衛星の機能モデル

（Functional Model of Spacecrafts (FMS)）

2020年3月31日 制定
宇宙航空研究開発機構
免責条項
ここに含まれる情報は、一般的な情報提供のみを目的としています。JAXA は、かかる情報の正確性、有用性又は適時性を含め、明示又は黙示に何ら保証するものではありません。また、JAXA は、かかる情報の利用に関連する損害について、何ら責任を負いません。

Disclaimer
The information contained herein is for general informational purposes only. JAXA makes no warranty, express or implied, including as to the accuracy, usefulness or timeliness of any information herein. JAXA will not be liable for any losses relating to the use of the information.

発行
〒305-8505 茨城県つくば市千現 2-1-1
宇宙航空研究開発機構 安全・信頼性推進部
JAXA（Japan Aerospace Exploration Agency）
This document was originally drafted in English, then subsequently translated into Japanese and authorized by the Japanese space agency, JAXA.

The English translation is for reference purposes only, except for some tables and figures that contain English only, in which case they are the original. If there is anything ambiguous about the content of the text, please refer to both the Japanese version and the English version and contact JAXA Safety and Mission Assurance Department.
Functional Model of Spacecrafts (FMS)
CONTENTS // 目次

1. INTRODUCTION // はじめに.............................................................................................................. 1
  1.1. Purpose // 目的................................................................................................................................. 1
  1.2. Scope // 範囲................................................................................................................................. 2
  1.3. Applicability // 適用先................................................................................................................... 2
  1.4. References // 関連文書.................................................................................................................. 2
     1.4.1. Normative References // 引用文書.......................................................................................... 2
     1.4.2. Informative References // 参考文書...................................................................................... 2
  1.5. Document Structure // 本書の構成................................................................................................... 3
  1.6. Definitions and Notations // 定義及び表記法............................................................................... 4
     1.6.1. Terms defined in this document // 本書で定義される用語...................................................... 4
     1.6.2. Notations // 表記法.................................................................................................................. 4
  1.7. Verbal forms // 表現形式.............................................................................................................. 5

2. OVERVIEW ........................................................................................................................................... 7
  2.1. General // 一般............................................................................................................................. 7
  2.2. Purpose of This Model // このモデルの目的.................................................................................... 7
  2.3. Functional Object.......................................................................................................................... 8
  2.4. Spacecraft Information Base 2 ...................................................................................................... 10
  2.5. Communications Between Functional Objects and Other entities Functional Objects と他の構成要素
       の間の通信........................................................................................................................................ 11

3. METHOD OF MODELING WITH FUNCTIONAL OBJECTS　FUNCTIONAL OBJECTS によるモデル化
       の方法............................................................................................................................................. 13
  3.1. General // 一般............................................................................................................................. 13
  3.2. Functional Objects .......................................................................................................................... 15
     3.2.1. General // 一般.......................................................................................................................... 15
     3.2.2. Parent-Child Relation // 親子関係............................................................................................ 16
     3.2.3. Effective-Conditions of Functional Objects (Valid/Invalid and Effective-Condition Expressions)
       Functional Objects の有効条件（有効/無効及び有効条件式）...................................................... 17
  3.3. Attributes...................................................................................................................................... 18
     3.3.1. General // 一般.......................................................................................................................... 18
     3.3.2. Effective-Conditions of Attributes (Valid/Invalid and Effective-Condition Expressions)
       Attributes の有効条件（有効/無効及び有効条件式）............................................................. 19
     3.3.3. Criticality Levels of values ....................................................................................................... 19
     3.3.4. Initial Value .............................................................................................................................. 20
     3.3.5. Scalar Attributes and Complicated Attributes ........................................................................ 20
     3.3.6. Enumerative Attribute ............................................................................................................. 21
        3.3.6.1. General // 一般.......................................................................................................................... 21
        3.3.6.2. Values and Enumerative Names // 値及び Enumerative Names........................................... 21
        3.3.6.3. Criticality Level .................................................................................................................... 21
     3.3.7. Numerical Value Attribute....................................................................................................... 22
        3.3.7.1. General // 一般.......................................................................................................................... 22
        3.3.7.2. Valid Ranges.......................................................................................................................... 22
        3.3.7.3. Action Limits, Caution Limits, and Criticality Level............................................................... 22
  3.4. Operations ..................................................................................................................................... 23
     3.4.1. General // 一般.......................................................................................................................... 23
     3.4.2. Attribute Change Rules ............................................................................................................ 24
     3.4.3. No Operation (NOP) .................................................................................................................. 24
3.4.4. Attributes and Parameters of Operations .............................................................................. 25
  3.4.4.1. General // 一般 ......................................................................................................................... 25
  3.4.4.2. Valid Ranges ............................................................................................................................... 25
3.4.5. Effective-Conditions of Operations (Valid/Invalid and Effective-Condition Expressions) 
  Operations の有効条件 (有効/無効及び有効条件式) ............................................................................. 26
3.4.6. Criticality Level ............................................................................................................................ 27
3.5. Event classes ................................................................................................................................. 28
  3.5.1. Event and Event classes ............................................................................................................... 28
    3.5.1.1. General // 一般 ......................................................................................................................... 28
    3.5.1.2. Trigger Conditions .................................................................................................................. 28
  3.5.2. Alerts and Alert classes .............................................................................................................. 29
    3.5.2.1. General // 一般 ......................................................................................................................... 29
    3.5.2.2. Triger classes of Alerts classes ............................................................................................... 29
    3.5.2.3. Parameters of Alerts classes .................................................................................................. 30
3.6. State Machines ........................................................................................................................... 31
  3.6.1. General // 一般 ........................................................................................................................... 31
  3.6.2. States, Names of States, and Current State .............................................................................. 31
  3.6.3. State Attributes .......................................................................................................................... 31
  3.6.4. Valid/Invalid // 有効/無効 .......................................................................................................... 33
  3.6.5. Initial State .................................................................................................................................. 33
  3.6.6. Transitions and State Transition classes // 遷移及び State Transition classes ......................... 34
    3.6.6.1. General // 一般 ......................................................................................................................... 34
    3.6.6.2. Trigger classes of State Transition classes ............................................................................... 34
    3.6.6.3. Maximum Allowable Transition Time and Minimum Allowable Transition Time .............. 35
    3.6.6.4. Effective States for Operations ............................................................................................... 35
3.7. Diagnostic Rules .......................................................................................................................... 36
3.8. Other Features // その他 .............................................................................................................. 37
  3.8.1. Condition Expression // 条件式 .................................................................................................. 38
4. FUNCTIONAL CLASS .................................................................................................................. 39
  4.1. General // 一般 ........................................................................................................................... 39
  4.2. Parent-Child Relation // 親子関係 ............................................................................................... 40
  4.3. standard Functional Class // 標準的な Functional Class ............................................................ 40
5. MEMORY FUNCTIONAL CLASS .................................................................................................. 41
  5.1. General // 一般 ........................................................................................................................... 41
  5.2. Design Parameters // 設計パラメータ ......................................................................................... 42
    5.2.1. General // 一般 ........................................................................................................................... 42
    5.2.2. FirstAddress and LastAddress ............................................................................................... 42
    5.2.3. MaximumUploadLength ........................................................................................................... 42
    5.2.4. AlignmentLength ...................................................................................................................... 42
  5.3. Operations .................................................................................................................................... 43
    5.3.1. General // 一般 ........................................................................................................................... 43
    5.3.2. MemoryLoad .............................................................................................................................. 43
    5.3.3. MemoryDump ............................................................................................................................. 44
1. INTRODUCTION // はじめに

1.1. PURPOSE // 目的

The purpose of this document is to specify the method to model functions of spacecrafts and their onboard instruments (hereafter, the term spacecraft refers to an entire spacecraft and/or its onboard instruments). The model specified in this document serves as a guideline for the functional design of spacecrafts in the sense that spacecraft functions are designed in a way that they can be specified with this model.

This model sets a set of standardized methods to specify functions of spacecrafts and to manage electronically information of their functions. This standardized model would make systematic development of spacecraft functions easier and make reusing existing onboard instruments or parts of them practical. Then, the ultimate purpose is to reduce the cost of development of new spacecrafts and to enhance their reliability.

The Generic Spacecraft Test and Operations Software (GSTOS) specified in [R2] assumes that the design of the spacecraft follows this document and the Spacecraft Monitor and Control Protocol (SMCP) [R3]. The individual specifications according to this model are supposed to be stored and managed by the Spacecraft Information Base 2 (SIB2) [R1].
1.2. SCOPE // 範囲

This document specifies the Functional Model of Spacecrafts (FMS), i.e., how spacecraft functions are specified.

This document does not specify how these requirements are implemented with hardware or software.

