January 2016



# Hardware Abstraction Layer for Robotic Technology (HAL4RT)

Version: 1.0 - Beta 1

#### OMG Document Number: dtc/2016-01-01 Standard document URL: http://www.omg.org/spec/HAL4RT/1.0

This OMG document replaces the submission document (robotics/15-12-14, Alpha). It is an OMG Adopted Beta specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to <u>issues@omg.org</u> by January 8, 2016.

You may view the pending issues for this specification from the OMG revision issues web page <u>http://www.omg.org/issues/</u>.

The FTF Recommendation and Report for this specification will be published on December 16, 2016. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

## 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, royaltyfree, 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

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: <u>http://www.omg.org/legal/tm\_list.htm</u>. 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

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.htm.)

## **Table of Contents**

1.	Scope	9
2.	Conformance	9
3. 3.1.	References Normative References	9 9
4.	Terms and Definitions	10
5.	Symbols	10
6.	Additional Information	
6.1.	Acknowledgements	10
-		4.0
7.	Hardware Abstraction Layer for Robotic Technology (HAL4RT)	13
7. 7.1.	General	13 13
7.1. 7.2.	General Format and Conventions	13 13
7.1. 7.2.	General Format and Conventionsand Interface	13 13
<b>7.1.</b> <b>7.2.</b> Class = Enum	General Format and Conventions and Interface	<b>13</b> <b>13</b> 13 14
7.1. 7.2. Class Enum 7.3.	General Format and Conventions and Interface heration Return Codes	<b> 13</b> 13 13 14 14
7.1. 7.2. Class Enum 7.3. 7.4.	General Format and Conventions and Interface eration Return Codes Platform Independent Model (PIM)	<b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b>
7.1. 7.2. Class Enum 7.3. 7.4.	General Format and Conventions and Interface heration Return Codes	<b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b>
<b>7.1.</b> <b>7.2.</b> Class Enum <b>7.3.</b> <b>7.4.</b> 7.5.1.1	General Format and Conventions and Interface reration Return Codes Platform Independent Model (PIM) 1.1 char [ISO/IEC-9899] 1.2 Octet [RTC]	<b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>15</b> <b>16</b> <b>16</b> <b>17</b> <b>17</b> <b>17</b> <b>17</b> <b>18</b> <b>18</b> <b>18</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b>
<b>7.1.</b> <b>7.2.</b> Class Enum <b>7.3.</b> <b>7.4.</b> 7.5.1.1	General Format and Conventions and Interface erration Return Codes Platform Independent Model (PIM) 1.1 char [ISO/IEC-9899]	<b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>14</b> <b>13</b> <b>15</b> <b>16</b> <b>16</b> <b>17</b> <b>17</b> <b>17</b> <b>17</b> <b>18</b> <b>18</b> <b>18</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b> <b>19</b>
<b>7.1.</b> <b>7.2.</b> Class Enum <b>7.3.</b> <b>7.4.</b> <b>7.5.1.</b> <b>7.5.1.</b> <b>7.5.1.</b> <b>7.5.1.</b> <b>7.5.1.</b> <b>7.5.1.</b>	General Format and Conventions and Interface reration Return Codes Platform Independent Model (PIM) 1.1 char [ISO/IEC-9899] 1.2 Octet [RTC]	13 13 13 14 14 14 33 33 33 33

## 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<sup>TM</sup>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<sup>TM</sup>s specifications include: UML® (Unified Modeling Language); CORBA® (Common Object Request Broker Architecture); CWM (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. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec\_catalog.htm

Specifications within the Catalog are organized by the following categories:

#### **OMG Modeling Specifications**

- •UML
- MOF
- •XMI
- •CWM
- Profile specifications

#### **OMG Middleware Specifications**

- •CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM)

#### **Platform Specific Model and Interface Specifications**

- CORBAservices
- CORBAfacilities
- •OMG Domain specifications
- •OMG Embedded Intelligence specifications
- •OMG Security specifications

OMG Headquarters 109 Highland Ave, Needham, MA 02494 USA USA

Tel: +1-781-444-0404 Fax: +1-781-444-0320 Email: pubs@omg.org

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.

