JAXA-RPR-MX16306
RELEASE DATE: Mar.1, 2017

MARTIAN MOONS EXPLORATION (MMX) MISSION

COMPONENT TELEMETRY AND COMMAND DESIGN CRITERIA

REVISION AND HISTORY PAGE

Status	Revision Number	Change Number	Description	Release Date

TABLE OF CONTENTS

REVISED PAGE LIST

pp. Rev	pp. Rev	pp. Rev	pp. Rev	pp. Rev

1. INTRODUCTION

1.1. Purpose and scope of this document

This document (MMX-C-TCDC) describes the criteria on telemetry and command designs for Martian Moons Exploration (MMX) on-board components.

In case of conflict, the MMX-I-IRD will take precedence over this document.

1.2. References

See Section 10.

1.3. Notation

[TBD-Sys] [TBC-Sys]
The results of previous study for MMX or other projects are described as reference information for systems and the PI instrument. They will be established by system design in Phase A or beyond.

[TBD-Sys/PI] [TBC-Sys/PI]
The results of previous study for MMX or other projects are described as reference information for systems and the PI instrument. They will be established through coordination between the PI instrument and systems in Phase A or beyond.

[TBD-Doc] [TBC-Doc]
Information that can be described through coordination with design standards and other documents. (Phase A or beyond)

[TBD-Plan] [TBC-Plan]
Information determined after government approval of the project plan.

2. FUNCTIONAL MODEL OF SPACECRAFT

The function of the entire MMX spacecraft shall be described according to the latest issue of GSTOS 201 "Functional Model of Spacecraft (FMS)" ([A1]).

As the telemetry and commands of each component are aspects of the spacecraft function, they shall be registered and managed in the Spacecraft Information Base 2 (SIB2), which is defined in the latest issue of GSTOS 300 "Definition of Spacecraft Information Base 2 (DSIB2)".

SIB2 is directly read in by the Generic Spacecraft Test and Operation Software (GSTOS) (TBD-Doc) and used to issue commands and interpret the telemetry by ground stations.

TBD-Sys

3. END-TO-END PROTOCOL ARCHITECTURE

The protocol of the telemetry and commands shall be designed according to the latest issue of GSTOS 200 "Spacecraft Monitor and Control Protocol (SMCP)" ([A10]).

SMCP treats the onboard information infrastructure as a part of the communication path between the spacecraft control computer and each onboard component. MMX shall have onboard SpaceWire networks as the information infrastructure, which shall be designed according to the MMX network design guideline (TBD-sys) and the MMX telemetry and commands design guideline (TBD-sys), which shall be inherited from JERG-2-432 "SpaceWire onboard subnetwork design guideline"([A20]).

TBD-Sys

4. SPACE LINK PROTOCOL ARCHITECTURE

4.1. General

This chapter specifies managed parameters required for usage of the Space Link Protocol Architecture ([A4]).

TBD-Sys

4.2. TC Space Data Link Protocol

The managed parameters of the TC Space Data Link Protocol (TC SDLP) [A3] are specified by tables in this section.

TBD-Sys

4.3. Communications Operation Procedure-1

The managed parameters of the Communications Operation Procedure-1 (COP-1) [A13] are specified by tables in this section.

TBD-Sys

4.4. TC Synchronization and Channel Coding

Managed parameters of the TC Synchronization and Channel Coding (TC SCC) [A14] are specified in this section.

TBD-Sys

4.5. AOS Space Data Link Protocol

The managed parameters of the AOS Space Data Link Protocol (AOS SDLP) [A15] are specified by tables in this section.

TBD-Sys

4.6. TM Synchronization and Channel Coding

Table 4-14 (corresponding to Table 11-1, 11-2, 11-3, 11-4, 11-5, and 11-6 of [A16]) specifies the managed parameters of the TM Synchronization and Channel Coding (TM SCC) [A16].

TBD-Sys

5. SPACE LINK MODULATION METHODS

This chapter specifies managed parameters required for usage of the Space Link Modulation Methods ([A5]).

TBD-Sys

6. ONBOARD PROTOCOL ARCHITECTURE

This chapter specifies managed parameters required for usage of the Onboard Protocol Architecture ([A6]) and the SpaceWire Onboard Subnetwork Design Standards ([A20]).

TBD-Sys

7. GROUND PROTOCOL ARCHITECTURE

This chapter specifies managed parameters required for usage of the Ground Protocol Architecture ([A7]).

TBD-Sys

8. COMMON FUNCTIONS

This chapter specifies managed parameters required for usage of the Common Functions ([A8]).

8.1. Timeline-command function

TBD-Sys

8.2. Macro-command function

TBD-Sys

9. TIME MANAGEMENT

This chapter specifies managed parameters required for usage of the Time Management ([A9]).

9.1. Spacecraft Monitor and Control Protocol

TBD-Sys

10. DOCUMENTS

10.1. Applicable documents

[A1] JAXA, “Functional Model of Spacecraft (FMS),” GSTOS 201, draft version (See appendix-1 of this document).

[A2] JAXA, “Standard Communications and Data Handling Architecture, Part 1: General (SCHDA1),” SCHDA 110, To be issued.

[A3] JAXA, “Standard Communications and Data Handling Architecture, Part 2: End-to-End Protocol Architecture (SCHDA2),” SCHDA 120, To be issued.

[A4] JAXA, “Standard Communications and Data Handling Architecture, Part 3: Space Link Protocol Architecture (SCHDA3),” SCHDA 130, To be issued.

[A5] JAXA, “Standard Communications and Data Handling Architecture, Part 4: Space Link Modulation Methods (SCHDA4),” SCHDA 140, issue 0.0a, To be issued.

[A6] JAXA, “Standard Communications and Data Handling Architecture, Part 5: Onboard Protocol Architecture (SCHDA5),” SCHDA 150, To be issued.

[A7] JAXA, “Standard Communications and Data Handling Architecture, Part 6: Ground Protocol Architecture (SCHDA6),” SCHDA 160, issue 0.0a, To be issued.

[A8] JAXA, “Standard Communications and Data Handling Architecture, Part 7: Common Functions (SCHDA7),” SCHDA 170, To be issued.

[A9] JAXA, “Standard Communications and Data Handling Architecture, Part 8: Time Management (SCHDA8),” SCHDA 180, To be issued.

[A10] JAXA, “Spacecraft Monitor and Control Protocol (SMCP),” GSTOS 200, draft version (See appendix-2 of this document).

[A11] CCSDS, “Space Packet Protocol,” CCSDS 133.0-B-1, September 2003, To be issued.

[A12] CCSDS, “TC Space Data Link Protocol,” CCSDS 232.0-B-3, September 2015, To be issued.

[A13] CCSDS, “Communications Operation Procedure-1,” CCSDS 232.1-B-2, September 2010, To be issued.

[A14] CCSDS, “TC Synchronization and Channel Coding,” CCSDS 231.0-B-2, September 2010, To be issued.

[A15] CCSDS, “AOS Space Data Link Protocol,” CCSDS 732.0-B-3, September 2015, To be issued.

[A16] CCSDS, “TM Synchronization and Channel Coding,” CCSDS 131.0-B-2, August 2011, To be issued.

[A17] CCSDS, “Radio Frequency and Modulation Systems – Part 1: Earth Stations and Spacecraft,” CCSDS 401.0-B-25, February 2015, To be issued.

[A18] CCSDS, “Time Code Formats,” CCSDS 301.0-B-4, November 2010, To be issued.

[A19] JAXA, ＲＦ回線設計標準, JERG-2-420B, 20 May, 2016, To be issued.

[A20] MMX SpaceWire Onboard Subnetwork Design Standards, JAXA-RPR-MX16309 (as a draft of English version of JERG-2-432)

[A21]宇宙航空研究開発機構 宇宙科学研究所 衛星運用データ利用センター, “DIOSA（Distributed Operations System Architecture）インタフェース仕様 : 宇宙データ転送プロトコル（SDTP）,” OSO 501, To be issued.

[A22]CCSDS, “Cross Support Reference Model – Part 1: Space Link Extension Service,” CCSDS 910, To be issued.

10.2. Reference documents

[R1] JAXA, 通信設計標準利用ガイドライン（テレコマンドデータリンクプロトコル編）, JERG-2-400-HB001A, 29 March 2013, To be issued.

[R2] JAXA, 通信設計標準利用ガイドライン（AOS データリンクプロトコル編）, JERG-2-400-HB002A, 17 March 2014, To be issued.

[R3] JAXA, 通信設計標準利用ガイドライン（スペースコミュニケーション・エンドツーエンドプロトコル編）, JERG-2-400-HB003, 10 May 2012, To be issued.

11. ACRONYMS

COP	Communications Operation Procedure
FMS	Functional Model of Spacecraft
GSTOS	Generic Spacecraft Test and Operation Software
ISAS	Institute of Space and Astronautical Science
JAXA	Japan Aerospace Exploration Agency
SCC	Synchronization and Channel Coding
SDLP	Space Data Link Protocol
SIB2	Spacecraft Information Base 2
SMCP	Spacecraft Monitor and Control Protocol

APPENDIX-1 FUNCTIONAL MODEL OF SPACECRAFT (FMS)

Functional Model of Spacecraft (FMS)

DRAFT

GSTOS 201-0.10e
Issue 0.10e
17 Oct 2016

Japan Aerospace Exploration Agency (JAXA)

CONTENTS

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to specify a method of modeling functions of spacecraft and their onboard instruments (hereafter, the term spacecraft is used to mean an entire spacecraft and/or its onboard instruments). The model specified in this document is to be used for specifying functions of spacecraft, but it is also a guideline for functional design of spacecraft in the sense that spacecraft functions should be designed in such a way that they can be specified with this model.

The purpose of establishing this model is to facilitate description and understanding of spacecraft functions and to enable electronic management of information on spacecraft functions by using a standard method for specifying functions of any onboard instrument of any spacecraft. Furthermore, this model helps to promote reuse of onboard instruments by standardizing methods of functional design.

The spacecraft functions specified with this model can be registered and managed in the Spacecraft Information Base 2 (SIB2) [R1]. The spacecraft specified with this model can be monitored and controlled by the Generic Spacecraft Test and Operations Software (GSTOS) [R2] using the Spacecraft Monitor and Control Protocol [R3].

This document is a part of the standards of the Japan Aerospace Exploration Agency (JAXA). Against the custom of the JAXA's standards, this document is described in English in order to encourage international collaborations in space science project.

[Note] Handbook written in Japanese may be available in the future.

1.2. SCOPE

This document specifies the Functional Model of Spacecraft in terms of how spacecraft functions should be specified. It does not specify how spacecraft functions should be designed or implemented on any particular spacecraft.

1.3. APPLICABILITY

This document applies to development of spacecraft functions for those projects that have decided to use the full capability of the Spacecraft Information Base 2 (SIB2) [R1] and the Generic Spacecraft Test and Operations Software (GSTOS) [R2].

1.4. REFERENCES

1.4.1. Applicable Documents

None.

1.4.2. Reference documents

[R1] Takahiro Yamada and Keiichi Matsuzaki, "Definition of Spacecraft Information Base 2 (DSIB2)," GSTOS 300, latest issue.

[R2] Takahiro Yamada, "Generic Spacecraft Test and Operations Software (GSTOS), Development Plan," GSTOS 100, under development.

[R3] Takahiro Yamada, "Spacecraft Monitor and Control Protocol (SMCP)," GSTOS 200, latest issue.

