MESSENGER: Flight Software Design for a Deep Space Mission

David A. Artis, Brian K. Heggestad, Christopher J. Krupiarz, M. Annette Mirantes, J. Doug Reid
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099
E-mail: {David.Artis, Brian.Heggestad, Christopher.Krupiarz, Annette.Mirantes, Doug.Reid}@jhuapl.edu

Abstract—MErcury Surface, Space ENvironment, GEochemistry, and Ranging (MESSENGER) is a NASA Discovery mission to study the planet Mercury. Launched in August 2004, it will perform one more flyby of Venus and three flybys of Mercury, followed by Mercury orbit insertion in 2011 for a one-year science-gathering mission. Throughout the mission, MESSENGER will use seven instruments to collect data about key characteristics of the planet to understand Mercury and the formation of the inner solar system. [1] [2]

With the requirement that MESSENGER operate at distances of tens of millions of kilometers from Earth, the flight software team at The Johns Hopkins University Applied Physics Laboratory (JHU/APL) designed a system software architecture to work within the confines of deep space as well as the hostile near-Sun environment of Mercury. The constraints imposed by this setting include limited on-board storage, low bandwidth, long light-time delays, and intermittent connectivity with Earth. The team systematically targeted these problem areas by introducing various technologies: (1) an on-board file system using configurable telemetry storage to address the limited on-board storage, (2) a prioritized naming scheme for the file system providing for optimal use of low bandwidth to ensure that critical data are received by the ground as required, (3) use of the Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP) standard for reliable file data delivery, to address the light time delays and intermittent connectivity encountered by MESSENGER, and (4) a configurable and robust autonomy and safing system to protect the spacecraft while it is out of contact with Earth.

The MESSENGER guidance and control (G&C) software is a key component in all phases of the mission. This highly complex set of algorithms maintains spacecraft attitude, manages spacecraft momentum, executes deep-space propulsive maneuvers, controls the solar arrays for optimized pointing to the Sun, manages spacecraft thermal environment by ensuring the sunshade always faces the Sun, and, finally, enables a host of pointing options and instrument pointing control in support of science operations. All of this complexity was modeled in Mathwork's SIMULINK environment and was ultimately auto-generated into flight code using Real Time Workshop (RTW). In support of this code were over one thousand different parameters as well as complex and extensive management of on-board ephemerides. Additionally, the G&C software was co-resident with the command and data handling software, consuming approximately slightly more than half of processor bandwidth and memory.

This paper first provides a high-level overview of the MESSENGER flight computer system including both the overall hardware and software architectures. Following these descriptions is a detailed review of the specific new technology introduced to address the constraints and limitations of operating in the deep space and Mercury environments. The paper concludes with an analysis of the performance of the software over MESSENGER's first year in space including details of G&C operations during mission milestones relating to deep space maneuvers and the uploading of upgraded versions of the flight software.

TABLE OF CONTENTS

1. INTRODUCTION

MErcury, Surface, Space ENvironment, GEochemistry, and Ranging (MESSENGER) is a NASA Discovery mission to study the planet Mercury. MESSENGER was launched on August 3, 2004, flew by the Earth a year later in 2005 with a resultant image shown in Figure 1, and swung by Venus on October 24, 2006. The remaining major events of the mission include one more flyby of Venus, three flybys of Mercury, and, finally, insertion into Mercury orbit in March 2011 for a one-year science-gathering mission [1]. Throughout all phases of the mission, MESSENGER will use seven instruments to collect data about key characteristics of the planet to further understand Mercury and the formation of the inner solar system [2]. As part of

[1] 1-4244-0525-4/07/$20.00 ©2007 IEEE.
[2] IEEEAC paper #1089, Version 6, Updated July 1, 2006

the system software developed by The Johns Hopkins University Applied Physics Laboratory (JHU/APL), MESSENGER accommodates the constraints placed on it by operating in a deep space mission environment.

Figure 1 – Earth from MESSENGER, August 2005

The MESSENGER flight software runs on flight electronics called the integrated electronics module (IEM). The IEM has two RAD6000 processors: the main processor (MP) and the fault protection processor (FPP). The MP is responsible for both command and data handling (C&DH) and guidance and control (G&C). The MP has 8 MB of RAM storage, and it communicates with a 1-GB solid-state recorder (SSR) via the peripheral component interconnect (PCI) backplane. The application software runs over the VxWorks operating system developed by Wind River, Inc.

