BS EN 61800-7-202:2016

BSI Standards Publication

Adjustable speed electrical power drive systems

Part 7-202: Generic interface and use of profiles for power drive systems — Profile type 2 specification

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 61800-7-202:2016. It is identical to IEC 61800-7-202:2015. It supersedes BS EN 61800-7-202:2008, which will be withdrawn on 12 October 2018.

The UK participation in its preparation was entrusted to Technical Committee PEL/22, Power electronics.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2016

ISBN 978 0 580 82129 5
ICS 29.200; 35.100.05

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 March 2016.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 61800-7-202

February 2016

ICS 29.200; 35.100.05

Supersedes EN 61800-7-202:2008

English Version

Adjustable speed electrical power drive systems - Part 7-202: Generic interface and use of profiles for power drive systems - Profile type 2 specification (IEC 61800-7-202:2015)

Entraînements électriques de puissance à vitesse variable - Partie 7-202: Interface générique et utilisation de profils pour les entraînements électriques de puissance - Spécification de profil de type 2 (IEC 61800-7-202:2015)

Elektrische Leistungsantriebssysteme mit einstellbarer Drehzahl - Teil 7-202: Generisches Interface und Nutzung von Profilen für Leistungsantriebssysteme (PDS) - Spezifikation von Profil-Typ 2 (IEC 61800-7-202:2015)

This European Standard was approved by CENELEC on 2015-12-25. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No. EN 61800-7-202:2016 E

European foreword

The text of document 22G/308/FDIS, future edition 2 of IEC 61800-7-202, prepared by SC 22G “Adjustable speed electric drive systems incorporating semiconductor power converters” of IEC/TC 22 “Power electronic systems and equipment" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61800-7-202:2016.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2016-09-25
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2018-12-25

This document supersedes EN 61800-7-202:2008.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 61800-7-202:2015 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61131-3	NOTE	Harmonized as EN 61131-3.
IEC 61158 Series	NOTE	Harmonized as EN 61158 Series.
IEC 61158-2:2014	NOTE	Harmonized as EN 61158-2:2014 (not modified).
IEC 61158-3-2:2014	NOTE	Harmonized as EN 61158-3-2:2014 (not modified).
IEC 61499-1:2005	NOTE	Harmonized as EN 61499-1:2005 [1)] (not modified).
IEC 61784-1:2014	NOTE	Harmonized as EN 61784-1:2014 (not modified).
IEC 61784-2:2014	NOTE	Harmonized as EN 61784-2:2014 (not modified).
IEC 61800 Series	NOTE	Harmonized as EN 61800 Series.
IEC 61800-7 Series	NOTE	Harmonized as EN 61800-7 Series.
IEC 61800-7-201	NOTE	Harmonized as EN 61800-7-201.
IEC 61800-7-203	NOTE	Harmonized as EN 61800-7-203.
IEC 61800-7-204	NOTE	Harmonized as EN 61800-7-204.
IEC 61800-7-301	NOTE	Harmonized as EN 61800-7-301.
IEC 61800-7-302	NOTE	Harmonized as EN 61800-7-302.
IEC 61800-7-303	NOTE	Harmonized as EN 61800-7-303.
IEC 61800-7-304	NOTE	Harmonized as EN 61800-7-304.
IEC 62026-3	NOTE	Harmonized as EN 62026-3.

1) Superseded by EN 61499-1:2013 (IEC 61499-1:2012).

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu

Publication	Year	Title	EN/HD	Year
IEC 60204-1	-	Safety of machinery - Electrical equipment of machines - Part 1: General requirements	EN 60204-1	-
IEC 61158-4-2	2014	Industrial communication networks - Fieldbus specifications - Part 4-2: Data-link layer protocol specification - Type 2 elements	EN 61158-4-2	2014
IEC 61158-5-2	2014	Industrial communication networks - Fieldbus specifications - Part 5-2: Application layer service definition - Type 2 elements	EN 61158-5-2	2014
IEC 61158-6-2	2014	Industrial communication networks - Fieldbus specifications - Part 6-2: Application layer protocol specification - Type 2 elements	EN 61158-6-2	2014
IEC 61588	2009	Precision clock synchronization protocol for networked measurement and control systems	-	-
IEC 61800-7-1	2015	Adjustable speed electrical power drive systems - Part 7-1: Generic interface and use of profiles for power drive systems - Interface definition	EN 61800-7-1	2016
IEEE Std 112	2004	IEEE Standard Test Procedure for Polyphase Induction Motors and Generators	-	-

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

ADJUSTABLE SPEED ELECTRICAL POWER DRIVE SYSTEMS –

Part 7-202: Generic interface and use of profiles for power drive systems – Profile type 2 specification

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

International Standard IEC 61800-7-202 has been prepared by subcommittee SC 22G: Adjustable speed electric drive systems incorporating semiconductor power converters, of IEC technical committee TC 22: Power electronic systems and equipment.

This second edition cancels and replaces the first edition published in 2007. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) update of patent information;
b) new revision of the Drive Profile and Drive Axis specifications, with multiple clarifications and enhancements.

The text of this standard is based on the following documents:

FDIS	Report on voting
22G/308/FDIS	22G/323/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts of the IEC 61800 series, under the general title *Adjustable speed electrical power drive systems*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

0.1 General

The IEC 61800 series is intended to provide a common set of specifications for adjustable speed electrical power drive systems.

IEC 61800-7 specifies profiles for Power Drive Systems (PDS) and their mapping to existing communication systems by use of a generic interface model.

IEC 61800-7 describes a generic interface between control systems and power drive systems. This interface can be embedded in the control system. The control system itself can also be located in the drive (sometimes known as "smart drive" or "intelligent drive").

A variety of physical interfaces is available (analogue and digital inputs and outputs, serial and parallel interfaces, fieldbuses and networks). Profiles based on specific physical interfaces are already defined for some application areas (e.g. motion control) and some device classes (e.g. standard drives, positioner). The implementations of the associated drivers and application programmers interfaces are proprietary and vary widely.

IEC 61800-7 defines a set of common drive control functions, parameters, and state machines or description of sequences of operation to be mapped to the drive profiles.

IEC 61800-7 provides a way to access functions and data of a drive that is independent of the used drive profile and communication interface. The objective is a common drive model with generic functions and objects suitable to be mapped on different communication interfaces. This makes it possible to provide common implementations of motion control (or velocity control or drive control applications) in controllers without any specific knowledge of the drive implementation.

There are several reasons to define a generic interface:

For a drive device manufacturer

– less effort to support system integrators;
– less effort to describe drive functions because of common terminology;
– the selection of drives does not depend on availability of specific support.

For a control device manufacturer

– no influence of bus technology;
– easy device integration;
– independent of a drive supplier.

For a system integrator

– less integration effort for devices;
– only one understandable way of modeling;
– independent of bus technology.

Much effort is needed to design a motion control application with several different drives and a specific control system. The tasks to implement the system software and to understand the functional description of the individual components may exhaust the project resources. In some cases, the drives do not share the same physical interface. Some control devices just support a single interface which will not be supported by a specific drive. On the other hand, the functions and data structures are often specified with incompatibilities. This requires the system integrator to write special interfaces for the application software and this should not be his responsibility.

Some applications need device exchangeability or integration of new devices in an existing configuration. They are faced with different incompatible solutions. The efforts to adapt a solution to a drive profile and to manufacturer specific extensions may be unacceptable. This will reduce the degree of freedom to select a device best suited for this application to the selection of the unit which will be available for a specific physical interface and supported by the controller.

IEC 61800-7-1 is divided into a generic part and several annexes as shown in Figure 1. The drive profiles types for CiA® 402[1], CIP Motion™[2], PROFIdrive[3] and SERCOS®[4] are mapped to the generic interface in the corresponding annex. The annexes have been submitted by open international network or fieldbus organizations which are responsible for the content of the related annex and use of the related trademarks.

This part of IEC 61800-7 specifies the profile type 2 (CIP Motion™).

The profile types 1, 3 and 4 are specified in IEC 61800-7-201, IEC 61800-7-203 and IEC 61800-7-204.

---

1 CiA® 402 is a registered trade mark of CAN in Automation e.V (CiA). This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the registered trade mark CiA® 402. Use of the registered trade mark CiA® 402 requires permission of CAN in Automation e.V (CiA).

2 CIP Motion™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark CIP Motion™. Use of the trade mark CIP Motion™ requires permission of ODVA, Inc.

3 PROFIdrive is a trade name of PROFIBUS & PROFINET International. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name PROFIdrive. Use of the trade name PROFIdrive requires permission of PROFIBUS & PROFINET International.

4 SERCOS® is a registered trade mark of SERCOS International e.V. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the registered trade mark SERCOS®. Use of the registered trade mark SERCOS® requires permission of the trade mark holder.

IEC 61800-7-301, IEC 61800-7-302, IEC 61800-7-303 and IEC 61800-7-304 specify how the profile types 1, 2, 3 and 4 are mapped to different network technologies (such as CANopen®[5], CC-Link IE® Field Network[6], EPA™[7], EtherCAT®[8], Ethernet Powerlink™[9], DeviceNet™[10], ControlNet™[11], EtherNet/IP™[12], PROFIBUS[13], PROFINET[14] and SERCOS®).

---

[5] CANopen® is a registered trade mark of CAN in Automation e.V. (CiA). This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the registered trade mark CANopen®. Use of the registered trade mark CANopen® requires permission of CAN in Automation e.V. (CiA). CANopen® is an acronym for *C*ontroller *A*rea *N*etwork *open* and is used to refer to EN 50325-4.

[6] CC-Link IE® Field Network is a registered trade mark of Mitsubishi Electric Corporation. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the registered trade mark CC-Link IE® Field Network. Use of the registered trade mark CC-Link IE® Field Network requires permission of Mitsubishi Electric Corporation.

[7] EPA™ is a trade mark of SUPCON Group Co. Ltd. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark EPA™. Use of the trade mark EPA™ requires permission of the trade mark holder.

[8] EtherCAT® is a registered trade mark of Beckhoff, Verl. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the registered trade mark EtherCAT®. Use of the registered trade mark EtherCAT® requires permission of the trade mark holder.

[9] Ethernet Powerlink™ is a trade mark of Bernecker & Rainer Industrieelektronik Ges.m.b.H., control of trade mark use is given to the non profit organization EPSG. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark Ethernet Powerlink™. Use of the trade mark Ethernet Powerlink™ requires permission of the trade mark holder.

[10] DeviceNet™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark DeviceNet™. Use of the trade mark DeviceNet™ requires permission of ODVA, Inc.

[11] ControlNet™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark ControlNet™. Use of the trade mark ControlNet™ requires permission of ODVA, Inc.

[12] EtherNet/IP™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark EtherNet/IP™. Use of the trade mark EtherNet/IP™ requires permission of ODVA, Inc.

[13] PROFIBUS is a trade name of PROFIBUS & PROFINET International. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name PROFIBUS. Use of the trade name PROFIBUS requires permission of PROFIBUS & PROFINET International.

[14] PROFINET is a trade name of PROFIBUS & PROFINET International. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance to this profile does not require use of the trade name PROFINET. Use of the trade name PROFINET requires permission of PROFIBUS & PROFINET International.

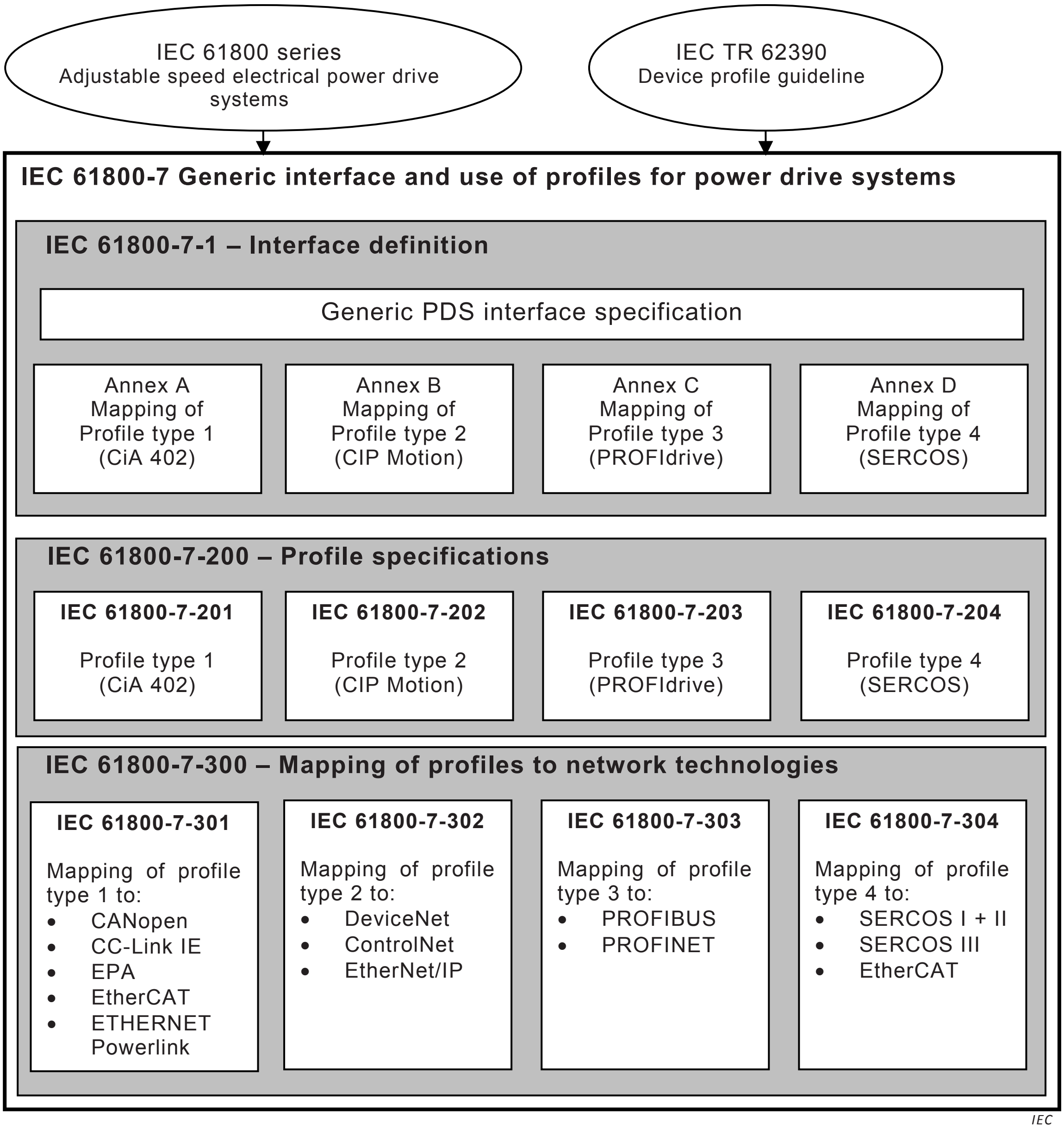

Figure 1 – Structure of IEC 61800-7

0.2 Patent declaration

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning the following. This patent is held by its inventors under license to ODVA, Inc:

Publication / Application serial number	Holder	Title
US 7,983,769 EP 1659465	[ODVA]	Time stamped motion control network protocol that enables balanced single cycle timing and utilization of dynamic data structures

IEC takes no position concerning the evidence, validity and scope of this patent right.

ODVA and the holder of this patent right have assured the IEC that ODVA is willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of ODVA and the holder of this patent right is registered with IEC. Information may be obtained from:

[ODVA]	ODVA, Inc. 2370 East Stadium Boulevard #1000 Ann Arbor, Michigan 48104 USA Attention: Office of the Executive Director email: odva@odva.org

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the data bases for the most up to date information concerning patents.

ADJUSTABLE SPEED ELECTRICAL POWER DRIVE SYSTEMS –

Part 7-202: Generic interface and use of profiles for power drive systems – Profile type 2 specification

1 Scope

This part of IEC 61800 specifies profile type 2 (CIP Motion™) for Power Drive Systems (PDS). Profile type 2 can be mapped onto different communication network technologies.

The functions specified in this part of IEC 61800-7 are not intended to ensure functional safety. This requires additional measures according to the relevant standards, agreements and laws.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60204-1, *Safety of machinery – Electrical equipment of machines – Part 1: General requirements*

IEC 61158-4-2:2014, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-5-2:2014, *Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements*

IEC 61158-6-2:2014, *Industrial communication networks – Fieldbus specifications – Part 6-2: Application layer protocol specification – Type 2 elements*

IEC 61588:2009, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61800-7-1:2015, *Adjustable speed electrical power drive systems – Part 7-1: Generic interface and use of profiles for power drive systems – Interface definition*

IEEE Std 112-2004, *IEEE Standard Test Procedure for Polyphase Induction Motors and Generators*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

3.1.1
algorithm
completely determined finite sequence of operations by which the values of the output data can be calculated from the values of the input data

[SOURCE: IEC 61800-7-1:2015, 3.2.1]

3.1.2
application
software functional element specific to the solution of a problem in industrial-process measurement and control

Note 1 to entry: An application may be distributed among resources, and may communicate with other applications.

[SOURCE: IEC 61800-7-1:2015, 3.2.2]

3.1.3
application mode
type of application that can be requested from a PDS

Note 1 to entry: The different application modes reflect the control loop for torque control, velocity control, position control or other applications such as homing.

[SOURCE: IEC 61800-7-1:2015, 3.3.1.2]

3.1.4
attribute
property or characteristic of an entity

[SOURCE: IEC 61800-7-1:2015, 3.2.3]

3.1.5
axis
logical element inside an automation system (e.g. a motion control system) that represents some form of movement

Note 1 to entry: Axes can be rotary or linear, physical or virtual, controlled or simply observed.

[SOURCE: IEC 61800-7-1:2015, 3.2.4]

3.1.6
bus regulator
any method used to limit the rise in DC bus voltage level that occurs when decelerating a motor

3.1.7
CIP Motion™ [15]
extensions to the CIP services and protocol to support motion control over CIP networks

[SOURCE: IEC 61800-7-1:2015, 3.3.3.1]

---

[15] CIP Motion™ and CIP Sync™ are trade marks of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade marks CIP Motion™ or CIP Sync™. Use of the trade marks CIP Motion™ or CIP Sync™ requires permission of ODVA, Inc.

3.1.8
CIP Motion™ controller
CIP compliant controller containing a Motion Control Axis Object that can interface to a CIP Motion device via a CIP Motion I/O Connection

Note 1 to entry: A description of the Motion Control Axis Object is beyond the scope of IEC 61800-7.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.2]

3.1.9
CIP Motion™ device
CIP compliant device containing one or more Motion Device Axis Object instances that can communicate to a CIP Motion controller via a CIP Motion I/O Connection

EXAMPLE: A CIP Motion drive is a particular case of a CIP Motion device.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.3]

3.1.10
CIP Motion™ drive profile
collection of objects used to implement a CIP Motion drive device that includes the Motion Device Axis Object, as well as standard support objects like the Identity Object and the Time Sync Object

Note 1 to entry: The Device Type assigned to the CIP Motion drive profile is 25_{hex}.

3.1.11
CIP Motion™ I/O Connection
CIP Motion™ Connection
periodic bi-directional, class 1, CIP connection between a controller and a drive that is defined as part of the CIP Motion specification

[SOURCE: IEC 61800-7-1:2015, 3.3.3.4]

3.1.12
CIP Sync™[15]
extensions to the CIP services and protocol to encapsulate IEC 61588:2009 time synchronization functionality over a CIP Network

Note 1 to entry: See Time Sync Object in IEC 61158-5-2 and IEC 61158-6-2.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.5]

3.1.13
class
description of a set of objects that share the same attributes, operations, methods, relationships, and semantics

[SOURCE: IEC 61800-7-1:2015, 3.2.5]

3.1.14
closed loop
methods of control where there is a feedback signal of some kind that is used to drive the actual dynamics of the motor to match the commanded dynamics by servo action

Note 1 to entry: In most cases, there is a literal feedback device to provide this signal, but in some cases, the signal is derived from the motor excitation (i.e. sensorless operation).

3.1.15
commands
set of commands from the application control program to the PDS to control the behavior of the PDS or functional elements of the PDS

Note 1 to entry: The behavior is reflected by states or operating modes.

Note 2 to entry: The different commands may be represented by one bit each.

[SOURCE: IEC 61800-7-1:2015, 3.3.1.3]

3.1.16
control
purposeful action on or in a process to meet specified objectives

[SOURCE: IEC 61800-7-1:2015, 3.2.6]

3.1.17
control device
physical unit that contains – in a module/subassembly or device – an application program to control the PDS

[SOURCE: IEC 61800-7-1:2015, 3.2.7]

3.1.18
converter
device that generally converts AC input to DC output

Note 1 to entry: A converter is also commonly called the Drive Power Supply. In the context of a drive system, the converter is responsible for converting AC main input into DC bus power.

3.1.19
Cyclic Data Block
high priority real-time data block that is transferred by a CIP Motion Connection on a periodic basis

3.1.20
data type
set of values together with a set of permitted operations

[SOURCE: IEC 61800-7-1:2015, 3.2.8]

3.1.21
device
field device
<function blocks> networked independent physical entity of an industrial automation system capable of performing specified functions in a particular context and delimited by its interfaces

[SOURCE: IEC 61800-7-1:2015, 3.2.9]

3.1.22
device
field device
<system integration> entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system

[SOURCE: IEC 61800-7-1:2015, 3.2.10]

3.1.23
device profile
representation of a device in terms of its parameters, parameter assemblies and behaviour according to a device model that describes the data and behaviour of the device as viewed through a network, independent from any network technology

[SOURCE: IEC 61800-7-1:2015, 3.2.11]

3.1.24
drive
device designed to control the dynamics of a motor

3.1.25
Event Data Block
medium priority real-time data block that is transferred by a CIP Motion Connection only after a specified event occurs

Note 1 to entry: Registration and marker input transitions are typical drive events.

3.1.26
functional element
entity of software or software combined with hardware, capable of accomplishing a specified function of a device

Note 1 to entry: A functional element has an interface, associations to other functional elements and functions.

Note 2 to entry: A functional element can be made out of function block(s), object(s) or parameter list(s).

[SOURCE: IEC 61800-7-1:2015, 3.2.13]

3.1.27
input data
data transferred from an external source into a device, resource or functional element

[SOURCE: IEC 61800-7-1:2015, 3.2.14]

3.1.28
interface
shared boundary between two entities defined by functional characteristics, signal characteristics, or other characteristics as appropriate

[SOURCE: IEC 61800-7-1:2015, 3.2.15]

3.1.29
inverter
device that generally converts DC input to AC output

Note 1 to entry: An inverter is also commonly called the Drive Amplifier. In the context of a drive system, the inverter is responsible for controlling the application of DC bus power to an AC motor.

3.1.30
model
mathematical or physical representation of a system or a process, based with sufficient precision upon known laws, identification or specified suppositions

[SOURCE: IEC 61800-7-1:2015, 3.2.17]

3.1.31
motion
any aspect of the dynamics of an axis

Note 1 to entry: In the context of this part of IEC 61800-7, it is not limited to servo drives but encompasses all forms of drive based motor control.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.8]

3.1.32
Motion Control Axis Object
object that defines the attributes, services, and behavior of a controller based axis according to the CIP Motion specification

3.1.33
Motion Device Axis Object
object that defines the attributes, services, and behavior of a motion device based axis according to the CIP Motion specification

Note 1 to entry: This object includes Communications, Device control, and Basic drive FE elements as defined in IEC 61800-7.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.9]

3.1.34
open loop
methods of control where there is no application of feedback to force the actual motor dynamics to match the commanded dynamics

EXAMPLE Examples of open loop control are stepper drives and variable frequency drives.

3.1.35
operating mode
characterization of the way and the extent to which the human operator intervenes in the control equipment

[SOURCE: IEC 61800-7-1:2015, 3.2.18]

3.1.36
output data
data originating in a device, resource or functional element and transferred from them to external systems

[SOURCE: IEC 61800-7-1:2015, 3.2.19]

3.1.37
parameter
data element that represents device information that can be read from or written to a device, for example through the network or a local HMI

Note 1 to entry: A parameter is typically characterized by a parameter name, data type and access direction.

[SOURCE: IEC 61800-7-1:2015, 3.2.20]

3.1.38
profile
representation of a PDS interface in terms of its parameters, parameter assemblies and behavior according to a communication profile and a device profile

[SOURCE: IEC 61800-7-1:2015, 3.2.21, modified – Note 1 to entry is deleted]

3.1.39
read
get
operation that involves the retrieving of an attribute value from the perspective of the controller side of the interface

3.1.40
registration
high speed record of motion axis position triggered by an event

3.1.41
Service Data Block
lower priority real-time data block associated with a service message from the controller that is transferred by a CIP Motion Connection on a periodic basis

Note 1 to entry: Service data includes service request messages to access Motion Device Axis Object attributes or perform various drive diagnostics.

3.1.42
set-point
value or variable used as output data of the application control program to control the PDS

Note 1 to entry: The value sent to the drive is used to directly control some aspect of the motor dynamics, which includes (but is not limited to) position, velocity, acceleration, and torque.

[SOURCE: IEC 61800-7-1:2015, 3.3.1.5, modified – Note 1 to entry is added]

3.1.43
shunt regulator
specific bus regulator method that switches the DC bus across a power dissipating resistor to dissipate the regenerative power of a decelerating motor

3.1.44
status
set of information from the PDS to the application control program reflecting the state or mode of the PDS or a functional element of the PDS

Note 1 to entry: The different status information may be coded with one bit each.

[SOURCE: IEC 61800-7-1:2015, 3.3.1.6]

3.1.45
synchronized
condition where the local clock value on the drive is locked onto the master clock of the distributed System Time

Note 1 to entry: When synchronized, the drive and controller devices may utilise time stamps associated with CIP Motion Connection data.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.11]

3.1.46
System Time
absolute time value as defined in the CIP Sync specification in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock

Note 1 to entry: In the context of CIP Motion, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1970-01-01.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.12]

3.1.47
thermal model
any algorithm that attempts to model the thermal behavior of the associated device during operation

3.1.48
time offset
System Time offset value associated with the CIP Motion Connection data that is associated with source device

Note 1 to entry: The System Time offset is a 64-bit offset value that is added to a device's local clock to generate System Time for that device

3.1.49
time stamp
System Time stamp value associated with the CIP Motion Connection data that conveys the absolute time when the associated data was captured, or that can also be used to determine when the associated data shall be applied

Note 1 to entry: CIP time stamps are always in the context of a distributed time system where all nodes on the CIP control network have clocks that are synchronized with a master clock source using CIP Sync.

[SOURCE: IEC 61800-7-1:2015, 3.3.3.13]

3.1.50
type
hardware or software element which specifies the common attributes shared by all instances of the type

[SOURCE: IEC 61800-7-1:2015, 3.2.23]

3.1.51
variable
software entity that may take different values, one at a time

Note 1 to entry: The values of a variable as well as of a parameter are usually restricted to a certain data type.

[SOURCE: IEC 61800-7-1:2015, 3.2.25]

3.1.52
variable frequency drive
VFD
class of drive products that seek to control the speed of a motor, typically an induction motor, through a proportional relationship between drive output voltage and commanded output frequency

Note 1 to entry: Variable frequency drives are therefore sometimes referred to as a Volts/Hertz drives.

Note 2 to entry: The English abbreviation VFD is also used in French.

3.1.53
vector drive
class of drive products that seek to control the dynamics of a motor via closed loop control which includes, but is not limited to, closed loop control of both torque and flux vector components of motor stator current relative to the rotor flux vector

3.1.54
write
set
operation that involves the setting of an attribute to a specified value from the perspective of the controller side of the interface

3.2 Abbreviated terms

Ack	acknowledge
Act	actual
Blk	block
CIP™ [16]	Common Industrial Protocol (see IEC 61158 Type 2, IEC 61784-1 and IEC 61784-2 Communication Profile Family 2)
Cmd	command
Cyc	cyclic
HMI	human machine interface
ID	identifier
impl.	implementation
I/O	Input/Output
Opt.	optional
PDS	power drive system
Req.	required
RMS	root mean square
r/min	revolutions per minute
Sync'd	synchronized
TCP	Transmission Control Protocol (see IETF RFC 793)
UDP	User Datagram Protocol (see IETF RFC 768)
VFD	variable frequency drive

4 Overview

4.1 General

CIP Motion devices control, monitor or support the motion of one or more moving component of a machine. Machine motion is typically generated by rotary or linear motion actuators, i.e.: motors, and monitored by feedback devices. Each motor is typically driven by a power structure and a motion control algorithm that comprise a CIP Motion Drive Device Type.

Drive functionality supported by the CIP Motion drive device profile can be applied to a variety of motor technologies, and can range from very simple "open loop" variable frequency drives to sophisticated "closed loop" vector controlled servo drives. In either case, motion is controlled via a command reference that can be configured for position control, velocity control, acceleration control, or current/torque control. The CIP Motion drive device profile also supports position, velocity and acceleration monitoring through multiple feedback, as does the CIP Motion encoder device profile.

[16] CIP™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark CIP™. Use of the trade mark CIP™ requires permission of ODVA, Inc.

All the attributes, services, and state behavior of a CIP Motion device are encapsulated in one or more Motion Device Axis Object instances. In addition to motion control and feedback monitoring functionality, the Motion Device Axis Object includes support for event monitoring, such position capture on a Registration event, and DC Bus management associated with a Power Converter. A CIP Motion device can manifest any combination of these functions to create various classes of CIP Motion compliant devices ranging from full featured servo drives, to CIP Motion Encoders, to standalone Power Converters, differentiated by the CIP Motion Device Type.

This CIP Motion drive profile specification and the associated Motion Device Axis Object specification define the interface, the specific attributes, and command behaviors required to support motion control when connected to a CIP Motion compliant controller through a CIP network.

Of the applicable CIP Networks, EtherNet/IP™ is the network of choice for high performance, synchronized multi-axis control. Other CIP networks such as ControlNet™ and DeviceNet™ could be applied to lower performance non-synchronized motion device applications such as simple variable frequency drives, velocity loop drives, and indexing drives.

4.2 Control modes

4.2.1 General

The Motion Device Axis Object covers the behavior of various motion control system devices that includes feedback devices, drive devices, standalone converters and motion I/O devices. For drive devices, the Motion Device Axis Object covers a wide range of drive types from simple variable frequency (V/Hz) drives, to sophisticated position control servo drives, with or without integral converters. Indeed, many commercial drive products can be configured to operate in any one of these different motion control modes depending on the specific application requirements. The attributes of the Motion Device Axis Object are therefore organized to addresss this broad range of functionality and the framework for this organization is described in 4.2.

IEC 61800-7-1 and this part of the IEC 61800-7 series are organised around the general philosophy that position control is the highest form of dynamic control. That is, position control implies velocity control, and velocity control implies acceleration control. Acceleration is related to torque or force by the inertia or mass of the load, respectively, acceleration control implies torque control. And finally, since motor torque or force is generally related to motor current by a torque or force constant, respectively, torque control implies current control. The torque or force constant can be a function of the motor magnets as in a Permanent Magnet motor, or the induced flux of an Induction motor.

Since acceleration, torque/force, and current are generally related by a constant, these terms are sometimes used interchangeably in the industry. For example IEC 61800-7-1 refers to a torque control loop rather than a current control loop. This specification attempts to differentiate between these control properties where applicable. This is particularly useful when the relationship between them is not static, such as when inertia/mass changes with position or time, or when the torque/force constant changes due to temperature change or motor flux variation.

4.2.2 Control methods

Within this basic control paradigm, there is latitude for different control methods, both closed loop and open loop. By closed loop, it is generally implied that there is a feedback signal that is used to drive the actual dynamics of the motor to match the commanded dynamics by servo action. In most cases, there is a literal feedback device to provide this signal, and in some cases, the signal is derived from the motor excitation (i.e. sensorless/encoderless operation). By open loop, it is implied that there is no application of feedback to force the actual dynamics to match the commanded dynamics. While precision and performance are the hallmarks of closed loop control, simplicity and economy are the hallmarks of open loop control.

4.2.3 Control nomenclature

Finally, as evident in the description above, linear and rotary control applications can affect the control nomenclature. While rotary applications speak of torque and inertia, linear applications speak of force and mass. For the purposes of this part of the IEC 61800-7 series, when referring to rotary nomenclature, the defined behavior can generally be applied to linear applications by substituting the terms, force for torque and mass for inertia. With that understanding, Figure 2 to Figure 8 follow the IEC 61800-7-1 nomenclature using torque rather than force without loss of generality.

4.2.4 Position control

4.2.4.1 General concepts

In position control application mode, either the application control program (command execution function) or the motion planner (move trajectory control function) provide a set-point value to the CIP Motion device via the cyclic data connection. The position control method can be either open loop or closed loop.

4.2.4.2 Open loop position control

A device configured for open loop position control applies to a class of drive devices called stepper drives. This type of drive is illustrated in Figure 2.

