Occultation for Ice Experiment (SOFIE) instrument took solar occultation measurements at sunrise and sunset that were used to characterize the cloud properties and the PMC environment. The Cloud Imaging and Particle Size (CIPS) experiment was a wide-angle imager that consisted of four identical cameras that took images of the PMC backscatter radiance in order to study cloud morphology and particle size. Finally, the Cosmic Dust Experiment (CDE) was used to monitor the variability of cosmic dust influx into Earth's mesosphere. Both CIPS and SOFIE took data until the end of the mission. CDE was turned off in July 2012 after an anomaly severely degraded the data quality.

AIM was placed into a sun-synchronous, polar orbit at 600km altitude. During the first two years of the prime mission, the beta angle was 0 +/- 9 degrees, meaning that it was closely aligned with the Sun. However, as the mission continued, the spacecraft operated at all beta angles. This resulted in two full-sun periods where AIM did not see any Earth eclipses.

Initial mission operations post-launch were nominal. The Mission Operations Center (MOC) was able to establish good downlink and uplink with the AIM spacecraft. The spacecraft and SOFIE commissioning were completed on schedule. However, nine days after launch, the AIM receiver began to have issues locking onto the command uplink subcarrier modulation and the MOC was unable to reliably send commands to the spacecraft. The MOC, along with members from the spacecraft engineering and science teams, worked diligently to resolve the problem. Numerous tests were conducted in attempt to troubleshoot the intermittent bitlock and better characterize the conditions for when the transceiver could and could not lock onto the subcarrier modulation. Unfortunately, no solution was found and the transceiver continued to struggle and command outages lasted from hours to days. The Flight Operations Team (FOT) moved quickly to overhaul the operations concept and create a new one centered around a fully autonomous spacecraft. CIPS and CDE commissioning, data downlinks, orbital knowledge, basic orbit slews, and science observations were all automated. Additional details on the initial debugging actions regarding the transceiver anomaly and development of AIM autonomy can be found in AIM Autonomy

Development – Long Term Care for a Deaf Spacecraft SpaceOps paper by Debra McCabe, Sean Ryan, David Welch, and John Fulmer, which was presented at SpaceOps 2008.

Despite their heroic efforts, the FOT was unable to determine the cause of the bitlock anomaly and fix the transceiver. Bitlock remained intermittent for the next five years until July 29, 2012. After that there was no bitlock for 1684 days. Figure 1 shows the percentage of time bitlock was achieved on a daily basis over the course AIM's mission life.