Figure 2 shows MESSENGER instrument data flow. The MESSENGER flight computer collects data from several instruments depicted in blue including the MESSENGER Dual Imaging System (MDIS), the Gamma-Ray and Neutron Spectrometer (GRNS), the Magnetometer (MAG), the Mercury Atmospheric and Surface Composition Spectrometer (MASCS), the Energetic Particle and Plasma Spectrometer (EPPS), and the X-Ray Spectrometer (XRS) [3]. With the exception of MDIS, instruments flow their data in the form of Consultative Committee for Space Data Systems (CCSDS) data packets through the Data Processing Unit (DPU) shown in green and then across the 1553 bus to the MP. In the case of MDIS, data flow directly to the IEM via a high-speed interface. Image segments are collected in a buffer within the IEM and then transferred via Direct Memory Access (DMA) to the SSR depicted in yellow. Images are compressed when necessary and downlinked to the ground via CFDP.

2. Considerations for a Deep Space Mission

In the early days of defining the MESSENGER spacecraft architecture and mission design, several unique aspects of the mission became key drivers that greatly influenced flight software design choices:

- As a deep space mission, MESSENGER has much lower communication data rates than are typical with Earth-orbiting spacecraft. The communication rate can range from 10 bps to 104 kbps with round-trip light-time delays as long as 24 minutes. The low downlink data rate, combined with an analysis of science data collection while in orbit around Mercury, resulted in selecting an SSR that could store 1 GB of data.

- Storage of images taken by MDIS, while in orbit at

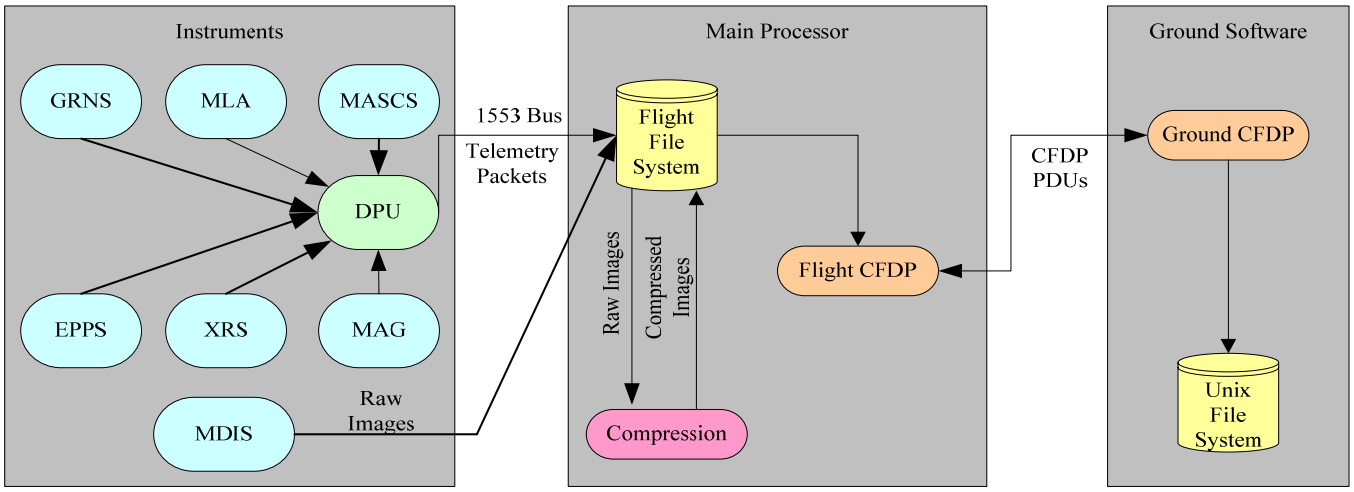

Figure 2 – MESSENGER instrument data flow with corresponding mirrored CFDP and file storage

Mercury, would be a major challenge. Because of the low downlink rates coupled with an image acquisition rate of 3.2 Mbps, images would have to be stored while close to Mercury, and transmitted to Earth at a later time. To conserve SSR space and downlink bandwidth, images that are initially stored on the SSR in uncompressed format must be compressed for longer-term storage.

- MESSENGER is a 3-axis stabilized spacecraft with pointing controlled by G&C software that operates reaction wheels and a complex propulsion system. The spacecraft sunshade must at all times be kept pointing to the Sun to ensure that delicate spacecraft electronics are protected from damaging solar heat. While temperatures on the face of the sunshade can reach 350° C, the shaded electronics remain at room temperature and even require heaters. Should G&C fail to keep the spacecraft Sun-pointed for any longer than 30 min, the spacecraft could be lost due to overheated electronics and solar arrays that fail to charge the battery.

- At times during the multi-year journey to reach Mercury, solar conjunctions will block the Earth from having communication contact with the spacecraft. These blackout periods can be as long as 40 days. Software must support a rule-based autonomy system that could monitor for faults and take corrective action, such as switching to the backup unit for a specific subsystem on the highly redundant spacecraft, without waiting for ground intervention. Furthermore, G&C software would need to be able to operate thrusters autonomously should some anomaly cause a Sun-pointing violation.

These key features of the MESSENGER mission led to several software design decisions that were new for a JHU/APL spacecraft mission.