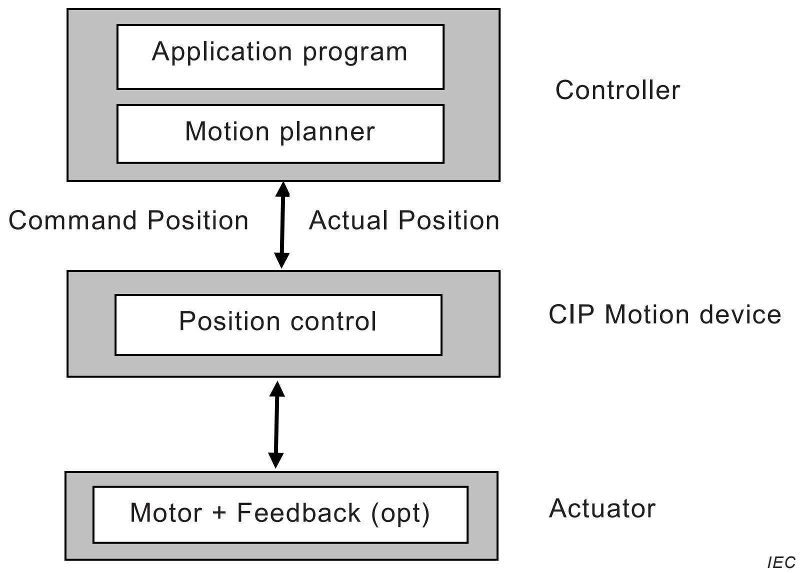

Figure 2 – Open loop position control

A feedback device for this configuration is optional. In the absence of a feedback device, actual position can be estimated by the drive and returned to the controller.

4.2.4.3 Closed loop position control

A motor control device configured for closed loop position control is traditionally referred to as position loop drive or position servo drive. A position servo drive implies an inner velocity and torque control loop as shown in Figure 3. The presence of the torque/current control loop sometimes results in this kind of drive being referred to as a vector drive.

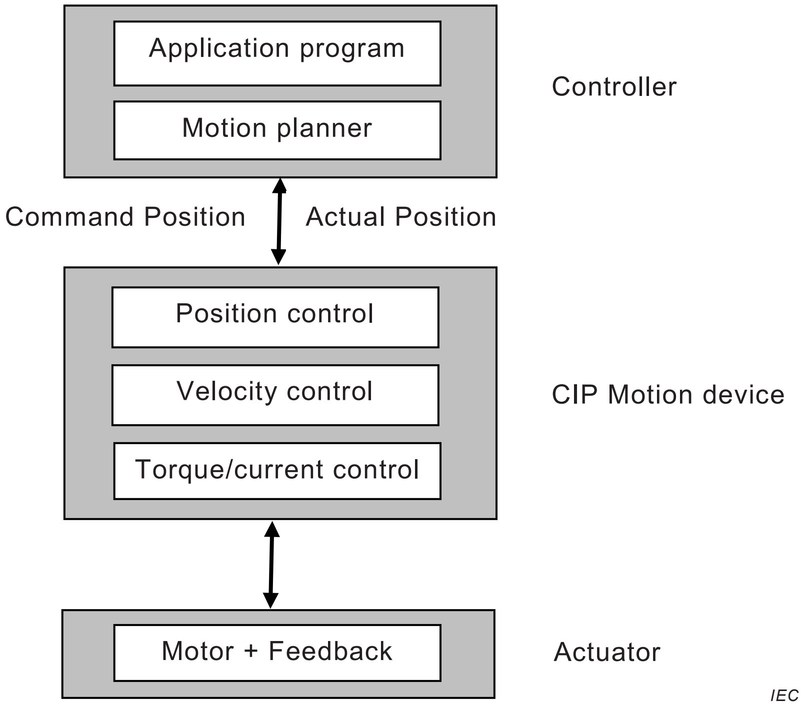

Figure 3 – Closed loop position control

A feedback device for this configuration is generally required to achieve good positioning accuracy. The feedback device may also be used to return Actual Velocity, and Actual Acceleration data to the controller via the cyclic data connection.

In addition to Command Position, the controller can pass Command Velocity and Command Acceleration for the purposes of forward control.

4.2.5 Velocity control

4.2.5.1 General concepts

In velocity control application mode, the application control program and motion planner provide a set-point value to the CIP Motion device via the cyclic data connection. The velocity control method can be either open loop or closed loop.

4.2.5.2 Open loop velocity control

A motor control device configured for open loop velocity control is traditionally referred to as variable frequency, or V/Hz, or VFD, drive. This type of drive is illustrated in Figure 4.

A feedback device for this configuration is optional. In the absence of a feedback device, actual velocity can be estimated by the drive and returned to the controller.

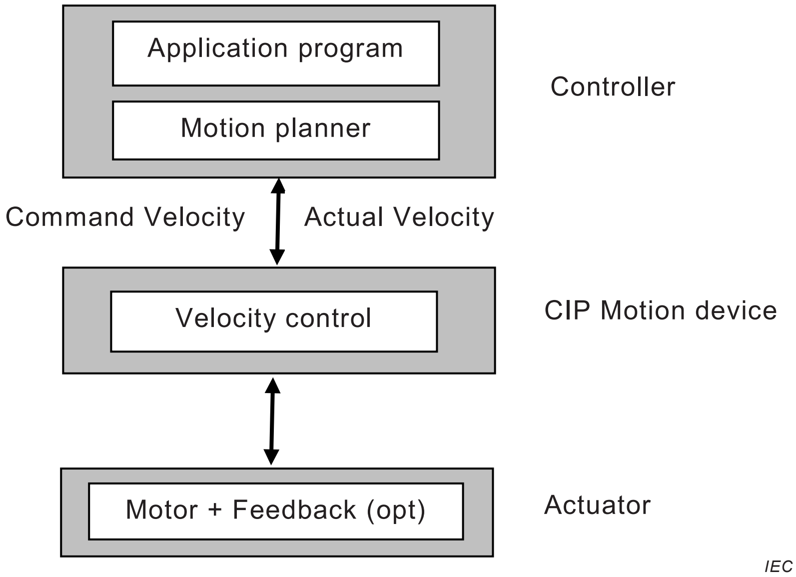

Figure 4 – Open loop velocity control

4.2.5.3 Closed loop velocity control

A motor control device configured for closed loop velocity control is traditionally referred to as velocity loop drive or velocity servo drive. A closed loop velocity control drive implies an inner torque/current control loop (see Figure 5) and therefore is sometimes referred to as a vector drive.

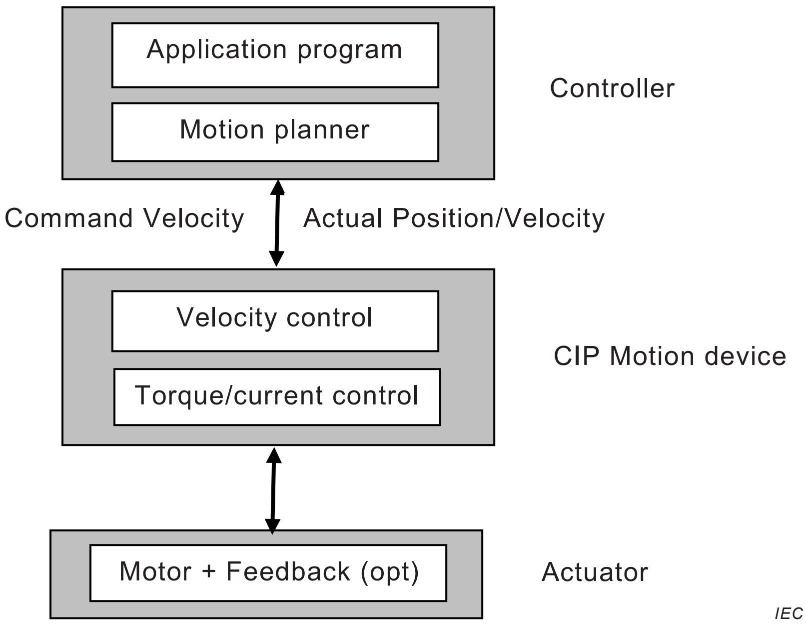

Figure 5 – Closed loop velocity control

A feedback device for the velocity loop drive configuration is optional. Tighter speed regulation is achieved when using a feedback device, particularly at low speed. When the feedback device is included, it may be used to return actual position, velocity, and acceleration data to the controller via the cyclic data connection. When the feedback device is not included, only estimated velocity can typically be returned to the controller.

In addition to Command Velocity, the controller can also pass Command Acceleration for the purposes of forward control.

4.2.6 Acceleration control

While neither a mainstream control mode in the industry, nor mentioned in IEC 61800-7-1, the acceleration control mode is included here to complete the dynamic progression from velocity control to torque control and because the Motion Device Axis Object can support an Acceleration Command, potentially derived from the controller's motion planner. In the acceleration control mode, the application control program and motion planner provide acceleration set-point values to the CIP Motion device via the cyclic data connection. The drive converts the acceleration set-point into a torque command using the estimated system inertia. Acceleration control works in concert with the inner torque/current control loop as shown in Figure 6.

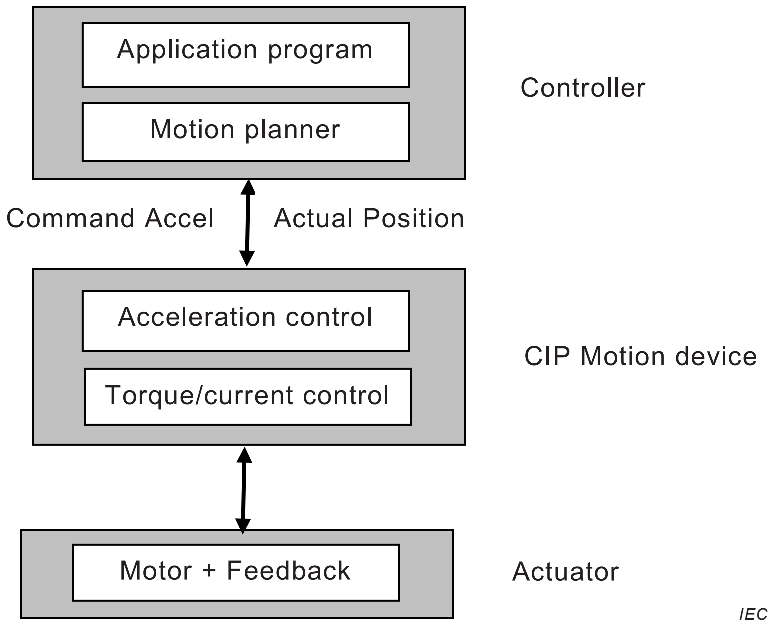

Figure 6 – Acceleration control

A feedback device for the acceleration control configuration is mandatory and may be used to return actual position, velocity, and acceleration data to the controller via the cyclic data connection.

4.2.7 Torque control

In torque control application mode, the application control program or the motion planner provide torque set-point values to the device via the cyclic data connection (see Figure 7). Because motor current and motor torque are generally related by a torque constant, Kt, torque control is often synonymous with current control.

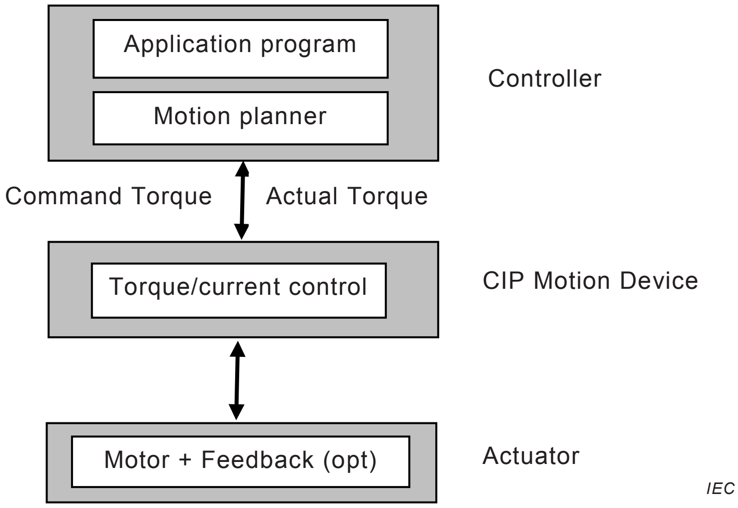

Figure 7 – Torque control

A position feedback device for this control mode is optional. If a feedback device is present, it may be used to return actual position, velocity, and acceleration data to the controller via the cyclic data connection.

4.2.8 No Control

The Motion Device Axis Object supports a "No Control" application mode where there is no dynamic motor control function. This mode is often used to support "feedback only" or "master feedback" functionality where a particular feedback channel in a CIP Motion drive device is serving as a master feedback source to the rest of the control system. This could also apply to integrated CIP Motion Encoder device types where the CIP Motion interface is applied directly to an Encoder.

In this "No Control" mode of operation, no set-point value is supplied to the CIP Motion device via the cyclic data connection, but actual position, velocity, and acceleration can be supplied by the device to the controller via the Cyclic Data Channel, if applicable. The No Control mode for Feedback Only functionality is illustrated in Figure 8.

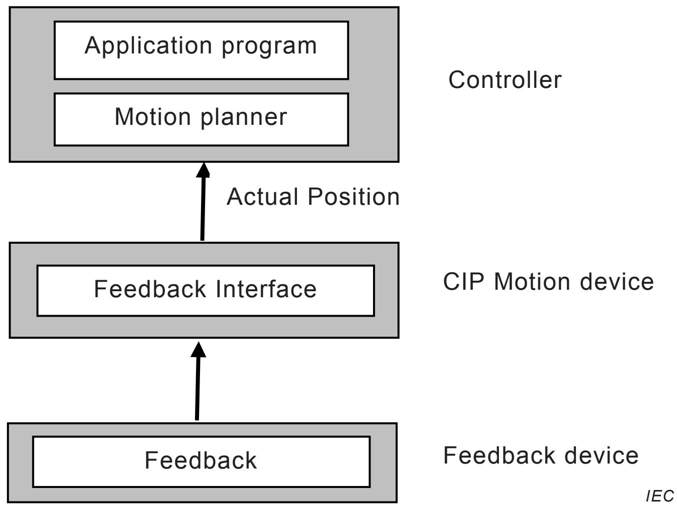

Figure 8 – No Control (Feedback Only)

No Control mode also applies to other CIP Motion device types, such as standalone Bus Power Converters and dedicated motion I/O device types. Since there is no feedback channel associated with these device types, no actual position is returned to the controller.

5 Data types

5.1 Data type overview

Table 1 shows references of data types used in this profile and the related definitions.

Table 1 – Data types

Data types used in CIP Motion	Reference to definition
BOOL	Boolean (see 5.3.1.1.2 of IEC 61158-5-2:2014)
SINT	Integer8 (see 5.3.1.4.2.2 of IEC 61158-5-2:2014)
INT	Integer16 (see 5.3.1.4.2.4 of IEC 61158-5-2:2014)
DINT	Integer32 (see 5.3.1.4.2.6 of IEC 61158-5-2:2014)
LINT	Integer64 (see 5.3.1.4.2.8 of IEC 61158-5-2:2014)
USINT	Unsigned8 (see 5.3.1.4.3.2 of IEC 61158-5-2:2014)
UINT	Unsigned16 (see 5.3.1.4.3.4 of IEC 61158-5-2:2014)
UDINT	Unsigned32 (see 5.3.1.4.3.6 of IEC 61158-5-2:2014)
ULINT	Unsigned64 (see 5.3.1.4.3.8 of IEC 61158-5-2:2014)
REAL	Float32 (see 5.3.1.4.1.2 of IEC 61158-5-2:2014)
LREAL	Float64 (see 5.3.1.4.1.4 of IEC 61158-5-2:2014)
SWORD/BYTE	Bitstring8 (see 5.3.1.2.2 of IEC 61158-5-2:2014)
WORD	Bitstring16 (see 5.3.1.2.4 of IEC 61158-5-2:2014)
DWORD	Bitstring32 (see 5.3.1.2.6 of IEC 61158-5-2:2014)
LWORD	Bitstring64 (see 5.3.1.2.8 of IEC 61158-5-2:2014)

5.2 Conventions

Hexadecimal numbers in this part of the IEC 61800-7 series may be represented by nn_{hex} (if their exact data type is not defined) or 0xnn, 0xnnnn (if their data type is specified).

6 CIP Motion drive profile

6.1 Object model

6.1.1 Object overview

The object model in Figure 9 represents a CIP Motion device.

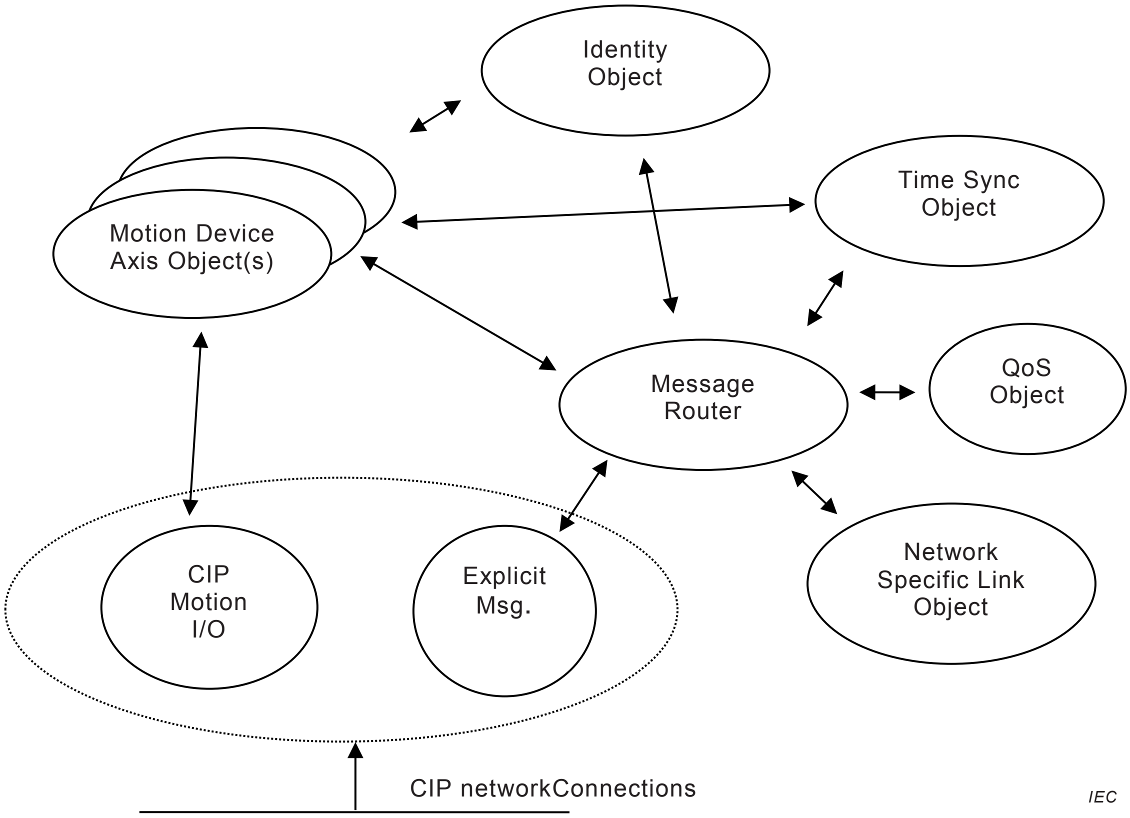

Figure 9 – Object Model for a CIP Motion device

Table 2 indicates:

- the object classes present in this device,
- whether or not the class is required,
- the number of instances present in each class.

Table 2 – Objects present in a CIP Motion device

Object class	Optional or required	Number of instances
CIP common required objects	Required	See IEC 61158-5-2 and IEC 61158-6-2
Motion Device Axis Object (0x42)	Required	1 per axis[b]
Time Sync Object (0x43)	Optional	1
QoS Object (0x48)	Conditional[a]	1

[a] Required for EtherNet/IP

[b] An axis is an abstraction associated with a moving machine component, a converter power structure or a motion I/O device.

Refer to IEC 61158-5-2, IEC 61158-6-2 and 6.4.6.1 for more details about these objects.

6.1.2 Object description

The object model in Figure 9 shows the main functional components of the CIP Motion device profile.

This object model also illustrates the use of multiple instances of a Motion Device Axis Object to implement a multi-axis motion device, such as a multi-axis drive. Each Motion Device Axis

Object instance governs the behavior of the associated axis. In this device profile, the term "axis" is synonymous with a "Motion Device Axis Object instance". The implemented content of the Motion Device Axis Object instances is dictated by the specific CIP Motion Device Type according to Table 3.

Table 3 – Motion Device Axis Object content by Device Type

Device type	Motion Device Axis Object content
CIP Motion Drive	Support for one or more of F, P, V, T Device Function Codes
CIP Motion Encoder	Support for E, but no support for F, P, V, T Device Function Codes
CIP Motion Converter	Support for B Device Function Code only
CIP Motion I/O	Support for I/O Device Function Code only

A single bi-directional I/O connection to the Motion Device Axis Object class instance provides a cyclic data path between the controller and each individual Motion Device Axis Object instance. This connection passes on a special data structure whose self-defining format can be used to transfer cyclic, event, and service related data.

An optional Time Sync Object is included in the object model to facilitate accurate time synchronization between CIP Motion controllers and drive devices for high performance motion control. For lower performance drives such as V/Hz (VFD) drives, or simple velocity servo drives, the Time Sync Object is not necessary for drive operation.

A CIP Motion I/O Connection is a standard, cyclic, bidirectional CIP I/O connection whose packets can vary in size, based on the formats outlined in 6.4.2.2.

6.2 How objects affect behavior

The objects for this device affect the device's behavior as shown in Table 4.

Table 4 – Object effect on behavior

Object class	Effect on behavior
CIP common required objects	See IEC 61158-5-2 and IEC 61158-6-2
Motion Device Axis Object	Provides dynamic control interface to drive, motor, and feedback components that comprise an axis.
Time Sync Object	Provide absolute time synchronization services between devices on the control network.
QoS Object	Provides configuration information associated with Ethernet QoS (Quality of Service) function.

6.3 Defining object interfaces

Objects supported for the CIP Motion device have the interfaces listed in Table 5.

Table 5 – Object interfaces

Object	Interface
CIP common required objects	See IEC 61158-5-2 and IEC 61158-6-2
Motion Device Axis Object	Message Router, CIP Motion I/O Connection, Time Sync Object, Identity Object
Time Sync Object	Message Router
QoS Object	Message Router

6.4 I/O connection messages

6.4.1 General

The CIP Motion device profile supports a Transport Class 1 point-to-point bi-directional I/O connection between the controller and the Motion Device Axis Object class: this I/O connection is specifically referred to as the CIP Motion I/O Connection.

The Motion Device Axis Object distributes the data in this connection to each instantiated Motion Device Axis Object instance.

6.4.2 CIP Motion I/O Connection

6.4.2.1 Overview

The following subclause specifies the CIP Motion I/O Connection format that includes the Controller-to-Device (C-to-D) Connection and the Device-to-Controller (D-to-C) Connection for bi-directional data transfer between a CIP Motion controller and a CIP Motion device, as shown in Figure 10.

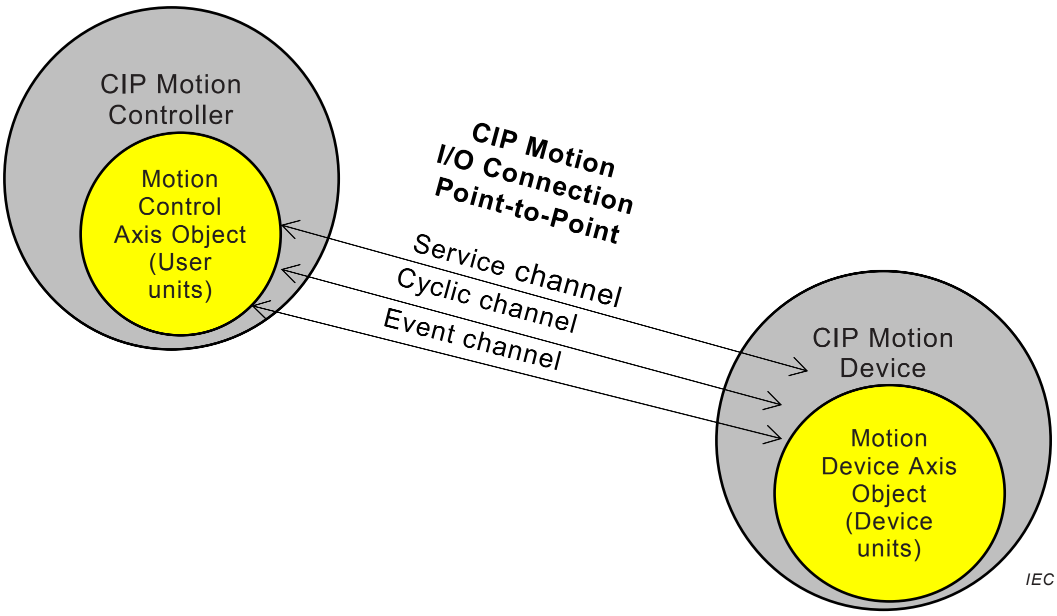

Figure 10 – CIP Motion I/O Connection model

6.4.2.2 CIP Motion I/O Connection structure

Both CIP Motion I/O Connection data structures (Controller-to-Device and Device-to-Controller) begin with a Connection Header followed by a Time Data Block that typically includes a 64-bit time stamp. The Time Data Block is then followed by one or more Instance Data Blocks for each axis supported by the device node.

The size and contents of the Time Data Block and Instance Data Blocks vary dynamically as determined by the associated headers. This ability to vary the contents of the data blocks from update to update allows the device and controller to only send data that has changed from the last update, dramatically reducing the overall size of the typical CIP Motion Connection packet.

Each Instance Data Block within the CIP Motion I/O Connection packet consists of three sets of data blocks associated with the cyclic, event, and service data channels. From the device's perspective, these three distinct data channels have different data processing priorities as illustrated in Figure 11.

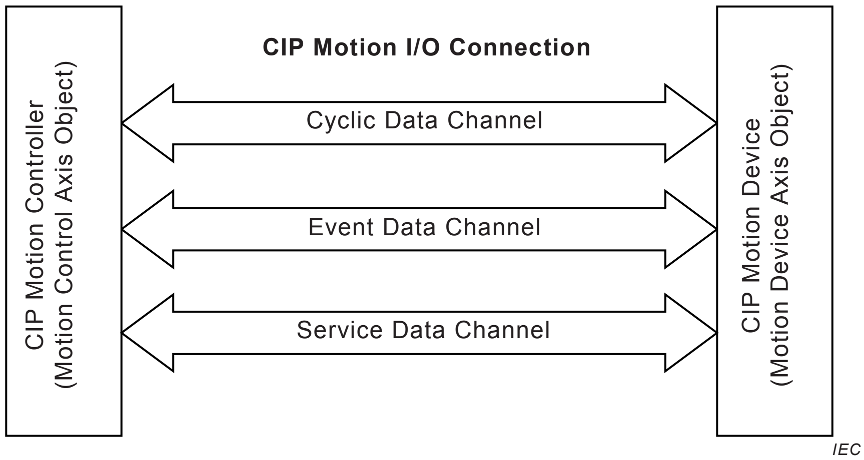

Figure 11 – CIP Motion I/O Connection channels

The specific functionality of these three data channels is as follows.

- Cyclic Data Channel: carries Cyclic Data Blocks that are sampled or calculated every Controller Update Period and synchronized with other nodes in the motion control system through use of distributed System Time. Cyclic data is high priority data that shall be immediately processed and applied to the device axis within one Device Update Period.
- Event Data Channel: carries event data associated with device event(s) (e.g. registration, homing, etc.) that have occurred within the last Controller Update Period. Event data is medium priority and shall be processed and applied within one Controller Update Period.
- Service Data Channel: carries data associated with service requests to read or write attribute values of the Motion Device Axis Object as part of on-line configuration and diagnostic functionality, as well as service requests to affect Motion Device Axis Object behavior as part of controller instruction execution. Service data has lowest priority and is typically buffered and processed as a background task. There is no guarantee that a service request will be processed within Controller Update Period.

Taken together, these three data channels provide a comprehensive controller to device data connection solution for industrial motion control.

6.4.2.3 I/O connection formats

An overview of the CIP Motion I/O Connection format is shown in Figure 12 and Figure 13.

Not shown in these format figures is the encapsulation associated with the Class 1 Transport that includes a Sequence Count. For a detailed description of the Class 1 Transport header refer to IEC 61158-5-2 and IEC 61158-6-2. Multi-octet data in the CIP I/O Connection data structure follows standard little-endian octet-addressing rule. In the figures, octet ordering reads left to right. The gray banners in the figures are section labels and are not part of the actual connection data structure. Unless otherwise stated, data structure elements defined as "reserved" or marked with a hyphen, "-", shall be set to zero (0).

← 32-bit Word (octet 1 | octet 2 | octet 3 | octet 4) →

Controller-to-Device Connection format
Connection Header
Time Data Block
Instance Data Blocks

Connection Header and Time Data Block			
Connection Format	Format Revision	Update ID	Node Control
Instance Count	–	Last Received ID	Time Data Set
Controller Time Stamp			
Controller Time Offset			

Instance Data Block
Instance Data Header
Cyclic Data Block
Cyclic Write Data Block
Event Data Block
Service Data Block

Instance Data Header			
Instance Num	–	Instance Blk Size	Cyclic Blk Size
Cyc. Cmd Blk Size	Cyc. Write Blk Size	Event Blk Size	Service Blk Size

Cyclic Data Block			
Control Mode	Feedback Mode	Axis Control	Control Status
Command Data Set	Actual Data Set	Status Data Set	Command Control
Cyclic Data			

Cyclic Write Data Block			
Cyclic Write Blk ID	–	Cyclic Read Blk ID	–
Cyclic Write Data			

Event Data Block			
Event Checking Control			
Reg Data Set	Home Data Set	Watch Data Set	–
Registration Event Data			
Home Event Data			
Watch Event Data			
Event Ack. ID1	Event Ack. Status 1	Event Ack. ID2	Event Ack. Status 2
…			

Service Data Block			
Transaction ID	Service Code	–	–
Service Specific Request Data			

IEC

Figure 12 – Controller-to-Device Connection format (Connection Point 2)

← 32-bit Word (octet 1 | octet 2 | octet 3 | octet 4) →

Device-to-Controller Connection format
Connection Header
Time Data Block
Instance Data Blocks

Connection Header and Time Data Block			
Connection Format	Format Revision	Update ID	Node Status
Instance Count	Node Fault/Alarm	Last Received ID	Time Data Set
Device Time Stamp			
Device Time Offset			
Lost Updates	Late Updates	–	
Timing Diagnostic Data			

Instance Data Block
Instance Data Header
Cyclic Data Block
Cyclic Read Data Block
Event Data Block
Service Data Block

Instance Data Header			
Instance Num	–	Instance Blk Size	Cyclic Blk Size
Cyc. Act. Blk Size	Cyc. Read Blk Size	Event Blk Size	Service Blk Size

Cyclic Data Block			
Control Mode	Feedback Mode	Axis Response	Response Status
–	Actual Data Set	Status Data Set	Axis State
Cyclic Data			

Cyclic Read Data Block			
Cyclic Write Blk ID	Cyclic Write Status	Cyclic Read Blk ID	Cyclic Read Status
Cyclic Read Data			

Event Data Block			
Event Checking Status			
Reg Data Ack	Home Data Ack	Watch Data Ack	–
Event ID	Event Status	Event Type	–
Event Position			
Event Time Stamp			
…			

Service Data Block			
Transaction ID	Service Code	General Status	Extended Status
Service Specific Response Data			

IEC

Figure 13 – Device-to-Controller Connection format (Connection Point 2)

6.4.3 Controller-to-Device Connection

6.4.3.1 General

To facilitate a detailed description of each of its constituent data elements, the CIP Motion Controller-to-Device Connection is organised as shown in Figure 14.

CIP Motion Controller-to-Device Connection format
Connection Header
Time Data Block
Instance Data Blocks

IEC

Figure 14 – CIP Motion Controller-to-Device Connection format

6.4.3.2 Controller-to-Device Connection Header

6.4.3.2.1 General structure

The Controller-to-Device Connection Header contains critical axis configuration information needed to parse the Instance Data Blocks. The fixed portion of the Connection Header is defined as shown in Figure 15.

Connection Header			
Connection Format	Format Revision	Update ID	Node Control

IEC

Figure 15 – Connection Header

- Connection Format: This byte determines the format of the CIP Motion Connection according to the definition in Figure 16.

Bit 7 Bit 0

Connection Format	
(Reserved)	Connection Type

IEC

Figure 16 – Connection Format

- Connection Type: This 4-bit value enumeration defines the CIP Motion Connection type as shown below. Valid values for a Controller-to-Device Connection are 2 for Fixed format and 6 for Variable format. Fixed connections have a fixed connection size during operation to support lower-performance CIP networks and are typically associated with a simple single axis device that does not support time synchronization services. Variable connections allow the connection data structure to vary in size during operation and are targeted for high-performance devices and CIP networks like EtherNet/IP.