1.3. APPLICABILITY // 適用先

The specifications described in this document apply to the GSTOS and the software which implements the SIB2.

The specifications described in this document apply also to the spacecrafts and their ground systems for the projects that adopt the GSTOS and the SIB2.

1.4. REFERENCES // 関連文書

1.4.1. Normative References // 引用文書

None.

1.4.2. Informative References // 参考文書


1.5. DOCUMENT STRUCTURE // 本書の構成

This document is organized as follows:

Chapter 1 (this chapter) states the purpose, scope, and applicability of the document, and lists the references, definitions, and notations used throughout the document.

Chapter 2 presents an overview of the FMS.

Chapter 3 specifies the Functional Object, which is the core concept of the FMS.

Chapter 4 specifies Functional Class, which is used as a template for defining the Functional Objects.

Chapter 5 specifies the Memory Functional Class, which is a Functional Class for defining the Functional Objects that represent onboard memories.

Appendix A lists all the acronyms used in this document.

Appendix B shows an example of a Functional Object.
1.6. DEFINITIONS AND NOTATIONS // 定義及び表記法

1.6.1. Terms defined in this document // 本書で定義される用語
None.

1.6.2. Notations // 表記法
The following notations are used throughout this document.

A paragraph that begins with “[Example]” (or “[Example n]”, where n is a positive integer) presents an example that is aimed to help readers to understand the specification, and is not a part of the specification.

A paragraph that begins with “[Rational]” (or “[Rational n]”, where n is a positive integer) contains a rational for the specification, but is not a part of the specification.

A paragraph that begins with “[Note]” (or “[Note n]”, where n is a positive integer) contains an informative note that is aimed to help readers to understand the specification, and is not a part of the specification.
1.7. VERBAL FORMS // 表現形式

The following conventions apply throughout this document:

a) the auxiliary verb ‘shall’ implies mandatory conditions.

b) the auxiliary verb ‘should’ implies optional but desirable conditions.

c) the auxiliary verb ‘may’ implies optional conditions.

d) the auxiliary verb ‘can’ implies capability or ability to do something.

e) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

The words ‘shall’, ‘should’, ‘may’ are highlighted in red and bold font.

本書では以下の決まりに従い記述する。

「…こと」「…なければならない」は、必須な仕様です。　

「…べき…」は、任意であるが推奨される仕様を示す。

「…良い…」は、許容される仕様を示す。

「…できる…」は、何かをする事が可能な事を示す。

他のパターンの記述は、事実を示す文である。

「…こと」「…なければならない」「…べき…」「…良い…」は、読者の仕様の理解の助けのため、赤字・太字で示す。

[注] 本书では、要求事項を電子的に検索しやすいように、英文の‘shall’の訳語として、「こと」を使用している。逆に、‘shall’の訳語以外では「こと」は使用せず、「事」を用いている。また、英文の‘may’に対応する訳語として、「良い」という当て字を使用している。逆に、‘may’の訳語以外で「良い」は使用していない。

「A, B, 及び C」という表記は、英文の‘A, B, and C’に対応し、「A 及び B 及び C」である事を意味する。

「A, B, または C」という表記は、英文の‘A, B, or C’に対応し、「A または B または C」である事を意味する。

英語の‘a XXX ~ the XXX’という表現に対応し、日本語は「ある XXX～その XXX」という表現を用いる。

「A, B, 及び C」という表記は、英文の‘A, B, and C’に対応し、「A, B, 及び C」である事を意味する。

「A, B, または C」という表記は、英文の‘A, B, or C’に対応し、「A, または B, または C」である事を意味する。

英語の‘a XXX ~ the XXX’という表現に対応し、日本語は「ある XXX～その XXX」という表現を用いる。
When a translation into Japanese is provided, the original English version and its Japanese translation are given in the left and right sides, respectively, in principle, as in this paragraph. In some cases, e.g., titles of sections and captions of figures/tables, the English and Japanese versions are put in a single line separated by “//” in this order (“English // Japanese”) or in separate lines with no delimiter in between (“English [Line-Break] Japanese”).

In most cases, technical terms are not translated into Japanese. The English words in alphabet remain as they are in their Japanese translation. The forms in alphabet in English which distinguish the singular and plural words remain as they are in the Japanese version to preserve the information of the quantity, although the Japanese language does not inherently distinguish the singular and plural forms.

Technical terms are highlighted basically in green and in some cases in blue. The latter consists of names of documents and protocols, widely used technical terms, and those locally used in some sections (e.g., field names). Note that the first character of an English word in a technical term is written in a capital letter, except for that in widely used technical terms.

Japaneseへの翻訳が存在する場合、原則として、この段落のように、英語を左側に示し、日本語を右側に示す。また、章や図表のタイトル等は、英語、日本語の順に一行中に//で区切る（英語//日本語）か、二行に分けて区切り文字なし（英語 [改行] 日本語）で、記述する場合もある。

多くの場合、技術用語の翻訳は行わず、英単語を維持する。そこで、日本語にもアルファベットが登場する。それらは正本である日本語文中においてもアルファベット表記される。日本語の名詞に単数形・複数形の区別はないが、単複の情報を保つため、日本語文中においても、英語の単数形・複数形の違いはアルファベットでそのまま表記する。

技術用語は読者の便のため基本的に緑字、場合により青字で示す。後者は、文書名、プロトコル名、広く用いられている技術用語、及び、本文中の一部の章にしか登場しないもの（フィールド名等）からなる。ここで、技術用語は、広く用いられているものを除き、基本的に大文字始まりの単語で表記する。
2. OVERVIEW

2.1. GENERAL // 一般

A model is a framework which represents something from a certain point of view. This document specifies a framework (hereafter referred to as the Functional Model of Spacecrafts, FMS) which represents a spacecraft from a functional point of view. This chapter provides an informative overview of the FMS specified in the subsequent chapters.

モデルとは、ある観点から何かを表現するためのフレームワークの事である。本書は、機能的な観点から衛星を表現するためのフレームワーク（以下、Functional Model of Spacecrafts, FMS と称する）を定める。本章は、引き続く章で定める FMS の概要を示す。

2.2. PURPOSE OF THIS MODEL // このモデルの目的

A function of a spacecraft in this model is an abstract representation of a job of the spacecraft in orbit, such as performing an observation or an experiment. This abstraction is made by focusing on the outcome that outside observers can see when or after a spacecraft has performed a job, disregarding how the job is performed inside the spacecraft.

本モデルにおいて、衛星の機能とは、衛星が軌道上で実施する観測や実験の実施等の仕事を抽象化して表したものである。この抽象化では、衛星が仕事を実行したときまたは後に外部のオブザーバが見る事ができる結果に注目し、その仕事を衛星内部でどのように実施するかは無視する。

The FMS specifies the functions of a spacecraft. If its functions are designed in a way that they follow the specifications of this model, a part of the methods for functional designs of different spacecrafts will be unified. Therefore, this model also serves as a guideline for the functional design of spacecrafts. In other words, this model also aims at standardizing the functional design of spacecrafts in terms of how a spacecraft is operated by observers outside the spacecraft.

FMS は、ある衛星の機能を定める。このモデルに従い機能を定める事ができるように衛星の機能を設計すれば、異なる衛星の機能設計の方法の一部が統一される事にもなる。従って、このモデルは、衛星の機能設計に対するガイドラインにもなっている。言い換えれば、本モデルは、衛星の機能設計を、衛星外部のオブザーバからの運用性という観点から標準化する事を目指している。
2.3. FUNCTIONAL OBJECT

Since functions that a spacecraft has are generally complex, they are usually specified as groups of functions according to their characteristics. The model specified in this document refers to an entity which has a group of functions of these kinds as a Functional Object (see Chapter 3). The Functional Object is the core concept in the FMS.

A Functional Object specifies how a particular job of a spacecraft appears from a point of view of observers. The concept of the Functional Object is based on the concept of the object used in object-oriented programming. In object-oriented programming, a unit of a program is defined as a class. During execution of the program, an instance (which is also called an object) is dynamically generated according to the definition of the class, and the instance executes the program. In the case of spacecrafts, the concept of object-oriented programming cannot be directly applied because a spacecraft is not software but hardware (although software is used in some parts of the spacecraft). However, the core concept is still applicable. In this model of the Functional Object, 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 the time). In summary, a Functional Object corresponds to an instance in the software terminology, but exists permanently unlike the counterpart in software.

If two or more instruments that perform the same job, such as actuators with identical specifications, are installed on a spacecraft, they are defined as two distinct Functional Objects. In this case, the definition of these multiple Functional Objects can be specified as a template for generating these Functional Objects. A template for generating multiple Functional Objects is referred to as a Functional Class (see Section 4).