## 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 vendors 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 sub clause 7.3. The key words "MUST," "MUST NOT," "REQUIRED," "SHALL,""SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMEND," "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 that, 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 specification, http://www.omg.org/spec/RoIS/1.0/

[SMART]

Smart Transducers specification, <u>http://www.omg.org/spec/SMART/1.0/</u>

[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.

#### **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.

#### Extensive 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. Acknowledgements

The following organization submitted this specification:

 Japan Embedded Systems Technology Association (JASA)
 6-7 Nihonbashi Odenmacho, Chuo-ku, Tokyo, 103-0011 Japan Contact: Kenichi Nakamura (nakamura@upwind-technology.com)

The following organizations contributed to this specification:

• Central Information Center, Co., Ltd.

- CORE CORPORATION
- Dai-ichi Seiko Co., Ltd.
- ECS 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.

The following organizations supported this specification:

- Honda R&D, Co., Ltd.
- Japan Robot Association
- National Institute of Advanced Industrial Science and Technology
- Shibaura Institute of Technology

This page intentionally left blank.

## 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 of 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

#### **Class and Interface**

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

<b>Description</b> :	Description : <description></description>								
Derived Fron	n: <parent< td=""><td>t class&gt;</td><td></td><td></td><td></td></parent<>	t class>							
Attributes									
<attribute< td=""><td>e name&gt;</td><td><attribute type=""></attribute></td><td><obligation></obligation></td><td><occurrence></occurrence></td><td><description></description></td></attribute<>	e name>	<attribute type=""></attribute>	<obligation></obligation>	<occurrence></occurrence>	<description></description>				
Operations	Operations								
<operation r<="" td=""><td colspan="8"><pre><operation name=""> <description></description></operation></pre></td></operation>	<pre><operation name=""> <description></description></operation></pre>								
<direction></direction>	<parameters< td=""><td>eter name&gt;</td><td colspan="2"><pre><parameter type=""></parameter></pre></td><td><description></description></td></parameters<>	eter name>	<pre><parameter type=""></parameter></pre>		<description></description>				

#### Table 7.1 : < Class / Interface Name>

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.

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

unq: The appeared attribute values shall be unique.

#### Enumeration

Enumerations are documented as follows:

#### Table 7.2 : <enumeration name>

<constant name=""></constant>	<description></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.						

#### Table 7 2. Datus mCada ...

#### 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".

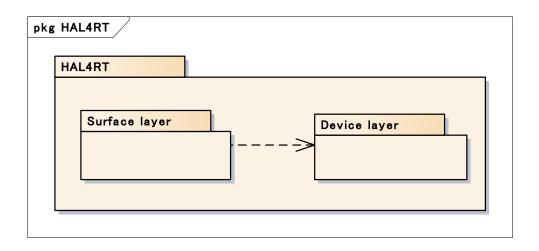


Figure 7.1 - Schematic structure of HAL4RT

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.

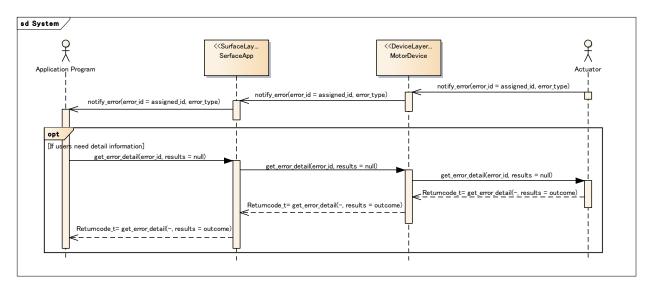


Figure 7.2 – Sequence diagram of System Interface (System Error)

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.