  0 = Fixed Controller Peer-to-Peer Connection

  1 = Fixed Device Peer-to-Peer Connection

  2 = Fixed Controller-to-Device Connection

  3 = Fixed Device-to-Controller Connection

  4 = Variable Controller Peer-to-Peer Connection

  5 = Variable Device Peer-to-Peer Connection

  6 = Variable Controller-to-Device Connection

  7 = Variable Device-to-Controller Connection

  8 to 15 = Reserved.

  The Connection Format is not only used by the device to correctly parse the Controller-to-Device Connection data, but also by network diagnostic tools to display the individual elements of the data structure. If the device receives a Connection Format value that it cannot support, it generates a fault. In this case, the fault shall be Node Fault Error Code 4: Data Format Error. The Connection Format value shall not change for the life of the connection and, therefore, need only be checked by the device for the first received packet.

- Format Revision: This edition of IEC 61800-7-202 defines Format Revision 2. Devices utilizing this edition of IEC 61800-7-202 shall only recognize Format Revision 2; Format Revision 1 is therefore rendered obsolete. This value is incremented by 1 for every revision of the Controller-to-Device Connection format that impacts the interface. The Format Revision allows newer devices to support the connection formats generated by an older controller. It also allows older devices to recognise a newer connection format from a controller that it cannot support and to generate a fault. In this case, the fault shall be Node Fault Error Code 4: Data Format Error. Network diagnostic tools can also key off the Format Revision to know how to parse the connection data packet. At the time the I/O Connection is established, the Format Revision number is also indicated by the I/O Connection Point in the Application Path of the Forward_Open service (see 6.4). The Format Revision value shall not change for the life of the connection and, therefore, need only be checked by the device for the first received packet.
- Update ID: This cyclic transaction number is incremented every update period. The Update ID is like the CIP sequence count and is used by the device to determine whether the connection buffer contains fresh data. In the case where the device is not synchronized, or does not support time synchronization services, the time stamp data is either not included or invalid, so the Update ID is the only way for the device to detect new connection data. In this case, the Controller-to-Device Connection Update ID is applied to the next Device-to-Controller Connection Update ID.

  The Update ID also allows the device to determine if an update has been lost. Any skip in the Update ID is an indication that an update packet has been lost during transmission. If the number of consecutive missed updates exceeds the value given by the class attribute, Controller Update Delay High Limit, the Control Connection Update Fault is indicated by the Node Faults class attribute of the Motion Device Axis Object. This fault can only be cleared by closing and opening the CIP Motion I/O Connection. An optional Controller Update Delay Low Limit attribute is also available. If the number of consecutive missed updates exceeds the maximum value given by the class attribute, Controller Update Delay Low Limit, the Controller Connection Update Alarm is indicated by the Node Alarm class attribute of the Motion Device Axis Object. The alarm is cleared as soon as a fresh update is received.
- Node Control: This element is applied to the Motion Device Axis Object class attribute “Node Control”, which is used to control the state of the associated device communications node. See 7.2.2.1 (Motion Device Axis Object) for details.

6.4.3.2.2 Fixed Connection Header

If the Connection Format is a Fixed Controller-to-Device Connection, the above header is immediately followed by the Instance Data Block.

6.4.3.2.3 Variable Connection Header

If the Connection Format is a Variable Controller-to-Device Connection, then the Connection Header contains additional fields related to multi-axis device addressing and time stamping, as shown in Figure 17.

Connection Header			
Connection Format	Format Revision	Update ID	Node Control
Instance Count	–	Last Received ID	Time Data Set
Controller Time Stamp			
Controller Time Offset			

IEC

Figure 17 – Connection Header

- Instance Count: This value reflects the number of Instance Data Blocks present in the CIP Motion Controller-to-Device Connection data structure.

- Last Received ID: This is the Update ID of the last Device-to-Controller Connection data block actually processed by the controller. The Last Received ID is used by the device to determine if the data that was sent was processed successfully. For example, Fault Codes passed as part of the Device-to-Controller Connections Status Data Block are sequenced into the controller's fault log. It is recommended that the Last Received ID element be read by the device to determine if the last Fault Code was entered into the Fault Log, thus allowing the device to send the next Fault Code in the sequence.
- Time Data Set: This bit-mapped byte contains flags that determine the usage and format of the controller and device timing information, as specified in Table 6. In general, the Time Data Set value sent by the controller in the Controller-to-Device Connection not only determines the contents of the Controller-to-Device Connection data block, but it also represents the requested Time Data Set to be sent by the device in the Device-to-Controller Connection. The value of this element is transferred to the Motion Device Axis Object attribute "Time Data Set".

Table 6 – Time Data Set

Bit	Definition	Syntax
0	Time Stamp	0 = no time stamp 1 = time stamp included
1	Time Offset	0 = no time offset – use last offset 1 = time offset included
2	Update Diagnostics	0 = no update diagnostic data in D-to-C 1 = update diagnostic data in D-to-C
3	Time Diagnostics	0 = no time diagnostic data in D-to-C 1 = time diagnostic data in D-to-C
4 to 7	(Reserved)	

  – Time Stamp – if this bit is set, the Controller Time Stamp is included in the Time Data Set block. Setting this bit also instructs the device to send Device Time Stamps back to the controller in the Device-to-Controller Connection in the next update. It also instructs the device to send the Device Time Offset if its value has changed since the last update. If the device does not support time synchronization, the device simply returns a value of zero for the Time Stamp and does not ever return a Time Offset.
  – Time Offset – if this bit is set, the Controller Time Offset is included in the Time Data Set block. This bit is only set when there is a change to the Controller Time Offset value, a relatively infrequent occurrence. Note that this bit does NOT determine if the device sends its Device Time Offset in the next update. That is determined by the Time Stamp bit and by whether or not the Device Time Offset has changed since the last update.
  – Update Diagnostics – although there is no data associated with this bit in the Controller-to-Device Connection header, the Update Diagnostic bit in the Controller-to-Device Connection instructs the device to send Update Diagnostic data in the next Device-to-Controller Connection update, but only if one or more of the Update Diagnostic values have changed since the last update. This Update Diagnostic data includes the current number of Lost or Late Updates from the controller over the Controller-to-Device Connection. Update Diagnostics apply to devices regardless of the whether they support time synchronization.
  – Time Diagnostics – although there is no data associated with this bit in the Controller-to-Device Connection header, the Time Diagnostic bit in the Controller-to-Device Connection instructs the device to send Time Diagnostic data in the next Device-to-Controller Connection update. This Time Diagnostic data includes the Controller-to-Device Data Received time stamp and the Device-to-Controller Data Transmit time stamp. If the device does not support time synchronization, the device simply returns a value of zero for the Time Diagnostic elements.
- Controller Time Offset: This element is included in the Connection Header if the Time Offset bit is set in the Time Data Set element. The Controller Time Offset represents the

64-bit System Time Offset value associated with the Controller Time Stamp that follows. The value of this element is transferred to the Motion Device Axis Object class attribute of the same name. The Controller Time Offset value is used by the device to determine if System Time as defined in the controller is skewed relative to System Time in the device. Normally, System Time between the device and the controller are closely matched at any given time. But every few seconds, according to the IEC 61588:2009 based CIP Sync protocol, the time master of the system can correct the System Time reference by a significant amount of time, perhaps an hour. Indeed, even the master itself can change as a higher quality time master is discovered by the system. These step changes to the System Time reference propagate through to the various devices on the network in such a way that the controller and the device are running with skewed System Time values. The Controller Time Offset value can be used together with the System Time Offset for the device to both determine if System Time between the device and the controller is skewed and if so, to compensate for the skew using the System Time Offset Compensation algorithm.

The Controller Time Offset value changes relatively infrequently, i.e. as a result of a IEC 61588 driven Sync update, which typically occurs every second. Thus, it is recommended that the controller only load and send the Time Offset data if the value has changed. The device need only parse the Controller Time Offset data if the associated Time Offset bit is set in the Time Data Set element, otherwise it continues to use the last Controller Time Offset received.

- Controller Time Stamp: This element is included in the Connection Header if the Time Stamp bit is set in the Time Data Set element. The Controller Time Stamp represents the 64-bit System Time value at the beginning of the Controller Update Period when the controller's update timer event occurred. In other words the Controller Time Stamp marks, in absolute time, the beginning of the current CIP Motion Connection cycle. The value of this time stamp is calculated by the controller as the sum of the controller's local clock value when update timer event occurred and the controller's System Time Offset value. The Controller Time Stamp is therefore directly associated with the command data contained in the connection. The value of this element, when read, is transferred to the Motion Device Axis Object class attribute Controller Time Stamp. Taken together with the Controller Update Period, established by the RPI of the Forward_Open service, and the Command Target Update (discussed later), the device has all the information it needs compute command interpolation polynomials to correct command data values for differences between the device and controller update timing. These differences can occur when the Controller Update Period is not an integer multiple of the Device Update Period or when the device updates are phase shifted relative to the controller.

  When the connection is synchronized, the Controller Time Stamp can be used by the device along with the Controller Update Period to check for missed or late updates. If the difference between the last Controller Time Stamp and the current local Device Time Stamp exceeds the maximum value given by,

  Max Fault Delay = Controller Update Delay High Limit × Controller Update Period,

  the Control Connection Update Fault is indicated by the Node Faults class attribute of the Motion Device Axis Object. An optional Controller Update Delay Low Limit attribute is also available. If the difference between the last Controller Time Stamp and the current local Device Time Stamp exceeds the maximum value given by,

  Max Alarm Delay = Controller Update Delay Low Limit × Controller Update Period,

  the Controller Connection Update Alarm is indicated by the Node Alarm class attribute of the Motion Device Axis Object.

  If either the Controller Time Offset or the Device Time Offset values have changed since the last update, System Time Offset Compensation shall be applied to the Controller Time Stamp. See 6.4.6.7 for details. If the time offsets have not changed, the typical case, the Controller Time Stamp can be directly applied to the Motion Device Axis Object.

6.4.3.3 Instance Data Blocks

6.4.3.3.1 General structure

After the Connection Header are one or more Instance Data Blocks as determined by the above Instance Count. The Instance Data Block has the basic structure shown in Figure 18.

Instance Data Block
Instance Data Header
Cyclic Data Block
Event Data Block
Service Data Block

IEC

Figure 18 – Instance Data Block

6.4.3.3.2 Instance Data Header

The Instance Data Header contains critical axis configuration information needed to parse and apply the data contained in the three data channels (see Figure 19). This header is only included in the Variable Connection format to accommodate multi-axis device applications. Information within the header can be used by the device communications interface to copy the individual data blocks into separate fixed memory locations for processing.

If configured for a Fixed Connection format, only the Cyclic Data Block for a single axis instance is supported so there is no need for any information to specify instance number or block sizing. The Instance Data Header is therefore not included.

Instance Data Header			
Instance Num	–	Instance Blk Size	Cyclic Blk Size
Cyc. Cmd. Blk Size	Cyc. Write Blk Size	Event Blk Size	Service Blk Size

IEC

Figure 19 – Instance Data Header

- Instance Number: This is the number that identifies the specific Motion Device Axis Object instance the following Instance Data Block applies to. Motion Device Axis Object instances are created as a contiguous series of instance numbers starting with instance 1. Within the connection data structure, the Instance Numbers for each consecutive Instance Data Block shall be an ordinal sequence, i.e. 0, 1, 2, 3, etc.

  NOTE 1 Instance 0 is defined as the class instance and is typically only used during initialization to configure class attributes of the Motion Device Axis Object via the Service Data Block. Otherwise, the Instance Data Block sequence starts with instance 1.

  Thus, in theory, up to 255 axis instances can be serviced by the CIP Motion I/O Connection. The instance numbers sequence of the Controller-to-Device Connection determines the instance number sequence of the subsequent Device-to-Controller Connection.

- Instance Block Size: This value represents the size of the Instance Data Block in 32-bit words including the header. The Instance Block Size is useful when the device wants to directly access the next Instance Data Block without having to add the sizes of the cyclic, event, and service blocks. If the Instance Block Size for the Controller-to-Device Connection is zero, i.e. no Instance Data Block, then the Instance Block Size for the following Device-to-Controller Connection shall also be zero.

- Cyclic Block Size: This value represents the size of the Cyclic Data Block in 32-bit word units including the header.

  NOTE 2 For instance 0, the class instance, the Cyclic Block Size is 0 indicating that there is no Cyclic Data Block.

- Cyclic Command Block Size: This value represents the size of the Cyclic Command Data Block in 32-bit word units including the header. A Cyclic Command Block Size of 0 indicates that there is no Cyclic Command Data for the device to process.

- Cyclic Write Block Size: This value represents the size of the Cyclic Write Data Block in 32-bit word units including the header. A Cyclic Write Block Size of 0 indicates that there is no Cyclic Write data currently configured for transfer to the device and therefore there is no Cyclic Write Block included for the device to process.
- Event Block Size: This value represents the size of the Event Data Block in 32-bit word units including the header. If the Event Block Size for the Controller-to-Device Connection is zero, there is no active event checking, so the Event Block Size for the Device-to-Controller Connection shall also be zero. This effectively closes the Event Channel.
- Service Block Size: This value represents the size of the Service Data Block in 32-bit word units including the header. A Service Block Size value of 0 indicates there is no service request to process. If the Service Block Size for the Controller-to-Device Connection is zero, then the Service Block Size for the Device-to-Controller Connection shall also be zero. This effectively closes the Service Channel.

6.4.3.3.3 Cyclic Data Block

The Cyclic Data Block consists of high priority control data that is updated every connection cycle based on the Controller Update Period. The Cyclic Data Header at the top of the Cyclic Data Block is always included regardless of the connection format. This header contains key elements related to the content of the Cyclic Data Block of both the Controller-to-Device Connection and Device-to-Controller Connection, and the context of the data as determined by the Control Mode and Feedback Mode (see Figure 20). The header also provides a mechanism to control the state of the targeted device axis.

Cyclic Data Block			
Control Mode	Feedback Mode	Axis Control	Control Status
Command Data Set	Actual Data Set	Status Data Set	Command Control
Cyclic Data			

IEC

Figure 20 – Cyclic Data Block

- Control Mode: The lower 4-bits of this 8-bit element determine the control mode context of the command data as presently configured in the motion controller, as shown in Figure 21.

Bit 7 Bit 0

Control Mode	
reserved – 4 bits	Control Mode – 4 bits

IEC

Figure 21 – Control Mode

This value can be changed while on-line and even while the device axis is in the Running state. If a particular Control Mode transition is not supported by the device, an exception is generated that can be configured to perform any one of a number of actions in response to the illegal transition.

The value of this element is transferred to the Motion Device Axis Object attribute of the same name. The complete Control Mode attribute definition can be found in the attribute tables of the Motion Device Axis Object (see 7.3.2.2.1).

- Feedback Mode: The lower 4-bits of this 8-bit element determine the feedback context of the command data as presently configured in the motion controller, as shown in Figure 22.

Bit 7 Bit 0

Feedback Mode	
reserved – 4 bits	Feedback Mode – 4 bits

IEC

Figure 22 – Feedback Mode

Command position data can be referenced to feedback counts of either Feedback 1 or Feedback 2, or motor units for sensorless operation. This value can be changed while on-line and, if supported by the device, even while the device axis is in the Running state. If a particular Feedback Mode transition is not supported by the device, an exception is generated that can be configured to perform any one of a number of actions in response to the illegal transition.

The value of this element is transferred to the Motion Device Axis Object attribute of the same name. The complete Feedback Mode attribute definition can be found in the attribute tables of the Motion Device Axis Object (see 7.3.6.2).

- Axis Control: An 8-bit enumerated code that can be used to directly execute Axis State change operations that do not require either passing or returning device parameters, and therefore, do not require a CIP service to initiate. This mechanism is fully described in the State Control subclause of the Motion Device Axis Object (see 7.6.1.2). Valid enumerations for this data element are shown in Table 7.

Table 7 – Axis Control

Request code	Requested operation
0	No Request
1	Enable Request
2	Disable Request
3	Shutdown Request
4	Shutdown Reset Request
5	Abort Request
6	Fault Reset Request
7	Stop Process Request
8	Change Actual Position Reference Request
9	Change Command Position Reference Request
10 to 126	(Reserved)
127	Cancel Request
128 to 255	(Vendor Specific)

- Control Status: A bit mapped value that can be interrogated by the device to determine the status of the control during the Initialization process. The value of this element is transferred to the optional Motion Device Axis Object attribute number 89, Control Status, if supported by the device. Valid bits for this data element are shown in Table 8.

Table 8 – Control Status

<table>
<tr><th>Bit</th><th>Control Status bit</th><th>Description</th></tr>
<tr><td>0</td><td>Configuration Complete</td><td>This bit is set when the controller has completed configuration of all axis instance attributes during Initialization phase.</td></tr>
<tr><td>1</td><td>(Reserved)</td><td rowspan="5">See Attribute 89 of the Motion Device Axis Object</td></tr>
<tr><td>2</td><td>Converter Bus Up</td></tr>
<tr><td>3</td><td>Converter Bus Unload</td></tr>
<tr><td>4</td><td>Converter AC Power Loss</td></tr>
<tr><td>5 to 7</td><td>(Reserved)</td></tr>
</table>

- Command Data Set: This bit mapped value has a bit defined for each possible real-time command reference (see Table 9). Command data appears in the same order in the Command Data Set as the bit numbers, so Command Acceleration would appear before Command Torque in the real-time data structure of the Controller-to-Device Connection.

  The value of this element is transferred to the Motion Device Axis Object attribute “Command Data Set”.

Table 9 – Command Data Set

Bit	Controller Command Data Element	Data type
0	Command Position	LREAL or DINT [a]
1	Command Velocity	REAL
2	Command Acceleration	REAL
3	Command Torque	REAL
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	Unwind Cycle Count	DINT
7	Position Displacement	DINT

[a] Data Type for Command Position determined by Command Control element defined below. The Command Position data type can be either an LREAL or a DINT as determined by the Command Control element.

The above Controller Command Data Elements apply to the Controller-to-Device Connection's cyclic data structure and map to corresponding attributes in the Motion Device Axis Object as shown in Table 10.

The units of the Command Data Elements match the units defined for the associated Motion Device Axis Object attribute.

Table 10 – Command Data Element to Motion Device Axis Object attribute mapping

Bit	Command Data Element	Motion Device Axis Object attribute
0	Command Position (LREAL) Command Position (DINT)	Controller Position Command – Float Controller Position Command – Integer
1	Command Velocity	Controller Velocity Command
2	Command Acceleration	Controller Acceleration Command
3	Command Torque	Controller Torque Command
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	Unwind Cycle Count	Local variable
7	Position Displacement	Local variable

It is the job of the controller to ensure that the necessary Command Data Elements are included in the connection data to support the specified Control Mode.

- Actual Data Set: This bit mapped value has a bit defined for each possible real-time actual data attribute that is to be included in the Actual Data Set of the Device-to-Controller Connection's Instance Data Block in the next update (see Table 11). Actual data appears in the same order as the bit numbers, so Actual Position would appear before Actual Velocity in the Actual Data Set structure. Using this mechanism, the contents of the Actual Data Set may be changed at any time during device operation. The value of this element is transferred to the Motion Device Axis Object attribute "Actual Data Set".

Table 11 – Actual Data Set

Bit	Actual Data Element produced	Data type
0	Actual Position	DINT
1	Actual Velocity	REAL
2	Actual Acceleration	REAL
3	(Reserved)	(Reserved)
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	Unwind Cycle Count	DINT
7	Position Displacement	DINT

The above Actual Data Elements map to corresponding attributes in the Motion Device Axis Object as shown in Table 12. The units of the Actual Data Elements match the units defined for the associated Motion Device Axis Object attribute.

Table 12 – Actual Data Element to Motion Device Axis Object attribute Mapping

Bit	Actual Data Element	Motion Device Axis Object attribute
0	Actual Position	Position Feedback
1	Actual Velocity	Velocity Feedback
2	Actual Acceleration	Acceleration Feedback
3	(Reserved)	(Reserved)
4	(Reserved)	(Reserved)
5	(Reserved)	(Reserved)
6	Unwind Cycle Count	Local variable
7	Position Displacement	Local variable

- Status Data Set: This bit-mapped byte contains flags that determine what the controller requests for contents of the Status Data Set of the Device-to-Controller Connection's Instance Data Block in the next update (see Table 13). These bits may NOT solely determine if the device sends the associated Status data in the next Device-to-Controller Connection update. That may be determined by the device based on whether or not the associated Status data has changed since the last Device-to-Controller Connection update.

  Status data appears in the same order as the bit numbers, so Axis Fault Type/Code data would appear before, for example, Axis Fault Time Stamp data in the Status Data Set structure. Multiple attributes comprising a selected Status Data Element are transmitted in the order listed from top to bottom, so the Axis Fault Type is transmitted before Axis Fault Code. The definitions of each of these Status Data Elements can be found by looking up the corresponding Motion Device Axis Object attribute (see 7.3) in the Motion Device Axis Object specification. The value of this element is transferred to the Motion Device Axis Object attribute "Status Data Set".

Table 13 – Status Data Set

Bit	Status Data Element produced	Data type
0	Axis Fault Type Axis Fault Code Axis Fault Sub Code Axis Fault Action Axis Fault Time Stamp	USINT USINT USINT USINT LINT
1	Axis Alarm Type Axis Alarm Code Axis Alarm Sub Code Axis Alarm State Axis Alarm Time Stamp	USINT USINT USINT USINT LINT
2	Axis Status Axis Status – Mfg	DWORD DWORD
3	Axis I/O Status Axis I/O Status – Mfg	DWORD DWORD
4	Axis Safety Status Axis Safety Status – Mfg Axis Safety State Pad[3]	DWORD DWORD USINT USINT[3]
5	(Reserved)	(Reserved)
6	(Vendor specific)	(Vendor specific)
7	(Vendor specific)	(Vendor specific)

If any of these bits are set and the associated status value has changed in the device, the device shall return the associated status value in the next available Device-to-Controller update. Status data values change relatively infrequently, so there can be many connection updates where various elements of the Status Data Set remain static. It is recommended that the device take advantage of this fact and only set the Status Data Set bit in the Device-to-Controller update when one of the associated status values has changed. Thus, the device need only load, say, the Axis Fault data associated with Status Data Set bit 0 when there is a new fault to report. Otherwise, it leaves bit 0 clear. On the controller side of the connection, the controller shall parse and process the Axis Fault data only if Status Data Set bit 0 is set. To insure that the data has been received, parsed, and processed by the controller before clearing the Status Data Set bit, the device can examine the Last Update ID of the next Controller-to-Device Connection update to confirm that it matches or exceeds the Update ID of this Device-to-Controller update.

- Command Control: The byte contains information needed to control the fine interpolation algorithm and determine the target time of the command data to the Axis Control structure (see Table 14). The value of this element is transferred to the Motion Device Axis Object attribute "Command Control".

Table 14 – Command Control

Bit	Definition	Syntax
0 to 1	Command Target Update	0 = Immediate 1 = Extrapolate (+1 Update Period) 2 = Interpolate (+2 Update Periods) 3 = (Reserved)
2 to 3	Command Position Data Type	0 = LREAL (64-bit Floating Point) 1 = DINT (32-bit signed Integer) 2 to 3 = (Reserved)
4 to 7	(Reserved)	

Command Target Update – This 2-bit integer defines a specific time relative to the Connection Time Stamp that the Command Data is targeted for, as defined by the controller's motion planner. The absolute Command Target Update is the sum of the

Controller Time Stamp from the controller and the product, Command Target Update × Controller Update Period.

– A Command Target Update of 0 implies that the Command Data is targeted for the beginning of the current update cycle and, thus, needs to be applied to the control structure immediately. In this case, there is no need for any fine interpolation. This situation can occur when the Controller Update Period is significantly shorter than the Device Update Period, or when the controlled motion of the axis is nearly constant during the span of the Controller Update Period.

– A Command Target Update of 1 implies that the target for the Command Data is the next Connection Update timer event. In this case, the command interpolator functions primarily as an extrapolator that estimates the next Command Data value based on the present trajectory of the axis. This is a typical setting when the Controller Update Period is comparable to the Device Update Period or when the controlled motion of the axis is relatively constant during the span of the Controller Update Period.

– A Command Target Update of 2 implies that the target for the Command Data is two Connection Update timer events from the Connection Time Stamp. In this case, the command interpolator can compute a smooth trajectory based on the current dynamics of the motor to reach the Command Data value at the targeted time. This is true fine interpolation and is applicable when the Controller Update Period is significantly larger than the Device Update Period.

– Table 15 summarizes the relationship between the Update Period Ratio and the recommended Command Target Update.

Table 15 – Command Target Update vs. Update Period Ratio

Update Period Ratio (Controller Update Period / Device Update Period)	Recommended Command Target Update
2 or more	2 = Interpolate (+2 Update Periods)
½ to 2	1 = Extrapolate (+1 Update Period)
½ or less	0 = Immediate

Command Position Data Type – This 2-bit integer defines the data type of the Command Position data element. This allows flexibility in handling the many different representations for command position in the industry.

– A Command Position Data Type of 0 corresponds to the LREAL, or double precision floating point data type. Support for this data type is preferred in the implementation since it provides fraction command count information to the device resulting in smoother motion.

– A Command Position Data Type of 1 corresponds to the DINT, or 32-bit signed integer data type. This data type is applicable to simple drive devices that either do not require the precision of a floating point command position value or do not have sufficient hardware support for double precision floating point math.

- Cyclic Command Data: The Cyclic Command Data contains high priority data that needs to be applied to the associated drive axis instance during the next device update. This block consists of command data elements that are applied as references to the device's control algorithms and explicitly determined by the Command Data Set element in the Cyclic Command Data Header.

6.4.3.3.4 Cyclic Write Data Block

The Cyclic Write Data Block can be used to synchronously update one or more targeted Motion Device Axis Object configuration parameters within the device. This mechanism can be used in conjunction with a Function Block program to implement sophisticated outer loop control, gain scheduling, and dynamic limiting algorithms. Unlike service channel Set Axis Attribute service requests, which may take several device update cycles to process, the Cyclic Write Data mechanism guarantees that the targeted parameter is applied at the next available device update.

The Cyclic Write Data Block is only supported in the Variable Connection format (see Figure 23).

Cyclic Write Data Block			
Cyclic Write Blk ID	–	Cyclic Read Blk ID	–
Cyclic Write Data			

IEC

Figure 23 – Cyclic Write Data Block

The associated header for this block contains key elements related to the content of the Cyclic Write Data and the Cyclic Read Data Block of the next Device-to-Controller Connection update.

- Cyclic Write Block ID: This 8-bit ID determines the pre-defined Cyclic Write Block structure to apply to the Cyclic Write Data for this update. Cyclic Write Block structures are defined using the Set Cyclic Write Data List service. The initial value for the Cyclic Write Block ID is 0, which corresponds to a null set of Cyclic Write Data. A successful response to the Set Cyclic Write Data List service includes a new, incremented, Cyclic Write Block ID that can be used in the next connection update to pass cyclic data in this format. The previous Cyclic Write Block ID and its associated structure shall be maintained until the controller requests transmission of the new Cyclic Write Block ID at which point the old Cyclic Write Block ID and structure definition are no longer required.
- Cyclic Read Block ID: This 8-bit ID determines the pre-defined Cyclic Read Block structure to apply to the Cyclic Read Data for next Device-to-Controller Connection update. Cyclic Read Block structures are defined using the Set Cyclic Read Data Block service. The initial value for the Cyclic Read Block ID is 0, which corresponds to a null set of Cyclic Read Data. A successful response to the Set Cyclic Read Data List service includes a new, incremented, Cyclic Read Block ID that can be used in the next connection update to allow the device to use the new Cyclic Read Data format for next available Device-to-Controller Connection update.
- Cyclic Write Data: The Cyclic Write Data contains high priority data that needs to be applied to the associated device axis instance during the next device update. This block consists of parametric data elements that are applied to Motion Device Axis Object attributes that are used by the control algorithms. The contents of the Cyclic Write Data are explicitly determined by the structure identified by the Cyclic Write Block ID found in the Cyclic Write Data Header. The ordering of the attribute data in the Cyclic Write Data Block is determined by the ordering of the Attr IDs in the Set_Cyclic_Write_List request. Attribute data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding shall be added to maintain word alignment.

  EXAMPLE Figure 24 shows a Cyclic Write Data Block with 3 attributes where Attributes 1 and 3 are 32-bit values while Attribute 2 is a 16-bit value.

Attribute value 1	
Attribute value 2	16-bits – reserved
Attribute value 3	

IEC

Figure 24 – Cyclic Write Data Block example

6.4.3.4 Event Data Block

The Event Data Block is used to convey information regarding the event channel. In particular, the Event Data Block for the Controller-to-Device Connection is used to control the arming of event checking functions in the device as well as acknowledge receipt of event notifications from the device that are sent via the Device-to-Controller Connection's Event Data Block.

The Event Data Block for the Controller-to-Device Connection has the format shown in Figure 25.

Event Data Block			
Event Checking Control			
Reg Data Set	Home Data Set	Watch Data Set	–
Registration Event Data			
Home Event Data			
Watch Event Data			
Event Ack. ID1	Event Ack. Status 1	Event Ack. ID2	Event Ack. Status 2
…			

IEC

Figure 25 – Event Data Block

When the Event Data Block is present, the Event Checking Control element determines which additional elements are included. The elements starting with Reg Data Set and extending through Watch Event Data constitute Extended Data and will only exist if the Extended Format bit is set in the Event Checking Control word. The Event Ack ID # and Event Ack Status # elements will be repeated from 0 to 7 times as determined by the Event Block Count field in the Event Checking Control word. If an odd number of Event Blocks are included, the Event Data Block will be padded to a 32-bit boundary with two unused bytes.

- Event Checking Control: This 32-bit word is copied into the Motion Device Axis Object attribute of the same name that is used to enable various device inputs, for example marker and registration inputs, to generate events. When these events occur, the device captures both the time and exact axis position when the event occurred. The last 4 bits of the Event Checking Control element is a special bit field, the least significant 3 bits of which specify the number of active events, which is literally the number of Event Acknowledge IDs listed in this Event Data Block. The most significant bit enables the extended Event Data Block format. A complete definition for the Event Checking Control attribute can be found in 7.3.7.2.1.

  The basic Event Control mechanism works as specified in Table 16.

Table 16 – Basic Event Cycle

Controller action	Connection data	Device action
Controller sets the appropriate Event Checking Control bit to look for a specific event condition to occur. Multiple bits in the Event Checking Control word may be set at any given time.	→ Event Checking bit = 1 → Event Block Count = 0 → Extended Format = 0	Device detects the Event Checking Control bit is set and then initiates the requested event checking action.
Controller sees the Event Checking Status bit from the device is set indicating that specified event trigger is "armed" at the device. Starting with this update the controller can process any event notifications from the device that match the specified event condition.	← Event Checking bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device sets the corresponding Event Checking Status bit to acknowledge that the device is now actively checking for the specified event condition. Starting with this update the device can include an event notification for the specified event condition.
Controller maintains the Event Checking bit to continue checking for the specified event.	→ Event Checking bit = 1 → Event Block Count = 0 → Extended Format = 0	Device sees Event Checking Control bit set and therefore continues checking for the specified event.
Controller continues to check for new Event Notifications from device.	← Event Checking bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device continues to set the corresponding Event Checking Status bit to indicate that event checking is active.

Controller action	Connection data	Device action
Controller processes the new Event Notification, storing the Event ID, checking the Event Status, and then processing the Event Data Block based on the Event Type.	← Event Checking bit = 1 ← Event Block Count = 1 ← Extended Format = 0 ← Event ID = id ← Event Status = 0 (success) ← Event Type = type ← Event Data Block	Device detects the specified event condition, increments the Event ID, and sends an Event Notification Data Block with this ID to the controller. The specific event condition is identified by its Event Type. Since the associated Auto-rearm bit is clear in the Event Checking Control word, the device discontinues checking for the specified event. However, the Event Checking Status bit remains set until the Event Checking Control bit is cleared by the controller. Note that notifications for other events can be sent in the same update. The total number of Event Notification Data Blocks in the update is represented by the Event Block Count.
Controller sends the Event Acknowledge with the associated Event ID and Event Status to the device. Since Auto-rearm is not enabled, the controller clears the associated Event Checking Control bit.	→ Event Checking bit = 0 → Event Block Count = 1 → Extended Format = 0 → Event ID = id → Event Status = 0 (success)	Device detects the Event Checking Control bit is clear and processes the Event Acknowledge Data Block.
Controller sees the Event Checking Status bit from the device is clear indicating that the event cycle is complete. Controller is now able to set the Event Checking Control bit again. So, the minimum time between when the Event Notification was received by the controller to the time when the device is rearmed for the next event is one Controller Update Period.	← Event Checking bit = 0 ← Event Block Count = 0 ← Extended Format = 0	Device clears the corresponding Event Checking Status bit. Since the device received a successful Event Acknowledge, there is no further need to send the Event Notification Data Block for that Event ID.
Controller may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.	→ Event Block Size = 0 → No Event Block	Device sees Event Block Size is zero indicating that there is no Event Block data to process.
Device sees Event Block Size is zero indicating that there is no Event Block data to process.	← Event Block Size = 0 ← No Event Block	Device may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.

If the device supports Device Scaling functionality, the Extended Event Data Block format is used to exchange addition event information between the controller and the device. The Extended Event cycle works as specified in Table 17.