Functional Objects can be monitored and controlled by other entities, i.e., entities outside the spacecraft (for example, spacecraft control systems and spacecraft test systems) and/or other Functional Objects on the spacecraft. This document does not specify methods of communications between Functional Objects and other entities (see also Section 2.5).

Functional Object は、衛星のある特定の仕事がオブジェクトの概念に基づいている。オブジェクト指向プログラミングでは、プログラムの単位をクラスとして定義する。プログラムの実行時にはクラス定義に従いインスタンス（オブジェクトともいう）が動的に生成され、そのインスタンスがプログラムを実行する。衛星の場合は、衛星はソフトウェアでなく（衛星の一部でソフトウェアを用いるにせよ）ハードウェアであるため、オブジェクト指向プログラミングの考えを直接適用はできない。しかし、中核概念は適用可能である。Functional Objectに関する本モデルでは、衛星の機能は永続的に存在するものと仮定する（ただし、ある特定の時点においてそれぞれの機能が実行可能か否かは、その時点での衛星の状態に依存する）。要約すると、Functional Objectはソフトウェアの用語のインスタンスに対応するが、インスタンスとは異なり永続的に存在する。

同一仕様のアクチュエータ等、衛星に同じ仕事を行う機器が二つ以上搭載されている場合、それは二つの別個なFunctional Objectsとして定義される。その場合、これら複数のFunctional Objectsの定義は、これらのFunctional Objectsを生成するひな型として規定できる。複数のFunctional Objectsを生成するためのひな型の事をFunctional Classと称する（4項参照）。

Functional Objects は、他の構成要素、つまり、衛星外部の構成要素（例えば衛星管制システムや衛星試験システム）、衛星上の他のFunctional Objectsの一方か双方かで監視制御できる。本書はFunctional Objects と他の構成要素の間の通信手段は定めない（2.5項も参照）。
Some entities monitor **Functional Objects** but do not control them. An entity that monitors and controls **Functional Objects** is called a **Controller** (see Section 3.1).

The concepts used to specify a Functional Object include the **Attribute**, **Operation**, **Event class**, **StateMachine**, and **Diagnostic Rule** (see Sections 3.3, 3.4, 3.5, 3.6, and 3.7, respectively).

An **Attribute** of a Functional Object is a variable representing a status of the **Functional Object**. The values of **Attributes** of a Functional Object at a given time can be obtained by other entities using telemetry messages.

An **Operation** of a Functional Object is an action performed by the **Functional Object** and is invoked from **Controllers** using a telecommand message. The values of some **Attributes** of a Functional Object can be set with **Operations** invoked by **Controllers**.

An **Event class** of a Functional Object is a classification of an **event** (a thing that has a particular significance). **Occurrences** of **events** of some **Event classes** are detected by the **Functional Object** itself. The occurrence of an **event** that is important to other entities can be reported to them using telemetry messages. The report is referred to as an **alert** and its classification is referred to as an **Alert class**.

A **StateMachine** of a Functional Object specifies how it behaves with a finite number of states and transitions between them (see Section 3.6).

A **Diagnostic rule** of a Functional Object is a rule used by other entities to diagnose whether the **Functional Object** is functioning correctly or not.

---

一部の構成要素は **Functional Objects** を監視するが制御は行わない。**Functional Objects** を監視及び制御する構成要素は **Controller** と呼ばれる（3.1 項参照）。

**Functional Object** を定めるために用いる概念には、**Attribute**, **Operation**, **Event class**, **StateMachine**, 及び **Diagnostic Rule**（それぞれ、3.3 項、3.4 項、3.5 項、3.6 項、及び 3.7 項参照）が含まれる。

**Functional Object** の **Attribute** は、その **Functional Object** の状態を表す変数である。ある時点における **Functional Object** の **Attributes** の値は、telmetry messages を用い、他の構成要素によって取得できる。

**Functional Object** の **Operation** は、その **Functional Object** が行う動作であり、telecommand messages を用い、Controllers によって呼び出される。**Functional Object** の **Attributes** の一部は、Controllers が呼び出した **Operations** により値を設定できる。

**Functional Object** の **Event class** は、**event**（特定の意味を有する出来事）の分類である。一部の **Event classes** の **events** の発生は、その **Functional Object** 自体で検出する。他の構成要素にとっては重要な **event** の発生を、それらに、telmetry messages を用い、通知する事ができる。この通知を **alert** と称し、その分類を **Alert class** と称する。

**Functional Object** の **StateMachine** は、有限個の状態とその間の移動によって、どのように動作するかを定めるものである（3.6 項参照）。

**Functional Object** の **Diagnostic Rule** は、その **Functional Object** が正常に機能しているか否かを診断するための規則であり、他の構成要素が用いる。
2.4. SPACECRAFT INFORMATION BASE 2

The definitions of the Functional Objects specified with the methods specified in this document can be registered in the Function Definition Part of the Spacecraft Information Base version 2 (SIB2) [R1]. The SIB2 is a database to manage information about a spacecraft electronically. The information about the functions that a spacecraft has should be managed uniformly, using this database, throughout all the phases concerning the spacecraft, including designing, testing, and flight operations.
2.5. COMMUNICATIONS BETWEEN FUNCTIONAL OBJECTS AND OTHER ENTITIES

A Functional Object communicates with other entities. A signal for communication to a Functional Object is referred to as a telecommand message and that from a Functional Object is referred to as a telemetry message in this model.

Although this document does not specify the methods for communications between a Functional Object and other entities, it is recommended that the Spacecraft Monitor and Control Protocol (SMCP) [R3] is used to perform the following operations (the types of telecommands / telemetries in [R3] are shown in parentheses):

- **a)** to invoke Operations of Functional Objects and to set the values of Attributes of the Functional Objects from a Controller (ACTION Telecommand or SET Telecommand),
- **b)** to report the values of Attributes of Functional Objects to other entities (VALUE Telemetry and NOTIFICATION Telemetry),
- **c)** to request reporting the values of Attributes of Functional Objects from Controllers (GET Telecommand),
- **d)** to report an alert of an Alert class to other entities (NOTIFICATION Telemetry),
- **e)** to request uploading and dumping memory data of Memory Functional Objects from Controllers (MEMORY UPLOAD Telecommand and MEMORY DUMP Telecommand, respectively), and
- **f)** to report memory data of Memory Functional Objects to other entities (MEMORY DUMP Telemetry).

The types of telecommands/telemetries in [R3] are mentioned in this document in the sections described in Table 2-1.
Table 2-1: Type of Telecommands/Telemetries in [R3]
[R3]におけるテレコマンド・テレメトリの種別

<table>
<thead>
<tr>
<th>Type</th>
<th>Section in this document</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION Telecommand</td>
<td>3.3.1 (Attributes), 3.4.1 (Operations)</td>
</tr>
<tr>
<td>SET Telecommand</td>
<td>3.3.1 (Attributes), 3.4.1 (Operations)</td>
</tr>
<tr>
<td>GET Telecommand</td>
<td>3.3.1 (Attributes)</td>
</tr>
<tr>
<td>VALUE Telemetry</td>
<td>3.3.1 (Attributes)</td>
</tr>
<tr>
<td>NOTIFICATION Telemetry</td>
<td>3.3.1 (Attributes), 3.5.2.1 (Alert classes)</td>
</tr>
<tr>
<td>MEMORY LOAD Telecommand</td>
<td>5.3.2 (Memory Functional Class, MemoryLoad)</td>
</tr>
<tr>
<td>MEMORY DUMP Telecommand</td>
<td>5.3.3 (Memory Functional Class, MemoryDump)</td>
</tr>
<tr>
<td>MEMORY DUMP Telemetry</td>
<td>5.3.3 (Memory Functional Class, MemoryDump)</td>
</tr>
</tbody>
</table>
3. METHOD OF MODELING WITH FUNCTIONAL OBJECTS

3.1. GENERAL // 一般

Functions of a spacecraft shall be specified with entities each of which is referred to as a Functional Object. A Functional Object has a group of functions that are easy to be specified and understood. The Functional Object is an abstract representation of a spacecraft function to express solely how they appear to outside entities.

The concepts used to specify how a Functional Object behaves are summarized in Figure 3-1, which include the Attribute, Operation, Event class, State Machine, and Diagnostic Rule (see Sections 3.3, 3.4, 3.5, 3.6, and 3.7, respectively).