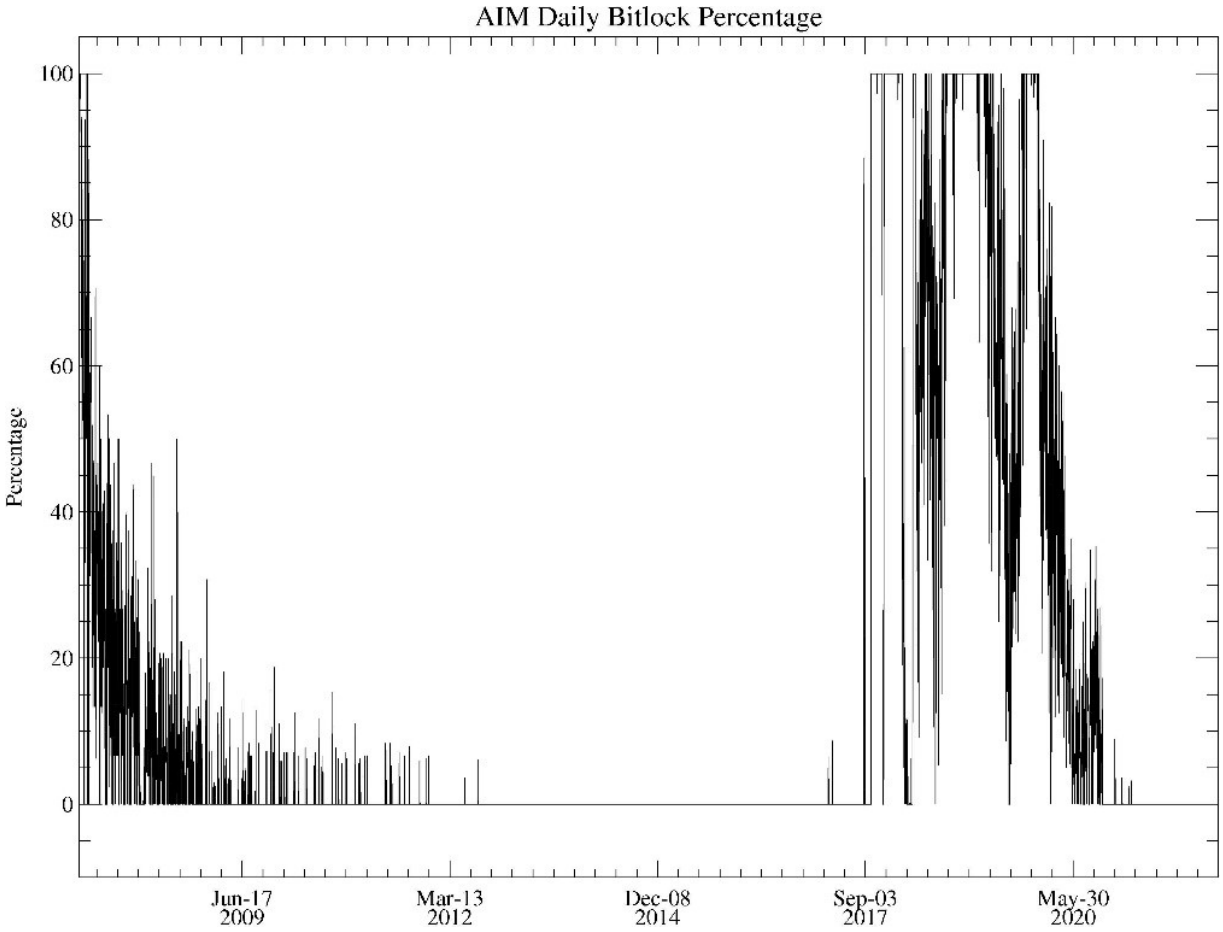

Figure 1 AIM Daily Bitlock Percentage

On March 5, 2017, the beta angle dropped below -67.5 degrees and AIM entered its first full-sun period (see Figure 2 below). Seventeen days later, on March 22, AIM unexpectedly had its first bitlock in over 4.5 years. As the full-sun season progressed, more was learned about how temperature affected bitlock. This knowledge allowed the MOC to make adjustments and resume nominal operations similar to the beginning of the mission. Some of the autonomy, such as automatic data downlinks, remained in place, but now absolute time sequences could be used to load state vectors to maintain normal pointing and the FOT did not have to adjust the on-board orbit determination and pointing autonomy. The first full-sun period lasted until October 30, 2017 and a second full-sun period occurred from February 15 to September 19, 2018.

After the second full-sun period, bitlock went away and the MOC did not see any more for the rest of the mission. A third full-sun period was predicted for the fall of 2024 but the battery failed and the mission was ended before AIM saw this period.

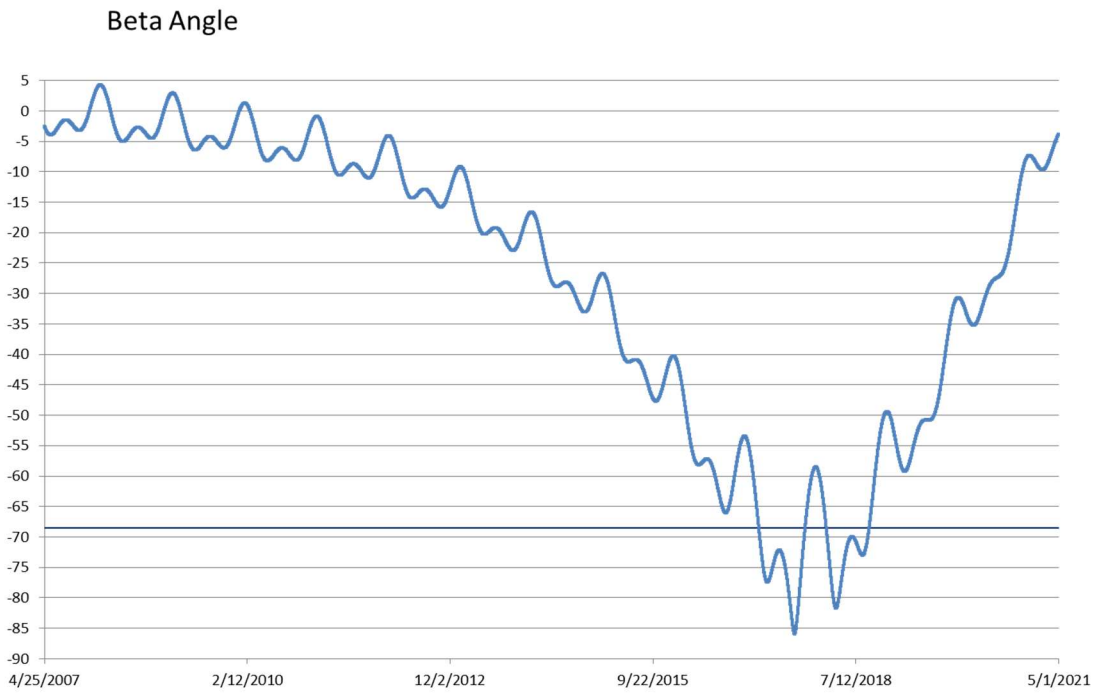

Figure 2 Beta Angle (April 25, 2007 - May 1, 2021)

2. MORSE CODE

While the transceiver struggled and often failed to lock onto the uplink modulation, it did still register uplink. The FOT took advantage of this and developed a process that involved modulating the radio frequency (RF) signal and using flight software (FSW) to detect the changes in RF signal. The FSW would record the specific high-low pattern where a nominal value would represent a 0 and a high value would represent a 1. The autonomy would trigger once a specific pattern was detected and execute a pre-defined sequence. The FOT called this new feature Morse Code (MC).

In order to register a nominal or high value, the FSW monitored the signal in 20 second intervals. It then calculated the average signal over that 20 seconds and compared it to the average of the previous 20 seconds. If the average was higher than the previous 20-second average, the FSW registered a 1 in Morse Code. If the average was lower, then it registered a 0. If there was no change or the change was small, the FSW registered whatever the previous Morse Code value was (i.e. if a 1 was previously registered, then the next Morse Code number would remain 1, and if it was previously 0, it would stay 0). The FSW continuously monitored and calculated the average. On-board autonomy was triggered when FSW saw specific 16-bit (binary) patterns. 16 was chosen to prevent inadvertently starting a Morse Code operation. Figure 3 is an example of the signal seen by the spacecraft during a Tracking and Data Relay Satellite (TDRS) contact when Morse Code

operations were being performed. While there is some fluctuation, there are clear periods of “high” and “low”. These are translated into ones and zeros, accordingly.

