Date: November 2015

Hardware Abstraction Layer for Robotic Technology (HAL4RT)

3rd Revised Submission

OMG Document Number: robotics/2015-11-01

Submission Contact: Kenichi Nakamura Japan Embedded Systems Technology Association. nakamura@upwind-technology.com Copyright © 2015, Japan Embedded Systems Technology Association. Copyright © 2015, Object Management Group, Inc.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 109 Highland Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management GroupTM, OMGTM, Unified Modeling LanguageTM, Model Driven Architecture LogoTM, Model Driven Architecture DiagramTM, CORBA logosTM, XMI LogoTM, CWMTM, CWM LogoTM, IIOPTM, MOFTM, OMG Interface Definition Language (IDL)TM, and OMG SysMLTM are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/report_issue.)

Table of Contents

| 0.1 | Copyright waiver | iv |
|--------------|--|----|
| 0.2 | Submission contact | v |
| 0.3 | IPR Mode | v |
| 0.4 | Resolution of RFP Requirements and Requests | v |
| A | 0.4.1 Mandatory requirements | |
| | 0.4.2 Non-mandatory requirements | |
| 0.5 | Responses to RFP Issues to Be Discussed | |
| | Scope | |
| | Conformance | |
| | References | |
| | Terms and Definitions | |
| | Symbols | |
| | Additional Information | |
| 6.1 | Acknowledgments | |
| 0.1 | Hardware Abstraction Layer for Robotic Technology (HAL4RT) | 3 |
| 7.1 | General | |
| 7.2 | Format and Conventions | |
| 1.2 | 7.2.1 Class and Interface | |
| | 7.2.2 Enumeration 4 | |
| 7.3 | Return Codes | 4 |
| 7.4 | Platform Independent Model (PIM) | |
| <u>∧</u> ∕.+ | 7.4.1 Overview 5 | |
| | 7.4.2 Interaction 6 | |
| | 7.4.2.1 System Interface | 6 |
| | 7.4.2.1 System metrace | |
| | 7.4.2.3 Event Interface | |
| | 7.4.3 Surface laver8 | 8 |
| | 7.4.3.1 Interface for Surface Layer | 0 |
| | 7.4.3.2 Interface for Application | |
| | 7.4.4 Message Data Structure | |
| | 7.4.5 Device layer 13 | 11 |
| | 7.4.5.1 HAL Component | 14 |
| | 7.4.5.2 Actuator 15 | 14 |
| | 7.4.5.3 MotorControlMode | 16 |
| | 7.4.5.4 Motor 16 | 10 |
| | 7.4.5.5 Sensor 17 | |
| | 7.4.5.6 Encorder 18 | |
| | | |
| | 7.4.5.7 Gyro Sensor 18 | 10 |
| | 7.4.5.8 Acceleration Sensor | |
| | 7.4.5.9 Forace Sensor | |
| | 7.4.5.10 Torque Sensor | |
| | 7.4.5.11 Tempereature Sensor | |
| | 7.4.5.12 Humidity Sensor | 19 |
| | 7.4.5.13 GPS Sensor 20 | |
| | 7.4.5.14 Direction Sensor | |
| 7.5 | Platform Specific Model (PSM) | |
| | 7.5.1 UML-to-C Transformation | |
| | 7.5.1.1 Type Definition | 22 |

| 1 | Field Code Changed | |
|----|--------------------|---------|
| 1 | Field Code Changed | |
| // | Field Code Changed | ···· |
| // | Field Code Changed | |
| // | Field Code Changed | |
| // | Field Code Changed |) |
| // | Field Code Changed | |
| // | Field Code Changed | |
| h | Field Code Changed |) |
| h | Field Code Changed | |
| Ŋ | Field Code Changed | ···· |
| Ŋ | Field Code Changed | · · · · |
| Ŋ | Field Code Changed | |
| // | Field Code Changed | |
| 1 | Field Code Changed | |
| // | Field Code Changed | |
| 1 | Field Code Changed | |
| λ | Field Code Changed | |
| Ì | Field Code Changed | |
| λ | Field Code Changed | |
| λ | Field Code Changed | |
| 1 | Field Code Changed |) |
| 1 | Field Code Changed | |
| 1 | Field Code Changed | |
| 1 | Field Code Changed |) |
| 1 | Field Code Changed |) |
| 1 | Field Code Changed | |
| - | Field Code Changed |) |
| -{ | Field Code Changed | |
| - | Field Code Changed |) |
| - | Field Code Changed | |
| - | Field Code Changed | |
| 4 | Field Code Changed | |
| 1 | Field Code Changed | |
| 1 | Field Code Changed | |
| Y | Field Code Changed | |
| J. | | |

| 7.5.1.1.1 char [ISO/IEC-9899] | Field Code Changed |
|---|------------------------|
| 7.5.1.1.2 Octet [RTC] 22 7.5.1.1.3 double [ISO/IEC-9899] | Field Code Changed |
| 7.5.2 C PSM 22 | Field Code Changed |
| 7.5.3 XML PSM 24 | Field Code Changed |
| | Field Code Changed |

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling LanguageTM); CORBA® (Common Object Request Broker Architecture); CWMTM (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

http://www.omg.org/spec

Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

- CORBA/IIOP
- Data Distribution Services
- Specialized CORBA

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

- UML, MOF, CWM, XMI
- UML Profile

Modernization Specifications

Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

- CORBAServices
- CORBAFacilities

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

OMG Domain Specifications

Signal and Image Processing Specifications

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters 109 Highland Avenue Needham, MA 02494 USA Tel: +1-781-444-0404 Fax: +1-781-444-0320 Email: <u>pubs@omg.org</u>

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt .: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

Courier - 10 pt. Bold: Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to http://www.omg.org/report_issue.htm.

0

0.1 Copyright waiver

"Each of the entities listed above: (i) grants to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version, and (ii) grants to each member of the OMG a nonexclusive, royalty-free, paid up, worldwide license to

iv

make up to fifty (50) copies of this document for internal review purposes only and not for distribution, and (iii) has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used any OMG specification that may be based hereon or having conformed any computer software to such specification."