Figure 3-1 Summary of the Concepts to Specify Functional Objects

衛星の機能を、それぞれ Functional Object と称する構成要素を用いて定めること。Functional Object は、定めるのも理解するのも容易になるようにグループ化された機能を持つ。Functional Object は、外部の構成要素からどのように見えるかのみを表し、衛星の機能を抽象化した表現である。

Functional Object がどのように動作するかを定めるのに用いる概念のサマリを Figure 3-1 に示す。これらには、Attribute、Operation、Event class、State Machine 及び Diagnostic Rule（それぞれ、3.3 項、3.4 項、3.5 項、3.6 項、及び 3.7 項参照）が含まれる。
To specify various conditions including the **Effective-Condition** of a Functional Object, Attribute or Operation (see Sections 3.2.3, 3.3.2, and 3.4.5) or the **Trigger Condition** of an Event class, an expression referred to as a **Condition Expression** is used (see Section 3.9).

A signal for communication to a Functional Object from another entity is referred to as a **telecommand message** and that from a Functional Object to another entity is referred to as a **telemetry message** in this model. An entity which sends telecommand messages is referred to as a **Controller**.

本モデルでは、ある Functional Object へ他の構成要素からの通信に用いる信号を **telecommand message**、ある Functional Object から他の構成要素への通信に用いる信号を **telemetry message** と称する。また、telecommand messages を送信する構成要素を **Controller** と称する。
3.2. FUNCTIONAL OBJECTS

3.2.1. General // 一般

An entire spacecraft shall be represented by one Functional Object which contains one or more Functional Objects. Any Functional Object of a spacecraft is either directly or indirectly contained in the Functional Object that represents the entire spacecraft.

衛星全体は一つ以上のFunctional Objectsを持ち一つのFunctional Objectで表されること。ある衛星のいかなるFunctional Objectも、その衛星全体を表すFunctional Objectに、直接的か間接的かの何れかに含まれる。

Figure 3-2 Acceptable / Unacceptable Combination of Parent-Child Relation between Functional Objects // Functional Objectsの親子関係で許容される・許容されない組み合わせ

The names of the Functional Objects follow the convention in [R1]. For xxx, a name cannot be allocated according to the convention.

Functional Objectsの名前は、[R1]の規則に従う。xxxには、規則に従い名前を割り当てる事はできない。
3.2.2. Parent-Child Relation // 親子関係

A Functional Object shall contain zero or more Functional Objects.

[Example 1] Functional Object /a in Figure 3-2 (1) contains Functional Object /a.b and Functional Object /a.c.

A Functional Object shall not be directly contained in two or more Functional Objects.

[Example 2] Functional Object d and Functional Object e in Figure 3-2 (2) contain Functional Object f, which is not allowed.

When a Functional Object contains one or more Functional Objects in it, the former is referred to as the "parent Functional Object" of the latter and the latter are referred to as "child Functional Objects" of the former. Furthermore, generalized terms of these are introduced; the Functional Objects at any upper and lower layers of the hierarchy of the parent/child relations, such as a parent of a parent (upper layer) and a child of a child (lower layer), are referred to as "ancestor Functional Object(s)" and "descendant Functional Object(s)", respectively.

Functional Object は、Functional Objects をゼロ個以上含むこと。

[例 1] Figure 3-2 (1) の Functional Object /a は、Functional Object /a.b と Functional Object /a.c を含む。

Functional Object は、二つ以上の Functional Objects には直接的に含まれないこと。

[例 2] Figure 3-2 (2) の Functional Object d と Functional Object e は、Functional Object f を含むが、これは許されない。

ある Functional Object に一つ以上の Functional Objects が含まれる場合、前者は後者の「親 Functional Object」を称し、後者は前者の「子 Functional Objects」と称する。さらに、これらを一般化した用語を導入する。親の親（上位階層）、子の子（下位階層）等、親/子関係の任意の上位階層及び下位階層にある Functional Objects は、それぞれ、「祖先 Functional Object(s)」及び「子孫 Functional Object(s)」と称する。
3.2.3. Effective-Conditions of Functional Objects (Valid/Invalid and Effective-Condition Expressions)

Functional Objects の有効条件（有効/無効及び有効条件式）

A Functional Object (except for the Functional Object representing an entire spacecraft) shall be either valid or invalid at a given time. When a Functional Object is invalid, it shall suspend all of its functions.

The condition which determines whether a Functional Object is valid or invalid at a given time is referred to as the Effective-Condition of the Functional Object.

For each Functional Object except for the Functional Object representing the entire spacecraft, its Effective-Condition shall be specified with a Condition Expression referred to as the Effective-Condition Expression.

The Functional Object representing an entire spacecraft shall be always valid. As for the other Functional Objects, one shall be valid when the evaluation result of its Effective-Condition is true and when its parent Functional Object is valid, or else it shall be invalid.
3.3. **ATTRIBUTES**

3.3.1. **General // 一般**

A variable that represents a status of a Functional Object at a given time is referred to as an Attribute of the Functional Object. The values of Attributes of a Functional Object at a given time **should** be able to be obtained by other entities using a telemetry message. A Functional Object shall have zero or more Attributes.

[Note 1] In [R3], the values of Attributes are contained in a telemetry message called a **VALUE Telemetry**. The values of Attributes are also contained in a telemetry message called a **NOTIFICATION Telemetry**.

[Note 2] In [R3], the values to be set for Attributes are contained in a telemetry message called a **SET Telecommand** or an **ACTION Telecommand**.

[Note 3] In [R3], a **VALUE Telemetry** is generated by a Functional Object spontaneously or in response to a request in a telecommand message called a **GET Telecommand** sent from a Controller.

The values of some Attributes of a Functional Object can be set by Controllers. In order for a value of an Attribute of a Functional Object to be set by a Controller, an Operation of the Functional Object shall be invoked (see Sections 3.4.2 and 3.4.4.1). Whether the value of an Attribute has been set correctly or not by a Controller can be verified by the Controller by inspecting the telemetry value of the Attribute.

Functional Object の Attributes の一部は、Controllers が値を設定できる。ある Functional Object の Attribute の値を Controller が設定するには、その Functional Object の Operation を呼び出すこと（3.4.2 項及び 3.4.4.1 項参照）。ある Controller がある Attribute の値を正常に設定したか否かは、その Controller がその Attribute のテレメトリ値を調べる事で検証できる。
3.3.2. **Effective-Conditions of Attributes (Valid/Invalid and Effective-Condition Expressions)**

An **Attribute** shall be either **valid** or **invalid** at a given time.

When an **Attribute** is **valid**, an **Attribute** shall take one of the values in the set specified for the **Attribute**.

When an **Attribute** is **invalid**, its value has no significance and cannot be reset by a **Controller**. In that case, the **Functional Object** that has the **Attribute** shall ignore requests received from **Controllers**.

A condition which determines whether an **Attribute** is **valid** or **invalid** at a given time is referred to as an **Effective-Condition** of the **Attribute**.

For each **Attribute**, an **Effective-Condition** shall be specified with a **Condition Expression** referred to as an **Effective-Condition Expression**.

An **Attribute** of a **Functional Object** at a given time shall be **valid** when the **Functional Object** is **valid** and when the evaluation result of the **Effective-Condition Expression** of the **Attribute** is **true**, or else it shall be **invalid**.

3.3.3. **Criticality Levels of values**

For the range or set of the values that each **Attribute** can take, a **Criticality Level** shall be specified. The **Criticality Levels** are used by an entity that monitors and controls a **Functional Object** to diagnose whether the **Functional Object** is functioning correctly or not.

The **Criticality Level** of a value shall be one of **Action**, **Caution**, and **Normal**.

- When the **Criticality Level** of the value of an **Attribute** of a **Functional Object** is **Action**, it shall indicate that the **Functional Object** is in danger (for example, the **Functional Object** approaches permanent failure).
- When the **Criticality Level** of the value of an **Attribute** of a **Functional Object** is **Caution**, it shall indicate that caution is needed for the **Functional Object** (for example, the **Functional Object** has temporarily ceased functioning).
- When the **Criticality Level** of the value of an **Attribute** of a **Functional Object** is neither **Action** nor **Caution**, the **Criticality Level** is **Normal**.
3.3.4. Initial Value

An Attribute should have a specific value referred to as an Initial Value. If an Attribute has the Initial Value, the Attribute shall take the value of the Initial Value when it becomes valid. If an Attribute does not have an Initial Value, which value in the specified set the Attribute takes is not predictable when it becomes valid.

3.3.5. Scalar Attributes and Complicated Attributes

An Attribute shall take one of the values in the set specified for the Attribute. An Attribute which takes a value of a simple data type (such as an enumeration type or integer) is referred to as a Scalar Attribute. An Attribute which takes a value of a more complicated data type (such as an array or structure type) is referred to as a Complicated Attribute.