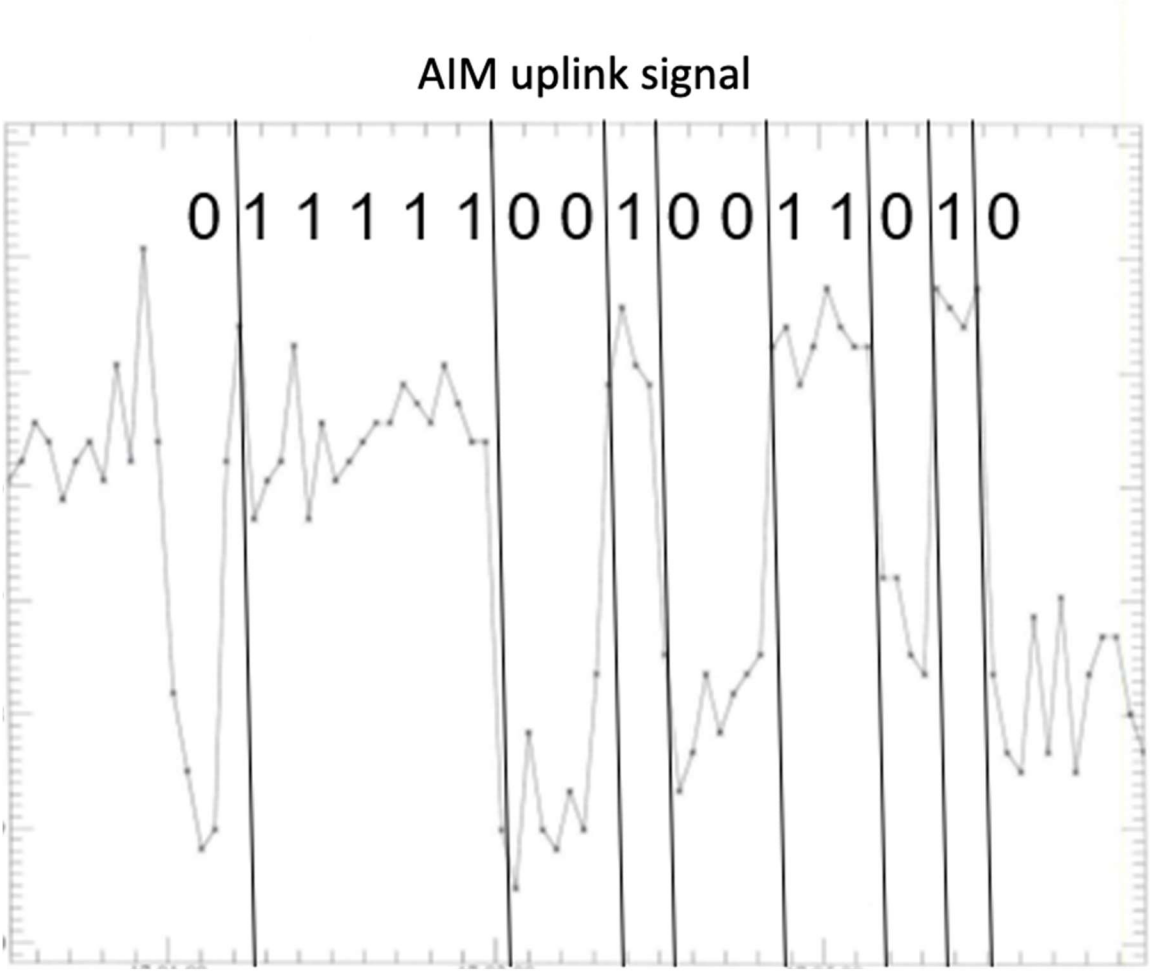

Figure 3 Plot of Uplink Signal Translated to Morse Code

It took 5 minutes and 20 seconds to send a 16-bit Morse Code pattern. The FOT created scripts to automatically change the signal at the appropriate intervals.

Morse Code operations were separated into two groups: Manual Morse Coding and Scratchpad Loads. Manual Morse Coding allowed the user to select and execute a pre-canned RTS (called a Library RTS). Scratchpad activities involved writing to a specific section of memory to build a custom RTS that could then be copied to a Library RTS slot and executed.

3. MANUAL MORSE CODING

Manual Morse Coding was the fastest and easiest way for the MOC to execute spacecraft commands from the ground. However, starting a Library RTS via manual MC still took three steps plus an optional fourth, clean-up step and each step required a 16-bit MC command. This flow is outlined in Figure 4. Morse Code operations must first be initialized/enabled. Once enabled, the

MOC could select and execute the desired Library RTS. Finally, there was the Stop and Reset step that would stop the Library RTS (if it was still executing) and reset MC on-board autonomy. This clean-up step needed to be completed before the MOC could execute another Library RTS. Each step must be completed in order. The MOC was not able to select a Library RTS until MC was enabled nor could a Library RTS be started without first selecting one. While cumbersome, this process greatly reduced the risk of inadvertently selecting and executing an RTS due to nominal and un-commanded TDRS signal fluctuations.

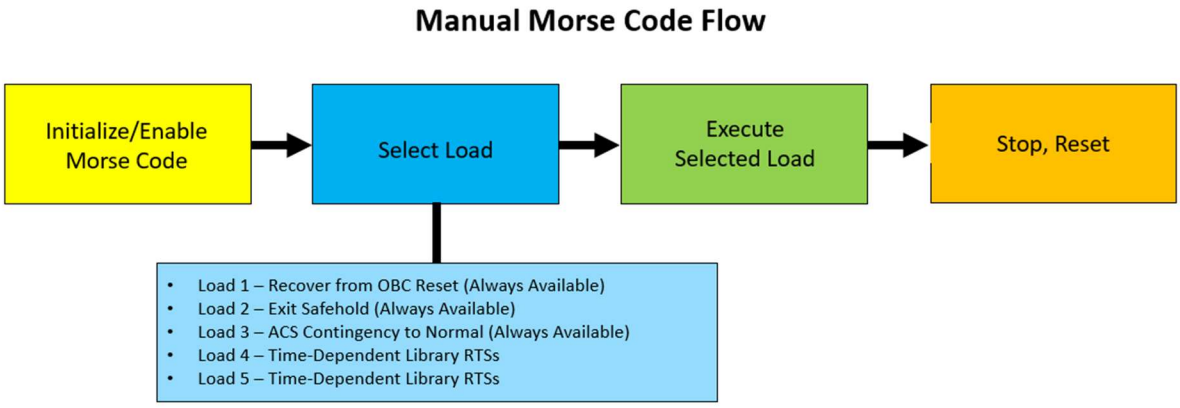

Figure 4 Manual Morse Code Flow

The MC initialize command informed the FSW that MC operations were now underway and to look for the MC select command. MC Initialize was always the same regardless of which Library RTS the MOC planned to execute.