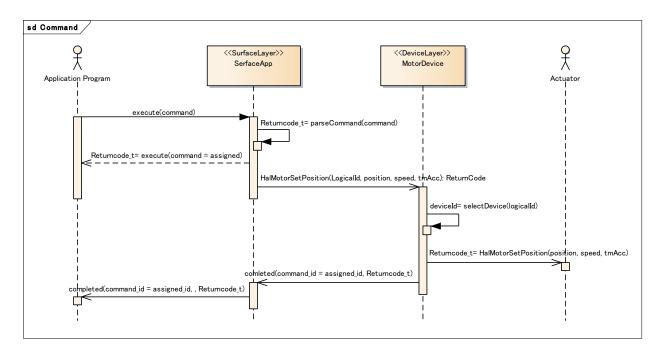


Figure 7.3 – Sequence Diagram of Command Interface (Motor position control)

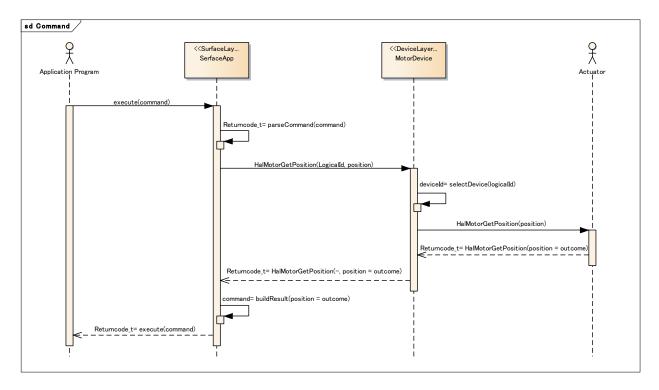


Figure 7.4 – Sequence Diagram of Command Interface (Get current motor position)

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 processing in the device layer. The device layer selects 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.

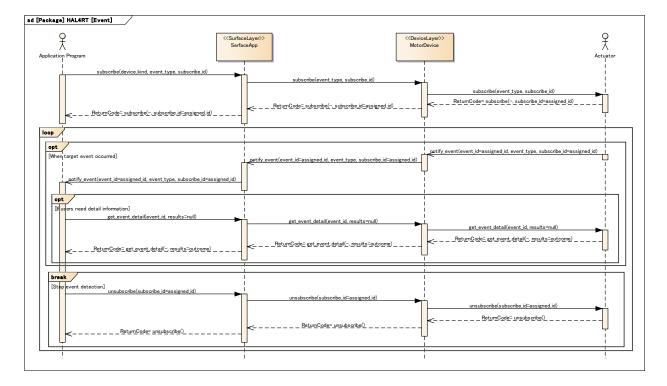


Figure 7.5 – Sequence Diagram of Event Interface

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

When the event occurs, the device notifies the application program and sends event\_id by using notify\_event(). The application program can find out the detail 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 the Application Programming Interface for the application programmer who uses HAL4RT devices. The surface layer parses the command from the application program, gets the device kind, and generates the logical id.

The following diagram is the PIM of the surface layer.

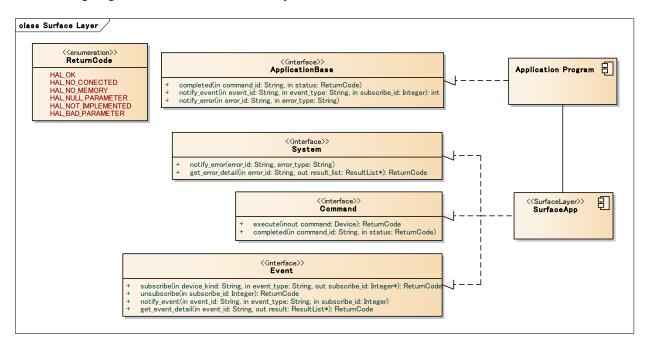


Figure 7.6 - Surface Layer

#### 7.4.3.1 Interface for Surface Layer

#### **Table 7.4: System Interface**

_	<b>Description</b> : 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	e error to the application pr	rogram.			
in	error_id		String	Id information to distinguish the error.			
in	error_type		String	Information to distinguish the kind of the error.			
get_err	or_detail	Get the d	etail of the error.				
in	in error_id		String Id information to distinguish the error that w notified by notify_error.				
out	result_list		ResultList	The detail information of the error.			