0.2 Submission contact

Kenichi Nakamura, Japan Embedded Systems Technology Association. nakamura@upwind-technology.com

0.3 IPR Mode

Every OMG Member that makes any written Submission in response to this RFP covenants that it will grant to an unrestricted number of applicants a nonexclusive, worldwide, non-sublicensable, perpetual patent license to its Essential Claims on fair, reasonable, and non-discriminatory terms to make, have made, use, import, offer to sell, sell, and otherwise directly or indirectly distribute Covered Implementations of the resulting OMG Formal Specification.

0.4 Resolution of RFP Requirements and Requests

0.4.1 Mandatory requirements

| Requirement | Description | Reference | Remarks |
|-------------|--|------------------|-----------|
| 6.5.1 | Platform Independent Model for HAL4RT | Section 7.4 | Satisfied |
| 6.5.1 a) | Proposals shall support Device Identification. The HAL PIM shall provide a mechanism to address each device instance controlled by the HAL. | Section 7.4 | Satisfied |
| 6.5.1 b) | Proposals shall support Sensors. For sensors the HAL PIM shall support at least sensor kind id, value of measurement and unit of measurement. | Section 7.4 | Satisfied |
| 6.5.1 c) | Proposals shall support Actuators. For actuators the HAL PIM shall at least support actuator kind id, motion commands and motion feedback information. | Section 7.4 | Satisfied |
| 6.5.2 | Proposals shall provide a mechanism to associate a Device Characteristics Profile with a device kind, or with a particular device instance. | Section 7.4.5 | Satisfied |
| 6.5.3 | Proposals shall provide a Platform Independent way to specify Device Characteristics Profiles. | Section 7.4.5 | Satisfied |
| 6.5.4 | Proposals shall design a Device Kind Id Registry, which allows the association of a unique device id to each physical device model. | Section 7.4.5 | Satisfied |
| 6.5.5 | Platform Specific Model for HAL4RT Proposals shall provide one Platform Specific Model of the Hardware Abstraction Layer with at least the minimum functionality outlined in 6.5.1 a-c. This HAL PSM shall be expressed in C language. | Section 7.5 | Satisfied |
| 6.5.6 | Proposals shall provide a mapping mechanism to map the generic HAL functions (shown as HAL surface layer in Figure 6-3) to specific device functions. | Section 7.4.5 | Satisfied |

Hardware Abstraction Layer for Robotic Technology (HAL4RT), 1

v

0.4.2 Non-mandatory requirements

| Requirement | Description | Reference | Remarks |
|-------------|-------------|-----------|---------|
| 6.6 | None | | |

0.5 Responses to RFP Issues to Be Discussed

| Requirement | Description | Reference | Remarks |
|---------------|--|-----------|---------------|
| 6.7, bullet 1 | Proposals shall discuss the way to manage ID for the devices | | Not discussed |
| 6.7, bullet 2 | Proposals shall discuss the way to add a device characteristics profile for the specific devices. | | Not discussed |

1 Scope

This specification defines the Platform-Independent Model (PIM) of a Hardware Abstraction Layer for robotic systems that is capable to support at least the following devices:

Sensors. Besides the actual, normalized measurement, sensor kind and unit of measure should be provided.
 Actuators. Commands to perform motions, and motion feedback information should be provided.

In addition this specification defines the Platform Specific Model (PSM) in language C based on the HAL PIM.

This specification aims to enable engineers such as device makers, device users, and software users to build robotic software without any concern about the differences among the targeted devices, by standardizing the API of these devices.

Target readers of this specification include:

- Software engineers who use the HAL4RT to develop middleware and software.
- Device venders and its engineers who develop devices and components which conforms to the HAL4RT.
- Engineers who are interested in robot and software development.

2 Conformance

An HAL4RT implementation conforms to this specification if and only if it implements the C PSM as specified in Clause 7.3. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119

3 References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

[UML] Object Management Group, OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.5, 2015

[RTC] Robotic Technology Component specification, http://www.omg.org/spec/RTC/1.1/

[RoIS] Robotic Interaction Service, http://www.omg.org/spec/RoIS/1.0

[SMART] Smart Transducers, http://www.omg.org/spec/SMART/1.0

[ISO/IEC-9899] International Organization for Standardization, Programming languages - C, 1999

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Robotic Technology Component (RTC)

A logical representation of a hardware and/or software entity that provides well-known functionality and services.

Hardware Abstraction Layer for Robotic Technology, 1

1

Robotic Technology (RT)

Robotic Technology (RT) is a general term of the technology originating in robotics, and it means not only the standalone robot but technical element which constitutes robots.

Extensible Markup Language (XML)

A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

XML Metadata Interchange (XMI)

An OMG standard for exchanging metadata information via XML.

5 Symbols

There are no special symbols or terms.

6 Additional Information

6.1 Acknowledgments

The following organization companies submitted this specification:

 Japan Embedded Systems Technology Association (JASA)
 <u>6-71-8-12</u> Nihonbashi <u>Oden-Ha</u>macho, Chuo-ku, Tokyo, 103-00<u>11</u>07 Japan Contact: Kenichi Nakamura (nakamura@upwind-technology.com)

The following organizations contributed this specification:

- Central Information Center Co., Ltd.
- CORE CORPORATION
- Dai-ichi Seiko Co., Ltd.
- ECS Co., Ltd.
- Global Assist Co., Ltd..
- Hitachi, Ltd.
- Keio University
- NDD Corporation
- ORIENTAL MOTOR Co., Ltd.
- ROBOTEC Inc.
- TDI Product Solution Co., Ltd.
- Tokyo Metropolitan Industrial Technology Institute
- Tokyo Metropolitan University
- Upwind Technology, Inc.
- •

2

The following organization companies supported this specification:

•___Honda R&D Co., Ltd.

Hardware Abstraction Layer for Robotic Technology (HAL4RT), 1

Formatted: Default Paragraph Font
Formatted: No bullets or numbering

Formatted: Bullet_1, Numbered + Level: 1 + Numbering Style: Bullet + Start at: 1 + Alignment: Left + Aligned at: 0.1" + Indent at: 0.35", Tab stops: 0.35", Left

Formatted: Indent: Left: 0.1", No bullets or numbering

Formatted: Bullet_1, Tab stops: 0.35", Left

- Japan Robot Association
- National Institute of Advanced Industrial Science and Technology
- Shibaura Institute of Technology

7 Hardware Abstraction Layer for Robotic Technology (HAL4RT)

7.1 General

Hardware Abstraction Layer for Robotic Technology (HAL4RT) is an open standard for the implementation of robotics and control software systems.