Once initialized, the MOC could select one of the 37 available sequences. Figure 5 is an example list of the Library sequences available. The list changed several times over the course of the mission as new Library RTSs were needed and old ones were no longer applicable to current operations.

Three of the sequences (Loads 1-3) were always available for execution: Recover from On-Board Computer (OBC) Reset, Exit Safehold, and command the Attitude Control System (ACS) from Contingency to Normal Mode. These RTSs were for critical recovery activities that the FOT wanted to be able to execute as quickly as possible.

The other 34 library RTSs were separated into two lists that were time dependent. Once MC was enabled, the FSW moved through these lists in parallel, making one RTS from each list available for execution at a time. Every three hours, the FSW moved down the lists and changed which Library RTSs were available. For example, at the start of MC operations, Enter Safehold and Reset APE were available in loads 4 and 5. After three hours, those RTSs were rotated out and Change Trickle Charge Setting and Zero Scratchpad Buffer became available. This continued until FSW reached the end of the list. The last time-dependent RTSs were available for three hours and once that three hours expired, MC operations were stopped and reset. This meant that the FOT had a 3-hour window to select each time dependent sequence. If the select command was unsuccessful, MC

operations would need to be reset and restarted and the MOC would have to wait for the time-dependent Library RTS to be available again.

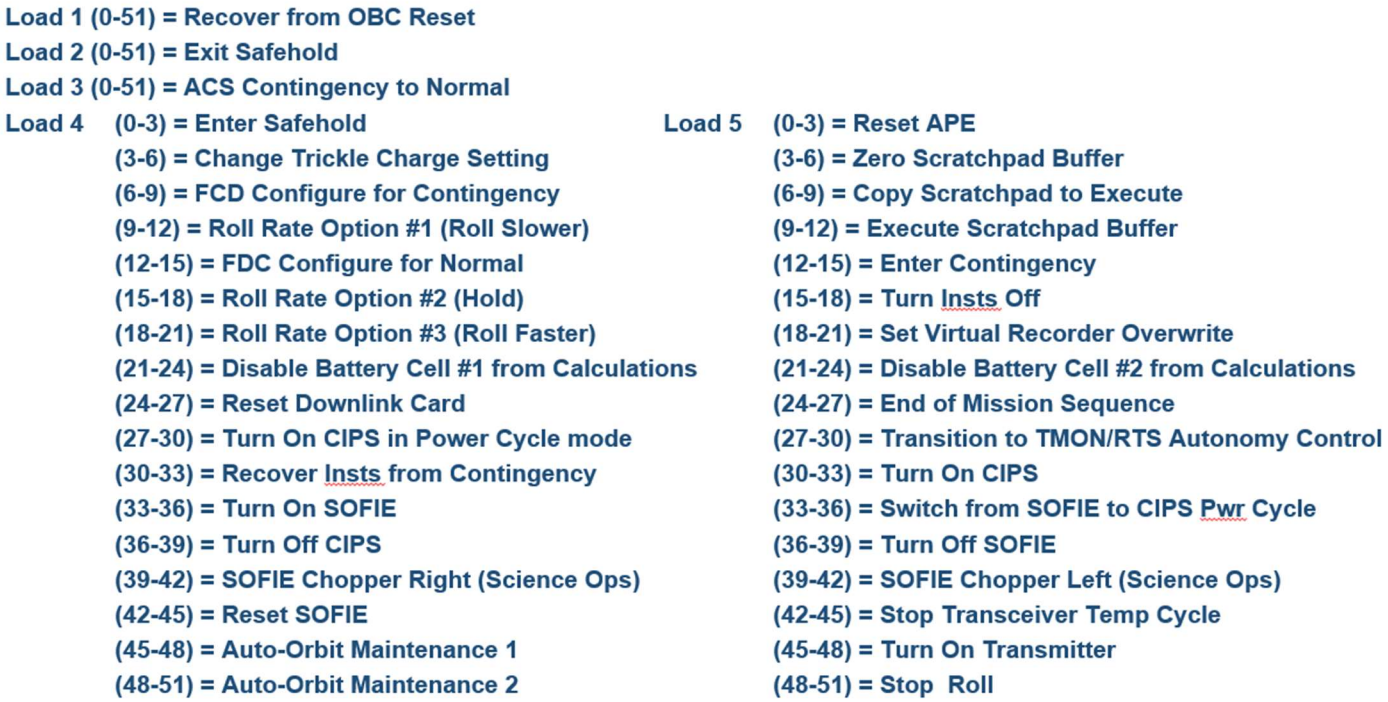

Figure 5 Example List of Library RTSs

The FOT identified the most important activities and put those first in the Library RTS lists. They considered the worst things that the spacecraft would need to recover from and also what activities the MOC would want to execute most often. In the event of an anomaly, the MOC may have wanted to put the spacecraft into safehold as soon as possible and then recover and return to science quickly, so those activities were given a high priority. Scratchpad operations that were anticipated to be used frequently and contain critical commands and were also placed early on the list. Activities such as turning the transceiver off or individual instrument turn on/off were not as high of a priority and put later in the order.

Once a Library RTS was selected, the MOC had 48 hours to execute the RTS. If the MOC failed to send the execute MC command, the Stop, Reset operation would automatically kick off, reset all of the MC on-board autonomy, and the selected RTS would no long be available to execute. The MOC would have to start over from the beginning and reenable MC in order to execute the RTS.