On-board file system and CCSDS File Delivery Protocol

Given the desire to maximize the return of science data and the need for flight software to store images and compress them at a later time, it was decided to manage the SSR using a file system. This allowed the SSR to be treated as if it were a virtual disk drive, enabling the software to read files containing uncompressed images, apply an integer wavelet compression algorithm, and store the smaller file back to the SSR. The file system allowed multiple usages of the same memory areas depending on what was needed. A given area could easily be used at one time for raw science data storage and at another time for compressed image storage or SC housekeeping and engineering data storage.

Another innovation was the integration of a newly emerged standard for the transmission of file data, the CCSDS File Delivery Protocol (CFDP) [4]. This standard is essentially an "FTP in space" process that transfers a file between the spacecraft and the ground system using a guaranteed data delivery protocol with handshaking between the flight and ground CFDP clients to retransmit pieces of a file lost due to data dropouts. MESSENGER became the first U.S. mission to launch with CFDP. The ability for software to automatically retransmit missing data rather than entire images or files maximizes the downlink of science data. Prior JHU/APL missions had treated data storage devices as virtual tape recorders using ground commands to manipulate read and write pointers.

Automated File Playback

To simplify the management of the flight file system and CFDP, MESSENGER flight software implements an automatic SSR playback process. A prioritized directory structure was created for the storage of science and engineering files. This directory structure was organized according to downlink priority. The software "walks" through directories in priority order, continuously transmitting files of data. If necessary, these priorities can be overridden. At the beginning of a data transmission contact with Earth, the playback manager is enabled by command and begins to transmit the data of highest priority. The use of CFDP, the file system, and the priority directory structure solved another concern related to the long periods of time the spacecraft would be out of contact with Earth. During those times, large amounts of engineering data are stored in low-priority directories. Should an anomaly occur while out of contact, the ground team can later promote those files to a higher priority and downlink them to support analysis activities. But when everything is nominal, CFDP directives can be sent to the spacecraft to delete the unneeded contingency files, freeing valuable SSR space and downlink bandwidth for the science data.

Autonomy

Given the critical requirement to maintain a Sun-pointing attitude, and the fact that there would be long periods of time when MESSENGER would have no Earth contact, a sophisticated autonomy system was designed into the flight software. Relevant engineering telemetry data are collected each second and stored in an onboard data collection buffer (DCB). Autonomy rules, expressed in reverse polish notation, are uploaded to the spacecraft. Each rule has conditional logic in the premise to check data values in the DCB. A rule evaluation engine in the flight software processes each rule every second; for each rule that evaluates "TRUE," a corresponding stored command sequence is executed to take corrective action. This rule capability allows the autonomy system to respond to faults and take actions such as switching to the backup flight processor, switching to a backup star tracker, turning off power loads if a low voltage condition arises, and so on.

Guidance and Control

MESSENGER flight software includes sophisticated G&C attitude estimation and attitude control algorithms. The software not only maintains a 3-axis stabilized Sun-pointing

attitude, but it also supports a number of pointing options to facilitate instrument operations while in orbit around Mercury, such as a limb-scanning function to support acquiring image mosaics from the MDIS instrument. G&C also manages the propulsion system and the flow of fuel in tanks to support thruster maneuvers, which can be commanded from ground control or can take place autonomously in the event of a Sun-pointing violation or a need to reduce momentum to slow down the reaction wheels. G&C also controls rotation of the solar panels to optimize power and keep the panels within a designated temperature range, controls a phased array antenna to keep it pointing at Earth, and controls an optical pivot platform in the MDIS instrument.

The remainder of this paper will discuss more details about the key features of the MESSENGER flight software.

3. On-board File System

MESSENGER data are stored in files onboard the spacecraft [5]. This differs from past JHU/APL missions where data are stored on the SSR in raw partitions which are managed via a custom method of data storage. With low bandwidth and long time delays, a file system facilitates a more autonomous method of data management.

As described previously, onboard data flow for MESSENGER starts at the instrument and ends at the SSR. The file system processing software is configured by mission operations to store telemetry packets that are received from the instruments, as well as from internal MP tasks, into files that are contained in a prioritized directory shown in Figure 3. When commanded, the flight software steps through this directory structure and starts downlinking files in order of time received. This directory structure allows mission operations personnel to place important information in higher priority directories to ensure the data will be returned expeditiously. As the data tree is traversed, it becomes less and less likely that a particular file will ever be downlinked due to bandwidth constraints. As a result, low-priority engineering data are typically placed in the lowest priority P9 directory and deleted on a periodic basis. If an anomaly occurs, the software provides the ability to move files to higher priority directories to cause the flight software to reevaluate the importance of those data.