1.5. DOCUMENT STRUCTURE

This document is organized as follows:

a) Section 1 (this section) shows the purpose, scope, and applicability of the document, and lists the references and notations used throughout the document;

b) Section 2 presents an overview of the Functional Model of Spacecraft;

c) Section 3 defines the Functional Object, which is the central concept of the Functional Model of Spacecraft;

d) Section 4 defines the Memory Functional Class, which is to be used as a template for defining Functional Objects that represent onboard memories of various sorts;

e) Appendix A lists all acronyms used in this document; and

f) Appendix B shows an example of a Functional Object.

1.6. NOTATIONS

The following notations apply throughout this document.

A paragraph after [Example] (or [Example *n*], where *n* is a positive integer) shows an example that helps to understand the specification, which is not part of the specification.

A paragraph after [Temporary Note] (or [Temporary Note *n*], where *n* is a positive integer) is an editorial note that will be removed from the final document.

A paragraph after [Note] (or [Note *n*], where *n* is a positive integer) contains an informative note that helps to understand the specification, which is not part of the specification.

2. OVERVIEW

This section provides an informative overview of the Functional Model of Spacecraft to help the readers to understand the specification of the model given in the subsequent sections.

2.1. PURPOSE OF THIS MODEL

A model is a framework with which to represent something from a certain point of view. This document specifies a framework (hereafter called the Functional Model of Spacecraft) with which to represent spacecraft from a functional point of view.

A function of a spacecraft, in this model, is an abstract representation of a job of the spacecraft in its orbit such as performing an observation or an experiment. This abstraction is made by focusing on what can be seen from outside the spacecraft (that is, what the spacecraft performs) and by leaving out how the job is performed inside the spacecraft.

The functional model specified in this document is for expressing the functions of spacecraft. If spacecraft functions are designed in such a way that they can be expressed with this model, the methods for functional design of spacecraft will be unified. Therefore, this model also services as a guideline for functional design of spacecraft. In other words, this model also aims at standardizing the methods for functional design of spacecraft in terms of how the spacecraft are operated from the outside world.

2.2. FUNCTIONAL OBJECT

Since the functions that a spacecraft has are generally complex, they are usually specified as groups of functions according to their characteristics. Such a group of functions is called a Functional Object in this model. The Functional Object is the central concept in the Functional Model of Spacecraft.

A Functional Object is used to describe how a particular job of a spacecraft can be seen from the outside of it. The concept of Functional Object is based on the concept of object used in object-oriented programming. In object-oriented programming, a program is defined as a class. During execution of the program, an instance (which is also called an object) is dynamically generated based on the definition of the class, and the instance actually executes the program. In the case of spacecraft, however, the concept of object-oriented programming cannot be directly applied because a spacecraft is hardware, not software (although software maybe used in some parts of the spacecraft). In this model, the functions of a spacecraft are assumed to exist permanently (although whether each function is executable or not at a particular time depends on the status of the spacecraft at that time), and a group of functions is specified as a Functional Object, which corresponds to an instance in software terminology but exists permanently. If two instruments that perform the same job are installed on a spacecraft, they are defined as two distinct Functional Objects. In this case, the definition of these two Functional Objects can be specified as a template for generating these Functional Objects. A template for generating multiple Functional Objects is called a Functional Class.

Functional Objects can be monitored and controlled by other Functional Objects and/or the outside of the spacecraft (for example, spacecraft control systems and/or spacecraft test systems). This model does not specify methods for communications between Functional Objects and the outside world (other Functional Objects and/or the outside of the spacecraft) (see also 2.4).

Functional Objects are specified using such notions as Attributes, Operations, Events, Behavior, and Diagnostic Rules.

Attributes of a Functional Object are parameters representing the status of the Functional Object at each point of time, and their values can be provided to the outside of the Functional Object. The values of some Attributes can be set by the outside world through invocation of Operations.

Operations of a Functional Object are actions performed by the Functional Object and must be invoked by the outside world.

Events of a Functional Object are occurrences of things that have particular significances. Occurrences of Events are detected by the Functional Object itself. Occurrences of Events that are important to the outside world can be reported to the outside world, and reports of occurrences of important Events are called Alerts.

The behavior of a Functional Object describes rules about correct actions taken by the Functional Object, and is used to determine correct sequences of Operations and whether Attributes are valid or not, etc. The behavior of a Functional Object can be specified with a set of State Machines.

Diagnostic rules of a Functional Object are those used by the outside world (other Functional Objects and/or the outside of the spacecraft) to determine whether the Functional Object is functioning correctly or not.

2.3. SPACECRAFT INFORMATION BASE 2

The definition of Functional Objects specified with the method specified in this document can be registered in the Function Definition Part of the Spacecraft Information Base 2 (SIB2) [R1]. The Spacecraft Information Base 2 is a tool for managing information on spacecraft electronically and it is recommended that the functional information on spacecraft should be managed uniformly using this database throughout all the phases of spacecraft design, testing and flight operations.

2.4. COMUNICATIONS BETWEEN FUNCTIONAL OBJECTS AND THE OUTSIDE WORLD

Functional Objects communicate with the outside world (other Functional Objects and/or the outside of the spacecraft), but methods for the communications between Functional Objects and the outside world are not specified by this model. Methods for performing the following are needed and it is recommended that the Spacecraft Monitor and Control Protocol (SMCP) [R3] should be used if a space link is used to perform the following.

a) Invoke Operations of Functional Objects by the outside world;

b) Report values of Attributes of Functional Objects to the outside world; and

c) Report Alerts (occurrence of important Events) to the outside world.

3. FUNCTIONAL OBJECTS

3.1. GENERAL

Functions of spacecraft shall be specified with a unit called the Functional Object. The Functional Object is a unit for specifying spacecraft functions and represents a group of functions that are easy to be specified or understood when grouped together as a single unit. Functional Objects are abstract representations of spacecraft functions in the sense that spacecraft functions are represented only in terms of how they are seen from the outside world.

A Functional Object can contain one or more other Functional Objects.

An entire spacecraft is represented by a Functional Object, which contains multiple Functional Objects, each of which may contain one or more Functional Objects. Any Functional Object of a spacecraft is contained in the Functional Object that represents the entire spacecraft either directly or indirectly. Functional Objects contained in a Functional Object are called child Functional Objects of the Functional Object that contains them. The Functional Object that contains child Functional Objects is called the parent Functional Object of the child Functional Objects.

Functional Objects (except for the Functional Object representing the entire spacecraft) are valid or invalid depending on the occasion. When A Functional Object is invalid, it is suspending all of its functions. Whether a Functional Object is valid or invalid at a particular time is determined by the value of a Logical Expression: logical expression (an expression that has true or false as its value) for the value(s) of one or more Attributes. For Logical Expressions, the following operations and their combinations can be used: “is equal to”, “greater than”, “greater than or equal to”, “smaller than”, “smaller than or equal to”, “NOT”, “AND”, “OR”, and “XOR”.

Functional Objects have Attributes and Operations, through which Functional Objects are monitored and controlled from outside of them (other Functional Objects and/or the outside of the spacecraft). Attributes of a Functional Object are parameters that represent the status of the Functional Object at each point of time, each of which takes a value from a specified set. Operations of a Functional Object are actions performed by the Functional Object.

Things that occur in a Functional Object and have particular significances for that Functional Object and/or the outside world (i.e., other Functional Objects and/or the outside of the spacecraft) are called Events. Functional Objects can report to the outside world occurrences of Events that are important to the outside world. Such reports of occurrences of Events are called Alerts.

Functional Objects have Behaviors (rules about how they behave), and they are specified with State Machines.

Rules for the outside world (other Functional Objects and/or the outside of the spacecraft) to determine whether a Functional Object is functioning correctly or not can be specified, and they are called Diagnostic Rules.

This document does not specify methods for the outside world (other Functional Objects and/or the outside of the spacecraft) to communicate with Functional Objects (see also 2.4).

If the same or similar Functional Objects are used in one or multiple spacecraft, the common features of these Functional Objects can be specified as a template. Templates for Functional Objects specified in this way are called Functional Classes. A Functional Class specifies Attributes, Operations, Events, Alerts, Behavior, and Diagnostic Rules that are commonly used by multiple Functional Objects. Individual Functional Objects can be defined by using the Functional Class specification as it is or by adding additional features (such as additional Attributes, additional Operations, and so on) to the Functional Class specification.

This document specifies the following standard Functional Class:

1) Memory Functional Class.

3.2. ATTRIBUTES

Attributes of a Functional Object are parameters that represent the status of the Functional Object at each point of time, each of which takes a value from a specified set.

Some Attributes have discrete values (such as the current mode of an instrument), and some Attributes have continuous values (such as a temperature). Some Attributes take values of simple data types (such as an enumeration or an integer), and some Attributes take values of complex data types (such as an array or a record).

[Temporary Note] Complex data types are not implemented in the tools of SIB2/GSTOS yet although its proxy is in the investigation.

The values of some Attributes can be set by the outside of the Functional Object (other Functional Objects and/or the outside of the spacecraft), but the values of some Attributes cannot. When values are set to Attributes from the outside, Operations (see 3.3) shall be invoked. Whether the value of an Attribute has been set correctly or not from the outside can be verified by inspecting the value of the Attribute.

Attributes are valid or invalid depending on the occasion. Some Attributes of a Functional Object are valid whenever the Functional Object is valid. For some Attributes, whether an Attribute is valid or invalid at a particular time is determined by the value of a Logical Expression. When an Attribute is invalid, its value does not have any significance and cannot be set by the outside world (the Functional Object ignores such requests received from the outside world).

Attributes may have particular values when they become valid. These values are called initial values of these Attributes.

For an Attribute with a continuous value, Action Limit: a pair of limit values (upper and lower limits) can be specified indicating that the Functional Object will be in danger (for example, that the Functional Object will cease functioning) if the value of the Attribute exceeds its upper limit or falls below its lower limit. For such an Attribute, Caution Limit: another pair of limit values (upper and lower limits) can be specified indicating that caution is needed for the Functional Object (for example, that there is a possibility that the Functional Object will cease functioning) if the value of the Attribute exceeds its upper limit or falls below its lower limit. Furthermore, Valid Range: a pair of limit values can be specified for each Attribute with a continuous value to show that the value of the Attribute has a meaning only when the value is inside that range (because of the physical limitation of the sensor that measures the value of the Attribute, for example).

[Temporary Note] Valid Range is not implemented in the tools of SIB2/GSTOS.

For an Attribute with a discrete value, two sets of values can be specified; that is, (1) Action: a set of values that indicate that the Functional Object will be in danger if the value of the Attribute is one of them and (2) Caution: another set of values that indicate that caution is needed for the Functional Object if the value of the Attribute is one of them.

These limit values are used by the entity that monitors and controls the Functional Object (other Functional Objects and/or the outside of the spacecraft) to determine whether that Functional Object is functioning correctly or not.

3.3. OPERATIONS

Operations of a Functional Object are actions performed by the Functional Object. Operations include those used for setting values to one or more Attributes. Here, Operation which does not accompany any changes is called No Operation (NOP).

Operations are invoked by receiving signals from the outside of the Functional Object (i.e., other Functional Objects and/or the outside of the spacecraft).

Generally, as a result of executing an Operation, the values of one or more Attributes will change. The state of the Functional Object (see 3.5) may also change, in which case the value of the Attribute that shows the current state will change.

[Note] Functional Objects should be designed in such a way that the values of some Attributes change as a result of executing an Operation except for cases in which it is physically impossible to do so due to design constraints.

Whether an Operation has been executed correctly or not can be verified by inspecting whether the values of the Attributes that are supposed to change as a result of executing that Operation have actually changed or not.