A Library RTS was started as soon as the FSW registered the Execute MC command and would run to completion unless stopped by the MOC from the ground. The MOC had the option of sending the Stop, Reset MC command to stop an executing Library RTS at any point after the Execute command. The Stop, Reset operation also cleaned up the necessary on-board autonomy in

preparation for the next MC activity. Another MC operation could not be started until the Stop, Reset had completed.

4. SCRATCHPAD LOADS AND AUTOMATIC MORSE CODING

The Scratchpad provided a way for the FOT to create a custom, one-off RTS that could be used to execute special activities not available in a Library RTS. While scratchpad loads were time consuming (it took one 17-minute contact to load a single word) it provided a lot of flexibility and successfully allowed the FOT to command the spacecraft in periods of no bitlock. The RTS could be any length up to 300 words.

Loading a word to the Scratchpad took 41 MC bits that needed to be sent back-to-back. This included a 16-bit command that told the FSW that the ground was loading a word to the Scratchpad, followed by a 9-bit offset that told the FSW which word was being loaded, followed by the 16-bit data word. The entire load took 13 minutes and 40 seconds to complete.

Ground automation was developed to start the procedures which loaded and verified all non-zero words. This allowed words to be loaded during non-working hours and greatly reduced the time necessary to complete a Scratchpad Load.

The FOT developed a GUI to track the status of a Scratchpad load. The GUI read the latest telemetry and real-time event messages and compared the last word written and the hex value of the data for that word to the load procedures to confirm whether or not a word was successfully loaded to the Scratchpad. Figure 6 shows an example of what the Scratchpad GUI would like during a load. Each box represented a word (memory location offset). Green indicated that a word had been successfully written with the correct data while red meant that offset did contain data, but the data was incorrect. Yellow was for offsets with data but located outside of the RTS. These would ultimately be ignored by the FSW when the Scratchpad RTS was executed. Grey boxes were zeroed-out words that were ignored by FSW. Grey boxes outlined in black were the four zeros used to indicate the end of the Scratchpad RTS. Note that there was a fifth outlined grey box. This was used by the ops team as a buffer in case the last word in the Scratchpad load happened to be zero and needed to be executed as part of the RTS. The GUI also listed which word/offset was last written to, the current value of that word in hex, and the desired value for that word (also in hex).

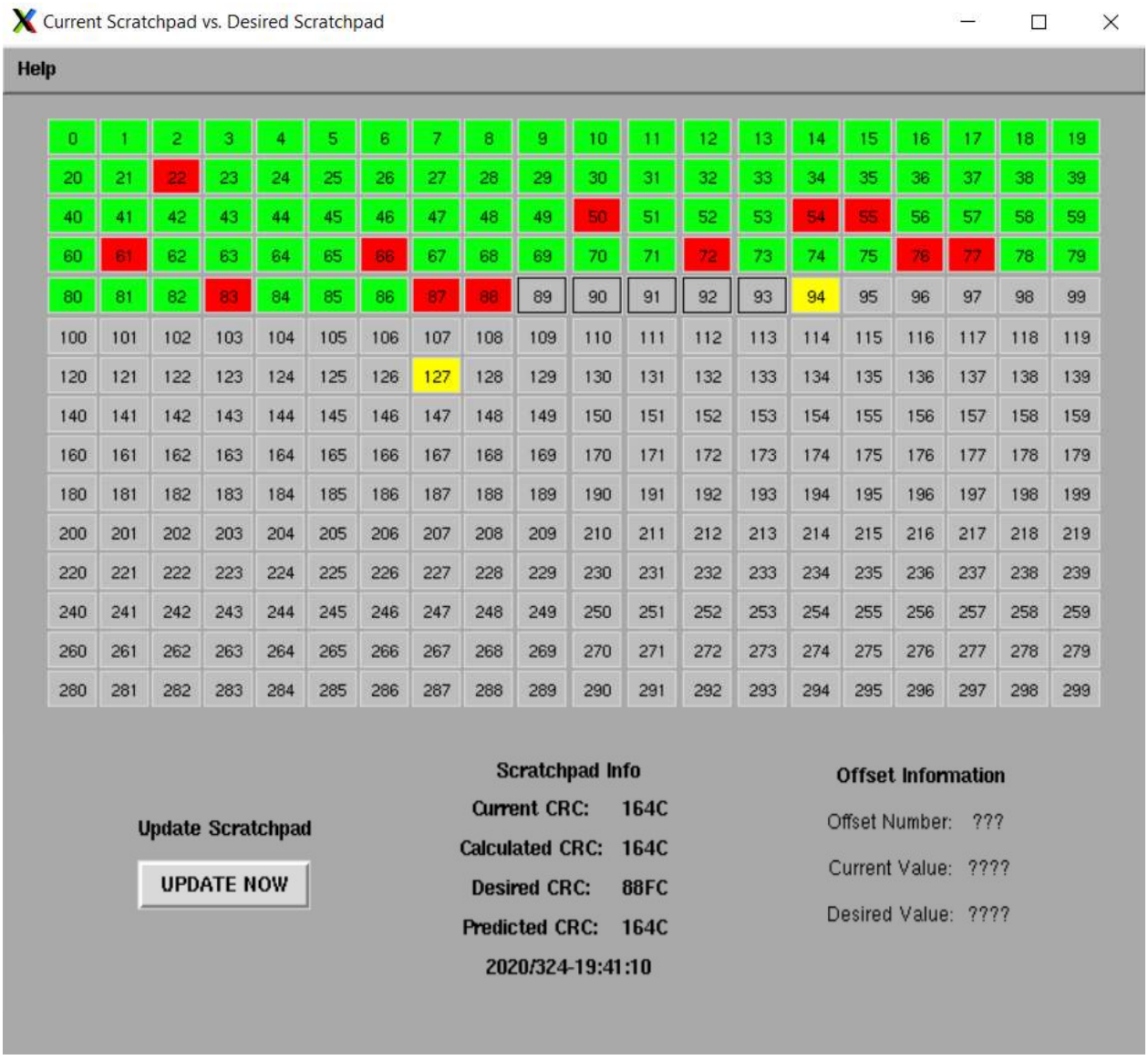

Figure 6 Scratchpad Load Monitoring GUI

Once all the words had been loaded with the correct data, the Flight Director (FD) or Flight Controller (FC) verified the Scratchpad checksums (CRCs).