4. CCSDS Delivery File Protocol

CCSDS is a standards body consisting of the agencies and organizations from space faring countries worldwide. The committee members develop and discuss recommendations in order to create a standard set of protocols and operations across missions and across agencies. As part of this effort, CCSDS developed the CCSDS File Delivery Protocol to provide a method for transferring files between two points in a space network.

In the course of developing the MESSENGER flight software, the MESSENGER team concluded that the spacecraft's flight profile required a standard method for delivering files from the spacecraft to the Mission Operations Center (MOC) at JHU/APL. The team evaluated CFDP, but at that time the protocol was still in its infancy and was deemed premature for flight. The flight software developers took a different track to create a customized protocol for file transfer, but after an initial attempt, they

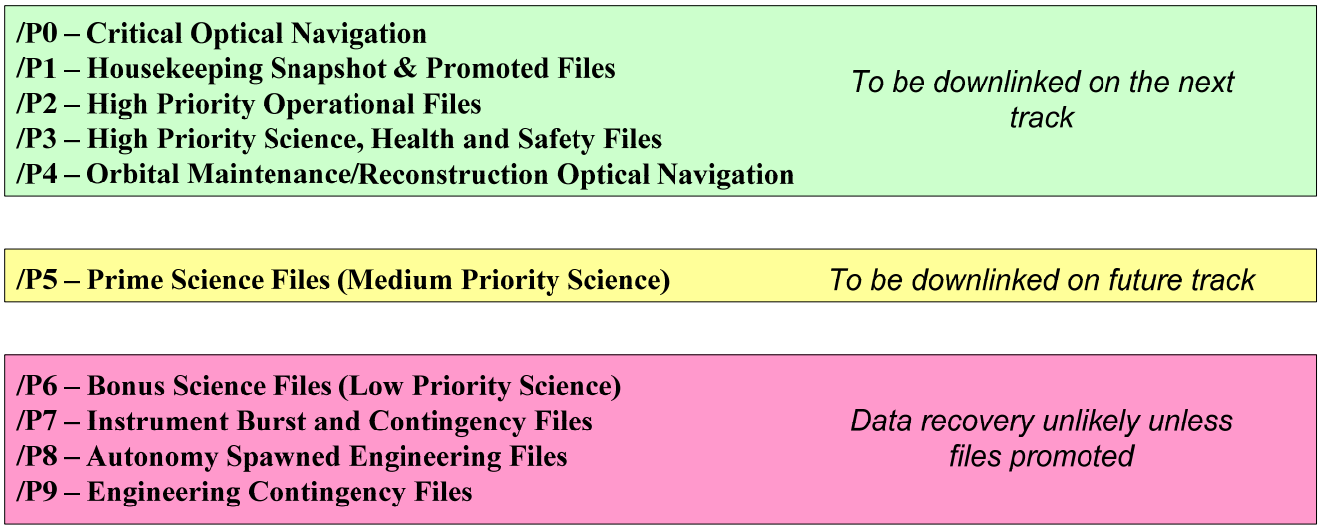

Figure 3 – MESSENGER file system downlink directory layout provides for prioritized downlinking

realized that the end result was going to be remarkably similar to CFDP and, as result, turned back to CFDP as a solution for this problem.

CFDP is configured in multiple ways. It can use a reliable method of file system communication where all data are guaranteed to be transmitted, or it can perform a "best effort" where data delivery is not assured. CFDP is tuned to the operational environment, whether that is low Earth orbit or deep space.

In the case of MESSENGER, the MESSENGER software team selected a deep space configuration, and the following example, diagramed in Figure 4, is illustrative of a typical CFDP transaction. First, a transaction is started either via command or the aforementioned autonomous downlink capability. Next, initial information concerning the file is transmitted in a metadata protocol data unit (PDU). A PDU is the common language between CFDP entities with a metadata PDU being a particular kind. Next, the file is divided into smaller segments and these file pieces are transmitted in file data units (FDUs). This continues until an end of file (EOF) PDU is communicated to the ground system indicating the file has been completely sent. During this time, the ground software at the MESSENGER MOC is accumulating the FDUs and reproducing the file local to the MESSENGER operations team and scientists. Along with creating the file, the CFDP ground software is also performing an accounting procedure to ensure that all pieces of the file have been received. If any are missing a negative acknowledgement (NAK) is transmitted back to the spacecraft. The NAK contains information on which pieces of the file are currently missing in the received file. Once the flight software receives a NAK, it retransmits the missing piece of a file and, when that is received on the ground, the MOC software responds with a finished indicator (FIN). MESSENGER receives this transmission, acknowledges to the ground that it did so, and then closes the transaction.