An Operation can be executed only when (1) the Functional Object is in one or more specific States of a State Machine (see 3.5) or (2) the value of a Logical Expression is true. For the case (1), the Operation appears as a trigger of one or more State Transitions (see 3.5) and can be executed only when the Functional Object is in a State that has a State Transition departing that State and having the Operation as a trigger.

If a Functional Object has received a signal to execute an Operation when the condition for executing that Operation described above is not met, the signal is ignored by the Functional Object.

For Operations, their criticality levels can be specified (for example, whether an Operation is hazardous or not).

When invoking an Operation, one or more parameters can be specified. These parameters can be used to (1) contain values to be set to Attributes or (2) to specify detailed information on how the Operation should be executed.

3.4. EVENTS

Events of a Functional Object are things that occur in the Functional Object and have particular significances for the Functional Object and/or the outside of the Functional Object (i.e., other Functional Objects and/or the outside of the spacecraft).

An Event occurs when a condition specified for it is met. For Events, explicit conditions may or may not be specified. If an explicit condition is specified for an Event, it is specified by a Logical Expression. An Event occurs when the value of the Logical Expression specified for it becomes true.

Events are used as triggers of State Transitions (see 3.5) and/or triggers of Alerts (see below).

Functional Objects can report to the outside of the Functional Object (i.e., other Functional Objects and/or the outside of the spacecraft) occurrences of Events that are important to the outside world. Such reports of occurrences of Events are called Alerts.

Alerts may also contain the values of one or more Attributes that are relevant to the Events and/or one or more parameters to explain the Events in detail.

Alerts are conveyed to the outside of the Functional Object by some signals.

[Note] The outside world may be able to detect the occurrence of an Event by monitoring the values of some Attributes periodically, but the Alert is used to report the occurrence of an Event more actively and promptly because the values of Attributes may not be delivered to the outside world frequently enough.

3.5. BEHAVIOR

Functional Objects have Behaviors (rules about how they behave), which are specified with State Machines.

In a State Machine, all the States are connected by some State Transitions. That is, any State is reachable from at least one other State in the State Machine.

A State Transition is triggered by the execution of an Operation, the occurrence of an Event, or some internal activity.

Which State of a State Machine the Functional Object is currently in is indicated by the value of an Attribute. Such Attributes are called State Attributes. A State Attribute takes as its values the names of all the States of the State Machine that it represents.

When a State Transition occurs, the value of the associated State Attribute changes. The values of some other Attributes may change as a result of the State Transition.

State Machines are valid or invalid depending on the occasion. Some State Machines are valid whenever the Functional Object is valid. For some State Machines, whether a State Machine is valid or not is determined by whether the State Attribute associated with it is valid or not.

A Functional Object is in a State in each of the State Machines that are valid at that time.

If the Functional Object is always in a specific State of a State Machine when the State Machine becomes valid, that State is called the Initial State. Each State Machine has either one Initial State or no Initial State. In a State Transition Diagram that does not have an Initial State, it cannot be determined which of the Sates the Functional Object is in when the State Machine becomes valid.

Each of the States in a State Machine can have its criticality level (that is, whether the Functional Object is dangerous or needs caution when the Functional Object is in that State).

For each State Transition, the maximum time that the Transition can take can be specified. If a State Transition is started by an Operation and the State Transition is not complete within the maximum transition time, the Functional Object is diagnosed by the outside world (i.e., other Functional Objects and/or the outside of the spacecraft) as not functioning correctly. Likewise, the minimum time that the Transition can take can be specified for each State Transition. If a State Transition is started by an Operation and the State Transition is complete within the minimum transition time, the Functional Object is diagnosed by the outside world as not functioning correctly.

[Note] The State that the Functional Object is currently in determines the basic Operations that can be invoked at that time. Therefore, the States determines the correct behavior of the Functional Object at each specific time.

3.6. DIAGOSTIC RULES

The outside world (other Functional Objects and/or the outside of the spacecraft) can determine whether a Functional Object is functioning correctly or not, using the Diagnostic Rules specified for that Functional Object.

A Diagnostic Rule is specified as a logical expression for the values of some of the Attributes that the Functional Object has (including those that its child and descendant Functional Objects have). If the value of the logical expression is true, the Functional Object is diagnosed to be functioning normally, and if not, the Functional Object is diagnosed to be abnormal.

For each Diagnostic Rule, a text message containing additional information on the diagnosis (for example, the level of abnormality and/or methods for handling the abnormality) can be specified.

[Note 1] In case a Functional Object can be diagnosed by checking whether the value of an Attribute exceeds an upper limit or falls below a lower limit, this Diagnostic Rule shall be specified as limit values for that Attribute (see 3.2).

[Note 2] When a Functional Object detects an anomaly in itself, it can report its occurrence to the outside world by issuing an Alert (see 3.4). Diagnostic Rules are used by the outside world to detect anomalies in a Functional Object for cases in which it is not easy for the Functional Object itself to detect the anomalies.

[Temporary Note] It is a future study item to investigate how to refer, in Diagnostic Rules, to the values of Attributes not belonging to the spacecraft (for example, the values of Attributes of ground station equipment).

3.7. OTHER FEATURES

Functional Objects can generate data that are not specified as Attribute values in the Functional Object definitions as a result of executing its functions. For example, data showing results of observations or experiments (like images) can be generated by Functional Objects. Functional Objects may also refer to the values of A ttributes that other Functional Objects have. Methods for sending and receiving these data items should be specified as part of the protocol that supports communications with the outside world, like the Spacecraft Monitor and Control Protocol (SMCP) [R3] (see also 2.4).

4. MEMORY FUNCTIONAL CLASS

4.1. GENERAL

A Memory Functional Object is a Functional Object that represents a memory device (a device that stores data). Memory Functional Objects are a kind of Functional Objects but they shall have the Operations and Attributes specified below. Each Memory Functional Object may have (1) other Operations and Attributes in addition to those specified below and (2) some other characteristics of Functional Objects (such as Events and Alerts).

The Memory Functional Class is the set of all Memory Functional Objects and specifies the Operations and Attributes that any Memory Functional Object shall have.

4.2. OPERATIONS

Any Memory Functional Object shall have the following Operations. The parameters for these Operations are shown in parentheses.

1) MemoryLoad (StartAddress, MemoryData); and

2) MemoryDump (NoOfDumps, StartAddress, Length).

The Operation MemoryLoad is an Operation for uploading data to the memory. The value of the parameter MemoryData is written into the memory from the address specified by the StartAddress parameter.

For each Memory Functional Object, maximum size (e.g. 256 octets) of the MemoryData can be specified. Moreover, alignment restriction for the StartAddress and size of the MemoryData can be specified: either 1 octet (i.e. any address), 2 octets, 4 octets, or the maximum size of the MemoryData. Here, size of MemoryData and Length shall be multiple of the alignment size. If the alignment size is ‘the maximum size of the MemoryData’, size of MemoryData is always ‘the maximum size of the MemoryData’.

The Operation MemoryDump is an Operation for directing the Memory Functional Object to dump memory data. Upon receiving this direction, the Memory Functional Object dumps memory data of the length (number of octets) specified by the Length parameter, starting with the address specified by the StartAddress parameter, for the number of times specified by the NoOfDumps parameter.

[Note] The method and data formats for invoking these Operations and dumping data are specified by a communications protocol that performs communications between Functional Objects and the outside world, such as the Spacecraft Monitor and Control Protocol (SMCP) [R3].

4.3. ATTRIBUTES

[Temporary Note] The specification of this sub-section is not complete. A future issue of this document will provide a complete list of attributes that Memory Functional Objects shall have.

Any Memory Functional Object shall have the following Attributes. The values of these Attributes shall be fixed at the time of the design of the Memory Functional Object and are not resettable or rewriteable.

1) FirstAddress;

2) LastAddress;

3) …

The FirstAddress and LastAddress specify the first and last addresses of the memory area with which the MemoryLoad and MemoryDump Operations are possible.

APPENDIX A ACRONYMS

GSTOS	Generic Spacecraft Test and Operations Software
NOP	No Operation
SIB	Spacecraft Information Base
SMCP	Spacecraft Monitor & Control Protocol
XOR	Exclusive OR

APPENDIX B AN EXAMPLE OF A FUNCTIONAL OBJECT

In this Appendix, a Functional Object called X_A is presented as an example of a Functional Object. This Functional Object models basic functions of controlling a simple component.

B.1 FUNCTIONAL OBJECT

The Functional Object X_A is contained in a Functional Object called X_SubSystem that represents a Sub-System X. It specifies the functions of the component X_A contained in Sub-System X.

The Functional Object X_A contains other Functional Objects called X_A1 and X_A2, each of which represents a set of functions performed by the component X_A. The Functional Object X_A specifies the functions that apply to the entire component.

The Functional Object X_A is valid only when some Attributes of the parent Functional Object X_SubSystem meet a certain condition. The children Functional Objects X_A1 and X_X2 are valid only when the value of the State Attribute X_A_RunStop is RUN (that is, the Functional Object X_A is in the State RUN) (see B.5).

The Functional Object X_A has Attributes and Operations for monitoring and controlling the entire component X_A. It has a behavior that represents the rules concerning its actions. It also has an Event and issues an Alert.

B.2 ATTRIBUTES

The Functional Object X_A has the following four Attributes:

1) X_A_OnOff;
2) X_A_RunStop;
3) X_A_ErrorStatus; and
4) X_A_CheckMode.

The values of the Attributes X_A_OnOff and X_A_RunStop represent the states the Functional Object is in at that time. Therefore, they are State Attributes.

Of these Attributtes, only the value of the Attribute X_A_CheckMode can be set from the outside of the Functional Object.

The Attribute X_A_OnOff is valid whenever the Functional Object is valid, but the Attributes X_A_RunStopX, A_ErrorStatuss, and X_A_CheckMode are valid only when the value of the State Attribute X_A_OnOff is ON (that is, the Functional Object is in the State ON) (see B.5).

B.3 OPRATIONS

The Functional Object X_A has the following five Operations:

1) X_A_On;
2) X_A_Start;
3) X_A_Stop;
4) X_A_Off; and
5) X_A_SetCheckMode.

The Operation X_A_SetCheckMode is used to set a value to the Attribute X_A_CheckMode and has a parameter that contains the value to be set.

These Operations can be executed when the following conditions are met. The Operations that trigger State Transitions appear as triggers of State Transitions in the State Machine shown in Figure B-1.

1) X_A_On X_A_OnOff=OFF;
2) X_A_Start X_A_OnOff=ON AND X_A_RunStop=STOP;
3) X_A_Stop X_A_OnOff=ON AND X_A_RunStop=RUN;
4) X_A_Off X_A_OnOff=ON; and
5) X_A_SetCheckMode X_A_OnOff=ON.

B.4 EVENTS

The Functional Object X_A has the following one Event:

1) X_A_ErrorDetect.

This Event occurs when the following condition is met:

NOT (X_A_ErrorStatus=NORMAL)

The Functional Object X_A issues the following one Alert:

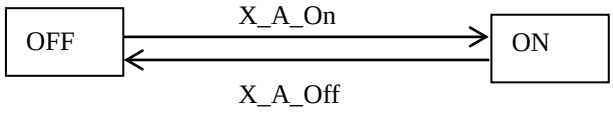
1) X_A_ErrorDetected.

This alert is used to report to the outside world an occurrence of the Event X_A_ErrorDetect. It has the value of the Attribute X_A_ErrorStatus as a parameter.