#### Table 7.5: Command Interface

Descri	<b>Description</b> : The application program calls the command of the device.					
Derive	ed From: None					
Opera	ntions					
execute Execute the command of the device.						
inout command		Device The kind of the target device, the detail of the command, the result of the command.				
completed Notify the completion of the command.				and.		
in	command_id		String	ID information of the command.		
in	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	Derived From: None						
Ope	erations						
subs	scribe	Regis	ster the event.				
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	ıbscribe	Unre	gister the event.				
in	subscribe_id		Integer	ID information of the registered event.			
get_	event_detail	Get th	ne detail of the event				
in	event_id		String	ID information of the event.			
out	out results		ResultList	The detail information of the event.			
noti	notify_event Notif		y 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 the interface for the application program.

			Table 7.7: Application	Base Interface			
Desc	ription: The applicati	on prog	ram implements this interfac	e to receive the information from the device.			
Deriv	ved From: None						
Oper	ations						
comp	leted	Notify	completion of the process i	n the device.			
in	n command_id		String	The command ID.			
in	status		ReturnCode	The result of command.			
notif	y_event	Notify	lotify 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.			
notify_error Notify			the error in the device.				
in	error_id		String	ID information of the error.			
in	error_type		String	The type of the error.			

#### **Table 7.7: Application Base Interface**

#### 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().

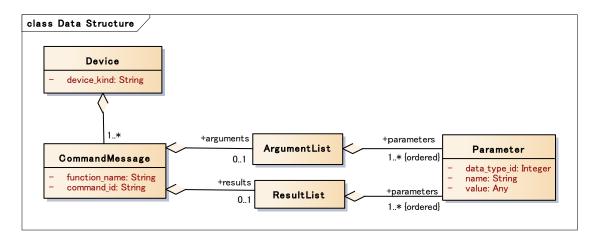


Figure 7.7 – Data Structure of Message

The following tables are the detailed definition of each class.

#### Table 7.8: Device

<b>Description</b> : Data type	<b>Description</b> : Data type to distinguish the device.						
Derived From : None	Derived From : None						
Attributes	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.			

#### Table 7.9: Command Message

<b>Description</b> : Data type to specify the API and to keep the result.								
Derived From : None	Derived From : None							
Attributes								
function_name	String	М	1	The name of API.				
command_id	String	М	1	The ID information to distinguish the CommandMessage. The 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.				

#### Table 7.10: ArgumentList

<b>Description</b> : Data type to keep the parameter.							
Derived From : None							
Attributes	Attributes						
parameters	Parameter	М	Nord	Parameter information.			

#### Table 7.11: ResultList

<b>Description</b> : Data type to keep the result.									
Derived From : None									
Attributes	Attributes								
parameters	Parameter	М	Nord	The information of the result.					
parameters	Parameter	М	Nord	The information of the result.					

#### Table 7.12: Parameter

<b>Description</b> : Data type to send the parameter and to keep the result.								
Derived From : None								
Attributes	Attributes							
data_type_id	Integer	М	1	The ID of data type.				
name	String	М	1	Parameter name.				
value	Any	М	1	Parameter value.				