The keys to CFDP in the deep space domain are the handshake elements and their corresponding timers. CFDP does not require a large amount of protocol handshaking, unlike typical terrestrial file system schemes. This is necessary in the deep space since a single handshake (such as a sender asking a receiver if it is ready to receive a file and the receiver responding that it is) costs one round trip light time. This is insignificant near Earth where that round trip light time can be measured in milliseconds, but in deep space the round trip can be minutes or even hours. Additionally, CFDP as configured for MESSENGER assumes an unreliable link between the two elements in the space network. This adds a series of timers at three points in the data transmission. The first is started when the EOF is sent to the ground, the second when any NAKs are sent, and the third when the FIN is sent. There is also an overall inactivity timer which ensures that the system as a whole is still functioning. These timers accommodate a typical noisy

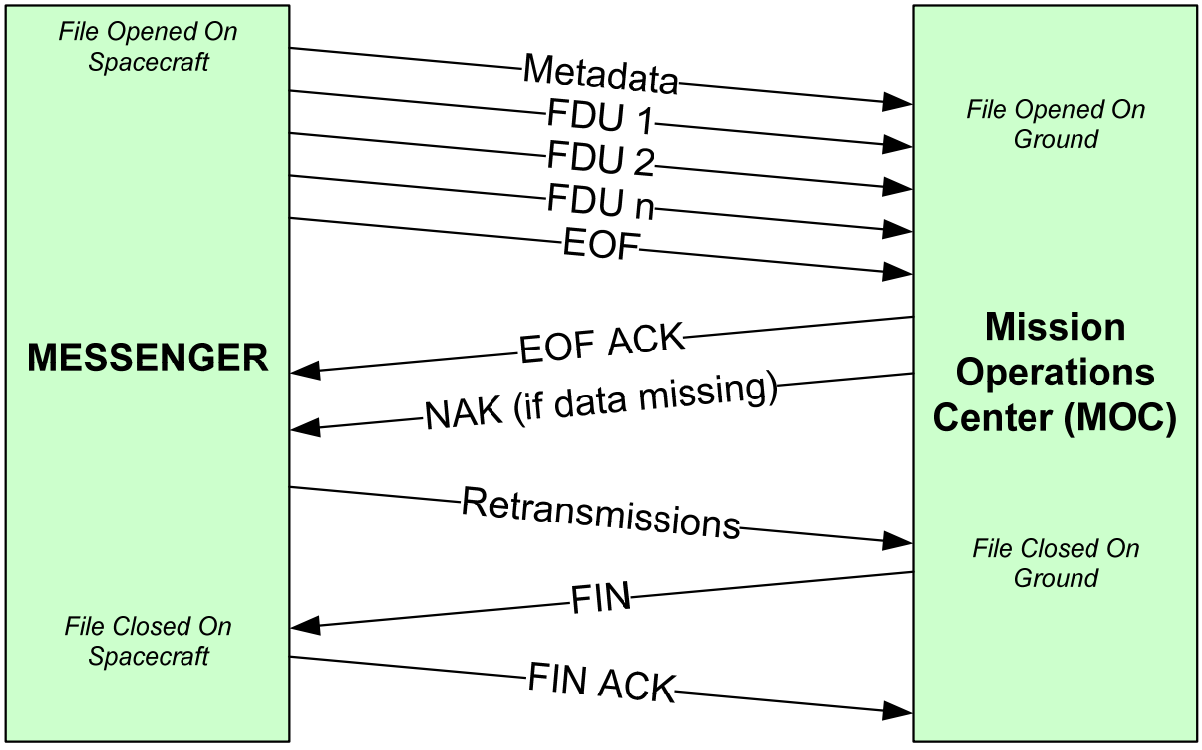

Figure 4 – Typical CFDP transaction provides reliable transmission of spacecraft file data

deep space link in that the EOFs, NAKs, and FINs are resent if the CFDP entity realizes that it has never received a response to the respective transmissions.

The MESSENGER software system uses two different implementations of CFDP. The first is a JHU/APL custom implementation on the spacecraft that meets the needs of a processor- and memory-constrained mission [6]. The second is for the ground software and is built around a CFDP implementation developed by the NASA Jet Propulsion Laboratory (JPL) [7]. The use of the JPL ground software is made possible because CFDP is an international standard. This type of software reuse would have been impossible with a closed, proprietary file transfer protocol.

The addition of CFDP has enabled the reliable transfer of thousands of files, but the introduction of the protocol was not without pitfalls (although none were directly a result of the protocol itself). Two primary lessons learned have been derived from the experience of operating CFDP. The first is to ensure that the default timer settings are sufficient for initial operations that are typically not nominal. During the initial days of the MESSENGER mission, there were some hiccups that resulted in prolonged transaction times, which resulted in some files not being properly delivered to the ground system. Since extra bandwidth was available during this period, some laxness in the timer settings would have allowed for the data to be properly delivered. The second main lesson learned was to have a CFDP engineer responsible for overseeing the closed-loop operation of CFDP.