Table 17 – Extended Event Cycle

Controller action	Connection data	Device action
Controller sets the appropriate Event Checking Control bit to look for a specific event condition to occur. Multiple bits in the Event Checking Control word may be set at any given time. Using the Extended Event Checking Data block format, the controller passes any parameters needed by the event checking function in the Registration, Home, and Watch Event Data Blocks. The Event Format bit in the Event Checking Control element shall be set to include Extended Event Checking Data. Extended Event Checking Data is required when the device is configured for Device Scaling.	→ Event Checking bit = 1 → Event Block Count = 0 → Extended Format = 1 → Extended Event Data	Device detects the Event Checking Control bit is set, loads any associated event checking parameters from the Extended Event Checking Data block, and then initiates the requested event checking action.
Controller sees the Event Checking Status bit from the device is set indicating that specified event trigger is "armed" at the device. Starting with this update the controller can process any event notifications from the device that match the specified event condition. If the controller previously sent event checking parameters via the Extended Event Checking Data block, it also checks the returned Registration, Home, and Watch Data Set values to determine if the device has processed the Registration, Home, and Watch Event Data sent in the previous update. If the Data Ack values match the Data Set value that were sent, the Extended Event Checking Data has been processed by the device.	← Event Checking bit = 1 ← Event Block Count = 0 ← Extended Format = 1 ← Extended Event Data Ack	Device sets the corresponding Event Checking Status bit to acknowledge that the device is now actively checking for the specified event condition. If the device received event checking parameters from the Extended Event Checking Data block it acknowledges receipt of parameters passed in the Registration, Home, and Watch Event Data Blocks by returning the Registration, Home, and Watch Data Set values as the Data Ack values, respectively. Starting with this update the device can include an event notification for the specified event condition.
Controller maintains the Event Checking bit to continue checking for the specified event. With the device acknowledging receipt of the Extended Event Checking Data, the Controller clears the Extended Format bit for the Controller-to-Device Connection update.	→ Event Checking bit = 1 → Event Block Count = 0 → Extended Format = 0	Device sees the Event Checking Control bit set and therefore continues checking for the specified event.
Controller continues to check for new Event Notifications from device.	← Event Checking bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device continues to set the corresponding Event Checking Status bit to indicate that event checking is active. With no Extended Event Checking Data to acknowledge, the Device clears the Extended Format bit.

Controller action	Connection data	Device action
Controller processes the new Event Notification, storing the Event ID, checking the Event Status, and then processing the Event Data Block based on the Event Type.	← Event Checking bit = 1 ← Event Block Count = 1 ← Extended Format = 0 ← Event ID = x ← Event Status = 0 ← Event Type = y ← Event Data Block	Device detects the specified event condition, increments the Event ID, and sends an Event Notification Data Block with this ID to the controller. The specific event condition is identified by its Event Type. Since the associated Auto-rearm bit is clear in the Event Checking Control word, the device discontinues checking for the specified event. However, the Event Checking Status bit remains set until the Event Checking Control bit is cleared by the controller. Note that notifications for other events can be sent in the same update. The total number of Event Notification Data Blocks in the update is represented by the Event Block Count.
Controller sends the Event Acknowledge with the associated Event ID and Event Status to the device. Since Auto-rearm is not enabled, the controller clears the associated Event Checking Control bit.	→ Event Checking bit = 0 → Event Block Count = 1 → Extended Format = 0 → Event ID = id → Event Status = 0 (success)	Device detects the Event Checking Control bit is clear and processes the Event Acknowledge Data Block.
Controller sees the Event Checking Status bit from the device is clear indicating that the event cycle is complete. Controller is now able to set the Event Checking Control bit again. So, the minimum time between when the Event Notification was received by the controller to the time when the device is rearmed for the next event is one Controller Update Period.	← Event Checking bit = 0 ← Event Block Count = 0 ← Extended Format = 0	Device clears the corresponding Event Checking Status bit. Since the device received a successful Event Acknowledge, there is no further need to send the Event Notification Data Block for that Event ID.
Controller may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.	→ Event Block Size = 0 → No Event Block	Device sees Event Block Size is zero indicating that there is no Event Block data to process.
Device sees Event Block Size is zero indicating that there is no Event Block data to process.	← Event Block Size = 0 ← No Event Block	Device may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.

In the case of a Registration event where Auto-rearm Event Checking is requested, the event handling sequence is as specified in Table 18.

Table 18 – Basic Event Cycle with Auto-rearm

Controller action	Connection data	Device action
Controller sets the appropriate Event Checking Control bit to look for a specific event condition to occur and also sets the Auto-rearm bit for that event condition. Multiple bits in the Event Checking Control word may be set at any given time.	→ Event Checking bit = 1 → Event Auto-rearm bit = 1 → Event Block Count = 0 → Extended Format = 0	Device detects the Event Checking Control bit is set and then initiates the requested event checking action.
Controller sees the Event Checking Status bit from the device is set indicating that specified event trigger is “armed” at the device. Starting with this update the controller can process any event notifications from the device that match the specified event condition.	← Event Checking bit = 1 ← Event Auto-rearm bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device sets the corresponding Event Checking Status bit to acknowledge that the device is now actively checking for the specified event condition and also sets the corresponding Auto-rearm bit. Starting with this update the device can include an event notification for the specified event condition.
Controller maintains the Event Checking bit to continue checking for the specified event.	→ Event Checking bit = 1 → Event Auto-rearm bit = 1 → Event Block Count = 0 → Extended Format = 0	Device sees the Event Checking Control bit set and therefore continues checking for the specified event.
Controller continues to check for new Event Notifications from device.	← Event Checking bit = 1 ← Event Auto-rearm bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device continues to set the corresponding Event Checking Status bit to indicate that event checking is active.
Controller processes the new Event Notification, storing the Event ID, checking the Event Status, and then processing the Event Data Block based on the Event Type.	← Event Checking bit = 1 ← Event Auto-rearm bit = 1 ← Event Block Count = 1 ← Extended Format = 0 ← Event ID = id1 ← Event Status = 0 (success) ← Event Type = type ← Event Data Block	Device detects the specified event condition, increments the Event ID, and sends an Event Notification Data Block with this ID to the controller. The specific event condition is identified by its Event Type. Since the corresponding Auto-rearm bit is set, the device continues checking for the specified event. The Event Checking Status bit remains set until the Event Checking Control bit is cleared by the controller. If the Device detects another specified event condition prior to transmission of the first event to the controller, it increments the Event ID again, and appends another Event Notification with this ID to the Event Data Block to the controller. Note that notifications for other events can also be sent in the same update. The total number of Event Notification Data Blocks in the update is represented by the Event Block Count.
Controller sends the Event Acknowledge with the associated Event ID and Event Status to the device. Since the Auto-rearm bit is enabled, it leaves the associated Event Checking bit set to allow the device to continue checking for the next event.	→ Event Checking bit = 1 → Event Auto-rearm bit = 1 → Event Block Count = 1 → Extended Format = 0 → Event ID = id1 → Event Status = 0 (success)	Device detects the Event Checking Control bit is set and processes the Event Acknowledge Data Block. Since the corresponding Auto-rearm bit is set, the device continues checking for the specified event. The device can now send another Event Notification if the specified event has occurred again.

Controller action	Connection data	Device action
Controller continues to check for new Event Notifications from device.	← Event Checking bit = 1 ← Event Auto-rearm bit = 1 ← Event Block Count = 0 ← Extended Format = 0	Device sets the corresponding Event Checking and Auto-rearm Status bits indicating that the device is continuing to check for events. Since the device received a successful Event Acknowledge, there is no further need to send the Event Notification Data Block for that Event ID.
Controller processes the new Event Notification, storing the Event ID, checking the Event Status, and then processing the Event Data Block based on the Event Type.	← Event Checking bit = 1 ← Event Auto-rearm bit = 1 ← Event Block Count = 1 ← Extended Format = 0 ← Event ID = id2 ← Event Status = 0 ← Event Type = type ← Event Data Block	Device continues checking for the specified event condition and sends an Event Notification Data Block to the controller whenever the event condition occurs. The specific event condition is identified by its Event Type. The Event Checking Status bit remains set until the Event Checking Control bit is cleared by the controller. With the Auto-rearm feature, event checking is continuously enabled, insuring that no registration events are missed during the normal one cycle delay in re-arming the event checking mechanism. The down side of Auto-rearm feature is that it can generate a multitude of uninteresting events to process, in fact, multiple specified events per Controller Update Period. Therefore, these events shall either be buffered in the controller in an event array attribute or filtered based on the event data. The latter case is how Windowed Registration functionality is implemented; the Windowed Registration feature checks each registration event that arrives from the device to see if the Event Position yields an absolute position value that is within the configured Registration Window. If the computed registration position is outside the window, the event is thrown away. If the computed registration position is within the window the registration position and registration time stamp is stored in the controller
Controller sends the Event Acknowledge with the associated Event ID and Event Status to the device. Since the Auto-rearm bit is enabled, it leaves the associated Event Checking bit set to allow the device to continue checking for the next event.	→ Event Checking bit = 1 → Event Auto-rearm bit = 1 → Event Block Count = 1 → Extended Format = 0 → Event ID = id2 → Event Status = 0 (success)	Device detects the Event Checking Control bit is clear and processes the Event Acknowledge Data Block. Since the corresponding Auto-rearm bit is set, the device continues checking for the specified event. The device can now send another Event Notification if the specified event has occurred again. This Event Notification – Event Acknowledge cycle repeats until the controller clears the associated Event Checking Control bit.
…	…	…
Controller clears the Event Checking Control bit and the Auto-rearm bit to disable event checking.	→ Event Checking bit = 0 → Event Auto-rearm bit = 0 → Event Block Count = 0 → Extended Format = 0	Device detects the Event Checking Control bit is clear and disables event checking action.

Controller action	Connection data	Device action
Controller sees the Event Checking Status bit from the device is clear indicating that the event checking is disabled.	← Event Checking bit = 0 ← Event Auto-rearm bit = 0 ← Event Block Count = 0 ← Extended Format = 0	Device clears the corresponding Event Checking Status bit.
Controller may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.	→ Event Block Size = 0 → No Event Block	Device sees Event Block Size is zero indicating that there is no Event Block data to process.
Device sees Event Block Size is zero indicating that there is no Event Block data to process.	← Event Block Size = 0 ← No Event Block	Device may now set the Event Block Size to zero for all subsequent updates until such time as there is request to check for new events.

In the case of an Extended Registration event where Auto-rearm Event Checking is requested, the Auto-rearm sequence above simply includes the Extended Event data as shown in Table 17.

- Registration Data Set: This bit mapped byte determines the contents of Registration Event Data Block that contains parameters needed to setup the registration event checking function, as specified in Table 19. The data in this block appears in the same order as the Registration Data Set bit numbers, so for example, Reg 1 Pos Window would appear before Reg 1 Neg Window in the Event Checking Data Block structure. The definitions of each of these Event Checking Data elements can be found by looking up the corresponding attribute in the Motion Device Axis Object specification. The value of this element is transferred to the Motion Device Axis Object attribute, Registration Data Set. This mechanism is not restricted to initial setup, but can also be used to change parameters associated with registration event checking function while the checking function is active.

Table 19 – Registration Data Set

Bit	Registration Data Element	Data type
0	Reg 1 Pos Window	DINT (max window) DINT (min window)
1	Reg 1 Neg Window	DINT (max window) DINT (min window)
2	Reg 2 Pos Window	DINT (max window) DINT (min window)
3	Reg 2 Neg Window	DINT (max window) DINT (min window)
4 to 7	(Reserved)	(Reserved)

- Home Data Set: This bit mapped byte determines the contents of Home Event Data Block that contains parameters needed to setup the home event checking function, as specified in Table 20. The data in this block appears in the same order as the Home Data Set bit numbers, so for example, Home Torque Threshold would appear before Home Torque Time. The definitions of each of these Event Checking Data elements can be found by looking up the corresponding attribute in the Motion Device Axis Object specification. The value of this element is transferred to the Motion Device Axis Object attribute, Home Data Set. This mechanism is not restricted to initial setup, but can also be used to change parameters associated with home event checking function while the checking function is active.

Table 20 – Home Data Set

Bit	Home Data Element	Data type
0	Home Torque Threshold	REAL
1	Home Torque Time	REAL
2 to 7	(Reserved)	(Reserved)

- Watch Data Set: This bit mapped byte determines the contents of Watch Event Data Block that contains parameters needed to setup the watch event checking function, as specified in Table 21. The data in this block appears in the same order as the Watch Data Set bit numbers, so for example, Watch 1 Position would appear before Watch 2 Position. The definitions of each of these Event Checking Data elements can be found by looking up the corresponding attribute in the Motion Device Axis Object specification. The value of this element is transferred to the Motion Device Axis Object attribute, Watch Data Set. This mechanism is not restricted to initial setup, but can also be used to change parameters associated with watch event checking function while the checking function is active.

Table 21 – Watch Data Set

Bit	Watch Data Element	Data type
0	Watch 1 Position	DINT
1	Watch 2 Position	DINT
2 to 7	(Reserved)	(Reserved)

- Event Acknowledge ID: Transaction number assigned to this event by the original event notification. Each event is assigned a new Event ID by incrementing the current Event ID stored in the device. Using the Event ID, the device is able to match the event acknowledgement to the appropriate event notification to complete the event data transaction.
- Event Acknowledge Status: Enumerated value indicating controller response to the event. A value of 0 indicates that the event was successfully processed. A non-zero value indicates that an error occurred in the event processing and the event shall be resent.

6.4.3.5 Service Data Block

The Service Data Block allows one service request per instance to be sent to the device in a given update. The service request requires a specific service response from the device indicating success or an error. In some cases, the response service contains requested data. In any case, the service request data persists in the Controller-to-Device Connection data structure until the controller receives the associated service response from the device.

Each service request is represented by a block of data organized as shown in Figure 26.

NOTE By design, the first 4 bytes of the Service Data Block do not follow the traditional CIP standard messaging format. That is primarily because this connection structure is, fundamentally, a CIP Implicit I/O connection, not an Explicit Messaging connection. However, in the case of a Fixed Connection format, the Service Specific Request Data defined below is sent via an Explicit Messaging connection and follows the CIP rules for explicit service request format.

Service Data Block			
Transaction ID	Service Code	–	–
Service Specific Request Data			

IEC

Figure 26 – Service Data Block

- Transaction ID: Transaction number assigned to this service request by the controller. Each service request is assigned a new Transaction ID by incrementing the current Transaction ID stored in the controller. Using the Transaction ID, the controller is able to match the service response to the appropriate service request and complete the service transaction.

- Service Code: Identifier that determines the object specific service request that follows. The list of supported Service Codes can be found in the Object Specific Services subclause of this part of the IEC 61800-7 series (see 7.5.1). CIP Common services are not applicable to the Service Data Block.
- Service Specific Request Data: The format and syntax of the Service Specific Request Data depends on the specified Service Code. This is true regardless of whether the service specific request data is passed in the Controller-to-Device Connection or as part of an Explicit messaging connection.

6.4.4 Device-to-Controller Connection

6.4.4.1 General

Like the Controller-to-Device Connection data structure described above, the CIP Motion Device-to-Controller Connection is organised as shown in Figure 27.

CIP Motion Device-to-Controller Connection format
Connection Header
Time Data Block
Instance Data Blocks

IEC

Figure 27 – CIP Motion Device-to-Controller Connection format

6.4.4.2 Device-to-Controller Connection Header

6.4.4.2.1 General structure

The Device-to-Controller Connection Header contains critical axis configuration information needed to parse the Device-to-Controller Connection data block. The fixed portion of the Connection Header is defined as shown in Figure 28.

Connection Header			
Connection Format	Format Revision	Update ID	Node Status

IEC

Figure 28 – Connection Header

- Connection Format: Same as Controller-to-Device definition except the required value for the Connection Type is either 3, indicating a Fixed Device-to-Controller Connection type or 7, indicating a Fixed Device-to-Controller Connection type.

  0 = Fixed Controller Peer-to-Peer Connection

  1 = Fixed Device Peer-to-Peer Connection

  2 = Fixed Controller-to-Device Connection

  3 = Fixed Device-to-Controller Connection

  4 = Variable Controller Peer-to-Peer Connection

  5 = Variable Device Peer-to-Peer Connection

  6 = Variable Controller-to-Device Connection

  7 = Variable Device-to-Controller Connection

  8 to 15 = Reserved.

  The Connection Format can be used to correctly parse the Device-to-Controller Connection data, not only by the device, but also by network diagnostic tools. The Connection Format value shall not change for the life of the connection and, therefore, need only be checked by the controller for the first received packet.
- Format Revision: The Format Revision number is 2 for any device that utilizes this edition of IEC 61800-7-202. Since controllers utilizing this edition only recognize Format Revision

2, Format Revision 1 is rendered obsolete and need not be supported by any CIP Motion device. This value is incremented by 1 for every revision of the Device-to-Controller Connection format that impacts the interface. The Format Revision allows newer controllers to support the connection formats generated by older devices. It also allows older controllers to recognize a newer connection format from a device that it cannot support and generate an appropriate error to its application. Network diagnostic tools can also key off the Format Revision to know how to parse the connection data packet. At the time the I/O Connection is established, the Format Revision number is also indicated by the I/O Connection Point in the Application Path of the Forward_Open service (see 6.4).The Format Revision value shall not change for the life of the connection and, therefore, need only be checked by the controller for the first received packet.

- Update ID: The Device-to-Controller Connection Update ID shall match the Update ID of the Controller-to-Device Update ID for a given cycle, and therefore shall be incremented every update period. Note that, when the device is in Synchronous Mode, this does not imply that the Controller-to-Device packet has to be processed prior to the Device-to-Controller packet being assembled to provide a matching Update ID. Maintaining matching Update IDs only requires the device increment the Device-to-Controller Update ID by one every cycle. Once the Update IDs are matched at Start-up they remain matched by incrementing every cycle thereafter. In the case where the associated Controller-to-Device packet is lost or late, the Device-to-Controller Update ID shall be incremented as if the Controller-to-Device packet had arrived on time. This allows the CIP Motion controller to ride through a lost or missed Controller-to-Device packets and maintain synchronization with matching Update IDs.

  The Update ID is like the CIP message sequence count and is used by the controller to determine whether the connection buffer contains fresh data. If the Update ID, as seen by the controller, has not changed (late packet) or has skipped an Update ID (lost packet), the controller rides through the late or lost update using an extrapolation method based on the previous axis trajectory until a fresh update arrives. Like the CIP Motion device, if the number of consecutive missed updates reaches a configured limit, the controller declares a connection synchronization fault.

- Node Status: Contains bits used to indicate the status of the associated device communications node. The value of this element is derived from the Motion Device Axis Object class attribute of the same name. See 7.2.2.2 (Motion Device Axis Object) for details.

6.4.4.2.2 Fixed Connection Header

If the Connection Format is a Fixed Device-to-Controller Connection, the above header is immediately followed by the Instance Data Block.

6.4.4.2.3 Variable Connection Header

If the Connection Format is a Variable Device-to-Controller Connection, then the connection header contains additional fields related to multi-axis device addressing and time stamping (see Figure 29).

Connection Header			
Connection Format	Format Revision	Update ID	Node Status
Instance Count	Node Fault/Alarm	Last Received ID	Time Data Set
Device Time Stamp			
Device Time Offset			
Lost Updates	Late Updates	–	
Data Received Time Stamp			
Data Transmit Time Stamp			

IEC

Figure 29 – Connection Header

- Instance Count: Same as Controller-to-Device definition.
- Node Fault/Alarm: This 8-bit element contains two 4-bit codes for fault and alarm conditions associated with the device communications node, as specified in Figure 30.

The values for these elements are derived from the Motion Device Axis Object class attributes of the same name. For details refer to the Motion Device Axis Object.

Bit 7	Bit 0
Node Fault/Alarm	
Node Fault Code – 4 bits	Node Alarm Code – 4 bits

IEC

Figure 30 – Node Fault/Alarm

- Last Received ID: This is the Update ID of the last Controller-to-Device connection data block actually processed by the device. The Last Received ID is used by the controller to determine if the data that was sent was processed successfully by the device. For example, the Controller Time Offset is sent by the controller in the Controller-to-Device Connection only if the value has changed since the last update. It is recommended that the Last Received ID element be read by the controller to determine if the Controller Time Offset has been processed by the device and can therefore be removed from subsequent Controller-to-Device Connection updates.
- Time Data Set: Same as Controller-to-Device definition.
- Device Time Offset: This element is included in the Connection Header if the Time Stamp bit is set in the Time Data Set element. The Device Time Offset element represents the 64-bit System Time Offset value associated with the Device Time Stamp that follows. The Device Time Offset value is used by the controller to determine if System Time as defined in the device is skewed relative to System Time in the controller. Normally, System Time between the device and the controller are closely matched at any given time. But every few seconds, according to the IEC 61588:2009-based CIP Sync protocol, the time master of the system can correct the System Time reference, sometimes by a significant amount of time, perhaps as much as an hour. Indeed, even the master itself can change as a higher quality time master is discovered by the system. These step changes to the System Time reference propagate through to the various devices on the network in such a way that the controller and the device are running with skewed System Time values. The Device Time Offset value can be used together with the System Time Offset for the controller to both determine if System Time between the device and the controller is skewed and if so, to compensate for the skew.
- Device Time Stamp: This element is included in the Connection Header if the Time Stamp bit is set in the Time Data Set element. The Device Time Stamp represents the 64-bit System Time value, in nanoseconds, when the device's update timer event occurred that is associated with the actual data in the connection structure, for example when the actual data was captured. It is calculated by the device as the sum of the device's local clock value when update timer event occurred and the device's System Time Offset value. With the Device Time Stamp, the controller has all the information it needs to correct actual data values for differences between the device and controller update timing that result when the Controller Update Period is not an integer multiple of the Device Update Period or when the device updates are phase shifted relative to the controller. It is assumed in this timing model that the Device Time Stamp is registered to the beginning of the Device Update Period and is also the time when feedback was last captured. In the case where the Device Time Stamp does not match the local update time stamp of the controller, the controller extrapolates the actual response data value based on trajectory to correspond to the controller's time stamp. The timing diagram in Figure 31 illustrates how axis position data from the device is adjusted by the controller based on the relative time stamps between the device and the controller.

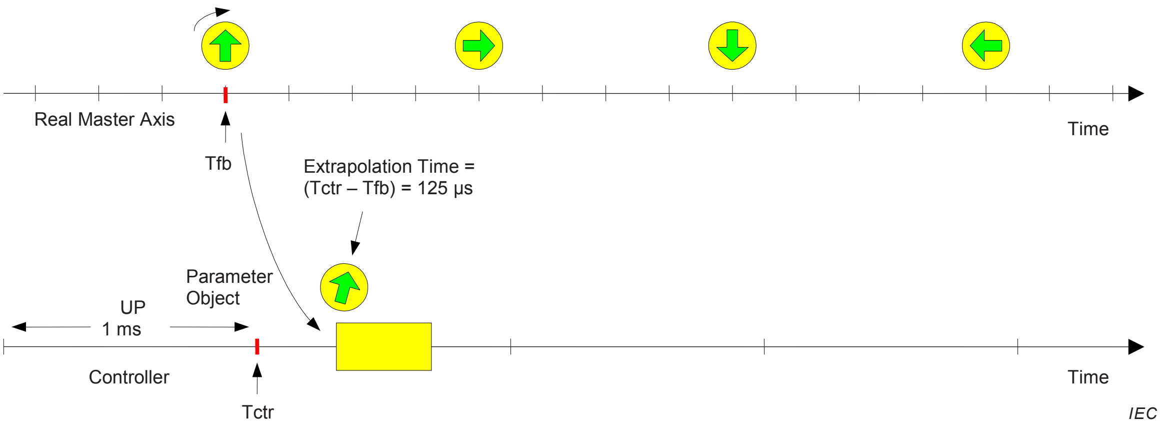

Figure 31 – Adjustment of actual position data based on Device Time Stamp

- Lost Updates: This element is included in the Connection Header if the Update Diagnostics bit is set in the Time Data Set element. The Lost Updates value represents the number of lost Controller-to-Device Connection packets detected since the connection was opened and synchronized. Lost packets are detected by examining the Update ID of received packets. A packet is determined to be lost when its expected Update ID is skipped. The Lost Packet class attribute is incremented for every Update ID that is skipped. The counter rolls over every 256 missed updates.
- Late Updates: This element is included in the Connection Header if the Update Diagnostics bit is set in the Time Data Set element. The Late Updates value represents the number of late Controller-to-Device Connection packets detected since the connection was opened and synchronized. Late packets are detected by examining the Time Stamp of received packet. A packet is determined to be late when the difference between its Controller Time Stamp and the device's Data Received Time Stamp exceeds the Controller Update Period. The counter rolls over every 256 late updates.
- Data Received Time Stamp: This element is included in the Connection Header if the Time Diagnostics bit is set in the Time Data Set element. The Data Received Time Stamp represents the 64-bit System Time value at the moment that the last Controller-to-Device Connection data, indicated by the Last Update ID, was written into device memory and ready for processing by the application layer. The time stamp units are nanoseconds. This value, when combined with controller's corresponding data transmit time stamp, can be used by the controller to generate Controller-to-Device Connection data delivery statistics.
- Data Transmit Time Stamp: This element is included in the Connection Header if the Time Diagnostics bit is set in the Time Data Set element. The Data Transmit Time Stamp represents the 64-bit System Time value at the moment that the application layer has initiated transmission of the Device-to-Controller Connection data to the controller. The time stamp units are nanoseconds. This value, when combined with controller's corresponding data received time stamp, can be used by the controller to generate Device-to-Controller Connection data delivery statistics.

6.4.4.3 Instance Data Blocks

6.4.4.3.1 General structure

After the Connection Header are one or more Instance Data Blocks as determined by the above Instance Count. The Instance Data Block is very similar to that of the Controller-to-Device Connection and has the basic structure shown in Figure 32.

Instance Data Block
Instance Data Header
Cyclic Data Block
Event Data Block
Service Data Block

IEC

Figure 32 – Instance Data Block

6.4.4.3.2 Instance Data Header

The Instance Data Header contains critical axis configuration information needed to parse and apply the data contained in the three data channels (see Figure 33). This header is only included in the Variable Connection format to accommodate multi-axis device applications. Information within the header can be used by the device communications interface to copy the individual data blocks into separate fixed memory locations for processing.

If configured for a Fixed Connection format, only the Cyclic Data Block for a single axis instance is supported so there is no need for any information on instance number or block sizing. Hence, the Instance Data Header is not included in the connection structure.

Instance Data Header			
Instance Num	–	Instance Blk Size	Cyclic Blk Size
Cyc. Act. Blk Size	Cyc. Read Blk Size	Event Blk Size	Service Blk Size

IEC

Figure 33 – Instance Data Header

- Instance Number: Same as Controller-to-Device definition.
- Instance Block Size: Same as Controller-to-Device definition.
- Cyclic Block Size: Same as Controller-to-Device definition.
- Cyclic Actual Block Size: This value represents the size of the Cyclic Actual Data Block in 32-bit word units including the header. A Cyclic Actual Block Size of 0 indicates that there is no Cyclic Actual Data for the controller to process.
- Cyclic Read Block Size: This value represents the size of the Cyclic Read Data Block in 32-bit word units including the header. A Cyclic Read Block Size of 0 indicates that there is no Cyclic Read data currently configured for transfer to the controller and therefore there is no Cyclic Read Block included for the controller to process.
- Event Block Size: Same as Controller-to-Device definition. An Event Block Size of 0 indicates that there is currently no active event checking in progress.
- Service Block Size: Same as Controller-to-Device definition. A Service Block Size of 0 indicates that there is currently no service request to respond to.

6.4.4.3.3 Cyclic Data Block

The Cyclic Data Header at the top of the Cyclic Data Block of the Device-to-Controller Connection is always included regardless of the connection format. This header contains key elements related to the content of the Cyclic Data Block and the context of the data within the block with respect to the device (see Figure 34). Most of these elements are established by, and are therefore direct copies of, corresponding elements of the previous Controller-to-Device Connection Cyclic Data Block. Thus, the content of the Cyclic Data Block for the Device-to-Controller Connection is ultimately determined by the controller.

Cyclic Data Block			
Control Mode	Feedback Mode	Axis Response	Response Status
–	Actual Data Set	Status Data Set	Axis State
Cyclic Actual Data			
Cyclic Status Data			

IEC

Figure 34 – Cyclic Data Block

- Control Mode: Same as Controller-to-Device definition.
- Feedback Mode: Same as Controller-to-Device definition.
- Axis Response: The 8-bit Axis Response is an enumerated value that is used for handshaking with the corresponding Axis Control element of the Controller-to-Device Connection to directly initiate device operations that do not require a CIP service request. Valid Acknowledge Codes match the corresponding Request Codes of the Axis Control element, and are shown in Table 22.

Table 22 – Axis Response

Acknowledge Code	Axis Response
0	No Acknowledge
1	Enable Acknowledge
2	Disable Acknowledge
3	Shutdown Acknowledge
4	Shutdown Reset Acknowledge
5	Abort Acknowledge
6	Fault Reset Acknowledge
7	Stop Process
8	Change Actual Position Reference Acknowledge
9	Change Command Position Reference Acknowledge
10 to 126	(Reserved)
127	Cancel Acknowledge
128 to 255	(Vendor Specific)

This Device Command/Axis Response mechanism for initiating state changes is fully described in the State Control subclause of the Motion Device Axis Object (see 7.6.1.2).

- Response Status: When there is a non-zero Acknowledge Code in the Axis Response, a Response Status value is also provided to indicate success or failure of the requested Axis Control operation. A Response Status of 0 indicates success, while a non-zero value indicates an error. The Response Status values comply with the CIP specification for General Status codes (see IEC 61158-6-2).
- Actual Data Set: Same as Controller-to-Device definition.
- Status Data Set: Same as Controller-to-Device definition. Status Data Set bits are only set when, 1) the corresponding bit in the Controller-to-Device connection’s Status Data Set is set, and 2) there is change in the associated status data values since the last connection update.
- Axis State: This data element contains the enumerated Axis State value indicating the current state of this device axis instance according to the Motion Device Axis Object State Model.
- Cyclic Actual/Status Data: The Cyclic Actual/Status Data contains high priority data that needs to be applied to the associated device axis instance during the next device update. This block consists of actual data elements and status data elements that are consumed by the controller as explicitly determined by the Actual Data Set and the Status Data Set elements in the Cyclic Data Header. See the Controller-to-Device definitions in 6.4.3 for details of Cyclic Actual/ Status Data structure. Cyclic Actual Data shall be referenced to the Device Time Stamp.

6.4.4.3.4 Cyclic Read Data Block

The Cyclic Read Data Block can be used to synchronously update one or more targeted Motion Control Axis Object attributes within the controller based on the current value of associated attributes in the device. This mechanism can be used in conjunction with, for example, an IEC 61131-3 based Function Block program to implement sophisticated outer loop control based on a wide variety of available Axis Control signals. Unlike service channel Get Axis Attribute service requests, which may take several device update cycles to process, the Cyclic Read Data mechanism guarantees the targeted parameter is updated every connection cycle.

The Cyclic Read Data Block is only supported in the Variable Connection format (see Figure 35).

Cyclic Read Data Block			
Cyclic Write Blk ID	Cyclic Write Status	Cyclic Read Blk ID	Cyclic Read Status
Cyclic Read Data			

IEC

Figure 35 – Cyclic Read Data Block

The associated header for this block contains key elements related to the content of the Cyclic Read Data as well as the Cyclic Write Data from the previous Controller-to-Device connection update.

- Cyclic Write Block ID: This 8-bit ID determines the pre-defined Cyclic Write Block structure that was sent in the last Controller-to-Device connection update. A successful Cyclic Write Data update is indicated by echoing the associated Cyclic Write Block ID in the following Cyclic Read Data Block header. If the device determines that there is an error associated with the Cyclic Write Data, the device identifies the error with a non-zero Cyclic Write Status value.
- Cyclic Write Status: The Cyclic Write Status value is provided to indicate success or failure of the associated Cyclic Write Data update. A Cyclic Write Status of 0 indicates success, while a non-zero value indicates an error. The Cyclic Write Status values comply with the CIP specification for General Status codes (see IEC 61158-6-2).
- Cyclic Read Block ID: This 8-bit ID determines the pre-defined Cyclic Read Block structure to apply to the Cyclic Read Data for this update. Cyclic Read Block structures are defined using the Set Cyclic Read Data List service. A successful response to this service includes a new Cyclic Read Block ID that can be used in the next connection update to pass cyclic data in this format. If the device determines that there is an error associated with the Cyclic Read Data, the device identifies the error with a non-zero Cyclic Read Status value and no Cyclic Read Data.
- Cyclic Read Status: The Cyclic Read Status value is provided to indicate success or failure of the associated Cyclic Read Data update. A Cyclic Read Status of 0 indicates success, while a non-zero value indicates an error. The Cyclic Read Status values comply with the CIP specification for General Status codes (see IEC 61158-6-2).
- Cyclic Read Data: The Cyclic Read Data contains high priority data that needs to be applied to the associated controller axis instance. This block consists of signal and status data elements that are to be scaled and applied to corresponding Motion Control Axis Object attributes. The contents of the Cyclic Read Data are explicitly determined by the structure identified by the Cyclic Read Block ID found in the Cyclic Read Data Header. The ordering of the attribute data in the Cyclic Read Data Block is determined by the ordering of the Attr IDs in the Set_Cyclic_Read_List request. Attribute data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding may be added to maintain word alignment.