r

#### 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.

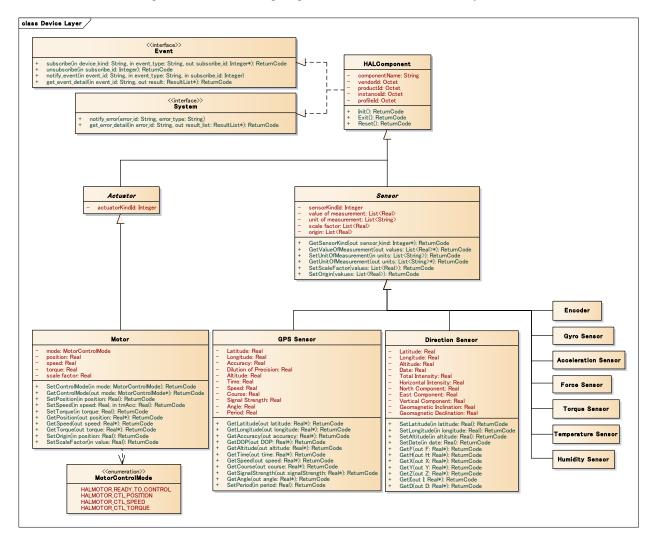


Figure 7.8 – Device Layer

Products conforming to HAL4RT, for each affected device must implement the 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 the surface layer is omitted, it is not necessary to implement these interfaces.

#### 7.4.5.1 HAL Component

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

<b>Derived From</b> : N	Derived From: None								
Attributes	Attributes								
componentName	String		М	1	Component name.				
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 (	Compon	ent				
Exit Finish HALComponent.									
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 0xFFFFFFFF. The Vendor names and Vendor IDs are maintained by the OMG. The Vendor name is a 2 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 0xFFFFFFFF. The Instance IDs are defined by the application developers.

#### profileId

A 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

0x0000001, Motor

0x00000002, Encoder 0x00000003, GyroSensor 0x00000004, TorqueSensor

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

#### Device Characteristics Profile

A profile that will 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".

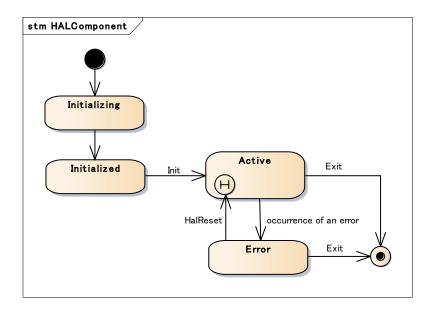


Figure 7.9 – State Machine of HAL Component

The detail of each state is below.

- Initializing: The status to initialize the device. The application program cannot 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								
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.

Derived From: Actuator										
Attributes	Attributes									
mode	MotorControlMode	М	1	The control mode of the motor.						
position	Real	М	1	The current position/angle.						
speed	Real	М	1	The current velocity/angular velocity.						
torque	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 pi. Linear system : The value per meter. The initial value is1.0.						

Oper	ations						
SetControlMode			Set control mode of the motor.				
in	mode	Moto	rControlMode	М	Control mode.		
GetC	ontrolMode		Get control mod	e of the	e motor.		
out	mode	Moto	rControlMode	М	The current control mode.		
SetPo	osition		Move to the targ to position contr	-	tion/angle. If the control mode is not position control mode, change e.		
in	position	Real		М	The target position. Unit: scale factor.		
				Increase/Decrease velocity/angular velocity in the specified acceleration/deceleration time. If the control mode is not speed control mode, change to speed control mode.			
in	speed	Real		М	The target speed. Unit:scale factor/second.		
in	tmAcc	Real		М	Acceleration/Deceleration time to the target speed. Unit:second.		
SetTo	SetTorque		Output the specified torque. If the control mode is not torque control mode, change to torque control mode.				
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	peed	•	Get the current v	Get the current velocity/angular velocity.			
out	speed	Real		М	The current speed. Unit:scale factor/second		
GetTorque Get the c		Get the current to	orque.				
out	torque	Real		М	The current torque. Unit:[Nm] or [N]		
SetO	rigin		Set the current p	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.