Specifically, HAL4RT is an API (Application Program Interface) for the layer between on the first hand an application software or a middleware and on the other hand the drivers for devices such as sensor inputs, motor control commands and so on.

This standardized API increases the portability and reusability of these device drivers.

7.2 Format and Conventions

7.2.1 Class and Interface

Classes and interfaces described in this PIM are documented using tables of the following format:

Table 7.1 : <Class / Interface Name>

| Description : <description></description> | | | | | |
|---|---|---------------------------------|--|---------------------------|-----------------------------|
| Derived Fron | 1 : <parent< td=""><td>class></td><td></td><td></td><td></td></parent<> | class> | | | |
| Attributes | | | | | |
| <attribu< td=""><td>te name></td><td><attribute type=""></attribute></td><td><obligation></obligation></td><td><occurrence></occurrence></td><td><description></description></td></attribu<> | te name> | <attribute type=""></attribute> | <obligation></obligation> | <occurrence></occurrence> | <description></description> |
| | | | | | |
| Operations | | | | | |
| <pre><operation name=""> <description></description></operation></pre> | | | | | |
| <direction> <parameters< p=""></parameters<></direction> | | ter name> | <pre><parameter type=""></parameter></pre> | | <description></description> |
| | | | | | |

Note that derived attributes or operations are not described explicitly. Also, as the type of return code for every operation in this specification is Returncode, which is defined in Section7.3, Return Codes, this is omitted in the description table.

The 'obligation' and 'occurrence' are defined as follows.

Obligation

M (mandatory): This attribute shall always be supplied.

O (optional): This attribute may be supplied.

C (conditional): This attribute shall be supplied under a condition. The condition is given as a part of the attribute description.

Occurrence

The occurrence column indicates the maximum number of occurrences of the attribute values that are permissible. The followings denote special meanings.

N: No upper limit in the number of occurrences.

Hardware Abstraction Layer for Robotic Technology, 1 $\ensuremath{\mathfrak{3}}$

ord: The appearance of the attribute values shall be ordered.

unq: The appeared attribute values shall be unique.

7.2.2 Enumeration

Enumerations are documented as follows:

Table 7.2 : <enumeration name>

<constant name> <description>

7.3 Return Codes

At the PIM level we have modeled errors as operation return codes typed *ReturnCode*. Each PSM may map these to either return codes or exceptions. The complete list of return codes is indicated below.

Table 7.3: ReturnCode enumeration

| HAL_OK | The operation completed successfully. |
|---------------------|--|
| HAL_NO_CONNECTED | The target device is not connected. |
| HAL_NO_MEMORY | The target of the operation ran out of the memory needed to complete the operation. |
| HAL_NULL_PARAMETER | The parameter is not supported. |
| HAL_NOT_IMPLEMENTED | The operation is unsupported by the implementation (e.g., it belongs to a compliance point that is not implemented). |
| HAL_BAD_PARAMETER | The operation failed because an illegal argument was passed to it. |

7.4 Platform Independent Model (PIM)

7.4.1 Overview

This section specifies the PIM for service interfaces and data models. HAL4RT has two layers: "Surface layer" and "Device layer".

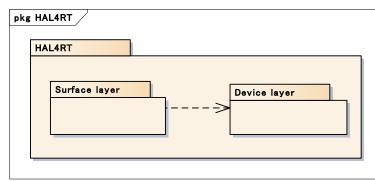


Figure 7.1 – Schematic structure of HAL4RT

The Surface layer provides standardized API (Application Program Interface) to application software and middleware. The Surface layer will so enable software developers to build application software and middleware without any concern about the differences among device interfaces she or he uses.

The Device layer consists of HAL4RT components. Application (including middleware) developers do not need to be aware of the presence of the Device layer.

Device suppliers and manufacturers provide HAL4RT component to their customer along with their actuators or sensors. The Device layer serves to bridge the gap between the API of the Surface Layer and the actual operation of the device. [OPTIONAL] In addition, the Surface layer is not required because HAL4RT aims "light and compact" specification.

7.4.2 Interaction

7.4.2.1 System Interface

System Interface is the interface to notify the status of the device to the application program.

For examples, the following sequence diagram shows the behavior when the application program receives the status of the actuator by using system interface.

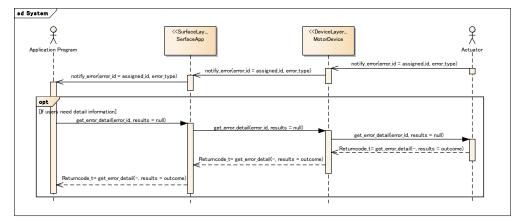


Figure 7.2 – Sequence Diagram of System Interface (System Error)

When the error occurred in the actuator, the application receives the asynchronous error notification by using notify_error(). notify_error() notify the error_id and the error_type. The application program can know the detail of the error by using get_error_detail() and the error_id.

7.4.2.2 Command Interface

Command interface is the interface to execute the commands of the device from the application program.

For the examples, the following sequence diagram show the behavior to send the position command to the actuator, and to receive the position information from the actuator by using Command interface.

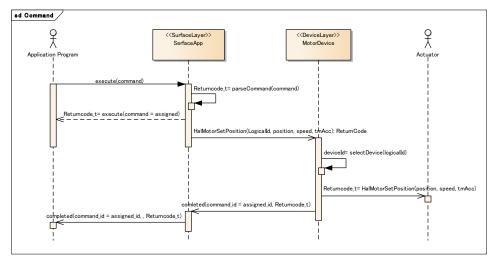


Figure 7.3 – Sequence Diagram of Command Interface (Motor position control)

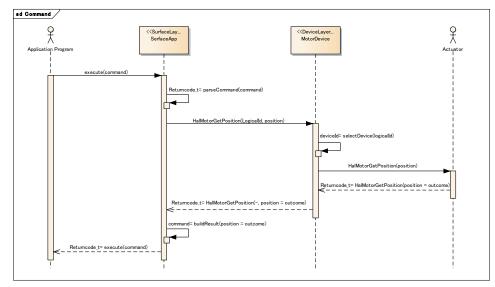


Figure 7.4 – Sequence Diagram of Command Interface (Get current motor position)

Hardware Abstraction Layer for Robotic Technology, 1

The application program specifies the command of the target device by using execute() in the surface layer. The surface layer parse the command and select the element in the device layer.