B.5 BEHAVIOR

The Behavior of the Functional Object X_A is specified with two State Machines X_A_OnOff and X_A_RunStop. These State Machines are shown in Figure B-1. The triggers are shown with the arrows that show State Transitions.

State Machine X_A_OnOff



State Machine X_A_RunStop

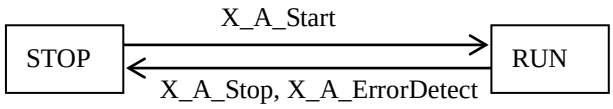


Figure B-1: State Machines of Functional Object X_A

The State Machine X_A_OnOff has two States OFF and ON. The Initial State is OFF. The current State of this State Machine is indicated by the State Attribute X_A_OnOff. This State Machine is valid whenever the Functional Object is valid.

The State Machine X_A_RunStop has two States STOP and RUN. The Initial State is STOP. The current State of this State Machine is indicated by the State Attribute X_A_RunStop. This State Machine is valid only when the value of X_A_OnOff is ON (that is, when the Functional Object is in the State ON).

APPENDIX-2 SPACECRAFT MONITOR AND CONTROL PROTOCOL (SMCP)

Spacecraft Monitor and Control Protocol (SMCP)

DRAFT

GSTOS 200-0.14j
Issue 0.14j
20 Oct 2016

Japan Aerospace Exploration Agency (JAXA)

CONTENTS

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to specify (1) a method of constructing the contents of commands and telemetry (hereafter called Command and Telemetry Messages) for monitoring and controlling spacecraft and their onboard instruments (hereafter, the term spacecraft is used to mean an entire spacecraft and/or its onboard instruments) and (2) transmission sequences of Command and Telemetry Messages.

This protocol assumes that the functions of spacecraft are specified according to the Functional Model of Spacecraft (FMS) [A1]. This protocol specifies (1) a method for constructing Command and Telemetry Messages to monitor and control Functional Objects specified in FMS from the outside of them (that is, from other Functional Objects or from outside of the spacecraft) using communications lines, and (2) transmission sequences of these Messages.

This protocol is located at a layer above the layer for transferring telemetry and commands over communications lines (for example, the Space Packet Protocol [R5]) and is transferred between a spacecraft and the ground, within a spacecraft, and on the ground using lower layer transmission protocols.

[Note 1] This protocol assumes that typical radio frequency links used between spacecraft and the ground (that is, those with low data rates and intermittent connectivity) are used as communications lines. A variation of this protocol may be developed at a later time if this protocol is to be used over communications media with different characteristics (for example, high-speed data buses or shared memories).

[Note 2] This protocol can be used on top of any transport protocol, but if the Standard Communications and Data Handling Architecture (SCDHA) [R1] is used, this protocol shall be used as an end-to-end protocol on top of the other standard protocols as specified in [R1] and [R2].

The purpose of establishing this protocol is to enable project to develop (1) onboard software, (2) software used on the ground for testing and operations of spacecraft, and (3) databases that store information on spacecraft, efficiently by standardizing construction rules for Command and Telemetry Messages and their transmission sequences across all spacecraft and all onboard instruments.

The Generic Spacecraft Test and Operations Software (GSTOS) [R3] assumes that the spacecraft are developed using FMS [A1] and this protocol. The formats of individual Messages defined according to the specifications of this document are stored and managed by the Spacecraft Information Base 2 (SIB2) [R4].

This document is a part of the standards of the Japan Aerospace Exploration Agency (JAXA). Against the custom of the JAXA's standards, this document is described in English in order to encourage international collaborations in space science project.

[Note] Handbook written in Japanese may be available in the future.

1.2. SCOPE

This document specifies the Spacecraft Monitor and Control Protocol in terms of the interface for message exchanges between elements that monitor and control Functional Objects and the Functional Objects that are monitored and controlled. It does not specify how this protocol should be implemented by hardware or software.

1.3. APPLICABILITY

This document applies to development of spacecraft and their ground systems for those projects that have decided to use the full capability of the Generic Spacecraft Test and Operations Software (GSTOS) [R3] and the

Spacecraft Information Base 2 (SIB2) [R4].

1.4. REFERENCES

1.4.1. Applicable Documents

[A1] Takahiro Yamada, “Functional Model of Spacecraft (FMS),” GSTOS 201, latest issue.

[A2] CCSDS, “Time Code Formats,” CCSDS 301.0-B-4, November 2010

[A3] ISO, Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model. International Standard, ISO/IEC 7498-1. 2nd ed., 1994.

1.4.2. Reference documents

[R1] Takahiro Yamada, “Standard Communications and Data Handling Architecture, Part 1: General (SCHDA1),” SCHDA 110, latest issue.

[R2] Takahiro Yamada, “Standard Communications and Data Handling Architecture, Part 2: End-to-End Protocol Architecture (SCHDA2),” SCHDA 120, latest issue.

[R3] Takahiro Yamada, “Generic Spacecraft Test and Operations Software (GSTOS), Development Plan,” GSTOS 100, under development.

[R4] Takahiro Yamada and Keiichi Matsuzaki, “Definition of Spacecraft Information Base 2 (DSIB2),” GSTOS 300, latest issue.

[R5] CCSDS, “Space Packet Protocol,” CCSDS 133.0, latest issue.

[R6] 通信設計標準, JERG-2-400A, 29 March 2013

[R7] 通信設計標準利用ガイドライン（スペースコミュニケーション・エンドツーエンドプロトコル編）, JERG-2-400-HB003, 10 May 2012

1.5. DOCUMENT STRUCTURE

This document is organized as follows:

a) Section 1 (this section) shows the purpose, scope, and applicability of the document, and lists the references and notations used throughout the document;

b) Section 2 presents an overview of the Spacecraft Monitor and Control Protocol;

c) Section 3 specifies concepts and rules that apply to both SMCP Command and SMCP Telemetry;

d) Section 4 specifies the functions and information items of SMCP Command;

e) Section 5 specifies the functions and information items of SMCP Telemetry;

f) Appendix A lists all acronyms used in this document; and

g) Appendix B summarizes information on interactions between SMCP Command and SMCP Telemetry.

1.6. DEFINITIONS AND NOTATIONS

1.6.1. Definitions from the Open Systems Interconnection (OSI) Basic Reference

This document makes use of the following terms defined in the “Open Systems Interconnection (OSI) Basic

Reference” [A3]:

a) Protocol Control Information (PCI).

[Note] Protocol Control Information is information to control protocol which usually included in data structure called header.

1.6.2. Definitions from the Functional Model of Spacecraft

This document makes use of the following terms defined in the “Functional Model of Spacecraft” [A1]:

a) Alert;

b) Attribute;

c) Event;

d) Functional Object;

e) MemoryDump;

f) Memory Functional Object;

g) MemoryLoad;

h) No Operation (NOP); and

i) Operation.

1.6.3. Terms defined in this document

The following definitions apply throughout this document.

Combined Upper Functional Object and Route Identifier (CUFORID): an identifier to express a group of Functional Objects and Route to the group of the Functional Objects.

Command Message: a message sent from a Controller to a Target to control the Target.

Controller: an entity that monitors and controls Targets.

Functional Object Identifier (FOID): an identifier of a Functional Object.

Lower Functional Object Identifier (Lower FOID): an identifier of a Functional Object within a group of Functional Objects.

Management System: a system which exchanges information out side of the Protocol (term used in [R5]).

Message: the unit of data exchanged between a Controller and a Target.

Target: an entity that is monitored and controlled by a Target.

Telemetry Message: a message sent from a Target to a Controller to be monitored by the Controller.

Upper Functional Object Identifier (Upper FOID): an identifier to express a group of Functional Objects.

Project: a project (refer ISO 9000 for example) that develops onboard and ground communications and data handling systems for spacecraft.

1.6.4. Notations

The following notations apply throughout this document.

A paragraph after [Example] (or [Example *n*], where *n* is a positive integer) shows an example that helps to understand the specification, which is not part of the specification.

A paragraph after [Temporary Note] (or [Temporary Note *n*], where *n* is a positive integer) is an editorial note that will be removed from the final document.

A paragraph after [Note] (or [Note *n*], where *n* is a positive integer) contains an informative note that helps to understand the specification, which is not part of the specification.

1.7. CONVENTION

In this document, the following convention is used to identify each bit in an N-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be ‘Bit 0’; the following bit is defined to be ‘Bit 1’ and so on up to ‘Bit N-1’.

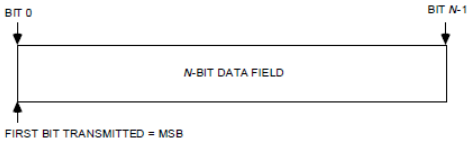


Figure 1-1: Bit Numbering Convention

When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., ‘Bit 0’ (see Figure 1-1).

In accordance with standard data-communications practice, data fields are often grouped into eight-bit ‘words’ which conform to the above convention. Throughout this document, such an eight-bit word is called an ‘octet’.

The numbering for octets within a data structure starts with zero.

By CCSDS convention, all ‘spare’ bits shall be permanently set to ‘0’.

2. OVERVIEW

This section provides an informative overview of the Spacecraft Monitor and Control Protocol (SMCP) to help the readers to understand the specification of the protocol given in the subsequent sections.

2.1. LAYERD STRUCTURE

This protocol specifies a method for constructing Command and Telemetry Messages and transmission sequences of these Messages. This protocol is located at a layer above the layer for transferring telemetry and commands over communications lines, and Messages are transferred between a spacecraft and the ground, within a spacecraft, and on the ground, using lower-layer transmission protocols.

Figure 2-1 shows an example of how Messages of this protocol are transferred using lower-layer protocols.

[Note] If the Standard Communications and Data Handling Architecture (SCDHA) [R1] is used, a specific protocol configuration is recommended in [R1] and the documents referred to therein.

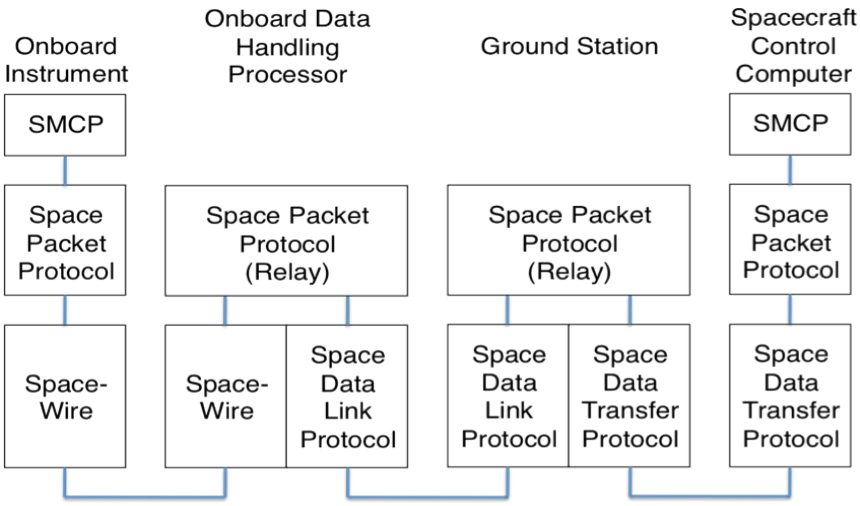


Figure 2-1: An Example of an End-to-End Protocol Configuration

2.2. OVERVIEW OF SMCP FUNCTIONS

The Spacecraft Monitor and Control Protocol (SMCP) is used to monitor the execution status of spacecraft functions (that is, jobs performed by spacecraft in flight such as observations and experiments) and to control the execution of these functions.