  EXAMPLE Figure 36 shows a Cyclic Read Data Block with 3 attributes where Attributes 1 and 3 are 32-bit values while Attribute 2 is a 16-bit value.

Attribute value 1	
Attribute value 2	16-bits – reserved
Attribute value 3	

IEC

Figure 36 – Cyclic Read Data Block example

6.4.4.4 Event Data Block

The Event Data Block allows multiple event notifications to be sent to the controller in a given update. Each event notification requires a specific event acknowledge indicating success or an error. The event notification data persists in the Device-to-Controller Connection data structure until the device receives the corresponding event acknowledgement from the controller.

The Event Data Block for the Device-to-Controller Connection has the format shown in Figure 37.

Event Data Block			
Event Checking Status			
Reg Data Ack	Home Data Ack	Watch Data Ack	–
Event ID 1	Event Status 1	Event Type 1	–
Event Position 1			
Event Time Stamp 1			
Event ID 2	Event Status 2	Event Type 2	–
Event Position 2			
Event Time Stamp 2			
…			

IEC

Figure 37 – Event Data Block

- Event Checking Status: This 32-bit word indicates if the device is currently checking for events based on various device inputs, for example marker, home, and registration inputs. Event checking is initiated when the corresponding Event Checking Control bit is set in the Controller-to-Device Connection. When an event occurs, the device captures both the time and exact axis position and passes this information to the controller in Event Data Blocks. But for the controller to process the event data, the corresponding Event Checking Status bit shall be set. For more detail on how this word is used in event operations, refer to the event mechanism sequence in 6.4.3.4. The last 4 bits of the Event Checking Control element is a special bit field, the least significant 3 bits of which specify the number of active events, which is literally the number of Event IDs listed in this Event Data Block. The most significant bit enables the extended Event Data Block format. A complete definition for the Event Checking Status attribute can be found in 7.3.7.2.2.

  When the Event Data Block is present, the Event Checking Status element determines which additional elements are included. The word including Reg Data Ack, Home Data Ack, and Watch Data Ack are only included if the Extended Format bit is set in the Event Checking Status word. The Event ID #, Event Status #, Event Type #, Event Position #, and Event Time Stamp # elements will be repeated from 0 to 7 times as determined by the Event Block Count field in the Event Checking Status word.
- Registration Data Ack: This bit mapped byte is used to acknowledge receipt of any Registration Data Block parameters from a previous Controller-to-Device Connection update. There is no data in the Device-to-Controller Connection data structure associated with these bits.
- Home Data Ack: This bit mapped byte is used to acknowledge receipt of any Home Data Block parameters from a previous Controller-to-Device Connection update. There is no data in the Device-to-Controller Connection data structure associated with these bits.
- Watch Data Ack: This bit mapped byte is used to acknowledge receipt of any Watch Data Block parameters from a previous Controller-to-Device Connection update. There is no data in the Device-to-Controller Connection data structure associated with these bits.
- Event ID: Transaction number assigned to this specific event by the original event notification. Each event is assigned a new Event ID by incrementing the current Event ID stored in the device. Using the Event ID, the device is able to match the event acknowledgement to the appropriate event notification to complete the event data transaction.

- Event Status: Enumerated value indicating the status of the original event notification. A value of 0 indicates that the event was successfully detected by the device. A non-zero value indicates that an error occurred in checking for the event that prevents the event from being successfully detected. This might occur if there is a problem with event related hardware or that the required hardware is not supported by the device. The definition for Event Status error codes is currently left to the vendor discretion.
- Event Type: This enumerated value describes the type of event that occurred. Valid event types are as specified in Table 23.

Table 23 – Event Type

Event type	Event description
0	Registration 1 Positive Edge
1	Registration 1 Negative Edge
2	Registration 2 Positive Edge
3	Registration 2 Negative Edge
4	Marker Positive Edge
5	Marker Negative Edge
6	Home Switch Positive Edge
7	Home Switch Negative Edge
8	Home Switch-Marker ++
9	Home Switch-Marker +-
10	Home Switch-Marker -+
11	Home Switch-Marker --
12	Home Torque Threshold
13	Watch 1 Position Forward
14	Watch 1 Position Reverse
15	Watch 2 Position Forward
16	Watch 2 Position Reverse
17 to 127	(reserved)
128 to 256	(vendor specific)

- Event Position: 32-bit integer representation of the axis position when the designated event occurred. If the Event Status element has a non-zero error code, the Event Position is set to 0.
- Event Time Stamp: This element represents the 64-bit System Time value when the specified event occurred. The time stamp units are nanoseconds. Taken together with device's System Time Offset value that is part of the Device-to-Controller Connection Header, the controller has all the information it needs compute the absolute System Time when the event occurred. If the Event Status element has a non-zero error code, the Event Time is set to 0.

6.4.4.5 Service Data Block

The Service Data Block allows one service response per instance to be sent to the controller in a given update. Each service request requires a specific service response from the device indicating success or an error. In some cases, the response service contains requested data. In any case, the service response data persists in the Device-to-Controller Connection data structure until the device sees the associated service request removed from the Controller-to-Device Connection Instance Data Block (Service Block Size = 0) or a new service request is issued by the controller (incremented Transaction ID).

Each service response is represented by a block of data organised as shown in Figure 38.

NOTE Like the request structure, the structure of the service response does not follow the traditional CIP standard messaging format. That is primarily because this connection structure is, fundamentally, a CIP Implicit I/O connection, not an Explicit Messaging connection. However, the case of a Fixed Connection format, the Service Specific Request Data defined below is sent via an Explicit Messaging connection and follows the CIP rules for explicit service request format.

Service Data Block			
Transaction ID	Service Code	General Status	Extended Status
Service Specific Response Data			

IEC

Figure 38 – Service Data Block

- Transaction ID: Transaction number assigned to this service response derived from the Transaction ID of the original request. Each service request is assigned a new Transaction ID by incrementing the current Transaction ID stored in the controller. Using the Transaction ID in the response, the controller is able to match the service response to the appropriate service request.
- Service Code: Identifier that determines the specific service response that follows, which shall match the Service code of the originating service request. A list of valid Service Codes for the Motion Device Axis Object is given in the Controller-to-Device Connection subclause.
- General Status: The General Status value is provided to indicate success or failure of the requested service request. A General Status of 0 indicates success, while a non-zero value indicates an error. The General Status values comply with the CIP specification for General Status codes (see IEC 61158-6-2).
- Extended General Status: The Extended General Status provides a method for defining vendor specific or service specific error codes. There is currently no standard definition for these codes.
- Service Specific Response Data: The format and syntax of the Service Specific Response Data depends on the specified Service Code.

6.4.5 Fixed Motion I/O connection format

By specifying a Fixed Connection Format, the CIP Motion I/O Connection can be reduced to a size that is readily applicable to lower-performance CIP networks or devices. In keeping with this application context, the following features have been removed from the connection structure to support the requirements of a fixed connection size and limited network bandwidth.

– Time Stamping
– Node Faults/Alarms
– Multiple instance support
– Dynamic Block Sizing
– Cyclic Read/Write Data Block
– Event Data Block
– Service Data Block

With Fixed Connection Format, service requests to the Motion Device Axis Object are supported only as an Explicit Messaging service.

Figure 39 and Figure 40 show an example of the Fixed Connection Format being used in a simple variable speed drive application requiring only a velocity command and returning actual velocity. In this case, the connection size has been reduced to 16-bytes, a size that is well suited for lower performance networks like DeviceNet.

Controller-to-Device Connection format			
Connection Format	Format Revision	Update ID	Node Control
Control Mode	Feedback Mode	Axis Control	–
Command Data Set	Actual Data Set	Status Data Set	–
Command Velocity			

IEC

Figure 39 – Fixed Controller-to-Device Connection format (fixed size = 16 bytes)

Device-to-Controller **Connection format**			
Connection Format	Format Revision	Update ID	Node Status
Control Mode	Feedback Mode	Axis Response	Response Status
–	Actual Data Set	Status Data Set	Axis State
Actual Velocity			

IEC

Figure 40 – Fixed Device-to-Controller Connection format (fixed size = 16 bytes)

6.4.6 CIP Motion I/O Connection timing model

6.4.6.1 General

The general timing model for the CIP Motion I/O Connection data exchange is described in this subclause in the context of a CIP Motion Drive device, but also applies to other CIP Motion device types. Data exchange between the drive and the controller is paced by the Controller Update Period with one Device-to-Controller data packet sent for every Controller-to-Device data packet received. The Controller-to-Device Connection packets are sent periodically according to the configured Controller Update Period. The Device Update Period, i.e. the update period at which the drive performs its control calculations, is typically much faster than the Controller Update Period. The basic CIP Motion 1-Cycle timing model is illustrated in Figure 41.

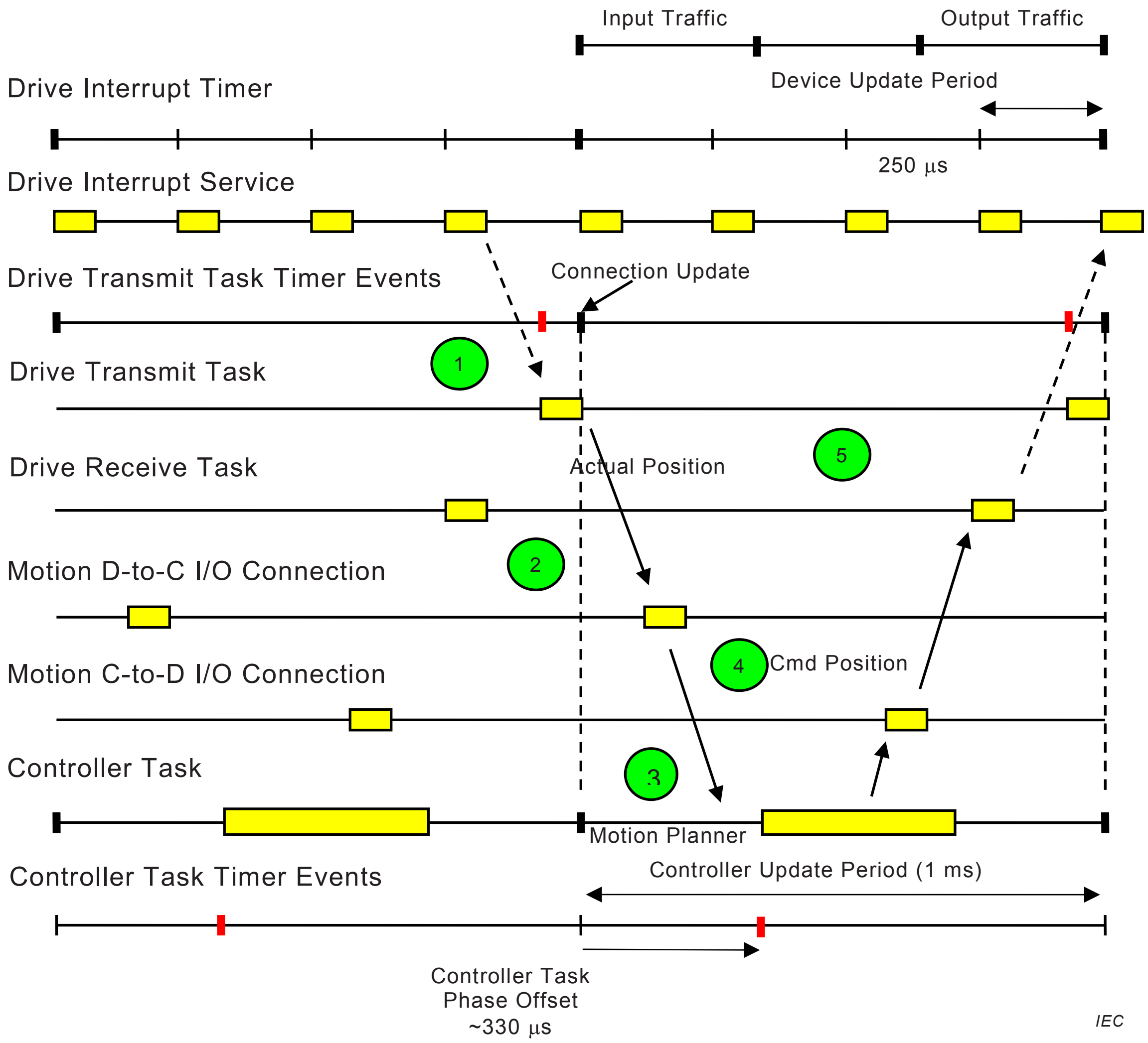

Figure 41 – CIP Motion 1-Cycle timing model

This is classified as a 1-Cycle Timing Model since a complete I/O data transaction occurs within one connection update cycle. The timing model consists of five basic steps which are shown in Figure 41 and described as follows.

1) A periodic Drive Transmit Timer Event (top red tick-mark shown above) scheduled just before the next Connection Update cycle starts the Drive Transmit Task that is responsible for managing the Device-to-Controller connection. The Drive Transmit Task gets the latest Actual Position value calculated by the last Drive Interrupt Service (assuming Position Control Mode in this example) and the associated Time Stamp.

2) Once the Actual Position and Time Stamp data is assembled, along with other axis data, into the Device-to-Controller Connection data structure, the drive transmits the packet to the controller. For best performance, it is recommended that the actual transmission be as close to the start of the Connection Update cycle as possible. The only timing constraint is that during normal operation, the drive device shall begin transmission no later than the start of the cycle. This allows time for the Device-to-Controller Connection packet to traverse the network and arrive at the controller prior to the start of the phase delayed Controller Task.

3) After a predetermined Phase Offset time from the start of last Controller Update cycle, which, in this model, is set to be 1/3 of the Controller Update Period, a Controller Task Timer Event triggers the Controller Task to start. The Controller Task begins by parsing the data from the drive, including the Actual Position value. The controller then runs a Motion Planner to compute a new Command Position value to send back to the drive. When gearing or camming operations are active, it may be necessary to use the Actual

Position of a master drive axis as input to compute the Command Position of one or more slave drive axes.

4) Once the Command Position is calculated, it is assembled, along with other axis data, into the Controller-to-Device Connection structure, the controller transmits the data to the drive. In general, it is recommended that the transmission occur no later than 2/3rds of the way though the Controller Update Period to allow time for the packets to reach the targeted drive before the end of the current cycle. The Controller-to-Device Connection packet traverses the network and arrives at the drive prior to the start of the next Connection Update cycle. There is no hard timing constraint here; the packet may arrive much earlier in the update cycle or possibly later than the start of the next update cycle. The drive device simply processes the packet upon arrival.
5) A Drive Receive Task executes in response to notification from the network interface that a new Controller-to-Device packet has arrived. The Drive Receive Task begins by parsing the data from the controller, including the Controller Time Stamp and Command Position value. The drive then applies the Command Position data using the time stamp to calculate new coefficients for the fine interpolation polynomial. These coefficients are used by subsequent Drive Interrupt Services to compute the fine command position value that is applied to the input of the drive’s position control loop. For best performance, it is recommended that the polynomial coefficients be applied to the fine interpolator as closely as possible to the start of the next update cycle. Applying the new coefficients early can create unnecessary error in the command trajectory when splicing the current fine interpolator segment to the new fine interpolator segment that begins with the next update cycle. Applying the new coefficients late forces the fine interpolator to run as a fin extrapolator into the next cycle, introducing unnecessary extrapolation error in the command trajectory when the extrapolated segment is spliced to the next fine interpolator segment. The Drive Receive Task concludes with the device scheduling the next Drive Transmit Timer Event relative to the start of the next Connection Update cycle, which is the sum of the Controller Time Stamp and the Controller Update Period.

The above example sequence represents an implementation that uses separate, interrupt-driven, Drive Transmit and Drive Receive Tasks. Other implementations may choose to combine one or both of these tasks with the Drive Interrupt Service that performs the control computations. In this case, processing the CIP Motion Connection data is done on a polled basis; the Drive Interrupt Service would need to check if a Controller-to-Device packet has been received and also check if it is time to assemble and send a Device-to-Controller packet.

While other timing models are possible based on the controller configuration, all CIP Motion timing models begin with the drive sending actual data to the controller at the beginning of the Controller Update cycle and end with command data arriving before a subsequent Controller Update cycle. However, it is not required that the command data sent to the drive in a given Controller Update cycle be the result of calculations performed during that cycle. These rules form the basis for multi-cycle timing models.

Figure 42 and Figure 43 illustrate these common CIP Motion timing model characteristics, by considering the 2-Cycle and 3-Cycle timing models.

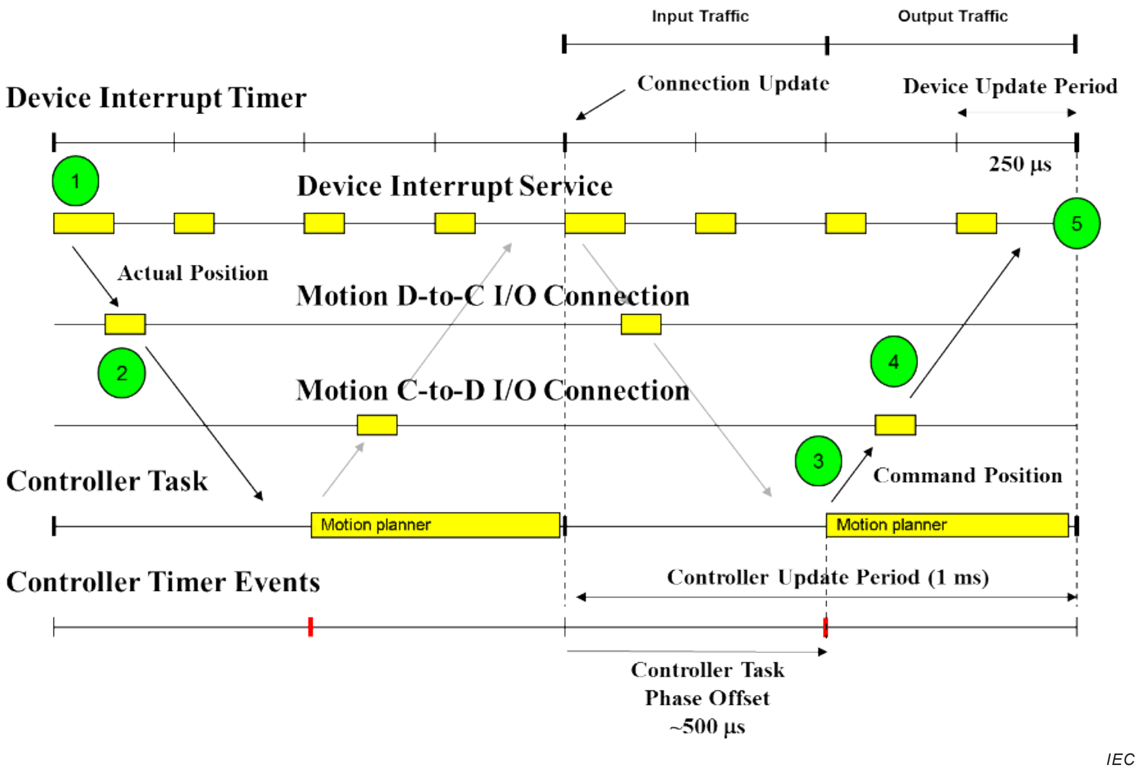

Figure 42 – CIP Motion 2-Cycle timing model

As with all CIP Motion timing models, the 2-Cycle Timing Model shown in Figure 42 begins with the drive transmitting the D-to-C connection packet to the controller at the beginning of the update cycle. But in this case the Controller Task does not start until half way through the update cycle, thus allowing more time for the D-to-C connection packet to reach the controller before the motion planner task runs. Unlike the 1-Cycle Timing Model, the C-to-D connection packet is not transmitted back to the drive until the next time the motion planner task runs. This again allows more time for the C-to-D connection packet to reach the drive. Thus, it takes 2 connection cycles to complete the I/O data transaction with the drive device.

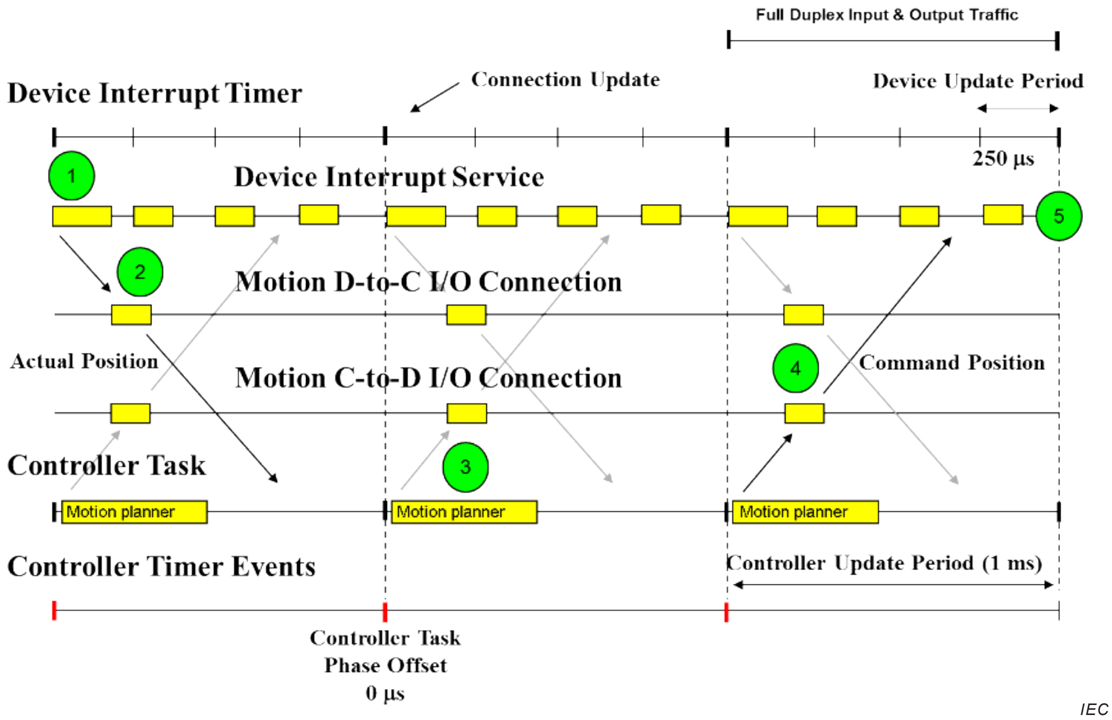

Figure 43 – CIP Motion 3-Cycle timing model

The above 3-Cycle Timing Model shown in Figure 43, once again, begins with the drive transmitting the D-to-C connection packet to the controller at the beginning of the update cycle. But in this case, the D-to-C connection packet has the entire update cycle to reach the controller before the motion planner task runs at the beginning of the next cycle. After the motion planner task runs, the resultant C-to-D connection packet is not immediately transmitted back to the drive, but rather is transmitted by the Motion Planner task at the beginning of the next cycle, again, giving the packet a full update cycle to reach the targeted drive. So, in this case, it takes 3 connection cycles to complete the I/O data transaction with the drive device. With the 3-Cycle Timing Model, CIP Motion input and output packet traffic is traversing the Full Duplex Ethernet media in both directions.

Based on these common timing model characteristics, the drive does not need to know the specific timing model the controller is applying. The drive simply transmits D-to-C packets to the controller at the beginning of the update cycle and processes C-to-D packets as they arrive from the controller.

6.4.6.2 Controller-to-Device connection timing

Most motion control protocols require the Controller Update Period to be an integer multiple of the Device Update Period, that is the drive's internal control loop update period. But because the CIP Motion I/O Connection packet includes a Time Stamp, the update period of the controller does not need to have any fixed relationship with the update period of the drive.

The following subclauses present a more detailed description of the CIP Motion timing model., presented in the context of a CIP Motion Drive device type with the understanding that the timing model applies generally to all CIP Motion device types.

One leg of the CIP Motion I/O Connection data exchange cycle is initiated by the controller via the Controller-to-Device Connection packet. The inclusion of Time Stamp and Time Offset information along with the command data in this packet relieves the stringent timing requirements imposed by other motion control network protocols. Specifically, time stamping allows the drive to compute command data values based on its own Device Updates that, unlike other motion control network protocols, does not need be in fixed relationship to the Controller Updates.

Figure 44 illustrates how command data and time stamps delivered by the Controller-to-Device connection are applied to the drive axis when fine interpolation is required. In this example, the Drive Transmit and Receive Tasks described in above Timing Model section have been combined with the Drive Interrupt Service into a singular, periodic Drive Task that runs at the Device Update Period.

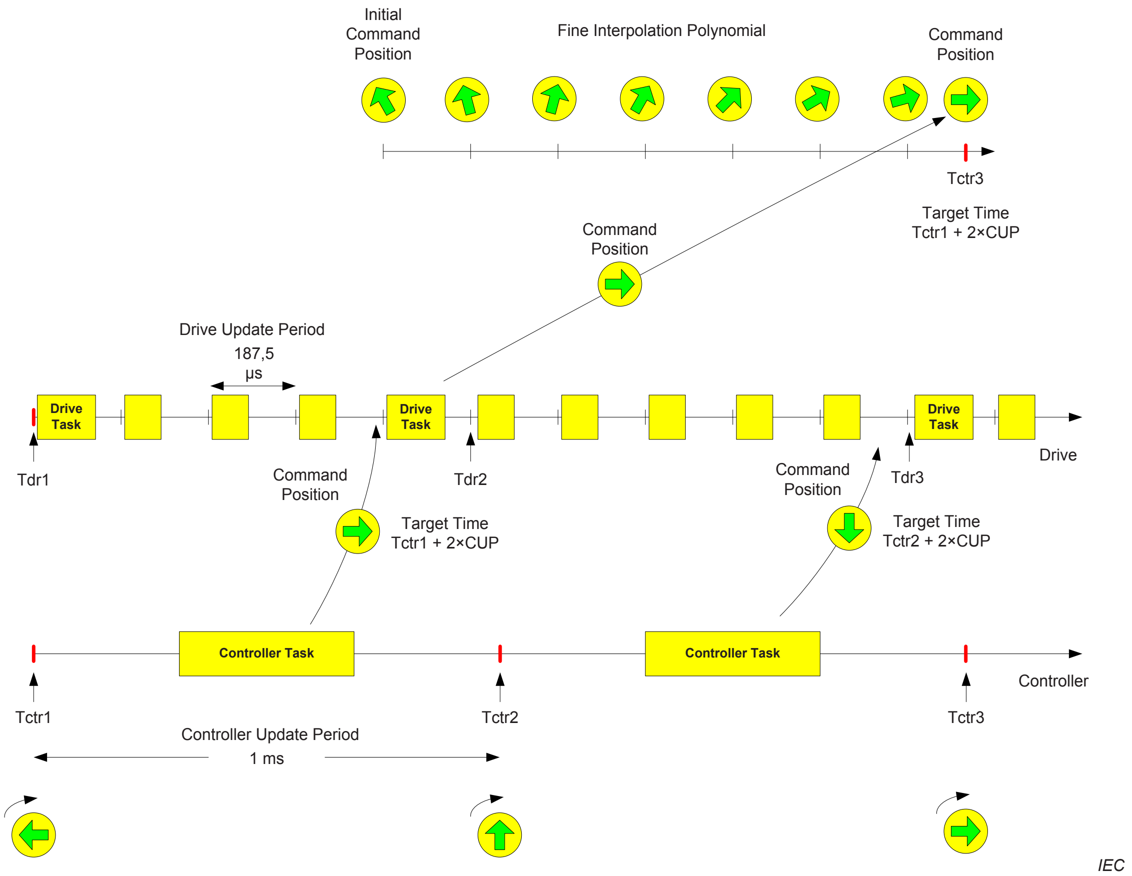

Figure 44 – Controller-to-Device Connection timing with fine interpolation

The following steps describe in detail how connection data is transferred from the controller to the drive for fine interpolation during a typical connection cycle in the general case where the Controller Update Period (CUP) is not an integer multiple of the Device Update Period.

1) Controller Transmit: As part of the Controller Task, the controller initiates transmission of a Controller-to-Device Connection packet with new command data to the targeted drive with an incremented Update ID and a new Controller System Time Stamp (and Controller System Time Offset if changed since last update) referencing the system time at the start of the current Controller Update cycle. The Instance Data Block for the targeted axis also contains the Command Target Update, which in this example is set to +2 to account for the one Controller Update Period (CUP) transport delay and the one Controller Update Period (CUP) delay for fine interpolation. It is recommended that the controller transmit the Controller-to-Device Connection packet no later than 2/3rds of the way though the Controller Update Period to allow time for the packets to reach the targeted drive. The Controller-to-Device Connection packet is received by the drive and processed by the network stack upon arrival. The network stack presents the new Controller-to-Device Connection data to the application layer's Drive Task for parsing.
2) Update ID Check: In response to new data notification by the stack, the Drive Task begins parsing the Controller-to-Device Connection packet data by first checking the Update ID. If

the Drive Task discovers that the Update ID has incremented by more than 1 since the last update, the drive increments the Lost Controller Updates value.

3) Synchronous Operation Check: Next, the Drive Task begins parsing the Controller-to-Device Connection packet data by first checking if the drive is synchronized. If not synchronized, skip to the Apply Command Data step since there is no need to perform the System Time Offset Check or the Late Update Check. Bypassing these checks allows for control of the drive during start-up or even in the case where the drive does not have any time synchronization services. This is referred to as Asynchronous Operation.

4) System Time Offset Check: Synchronous operation requires that System Time between the drive and the controller match to make the time stamps exchanged between the devices meaningful. Since System Time is adjusted periodically by the time master and the adjustments can propagate to the different devices in the network at different times, it is possible for System Time between the controller and the drive to be temporarily skewed. This skew shall be detected and, if present, the drive shall adjust the Controller Time Stamp to compensate for the skew. The drive does this offset compensation only if the Controller or Device System Time Offsets have changed since the last update. See 6.4.6.7 (System Time Offset Compensation) for details of this algorithm.

5) Late Update Check: Assuming synchronous operation, the drive then computes the difference between the current Drive Task time stamp and the Controller Time Stamp passed in the Controller-to-Device Connection packet (adjusted, if necessary). A difference of more than one Controller Update Period increments the Late Controller Updates value. If the difference is greater than (Controller Update Delay Low Limit × Controller Update Period), the drive throws a Controller Connection Update Alarm. If the difference is greater than (Controller Update Delay High Limit × Controller Update Period), the drive throws a Control Connection Update Fault.

NOTE 1 If the time difference has exceeded the Controller Update Period, the current fine interpolator polynomial has become, effectively, an extrapolator polynomial allowing the drive to ride through the late data condition until the new data arrives.

6) Apply Command Data: Assuming synchronous operation and a Command Target Update of +2 for fine interpolation, the drive computes coefficients for the fine interpolation polynomial based on the command reference being applied at the computed Target Time, which is the sum of the (adjusted) Controller Time Stamp (CUP), Tctr1, and the product of the Command Target Update and Controller Update Period, or in this case, 2 × CUP. These coefficients are used by the drive’s control service routine to compute the fine command position based on the time of the service. Ordinarily these new coefficients are applied for the duration of the next update cycle, providing fine interpolation until System Time has reached Target Time. If the Target Time is less than the current System Time when the Controller-to-Device Connection packet was received, i.e. this is a late packet, then new coefficients to the polynomial are still computed based on this command data to improve the accuracy of the extrapolation calculations. In general, whenever command data is late, the data still represents the freshest command data available and the coefficients in this case shall be applied as soon as possible. If asynchronous operation, the command data is applied to the drive’s control service immediately.

7) Schedule Next Device-to-Controller Connection Update: The drive calculates System Time for the next Connection Update Cycle as the sum of the Controller Time Stamp and the Controller Update Period. This value is used to establish the Actual Update Window criteria for the next Device-to-Controller Connection Update or to directly schedule the Device-to-Controller Connection Update via a timer event.

If the Command Target Update is set to +1, the computed polynomial in step 6 is not applied for the purpose of fine interpolation but rather for extrapolation; the extrapolation polynomial allows the drive to compute an accurate command data value at the time the drive performs its control calculations based on previous axis trajectory.

Figure 45 illustrates this timing model in the general case where the Controller Update Period (CUP) is not an integer multiple of the Device Update Period and the Command Target Update is set to +1 for Extrapolation.