[Note] Complicated data types have not been implemented in the tools of SIB2/GSTOS yet. Their proxy is under review.
### 3.3.6. Enumerative Attribute

#### 3.3.6.1. General / 一般

A Scalar Attribute whose value (integer number) is labeled with a name (identifier consisting of alphabets and numbers) is referred to as an Enumerative Attribute.

#### 3.3.6.2. Values and Enumerative Names / 値及び Enumerative Names

Enumerative Attributes shall have discrete values (such as the current mode of an instrument). For an Enumerative Attribute, the Valid Value Set, which is the set of the values that the Enumerative Attribute can take, shall be specified. For each of the values, a name referred to as an Enumerative Name shall be specified.

When an Enumerative Attribute is valid, the value of the Enumerative Attribute shall be one of the values in the Valid Value Set of the Enumerative Attribute. When an Enumerative Attribute is invalid, the value of the Enumerative Attribute may not be one of the values in the Valid Value Set of the Enumerative Attribute.

#### 3.3.6.3. Criticality Level

For a value in the Valid Value Set of an Enumerative Attribute, a Criticality Level shall be specified (See Section 3.3.3). The Criticality Level of a value is one of Action, Caution, and Normal.

[Note] Diagnosis with more complexed conditions can be specified with Diagnostic Rules (See Section 3.7).
3.3.7. **Numerical Value Attribute**

3.3.7.1. **General** // 一般

A Scalar Attribute whose values are not labeled with names is referred to as a **Numerical Value Attribute**.

A Numerical Value Attribute *shall* have either a continuous value (*e.g.* a temperature) or a discrete value (*e.g.* values of a counter: 0, 1, 2, …).

3.3.7.2. **Valid Ranges**

A pair of upper and lower limits, referred to as **Valid Range**, may be specified for a Numerical Value Attribute. If a Valid Range is specified for a Numerical Value Attribute, the Numerical Value Attribute *shall* take a value within its Valid Range (for example, due to physical limitation of the sensor that measures the value of the Attribute).

When a Numerical Value Attribute is *valid*, the value of the Numerical Value Attribute *shall* be in the Valid Range of the Numerical Value Attribute. When a Numerical Value Attribute is *invalid*, the value of the Numerical Value Attribute *may* not be in the Valid Range of the Numerical Value Attribute.

3.3.7.3. **Action Limits, Caution Limits, and Criticality Level**

Another pair of upper and lower limits, referred to as an **Action Limit**, may be specified. For a Numerical Value Attribute, another pair of upper and lower limits, referred to as a **Caution Limit**, may also be specified.

These limit values are used by an entity that monitors and controls a Functional Object to diagnose whether the Functional Object is functioning correctly or not.

When the value of a Numerical Value Attribute falls out of the range of its Action Limit, the Criticality Level of the value *shall* be defined as *Action*. When the value of the Numerical Value Attribute falls within its Action Limit, but out of the range of its Caution Limit, the Criticality Level of the value *shall* be defined as *Caution*. Otherwise, the Criticality Level of the value *shall* be defined as *Normal*.

[Note] Diagnosis with more complexed conditions can be specified with Diagnostic Rules (See Section 3.7).
3.4. OPERATIONS

3.4.1. General // 一般

An action performed by a Functional Object (e.g. to set a value to an Attribute) is referred to as an Operation of the Functional Object. A Functional Object shall have zero or more Operations.

[Note 1] Most Functional Objects have one or more Operations.

An Operation of a Functional Object shall be invoked when the Functional Object receives a telecommand message from a Controller.

[Note 2] In [R3], an Operation is invoked with a telecommand message called an ACTION Telecommand. An Operation is also invoked with a telecommand message called a SET Telecommand.
3.4.2. **Attribute Change Rules**

The rule for changes of Attribute values after an execution of an Operation is referred to as the Attribute Change Rule of the Operation.

Functional Objects should be designed in a way that the values of one or more Attributes change as a result of execution of an Operation except for the cases in which it is physically impossible to do so due to design constraints. If so designed, whether an Operation has been executed correctly or not can be verified by checking whether the Attribute values that are supposed to change as a result of the execution of the Operation have actually changed or not.

For an Operation, zero or more Attributes whose values are predictable constant values after execution of the Operation shall be specified. For each of the Attributes, a constant value after execution of the Operation shall be specified. The entity which sends a telecommand message to invoke an Operation should verify that the values of the Attributes are the specified constant values after the execution of the Operation, referring to telemetry messages.

[Note] For some Operations of a Functional Object, Current States of its State Machines (see Section 3.6.2) also change as a result of their execution. The Current States of a Functional Object are indicated by the values of the State Attributes of the Functional Object (see Section 3.6.2).

3.4.3. **No Operation (NOP)**

An Operation which accompanies no changes is referred to as No Operation (NOP).

Each Functional Object should have a NOP to check the health of the communication channel.
3.4.4. Attributes and Parameters of Operations

3.4.4.1. General // 一般

For an Operation of a Functional Object, zero or more Attributes and zero or more parameters referred to as Parameters shall be specified. A telecommand message to invoke the Operation shall contain the values to be set to the Attributes specified for the Operation and the values of the Parameters specified for the Operation. The values of the Parameters shall describe detailed information on how the Operation is executed. The Functional Object shall set the received values to the Attributes when the Operation is invoked. The entity which sends a telecommand message to invoke the Operation should verify that the values of the Attributes are the sent values after the execution of the Operation, referring to telemetry messages.

3.4.4.2. Valid Ranges

A pair of upper and lower limits, referred to as a Valid Range, may be specified for a Parameter. If a Valid Range is specified for a Parameter, Controllers should only send a telecommand message which contains a value of the Parameter within the Valid Range. Similarly, if a Valid Range is specified for an Attribute (see Section 3.3.7.2), Controllers should only send a telecommand message which contains a value of the Attribute within the Valid Range.
3.4.5. Effective-Conditions of Operations (Valid/Invalid and Effective-Condition Expressions)

An Operation shall be either valid or invalid at a given time. When an Operation is valid, the Operation shall be executed. When an Operation is invalid, Controllers (including onboard Controllers, desirably) should not send a telecommand message to execute the Operation. Controllers on the ground shall have a function to warn of sending of a telecommand message to execute an Operation when the Operation is invalid. If the Functional Object which has an Operation has received a telecommand message to execute the Operation, the Functional Object shall ignore the telecommand message.

The condition which determines whether an Operation is valid or invalid at a given time is referred to as an Effective-Condition of the Operation.

An Effective-Condition shall consist of a Condition Expression and a condition of Effective States (see Section 3.6.6). The Condition Expression is referred to as the Effective-Condition Expression.

For each Operation, an Effective-Condition Expression shall be specified.

An Operation of a Functional Object at a given time shall be valid when the Functional Object is valid, when the evaluation result of the Effective-Condition Expression of the Operation is true, and when all the State Machines which have Effective States for the Operation are in one of the States classified as the Effective States for the Operation, or else it shall be invalid.

Effective-Conditions of Operations (Valid/Invalid and Effective-Condition Expressions)

Operations の有効条件 (有効/無効及び有効条件式)

Operation は、ある時点に有効か無効の何れかである。ある Operation が有効である場合、その Operation は実行されること。ある Operation が無効である場合、Controllers（できれば衛星搭載の Controllers も含めて）はその Operation を実行するための telecommand message を送るべきではない。地上的 Controllers は、ある Operation が無効である場合、その Operation を実行するための telecommand message の送信に対して、警告をする機能をもつこと。ある Operation を持つ Functional Object は、その Operation を実行するための telecommand message を受信した場合、その telecommand message を無視すること。

ある時点で Operation が有効か無効かを決める条件を、その Operation の有効条件と称する。

有効条件は、Effective States (3.6.6 項参照) の条件と条件式からなること。この条件式を有効条件式と称する。

各 Operation に、有効条件式を定めること。

ある Functional Object のある Operation は、ある時点で、その Functional Object が有効であり、かつ、その Operation の有効条件式を評価した結果が真であり、かつ、その Operation に対する Effective States を持つ全ての State Machines がその Operation の Effective States に分類される State の何れかにある場合、有効であること。さもなくければ無効であること。
3.4.6. Criticality Level

For an Operation, its Criticality Level shall be specified. A Criticality Level shall be either prohibited, warning, or normal.

If the Criticality Level of an Operation is prohibited, the Operation shall not be executed. Controllers on the ground shall have a function to prevent sending telecommand messages to execute Operations whose Criticality Levels are prohibited. Controllers aboard a spacecraft should have a function to prevent sending telecommand messages to execute Operations whose Criticality Levels are prohibited, desirably.

If the Criticality Level of an Operation is warning, caution shall be need for execution of the Operation. Controllers on the ground shall have a function to call for attention when sending telecommand messages to execute Operations whose Criticality Levels are warning.
3.5. EVENT CLASSES

3.5.1. Event and Event classes

3.5.1.1. General 一般

A classification of an event (a thing that has a particular significance) that occurs in a Functional Object is referred to as an Event class of the Functional Object.