When the command is asynchronous, the surface layer returns command_id to the application program. After the command finished execution in the device, the device layer call completed() and notifies to the application program.

On the other hand, when the command is synchronous, the surface layer calls <u>prossesingprocessing</u> in the device layer. The device <u>layer selectlayer selects</u> the target device, and calls the command. The surface layer returns the return value to the application program.

7.4.2.3 Event Interface

Event Interface is the interface to receive the notification of changing state of the device. This interface uses subscribe/unsubscribe to register/unregister the event.

The following sequence diagram shows the example.

| eckage] HAL4RT [Event] | | |
|--|---|--|
| | | viceLayes>> torDevice |
| subscribe(device_kind, event_type, subscribe_id) | | |
| Betum Code= subscribe(subscribe_id=assigned_id) | | subscribe/overt.type.subscribe.id) |
| | | |
| Opt [When target event occurred] | petify_event(event_id=assigned_id, event_type, subscribe_id=assigned_id | potify event(event.jd=assigned.jd, event_type, subscribe.jd=assigned.jd) |
| potify_event(event_id=assigned_id_event_type, subscribe_id=assigned_i | id) | |
| opt []f uters need detail information] get.event.detail(event.id, results=null) | Ret_event_detail(event_id, results=null) | get event detai(event id. results=null) |
| ReturnCode= get_event_detail(results=codcome) | | |
| break | | |
| [Step event detection] unsubscribe(subscribe_id=assigned_id) | unsubscribe(subscribe_id=assigned_id) | unsubscribe(subscribe_id=assigned_id) |
| Return:Ocde: unsubscribe() | | |
| | | |

Figure 7.5 – Sequence Diagram of Event Interface

The application program registers the event by using subscribe(). The device returns the result and subscribe_id.

When the event occur<u>sred</u>, the device notifies the application program and sends event_id by using notify_event(). The application program can <u>find outknow</u> the <u>detaioldetail</u> of the event using get_event_detail() and event_id.

The application calls unsubscribe() with subscribe_id to cancel the notification of the event.

7.4.3 Surface layer

The surface layer is the specification of <u>the Application Programming Interface</u> for the application programmer who uses HAL4RT devices. The surface layer parses the command from the application program, <u>getsknow</u> the device kind, <u>and</u> generates the logical id.

The following diagram is the PIM of the surface layer.

8

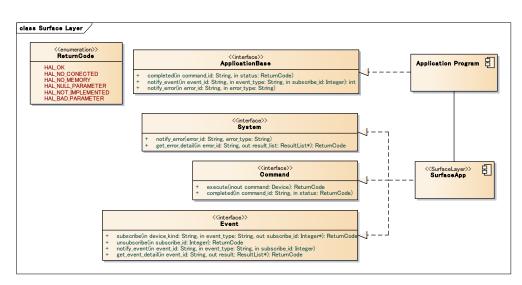


Figure 7.6 – Surface Layer

7.4.3.1 Interface for Surface Layer

The details of System Interface, Command Interface and Event Interface in surface layer are below. Table 7.4: System Interface

| - | Description : The device notifies the error to the application program. Or, the application program gets the detail of the error. | | | | |
|------------------------|--|-----------|--------------------------------------|--|--|
| Derive | d From: None | | | | |
| Operat | tions | | | | |
| notify_error Notify th | | Notify th | ne error to the application program. | | |
| in | error_id | | String | Id information to distinguish the error. | |
| in | n error_type | | String | Information to distinguish the kind of the error. | |
| get_erre | get_error_detail Get the d | | etail of the error. | | |
| in | n error_id | | String | Id information to distinguish the error that was notified by notify_error. | |
| out | result_list | | ResultList | The detail information of the error. | |

Table 7.5: Command Interface

| Descri | Description: The application program calls the command of the device. | | | | |
|--------|---|--|-----------------------------------|--|--|
| Derive | Derived From: None | | | | |
| Opera | tions | | | | |
| execut | execute | | xecute the command of the device. | | |
| inout | command | | Device | The kind of the target device, the detail of the command, the result of the command. | |
| compl | completed No | | y the completion of the comm | and. | |
| in | in command_id | | String | ID information of the command. | |
| in | status | | ReturnCode | The result of the command. | |

Table 7.6: Event Interface

| Des | Description: The device notifies the event to the application program. | | | | |
|--------------------|--|---------------------|---|--|--|
| Der | ived From: None | | | | |
| Operations | | | | | |
| subs | cribe | Register the even | | | |
| in | device_kind | String | The kind of the device. | | |
| in | event_type | String | The type of the event. | | |
| out | subscribe_id | Integer | ID information of the registered event. | | |
| unsu | Ibscribe | Unregister the ev | ent. | | |
| in | subscribe_id | Integer | ID information of the registered event. | | |
| get_ | event_detail | Get the detail of t | ne event. | | |
| in | event_id | String | ID information of the event. | | |
| out | results | ResultList | The detail information of the event. | | |
| notify_event Notif | | Notify the event. | | | |
| in | event_id | String | ID information of the event. | | |
| in | event_type | String | The type of the event. | | |
| in | subscribe_id | String | ID information of the registered event. | | |

7.4.3.2 Interface for Application

The detail of <u>the</u> interface for the application program.

Table 7.7: Application Base Interface

| | Description : The application program implementprogram implements this interface to receive the information from the device. | | | | | | | | | |
|--------------------|---|--------|---------------------------------|---|--|--|--|--|--|--|
| Derived From: None | | | | | | | | | | |
| Oper | ations | | | | | | | | | |
| comp | oleted | Notify | completion of the process in | the device. | | | | | | |
| in | command_id | | String | The command ID. | | | | | | |
| in | status | | ReturnCode | The result of command. | | | | | | |
| notif | y_event | Notify | Notify the event in the device. | | | | | | | |
| in | event_id | • | String | ID information of the event. | | | | | | |
| in | event_type | | String | The type of the event. | | | | | | |
| in | subscribe_id | | Integer | ID information of the registered event. | | | | | | |
| notif | y_error | Notify | y the error in the device. | | | | | | | |
| in | in error_id | | String | ID information of the error. | | | | | | |
| in | error_type | | String | The type of the error. | | | | | | |

7.4.4 Message Data Structure

When the application program executes the command of the device, it uses execute() in the command interface. The following diagram shows the data structure of the message that is the argument of execute().