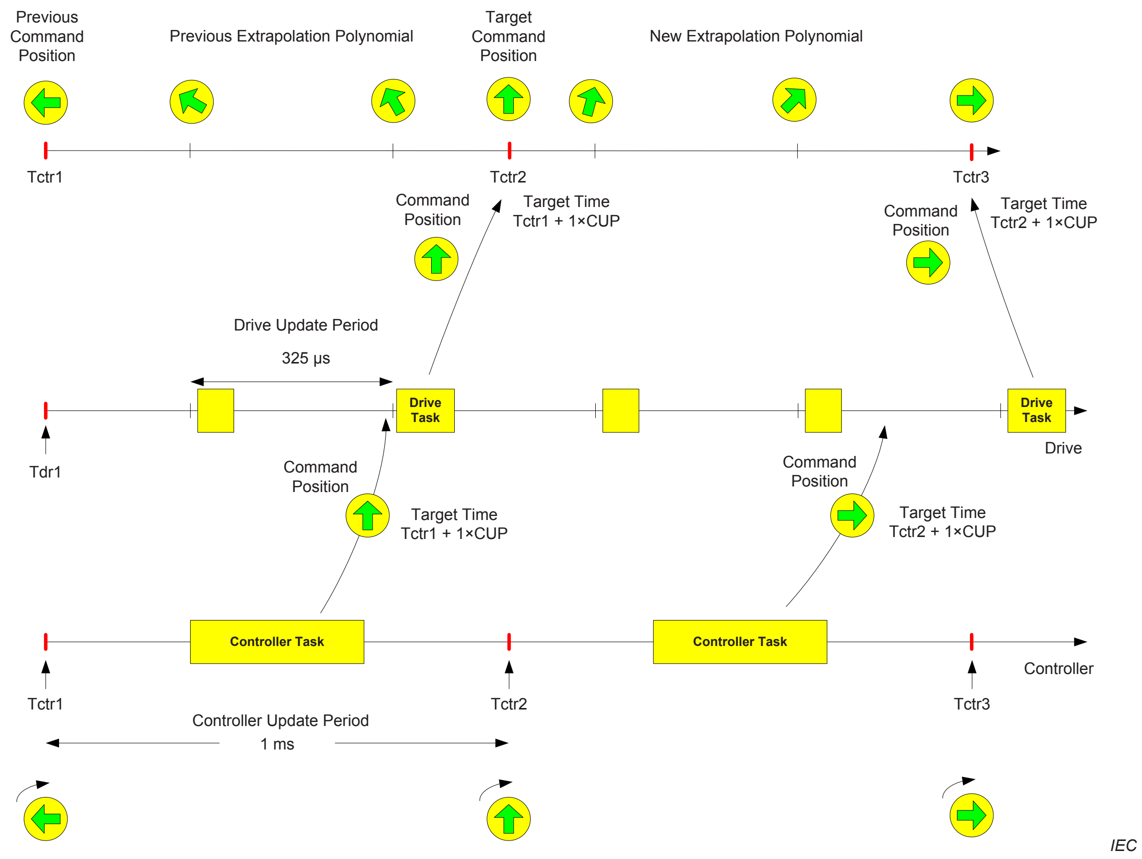

Figure 45 – Controller-to-Device Connection timing with extrapolation

NOTE 2 In the above example, there are not many Device Update Periods in a given Controller Update Period. When this is the case, fine interpolation is not critical to drive performance and command data can be applied more directly to the drive's control structure using extrapolation without the extra delay required to support fine interpolation. Extrapolation has the disadvantage however that extrapolation error is manifested more directly to the command data resulting in rougher motion than when using fine interpolation.

All cyclic data associated with the Controller-to-Device Connection packet shall be applied in the Drive Task command update to make the earliest possible use of fresh command data, computing new interpolation/extrapolation polynomial coefficients.

6.4.6.3 Device-to-Controller connection timing

The other leg of the CIP Motion I/O Connection data exchange cycle is initiated by the drive via the Device-to-Controller Connection packet. When in synchronous mode, the CIP Motion Device-to-Controller Connection includes a Device Time Stamp (and Device Time Offset) with the actual data to allow the controller to calculate the position of the drive axis at the time the Controller Task update occurs. Time stamping allows the drive to sample feedback and compute actual data values based on its own Device Updates that, unlike other motion control network protocols, does not need be strictly related to the Controller Updates. Figure 46 illustrates how actual data and time stamps delivered by the Device-to-Controller Connection are used to adjust drive axis actual position, for example, to the controller's timebase.

Again, for simplicity, the Drive Transmit and Receive Tasks described in the above Timing Model section have been combined with the Drive Interrupt Service into a singular, periodic Drive Task that runs at the Device Update Period.

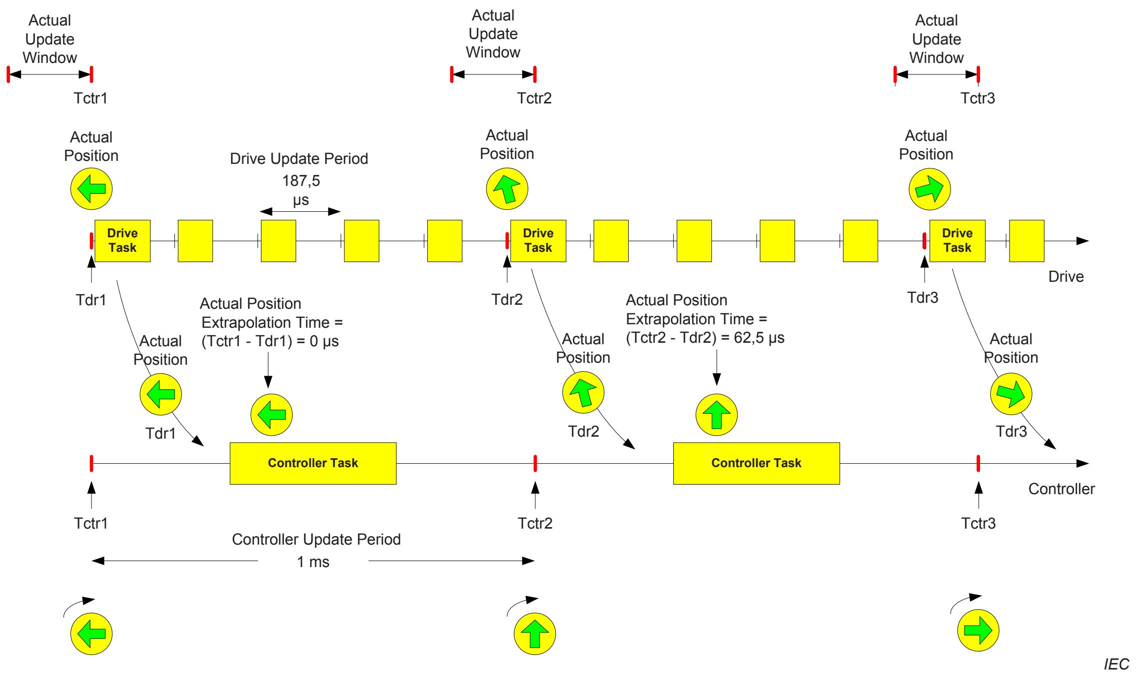

Figure 46 – Use of Time Stamp to adjust actual position to the controller's timebase

The following steps describe in detail how connection data is transferred from the drive to the controller during a typical connection cycle in the general case where the Controller Update Period (CUP) is not an integer multiple of the Device Update Period. Of course, this sequence also applies to the special case where the Controller Update Period (CUP) is an integer multiple of the Device Update Period.

1) Actual Update Check: If the axis is synchronized, the drive compares current Drive Task time stamp with the Actual Update Window established in the last update cycle. The Actual Update Window has duration of 1 Device Update Period and ends at the latest possible time for the drive to completely assemble the Device-to-Controller Connection packet and have it ready for transmission by the start of the next update cycle. If the Drive Task time stamp is within the time window, this is an actual data update cycle. If the time stamp is before the window, then it is not time to send data to the controller. (The purpose of the Actual Update Window is to schedule the actual data transmission as close to the start of the Connection Update cycle as possible, while also minimizing time between the feedback capture and the start of the Connection Update cycle.) For timer event interrupt-driven Transmit Tasks, there is no need for the Actual Update check since the Transmit Task timer event service is already scheduled to execute at the proper time. If the axis is not synchronized and we have just received a command update via the Controller-to-Device Connection, or the time since the last command update has exceeded 1 Controller Update Period, then this is also an actual update cycle, so move on to the Drive Transmit step.
2) Drive Transmit: If this is an actual update cycle, then the Drive Task assembles and transmits the Device-to-Controller Connection packet to the controller with the latest Actual Data, incremented Update ID and, for synchronous devices, the associated Device Time Stamp (and Device Time Offset if changed since the last update). For best performance, it is recommended that the packet transmission be as close to the start of the Connection Update cycle as possible. The only timing constraint is that during normal operation, the drive device shall begin transmission no later than the start of the actual update cycle. This is the most important timing constraint associated with the CIP Motion timing model. As such, Device-to-Controller Connection packet processing shall have priority over Controller-to-Device Connection packet processing. The extra delay this may introduce to Controller-to-Device Connection packet processing does not have a significant effect on motion control quality as long as the Device Update Period is much shorter than the Controller Update Period.

3) Update ID Check: In the next Controller Task, the controller checks for new data from the drive by checking for a changed Update ID. If the Update ID has not changed since the last update, the controller increments the Lost Updates value associated with the drive's Device-to-Controller connection. The following steps are performed regardless of whether on not the Update ID has changed.
4) Sync Mode Check: Drive checks the Sync Mode bit of the drive's Node Status byte to determine if the drive axis is synchronized. If not synchronized, skip to the Apply Actual Data step to avoid Late Update checking and Time-Stamp Correction. Bypassing these subsequent steps allows the drive to operate during start-up or even in the case where the drive does not have any time synchronization services.
5) System Time Offset Check: Synchronous operation requires that System Time between the drive and the controller to match to make the time stamps exchanged between the devices meaningful. Since System Time can be adjusted periodically by the time master and the adjustments can propagate to the different devices in the network at different times, it is possible for System Time between the controller and the drive to be skewed. This skew shall be detected, and if present, the drive shall adjust the Device Time Stamp to compensate for the skew. The controller does this offset compensation only if the Controller or Device System Time Offsets have changed since the last update. See 6.4.6.7 (System Time Offset Compensation) for details of this algorithm.
6) Late Update Check: The controller computes the difference between the current Controller Task time stamp and the drive's Device Time Stamp in the Device-to-Controller Connection packet. If the difference is greater than (Controller Update Delay Low Limit × Controller Update Period), the controller throws a Control Sync Alarm. If the difference is greater than (Controller Update Delay High Limit × Controller Update Period), the controller throws a Control Sync Alarm.
7) Time-Stamp Correction: If the previously computed time stamp difference is non-zero, then extrapolate the actual data value based on previous axis actual trajectory to line up with the controller's time stamp. This correction is necessary because the motion planner assumes that actual input data is implicitly time stamped to the beginning of the Controller Update Period. Under normal operation, the computed time stamp difference is small relative to the Controller Update Period and the resultant position correction is small, but in cases where the Device-to-Controller Connection packet is late or lost, this position correction is significant and critical for continued operation as it allows the control system to "ride through" this condition based on previous axis trajectory.
8) Apply Actual Data: Controller applies actual data as inputs to the motion planner, which computes new command reference data for the next Controller-to-Device update.

6.4.6.4 Scheduling the next update

To insure that a Device-to-Controller Connection update occurs regardless of having received a Controller-to-Device Connection packet, during every cycle the drive shall schedule the time for next cycle's Device-to-Controller Connection update.

For synchronous operation, the drive computes the time for the next cycle's Device-to-Controller Connection update to be enough before the end of the next update cycle to allow time for Device-to-Controller Connection data to be assembled and ready to transmit before the end of the cycle. For timer event interrupt-driven Transmit Tasks, this rescheduling occurs automatically when the local periodic timer reloads. For implementations where the Device-to-Controller Connection is handled by the Drive Task, the rescheduling shall be done explicitly by the drive at the conclusion of every Device-to-Controller Connection update.

For asynchronous operation, the drive initiates a task to assemble and transmit the Device-to-Controller Connection packet immediately after receiving (and parsing) a Controller-to-Device packet. The drive then schedules the transmission of the next Device-to-Controller packet to be 1 Controller Update Period from the time this Controller-to-Device packet was received. Thus, the transmission of that Device-to-Controller packet will take place even if the next Controller-to-Device packet is lost or late. The Update ID for this Device-to-Controller packet shall be the Controller-to-Device packet's Update ID + 1 so that the Update IDs for the

Device-to-Controller and Controller-to-Device packets match for a given update cycle thereafter.

6.4.6.5 Device Update Period independence

The timing diagram in Figure 47 illustrates how two drive axes can be tightly coordinated despite having different Device Update Periods and despite having an associated Controller Update Period that is not an integer multiple of either Device Update Period.

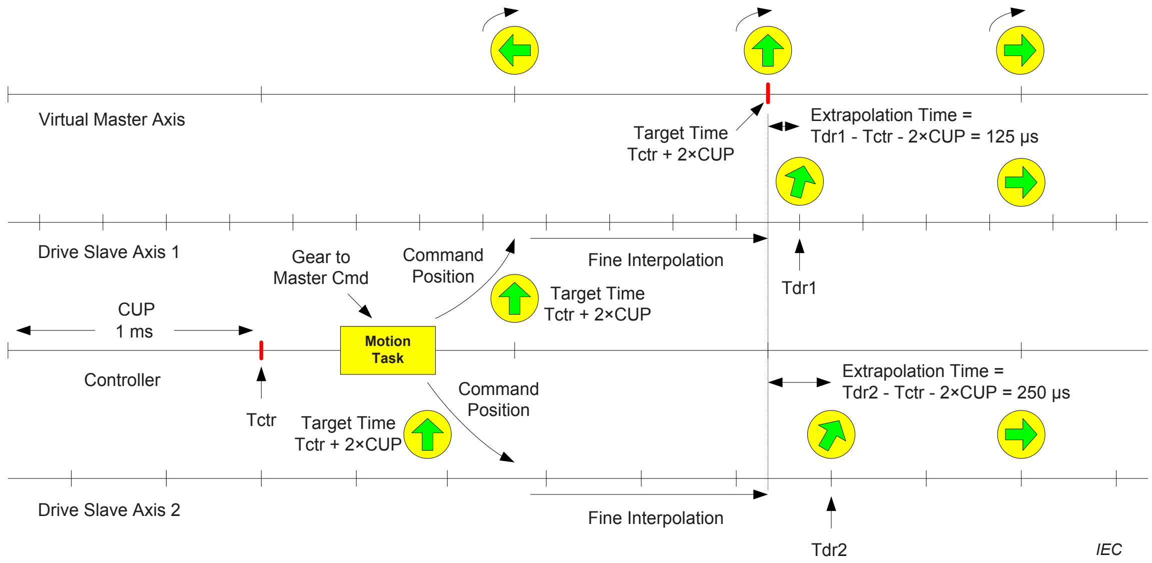

Figure 47 – Coordination of two drives with different Update Periods

In the above timing diagram, the controller’s motion planner task sends identical command positions and time stamps to two slave drive axes that, while synchronized with System Time, are running at different drive update rates. When the command position data arrives at the two drives, they use the Controller Time Stamp, the Command Target Update, and the Controller Update Period to compute new coefficients to the interpolation polynomial based on the constraint that the polynomial value at time equal to (Controller Time Stamp + Command Target Update × Controller Update Period) is the specified Command Position value. Since there is no dependency on the drive update rate, the polynomial coefficients computed by each drive are identical. Since neither drive has an update that coincides with this target time, the drives use the fine interpolation polynomial to calculate the command position reference for each drive update until a fresh command position is received from the controller. If a new command position does not arrive until well after the target time, the drive continues to use the same polynomial equation to “extrapolate” command position for subsequent drive updates as shown in Figure 47. This extrapolation continues until fresh data arrives and new coefficients can be calculated. In this way, whether by interpolation or extrapolation, each slave axis runs smoothly and the two axes stay phase locked with the master axis.

6.4.6.6 Transmission latency independence

Precise coordination of multiple CIP Motion drive axes can be maintained even when the Controller-to-Device Connection packets incur significant delays while travelling across the CIP network. In Figure 48, the packet for Slave Drive Axis 2 has incurred a significant delay during transmission. As a result, the command position for this axis shall be extrapolated from the last fine interpolation polynomial. This allows the axis to move smoothly through a transmission latency disturbance. When the new command data does arrive, the new command value may not agree with extrapolated value due to extrapolation error. This error can result in a disturbance to the motion profile. The magnitude of the extrapolation error depends on the dynamics of the motion profile and the controller update rate. In most real-world applications, transmission latencies lasting several update periods can occur without any noticeable disturbance to the associated motion profile.

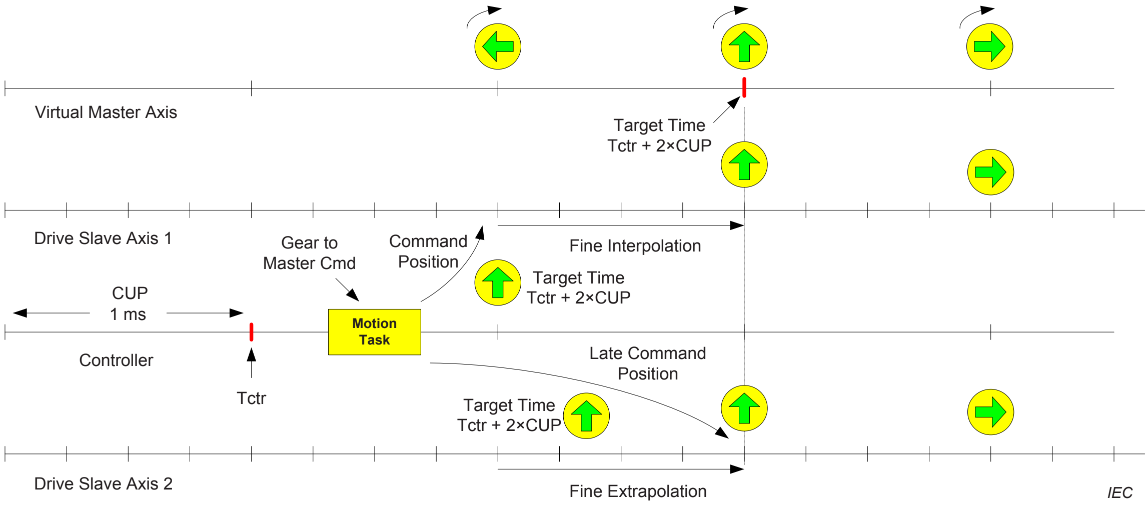

Figure 48 – Coordination of multiple drive axes in case of delayed Controller-to-Device Connection packets

6.4.6.7 System Time Offset compensation

6.4.6.7.1 General

CIP Motion is built on the foundation of CIP Sync and its underlying IEC 61588:2009 time synchronization protocol, which defines a mechanism to distribute and synchronize time for devices on a network.

Using the IEC 61588:2009-based CIP Sync protocol, each device on the network has a representation of System Time that is tightly synchronized with the Grand Master Clock of the network. System Time for each device is defined as the sum of the device's local clock time and the device's System Time Offset. With CIP Sync, it is important to note that while the local clock runs at the same frequency as the Grand Master Clock, it is not strictly phase locked to the Grand Master Clock. Only the device's System Time value is phase locked to the Grand Master Clock and is suitable for synchronizing data exchange between the controller and the drive device.

According to the CIP Sync protocol, System Time can incur large step changes, from a few milliseconds to many years, and these step changes are propagated through the network to each device. These step changes in time may be caused, for example, by the user manually adjusting the master's absolute time to correct for clock drift, or by a change in time mastership. IEC 61588:2009 does not address step changes in time as part of its protocol and, therefore, has no provision for handling this condition during real time control operation. The CIP Sync standard defines a concept called Time Step Compensation that, when implemented in device's Motion Device Axis Object, effectively handles this condition. The following subclause describes this algorithm in detail.

6.4.6.7.2 Time step compensation

According to CIP Sync, step changes to the master clock time shall propagate through the distributed time system; the time to propagate this step change in time through the system from the time master to the furthest time slave is a function of the sync packet interval (default is 2 s) and the number of hops (1 hop per IEC 61588:2009 boundary clock, for example a switch implementing the IEC 61588:2009 protocol). This is illustrated in Figure 49.

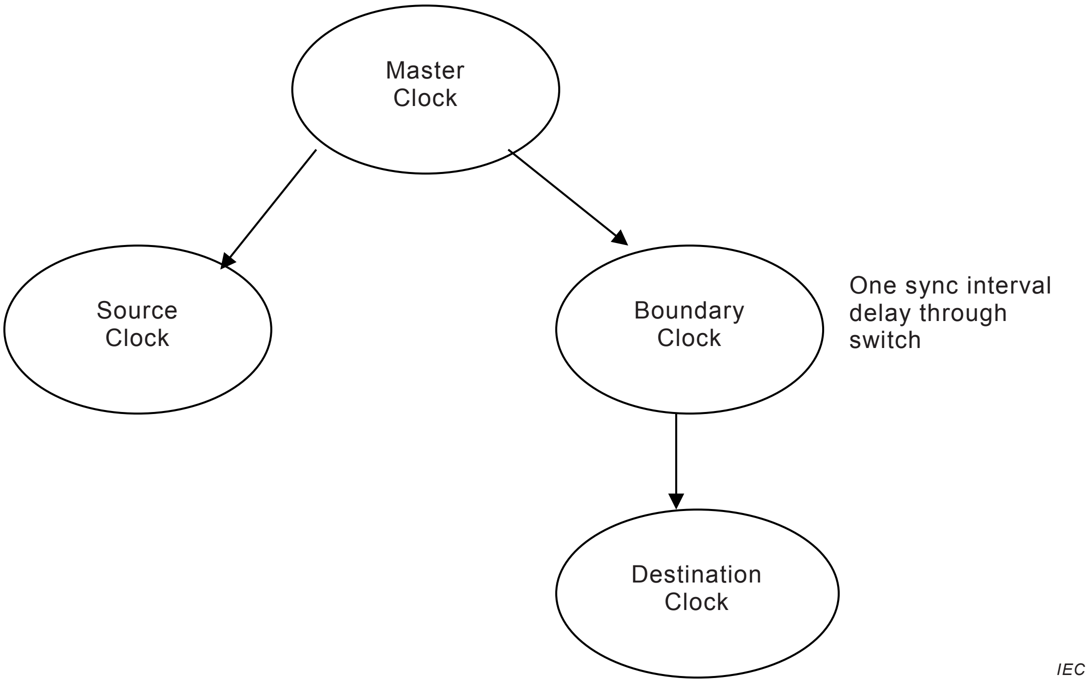

Figure 49 – Propagation of a step change in time

In Figure 49, the Source Clock is going to receive a System Time adjustment one sync interval before the Destination Clock. Thus, System Time for the Source Clock is going to be skewed relative to System Time for the Destination Clock for an entire sync packet interval.

The variability in propagating this time adjustment though the system creates a problem for time stamped based motion control in that the time stamp in the source device, say the controller, may not be applicable when received by the destination device, say a drive, because System Time for the source and destination devices is skewed. This effect is further illustrated in Table 24.

Table 24 – Propagation of a step change in time (example 1)

Time Sync Interval	Master time	Source time (controller)	Destination time (drive)
1	100	100	100
2	200 + 1 000	200	200
3	1 300	300 + 1 000	300
4	1 400	1 400	400 + 1 000
5	1 500	1 500	1 500

At Time Sync Intervals 1 and 2, both the timestamp source clock in the controller, and timestamp destination clock in the drive both have the same notion of time, i.e. System Time. At Sync Interval 2, a step change in time occurs at the master clock that propagates to the controller in Sync Interval 3, thus affecting the source clock time stamp sent to the drive device. It is not until Sync Interval 4 that the drive sees the step change. Any time stamp data sent from the controller to the drive between Sync Intervals 3 and 4 will not correlate with the drive’s notion of time.

CIP Motion and CIP Sync extend the IEC 61588:2009 protocol to handle this condition. This is done by defining a run-time algorithm in the destination drive device that can detect this time step condition and adjust the value of the received timestamp so that it is accurate. Two conditions are possible:

1) the source device has seen a step change in time but the destination device has not;
2) the destination device has seen a step change in time but the source device has not.

The key component of the algorithm is the System Time Offset value defined as part of the CIP Sync's Time Sync Object. This offset value is added to the local clock time to generate System Time, that is,

$$\text{System Time} = \text{Local Clock} + \text{System Time Offset(drive)}$$

When the system is synchronized, at a given moment of time, the Local Clock values and the System Time Offset values may differ from device to device, but the System Time value shall be the same within the accuracy of the IEC 61588:2009 protocol implementation.

NOTE The System Time Offset value associated with the drive device is provided by the Time Sync Object and comes pre-filtered to attenuate the influence of network traffic induced noise.

A step change in time is indicated by a change in the System Time Offset value of either the source or destination devices. The source's System Time Offset is sent to the destination device as an offset along with the source's System Time as the data time stamp. The destination device compares the offset received to the previously received offset to determine if a step change has occurred and adjusts the received timestamp value accordingly.

The Offset Compensation algorithm, as it applies to the received Timestamp from the controller via the C-to-D connection, is stated as follows:

$$\text{Timestamp}_{comp} = \text{Timestamp}_{rec} + \text{Offset Compensation}$$

$$\text{Offset Compensation} = \text{Offset}_{dest} - \text{Offset}_{dest(last)}) - (\text{Offset}_{src} - \text{Offset}_{src(last)}))$$

where,

Timestamp_{rec}	= received time stamp from controller.
Timestamp_{comp}	= compensated time stamp from controller.
Offset_{dest}	= the current value of the System Time Offset at the destination.
$\text{Offset}_{dest(last)}$	= the previous value of the System Time Offset at the destination.
Offset_{src}	= the received value of the System Time Offset from the source.
$\text{Offset}_{src(last)}$	= the previous value of the System Time Offset from the source.

Whenever the drive determines that either the source or destination System Time Offsets have changed, the following calculation is performed to check if there has been a system wide step change in time.

If |Offset Compensation| <= Step Threshold

$\text{Offset}_{dest(last)} = \text{Offset}_{dest}$

$\text{Offset}_{src(last)} = \text{Offset}_{src}$

where Step Threshold is a number that defines the smallest time step that can occur in the system. For CIP Motion systems this number is typically set to the RPI of the CIP Motion Connection via the optional Motion Device Axis Object class attribute, Step Threshold. If the Step Threshold attribute is not supported in the device, the default Step Threshold value of 1 ms is used.

If Offset Compensation is greater than the Step Threshold, indicating that a system wide step change in time has occurred, the Offset Compensation algorithm will hold (rather than reinitialize) the last "pre-step change" Time Offset values for the drive and the controller until the matching Time Offset change is detected and drops the Offset Compensation below the Step Threshold. Once the step time change is complete, the Offset Compensation value is once again small and the controller System Time closely matches the drive System Time.

Depending on the network location of the Grand Master clock, the drive device might see the Time Offset change before the controller or visa versa. The algorithm handles both cases.

Table 25 provides an example of how System Time Offset Compensation works. For illustration purposes, a packet is assumed to be sent at the sync interval with no propagation delay.

Table 25 – Propagation of a step change in time (example 2)

Time Sync Interval	Master time	Timestamp source (controller)					Timestamp destination (drive)					
		Local clock	Offset	Last offset	Local time	Time stamp	Local clock	Offset	Last offset	Local time	Offset comp	Adj.time stamp
1	100	0	100	100	100	100	0	100	100	100	0	100
2	200 + 1 000	100	100	100	200	200	100	100	100	200	0	200
3	1 300	200	100 + 1 000	100	1 300	1 300	200	100	100	300	-1 000	300
4	1 400	300	1 100	100	1 400	1 400	300	100 + 1 000	100	1 400	0	1 400
5	1 500	400	1 100	1 100	1 500	1 500	400	1 100	1 100	1 500	0	1 500

At Sync Interval 2 a step change of 1 000 occurs in the master clock and is propagated to the timestamp source clock, say the controller, in Sync Interval 3. The time stamp received at the destination clock, the drive in this example, is compensated for by the Offset Compensation value of 1 000 according to the above algorithm, resulting in an Adjusted Controller Time Stamp value that is consistent with the drive's Local Time even though it has not yet seen the step change. In Sync Interval 4, the drive sees the step change and the computed Offset Compensation value is once again 0 because in the Offset Compensation algorithm the source and destination delta Offset terms cancel. Since the Offset Compensation value is now less than the Step Threshold, the Last Offset values are initialized to the current Offset values as shown in sync interval 5 and no further Offset Compensation is required.

6.4.6.7.3 Frequency step compensation

While the above algorithm works well for cases where a significant step change in System Time is propagated through the system, it does not handle the case where there is a step change to the master clock frequency as can occur when introducing a new Grand Master clock. When this occurs, a significant change to the System Time Offset occurs whenever the device receives an IEC 61588:2009 Sync message for as many Time Sync cycles as it takes to fully synchronize the device's clock to the new Grand Master clock. These Time Offsets may or may not occur for the controller, especially if the controller is assuming the role of the new Grand Master, in which case the controller Time Offset value never changes! If the controller never reports a matching Time Offset change then the Time Offset changes accumulate in the device and introduce a phase shift between the device and the controller that can be quite large, i.e. greater than 1 ms. This is generally unacceptable when running a high performance motion control application.

To mitigate this Step Frequency Change case the above Offset Compensation algorithm shall be modified to compensate for the associated phase shift. The maximum frequency change that can occur during a Grand Mastership change is limited to be 0,02 % by the IEC 61588:2009 specified crystal tolerance. This limits the magnitude of the Time Offset changes to the product of the maximum frequency differential, the Time Sync Interval, and the number of boundary clock hops between the Grand Master and the device. A practical limit for the Time Offset changes due to change in Grand Mastership is placed at around 1 ms.

The amount of phase shift that can be introduced between Sync message updates due to the frequency step can be substantial, e.g. around 100 µs. Attempting to adjust the Offset

Compensation value by this amount immediately would cause a significant motion disturbance if the device was operational at the time. So, to avoid disturbing motion, the Time Offset Compensation algorithm shall be modified to gradually reduce the Offset Compensation value to zero to eliminate the phase shift. This gradual correction is done at a rate of 1 μs of phase correction per millisecond of running time, insuring that a 1 ms phase correction can be completed in a 1 s Time Sync Interval.

If the device is in the middle of performing such a phase correction and a new Time Offset is sent to the device from the controller, it is possible that the phase correction process is going to stop prematurely, resulting in a persistent phase shift between the controller and the device System Time clocks. Since this is unacceptable, the modified algorithm insures that any uncorrected phase shift is carried into the next Time Sync Interval.

The modified Time Offset Compensation algorithm is as follows:

$Timestamp_{comp}$ = $Timestamp_{rec}$ + Offset Compensation.

where,

Offset Compensation = Variance – Phase Correction.

The values for the new terms Variance, and Phase Correction are calculated based on whether or not the Time Offset values have changed. Here are the two cases.

If $Offset_{dest}$ or $Offset_{src}$ have changed from their last value:

Variance = $((Offset_{dest} - Offset_{dest(last)}) - (Offset_{src} - Offset_{src(last)}))$

Variance = Variance + Phase Remaining
Phase Correction = 0
Phase Remaining = 0

If |Variance| <= Step Threshold
  Case = Normal Sync
  $Offset_{dest(last)}$ = $Offset_{dest}$
  $Offset_{src(last)}$ = $Offset_{src}$

Else |Variance| > Step Threshold
  Case = Time Step Change

Else if $Offset_{dest}$ and $Offset_{src}$ have not changed:

Case = Normal Sync
  If Variance = Phase Correction
    No Operation
  Else If Variance > Phase Correction
    Phase Correction = Phase Correction + RPI/1000
    If Phase Correction > Variance
      Phase Correction = Variance
    Phase Remaining = Variance-Phase Correction
  Else If Variance < Phase Correction

Phase Correction = Phase Correction – RPI/1000

Phase Correction < Variance

Phase Correction = Variance

Phase Remaining = Variance – Phase Correction

Case = Time Step Change

No Operation

This modified version Time Offset Compensation algorithm is very robust, covering step changes in System Time, and step changes in Grand Master clock frequency. The algorithm will eliminate any persistent phase shifts between the controller and the device under these circumstances without disturbing motion during real-time operation.

6.5 Device startup procedure

6.5.1 General

In this subclause, the start-up process of a CIP Motion device is discussed. For the sake of brevity, the exact messages and their formats will not be presented. The objective is to outline the steps and the exchanges that take place between the controller and the device in order to correctly configure the CIP Motion device.

The device startup involves three distinct processes which are initiated by the CIP Motion controller and responded to by each CIP Motion device on the network. They are:

1) Motion I/O Connection creation;
2) Motion Object configuration;
3) time synchronization.

These processes may occur in parallel or consecutively depending on the particular implementation. Device Configuration may occur via the CIP Motion I/O Connection or separate connected explicit messaging. The startup process is outlined in the following subclauses.

6.5.2 Motion I/O Connection creation

Motion I/O Connections are created on EtherNet/IP using either the common CIP Forward_Open or Large_Forward_Open services (see IEC 61158-5-2 and IEC 61158-6-2). These services specify the maximum size of the Control-to-Device and Device-to-Controller Connection data structures and explicitly target an Input and Output Connection Point for the Motion Device Axis Object class.