In order to perform monitor and control of spacecraft functions systematically, this protocol assumes that the functions of spacecraft are specified with the Functional Model of Spacecraft (FMS) [A1]. The central concept of this Model is a unit called the Functional Object, and the functions of a spacecraft are specified as a set of Functional Objects.

This protocol specifies Command and Telemetry Messages in terms of monitoring and controlling of Functional Objects from the outside of them. It also specifies sequences of Message exchanges for verifying the correct transfer of Messages.

2.3. FUNCTIONAL OBJECTS

This sub-section provides an informative overview of Functional Objects. For the exact definition of Functional

Objects, refer to [A1].

Since the functions that a spacecraft has are generally complex, they are usually specified as groups of functions, each consisting of functions closely related to each other. Such a group of functions is called a Functional Object in the Functional Model of Spacecraft (FMS).

Functional Objects are specified using such notions as Attributes, Operations, Events, Behavior, and Diagnostic Rules.

Attributes of a Functional Object are parameters representing the status of the Functional Object at each point of time, and their values can be provided to the outside of the Functional Object. The values of some Attributes can be set by the outside world through invocation of Operations.

Operations of a Functional Object are actions performed by the Functional Object and must be invoked by the outside world.

Events of a Functional Object are occurrences of things that have particular significances. Occurrences of Events are detected by the Functional Object itself. Occurrences of Events that are important to the outside world can be reported to the outside world, and reports of occurrences of important Events are called Alerts.

The behavior of a Functional Object describes rules about correct actions taken by the Functional Object, and is used to determine correct sequences of Operations and whether Attributes are valid or not, etc. The behavior of a Functional Object can be specified with a set of State Machines.

Diagnostic rules of a Functional Object are used by the outside world (other Functional Objects and/or the outside of the spacecraft) to determine whether the Functional Object is functioning correctly or not.

The definition of Functional Objects specified with FMS can be registered in the Spacecraft Information Base 2 (SIB2) [R4].

Functional Objects communicate with the outside world (other Functional Objects and/or the outside of the spacecraft), but methods for the communications between Functional Objects and the outside world are not specified by FMS. The SMCP provides methods for performing the following for monitoring and controlling Functional Objects using communications lines:

a) Invoke Operations of Functional Objects by the outside world;

b) Report values of Attributes of Functional Objects to the outside world; and

c) Report Alerts (occurrence of important Events) to the outside world.

2.4. IMPLEMENTATION OF SMCP MESSAGES

This document specifies types of SMCP Command and SMCP Telemetry and SMCP Messages related with the SMCP Commands and SMCP Telemetry.

Not all of the information related with SMCP Command and SMCP Telemetry is included in the SMCP Messages. Additional information is carried by the management system. In some cases, the management system uses fields in PCI of a lower layer protocol that carries the SMCP Message. For example, the management system described in [R2] uses fields in the Primary Header of the Space Packet to carry the additional information.

If a communications protocol (such as the Space Packet Protocol [R5]) is used for delivering SMCP Messages, the Messages are contained in the user data area of packets. In this case, the formats of individual Messages are stored in the Spacecraft Information Base 2 (SIB2) [R4].

If a shared memory is used for delivering messages, different SMCP Messages are written into and read from different memory addresses. In this case also, the memory addresses used for delivering Messages and the formats of individual Messages are stored in SIB2.

[Note] The current version of SIB2 does not support the case in which a shared memory is used for delivering SMCP Messages.

3. GENERAL

3.1. MONITOR AND CONTROL MODEL

The SMCP is used by a Controller to control a Target (see Figure 3-1). This protocol assumes that the Target is modeled as a Functional Object specified by the Functional Model of Spacecraft (FMS) [A1].

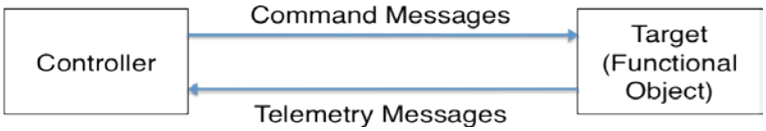


Figure 3-1: Monitor and Control Model

If both sender and receiver of messages are instruments or sub-systems onboard a spacecraft, it may not be clear which side should be modeled as the Controller. However, messages transferred regularly in one direction (e.g. time code or ancillary data) should be recognized as the Telemetry message. For the other messages, criteria between telemetry and command shall be determined for each project.

[Note] Specification in the previous paragraph is compatible with the recommendation in section 4.3.2.1.1 of [R7].

[Note] This protocol assumes that the Targets are instruments or sub-systems onboard a spacecraft, but anything (for example, ground support equipment, ground station equipment, etc.) can be a Target if it is modeled as a Functional Object. Typical Controllers include spacecraft control (or test) systems on the ground, onboard command and data handling computers, onboard attitude control computers, onboard payload (or mission) data processors, etc.

The Controller sends SMCP Command to the Target to control the Target. The Target sends SMCP Telemetry back to the Controller to be monitored by the Controller. Most of information items related with SMCP Commands and SMCP Telemetry are carried by SMCP message. The other information items related with SMCP Commands and SMCP Telemetry are carried by the management system. In some cases, the management system uses fields in PCI of a lower layer protocol that carries the SMCP Message.

A Controller can monitor and control multiple Targets. Different Targets shall be identified with the Functional Object Identifiers (FOIDs).

A Target can be monitored and controlled by multiple Controllers. Different Controllers shall be identified with the capability of a lower layer protocol (e.g. Space Packet Protocol [R5]). If multiple Controllers are used to control a Target, the behavior of these Controllers shall be coordinated by some activity, but how to do it is not in the scope of this protocol. If multiple Controllers are being used to monitor a Target at a time, Telemetry Messages shall be delivered to all these Controllers using the capability of a lower layer protocol.

3.2. MESSAGES

Each SMCP message shall consist of a Message Header and a Message Body (see Figure 3-2).

Figure 3-2: SMCP Message

There are two different formats for the Message Header: Command Message Header (used for Command Messages) and Telemetry Message Header (used for Telemetry Messages). The format of the Command

Message Header is specified in 4.2, and the format of the Telemetry Message Header is specified in 5.2

Command Messages and Telemetry Messages shall be distinguished by the management system. In some cases, the management system uses fields in PCI of a lower layer protocol that carries the SMCP Message.

The format of the Message Body is specified for each Message Type in sections 4 and 5.

Convention specified in 1.4 shall be applicable not only to the fields in the Message Header but also to the fields in the Message Body whose encoding rules should be stored in the Spacecraft Information Base 2 (SIB2) [R4].

3.3. COMMON INFORMATION ITEMS

Four information items are related with both SMCP Command and SMCP Telemetry:

a) SMCP Version Number;

b) Functional Object Identifier (FOID), Route Identifier (ID), Combined Upper Functional Object and Route Identifier (CUFORID), Upper FOID mask;

c) Attribute Identifier (ID); and

d) Command Message Identifier (ID)

3.3.1. SMCP Version Number

The value of the SMCP Version Number field shall identify the SMCP version. For this version of SMCP, the value of this field shall be 1.

3.3.2. Functional Object Identifier, Route Identifier, CUFORID, and Upper FOID mask

The Functional Object Identifier (FOID) shall uniquely identify the Functional Object that receives SMCP Command or generates SMCP Telemetry within a spacecraft. The FOID shall consist of 16 bits, which shall be divided into the Upper FOID (8 bits) and the Lower FOID (8bits).

[Note] An Upper FOID is identifier for a group of Functional Objects within a spacecraft (i.e. global identifier) and a Lower FOID is identifier for Functional Objects within a group of Functional Objects (i.e. local identifier). In order to encourage re-use of the Functional Objects from a spacecraft to the other spacecraft, not the global identifier, which may be different from a spacecraft to the other spacecraft, but only the local identifier appears in the SMCP messages.

The Route Identifier (ID) shall identify Route to a Functional Object via which Command or Telemetry Messages are delivered. Number of bits for the Route ID (n) shall be determined for each Functional Object and shall be in the range of 0-4 bits (see Figure 3-1).

For the Functional Object with n greater than 1, least significant n bits of the Upper FOID shall be 0. If a value (i) of the Upper FOID is allocated, usage of the Upper FOID value in the range between i+1 and $i+2^n-1$ is prohibited.

Sum of the Upper FOID and Route ID is an integer in 8 bits and called the CUFORID (Combined Upper Functional Object and Route Identifier). The CUFORID identifies a path via which Command or Telemetry Messages are delivered.

In order to extract the Upper FOID from the CUFORID, the integer called the Upper FOID mask is defined for each Upper FOID. The Upper FOID mask is 8 bits integer and all of the binary digits are 1 except for least significant n bits which are 0.

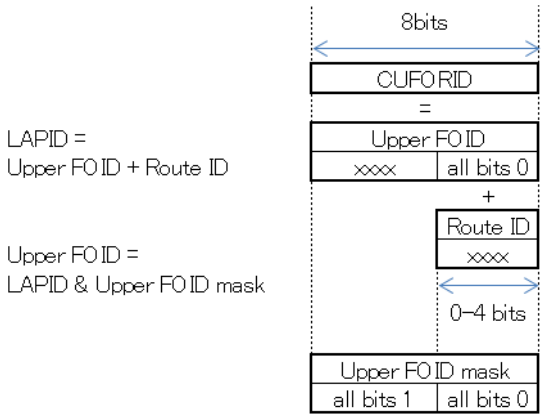


Figure 3-1: CUFORID, Upper FOID, Route ID, Upper FOID mask

While the Lower FOID shall be contained in the Lower FOID field of the SMCP Message Header (see 4.2.1 and 5.2.1), the Upper FOID nor the Route ID does not appear in the data structure defined by the SMCP. The value of the CUFORID is carried by the management system including the usage of a lower layer protocol.

[Example]

If there is a Functional Object with

- Number of bits for the Route ID : 3 bits (000b to 111b) ; and
- Upper FOID : 1010 1000b,

the Upper FOID mask is 1111 1000b and usage of the Upper FOID value between 1010 1001b and 1010 1111b is prohibited.

If the Route ID for a Route is 010b for the Functional Object, CUFORID is obtained by the following equation:

CUFORID = Upper FOID + Route ID

1010 1010b = 1010 1000b + 010b.

In the contrary, the Upper FOID is obtained from the CUFORID by the following equation:

Upper FOID = CUFORID & Upper FOID mask

1010 1000b = 1010 1010b & 1111 1000b.

Here, ‘&’ denotes binary multiplication operation.

3.3.3. Attribute ID

The Attribute ID shall identify the Attributes and Attribute Sequences uniquely within a Functional Object.

An Attribute Sequence is a sequence of Attributes.

Attributes and Attribute Sequences share the same identifier space.

[Note] The definition of each Attribute Sequence is stored in the Spacecraft Information Base 2 (SIB2) [R4].

3.3.4. Command Message ID

The value of the Command Message ID shall identify the Command Message.

4. SMCP COMMAND

4.1. GENERAL

The SMCP Command is categorized into the following four Types:

a) ACTION Command;

b) GET Command;

c) SET Command

d) MEMORY LOAD Command; and

e) MEMORY DUMP Command.

Most of information items related with SMCP Commands are carried by the Command Message. The other information items are carried by the management system. In some cases, the management system uses fields in PCI of a lower layer protocol that carries the Command Message.

Each Command Message shall consist of a Command Message Header and a Command Message Body. The format of the Command Message Header is specified in 4.2. The format of the Command Message Body is specified for each Message Type in 4.3 through 4.7.