After the Scratchpad load had been verified, the FOT could copy the Scratchpad to an executable location and execute the RTS. This took two rounds of Manual Morse Coding and the flow can been seen in Figure 7. The FOT used the Copy S/P to Execute Library RTS to copy the Scratchpad to the Execute Scratchpad Buffer Library RTS (i.e. the executable location). The FOT would then run through the Manual MC flow to actually execute the Scratchpad RTS by executing the Execute Scratchpad Buffer Library RTS. The Copy S/P RTS was not available until 6 hours after the MC initialize command and the Execute S/P RTS was not available until 9 hours after the initialize. Therefore, the entire process usually took about two days to complete.

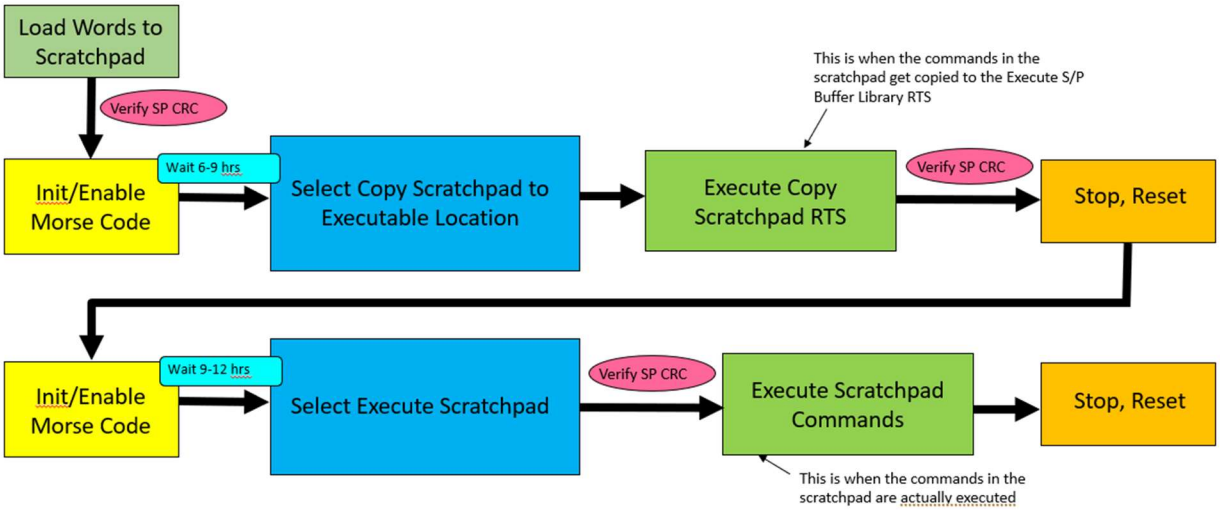

Figure 7 Scratchpad Operations Flow

The time it took to load and execute a Scratchpad varied depending on the size of the load and the availability of TDRS contacts. On average, it took 2-4 weeks but could take up to three months if doing a full scratchpad load and loading all 300 words. The FOT thus had to be very efficient in building Scratchpad RTSs and tried to limit their size as much as possible.

5. PLANNING AND SCHEDULING

TDRS scheduling was a crucial part of Morse Code operations. Contacts needed to be long enough to support all necessary commanding and the distance and line of sight between AIM and the scheduled TDRS needed to be good enough for the AIM transceiver to register the commanded change in signal. If the view between AIM and the TDRS was poor, the MOC would be able to see AIM telemetry, but the signal would not be constant enough for the FSW to accurately detect changes. In addition, there were certain times of the day where AIM was slewing and therefore MC operations were not possible.

Manual MC required TDRS contacts to be at least 8 minutes long: 2 minutes for the uplink sweep to complete after acquisition of signal (AOS), 5 minutes for the MC commanding, and a minimum of a 1-minute buffer prior to loss of signal (LOS). Contacts could be scheduled longer, and typically were, so long as the view between AIM and the TDRS was good. This provided the FC opportunity to retry a MC operation if the first attempt failed.

TDRS contacts for Scratchpad loads needed to be at least 17 minutes long: 2 minutes for the uplink sweep to complete, 5:20 to send the MC command to indicate that FSW should write to the Scratchpad, 3 minutes to send the Scratchpad offset location, 5 minutes to write the data to the designated Scratchpad location, and then at least the 1-minute buffer before LOS. Contacts were

rarely scheduled longer than 17-18 minutes. Generally, only one attempt to load to the scratchpad was made per contact. If the FSW did not properly register a 1 or a 0, then the attempt was considered a loss and the ground autonomy would try to load that word to the Scratchpad on the next Scratchpad contact.

TDRS contacts for MC operations could be scheduled at any time during the day and orbit except when the spacecraft was slewing. A spacecraft slew could distort the signal and cause MC bit flips. A list of keep-out zones was managed by the students and updated monthly.

In order to achieve the desired TDRS uplink coverage, there were three layers of TDRS scheduling. Since Scratchpad loads required longer contacts, those were scheduled first. Any remaining contacts that were at least 7 minutes long and had good views could be used for Manual Morse Coding and were scheduled next. And finally, short contacts or contacts with poor views were scheduled for monitoring or backup purposes.