In order to open this connection to the Motion Device Axis Object class, the originator constructs a Forward_Open service with the Electronic Key segment and two Application Paths that represent the Configuration data block and the set of Consuming/Producing data blocks to be used with this connection. See the Connection Manager Object in IEC 61158-5-2 and IEC 61158-6-2 for full details of the Forward_Open service.

The Configuration path is encoded as Connection Point 81 (0x51) of the Motion Device Axis Object (0x42). The I/O Connection data path is encoded as Connection Point 2 of the Motion Device Axis Object, and shall always match the Format Revision of the C-to-D and D-to-C Connection Data Structures. Next, the Data Segment (0x80) contains 4 words (8 bytes) of configuration data as defined by the Data Segment Configuration Block below. Thus, for this release of the specification the Application Path and Data Segment of the Forward_Open service should appear as follows:

20 42 2C 51 2C 02 80 04 …followed by the 4 word Configuration Block (Rev 1)

In addition the Application Path, the following information is also included in the Forward_Open service:

T-to-O Connection Size: 220 bytes (example only)

T-to-O Connection Size Type: Variable

T-to-O Connection Type: Point-Point

O-to-T Connection Size: 220 bytes (example only)

O-to-T Connection Size Type: Variable

O-to-T Connection Type: Point-Point

The O-to-T Connection info corresponds to the Controller-to-Device Connection while the T-to-O Connection info corresponds to the Device-to-Controller Connection. In the above example, the T-to-O and O-to-T Connection Size may vary depending on the controller implementation.

As stated above, in addition to Connection Points and device Keying information, the Forward_Open service also contains a Configuration Block in the Data Segment of the service request. The Configuration Block is used to verify that the power structure associated with the device matches that of the device selected by the controller.

The Configuration Block for Format Revision 1 is an 8-byte data structure defined as shown in Figure 50.

← 32-bit Word (octet 1 | octet 2 | octet 3 | octet 4) →			
Data Segment Configuration Block			
Configuration Rev	Configuration Bits	(reserved)	
Drive Power Structure Class ID			

IEC

Figure 50 – Configuration Block Format Revision 1 (Connection Point 81)

The Format Revision element represents the revision of the Configuration Block data structure. If the format of the Connection Block changes in future revisions of this specification, the Format Revision number shall be incremented as shall the associated Connection Point. The Format Revision for the Configuration Block as defined above is 1, which corresponds to Connection Point 81 (0x51).

The Configuration Bits field for a CIP Motion device has only one bit currently defined, the Verify Power Rating bit as bit 0. If this bit is set, the device shall verify that the Drive Power Structure Class ID of the Forward_Open matches the Drive Power Structure Class ID attribute value of the drive device. If the values do not match, the Forward_Open service response shall indicate a Data Segment error. Specifically, the drive shall return a General Status of 0x09 (Data Segment Error) with an Extended Status of 0x04.

The Configuration Block structure for Format Revision 2, Connection Point 82 (0x52), is a variable length data structure that extends the Format Revision 1 structure to carry Drive Power Structure Axis ID information needed to verify power structures associated with each axis instance of the multi-axis drive. The Data Segment size, in words, determines the number of Drive Power Structure Axis ID elements in the array.

The Configuration Block for Format Revision 2 is an 8-byte data structure defined as shown in Figure 51.

← 32-bit Word (octet 1 | octet 2 | octet 3 | octet 4) →		
Data Segment Configuration Block		
Format Revision	Config Bits	(reserved)
Drive Power Structure Class ID		
Drive Power Structure Axis ID[n]		

IEC

Figure 51 – Configuration Block Format Revision 2 (Connection Point 82)

In this case, if the Verify Power Rating bit is set, the device shall verify either that the Drive Power Structure Class ID of the Forward_Open matches the Drive Power Structure Class ID attribute value of the drive device or that the Drive Axis Power Structure Axis IDs match the corresponding Drive Power Structure Axis ID attribute values associated with each drive axis instance. If in either case, the values do not match, the Forward_Open service response shall indicate a Data Segment error. Specifically, the drive shall return a General Status of 0x09 (Data Segment Error) with an Extended Status of 0x04 if the Drive Power Structure Class ID values do not match. If the Drive Power Structure Axis ID values do not match the Forward_Open service returns and Extended Status of 0x05 + n, where n is the index of the first Drive Power Structure Axis ID element in the Configuration Block that failed. Single axis drives or multi-axis drives with identical power structures need only use the Drive Power Structure Class ID to verify the power rating, in which case the Revision 2 Configuration Block degenerates to the Revision 1 Configuration Block structure. But in the case of a multi-axis drive with independent power structures, the Drive Power Structure Class ID is set to 0 and the Drive Power Structure Axis ID array contains all the IDs that need to be verified against the Power Structure Axis ID attribute values associated with each drive axis instance. See descriptions for the Drive Power Structure Class ID class attribute and Drive Power Structure Axis ID instance attribute for further details.

Support for Format Revision 2 of the Configuration Block structure is required when implementing a modular multi-axis CIP Motion drive device with mixed power structures. All other types of CIP Motion devices (e.g. single axis drives, and multi-axis drives with identical power structure ratings, encoders, converters, etc) are required to either support Format Revision 1 or Format Revision 2. CIP Motion devices need only support one Format Revision. CIP Motion devices using Format Revision 1 are not required to support Format Revision 2. Similarly, CIP Motion devices using Format Revision 2 are not required to support Format Revision 1. Controllers are generally capable of sending either Configuration Block Format to the device, and shall use whatever Format Revision the specific device profile supports.

The RPI (Requested Packet Interval) of the Forward_Open for this Class 1 I/O connection is set to the Controller Update Period. This value is applied to the Controller Update Period class attribute of the Motion Device Axis Object. CIP Motion timing requires that the connection API (Actual Packet Interval) match the RPI of the controller’s Forward_Open service request. Thus, in the context of a CIP Motion I/O Connection, RPI and API are synonymous.

Once the C-to-D and D-to-C connections have been successfully established using Forward_Open service, the controller and the device are each responsible for transmission of data packets at the RPI.

The first C-to-D packet transmitted by the controller shall contain a valid Connection Format and Format Revision. The Update ID, Last Update ID, Instance Count, and Time Data Set elements are all set to zero. The Node Control element is set to 0x01 since the intention is Remote control of the device. Note that the Controller Data Valid bit in the Node Control field is not set indicating that the controller is not sending any valid instance data in this update. No additional data is required for the C-to-D connection. Figure 52 shows a typical initial C-to-D connection data block.

Initial C-to-D connection data block			
Connection Format = 6	Format Revision = 2	Update ID = 0	Node Control = 1
Instance Count = 0	0	Last Received ID = 0	Time Data Set = 0

IEC

Figure 52 – Typical initial C-to-D connection data block

Similarly, the first D-to-C packet transmitted by the device, in response to the initial C-to-D packet, shall contain a valid Connection Format and Format Revision. The Update ID, Instance Count, and Time Data Set elements are all set to zero. The Node Status element is set to 0x01 since the intention is Remote control of the device. Since there is little if any CIP Motion Connection handling functionality running in this early phase of the start-up sequence, the Node Alarms and Node Fault elements are both set to 0 as well. Note that the Device Data Valid bit in the Node Status field is not set indicating that the device is not sending any valid instance data at this time. Until the controller sets the Controller Data Valid bit in the Node Control field, the device simply continues to send just the D-C Connection Header with the Update ID, Instance Count, and Time Data Set elements all set to zero. No additional data is required for the D-to-C connection in this phase. Figure 53 shows a typical initial D-to-C connection data block.

Initial D-to-C connection data block			
Connection Format = 7	Format Revision = 2	Update ID = 0	Node Status = 0
Instance Count = 0	Node Fault/Alarm = 0	Last Update ID = 0	Time Data Set = 0

IEC

Figure 53 – Typical initial D-to-C connection data block

Once the first C-to-D and D-to-C packets have been transmitted, each device keeps its part of the CIP Motion I/O Connection alive by sending Refresh packets at the RPI. Refresh packets are defined as CIP packets that have the same CIP Sequence Count as the previously transmitted CIP packet. Transmission and processing of these Refresh packets is typically handled by low-level CIP drivers and transparent to the higher level CIP Motion Connection handler.

Once the controller has the Controller Task running and is ready to manage the CIP Motion Connection, it is ready to send configuration data to the device thus marking the end of the Connecting phase of the start-up procedure and the beginning of the Configuration phase.

6.5.3 Motion Device Axis Object configuration

Motion Device Axis Object configuration is accomplished using either the common Set_Attributes_List service or the object specific Set_Axis_Attributes_List service defined by the Motion Device Axis Object and supported by the CIP Motion I/O Connection. This subclause assumes that the controller uses the CIP Motion I/O Connection to configure the device, since this would be more typical, and the method of using the Set Attribute List service is already well understood.

The first step in the configuration process is to establish the class attributes of the Motion Device Axis Object. To do this the controller sends a Set_Axis_Attributes_List service to the class (i.e. instance 0) of the object. Figure 54 illustrates the typical contents of the first C-to-D class attribute configuration packet.

1st Class instance configuration C-to-D connection data block			
Connection Format = 6	Format Revision = 2	Update ID = 1	Node Control = 5
Instance Count = 1	0	Last Received ID = 0	Time Data Set = 0
Instance Num = 0	-	Instance Blk Size = 8	Cyclic Blk Size = 0
Cmd Blk Size = 0	Write Blk Size = 0	Event Blk Size = 0	Service Blk Size = 6
Transaction ID = 1	Service Code = 76	0	0
Set_Axis_Attributes_List Specific Request Data (2 attributes)			

IEC

Figure 54 – Typical contents of first C-to-D class attribute configuration packet

Note that the Update ID is now incrementing with every packet sent by the controller and that the Node Control element is now set to 5 indicating that the controller is now sending valid instance data. The Instance Count is 1 since in this case we are only sending data to the class, i.e. Instance Number 0. Also note that the Time Data Set is 1 and the controller is sending the Controller Update Period to the device. Only the Instance Block Size and Service Block Size have non-zero values. A value of 0 for the Cyclic Block Size and Event Block Size indicates that these blocks are not included in this C-to-D Connection Data Block.

The device processes this C-to-D packet as soon as possible to provide timely D-to-C transmission. The Update ID for the D-to-C packet is also now incrementing with every packet sent by the device and the device is indicating that the instance data is now valid. This packet may or may not include the Set_Axis_Attributes_List service request depending on the priority given to the service request processing. Figure 55 illustrates a typical response to the above first C-to-D class configuration packet assuming that the service response data is included.

1st Class instance configuration D-to-C connection data block			
Connection Format = 7	Format Revision = 2	Update ID = 1	Node Status = 5
Instance Count = 1	Node Fault/Alarm = 0	Last Received ID = 1	Time Data Set = 0
Instance Num = 0	-	Instance Blk Size = 7	Cyclic Blk Size = 0
Act Blk Size = 0	Read Blk Size = 0	Event Blk Size = 0	Service Blk Size = 5
Transaction ID = 1	Service Code = 76	General Status = 0	Extended Status = 0
Set_Axis_Attributes_List Specific Response Data (2 attributes)			

IEC

Figure 55 – Typical response to first C-to-D class configuration packet

In addition to sending this response, the device also schedules the next D-to-C transmission to be 1 Controller Update Period from the time the C-to-D packet was received. If the next C-to-D packet to the device is late or lost, the device sends the next D-to-C packet as previously scheduled, and schedules the next D-to-C packet transmission. Normally, however, the next C-to-D packet arrives prior to previously scheduled D-to-C transmission, so the device processes the packet and immediately sends the D-to-C response and schedules the next transmission one Controller Update Period later. This procedure synchronizes the device D-to-C packet transmissions with the Controller Task schedule without requiring that the controller and device clocks be synchronized to each other.

NOTE In general, the Update IDs associated with the C-to-D packet and D-to-C packet for a given cycle match.

Once the class attributes are configured, the Motion Device Axis Object configuration continues with configuration of the individual object instances until all instances are configured. This is typically done one instance at a time, but could be using multiple instances to configure all axes in parallel. If fact, the initial axis instance configuration could begin in the same packet as the class instance configuration. The device is required to handle all these cases. Figure 56 illustrates the typical contents of the first C-to-D axis instance configuration packet.

1st Axis instance configuration C-to-D connection data block			
Connection Format = 6	Format Revision = 2	Update ID = 4	Node Control = 5
Instance Count = 1	0	Last Received ID = 3	Time Data Set = 0
Instance Num = 1	–	Instance Blk Size = 84	Cyclic Blk Size = 0
Cmd Blk Size = 0	Write Blk Size = 0	Event Blk Size = 0	Service Blk Size = 82
Transaction ID = 1	Service Code = 76	0	0
Set_Axis_Attributes_List Specific Request Data (40 attributes)			

IEC

Figure 56 – Typical contents of first C-to-D axis instance configuration packet

In response to this packet the device sends the data packet as shown in Figure 57.

1st Axis instance configuration D-to-C connection data block			
Connection Format = 7	Format Revision = 2	Update ID = 4	Node Status = 5
Instance Count = 1	Node Fault/Alarm = 0	Last Received ID = 4	Time Data Set = 0
Instance Num = 0	-	Instance Blk Size = 45	Cyclic Blk Size = 0
Act Blk Size = 0	Read Blk Size = 0	Event Blk Size = 0	Service Blk Size = 43
Transaction ID = 1	Service Code = 76	General Status = 0	Extended Status = 0
Set_Axis_Attributes_List Specific Response Data (40 attributes)			

IEC

Figure 57 – Typical response to first C-to-D axis configuration packet

Once all object instances are configured, the controller marks the end of the Configuration phase of the Start-up process by setting the Configuration Complete bit in the Control Status field of the next C-to-D packet sent to the device.

6.5.4 Time Synchronization

All CIP Motion device nodes that support time synchronization services are required to be synchronized with a CIP Motion controller as part of a group, commonly embodied as a Motion Group. All Time Sync capable devices in this group are sent a Group_Sync service that contains the Grand Master Clock ID associated with the controller. This is a class service, so the packet would look something like Figure 58. Even though this example only shows the class Instance Data Block, the class instance and axis Instance Data Blocks can coexist in the same packet. This allows the Time Synchronization process and the Motion Device Object Configuration process to proceed concurrently. It is a controller requirement, however, that all class attributes associated with the Motion Device Axis Object of this device shall be configured prior to the controller sending the first Group_Sync service request.

Time Sync service request C-to-D connection data block			
Connection Format = 6	Format Revision = 2	Update ID = 12	Node Control = 5
Instance Count = 1	0	Last Received ID = 11	Time Data Set = 0
Instance Num = 0	-	Instance Blk Size = 5	Cyclic Blk Size = 0
Cmd Blk Size = 0	Write Blk Size = 0	Event Blk Size = 0	Service Blk Size = 3
Transaction ID = 3	Service Code = 28	0	0
Group_Sync Specific Request Data (Grand Master Clock ID)			

IEC

Figure 58 – Typical contents of C-to-D Time Sync service request packet

The objective of this procedure is to guarantee that all devices in the Motion Group are synchronized with each other, or in other words, that they have the same notion of time based on a common Grand Master Clock. In addition to insuring that the devices all share the same Grand Master as the controller, the procedure also guarantees that no step changes to System Time have occurred at the time of synchronization that could permanently skew a device clock relative to the controller clock.

Figure 59 illustrates how the controller first waits until its local clock is synchronized to the Master System Time clock. It is the responsibility of the Time Sync object in the controller to determine if the controller is synchronized. The controller then waits a minimum delay period, MinSyncDelay, to allow time for other components in the system to synchronize to a common Grand Master. A typical value for this delay period is 10 s.

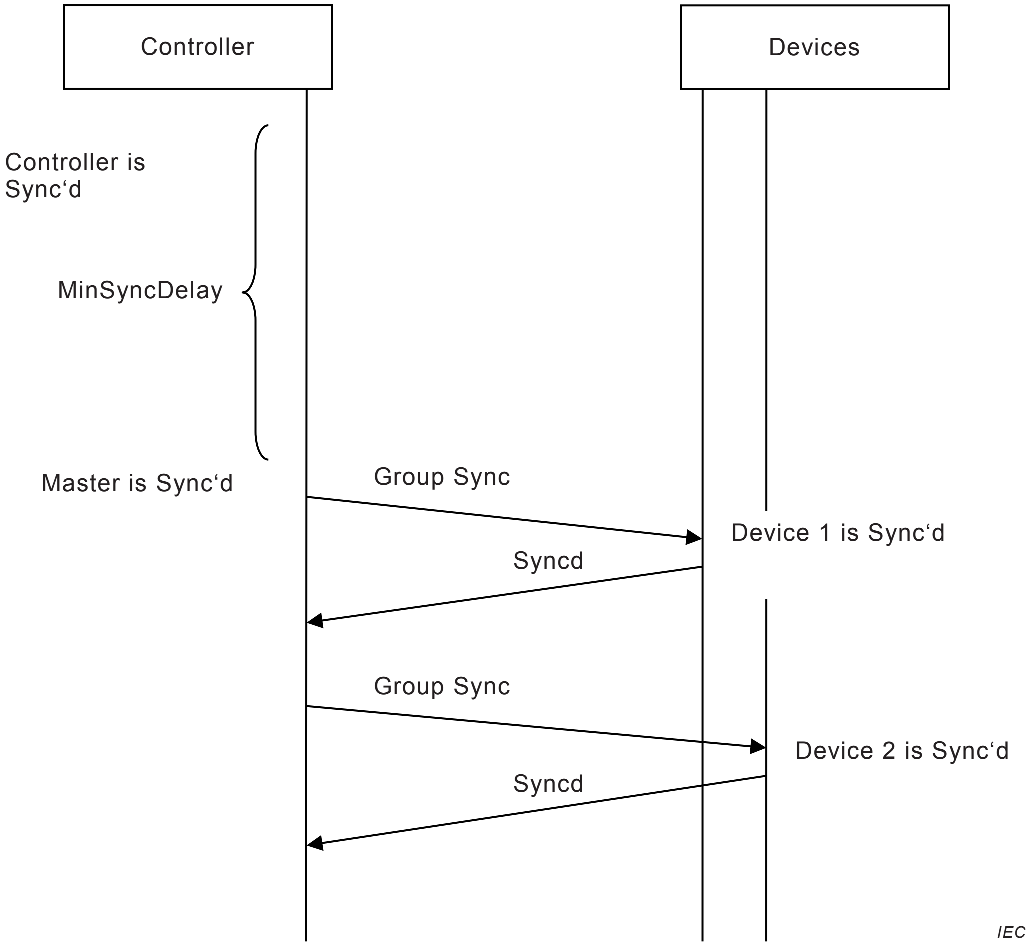

Figure 59 – Group Sync of CIP Motion devices

At the conclusion of this delay period, the controller initiates a series of Group Sync services to each of the CIP Motion devices. Each device returns a response indicating whether it is also synchronized to the controller according to the criterion described for the Motion Device Axis Object's Group_Sync service. If the Group_Sync response indicates that the device is synchronized to the controller, the controller does one last skew check and, if there's no evidence of time skew, the device and controller are considered fully synchronized. When all the devices are fully synchronized to the controller, the Motion Group is considered to be synchronized and the controller is free to begin sending cyclic data to all the CIP Motion devices associated with the Motion Group.

If the controller encounters a device that does not return a "device is synchronized" response to the Group_Sync service after repeatedly trying for at least 1 minute, the controller shall indicate that a connection fault has occurred for that device.

Once the controller and the CIP Motion devices are Group synchronized, the controller can then set the Synchronous Control bit in the Controller-to-Device Connection Header. Setting the Synchronous Control bit in the next cyclic packet sent to the devices allows the device to schedule its next Device-to-Controller Connection update based on the Controller Time Stamp contained in the connection data. Specifically, the device calculates the initial synchronous Device-to-Controller Connection update time to be:

Controller Time Stamp + 2 × Controller Update Period (the CIP Motion Connection RPI) – Transmission Processing Time.

The Transmission Processing Time is the time necessary to assemble and prepare the Device-to-Controller Connection for transmission. When the scheduled transmission occurs, the device indicates that it is now sending synchronized Device-to-Controller Connection data by setting the Synchronous Mode bit in the header of the Device-to-Controller packet sent to the controller.

The Device-to-Controller Connection Update ID shall be initialized for this first synchronous update cycle to match the Update ID of the Controller-to-Device Update ID for that cycle and shall be incremented by one every connection update period thereafter. Maintaining matched Update IDs per update cycle, therefore, does not require that the Controller-to-Device packet be processed, or even received, prior to the Device-to-Controller packet being assembled.

Data exchange between the controller and the device proceeds thereafter according the CIP Motion timing model. This marks the end of the Synchronization phase of the Start-up process.

If at the point of receiving the Synchronous Control request in the Controller-to-Device Connection Header the (Controller Time Stamp + Controller Update Period) time differs from the current device System Time by more than on Controller Update Period, the controller and device system time clocks are skewed.

That is,

If |(Controller Time Stamp + CUP) – Current System Time| > 1 CUP

Controller and Device Clocks are Skewed

Else

Controller and Device Clocks are OK

In this unlikely case, the device simply continues asynchronous operation and the Synchronous Mode bit in the header of the Device-to-Controller packet remains clear until either the calculated update time is within acceptable tolerance or a 1-minute time-out limit is exceeded. During the 1 minute time-out interval while the clocks are skewed, the Node Alarm Code from the device indicates a Clock Skew Alarm condition. Should the 1-minute clock skew time-out period be exceeded, the device faults and sets the Node Fault Code to indicate a Clock Skew Fault condition.

6.6 Device visualisation

The CIP Motion device requires visualisation components to quickly diagnose the condition of the device. These components range from bicolor LEDs to multi-character alphanumeric displays. Table 26 defines the requirements for these visualisation components and appropriate labels for these components when employed by the device.

Table 26 – CIP Motion visualisation components

Visualisation component	Need in implementation	Associated label
Network Status LED	Required	“Net Status” “NET”
Module Status LED	Required	“Mod Status” “MOD”
Axis Status LED	Conditional [a]	“Axis Status” “XS” “Axis # Status” “XS#” [b]
Seven-Segment Display	Optional	N/A
Multi-char alphanumeric display	Optional	N/A

[a] Requirement may be waived by using a multi-character alphanumeric display.

[b] Multi-axis devices need multiple Axis Status LEDs, one for each axis.

When applied to different CIP networks, the Network Status and Module Status LED behavior shall conform to the LED behavior of the associated CIP network to which the device is connected. The mapping of the device’s Module Status LED states to the Motion Device Axis Object states is fully defined in the Motion Device Axis Object section on Visualisation.

6.7 Ethernet/IP Quality of Service (QoS)

When a CIP Motion device supporting time synchronization services is targeted for operation on EtherNet/IP, standard Ethernet QoS or Quality of Service protocol shall be utilised to guarantee that CIP Sync related IEC 61588:2009 packets and CIP Motion I/O Connection packets have timely delivery through the network components. QoS for EtherNet/IP is defined in IEC 61158-4-2 (QoS behavior and optional QoS Object). While QoS mechanisms defined in IEC 61158-4-2 are specified as optional for EtherNet/IP devices, CIP Motion devices shall at least implement DSCP marking for CIP Motion packets. It is required that CIP Sync packets be passed with DSCP marking. CIP Urgent priority shall be set by default by the Forward_Open service establishing the CIP Motion Connection. It is expected that the CIP Motion device EtherNet/IP stack would give high priority to the processing of CIP Motion and IEC 61588:2009 packets over lower priority packets.

7 Motion Device Axis Object

7.1 General considerations

7.1.1 General

This subclause defines the behavior of a CIP object called the Motion Device Axis Object. This object is the main functional component of the CIP Motion device profile defined as part of the CIP Motion extensions to CIP Common. Instances of this object are required to support motion control when connected to a CIP Motion compliant controller through a CIP network. EtherNet/IP is the network of choice for high performance, synchronized multi-axis control. Other CIP networks such as ControlNet and DeviceNet could be applied to lower performance, non-synchronized drive applications such as simple variable frequency drives, and velocity loop drives and indexing drives.

7.1.2 Revision history

Clause 7 provides a complete description of the individual object attributes, their behavior, and associated object services. Since the initial release of this object class definition changes have been made that require a revision update of this object class. Table 27 represents the revision history.

Table 27 – Motion Device Axis Object revision history

Revision	Reason for object definition update
1	Initial revision (IEC 61800-7 Edition 1, obsolete)
2	Major interface changes (deprecated)
3	Modified behavior of Group_Sync service for clock skew detection. Recommended for new CIP Motion devices supporting time synchronization. Modified Get_Motor_Test_Data service to add an optional parameter request structure.

7.1.3 Object overview

The Motion Device Axis Object is one of two major object components required to support motion control under the CIP Motion control architecture. These object components are illustrated in Figure 60.

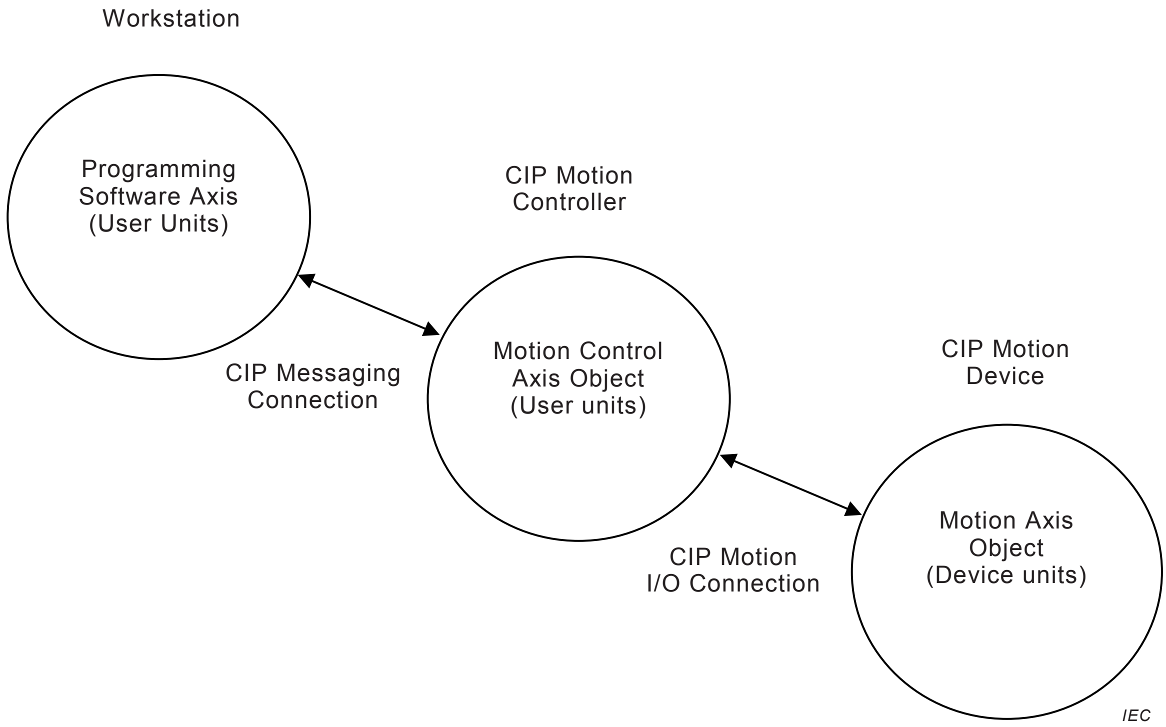

Figure 60 – Object components for CIP Motion control architecture

7.1.4 Motion Device Axis Object abstraction

The Motion Device Axis Object is an abstraction that can be applied to any motion control device function associated with a moving component of a motion control system. A given motion control device can therefore be represented by a Motion Device Axis Object instance, and since most of these device functions are associated with the motion of a machine component, a Motion Device Axis Object instance is sometimes simply referred to as an "Axis". A Motion Device Axis Object instance generally consists of one or more of the following integral elements:

- motion actuator, for example rotary and linear motors, hydraulic and pneumatic actuators;
- motion sensor, for example encoders, resolvers, linear displacement transducers;
- actuator control, for example vector, frequency, and stepper motor control, hydraulic valve control;
- amplifier that supplies power to move actuator, for example drive power structure for motors;
- converter: converts AC line power to DC bus power for one or more associated amplifiers;
- motion I/O: registration input sensors, output cam driven actuators.

Instance attributes are included in the Motion Device Axis Object for each of these integral elements to define axis behavior. When a Motion Device Axis Object has an actuator control element, command attributes are included in the object to control the dynamics of the actuator. The object also includes attributes that describe the dynamics of the actuator, whether estimated or actual.

While the most common embodiment of a Motion Device Axis Object is a "drive" device, the Motion Device Axis Object can also be applied to other motion related devices, for example a CIP Motion encoder, a CIP Motion Power Converter, a CIP Motion registration input module, etc. For these devices, only a subset of the Motion Device Axis Object attribute set is actually implemented, with the subset consisting of all attributes that are applicable to the device's function.

In the case of a multi-port feedback device, each feedback port is considered an axis and therefore assigned an instance of the Motion Device Axis Object. In the case of a multi-amplifier drive device, each amplifier is associated with an axis and assigned an instance of the Motion Device Axis Object. For drive axis instances, the Motion Device Axis Object can represent a complex closed loop vector controlled servo drive or a simple open loop Volts/Hertz drive (or VFD). Drive devices are often equipped with multiple feedback ports that can be dedicated to drive amplifier instances or used as independent master feedback sources. In the latter case, the master feedback channel is considered an axis and assigned an instance of the Motion Device Axis Object.

7.1.5 Motion Control Axis Object

For each device resident Motion Device Axis Object instance in the motion control system, there is a corresponding controller resident Motion Control Axis Object. Attribute values contained within the Motion Device Axis Object have matching attributes in the Motion Control Axis Object, the only difference being that the units by which these attributes are expressed. In the Motion Device Axis Object, attributes are expressed in fixed device units, for example feedback counts or r/min, as defined by the Motion Device Axis Object. Within the Motion Control Axis Object, attributes are expressed in user defined units, for example degrees, inches or centimeters. Since programming software interacts with the drive through the Motion Control Axis Object, all configuration, diagnostic, and motion programming is done using the user’s preferred units. Therefore, rather than programming the motion control application in, for example, feedback counts or r/min of the motor shaft, the user can program in, for example, centimeters, feet, products per minute, or whatever makes sense for the specific application.

When configuration or command attribute values are transferred to the Motion Device Axis Object via the CIP Motion I/O Connection, these attribute values are scaled by the Motion Control Axis Object into fixed device unit values required by the Motion Device Axis Object. Similarly, when signal attributes are transferred to the Motion Control Axis Object via the CIP Motion I/O Connection, these feedback-based unit values are scaled by the Motion Control Axis Object and stored in the object in user units.

7.1.6 Device control classification

Based on the variations in Control Mode and Control Method listed in 4.2, one can define a set of basic Device Function Codes around which the many attributes of the Motion Device Axis Object can be organized. Device Function Codes are designated in the tables of Clause 7 using a single letter identifier that can be used to determine what object attributes are required (or optional) for implementation of a given CIP Motion device. The list of Device Function Codes is as follows.

Function Codes:

B – Bus Power Converters (No Control Mode, No Control Method)

E – Encoder Feedback Only (No Control Mode, No Control Method)

P – Position Loop (Position Control Mode, Closed Loop Vector Control Method)

V – Velocity Loop (Velocity Control Mode, Closed Loop Vector Control Method)

T – Torque Loop (Torque Control Mode, Closed Loop Vector Control Method)

F – Frequency Control (Velocity Control Mode, Frequency Control Method)

Device Function Code Combinations:

Combinations of these letters can be used to designate a specific class of CIP Motion devices for the purposes of identifying applicable attributes.

EXAMPLE “FV” would refer to the class of all velocity controlled drives, either closed loop vector controlled or open loop frequency controlled.

Below are some combinations that appear so frequently that special Device Function Codes have been defined for convenience:

N = **BE** = All Device Functions using No Control Method

O = **F** = All Device Functions using Open Loop Control Methods, (e.g. Frequency Control)

C = **PVT** = All Device Functions using Closed Loop Control Methods (e.g. PI Vector Control)

D = **OC** = All Device Functions using Control Methods (Control Method != No Control)

In addition to these combinations there are many attributes that are applicable or not applicable to sensorless or encoderless device operation, i.e. device axis instances operating without a feedback device. Device Function Codes can be used to specify the following conditional implementation rules for attributes. In this case the following conditional implementation rules are specified:

E = Encoder-based device operation.

!E = Encoderless or Sensorless device operation.

7.1.7 Required vs. Optional in implementation

In the subclauses that follow, attributes and services are defined as Required or Optional in the implementation of the Motion Device Axis Object. Required attributes and services shall be supported in the implementation of the object. Optional attributes and services may or may not be supported in the implementation and are left to the discretion of the vendor.

For Instance Attributes, the determination of whether a given attribute or service is Required or Optional often depends on the associated Device Function Code as defined above. If an attribute or service is marked as Required for a given Device Function Code, then a device implementation shall support that attribute or service if it is intended to operate in that mode. For example, an attribute marked as required for Device Function Code “V” shall be implemented by any CIP motion device that intends to support Velocity Loop mode.

In some cases, an instance attribute or service may not be applicable to a given Device Function Code. This situation is implied when the attribute is defined as neither Required nor Optional.