4.2. COMMON INFORMATION ITEMS OF SMCP COMMAND

4.2.1. General

The format of the Command Message Header shall be as shown in Figure 4-1. In addition to the information items in the Command Message Header, CUFORID (see 3.3.2) and Command Message ID (see 3.3.4) shall be carried by the management system related with all types of the SMCP Command.

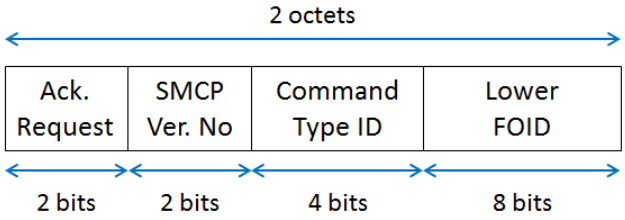


Figure 4-1: Command Message Header

4.2.2. Acknowledgement Request

The value of this field shall indicate whether an ACKNOWLEDGEMENT Telemetry Message (see 5.5) shall be generated upon reception of this Command Message. This field shall have one of the values shown in Table 4-1.

Table 4-1: Values of the Acknowledgement Request Field

Value (decimal)	Meaning
0	An ACKNOWLEDGEMENT Telemetry Message shall not be generated.
1	An ACKNOWLEDGEMENT Telemetry Message shall be generated.

[Note] As specified in 5.5.1, it can be determined for each Functional Object whether ACKNOWLEDGEMENT

Telemetry Messages are used or not. If it is determined that ACKNOWLEDGEMENT Telemetry Messages are not used for a particular Functional Object, that Functional Object shall never generate ACKNOWLEDGEMENT Telemetry Messages regardless of the value of this field.

4.2.3. SMCP Version Number

See 3.3.1.

4.2.4. Command Type Identifier

The value of the Command Type Identifier (ID) field shall identify the Type of the Command Message. This field shall have one of the values shown in Table 4-2.

Table 4-2: Values of the Command Type ID Field

Value (decimal)	Meaning
0	ACTION Command
1	GET Command
2	SET Command
4	MEMORY LOAD Command
5	MEMORY DUMP Command

4.2.5. Lower FOID

See 3.3.2.

4.2.6. Command Message ID

The value of Command Message ID (see 3.3.4) does not appear in the Command Message. This value is carried by the fields in PCI of a lower layer protocol that carries the Command Message.

[Note] [R2] specifies that the Packet Sequence Control field of the Space Packet that carries the Command Message shall be used for the value of the Command Message Identifier field if lower layer protocol is the Space Packet.

4.3. ACTION COMMAND

4.3.1. General

ACTION Commands shall be used to invoke Operations of Functional Objects [A1], except for the MemoryLoad and MemoryDump Operations of the Memory Functional Object [A1], which shall be invoked by the Memory Load Command (see 4.6) and the Memory Dump Command (see 4.7), respectively.

If an Operation has parameters, they shall be delivered to the Functional Object with the ACTION Command used to invoke the Operation (see 4.3.4).

Upon receiving an ACTION Command, the Functional Object shall execute the specified Operation.

[Note] As a result of executing an Operation, the values of some Attributes usually change. In some cases, a State Transition occurs as a result of executing an Operation and the value of the State Attribute associated with the State Chart changes. Whether the specified Operation has been successfully executed or not in response to a received ACTION Command can be verified by the Controller by checking the values of the Attributes that are supposed to change as a result of executing the Operation.

The format of the ACTION Command Message shall be as shown in Figure 4-2.

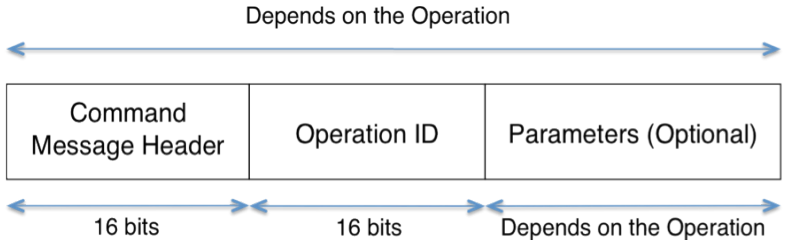


Figure 4-2: ACTION Command Message

4.3.2. Command Message Header

See 4.2.

4.3.3. Operation Identifier

The value of the Operation Identifier (ID) field shall identify the Operation to be invoked.

The Operation ID shall identify the Operations uniquely within a Functional Object.

The value of Operation ID, 0, is reserved for No Operation (NOP) except for already designed Functional Object.

[Note] Only one Operation can be invoked by an ACTION Command Message.

4.3.4. Parameters

The optional Parameters field, if used, shall contain values of the parameters specified for the Operation.

The number of parameters, their lengths, and their encoding rules shall be specified for each Operation. If the Operation has no parameter, this field shall be omitted. If the Operation has parameters, values of all the parameters shall be contained in this field.

The maximum size of the Parameters field shall be determined for each project.

[Note 1] The parameters, their lengths, and their encoding rules are stored in the Spacecraft Information Base 2 (SIB2) [R4] for each Operation.

[Note 2] There are Operations used for setting values to Attributes. In such case, the values to be set to the Attributes are delivered to the Functional Object as parameters of an ACTION Command.

4.4. GET COMMAND

4.4.1. General

GET Commands shall be used to invoke generation of a (response) VALUE Telemetry Message (see 5.3.1).

[Note] Whether the VALUE Telemetry Message has been successfully generated or not in response to a received GET Command can be verified by the Controller by checking whether the VALUE Telemetry Message has been received by the Controller.

The format of the GET Command Message shall be as shown in Figure 4-3.

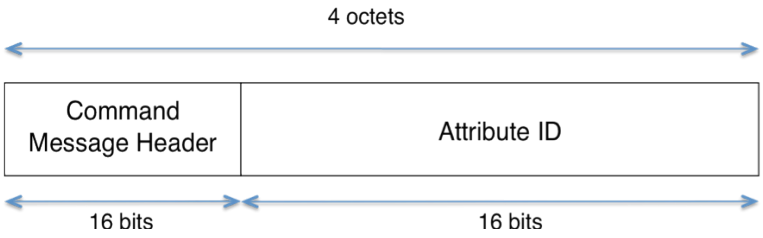


Figure 4-3: GET Command Message

4.4.2. Command Message Header

See 4.2.

4.4.3. Attribute ID

The value of the Attribute ID field shall identify the Attribute or Attribute Sequence whose value is to be returned with the VALUE Telemetry Message. See also 3.3.3.

[Note] There may be Attributes and Attribute Sequences whose values cannot be retrieved with a GET Command. The Attributes and Attribute Sequences whose values can be retrieved with a GET Command are registered in the Spacecraft Information Base 2 (SIB2) [R4].

4.5. SET COMMAND

4.5.1. General

SET Commands shall be used to set values to attributes. The SET Command also invokes the Operation including NOP. The format of the SET Command Message shall be as shown in Figure 4-3.

[Note] As a result of response to the SET Command, the values of corresponding Attributes usually change. The change can be verified by the Controller by checking the values of the Attributes that are supposed to change as a result of response to the SET Command.

[Note] As a result of response to either the ACTION Command or the SET Command, values of the Attributes may change. In the former case, the Attributes are specified in the Spacecraft Information Base 2 (SIB2) [R4] for each Operation. In the latter case, the Attributes are specified by the Attribute ID in the SET Command Message.

[Note] Role of the SET Command and ACTION Command have overlap. In a step of a design of a Functional Object, policy for usage of the SET Command and ACTION Command should be decided. Major difference between SET Command and ACTION Command is the name rule of the Command: For the SET Command, name of Attributes is encoded. For ACTION Command, name of the Operation is encoded.

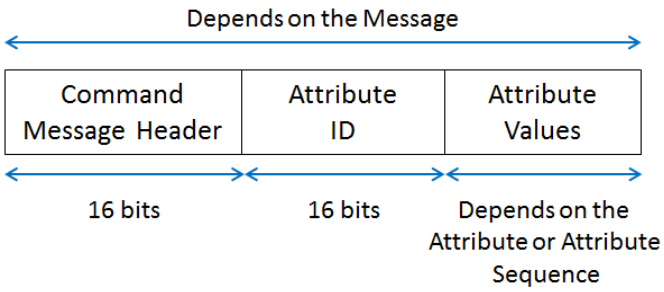


Figure 4-3: SET Command Message

4.5.2. Command Message Header

See 4.2.

4.5.3. Attribute ID

The value of the Attribute ID field shall identify the Attribute or Attribute Sequence whose value is to be set by the SET Command Message. See also 3.3.3.

[Note] There may be Attributes and Attribute Sequences whose values cannot be set with a SET Command. The Attributes and Attribute Sequences whose values can be set with a SET Command are registered in the Spacecraft Information Base 2 (SIB2) [R4].

4.5.4. Attribute Values

The Attribute Values field shall contain the value of the Attribute if the Attribute ID (see 5.3.3) is for an Attribute, or the values of all the Attributes contained in the Attribute Sequence if the Attribute ID is for an Attribute Sequence.

The length and encoding rule shall be specified for each Attribute.

[Note 1] The length and encoding rule are stored in the Spacecraft Information Base 2 (SIB2) [R4] for each Attribute.

The maximum size of the Attribute Values field shall be determined for each project (see Table C-1).

4.6. MEMORY LOAD COMMAND

4.6.1. General

MEMORY LOAD Commands shall be used to invoke the MemoryLoad Operation of Memory Functional Objects [A1].

Upon receiving a MEMORY LOAD Command, the Memory Functional Object shall store the Load Data contained in the Message into the specified memory area.

[Note] Whether the Load Data has been successfully stored in the specified memory area or not in response to a received MEMORY LOAD Command can be verified by the Controller by sending a MEMORY DUMP Command Message (see 4.7) and receiving a MEMORY DUMP Telemetry Message (see 5.6). It is also recommended that some check bits (such as CRC) be embedded in the Load Data and the Target verify whether it has received all the data correctly by performing a necessary check using the check bits.

The format of the MEMORY LOAD Command Message shall be as shown in Figure 4-4.

Some of the Memory Functional Object may have restriction on the Start Address and octet size of the Load

Data. Allowable restrictions are defined in [A1] and the restriction for each Memory Functional Object is registered in the Spacecraft Information Base 2 [R4].

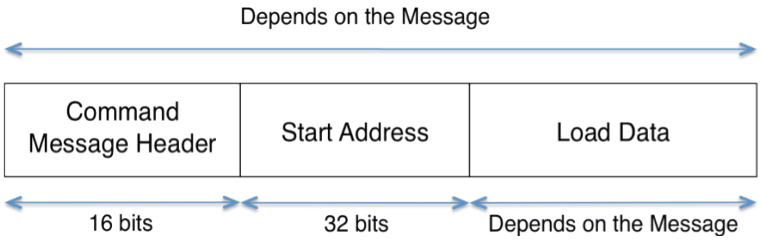


Figure 4-4: MEMORY LOAD Command Message

4.6.2. Command Message Header

See 4.2.

4.6.3. Start Address

The value of the Start Address field shall identify the first address of the memory area in which the data is to be written into.

How to map the value of this field to a physical address of the memory shall be determined for each Memory Object (identified by the FOID).

4.6.4. Load Data

The Load Data field shall contain the data to be written into the memory.

All the data in this field shall be written into the memory area that starts with the Start Address contiguously.

The maximum size of the Load Data field shall be determined for each project (see Table C-1).

4.7. MEMORY DUMP COMMAND

4.7.1. General

MEMORY DUMP Commands shall be used to invoke the MemoryDump Operation of Memory Functional Objects [A1].