5. AUTONOMY

The autonomy engine software runs on the Main Processor (MP) as well as the Fault Protection Processors (FPP). The FPPs are loaded with the rules responsible for the spacecraft safety and can support up to 512 rules. To date, the FPPs have been loaded with over 200 rules, which monitor telemetry to perform safing operations such as mode demotion, hardware switching, and power monitoring. The FPP rule premises also use ground configurable parameters such as mission phase to allow specific rules to be enabled or disabled during certain times in the mission. The MP contains rules that perform spacecraft maintenance sequences and can support up to 256 rules. These maintenance rules include routine SSR file operations as well as routine RF hardware reconfiguration sequences.

Each FPP executes identical application code that supports a command and telemetry interface to the MP. The main purposes of each FPP are to perform fault detection and to execute the appropriate responses. Each FPP implements health and safety rules that operate on data collected from the 1553 data bus, including a state message transmitted by the primary MP. Each FPP also serves as a 1553 bus monitor to collect spacecraft data that can be monitored by autonomy rules. A triggered rule can dispatch a command or series of commands via the primary MP to correct faults. The FPPs also have a custom serial interface to the power distribution units to receive critical status or send special commands if the loss of 1553 bus communications or a failed MP is detected. This interface to the power distribution unit allows the FPPs to take independent action such as resetting the MP or swapping the current MP bus controller.

6. GUIDANCE AND CONTROL

As stated in the introduction, the G&C software co-resides with the C&DH software in the RAD6000 computer. In fact, the G&C comprises approximately 50% of the linked object code and requires approximately 30% of the CPU bandwidth. In short, it is a major component of the flight software design as it exists in MESSENGER.

If there was any one innovation that reduced the development schedule, mission risk, and also cost of the MESSENGER flight software it is that the G&C "truth" dynamic and environmental algorithms, as well as the flight control, estimation, and guidance functions, were all modeled and tested within *MathWorks'* Simulink product. Simulink, as described by *MathWorks*, is a "platform for the multi-domain simulation and model-based design for dynamic systems" [8]. It is implemented as an interactive, graphical, desk-top environment comprised of a set of customizable "block" libraries dragged and dropped into the workspace by the analyst who uses them to model the environment and his or her control strategies. The main strength offered by Simulink is the flexibility it offers – inputs, outputs, parameters, and algorithms can all be readily modified at a symbolic level. In other words, it is not necessary to modify any extant C-code and all of the ramifications of that process just to make a relatively simple change. Another real advantage is the modeling of the spacecraft post-launch. Environmental or dynamic models can be modified at the workstation level, measured data or observed events duplicated, and a problem or observation quickly explained without invoking the very expensive and often time-consuming hardware-based simulations.

Ultimately these symbolic but very real models are converted directly into ANSI-C code using another tool provided by *MathWorks*: Real Time Workshop (RTW). This process of converting the models into a form that can be ported and compiled and linked into flight or test-bed environments is called auto-generation, or auto-coding, and it is the key technology that makes it possible to convert the highly symbolic models into "real world and real time" applications. But that's not the end of the auto-coding process. There also exists a set of customized Target

Language Compiler (TLC) and Perl-based utilities that auto-code the G&C command and telemetry database and test scripts and test parameters, as well as customized "special purpose" interface functions. The TLC utilities form part of the RTW build process and are invoked as part of that process. They are able to extract fundamental model characteristics during the RTW auto-generation process such as the epoch rate of the models, the number of distinct states, whether the models were multi or single tasking, and so on. In addition, they are responsible for the auto-generation of so-called "wrapper" functions ultimately invoked by the Human-Generated Code (HGC) to initialize, check run time status, and then "step" or execute the models. In this way, the HGC does not need to know any of the details of the internal workings or structures of the auto-generated models. (It should be noted here that *MathWorks* has subsequently released its *Embedded Coder* product, which offers similar "clean" interfaces for the control and management of the models.)

The associated Perl-based auto-generation utilities are numerous and extensive in purpose, but their focus is in the auto-generation of the command data base comprised, primarily, of up to 1000 G&C parameters of differing types and sizes. Perl scripts perform further processing of the RTW auto-generated code. Commands and parameters are extracted, sorted, and formatted as required by the database. This arrangement provided the means of allowing any change to any parameter, whether it was an addition, removal, or a simple modification, to be modeled and verified within Simulink, the models re-auto-generated via RTW, the changes quickly and accurately extracted via the Perl scripts, and the command data base subsequently updated. What makes this process workable is a mnemonic naming convention for inputs, outputs, and parameters that originated at their creation within the Simulink Model. Post RTW, these named objects become part of the auto-coded c-code which, in turn, allowed the Perl scripts to parse, sort and extract what was needed for the data base.. An example of this naming convention might be: *GC_GYROPROP_FineScaleFactor* - which "tells" the Perl-based tools that the parameter is a flight (GC) parameter; it is part of the wider set of gyro properties (GYROPROP); and, specifically, it defines the so-called "fine scale factor" for the gyro.