ある Functional Object で発生する event（特定の意味を有する出来事）の分類を、その Functional Object の Event class と称する。

A Functional Object shall have zero or more Event classes.

Functional Object は、ゼロ個以上の Event classes を持つこと。

An event shall belong to one Event class.

ある event は、一つの Event class に属すること。

Occurrences of events of some Event classes are detected by a Functional Object (see Section 3.5.2.1).

一部 Event classes の events の発生は、Functional Object で検出する（3.5.2.1 項参照）。

Event classes are used to specify Trigger classes of State Transition classes (see Section 3.6.6) and/or the Trigger classes of Alert classes (see Section 3.5.2.1).

Event classes は、State Transition classes の Trigger classes (3.6.6 項参照)、Alert classes の Trigger classes (3.5.2.1 項参照) の一方または双方の指定に用いる。

3.5.1.2. Trigger Conditions

For occurrence of an event of an Event class, an explicit condition referred to as a Trigger Condition may be specified.

ある Event class の event の発生に、Trigger Condition と称する明確な条件を定めて良い。

- If a Trigger Condition is specified for occurrence of an event of an Event class, it shall be specified by a Condition Expression. An event of the Event class shall occur when the evaluation result of its Condition Expression becomes true.

ある Event class の event の発生に、Trigger Condition を定める場合は、条件式で指定すること。

- If an explicit condition is not specified for occurrence of an event of an Event class, an event of the Event class occurs by some unidentified internal activity of the Functional Object.

ある Event class の event の発生に明確な条件を定らない場合は、その Event class の event は、Functional Object 内部の何らかの未知の活動によって、発生する。
3.5.2. **Alerts and Alert classes**

3.5.2.1. **General // 一般**

**Functional Objects** can report to other entities occurrence of an event of an Event class that is important to them. The report is referred to as an alert and its classification is referred to as an **Alert class**.

A **Functional Object** **shall** have zero or more **Alert classes**.

An alert **shall** belong to an **Alert class**.

An alert of a **Functional Object** is transferred to other entities with a telemetry message.

[Note 1] A telemetry message which transfers an alert is called a **NOTIFICATION Telemetry** in [R3].

[Note 2] In some cases, an entity can detect occurrence of an event of an Event class by monitoring the values of some relevant Attributes periodically. However, occurrence of an event is not necessarily detected by the entity because the values of the Attributes are not always delivered to the entity in a sufficiently frequent manner. By contrast, the alert reports occurrence of an event actively and more promptly.

3.5.2.2. **Trigger classes of Alerts classes**

The classification of the events which are the triggers of the reports of the alerts of an Alerts class is referred to as the **Trigger class** of the Alerts class. One Event class **shall** be specified as the Trigger class for an Alert class.

The **Functional Object** **shall** detect occurrence of an event of the Trigger class and report to other entities with an alert of the Alert class. The Functional Object which detects occurrence of events of a Trigger class may or may not be the Functional Object for which the Trigger class is specified.

Functional Objects は、他の構成要素にとり重要なEvent class のevent の発生をそれらに通知できる。この通知を alert と称し、その分類を Alert class と称する。

Functional Object は、ゼロ個以上の Alert classes を持つこと。

ある alert は、一つの Alert class に属すること。

ある Functional Object の alert は、telemetry message で他の構成要素に伝送される。

[注 1] Alert を伝送する telemetry message を、[R3] では、NOTIFICATION Telemetry と呼ぶ。

[注 2] ある構成要素は、幾つかの関連するAttributes 値を定期的に監視する事である Event class のevent の発生を検出できる場合がある。しかし、これらのAttributes の値は、その構成要素に必ずしも十分頻繁に配信されないため、必ずしもその構成要素は event の発生を検出できるとは限らない。対照的に、alert は event の発生を能動的かつより迅速に通知する。

ある Alerts class のalerts の通知のトリガとなるevents の分類を、その Alerts class の Trigger class と称する。ある Alert class に対して、Trigger class として一つの Event class を定めること。

その Functional Object は、その Trigger class のevent の発生を検出し、その Alert class の alert にて、他の構成要素に通知すること。ある Trigger classes のevents の発生を検出す Functional Object は、その Trigger classes を定めた Functional Object であっても良いし、なくても良い。
3.5.2.3. Parameters of Alerts classes
For an Alert class of a Functional Object, zero or more Attributes and zero or more parameters referred to as Parameters shall be specified. An alert classified as an Alert class shall contain the values of the Attributes and Parameters specified for the Alert class. These values of the Attributes shall be the values at the timing of occurrence of the event and the values of Parameters shall describe the detail of the event.
3.6. STATE MACHINES

3.6.1. General // 一般
A Functional Object shall include zero or more State Machines.

3.6.2. States, Names of States, and Current State
A state which a State Machine has is referred to as a State. A State Machine shall have two or more States.
A State shall have one Name (name).
The State which a State Machine currently takes is referred to as a Current State.

3.6.3. State Attributes
A Functional Object shall have one Enumerative Attribute corresponding to each of its State Machines, referred to as a State Attribute.

In Figure 3-3, the concepts associated with State Attributes and State Machines are summarized. The concepts associated with State Attributes have 1-to-1 relation with the concepts associated with State Machines. The concepts of State Attributes and the concepts of State Machines represent the same thing from different point of views.

The value which the State Attribute of a State Machine takes shall indicate the State of the State Machine.
The Enumerative Name of a value of the State Attribute corresponding to a State Machine shall be the Name of a State which corresponds to the value.

[Note 1] The Current States of the State Machines of a Functional Object determine a set of the Operations that can be invoked at the time (see Section 3.6.6.4).

[Note 2] A Criticality Level is specified for a value of a State Attribute (see Section 3.3.6.3).

[注 1] ある State Machine の State Attribute が取る値は、その State Machine の State を示すこと。
[注 2] ある State Machine に対する State Attribute の値の Enumerative Name は、その値に対応する State の Name であること。

Functional Object は、ゼロ個以上の State Machines を含むこと。
State Machine が持つ状態を State と称する。State Machine は、二つ以上の States を持つこと。
State は、一つの Name（名前）を持つこと。
ある State Machine が現在取る State を Current State と称する。
Functional Object は、その State Machines のそれぞれに、State Attribute と称する Enumerative Attribute を一つ持つこと。
Figure 3-3 に State Attributes と State Machines に関する概念を要約する。State Attributes に関する概念は、State Machines に関する概念と一対一の関係にある。State Attributes に関する概念と State Machines に関する概念は同じものを異なる観点から表現したものである。
ある State Machine に対する State Attribute が取る値は、その State Machine の State を示すこと。
ある State Machine に対応する State Attribute の値の Enumerative Name は、その値に対応する State の Name であること。
[注 1] ある Functional Object の State Machines の Current States は、その時点で呼び出す事ができる Operations の集合を決める（3.6.6.4 参照）。
[注 2] State Attribute の値には Criticality Level を定める（3.3.6.2 参照）。
The concepts associated with State Attributes and State Machines are extracted from Figure 3-1 and the relations between the concepts associated with State Attributes and those associated with State Machines are added. In this figure, compositions and 1-to-1 relations are shown in the vertical direction and generalizations are shown in the horizontal direction.
3.6.4. **Valid/Invalid // 有効/無効**

A State Machine shall be either valid or invalid at a given time. Whether a State Machine is valid or not shall be identical with that of the State Attribute (see Section 3.6.2) corresponding to the State Machine, which means some State Machines of a Functional Object are valid whenever the Functional Object is valid.

3.6.5. **Initial State**

If a State Machine is always in a specific State when the State Machine becomes valid, the State is referred to as the Initial State. Depending on whether the State Attribute for a State Machine has an Initial Value or not, the State Machine has or does not have an Initial State, respectively. If a State Machine does not have an Initial State, in which State the State Machine is when the State Machine becomes valid is not predictable.
3.6.6. Transitions and State Transition classes // 遷移及び State Transition classes

3.6.6.1. General / 一般

A pattern of transition of the State of a State Machine from one State to another State is referred to as a State Transition class.

State Machine の State が、ある State から他の State に遷移するパターンを State Transition class と称する。

A State Machine shall have one or more State Transition classes.

State Machine は、一つ以上の State Transition classes を持つこと。

Transition from one State to another State shall belong to a State Transition class.

ある State から他の State に遷移は、一つの State Transition class に属すること。

A State Transition class shall have a Begin State (source State in a transition) and an End State (target State in a transition).