Upon receiving a MEMORY DUMP Command Message, the Memory Functional Object shall send back the requested memory data using one or more MEMORY DUMP Telemetry Messages.

The format of the MEMORY DUMP Command Message shall be as shown in Figure 4-5.

The field shown as “Reserved” shall not be used and be filled with all 0’s.

Some of the Memory Functional Object may have restriction on the Start Address and Data Length. Allowable restrictions are defined in [A1] and the restriction for each Memory Functional Object is registered in the Spacecraft Information Base 2 [R4].

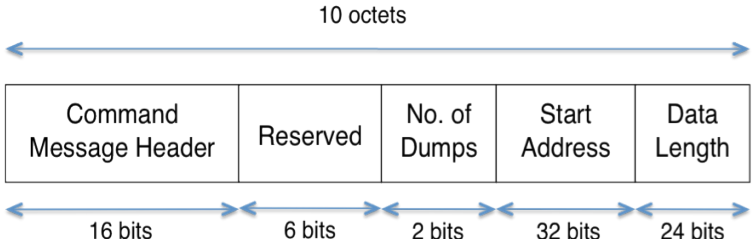


Figure 4-5: MEMORY DUMP Command Message

4.7.2. Command Message Header

See 4.2.

4.7.3. Number of Dumps

The Number of Dumps field shall contain *n*-1 if the same data shall be sent with MEMORY DUMP Telemetry Messages *n* times.

[Note] The maximum number of dumps is four.

[Note] If the condition of the communications link is not very good and the data must be received reliably, the same data can be dumped two or more times.

[Example] If the value of this field is 0, the memory data shall be sent once. If the value of this field is 1, then the memory data shall be sent twice.

4.7.4. Start Address

The value of the Start Address field shall identify the first address of the memory area to be sent.

How to map the value of this field to a physical address of the memory shall be determined for each Memory Object (identified by the FOID).

4.7.5. Data Length

The value of the Data Length field shall indicate the length of the data (in octets) to be dumped.

The data of this length, starting with the Start Address, shall be read out from the memory contiguously and sent with one or more MEMORY DUMP Telemetry Messages (see 5.6.1).

[Note] The Data Length (e.g. 512 octets or 128 octets) can be determined for each Memory Functional Objects according to its restriction for implementation.

The maximum value allowed for the Data Length field shall be determined for each project (see Table C-1).

5. SMCP TELEMETRY

5.1. GENERAL

The SMCP Telemetry is categorized into the following four Types:

a) VALUE Telemetry;

b) NOTIFICATION Telemetry;

c) ACKNOWLEDGEMENT Telemetry; and

d) MEMORY DUMP Telemetry.

Most of information items related with SMCP Telemetry are carried by the Telemetry Message. The other information items are carried by the management system. In some cases, the management system uses fields in PCI of a lower layer protocol that carries the Telemetry Message.

As specified in 3.2, each Telemetry Message consists of a Telemetry Message Header and a Telemetry Message Body. The format of the Telemetry Message Header is specified in 5.2. The format of the Telemetry Message Body is specified for each Message Type in 5.3 through 5.6.

5.2. COMMON INFORMATION ITEMS OF SMCP TELEMETRY

5.2.1. General

The format of the Telemetry Message Header shall be as shown in Figure 5-1. In addition to the information items in the Telemetry Message Header, CUFORID (see 3.3.2) shall be carried by the management system related with all types of the SMCP Telemetry.

The field shown as “Reserved” shall not be used and be filled with all 0’s.

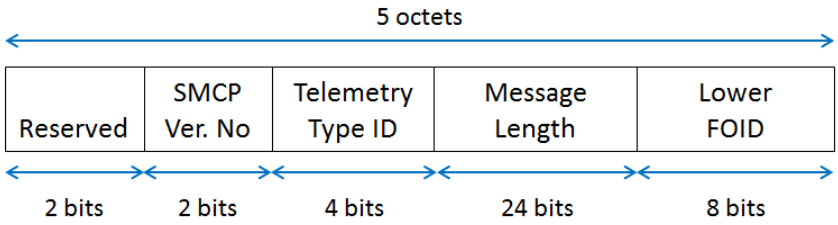


Figure 5-1: Telemetry Message Header

5.2.2. SMCP Version Number

See 3.3.1.

5.2.3. Telemetry Type Identifier

The value of the Telemetry Type Identifier (ID) field shall identify the Type of the Telemetry Message. This field shall have one of the values shown in Table 5-1.

Table 5-1: Values of the Telemetry Type ID Field

Value (decimal)	Meaning
0	VALUE Telemetry
1	NOTIFICATION Telemetry

2	ACKNOWLEDGEMENT (ACK) Telemetry
5	MEMORY DUMP Telemetry

5.2.4. Message Length

The value of the Message Length field shall indicate the length (in octet) of the entire Telemetry Message (including the Message Header).

[Note] This field enables simple implementation of the blocking and the segmentation of the Message in the lower layer protocol.

5.2.5. Lower FOID

See 3.3.2.

5.3. VALUE TELEMETRY

5.3.1. General

The VALUE Telemetry shall be used for sending the values of Attributes.

There are three kinds of VALUE Telemetry in terms of how Messages are generated:

a) Periodic VALUE Telemetry: Generated by the Functional Object periodically at a pre-specified interval;

b) Response VALUE Telemetry: Generated by the Functional Object in response to a received GET Command; and

c) VALUE Change Telemetry: Generated by the Functional Object when the value of an Attribute has changed.

[Note] The interval at which periodic VALUE Telemetry is generated may change depending on the operation mode of the spacecraft or the instrument.

When the value of an Attribute marked as “change-notify” has changed, either a VALUE Telemetry Message containing the new value of that Attribute or a VALUE Telemetry Message containing values of an Attribute Sequence containing that Attribute shall be generated.

[Note] Whether each Attribute is marked as “change-notify” or not should be registered in the Spacecraft Information Base 2 [R4] but is not implemented yet.

The format of the VALUE Telemetry Message shall be as shown in Figure 5-2.

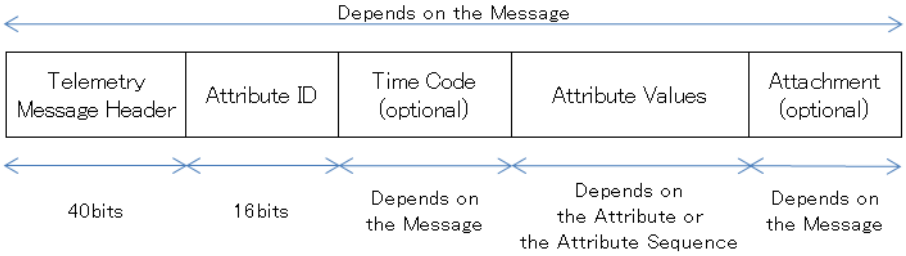


Figure 5-2: VALUE Telemetry Message

5.3.2. Telemetry Message Header

See 5.2.

5.3.3. Attribute ID

The value of the Attribute ID field shall identify the Attribute or Attribute Sequence whose value is contained in the Message. See also 3.3.3.

5.3.4. Time Code

The time at which each of the values contained in the Message is obtained shall be associated with the time specified for each Message, which is called the Message Time. How to associate the time of each value with the Message Time shall be determined for each Attribute or Attribute Sequence.

The Message Time can be described with the optional Time Code in the Value Telemetry Message and/or fields in a lower-layer protocol.

The format of the Time Code, if it exists, shall be CCSDS Unsegmented Time Code (CUC) defined by [A2].

[Note] If the Standard Communications and Data Handling Architecture (SCDHA) [R1] is used, the method for transferring the Message Time with a lower-layer protocol is specified in [R2].

5.3.5. Attribute Values

The Attribute Values field shall contain the value of the Attribute if the Attribute ID (see 5.3.3) is for an Attribute, or the values of all the Attributes contained in the Attribute Sequence if the Attribute ID is for an Attribute Sequence.

The length and encoding rule shall be specified for each Attribute.

[Note 1] The length and encoding rule are stored in the Spacecraft Information Base 2 (SIB2) [R4] for each Attribute.

The maximum size of the Attribute Values field shall be determined for each project (see Table C-1).

5.3.6. Attachment

The optional Attachment field, if used, shall contain binary data with any format related to the Attributes contained in the Message (for example, observation results).

The presence or absence of this field, and, if present, its format or encoding rule and its length shall be determined for each value of the Attribute ID.

The maximum size of the Attachment field shall be determined for each project (see Table C-1).

[Note] The format or encoding rule of attachments is not stored in the Spacecraft Information Base 2 (SIB2) [R4].

5.4. NOTIFICATION TELEMETRY

5.4.1. General

The NOTIFICATION Telemetry shall be used to for sending Alerts (reports of occurrences of important Events) [A1].

Whenever a Functional Object issues an Alert, a NOTIFICATION Telemetry Message shall be generated.

The format of the NOTIFICATION Telemetry Message shall be as shown in Figure 5-3.

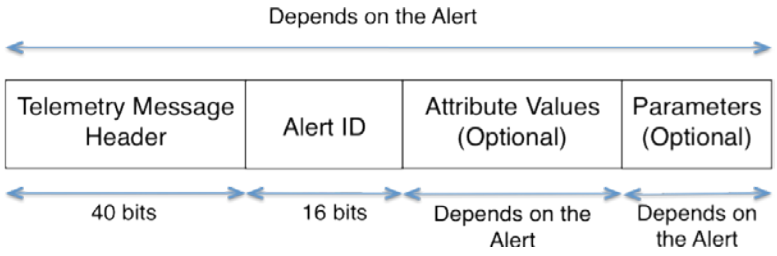


Figure 5-3: NOTIFICATION Telemetry Message

5.4.2. Telemetry Message Header

See 5.2.

5.4.3. Alert Identifier

The value of the Alert Identifier (ID) field shall identify the Alert being sent.

The Alert ID shall identify the Alerts uniquely within a Functional Object.

[Note] Only one Alert can be sent with a NOTIFICATION Telemetry Message.

5.4.4. Attribute Values

The optional Attribute Values field, if used, shall contain the values of Attributes specified for the Alert.

What Attribute values should be contained in a NOTIFICATION Telemetry Message shall be specified for each Alert. If the Alert has no Attribute associated with it, this field shall be omitted. If the Alert has Attributes associated with it, values of all the Attributes shall be contained in this field.

The maximum size of the Attribute Values field shall be determined for each project (see Table C-1).

[Note] What Attribute values should be contained, and their length and encoding rule are stored in the Spacecraft Information Base 2 (SIB2) [R4] for each Alert.

5.4.5. Parameters

The optional Parameters field, if used, shall contain the values of parameters specified for the Alert (see [A1]).

The number of parameters, their lengths, and their encoding rules shall be specified for each Alert. If the Alert has no parameter associated with it, this field shall be omitted. If the Alert has parameters associated with it, values of all the parameters shall be contained in this field.

The maximum size of the Parameters field shall be determined for each project (see Table C-1).

[Note] The parameters, their lengths, and their encoding rules are stored in the Spacecraft Information Base 2 (SIB2) [R4] for each Alert.

5.5. ACKNOWLEDGEMENT TELEMETRY

5.5.1. General

The ACKNOWLEDGEMENT (ACK) Telemetry shall be used for notifying the Controller that a Command Message has been received.

[Note] Whether a Command Message has been correctly executed or not can be verified with the procedures described in Notes in sub-sections that define Command Types.