Primarily because MESSENGER is a deep space mission and a certain amount of simplification was required from a management perspective, parameters are further organized into so-called "parameter blocks." From a G&C perspective, parameters with the same functional associations are grouped together and loaded or "committed" to the models at the same instant. As an example, all of the parameters associated with, say, the functional properties of the gyro became members of the GYROPROP parameter block. In the end, the 1000 or so individual parameters were organized into 100 parameter blocks with individual parameter blocks being managed as a single logical entity in that they are created, uploaded, verified, and stored in the on-board memory in the same way, using the same methods. In addition, methods were provided to load individual parameters but, in fact, were never used.

Another Perl-based set of utilities which is parameter-related, and which ultimately offered a great deal of flexibility and saved vast amounts of time, was the auto-generation of G&C test and verification scripts. These scripts are first run at the workstation level and then, to ensure the fidelity of the test, executed with the same anticipated results in the hardware test bed, and often at the spacecraft level. The same mnemonic naming discipline allowed this to happen. Old tests could be run with new parameters. New tests could be easily developed and verified before scheduling expensive time on the spacecraft. And post-launch, this allowed the creation of the parameters necessary for early maneuvers and deployments.

The Simulink models have the flexibility to execute in a so-called multi-tasking configuration. For the G&C there is a 50-Hz "control" task and a 1-Hz task that included the functionality of everything else required by the G&C - namely estimation, which includes a Kalman filter; guidance, which includes the ephemerides for the Sun, Earth, spacecraft, and target planet; solar array and antenna pointing logic; and a host of complex pointing modes and their associated short cuts derived from the on-board ephemeris. The ability to configure the Simulink models in a multi-tasking configuration significantly off-loads the CPU, making it possible to let the G&C run concurrently with the C&DH.

The control task is a high-priority task that was measured to take approximately 5 ms out of each 20-ms time-slice – thereby requiring about 25% of the CPU bandwidth. The low-rate task executes once a second, starting sometime after the start of the second and ending sometime before the start of the next. The data exchange between the two tasks is managed either by the HGC or by so-called "rate transition" blocks within the Simulink models themselves. It is estimated that if all the G&C functionality were allocated to a single high-priority task, it would require close to 95% of the available CPU bandwidth.

Lastly, one of the main requirements of a deep space mission providing extensive reconnaissance of a remote planet is the need for large volumes of Sun, Earth, and target planet ephemeris data which must be uploaded to the spacecraft on a weekly and sometimes on a daily basis. For MESSENGER this volume of data is estimated to be about 48 kbytes of data per week in the form of Chebyshev polynomials. To off-load the stress this introduces to the C&DH macro and time-tagged command functions, the G&C included a so-called "ephemeris manager" that buffers the ephemeris data, verifies them, reports their status – how many spans are valid, how many are expired, etc. – and,

finally, provides the next valid Chebyshev to the G&C ephemeris models when requested to do so.

7. SUMMARY

The JHU/APL MESSENGER flight software team considered several characteristics in developing the flight software for the mission. These included low downlink rates, periods of limited or no communication, and long light time delays. Operating in deep space impacted the design of autonomy, data handling, and guidance and control. The software architecture outlined in this paper continues to operate nominally during MESSENGER's third year in space.

ACKNOWLEDGEMENTS

The authors of this paper would like to thank the rest of the members of the JHU/APL MESSENGER flight software and G&C analyst teams who contributed to the successful deployment of the this architecture.

REFERENCES

[1] Santo, A. G., R. E. Gold, R. L. McNutt, Jr., S. C. Solomon, C. J. Ercol, R. W. Farquhar, T. J. Hartka, J. E. Jenkins, J. V. McAdams, L. E. Mosher, D. F. Persons, D. A. Artis, R. S. Bokulic, R. F. Conde, G. Dakermanji, M. E. Goss, Jr., D. R. Haley, K. J. Heeres, R. H. Maurer, R. C. Moore, E. H. Rodberg, T. G. Stern, S. R. Wiley, B. G. Williams, C. L. Yen, and M. R. Peterson, "The MESSENGER mission to Mercury: Spacecraft and mission design," Planet. Space Sci., 49, 1481-1500, 2001.

[2] Solomon, S. C., R. L. McNutt, Jr., R. E. Gold, M. H. Acuña, D. N. Baker, W. V. Boynton, C. R. Chapman, A. F. Cheng, G. Gloeckler, J. W. Head, III, S. M. Krimigis, W. E. McClintock, S. L. Murchie, S. J. Peale, R. J. Phillips, M. S. Robinson, J. A. Slavin, D. E. Smith, R. G. Strom, J. I. Trombka, and M. T. Zuber, "The MESSENGER mission to Mercury: Scientific objectives and implementation," Planet. Space Sci., 49, 1445-1465, 2001.