State Transition class は、Begin State (遷移元の State) と End State (遷移先の State) を持つこと。

A transition of a State Transition class occurs when it is triggered by 1) execution of an Operation, 2) occurrence of an event, or 3) some internal activity of the Functional Object.

State Transition class の遷移は、1) Operation の実行、2) event の発生、または 3) 何らかの Functional Object 内部の活動、によってトリガされた際に発生する。

When a transition of a State Transition class of a State Machine occurs, the value of the State Attribute of the State Machine changes.

ある State Machine で、ある State Transition class の遷移が発生すると、その State Machine の State Attribute の値が変化する。

3.6.6.2. Trigger classes of State Transition classes

A State Transition class shall have one or more Trigger classes. A Trigger class shall be an Operation, an Event class, or Spontaneous.

State Transition class は、一つ以上の Trigger classes を持つこと。Trigger class は、Operation、Event class、または Spontaneous であることを。

- If a State Transition class has an Operation as a Trigger class, a transition of the State Transition class shall occur as a result of execution of the Operation.

- ある State Transition class が、ある Operation を Trigger class として持つ場合、その Operation の実行の結果として、その State Transition class の遷移が発生すること。

- If a State Transition class has an Event class as a Trigger class, a transition of the State Transition class shall occur as a result of occurrence of an event of the Event class.

- ある State Transition class が、ある Event class を Trigger class として持つ場合、その Event class の event の発生の結果として、その State Transition class の遷移が発生すること。

- If a State Transition class has a Trigger class which is Spontaneous, a transition of the State Transition class occurs by some unidentified internal activity of the Functional Object.

- ある State Transition class が、Spontaneous である Trigger class を持つ場合、Functional Object 内部の何らかの未知の活動によって、その State Transition class の遷移が発生する。

A State of a State Machine shall be the Initial State or reachable from at least one of the other States in the State Machine. Hence, all the States are connected by one or more State Transition classes in a State Machine.

State Machine の State は、Initial State であるかその State Machine の他の少なくとも一つの States から遷移可能であること。したがって、State Machine では、全ての States が一つ以上の State Transition classes で接続されている。
3.6.6.3. **Maximum Allowable Transition Time** and **Minimum Allowable Transition Time**

For each State Transition class in a State Machine of a Functional Object, the **Maximum Allowable Transition Time** (the maximum time that a transition is allowed to take) may be specified. If a transition of a State Transition class invoked by an Operation is not completed within its **Maximum Allowable Transition Time**, the Functional Object shall be diagnosed by other entities as not functioning correctly.

Similarly, the **Minimum Allowable Transition Time** (the minimum time that a transition is allowed to take) may be specified for each type of State Transition class. If a transition of a State Transition class invoked by an Operation is completed in less than its **Minimum Allowable Transition Time**, the Functional Object shall be diagnosed by other entities as not functioning correctly.

3.6.6.4. **Effective States for Operations**

When a State Machine has State Transition classes which have an Operation as a Trigger classes, States referred to as **Effective States** for the Operation exist for the State Machine. Here, Effective States of a State Machine for an Operation are the States which are the Begin States of the State Transition classes that have the Operation as a Trigger class.
3.7. DIAGNOSTIC RULES

A rule with which other entities diagnose whether a Functional Object is functioning correctly or not is referred to as a Diagnostic Rule. Whether a Functional Object is functioning correctly or not can be diagnosed by another entity, using a set of the Diagnostic Rules specified for the Functional Object. A Functional Object shall have zero or more Diagnostic Rules.

A Diagnostic Rule may be specified as a Condition Expression. If the evaluation result of a Condition Expression which includes Attributes of a Functional Object is true, the Functional Object shall be diagnosed to be functioning normally, or else diagnosed to be functioning abnormally.

For each Diagnostic Rule, a text message containing additional information on the diagnosis (for example, (1) the level of abnormality and/or (2) methods to handle the abnormality) shall be specified.

The simplest Diagnostic Rule takes the form that specifies a pair of the boundary values for the allowed range for a Numerical Value Attribute of a Functional Object, with which other entities check whether the value of the Numerical Value Attribute is in or out of the allowed range (see Section 3.3.7).

[Note] When a Functional Object detects an anomaly in itself, it can report the anomaly to other entities by issuing an alert (see Section 3.5). If it is not easy for a Functional Object to detect an anomaly in itself, other entities diagnose the Functional Object, using the Diagnostic Rules specified for the Functional Object.
3.8. OTHER FEATURES // その他

A Functional Object may generate data that are not specified as Attributes in the Functional Object definition as a result of executing its functions. For example, data representing results of observations or experiments (e.g., images) may be generated by Functional Objects. Functional Objects may also reference the values of Attributes that other Functional Objects have.

[Note] The Spacecraft Monitor and Control Protocol (SMCP) [R3] does not specify any methods for sending and receiving data that are not specified as Attributes.

ある Functional Object は、その機能を実行した結果として、その Functional Object の定義で Attributes として定めていないデータを生成しても良い。例えば、観察や実験の結果（画像等）を表すデータを、Functional Objects が生成しても良い。Functional Objects は、他の Functional Objects が有する Attributes 値を参照しても良い。

[注] Spacecraft Monitor and Control Protocol (SMCP) [R3] は、Attributes として定めていないデータの送受信方法を定めない。
3.9. **CONDITION EXPRESSION // 条件式**

To specify a condition of various kinds (see Section 3.1), an expression referred to as a *Condition Expression* is used. A *Condition Expression* takes a boolean value of either *true* or *false* at a given time.

A *Condition Expression* consists of terms referred to as **Comparison Terms** and operators referred to as **Logical Operators**. The precise definition of *Condition Expression* is shown in List 3-1. A *Condition Expression* shall be one of a combination of “NOT” and a *Condition Expression*, a combination of “AND” and two or more *Condition Expressions*, a combination of “OR” and two or more *Condition Expressions*, or a *Comparison Term*. Here, “NOT” is a unary **Logical Operator** and “AND” and “OR” are multi-term **Logical Operators**. A *Comparison Term* shall be a combination of an **Attribute**, an operator referred to as a **Comparison Operator**, and a **Constant**, where a **Comparison Operator** shall be one of “equal to”, “not equal to”, “greater than”, “greater than or equal to”, “less than”, and “less than or equal to”.

<table>
<thead>
<tr>
<th>Condition Expression ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison Term</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparison Term ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Comparison Operator Constant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparison Operator ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal to</td>
</tr>
</tbody>
</table>

**List 3-1 Extended BNF Definition of Condition Expression**

拡張 BNF による条件式の定義

The left side of ::= indicates the item to be defined, and the right side indicates the content of the definition. + indicates the preceding term existing one or more times. | is used to separate alternative terms.

This notation is the extended BNF (*i.e.* BNF extended with regular expression) but does not specify the syntax.

| の左辺は定義される項目、右辺は定義内容を示す。 |
| + は前項が一つ以上登場する事を示す。 |
| | は前項・次項が選択肢である事を示す。 |

この表記法は拡張 BNF（つまり、正規表現で拡張された BNF）であるが、構文は定めない。
4. FUNCTIONAL CLASS

4.1. GENERAL // 一般

If identical or similar Functional Objects are used in one or multiple spacecrafts, the common features of these Functional Objects can be specified as a template, which is referred to as a Functional Class. Conversely, a Functional Object can be specified with a Functional Class as a template.

A Functional Class specifies design parameters and Attributes, Operations, Event classes, Alert classes, State Machines, and Diagnostic Rules that are used commonly by multiple Functional Objects.

A Functional Object shall be specified by using zero or more Functional Classes as the templates and with or without additional functions (such as Attributes and/or Operations).

Functional Class is a design template that specifies common attributes, operations, event classes, alert classes, state machines, and diagnostic rules for multiple functional objects. Functional objects can be specified with one or more functional classes as templates and optionally with additional functions.

---

Figure 4-1 Acceptable Combination of Parent-Child Relation between Functional Classes/Functional Objects // Functional Classes/Functional Objects の親子関係で許容される組み合わせ
4.2. PARENT-CHILD RELATION // 親子関係

A Functional Class shall contain zero or more Functional Classes.

[Example 1]
Figure 4-1 (1): Functional Class A contains Functional Class B and Functional Class C.
Figure 4-1 (3): “Functional Object /a specified with Functional Class A as the template” has “Functional Object /a.b specified with Functional Class B as the template” and “Functional Object /a.c specified with Functional Class C as the template”.

A Functional Class may be contained in two or more Functional Classes.

[Note] A Functional Object is not contained in two or more Functional Classes (see Section 3.2.1).