It can be determined for each Functional Object whether ACKNOWLEDGEMENT Telemetry Messages are used or not.

If it is determined that ACKNOWLEDGEMENT Telemetry Messages are used for a particular Functional Object, an ACK Telemetry Message shall be generated only if it is requested in the Acknowledgement Request field contained in the received Command Message Header (see 4.2.2). If the value of this field of a received Command Message is 1, an ACK Telemetry Message shall be generated for that Command Message. If the value of this field of a received Command Message is 0, an ACK Telemetry Message shall not be generated for that Command Message.

If it is determined that ACKNOWLEDGEMENT Telemetry Messages are not used for a particular Functional Object, that Functional Object shall never generate ACKNOWLEDGEMENT Telemetry Messages regardless of the value of the Acknowledgement Request field in the received Command Message Header.

The format of the ACKNOWLEDGEMENT Telemetry Message shall be as shown in Figure 5-4.

Figure 5-4: ACKNOWLEDGEMENT Telemetry Message

5.5.2. Telemetry Message Header

See 5.2.

5.5.3. Command Message ID

The value of the Command Message ID (see 3.3.4) field shall identify the Command Message being acknowledged by this ACKNOWLEDGEMENT Telemetry Message.

5.6. MEMORY DUMP TELEMETRY

5.6.1. General

The MEMORY DUMP Telemetry shall be used for sending memory data in response to a received MEMORY DUMP Command Message.

Upon receiving a MEMORY DUMP Command, a Memory Functional Object shall send back the requested memory data using one or more MEMORY DUMP Telemetry Messages (see 4.7.1).

The data of the length specified in the received MEMORY DUMP Command Message, starting with the Start Address specified in the same Command Message, shall be read out from the memory contiguously and sent with one or more MEMORY DUMP Telemetry Messages. If all the data cannot be sent with one MEMORY DUMP Telemetry Message, multiple Messages, each containing non-overlapping data, shall be used.

The requested data shall be sent the number of times specified in the MEMORY DUMP Command Message (see 4.7.3).

The format of the MEMORY DUMP Telemetry Message shall be as shown in Figure 5-4.

The field shown as “Reserved” shall not be used and be filled with all 0’s. (追記した、要確認)

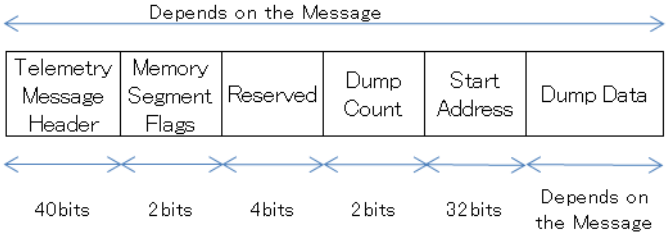


Figure 5-5: MEMORY DUMP Telemetry Message

5.6.2. Telemetry Message Header

See 5.2.

5.6.3. Memory Segment Flags

The value of the Memory Segment Flags field shall indicate whether the memory data specified in the received MEMORY DUMP Command Message are sent divided in multiple MEMORY DUMP Telemetry Messages or not, and, if divided, whether the data contained in this Message is the first segment, an intermediate segment, or the last segment of the entire memory data specified in the received MEMORY DUMP Command Message.

This field shall have one of the values shown in Table 5-1.

Table 5-1: Values of the Memory Segment Flags Field

Value (decimal)	Meaning
0	This Message contains an intermediate segment of the memory data.
1	This Message contains the first segment of the memory data.
2	This Message contains the last segment of the memory data.
3	This Message contains the entire data (not divided in multiple Messages).

5.6.4. Dump Count

The Dump Count field shall contain *n*-1 if this is the *n*-th transmission of the same data.

[Note] As specified in 4.7.3, the same data can be sent multiple times. If the data are sent only once, the value of this field shall be 0.

[Example] If the value of this field is 0, this is the first transmission of the data. If the value of this field is 1, this is the second transmission of the data.

5.6.5. Start Address

The value of the Start Address field shall identify the first address of the data contained in the Dump Data field.

How to map the value of this field to a physical address of the memory shall be determined for each Memory Object (identified by the FOID).

[Note] If the memory data requested by a MEMORY DUMP Command Message are sent in multiple MEMORY DUMP Telemetry Messages, the value of the Start Address field of the generated Telemetry Messages does not

coincide with the Start Address specified in the received Command Message except for the first Telemetry Message because the Start Address field of a MEMORY DUMP Telemetry Message contains the first address of the memory data contained in that Telemetry Message.

5.6.6. Dump Data

The Dump Data field shall contain the memory data to be sent.

The maximum size of the Dump Data field shall be determined for each project (see Table C-1).

APPENDIX A ACRONYMS

ACK	Acknowledgement
CUFORID	Combined Upper Functional Object and Route Identifier
FOID	Functional Object Identifier
GSTOS	Generic Spacecraft Test and Operations Software
ID	Identifier
PCI	Protocol Control Information
SIB	Spacecraft Information Base
SCDHA	Standard Communications and Data Handling Architecture
SMCP	Spacecraft Monitor and Control Protocol

APPENDIX B COMMAND-TELEMETRY INTERACTIONS (INFORMATIVE)

B.1 GENERAL

This informative annex summarizes information on interactions between Command Messages and Telemetry Messages.

B.2 ACKNOWLEDGEMENT

It can be determined for each Functional Object whether ACKNOWLEDGEMENT Telemetry Messages are used or not (see 5.5.1).

If it is determined that ACKNOWLEDGEMENT Telemetry Messages are used for a particular Functional Object, an ACK Telemetry Message shall be generated only if it is requested in the Acknowledgement Request field contained in the received Command Message Header (see 4.2.2). If the value of this field of a received Command Message is 1, an ACK Telemetry Message shall be generated for that Command Message. If the value of this field of a received Command Message is 0, an ACK Telemetry Message shall not be generated for that Command Message.

If it is determined that ACKNOWLEDGEMENT Telemetry Messages are not used for a particular Functional Object, that Functional Object shall never generate ACKNOWLEDGEMENT Telemetry Messages regardless of the value of the Acknowledgement Request field in the received Command Message Header.

B.3 ACTION COMMAND

B.3.1 General

The Functional Objects shall perform either of the two actions described in B.3.2 and B.3.3 when they have received an ACTION Command Message.

The Functional Objects that use ACKNOWLEDGEMENT Telemetry Messages shall also generate an ACKNOWLEDGEMENT Telemetry Message upon reception of an ACTION Command Message, if the value of the Acknowledgement Request field of the received Command Message is 1.

B.3.2 No Special Response

The Functional Object does not generate any special Telemetry Message upon execution of an ACTION Command Message. It generates VALUE Telemetry Messages periodically (Periodic VALUE Telemetry, see 5.3.1) whether or not it has received an ACTION Command Message.

The Controller verifies whether the sent ACTION Command has been executed or not by receiving periodic VALUE Telemetry Messages and by checking changes in the values of the Attributes that are supposed to change as a result of executing the Operation specified by the ACTION Command (see Figure B-1).

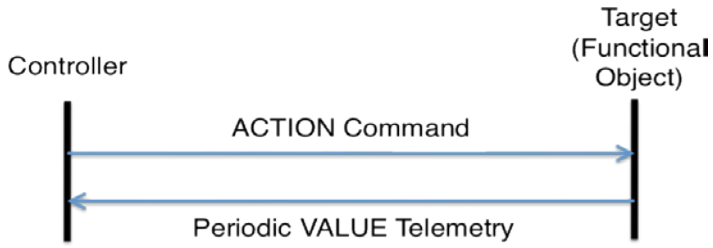


Figure B-1: No Special Response for ACTION Command

B.3.3 Response with Results

The Functional Object notifies the Controller of the execution results of a received ACTION Command Message by sending VALUE Change Telemetry Messages (see 5.3.1) containing the values of the Attributes that have changed as a result of executing the Operation specified by the ACTION Command (see Figure B-2). It may send multiple VALUE Change Telemetry Messages if the values of multiple Attributes change as a result of executing the Operation. It may send the same VALUE Change Telemetry Messages multiple times if the values of Attributes change multiple times in the course of executing the Operation.

The Functional Object may or may not generate Periodic VALUE Telemetry Messages (see 5.3.1) containing the values of the Attributes that change as a result of executing the Operation.

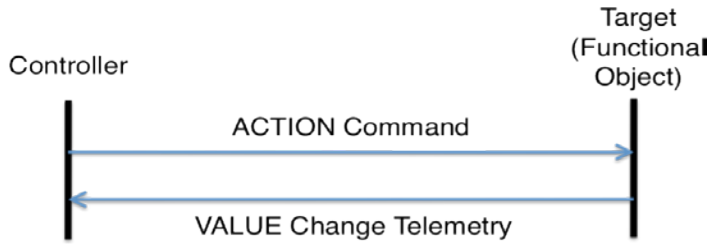


Figure B-2: Response with Results for ACTION Command

B.4 GET COMMAND

B.4.1 General

The Functional Objects shall perform the action described in B.4.2 when they have received a GET Command Message.

The Functional Objects that use ACKNOWLEDGEMENT Telemetry Messages shall also generate an ACKNOWLEDGEMENT Telemetry Message upon reception of a GET Command Message, if the value of the Acknowledgement Request field of the received Command Message is 1.

B.4.2 Response with Results

The Functional Object shall generate a Response VALUE Telemetry Message (see 5.3.1) containing the value of the Attribute or Attribute Sequence requested by the GET Command Message (see Figure B-3).

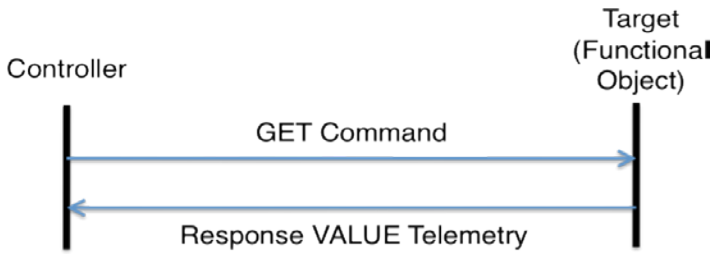


Figure B-3: Response with Results for GET Command

B.5 SET COMMAND

In the sense of interactions between Command Messages and Telemetry Messages, the SET Command is the same with the ACTION Command.

APPENDIX C MANAGED PARAMETERS

The values of the managed parameters are shown in Table C-1.

[Note] Allowed values may be restricted by the specification of usage of lower layer protocols (e.g. [R2]).

Table C-1: Table of the Managed Parameters

Managed Parameters	Section in This Document	Allowed Values	Unit
Maximum size of the Parameters field of the ACTION Command Message	4.3.4	Integer	Octet
Maximum size of the Attribute Values field of the SET Command Message	4.5.4	Integer	Octet
Maximum size of the Load Data field of the MEMORY LOAD Command Message	4.6.4	Integer	Octet
Maximum value allowed for the Data Length field of the MEMORY DUMP Command Message	4.7.5	Integer (up to FFFFFFh)	Octet
Maximum value for the sum of the sizes of the Attribute Values and Attachment fields of the VALUE Telemetry Message	5.3.5 and 5.3.6	Integer (up to FFFFF8h)	Octet
Maximum value for the sum of the sizes of the Attribute Values and Parameters fields of the NOTIFICATION Telemetry Message	5.4.4 and 5.4.5	Integer (up to FFFFF8h)	Octet
Maximum size of the Dump Data field of the MEMORY DUMP Telemetry Message	5.6.6	Integer (up to FFFFF5h)	