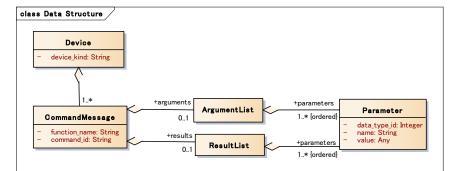


Figure 7.7 – Data Structure of Message.

The following tables are the detailed definition of each class.

Table 7.8: Device

Hardware Abstraction Layer for Robotic Technology, 1 $11\,$

| Description: Data typ | be to distinguish the | device. | • | |
|-----------------------|--------------------------|---------------------------------------|----------|--|
| Derived From : None | e | | | |
| Attributes | | | | |
| device_kind | String | М | 1 | The kind of the device. For examples, Motor, Encoder, Gyro sensor, Acceleration sensor, Force sensor, Torque sensor, Temperature sensor, Humidity sensor, GPS sensor, Direction sensor etc. |
| | | Tab | ole 7.9: | Command Message |
| Description: Data typ | be to specify the API | and to | keep ti | he result. |
| Derived From : None | e | | | |
| Attributes | | | | |
| function_name | String | М | 1 | The name of API. |
| command_id | String | М | 1 | The ID information to distinguish the CommandMessageThe element of the surface layer select the ID and notify the application program. In the asynchronous command, this is used to notify the result. |
| arguments | ArgumentList | 0 | 1 | Parameter information for the command. |
| results | ResultList | 0 | 1 | Parameter information for the result of the command. |
| | | Т | able 7. | 10: ArgumentList |
| Description: Data typ | be to keep the parameter | eter. | | |
| Derived From : None | e | | | |
| Attributes | | | | |
| parameters | Parameter | М | Nord | Parameter information. |
| | | | Table | 7.11: ResultList |
| Description: Data typ | be to keep the result. | | | |
| Derived From : None | e | | | |
| Attributes | | | | |
| parameters | Parameter | М | Nord | The information of the result. |
| | L. | | Table | 7.12: Parameter |
| Description: Data typ | be to send the parame | eter an | d to kee | pp the result. |
| Derived From : None | e | | | <u>-</u> |
| Attributes | | | | |
| data_type_id | Integer | М | 1 | The ID of data type. |
| name | String | М | 1 | Parameter name. |
| value | Any | М | 1 | Parameter value. |
| | | · · · · · · · · · · · · · · · · · · · | | |

7.4.5 Device layer

The device layer is the specification of API for the device manufacturer. In the same kind of device, it hides the differences between manufacturers, models etc.. It defines the common API for each device type. It converts the actual device ID to the logical ID. The following diagram shows PIM of the device layer.

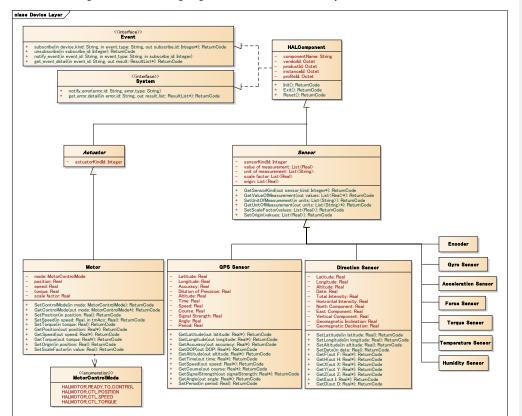


Figure 7.8 – Device Layer

Products conforming to HAL4RT, for each affected device must implement all thethe entire API. If the nature of the product, an API that does not function exists, it will implement all of the API. (A motor which is corresponding to only the speed control in, or if it can not support the torque control API · Position Control API) And, non-support API will be implemented as you plan to return the "API unimplemented error (HAL_NOT_IMPLEMENTED)" as the API of the return value.

Event Interface and System Interface is an interface that is required only when you are using a surface layer. If it is omitted the surface layer is omitted, it is not necessary to implement these interfaces.

Hardware Abstraction Layer for Robotic Technology, 1 $13\,$

7.4.5.1 HAL Component

HAL Component is an element to keep the common definition for all components in the device layer.

| Derived From: N | one | | | | | | | |
|-------------------------------|---|---|---|--|--|--|--|--|
| Attributes | Attributes | | | | | | | |
| componentName | String | М | | Component name. 固有の名前 | | | | |
| vendorId | Octet [RTC] | М | 1 | Manufacturer ID. Manufacturer provides a component for the device. | | | | |
| productId | Octet [RTC] | М | 1 | Product ID defined by manufacturer. | | | | |
| instanceId | Octet [RTC] | М | 1 | Instance ID for some same devices. | | | | |
| profileId | Octet [RTC] | М | 1 | ID for the kind of the device. | | | | |
| Operations | | | | | | | | |
| Init Initialize HAL Component | | | | | | | | |
| Exit Finish HALComponent. | | | | | | | | |
| Reset | Reset Reset the error status and recover. | | | | | | | |

componentName

Each HAL component is identified by a Profile ID (See the section 7.2.2), a Vendor ID (See the section 7.2.3) and a Product ID (See the section 7.2.4), and has a unique HAL component name.

Hal + Profile name + Vender Name + Product Name

vendorId

The Vendor names and Vendor IDs identify the device suppliers and manufacturers. A Vendor ID is defined as a 32bit unsigned integer type, with value between 0x00000000 and 0xFFFFFFF. The Vendor names and Vendor IDs are maintained by the OMG. The Vendor names is <u>a 2</u> to 16-character string beginning with an uppercase letter.

productId

The Product names and Product IDs identify the products of the device suppliers and manufacturers. A Product ID is defined as a 32bit unsigned integer type, with values between 0x00000000 and 0xFFFFFFFF. The Product names and Product IDs are defined by device suppliers and manufacturers

instanceId

An Instance ID identifies a specific device. An Instance ID is defined as a 32bit unsigned integer type, with values between 0x00000000 and 0xFFFFFFF. The Instance IDs are defined by the application developers.

profileId

An Profile ID identifies the kind of the device.