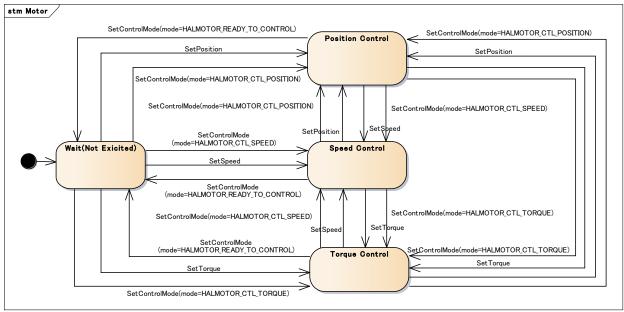


Figure 7.10 – State Machine of Motor

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.

Deri	Derived From: HAL Component								
Attr	Attributes								
sens	orKindId	Integ	er	М	1	ID information to distinguish the kind of the sensor.			
valu	value of measurement Real M			М	Nord	Value of measurement of the sensor.			
unit	unit of measurement String M			М	Nord	Unit of measurement.			
scale factor Real M N			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	Operations								
GetSensorKind Get the kind of the				the kind	of the s	ensor.			
out	sensor_kind	Integer			М	The kind of the sensor.			

Get V	GetValueOfMeasurement Get the value of		of meas	urement of the sensor.			
out	values	List <re< td=""><td>eal&gt;</td><td>М</td><td>Value of measurement of the sensors.</td></re<>	eal>	М	Value of measurement of the sensors.		
SetU	InitOfMeasureme	nt	Set the unit of	sensor	measurement.		
in	units	List <st< td=""><td>ring&gt;</td><td>М</td><td>Unit of sensor measurement.</td></st<>	ring>	М	Unit of sensor measurement.		
GetUnitOfMeasurement Get the unit of			Get the unit of	sensor	measurement.		
out	units	List <st< td=""><td colspan="2">List<string></string></td><td>Unit of sensor measurement.</td></st<>	List <string></string>		Unit of sensor measurement.		
SetS	caleFactor		Set the scale fa	ctor of	ctor of the value of measurement of the sensor.		
in	values	List <re< td=""><td>eal&gt;</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.		
GetS	ScaleFactor		Get the scale fa	actor of	f the value of measurement of the sensor.		
out	values	List <re< td=""><td colspan="2">List<real></real></td><td>Scale factor of the value of measurement of the sensor.</td></re<>	List <real></real>		Scale factor of the value of measurement of the sensor.		
SetOrigin Set the origin of			Set the origin of	of the s	ensor.		
in	values	List <re< td=""><td>eal&gt;</td><td>М</td><td>Origin of the sensor.</td></re<>	eal>	М	Origin of the sensor.		

#### 7.4.5.6 Encoder

Encoder is a sensor to measure the position / angle.

Derived From: Sensor	
Attributes: None	
<b>Operations</b> : None	

#### 7.4.5.7 Gyro Sensor

Gyro Sensor is a sensor to measure angular velocity.

Derived From: Sensor	
Attributes:None	
<b>Operations</b> : 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 Force Sensor

Force Sensor is a sensor to measure force.

Derived From: Sensor	
Attributes: None	
<b>Operations</b> : None	

#### 7.4.5.10 Torque Sensor

Torque Sensor is a sensor to measure torque.
Derived From: Sensor
Attributes:None
<b>Operations</b> : None

#### 7.4.5.11 Temperature Sensor

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	

#### 7.4.5.13 GPS Sensor

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

Derived From: Sensor							
Attributes							
latitude	Real	М	1	Latitude			
longitude	Real	М	1	Longitude			
accuracy	Real	М	1	Accuracy			
dilution of precision	Real	М	1	Dilution of precision (DOP)			