Once the TDRS schedule had been finalized, the MOC could plan the MC activities. For Manual MC activities, the FD or FC would reference the list of Library RTSs and determine how long after initializing MC would the desired RTS become available. They would then cross-reference the TDRS schedule and find the sets of contacts that were spaced far enough apart that if the FC enabled MC on the first contact, the desired RTS was available for selection on the second contact in the set. A prime and a backup contact were selected for each MC operation (enable, select, execute, stop/reset) in case the first attempt at sending the MC command failed or there was a blown pass. Figure 8 is an example of how the FOT would schedule the Copy Scratchpad Library RTS, which is available 6-9 hours after the MC is enabled/initialized.

AIM	2023/051	05:37:09	05:59:24	275	86881	22	--	Sun Feb 19 10:37PM
AIM	2023/051	07:11:08	07:33:57	275	86882	22	--	Mon Feb 20 12:11AM ← MC Init Prime
AIM	2023/051	08:46:52	09:08:29	275	86883	22	--	Mon Feb 20 01:46AM ← MC Init Backup
AIM	2023/051	10:25:48	10:43:01	275	86884	22	--	Mon Feb 20 03:25AM
AIM	2023/051	13:32:48	13:52:06	TDE	86886	22	--	Mon Feb 20 06:32AM ← MC Select Copy Scratchpad Prime
AIM	2023/051	15:04:59	15:26:38	TDE	86887	22	--	Mon Feb 20 08:04AM ← MC Select Copy Scratchpad Backup
AIM	2023/051	16:39:09	17:01:10	TDE	86888	22	--	Mon Feb 20 09:39AM ← MC Select Copy Scratchpad Backup

Figure 8 Example TDRSS Schedule and MC Planning

As the need for executing Scratchpad loads increased, the FOT created a small team of command controllers (students) and trained them to do the MC Enable and Stop, Reset on their own, without a professional.

6. EXAMPLE OPERATIONS

Morse Coding allowed the FOT to execute critical spacecraft and science operations. These activities include, but are not limited to, recovering from anomalies, updating spacecraft and instrument autonomy for full-sun operations, and managing the aging battery. Without Morse Code, the AIM mission would have ended years earlier.

6.1 Anomaly and Safehold Recovery

One of the primary uses of Library RTSs was to recover from Safehold after an anomaly. There was a period early in the mission where the spacecraft entered Safehold five times. MC was not in place during the first entry into Safehold and it took 39 days to recover with intermittent bitlock. At one point, there was a bitlock outage that lasted for 12 days and the MOC was stuck waiting for bitlock to return to continue the recovery. With the installation of MC, the FOT was able to drastically shorten the recovery time and recovered from the other four Safehold events in just 3.5 days.

6.2 Table and RTS Updates

The Scratchpad was used to make routine updates to spacecraft tables and RTSs (both regular and MC Library). Depending on the size of the table or RTS, a full reload could take anywhere from several weeks to several months to load to the Scratchpad. There was rarely enough time in the mission operations schedule to allow for this, so the MOC had to adapt. Instead of executing a full reload, the MOC would simply load the new data directly (poke) to the memory address of the desired table value or RTS command. For example, if the MOC wanted to change a heater setpoint value, they would create a command to poke only that value in the spacecraft table instead of loading the entire table. This meant updates could be made with just a few commands. This method was highly successful in allowing the FOT to make critical table and RTS updates relatively quickly.

6.3 Full-Sun Preparations

AIM on-board autonomy was designed to trigger many activities off eclipse enter and exit. With the upcoming full-sun periods, Earth eclipses would cease and much of the on-board autonomy would fail to work as designed. The FOT spent nearly two years preparing for full-sun operations and relied exclusively on MC operations.

During nominal operations, the CIPS instrument had a sequence that triggered at every sunrise and would run for one orbit. It executed all of the desired calibrations and science and did not require any action from the ground. Once in full-sun, the sunrise TMON would start only once, collect science data for one orbit, and then stop until the next eclipse which would occur months later. The MOC created sequences that would continue to loop and not depend on eclipse transitions. The new sequences consisted of multiple files that were loaded into the instrument and started during the transition into full-sun. It took over six months to load the CIPS FSW with the new sequences.

The Telemetry Monitors (TMONs) and RTSs that were previously used to control CIPS science were also disabled as part of the enter full-sun activities and re-enabled once the full-sun period ended.

It should be noted that SOFIE could not operate in full-sun since it monitored the sun at sunrise and sunset only, and those did not occur in full-sun.

With the return of bitlock during the full-sun periods, the exit full-sun activities could be executed quickly and easily from the ground using nominal commanding. However, the MOC had to rely on MC to execute all of the enter full-sun activities for the first full-sun period.

6.4 Battery Maintenance

As the battery aged, updates to the charge control became increasingly frequent. Sunset and sunrise TMONs and RTSs that managed various loads (heaters, instruments, etc.) in eclipse also needed periodic adjustments and changes. At first, the MOC could afford to wait a week or two for a current Scratchpad load to finish and execute before making a battery-related change. But eventually, the battery degraded the point where charge control updates needed to be made immediately and the MOC only had a few days to implement a change. Waiting too long increased the risk of an undervoltage event that would power off the OBC and wipe critical on-board autonomy. This was a mission-ending scenario and needed to be avoided at all costs. To accommodate quick-turnaround battery changes, the MOC implemented several strategies using the Scratchpad.

The initial strategy involved predicting when an adjustment to the battery parameters would be needed and either starting a special battery parameter update load or folding those updates into a planned Scratchpad load. The exact values did not need to be decided right away. The MOC could monitor battery performance while the rest of the Scratchpad was loaded, and then determine the correct battery parameters and load those last. If no modifications were necessary, the MOC could simply reinforce the current values or hold off on executing the Scratchpad sequence until a change was required.