Device kind ID Registry

A register to manage Device Kind ID, Device Name, componentName, vendorId, productId, instanceId and profileId.

For examples,

Device Kind ID, Device Name

0x00000001, Motor

0x0000002, Encoder

0x0000003, GyroSensor

14

0x00000004, TorqueSensor

Vendor ID, Product ID, Component Name 0x00000001, 0x00000001, AAA 0x00000002, 0x00000001, BBB

Device Characteristics Profile

A profile that will to have the kind of the device, command name, command ID, and parameters.

Device Characteristics Profile Definition

A register to have componentName, vendorId, productId, instanceId and profileId.

State Machine

The following diagram shows State Machine of HAL Component. State Machine of the device is defined in "Active".

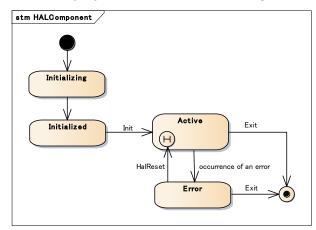


Figure 7.9 - State Machine of HAL Component

The detail of each state is below.

• Initializing : The status to initialize the device. The application program <u>can not</u> call the methods of HAL Component.

- · Initialized : The status after initializing of the device. The application program can call only Init.
- · Active : The state that HAL Component is working. The application program can call all API except Init and Reset.
- Error : The state that HAL Component is stopping. The application program can call Reset and Exit.

7.4.5.2 Actuator

Actuator is an element that has the common definition of the actuator.

Derived From: HAL Component

Hardware Abstraction Layer for Robotic Technology, 1 $15\,$

| Attributes | | | | | | | | |
|-------------------|---------|---|---|---|--|--|--|--|
| actuatorKind | Integer | М | 1 | ID that shows the kind of the actuator. | | | | |
| Operations : None | | | | | | | | |

7.4.5.3 MotorControlMode

| MotorControlMode is an enumeration to distinguish the control mode of the motor. | | | | | | | |
|--|------------------------|--|--|--|--|--|--|
| HALMOTOR_READY_TO_CONTROL | No power mode. | | | | | | |
| HALMOTOR_CTL_POSITION | Position control mode. | | | | | | |
| HALMOTOR_CTL_SPEED | Speed control mode. | | | | | | |
| HALMOTOR_CTL_TORQUE | Torque control mode. | | | | | | |

7.4.5.4 Motor

Motor is an abstract element that shows 1 DOF_(degree of freedom) motor including linear and rotary. HAL4RT doesn't support Multiple DOF motors.

| | Deriv | ved From: | Actuator | | | | | | | |
|---|-------------------|--------------------------|------------|--------------------------------|--------------------------------|--|--|--|--|--|
| | Attri | butes | | | | | | | | |
| | mode | ; | MotorCo | ntrolMode | М | 1 | The control mode of the motor. | | | |
| | positi | ion | Real | | М | 1 | The current position/angle. | | | |
| | speed | 1 | Real | | М | 1 | The current velocity/angular velocity. | | | |
| | torqu | e | Real | | М | 1 | The current force/torque. | | | |
| | scale factor Real | | | М | 1 | Unit(the scale of position/angle of output shaft) Rotating system : The value per round. The initial value is_2_ <u>pi</u> 朔. Linear system : The value per meter. The initial value is1.0. | | | | |
| | Oper | ations | I | | | | Encar system . The value per meter. The initial value is t.o. | | | |
| | SetCo | ontrolMode | • | Set control m | Set control mode of the motor. | | | | | |
| | in | in mode MotorControlMode | | | | М | Control mode. | | | |
| | GetC | ontrolMode | e | Get control n | Get control mode of the motor. | | | | | |
| | out | mode | Mot | orControlMode | | М | The current control mode. | | | |
| I | SetPo | osition | | | | | rget position/angle. If the control mode is not position control n control mode. | | | |
| | in | position | Rea | | | М | The target position. Unit: scale factor. | | | |
| | SetSp | beed | | | | | ty/angular velocity in the specified acceleration/deceleration time. of speed control mode, change to speed control mode. | | | |
| | in | speed | speed Real | | | М | The target speed. Unit:scale factor/second. | | | |
| | in | tmAcc | Rea | [| | М | Acceleration/Deceleration time to the target speed. Unit:second. | | | |
| | SetTo | orque | | Output the sp torque contro | | | que. If the control mode is not torque control mode, change to | | | |

| in | torque | Real | | М | The target torque. Unit:[Nm] or [N]. |
|------|---------------------------|---------------------------------|--------------------|---------|--|
| GetP | osition | Get the current position/angle. | | | |
| out | position | Real | | М | The current position/angle. Unit:scale factor. |
| GetS | GetSpeed Get th | | | elocity | /angular velocity. |
| out | speed | Real | Real | | The current speed. Unit:scale factor/second |
| GetT | GetTorque Ge | | Get the current to | orque. | |
| out | torque | Real | | М | The current torque. Unit:[Nm] or [N] |
| SetO | SetOrigin Set the current | | | osition | as the origin. |
| in | position | Real | | М | The target position. Unit:[rad] or [m] |

State Machine

The following diagram shows State Machine of Motor. It shows the State Machine in the Active state of HAL4RT Component.

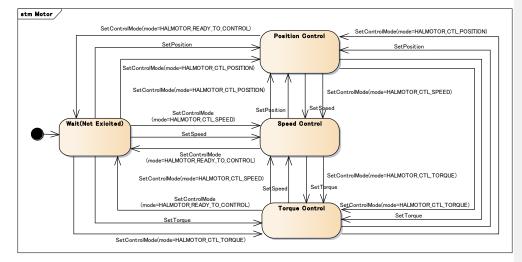


Figure 7.10 - State Machine of Motor

The details of each state are below.

- Wait(Not Excited) : Standby state that motor body is not energized.
- Position Control : The state running position control. If SetPosition is called, change to this state automatically.
- Speed Control : The state running speed control. If SetSpeed is called, change to this state automatically.
- Torque Control : The state running torque control. If SetTorque is called, change to this state automatically.

[OPTIONAL] It is not necessary to implement all control modes.

7.4.5.5 Sensor

Sensor is an abstract element that has the common definition of the sensor.