altitude Real		1	М	1		Altitude			
time	time Real		М	1		Time			
speed Real		М	1		Speed				
cours	se	Rea	1	М	1		Course		
signa	l strength	Rea	1	М	1		Signal strength		
angle	)	Rea	1	Μ	1		Angle		
perio	d	Rea	1	Μ	1		Period		
Oper	rations								
GetL	atitude		Get the lat	titude,	in de	egree	28.		
out	latitude	Re	eal			М	Latitude		
GetL	ongitude		Get the los	ngitud	e, in	degr	ees.		
out longitude Real					М	Longitude			
GetAccuracy Get the estimated a			d acc	curac	y of this location, in meters.				
out	accuracy	Re	eal			Μ	Accuracy		
GetD	OP		Get the va	lue of	DOF	P.			
out	DOP	Re	eal			М	Dilution of precision (DOP)		
GetA	ltitude		Get the alt	titude	if ava	ailab	le, in meters above the WGS 84 reference ellipsoid.		
out	altitude	Re	eal			М	Altitude		
GetT	ime		Return the	e UTC	time	e of th	his fix, in milliseconds since January 1, 1970.		
out time Real				Μ	Time				
GetS	peed		Get the sp	eed if	it is a	avail	able, in meters/second over ground.		
out	speed	Re	eal			Μ	Speed. Unit : [m/s]		
GetCourse Get the course if it is				ourse if	f it is	is available, in 000.0–359.9 degrees.			
out course Real				М	Course. Unit : [degree]				
GetSignalStrength Get signal strength			gth.						
out signal strength Real			Μ	Signal strength					
GetAngle Get angle.									
out angle Real				Μ	Angle				
SetPeriod Set period.									
in	period	Re	eal			М	Period		

#### 7.4.5.14 Direction Sensor

Direction Sensor is a sensor to measure direction.

Deri	ved From: Senso	or					
Attr	ibutes						
latitude			Real	М	1		Latitude90.00 to +90.00 degrees
longitude Real			М	1		Longitude180.00 to +180.00 degrees	
altitı	ıde		Real	М	1		Altitude. referenced to the WGS 84 ellipsoid OR the Mean Sea Level (MSL)
date			Real	Μ	1		Date. 2015.0 to 2020.0
total	intensity		Real	Μ	1		F - Total Intensity of the geomagnetic field
horiz	zontal intensity		Real	М	1		H - Horizontal Intensity of the geomagnetic field
nortl	n component		Real	М	1		X - North Component of the geomagnetic field
east	component		Real	М	1		Y - East Component of the geomagnetic field
verti	cal component		Real	М	1		Z - Vertical Component of the geomagnetic field
geor	nagnetic inclination	on	Real	М	1		I (DIP) - Geomagnetic Inclination
geomagnetic declination			Real	М	M 1		D (DEC) - Geomagnetic Declination (Magnetic Variation)
Ope	rations						
SetLatitude Set latitude.							-
in	latitude	Re	eal			Μ	Latitude
SetLongitude Set longitude.				ude.			-
in	longitude	Re	eal			Μ	Longitude
SetA	ltitude		Set altitud	le.			
in	Altitude	Re	eal			Μ	Altitude
SetD	Date		Set date.				
in	date	Re	eal			Μ	Date
GetF	7		Get F.				
out	F	Re	eal			М	F - Total Intensity of the geomagnetic field
GetH	I		Get H.		Ī		·
out	Н	Re	eal			Μ	H - Horizontal Intensity of the geomagnetic field
Get	K		Get X.				
out	Х	Re	eal			Μ	X - North Component of the geomagnetic field
Get Y. Get Y.							· · · · · · · · · · · · · · · · · · ·
out Y Real				Μ	Y - East Component of the geomagnetic field		
GetZ	7	•	Get Z.				·
out	Z	Re	Real			Μ	Z - Vertical Component of the geomagnetic field
GetI		•	Get I.				

out	Ι	Re	eal	М	I (DIP) - Geomagnetic Inclination
GetD			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; typedef struct Event { 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 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

#### 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>
                                                                                    parameters" type="Parameter" minOco
                       <xs:element name="
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>
```

This page intentionally left blank.