[Example 2]
Figure 4-1 (2): Functional Class D and Functional Class E both contain Functional Class F.
Figure 4-1 (4): “Functional Object /d specified with Functional Class D as the template” has “Functional Object /d.f specified with Functional Class F as the template”. “Functional Object /e specified with Functional Class E as the template” has “Functional Object /e.f specified with Functional Class F as the template”. Whereas both Functional Object /d.f and Functional Object /e.f are specified with Functional Class F as the template, Functional Object /d.f and Functional Object /e.f are different Functional Objects.

4.3. STANDARD FUNCTIONAL CLASS // 標準的な FUNCTIONAL CLASS

This document specifies the following standard Functional Class:

1) Memory Functional Class (see Chapter 5).

本書では標準的な Functional Class として以下のものを定める。

1）Memory Functional Class (5 章参照)
5. MEMORY FUNCTIONAL CLASS

5.1. GENERAL // 一般

A Memory Functional Object is a Functional Object that represents a memory device (a device that stores data). The Memory Functional Class is an abstraction of the properties of all the Memory Functional Objects and specifies the design parameters and Operations that any Memory Functional Object shall or may have.

The Memory Functional Object has the design parameters and Operations specified in this chapter. A Memory Functional Object may have (1) other Operations in addition to those specified below and (2) some other characteristics of Functional Objects (such as Attributes, Event classes and/or Alert classes).
5.2. DESIGN PARAMETERS // 設計パラメータ

5.2.1. General // 一般
A Memory Functional Object shall have the following design parameters. The values of these design parameters shall be specified and fixed at the time of the designing.

1) FirstAddress
2) LastAddress
3) MaximumUploadLength (optional)
4) AlignmentLength

5.2.2. FirstAddress and LastAddress
The FirstAddress and LastAddress shall specify, respectively, the first and last addresses of the memory area which can be accessed.

5.2.3. MaximumUploadLength
The MaximumUploadLength shall specify the maximum octet length (e.g. 256 octets) of the memory data handled in one Memory Load Operation (see Section 5.3).

5.2.4. AlignmentLength
The AlignmentLength shall specify the unit for memory data access, which shall be one of the following:
- 1 octet (i.e. any address),
- 2 octets,
- 4 octets, or
- MaximumUploadLength.
5.3. OPERATIONS

5.3.1. General // 一般

A Memory Functional Object may have the following Operations. The Parameters for these Operations are given in parentheses.

1) MemoryLoad (StartAddress, MemoryData) (optional)
2) MemoryDump (NoOfDumps, StartAddress, Length) (optional)

5.3.2. MemoryLoad

The MemoryLoad is an Operation for uploading data to the memory. If the MemoryLoad is invoked, the Memory Functional Object shall write the value of the MemoryData into the memory area starting from the address specified by the StartAddress.

Note that the length of the MemoryData shall be equal to or smaller than the MaximumUploadLength, and the values
- the StartAddress and
- the octet lengths of the MemoryData
shall be multiples of the AlignmentLength.

[Note 1] The octet length of MemoryData is always the MaximumUploadLength if the AlignmentLength is equal to the MaximumUploadLength.

[Note 2] A telecommand message which invokes a MemoryLoad is called a MEMORY LOAD Telecommand in [R3].
5.3.3. **MemoryDump**

The MemoryDump is an Operation for dumping memory data. If the MemoryDump is invoked, the Memory Functional Object shall dump the memory data of the octet length specified by the Length, starting at the address specified by the StartAddress, for the number of times specified by the NoOfDumps.

[Note 1] A telecommand message which invokes a MemoryDump is called a MEMORY DUMP Telecommand in [R3].

[Note 2] A telemetry message which transfers memory data is called a MEMORY DUMP Telemetry in [R3].
Appendix A.  Acronyms // 略語集

This chapter lists the acronyms used in this document.  本章では、本書が用いる略語一覧を示す。

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSTOS</td>
<td>Generic Spacecraft Test and Operations Software</td>
</tr>
<tr>
<td>NOP</td>
<td>No Operation</td>
</tr>
<tr>
<td>SIB</td>
<td>Spacecraft Information Base</td>
</tr>
<tr>
<td>SMCP</td>
<td>Spacecraft Monitor &amp; Control Protocol</td>
</tr>
<tr>
<td>XOR</td>
<td>Exclusive OR</td>
</tr>
</tbody>
</table>
Appendix B. An Example of a Functional Object

B.1. GENERAL // 一般

In this Appendix, a Functional Object named X_A is presented as an example of a Functional Object. This Functional Object models the basic functions to control a simple instrument.

B.2. FUNCTIONAL OBJECT

A Functional Object named X_SubSystem represents a Sub-System X. Functional Object X_A is contained in the parent Functional Object X_SubSystem and specifies the functions of the instrument X-a contained in Sub-System X.

X_A contains children Functional Objects named X_A1 and X_A2, each of which represents a set of functions performed by the instrument X-a. X_A specifies the functions concerning the entire instrument.

X_A is valid only when certain Attributes of the parent Functional Object X_SubSystem meet a certain condition. The children Functional Objects X_A1 and X_A2 are valid only when the value of the State Attribute X_A_RunStop (see Section B.6) is RUN, i.e., X_A is in the State “RUN”.

X_A has Attributes and Operations for monitoring and controlling the instrument X-a as a whole. It has a State Machines that represent the rules concerning its actions. It also has an Event class, for which an Alert class is defined.
B.3. ATTRIBUTES

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 X_A_OnOff and X_A_RunStop represent the States in which the Functional Object is at a given time. Thus, they are State Attributes. The other Attributes, X_A_ErrorStatus and X_A_CheckMode, are Numerical Value Attributes.

Of the four Attributes, only the value of X_A_CheckMode can be set from the outside of the Functional Object.

X_A_OnOff is valid whenever the Functional Object is valid, whereas X_A_RunStop, X_A_ErrorStatus, and X_A_CheckMode are valid only when the value of X_A_OnOff is ON, i.e., the Functional Object is in the State “ON” (see Section B.6).
B.4. OPERATIONS

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 holds a value which is set to the Numerical Value Attribute X_A_CheckMode.

These Operations can be executed when a set of the following conditions are met. These Operations also appear as the Trigger classes of the State Transition classes of the State Machine (see 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
5) X_A_SetCheckMode X_A_OnOff=ON

X_A は、次の五つの Operations を持つ。

Operation X_A_SetCheckMode は、Numerical Value Attribute X_A_CheckMode に値を設定する値を保持する。

これらの Operations は、以下の条件が満たされたときに実行できる。また、これらの Operations は、State Machine の State Transition classes の Trigger classes としても登場する（Figure B-1 参照）。

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
5) X_A_SetCheckMode X_A_OnOff=ON
B.5. EVENT CLASSES

X_A has the following single Event class:

1) X_A_ErrorDetect.

An event of this Event class occurs when the following condition is met:

\[ \text{NOT (} X_A\text{.ErrorStatus=NORMAL) } \]

X_A can issue alerts of the following single Alert class:

1) X_A_ErrorDetected.

An alert of this Alert class is used to report to other entities the occurrence of an event of the Event class X_A_ErrorDetect and has the value of the Attribute X_A_ErrorStatus as a Parameter.
B.6. STATE MACHINES

The behavior of $X_A$ is specified with two State Machines, $X_A_{\text{OnOff}}$ and $X_A_{\text{RunStop}}$ (see Figure B-1).

The boxes show the States. The arrows show the directions of a transition of the State Transition classes. The Trigger classes are shown above the arrows. // 四角は、States を示す。矢印は、State Transition classes の遷移方向を示す。矢印の上に Trigger classes を示す。

The State Machine $X_A_{\text{OnOff}}$ takes either of the two States “OFF” and “ON”. The Initial State is OFF. The Current State of this State Machine is indicated by the State Attribute $X_A_{\text{OnOff}}$. This State Machine is valid whenever the Functional Object is valid.

The State Machine $X_A_{\text{RunStop}}$ takes either of the two States “STOP” and “RUN”. The Initial State is STOP. The Current State of this State Machine is indicated by the State Attribute $X_A_{\text{RunStop}}$. This State Machine is valid only when the value of $X_A_{\text{OnOff}}$ is ON (i.e., when the Functional Object is in the State “ON”).

Figure B-1: State Machines of Functional Object $X_A$

State Machine $X_A_{\text{OnOff}}$

\begin{align*}
\text{OFF} & \quad \text{X}_A_{\text{On}} \quad \text{ON} \\
\text{X}_A_{\text{Off}} &
\end{align*}

State Machine $X_A_{\text{RunStop}}$

\begin{align*}
\text{STOP} & \quad \text{X}_A_{\text{Start}} \quad \text{RUN} \\
\text{X}_A_{\text{Stop}}, \text{X}_A_{\text{ErrorDetect}} &
\end{align*}