Derived From: HAL Component

Hardware Abstraction Layer for Robotic Technology, 1

17

| Attr | ibutes | | | | | |
|-----------------------------------|---------------------|---|--------|---------|---|---|
| sensorKindId Intege | | er | М | 1 | ID information to distinguish the kind of the sensor. | |
| valu | e of measurement | Real | | М | Nord | Value of measurement of the sensor. |
| unit | of measurement | String | g | М | Nord | Unit of measurement. |
| scale | e factor | Real | | М | Nord | Factor to convert the measurement content of the sensor to the specified measurement information. |
| origi | n | Real | | М | Nord | The origin of each measurement |
| Ope | rations | | | | | |
| GetS | ensorKind | | Get th | ne kino | d of the | sensor. |
| out | sensor_kind Integer | | | | М | The kind of the sensor. |
| GetValueOfMeasurement Get the val | | | | | ie of me | asurement of the sensor. |
| out values List <real></real> | | | | М | Value of measurement of the sensors. | |
| SetU | InitOfMeasuremen | nt | Set th | e unit | of sense | pr measurement. |
| in | units | List <st< td=""><td>ring></td><td></td><td>М</td><td>Unit of sensor measurement.</td></st<> | ring> | | М | Unit of sensor measurement. |
| Getl | JnitOfMeasureme | nt | Get th | ne unit | of sens | or measurement. |
| out | units | List <st< td=""><td>ring></td><td></td><td>М</td><td>Unit of sensor measurement.</td></st<> | ring> | | М | Unit of sensor measurement. |
| SetS | caleFactor | | Set th | e scal | e factor | of the value of measurement of the sensor. |
| in | values | List <re< td=""><td>eal></td><td></td><td>М</td><td>Scale factor of the value of measurement of the sensor.</td></re<> | eal> | | М | Scale factor of the value of measurement of the sensor. |
| GetScaleFactor Get the scale fa | | | | | e factor | of the value of measurement of the sensor. |
| out | values | List <re< td=""><td>eal></td><td></td><td>М</td><td>Scale factor of the value of measurement of the sensor.</td></re<> | eal> | | М | Scale factor of the value of measurement of the sensor. |
| SetC | Drigin | | Set th | e orig | in of the | sensor. |
| in | values | List <re< td=""><td>eal></td><td></td><td>М</td><td>Origin of the sensor.</td></re<> | eal> | | М | Origin of the sensor. |
| | | | | | | |

7.4.5.6 EncorderEncoder

| EncorderEncoder is a sensor to measure the position / ang | gle. |
|---|------|
|---|------|

| Derived From: Sensor | |
|----------------------|--|
| Attributes : None | |
| Operations : None | |
| | |

7.4.5.7 Gyro Sensor

| Gyro Sensor is a sensor to measure angular velocity. |
|--|
| Derived From: Sensor |
| Attributes : None |
| Operations : None |
| |

7.4.5.8 Acceleration Sensor

Acceleration Sensor is a sensor to measure acceleration.

| Derived From: Sensor | |
|----------------------|--|
| Attributes : None | |
| Operations : None | |

7.4.5.9 ForaceForce Sensor

Force Sensor is a sensor to measure force.

Derived From: Sensor Attributes : None

Operations : None

7.4.5.10 Torque Sensor

Torque Sensor is a sensor to measure torque.
Derived From: Sensor
Attributes : None
Operations : None

7.4.5.11 Tempereature Temperature Sensor

Tempereature Temperature Sensor is a sensor to measure temperature.

Derived From: Sensor Attributes : None

Operations : None

7.4.5.12 Humidity Sensor

Humidity Sensor is a sensor to measure humidity.

Derived From: Sensor

Attributes : None

Operations : None

Hardware Abstraction Layer for Robotic Technology, 1 $19\,$

7.4.5.13 GPS Sensor

GPS Sensor is a sensor to measure latitude and longitude, or heading and distance to GPS position.

| or b benot to a benot to incusate matade and iongraded of newing and assume to or b position | | | | | | | | |
|--|--|------------|----------------------|--|---------------------------------------|--|--|--|
| Derived From: Sensor | | | | | | | | |
| Attributes | | | | | | | | |
| latitude | Real | | М | 1 | Latitude | | | |
| longitude | Real | | М | 1 | Longitude | | | |
| accuracy | Real | | М | 1 | Accuracy | | | |
| dilution of precision | | Real | | 1 | Dilution of precision (DOP) | | | |
| altitude | Real | | М | 1 | Altitude | | | |
| time | Real | | М | 1 | Time | | | |
| speed | Real | | М | 1 | Speed | | | |
| course | Real | | М | 1 | Course | | | |
| signal strength | Real | | М | 1 | Signal strength | | | |
| angle | Real | | М | 1 | Angle | | | |
| period | Rea | 1 | М | 1 | Period | | | |
| Operations | | | | | | | | |
| GetLatitude | | Get the la | titude, | in degre | ees. | | | |
| out latitude | Re | eal | | М | Latitude | | | |
| GetLongitude | gitude Get the lo | | ngitude, in degrees. | | | | | |
| out longitude | Re | eal | | М | Longitude | | | |
| GetAccuracy | | Get the es | timate | d accura | acy of this location, in meters. | | | |
| out accuracy | Re | eal | | М | Accuracy | | | |
| GetDOP | | Get the va | alue of | DOP. | | | | |
| out DOP | | Real | | | Dilution of precision (DOP) | | | |
| GetAltitude | itude Get the altitude if available, in meters above the WGS 84 reference ellipsoid. | | | | | | | |
| out altitude F | | eal | | | Altitude | | | |
| GetTime Return the | | e UTC | time of | this fix, in milliseconds since January 1, 1970. | | | | |
| out time | | Real | | | Time | | | |
| GetSpeed Get t | | Get the sp | eed if | it is ava | ilable, in meters/second over ground. | | | |
| out speed | | eal | | М | Speed. Unit : [m/s] | | | |
| | | Get the co | ourse if | it is av | ailable, in 000.0–359.9 degrees. | | | |
| out course R | | Real | | | Course. Unit : [degree] | | | |
| GetSignalStrength | | Get signal | l streng | gth. | | | | |
| out signal strength | Re | eal | | М | Signal strength | | | |
| • • | | Get angle | | | | | | |
| out angle | | eal | | М | Angle | | | |
| SetPeriod | | Set period | | | | | | |
| in period | | | | М | Period | | | |
| | | | | | | | | |