[3] Gold, R. E., S. C. Solomon, R. L. McNutt, Jr., A. G. Santo, J. B. Abshire, M. H. Acuña, R. S. Afzal, B. J. Anderson, G. B. Andrews, P. D. Bedini, J. Cain, A. F. Cheng, L. G. Evans, W. C. Feldman, R. B. Follas, G. Gloeckler, J. O. Goldsten, S. E. Hawkins, III, N. R. Izenberg, S. E. Jaskulek, E. A. Ketchum, M. R. Lankton, D. A. Lohr, B. H. Mauk, W. E. McClintock, S. L. Murchie, C. E. Schlemm, II, D. E. Smith, R. D. Starr, and T. H. Zurbuchen, "The MESSENGER mission to Mercury: Scientific payload," Planet. Space Sci., 49, 1467-1479, 2001.

[4] CCSDS File Delivery Protocol (CFDP). Recommendation for Space Data System Standards, CCSDS 727.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002.

[5] Krupiarz, C. J., D. A. Artis, A. B. Calloway, C. M. Frangos, B. K. Heggestad, D. B. Holland, and W. C. Stratton, "File-based data processing on MESSENGER," Proceedings of the 5th International Academy of Astronautics International Conference on Low-Cost Planetary Missions, Special Publication SP-542, edited by R. A. Harris, pp. 435-442, European Space Agency, Noordwijk, The Netherlands, 2003.

[5] Krupiarz, C. J., S. C. Burleigh, C. M. Frangos, B. K. Heggestad, D. B. Holland, K. M. Lyons, and W. C. Stratton, "The use of the CCSDS file delivery protocol on MESSENGER," Space Operations 2002 Conference, World Space Congress, American Institute of Aeronautics and Astronautics, Paper T5-35, 8 pp., Houston, TX, October 2002.

[6] Stratton, W. C., C. M. Frangos, J. J. Harrison, and D. B. Holland, "Reuse of the JPL CFDP software in the APL Common Ground System," Proceedings of the 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO), Paper 103-A0036, 8 pp., Pasadena, CA, July 2003.

[7] Krupiarz, C. J., B. K. Heggestad, and R. D. Carper, "Flying CFDP on MESSENGER," International Telemetering Conference 2004, Paper 04-14-02, 8 pp., San Diego, CA, October 2004.

[8] MathWorks Simulink Corporate Website, URL: http://www.mathworks.com/products/simulink

BIOGRAPHY

David A. Artis *is a member of the Principal Professional Staff in the Embedded Applications Group of The Johns Hopkins University Applied Physics Laboratory, having joined the Space Department in 1981 after completing an M.A. in Physics at Indiana State University. He was the mission software systems engineer for the MESSENGER mission. Previously he was the mission software systems engineer for the Far Ultraviolet Spectroscopic Explorer mission and was the command & data handling system flight software lead engineer for the Near Earth Asteroid Rendezvous spacecraft.*

Brian K. Heggestad *is a Senior Professional Staff Member at The Johns Hopkins University Applied Physics Laboratory's Space Department. He has developed flight and test software for several missions including Near Earth Asteroid Rendezvous (NEAR), Far Ultraviolet Spectroscopic Explorer (FUSE), and New Horizons. He received his MS in Electrical Engineering at The Johns Hopkins University.*

Christopher J. Krupiarz *is a Senior Professional Staff Member in the Embedded Applications Group of The Johns Hopkins University Applied Physics Laboratory's Space Department where he develops and tests flight software for missions such as CONTOUR, MESSENGER, and New Horizons. He is also involved in various activities regarding space networking including being the Principal Investigator on the Mars Technology Program's Next Generation Mars Protocol Suite. He received his B.S. in Computer Science (1989) from Michigan State University.*

M. Annette Mirantes *is a member of the Senior Professional Staff at The Johns Hopkins University Applied Physics Laboratory. She received a BS in Electrical Engineering from Purdue University and an MS in Systems Software Engineering from George Mason University. She previously developed flight software for numerous missions at Orbital Sciences Corporation including the ORBCOMM constellation, the Far Ultraviolet Spectroscopic Explorer (FUSE), and the Solar Radiation and Climate Experiment (SORCE). At JHU/APL, she is the flight software lead for the MESSENGER Project.*

Doug Reid *is a member of the Senior Professional Staff in the Embedded Applications Group of The Johns Hopkins University Applied Physics Laboratory's Space Department, where he develops and tests Guidance and Control flight software as well as dynamic and environmental simulation software for missions such as TIMED, CONTOUR, MESSENGER, STEREO, and New Horizons. His interests include astronomy, celestial navigation, arboriculture, entomology, and sedimentary geology. He received his B.Sc. in Applied Science from the Royal Military College, Canada, and a Masters in Applied Physics from The Johns Hopkins University.*