This method worked well when the battery was still somewhat stable and a change would be good for two to three months. However, the battery continued to degrade and the frequencey of charge control adjustments increased. The MOC found it inefficient to constantly build unique Scratchpad loads for battery maintenance or try and optimize planning to either combine a battery update with another Scratchpad activity or delay the other Scratchpad load until the battery changes were complete.

To streamline the process and allow for maximum flexibility, the FOT put together a command template that would occupy a permanent section of the Scratchpad and execute at the start of every Scratchpad activity. The template was 121 words (~40% of the Scratchpad) and consisted of the following:

- 3 RTS commands that could be configured to enable, disable, start, and stop any RTS
- 2 TMON commands that could be configured to enable or disable any TMON
- A command to adjust the clock frequency

- 4 table poke commands that could be used to modify various table parameters. These were primarily used to update battery parameters such as the minimum/maximum charge rates, trickle charge settings, and taper charge settings.
- A command to clear all Scratchpad memory locations EXCEPT the locations used for the template

The remainder of the Scratchpad could be used to execute other activities. If no RTSs, TMONs, or tables needed to be modified, the FOT would update the Scratchpad template to simply reinforce what was already on-board. If changes were required, they were loaded first and four zeros (used to indicate the end of a Scratchpad RTS sequence) were always left at the end of the template. This provided much needed flexibility in the event that an immediate modification was required. The MOC could verify the template was updated accordingly, copy the Scratchpad to the executable location, and have only the template portion execute. The four zeros would eventually be filled in, but not until the rest of the words were loaded to the Scratchpad.

The template proved to be highly successful and was frequently utilized during the last couple years of the mission. It gave the FOT a way to implement quick turnaround changes and most certainly helped prolong the battery life and life of AIM.

6.5 Contingency Operations When Making Changes

Anytime the FOT modified a battery parameter, updated a heater setpoint, or made any change to a table that could impact spacecraft health and safety, they had to consider the scenario where the change did not have the desired effect and negatively impacted spacecraft performance. Creating separate, contingency Scratchpad loads to execute if things went poorly was not an option. Even if the Scratchpad was fully updated with contingency commands, it would take too long to copy it over to the executable location and execute.

To address this concern, anytime a parameter was updated, the FOT included a command to backout the change and return the parameter to its original value. This was done in the same Scratchpad load but with a long, 18+ hour delay between the commands to make and backout the change. This gave the FOT time to monitor the spacecraft and verify the change did what the FOT intended it to do. If the spacecraft performance was nominal, then the MOC would execute the Stop, Reset MC procedure and stop the Scratchpad sequence before the backout commands executed. If the update had an undesirable effect, then the MOC would let the Scratchpad sequence complete and send commands to undo the change.

7. CONCLUSION

The slow turnaround time of Morse Code operations ended up being a leading cause to the end of AIM operations. Towards the end of the mission, the MOC struggled to make battery adjustments fast enough to keep the battery temperatures down while also maintaining enough charge to operate the spacecraft, specifically the OBC, in eclipse. Eventually, enough common pressure vessels (CPVs) failed and the battery was not able to hold enough charge to keep the OBC powered

during eclipse. The OBC powered off and the ability to Morse Code was lost. The MOC monitored AIM for many weeks hoping to see bitlock, but that never occurred, and the final attempt to communicate with AIM occurred in the summer of 2023.

Thanks to the ingenuity of the entire Flight Operations and engineering teams, AIM was a highly successful mission. Despite significant communication difficulties, AIM flew for over 16 years and collected an invaluable amount of new and exciting science data. On-board autonomy developed early in the mission allowed the mission to continue with little to no ground commanding. The Morse Code alternative commanding proved to be an extremely effective way to command the spacecraft. Though slow and limited, the AIM FOT was able to use Morse Code to not only make routine updates, but also prepare for challenging scenarios, such as full-sun, and maintain the battery. LASP has taken strategies and lessons learned from AIM and applied them to other missions and will continue to do so in the future.

8. ACKNOWLEDMENTS

Many people and organizations contributed to the success of AIM. The authors would like to thank all of the engineers at Northrup Grumman who worked tirelessly to redesign the AIM conops and help create the brilliant alternate command strategy that was Morse Coding. We would like to specifically thank John Fulmer, Alan Wang, Grace Baird, and David Oberg. Thank you to David Gathright for his work on CIPS and all of the engineering support teams at GATS, SDL, and NASA. Thank you to all the professional engineers and students at LASP who supported operations at all hours of the day and night to keep AIM alive and happy and taking good science data. And thank you to the White Sands Scheduling group who helped ensure AIM had all the contacts she needed. We could not have done it without you all.

ACRONYMS

ACS – Attitude Control System
AIM - Aeronomy of Ice in the Mesosphere
AOS – Acquisition of Signal
CC – Command Controller (student)
CDE – Cosmic Dust Experiment
CIPS – Cloud Imaging and Particle Size
CPV – Common Pressure Vessel
FC – Flight Controller
FD – Flight Director
FOT – Flight Ops Team
FSW – Flight Software
GATS – G and A Technical Services
GMU - George Mason University
HU – Hampton University
LASP – Laboratory for Atmospheric and Space Physics
LOS – Loss of Signal

MC – Morse Code
MOC – Mission Operations center
NRL – Naval Research Laboratory
NTNU – Norwegian University of Science and Technology
PMC – Polar Mesospheric Clouds
RF – Radio Frequency
RTS – Relative Time Sequence
SDL – Space Dynamics Laboratory
SOFIE – Solar Occultation for Ice Experiment
S/P – Scratchpad
TDRS – Tracking and Data Relay Satellite
TMON – Telemetry Monitor
USU – Utah State University
VT – Virginia Polytechnic Institute and State University