7.4.5.14 Direction Sensor

Direction Sensor is a sensor to measure direction.

| | tion Sensor is a se ved From: Senso | | neasure u | | | |
|----------------------------|--|--------|-----------|-----|--|---|
| - | ibutes | | | | | |
| latitu | ıde | Rea | l M | [| 1 | Latitude90.00 to +90.00 degrees |
| long | itude | Rea | l M | [| 1 | Longitude180.00 to +180.00 degrees |
| altitu | ıde | Rea | l M | [| 1 | Altitude. referenced to the WGS 84 ellipsoid OR the Mean Sea Level (MSL) |
| date | | Rea | l M | [| 1 | Date. 2015.0 to 2020.0 |
| total intensity | | | l M | M 1 | | F - Total Intensity of the geomagnetic field |
| horizontal intensity | | | l M | [| 1 | H - Horizontal Intensity of the geomagnetic field |
| north component | | | l M | [| 1 | X - North Component of the geomagnetic field |
| east | component | Rea | l M | [| 1 | Y - East Component of the geomagnetic field |
| verti | cal component | Rea | l M | [| 1 | Z - Vertical Component of the geomagnetic field |
| geon | nagnetic inclination | on Rea | l M | [| 1 | I (DIP) - Geomagnetic Inclination |
| geomagnetic declination | | Rea | l M | [| 1 | D (DEC) - Geomagnetic Declination (Magnetic Variation) |
| Ope | rations | | | | | |
| SetL | atitude | Set | latitude. | | | |
| in | latitude | Real | | | М | Latitude |
| SetL | ongitude | Set | longitude | | | |
| in | longitude | Real | eal | | | Longitude |
| SetAltitude Set altitude. | | | | | | |
| in | in Altitude Real | | | | М | Altitude |
| SetD | ate | Set | date. | | | |
| in | date | Real | eal | | | Date |
| GetF Get F. | | | | | | |
| out | F | Real | eal | | | F - Total Intensity of the geomagnetic field |
| GetH | I | Get | H. | | | |
| out H Real | | | | М | H - Horizontal Intensity of the geomagnetic field | |
| GetX Get X. | | | | | | |
| out X Real | | | | М | X - North Component of the geomagnetic field | |
| GetY | 7 | Get | Υ. | | ÷ | |
| out | Y | Real | | | М | Y - East Component of the geomagnetic field |
| GetZ Get Z. | | | | | | |
| out | Z | Real | teal | | | Z - Vertical Component of the geomagnetic field |
| GetI | | Get | I. | | | · |
| out I Real | | | | | М | I (DIP) - Geomagnetic Inclination |
| GetI |) | Get | D. | | | · |
| out period Real | | | | М | D (DEC) - Geomagnetic Declination (Magnetic Variation) | |

7.5 Platform Specific Model (PSM)

This section specifies the PSM for HAL4RT. HAL4RT offers only one PSM, which is based on the ISO/IEC 9899:1999 Programming Language C (also known as C99).

7.5.1 UML-to-C Transformation

7.5.1.1 Type Definition

7.5.1.1.1 char [ISO/IEC-9899]

String is mapped to char.

7.5.1.1.2 Octet [RTC]

Octet is mapped to int32_t.

7.5.1.1.3 double [ISO/IEC-9899]

Real is mapped to double.

7.5.2 C PSM

/* HAL4RT_Surface.h */

```
#ifndef HAL4RT_SURFACE_H
#define HAL4RT_SURFACE_H
```

```
int32_t completed();
int32_t notify_event();
int32_t notify_error();
int32_t get_error_detail();
int32_t execute();
int32_t completed();
int32_t subscribe();
int32_t unsubscribe();
int32_t get_event_data();
typedef struct ApplicationBase {
  int32_t completed();
  int32_t notify_event();
  int32_t notify_error();
} APPLICATIONBASE;
typedef struct System {
  int32_t notify_error();
  int32_t get_error_detail();
} SYSTEM;
typedef struct Command {
  int32_t execute();
  int32_t completed();
} COMMAND;
```

22

```
int32_t subscribe();
int32_t unsubscribe();
  int32_t notify_event();
  int32_t get_event_data();
} EVENT;
/* HAL4RT_Device.h */
#ifndef HAL4RT_DEVICE_H
#define HAL4RT_DEVICE_H
typedef struct HALComponent {
  char ComponentName[32];
  uint32_t Vendor_ID;
  uint32_t Product_ID;
uint32_t Instance_ID;
  uint32_t Profile_ID;
} HALCOMPONENT;
typedef struct Actuator {
  uint32_t ActuatorKindId;
} ACTUATOR
typedef struct Sensor {
  uint32_t SensorKindId;
  int32_t ValueOfMeasurement[32];
  int32_t ValdeOnneasure
int32_t ScaleFactor[32];
int32_t Origin[32];
  int32_t GetSensorKind();
  int32_t GetValueOfMeasurement();
  int32_t SetUintOfMeasurement();
  int32_t GetUnitOfMeasurement();
  int32_t SetScaleFactor();
  int32_t SetOrigin();
} SENSOR
#endif HAL4RT_DEVICE_H
```

typedef struct Event {

7.5.3 XML PSM

```
The below is XML schema to express Message Data Structure.
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="CommandMessage" type="CommandMessage"/>
        <xs:complexType name="CommandMessage">
                <xs:sequence>
                        <xs:element name="function_name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="command_id" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="arguments" type="ArgumentList" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="results" type="ResultList" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="ArgumentList" type="ArgumentList"/>
        <xs:complexType name="ArgumentList">
                <xs:sequence>
                        <xs:element name=" p arameters" type="Parameter" minOccurs="1"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="Parameter" type="Parameter"/>
        <xs:complexType name="Parameter">
                <xs:sequence>
                        <xs:element name="data_type_id" type="xs:integer" minOccurs="1" maxOccurs="1"/>
                       <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="ResultList" type="ResultList"/>
        <xs:complexType name="ResultList">
                <xs:sequence>
                        <xs:element name="parameters" type="Parameter" minOccurs="1"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="Device" type="Device"/>
        <xs:complexType name="Device">
                <xs:sequence>
                        <xs:element name="device_kind" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="CommandMessage" type="CommandMessage" minOccurs="1"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
ß
```