Table 29 provides a convenient list of all the Instance Attributes of this object and identifies whether the attribute is Required or Optional, Conditional or Not Applicable in the implementation based on the above Device Function Code. Attribute Enumeration and Bit definitions are also designated as Required, Optional, or Conditional, with an appropriate Device Function Code if applicable. If no designation is associated with the Enumeration or Bit definition, then it is assumed that the enumeration is Required in the implementation.

For some attributes, there are conditional implementation rules that extend beyond the Device Function Code. These rules are specified in the Conditional Implementation column of Table 29.

EXAMPLE Table 28 shows an example where the Attribute PM Motor Resistance is Required (R) in the implementation if the device supports Frequency Control, Position Control, Velocity Control, or Torque Control AND the device supports Permanent Magnet motors, i.e.: "PM Motor only". The attribute is Not Applicable (-) for a Bus Power Converter or a Feedback Only device or a drive that does not support a PM motor.

Table 28 – Example for instance attribute implementation vs. Device Function Code

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C-Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
1327	Set	PM Motor Resistance	-	-	R	R	R	R	PM Motor only

Key to Table 28 and Table 29: (see 7.1.6 and paragraphs above for details)

(R)	Required attribute – shall be supported in the implementation.
(O)	Optional attribute – support is left to vendor's discretion.
(C)	Conditional attribute.
(-)	Not applicable attribute.
Set*	Indicates the attribute is normally set by the CIP Motion Connection data block and not by a Set service.

Conditional Implementation Key:

Rotary Motor	Rotary PM motor or Rotary Induction motor (motor type)
Linear Motor	Linear PM motor or Linear Induction motor (motor type)
PM	Rotary or Linear Permanent Magnet (SPM and IPM) motor (motor type)
IM	Rotary or Linear Induction Motor (motor type)
SPM	Rotary or Linear Surface PM motor (motor type)
IPM	Rotary or Linear Interior PM motor (motor type)
TT	Digital AqB (feedback type)
TP	Digital Parallel (feedback type)
SC	Sine/Cosine (feedback type)
HI	Hiperface® (feedback type)
DSL	Hiperface DSL® (feedback type)
E21	EnDat 2.1® (feedback type)
E22	EnDat 2.2® (feedback type)
SS	SSI (feedback type)
LT	LDT or Linear Displacement Transducer (feedback type)
RS	Resolver (feedback type)
Rotary Absolute	Feedback Unit – rev; Feedback n Startup Method- absolute
Linear Absolute	Feedback Unit – meter; Feedback n Startup Method- absolute
Drive Scaling	Drive device supports drive scaling functionality.

Table 29 – Instance attribute implementation vs. Device Function Code

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
40	Set*	Control Mode	R	R	R	R	R	R	
41	Set	Control Method	R	R	R	R	R	R	
45	Set	Motion Scaling Configuration	-	R	R	R	R	R	Drive Scaling only
77	Set	Motion Unit	-	R	R	R	R	R	Drive Scaling only
78	Set	Motion Resolution	-	R	R	R	R	R	Drive Scaling only
79	Set	Motion Polarity	-	R	R	R	R	R	Drive Scaling only
83	Set	Cyclic Unwind Control	-	R	R	R	R	R	Drive Scaling only
84	Set	Cyclic Unwind Position	-	R	-	R	R	R	Drive Scaling only; E
89	Set*	Control Status	O	O	O	O	O	O	
90	Set*	Actual Data Set	-	R	R	R	R	R	
91	Set*	Command Data Set	-	-	R	R	R	R	
92	Set*	Command Control	-	-	O	R	R	R	
93	Get	Command Target Time	-	-	O	R	O	O	
94	Set*	Status Data Set	R	R	R	R	R	R	
95	Set*	Registration Data Set	-	R	R	R	R	R	Drive Scaling only; E
96	Set*	Home Data Set	-	R	R	R	R	R	Drive Scaling only; E
97	Set*	Watch Data Set	-	R	R	R	R	R	Drive Scaling only; E
1310	Get	Motor Catalog Number	-	-	O	O	O	O	
1311	Get	Motor Serial Number	-	-	O	O	O	O	
1312	Get	Motor Date Code	-	-	O	O	O	O	
1313	Set	Motor Data Source	-	-	R	R	R	R	
1314	Set	Motor Device Code	-	-	R	R	R	R	
1315	Set	Motor Type	-	-	R	R	R	R	
1316	Set	Motor Unit	-	-	R	R	R	R	
1317	Set	Motor Polarity	-	-	O	O	O	O	
1318	Set	Motor Rated Voltage	-	-	R	R	R	R	
1319	Set	Motor Rated Continuous Current	-	-	R	R	R	R	
1320	Set	Motor Rated Peak Current	-	-	C	C	C	C	R – PM; O – IM
1321	Set	Motor Rated Output Power	-	-	C	C	C	C	R – PM; O – IM
1322	Set	Motor Overload Limit	-	-	O	O	O	O	
1323	Set	Motor Integral Thermal Switch	-	-	O	O	O	O	
1324	Set	Motor Max Winding Temperature	-	-	O	O	O	O	
1325	Set	Motor Winding To Ambient Capacitance	-	-	O	O	O	O	
1326	Set	Motor Winding To Ambient Resistance	-	-	O	O	O	O	
1327	Set	PM Motor Resistance	-	-	R	R	R	R	PM Motor only
1328	Set	PM Motor Inductance	-	-	R	R	R	R	SPM Motor only
1353	Set	PM Motor Lq Inductance	-	-	R	R	R	R	IPM Motor only
1354	Set	PM Motor Ld Inductance	-	-	R	R	R	R	IPM Motor only
1329	Set	Rotary Motor Poles	-	-	R	R	R	R	Rotary Motor only

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
1330	Set	Rotary Motor Inertia	-	-	O	O	O	O	Rotary Motor only
1331	Set	Rotary Motor Rated Speed	-	-	R	R	R	R	Rotary Motor only
1332	Set	Rotary Motor Max Speed	-	-	O	O	O	O	Rotary Motor only
1333	Set	Rotary Motor Damping Coefficient	-	-	O	O	O	O	Rotary Motor only
1334	Set	Linear Motor Pole Pitch	-	-	R	R	R	R	Linear Motor only
1335	Set	Linear Motor Rated Speed	-	-	R	R	R	R	Linear Motor only
1336	Set	Linear Motor Mass	-	-	O	O	O	O	Linear Motor only
1337	Set	Linear Motor Max Speed	-	-	O	O	O	O	Linear Motor only
1338	Set	Linear Motor Damping Coefficient	-	-	O	O	O	O	Linear Motor only
1339	Set	PM Motor Rated Torque	-	-	O	O	O	O	Rotary PM Motor only
1340	Set	PM Motor Torque Constant	-	-	O	O	O	O	Rotary PM Motor only
1341	Set	PM Motor Rotary Voltage Constant	-	-	R	R	R	R	Rotary PM Motor only
1342	Set	PM Motor Rated Force	-	-	O	O	O	O	Linear PM Motor only
1343	Set	PM Motor Force Constant	-	-	O	O	O	O	Linear PM Motor only
1344	Set	PM Motor Linear Voltage Constant	-	-	R	R	R	R	Linear PM Motor only
1345	Set	Induction Motor Rated Frequency	-	-	R	R	R	R	Induction Motor only
1346	Set	Induction Motor Flux Current	-	-	R	R	R	R	Induction Motor only
1347	Set	Induction Motor Stator Resistance	-	-	R	R	R	R	Induction Motor only
1348	Set	Induction Motor Stator Leakage Reactance	-	-	O	O	O	O	Induction Motor only
1349	Set	Induction Motor Magnetization Reactance	-	-	O	O	O	O	Induction Motor only
1350	Set	Induction Motor Rotor Resistance	-	-	O	O	O	O	Induction Motor only
1351	Set	Induction Motor Rotor Leakage Reactance	-	-	O	O	O	O	Induction Motor only
1352	Set	Induction Motor Rated Slip Speed	-	-	O	O	O	O	Induction Motor only
1370	Set	Load Type	-	O	O	O	O	O	Drive Scaling only
1371	Set	Transmission Ratio Input	-	O	O	O	O	O	Drive Scaling only
1372	Set	Transmission Ratio Output	-	O	O	O	O	O	Drive Scaling only
1373	Set	Actuator Type	-	O	O	O	O	O	Drive Scaling only
1374	Set	Actuator Lead	-	O	O	O	O	O	Drive Scaling only
1375	Set	Actuator Lead Unit	-	O	O	O	O	O	Drive Scaling only
1376	Set	Actuator Diameter	-	O	O	O	O	O	Drive Scaling only
1377	Set	Actuator Diameter Unit	-	O	O	O	O	O	Drive Scaling only
1400 + o	Get	Feedback n Catalog Number	-	O	-	O	O	O	E
1401 + o	Get	Feedback n Serial Number	-	O	-	O	O	O	E
1402 + o	Get	Feedback n Position	-	R	-	R	R	R	E
1403 + o	Get	Feedback n Velocity	-	R	-	R	R	R	E
1404 + o	Get	Feedback n Acceleration	-	R	-	R	R	R	E
42	Set*	Feedback Mode	-	R	R	R	R	R	E
43	Set	Feedback Master Select	-	O	-	-	-	-	
44	Set	Feedback Unit Ratio	-	-	-	O	O	-	E

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
1410 + o	Set	Feedback n Resolution Unit	-	O	-	O	O	O	E
1411 + o	Set	Feedback n Unit	-	R	-	R	R	R	E
1412 + o	Set	Feedback n Port Select	-	O	-	O	O	O	E
1413 + o	Set	Feedback n Type	-	R	-	R	R	R	E
1414 + o	Set	Feedback n Polarity	-	O	-	O	O	O	E
1415 + o	Set	Feedback n Startup Method	-	R	-	R	R	R	E
1416 + o	Set	Feedback n Cycle Resolution	-	R	-	R	R	R	E; Not LT
1417 + o	Set	Feedback n Cycle Interpolation	-	R	-	R	R	R	E; Not LT
1418 + o	Set	Feedback n Turns	-	R	-	R	R	R	E; Rotary Absolute Only
1419 + o	Set	Feedback n Length	-	R	-	R	R	R	E; Linear Absolute Only
1420 + o	Set	Feedback n Data Length	-	O	-	O	O	O	E; TP, SS
1421 + o	Set	Feedback n Data Code	-	O	-	O	O	O	E; TP, SS
1422 + o	Set	Feedback n Resolver Trans. Ratio	-	O	-	O	O	O	E; RS
1423 + o	Set	Feedback n Resolver Excitation Voltage	-	O	-	O	O	O	E; RS
1424 + o	Set	Feedback n Resolver Excit Frequency	-	O	-	O	O	O	E; RS
1425 + o	Set	Feedback n Resolver Cable Balance	-	O	-	O	O	O	E; RS
1426 + o	Set	Feedback n LDT Type	-	R	-	R	R	R	E; LT
1427 + o	Set	Feedback n LDT Recirculations	-	R	-	R	R	R	E; LT
1434 + o	Set	Feedback n Velocity Filter Bandwidth	-	O	-	O	O	O	E
1435 + o	Set	Feedback n Accel Filter Bandwidth	-	O	-	O	O	O	E
60	Set*	Event Checking Control	-	O	-	O	O	O	E
61	Get	Event Checking Status	-	O	-	O	O	O	E
62	Get	Registration 1 Positive Edge Position	-	O	-	O	O	O	E
63	Get	Registration 1 Negative Edge Position	-	O	-	O	O	O	E
64	Get	Registration 2 Positive Edge Position	-	O	-	O	O	O	E
65	Get	Registration 2 Negative Edge Position	-	O	-	O	O	O	E
66	Get	Registration 1 Positive Edge Time	-	O	-	O	O	O	E
67	Get	Registration 1 Negative Edge Time	-	O	-	O	O	O	E
68	Get	Registration 2 Positive Edge Time	-	O	-	O	O	O	E
69	Get	Registration 2 Negative Edge Time	-	O	-	O	O	O	E
70	Get	Home Event Position	-	R	-	R	O	O	E
71	Get	Home Event Time	-	O	-	O	O	O	E
72	Get	Watch 1 Event Time	-	R	-	R	O	O	E; Drive Scaling only
73	Get	Watch 2 Event Time	-	O	-	O	O	O	E; Drive Scaling only
360	Set*	Controller Position Command – Integer	-	-	-	O	-	-	

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
361	Set*	Controller Position Command – Float	-	-	-	O	-	-	
362	Set*	Controller Velocity Command	-	-	-	O	R	-	
363	Set*	Controller Acceleration Command	-	-	-	O	O	O	Accel Control Mode – R
364	Set*	Controller Torque Command	-	-	-	-	-	R	
365	Get	Fine Command Position	-	-	-	O	-	-	
366	Get	Fine Command Velocity	-	-	-	O	O	-	
367	Get	Fine Command Acceleration	-	-	-	O	O	O	
370	Set	Skip Speed 1	-	-	O	-	-	-	
371	Set	Skip Speed 2	-	-	O	-	-	-	
372	Set	Skip Speed 3	-	-	O	-	-	-	
373	Set	Skip Speed Band	-	-	O	-	-	-	
374	Set	Ramp Velocity – Positive	-	-	O	-	O	-	
375	Set	Ramp Velocity – Negative	-	-	O	-	O	-	
376	Set	Ramp Acceleration	-	-	O	-	O	-	
377	Set	Ramp Deceleration	-	-	O	-	O	-	
378	Set	Ramp Jerk Control	-	-	O	-	O	-	
380	Set	Flying Start Enable	-	-	O	-	O	-	
430	Get	Position Command	-	-	-	R	-	-	
431	Set*	Position Trim	-	-	-	R	-	-	
432	Get	Position Reference	-	-	-	R	-	-	
433	Get	Velocity Feedforward Command	-	-	-	R	-	-	
434	Get	Position Feedback	-	R	-	R	R	R	E
436	Get	Position Error	-	-	-	R	-	-	
437	Get	Position Integrator Output	-	-	-	R	-	-	
438	Get	Position Loop Output	-	-	-	R	-	-	
440	Set	Kvff	-	-	-	R	-	-	
441	Set	Kpp	-	-	-	R	-	-	
442	Set	Kpi	-	-	-	R	-	-	
443	Set	Position Lock Tolerance	-	-	-	R	-	-	
444	Set	Position Error Tolerance	-	-	-	R	-	-	
445	Set	Position Error Tolerance Time	-	-	-	O	-	-	
446	Set	Position Integrator Control	-	-	-	R	-	-	
447	Set	Position Integrator Preload	-	-	-	O	-	-	
448	Set	Position Limit – Positive	-	O	-	O	O	O	Drive Scaling only; E
449	Set	Position Limit – Negative	-	O	-	O	O	O	Drive Scaling only; E
450	Get	Velocity Command	-	-	R	R	R	-	
451	Set*	Velocity Trim	-	-	R	R	R	-	
452	Get	Acceleration Feedforward Command	-	-	-	R	R	-	
453	Get	Velocity Reference	-	-	R	R	R	-	
454	Get	Velocity Feedback	-	R	R	R	R	R	

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
455	Get	Velocity Error	-	-	-	R	R	-	
456	Get	Velocity Integrator Output	-	-	-	R	R	-	
457	Get	Velocity Loop Output	-	-	-	R	R	-	
460	Set	Kaff	-	-	-	R	R	-	
461	Set	Kvp	-	-	-	R	R	-	
462	Set	Kvi	-	-	-	R	R	-	
464	Set	Kdr	-	-	O	O	O	-	
465	Set	Velocity Error Tolerance	-	-	-	O	O	-	
466	Set	Velocity Error Tolerance Time	-	-	-	O	O	-	
467	Set	Velocity Integrator Control	-	-	-	R	R	-	
468	Set	Velocity Integrator Preload	-	-	-	O	O	-	
469	Set	Velocity Low Pass Filter Bandwidth	-	-	-	O	O	-	
470	Set	Velocity Threshold	-	O	O	O	O	O	
471	Set	Velocity Lock Tolerance	-	-	O	O	O	-	
472	Set	Velocity Standstill Window	-	R	R	R	R	R	
473	Set	Velocity Limit – Positive	-	-	O	O	O	-	
474	Set	Velocity Limit – Negative	-	-	O	O	O	-	
480	Get	Acceleration Command	-	-	-	O	O	O	
481	Set*	Acceleration Trim	-	-	-	O	O	O	
482	Get	Acceleration Reference	-	-	-	O	O	O	
483	Get	Acceleration Feedback	-	R	-	R	R	R	
485	Set	Acceleration Limit	-	-	-	O	O	O	
486	Set	Deceleration Limit	-	-	-	O	O	O	
490	Get	Torque Command	-	-	-	R	R	R	
491	Set*	Torque Trim	-	-	-	R	R	R	
492	Get	Torque Reference	-	-	-	R	R	R	
493	Get	Torque Reference – Filtered	-	-	-	R	R	R	
494	Get	Torque Reference – Limited	-	-	-	R	R	R	
496	Set	Kj	-	-	-	R	R	O	
498	Set	Friction Compensation – Sliding	-	-	-	O	O	O	
499	Set	Friction Compensation – Static	-	-	-	O	O	O	
500	Set	Friction Compensation – Viscous	-	-	-	O	O	O	
502	Set	Torque Low Pass Filter Bandwidth	-	-	-	O	O	O	
503	Set	Torque Notch Filter Frequency	-	-	-	O	O	O	
504	Set	Torque Limit – Positive	-	-	-	R	R	R	
505	Set	Torque Limit – Negative	-	-	-	R	R	R	
506	Set	Torque Rate Limit	-	-	-	O	O	O	
507	Set	Torque Threshold	-	-	-	O	O	O	
508	Set	Overtorque Limit	-	-	O	O	O	O	
509	Set	Overtorque Limit Time	-	-	O	O	O	O	
510	Set	Undertorque Limit	-	-	O	O	O	O	

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
511	Set	Undertorque Limit Time	-	-	O	O	O	O	
520	Get	Iq Current Command	-	-	-	R	R	R	
521	Get	Operative Current Limit	-	-	-	O	O	O	
522	Get	Current Limit Source	-	-	-	O	O	O	
523	Get	Motor Electrical Angle	-	-	-	R	R	R	PM Motors only
524	Get	Iq Current Reference	-	-	-	O	O	O	
525	Get	Id Current Reference	-	-	-	O	O	O	
527	Get	Iq Current Error	-	-	-	O	O	O	
528	Get	Id Current Error	-	-	-	O	O	O	
529	Get	Iq Current Feedback	-	-	-	O	O	O	
530	Get	Id Current Feedback	-	-	-	O	O	O	
531	Get	Vq Decoupling	-	-	-	O	O	O	
532	Get	Vd Decoupling	-	-	-	O	O	O	
533	Get	Vq Voltage Output	-	-	-	O	O	O	
534	Get	Vd Voltage Output	-	-	-	O	O	O	
535	Get	U Voltage Output	-	-	-	O	O	O	
536	Get	V Voltage Output	-	-	-	O	O	O	
537	Get	W Voltage Output	-	-	-	O	O	O	
538	Get	U Current Feedback	-	-	-	O	O	O	
539	Get	V Current Feedback	-	-	-	O	O	O	
540	Get	W Current Feedback	-	-	-	O	O	O	
541	Get	U Current Offset	-	-	-	O	O	O	
542	Get	V Current Offset	-	-	-	O	O	O	
543	Get	W Current Offset	-	-	-	O	O	O	
553	Set	Current Vector Limit	-	-	O	O	O	O	
554	Set	Kqp	-	-	-	O	O	O	
555	Set	Kqi	-	-	-	O	O	O	
556	Set	Kdp	-	-	-	O	O	O	
557	Set	Kdi	-	-	-	O	O	O	
558	Set	Flux Up Control	-	-	O	O	O	O	Induction Motor only
559	Set	Flux Up Time	-	-	O	O	O	O	Induction Motor only
560	Set	Commutation Startup Method	-	-	-	O	O	O	PM Motors only
561	Set	Commutation Offset	-	-	-	R	R	R	PM Motors only
562	Set	Commutation Self-Sensing Current	-	-	-	O	O	O	PM Motors only
563	Set	Commutation Polarity	-	-	-	O	O	O	PM Motors only
564	Set	Commutation Alignment	-	-	-	O	O	O	PM Motors only
565	Get	Slip Compensation	-	-	R	-	-	-	
570	Set	Frequency Control Method	-	-	R	-	-	-	
572	Set	Maximum Voltage	-	-	R	-	-	-	
573	Set	Maximum Frequency	-	-	R	-	-	-	Basic V/Hz only
575	Set	Break Voltage	-	-	R	-	-	-	Basic V/Hz only

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
576	Set	Break Frequency	-	-	R	-	-	-	Basic V/Hz only
577	Set	Start Boost	-	-	R	-	-	-	Basic V/Hz only
578	Set	Run Boost	-	-	R	-	-	-	Basic V/Hz only
600	Get	Output Frequency	-	-	R	O	O	O	
601	Get	Output Current	-	-	R	R	R	R	
602	Get	Output Voltage	-	-	R	R	R	R	
603	Get	Output Power	-	-	R	R	R	R	
604	Set	PWM Frequency	-	-	O	O	O	O	
608	Set	Zero Speed	-	-	O	O	O	O	
609	Set	Zero Speed Time	-	-	O	O	O	O	
610	Set	Stopping Action	-	-	R	R	R	R	
611	Set	Stopping Torque	-	-	-	R	R	R	
612	Set	Stopping Time Limit	-	-	-	O	O	O	
613	Set	Resistive Brake Contact Delay	-	-	O	O	O	O	PM Motors Only
614	Set	Mechanical Brake Control	-	-	O	O	O	O	
615	Set	Mechanical Brake Release Delay	-	-	O	O	O	O	
616	Set	Mechanical Brake Engage Delay	-	-	O	O	O	O	
590	Set	Proving Configuration	-	-	O	O	O	O	
591	Set	Torque Prove Current	-	-	O	O	O	O	
592	Set	Brake Test Torque	-	-	O	O	O	O	E
593	Set	Brake Prove Ramp Time	-	-	O	O	O	O	E
594	Set	Brake Slip Tolerance	-	-	O	O	O	O	E
620	Get	DC Bus Voltage	R	-	R	R	R	R	
621	Get	DC Bus Voltage – Nominal	R	-	R	R	R	R	
622	Set	Bus Configuration	O	-	O	O	O	O	
623	Set	Bus Voltage Select	-	-	R	R	R	R	
624	Set	Bus Regulator Action	R	-	R	R	R	R	Integral Converter only
625	Set	Regenerative Power Limit	-	-	O	O	O	O	Integral Converter only
627	Set	Power Loss Action	-	-	O	O	O	O	
628	Set	Power Loss Threshold	O	-	O	O	O	O	
629	Set	Shutdown Action	O	-	O	O	O	O	
630	Set	Power Loss Time	O	-	O	O	O	O	
634	Get	Integral Converter	O	-	O	O	O	O	
635	Get	Motor Capacity	-	-	R	R	R	R	
636	Get	Inverter Capacity	-	-	R	R	R	R	
637	Get	Converter Capacity	O	-	O	O	O	O	Integral Converter only
638	Get	Bus Regulator Capacity	O	-	O	O	O	O	
639	Get	Ambient Temperature	O	-	O	O	O	O	
640	Get	Inverter Heatsink Temperature	-	-	O	O	O	O	
641	Get	Inverter Temperature	-	-	O	O	O	O	
642	Get	Motor Temperature	-	-	O	O	O	O	

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
643	Get	Feedback 1 Temperature	-	O	O	O	O	O	E
644	Get	Feedback 2 Temperature	-	O	O	O	O	O	E
645	Get	Inverter Overload Limit	-	-	O	O	O	O	
646	Set	Motor Overload Action	-	-	O	O	O	O	
647	Set	Inverter Overload Action	-	-	O	O	O	O	
648	Set	Duty Select	-	-	O	O	O	O	
650	Get	Axis State	R	R	R	R	R	R	
651	Get	Axis Status	R	R	R	R	R	R	
652	Get	Axis Status – Mfg	R	R	R	R	R	R	
653	Get	Axis I/O Status	R	R	R	R	R	R	
654	Get	Axis I/O Status – Mfg	R	R	R	R	R	R	
655	Get	Axis Exceptions	R	R	R	R	R	R	
656	Get	Axis Exceptions – Mfg	R	R	R	R	R	R	
657	Get	Axis Faults	R	R	R	R	R	R	
658	Get	Axis Faults – Mfg	R	R	R	R	R	R	
659	Get	Axis Alarms	O	O	O	O	O	O	
660	Get	Axis Alarms – Mfg	O	O	O	O	O	O	
661	Get	Axis Fault Code	R	R	R	R	R	R	
662	Get	Axis Fault Type	R	R	R	R	R	R	
663	Get	Axis Fault Time Stamp	R	R	R	R	R	R	
664	Get	Axis Alarm Code	O	O	O	O	O	O	
665	Get	Axis Alarm Type	O	O	O	O	O	O	
666	Get	Axis Fault Sub Code	O	O	O	O	O	O	
667	Get	Axis Alarm Sub Code	O	O	O	O	O	O	
668	Get	Axis Fault Action	R	R	R	R	R	R	
669	Get	Axis Alarm State	O	O	O	O	O	O	
670	Get	Axis Alarm Time Stamp	O	O	O	O	O	O	
678	Get	Rotary Motor Overspeed Factory Limit	-	-	O	O	O	O	Rotary Motor only
679	Get	Linear Motor Overspeed Factory Limit	-	-	O	O	O	O	Linear Motor only
680	Get	Motor Overtemperature Factory Limit	-	-	O	O	O	O	
681	Get	Motor Thermal Overload. Factory Limit	-	-	O	O	O	O	
682	Get	Inverter Overtemperature Factory Limit	-	-	O	O	O	O	
683	Get	Inverter Thermal Overload Factory Limit	-	-	O	O	O	O	
684	Get	Converter Overtemperature Factory Limit	O	-	O	O	O	O	
685	Get	Converter Thermal Overload Factory Limit	O	-	O	O	O	O	
693	Get	Converter Ground Current Factory Limit	O	-	O	O	O	O	

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
686	Get	Bus Reg Overtemperature Factory Limit	O	-	O	O	O	O	
687	Get	Bus Reg Thermal Overload Factory Limit	O	-	O	O	O	O	
688	Get	Bus Overvoltage Factory Limit	O	-	O	O	O	O	
689	Get	Bus Undervoltage Factory Limit	O	-	O	O	O	O	
690	Get	Feedback Noise Factory Limit	-	O	O	O	O	O	E
691	Get	Feedback Signal Loss Factory Limit	-	O	O	O	O	O	E
692	Get	Feedback Data Loss Factory Limit	-	O	O	O	O	O	E
694	Set	Motor Phase Loss Limit	-	-	O	O	O	O	
695	Set	Motor Overspeed User Limit	-	-	O	O	O	O	
696	Set	Motor Overtemperature User Limit	-	-	O	O	O	O	
697	Set	Motor Thermal Overload User Limit	-	-	O	O	O	O	
698	Set	Inverter Overtemperature User Limit	-	-	O	O	O	O	
699	Set	Inverter Thermal Overload User Limit	-	-	O	O	O	O	
700	Set	Converter Overtemperature User Limit	O	-	O	O	O	O	Integral Converter only
701	Set	Converter Thermal Overload User Limit	O	-	O	O	O	O	Integral Converter only
709	Set	Converter Ground Current User Limit	O	-	O	O	O	O	Integral Converter only
702	Set	Bus Reg Overtemperature User Limit	O	-	O	O	O	O	Integral Converter only
703	Set	Bus Reg Thermal Overload User Limit	O	-	O	O	O	O	Integral Converter only
704	Set	Bus Overvoltage User Limit	O	-	O	O	O	O	
705	Set	Bus Undervoltage User Limit	O	-	O	O	O	O	
706	Get	Feedback Noise User Limit	-	O	O	O	O	O	E
707	Get	Feedback Signal Loss User Limit	-	O	O	O	O	O	E
708	Get	Feedback Data Loss User Limit	-	O	O	O	O	O	E
672	Set	Axis Exception Action	R	R	R	R	R	R	
673	Set	Axis Exception Action – Mfg	R	R	R	R	R	R	
674	Get	Initialization Faults	R	R	R	R	R	R	
675	Get	Initialization Faults – Mfg	R	R	R	R	R	R	
676	Get	Start Inhibits	R	-	R	R	R	R	
677	Get	Start Inhibits – Mfg	R	-	R	R	R	R	
756	Get	APR Faults	-	O	-	O	O	O	E; Drive Scaling only
757	Get	APR Faults – Mfg	-	O	-	O	O	O	E; Drive Scaling only
710	Get	Control Power-up Time	O	-	O	O	O	O	
711	Get	Cumulative Run Time	O	-	O	O	O	O	
712	Get	Cumulative Energy Usage	O	-	O	O	O	O	
713	Get	Cumulative Motor Revs	-	-	O	O	O	O	
714	Get	Cumulative Main Power Cycles	O	-	O	O	O	O	Integral Converter only

Instance attribute			Implementation by Device Function Code						
Attr. ID	Access rule	Attribute name	N		O	C- Closed Loop Control			Conditional implementation
			B	E	F	P	V	T	
715	Get	Cumulative Control Power Cycles	O	-	O	O	O	O	Integral Converter only
720	Get	Inverter Rated Output Voltage	-	-	R	R	R	R	
721	Get	Inverter Rated Output Current	-	-	R	R	R	R	
722	Get	Inverter Rated Output Power	-	-	R	R	R	R	
723	Get	Converter Rated Output Current	O	-	O	O	O	O	
724	Get	Converter Rated Output Power	O	-	O	O	O	O	
725	Get	Drive Power Structure Axis ID	-	-	O	O	O	O	
730	Get	Digital Inputs	O	-	O	O	O	O	
731	Set	Digital Outputs	O	-	O	O	O	O	
732	Get	Analog Input 1	O	-	O	O	O	O	
733	Get	Analog Input 2	O	-	O	O	O	O	
734	Set	Analog Output 1	O	-	O	O	O	O	
735	Set	Analog Output 2	O	-	O	O	O	O	
736	Set	Drive Enable Input Checking	O	-	O	O	O	O	
737	Set	Hardware Overtravel Input Checking	-	-	O	O	O	O	
750	Set	Local Control	O	O	O	O	O	O	
760	Get	Axis Safety State	-	O	O	O	O	O	Safety only
761	Get	Axis Safety Status	-	O	O	O	O	O	Safety only
762	Get	Axis Safety Status – Mfg	-	O	O	O	O	O	Safety only
763	Get	Axis Safety Faults	-	O	O	O	O	O	Safety only
764	Get	Axis Safety Faults – Mfg	-	O	O	O	O	O	Safety only

7.2 Class attributes

7.2.1 General

The table of attributes in Table 30 applies to the Motion Device Axis Object class, which are referenced as instance 0. These attributes exist even before any Motion Device Axis Object instances have been created. Since they are not tied to any particular axis instance of the CIP Motion device, the class attributes are generally used to address parametric behavior that applies to all axis instances, for example the communications node behavior. As instances of this class are created, they are given consecutive instance numbers starting at 1.

NOTE 1 No attributes associated with this object require Non-Volatile storage, so a "No" is implied for all attributes under the NV column.

NOTE 2 Vendor specific bits, and enumerations provide space for device vendors to provide additional product features.

Key to Table 30: (see 7.1.6 and 7.1.7 for details)
(R) – Required bit or enumeration – shall be supported in the implementation.
(O) – Optional bit or enumeration – support is left to vendor's discretion.
Set* – Indicates the attribute is normally set by the CIP Motion Connection data block and not by a Set service.

Table 30 – Class attributes for the Motion Device Axis Object

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1	Required	Get		Revision	UINT	Revision of this object.	The current value assigned to this attribute for this revision of the object is three (3). Revision 1 of this object is designated as obsolete. Revision 2 is no longer recommended for new devices supporting time synchronization.
2 to 7	These class attributes are optional and are described in IEC 61158-5-2 and IEC 61158-6-2.						
14	Required	Set*		Node Control	BYTE	Contains bits used to control the behavior of the associated device communications node.	See semantics in 7.2.2.1.
15	Required	Get		Node Status	BYTE	Contains bits used to indicate the status of the associated device communications node.	See semantics in 7.2.2.2.
16	Required	Get		Node Fault Code	USINT	Contains numeric code of active node fault condition. The Node Fault Code is used to provide diagnostic detail to pin-point the source of the fault condition.	See semantics in 7.2.2.3.
17	Optional	Get		Node Alarm Code	USINT	Contains numeric code of the current active alarm condition. The Node Alarm Code is used to provide diagnostic detail to pin-point the source of the alarm condition.	See semantics in 7.2.2.4.
18	Required	Set*		Controller Update Period	UDINT	Represents the period between updates of the controller that is also the period between CIP Motion C-to-D Connection updates.	Nanoseconds
19	Required	Set*		Controller Time Offset	LINT	This element represents the 64-bit value at the beginning of the Controller Update Period that is associated with the Controller Time Stamp value. The Controller Time Offset is the value that is added to the controller's local clock to produce System Time.	Nanoseconds
20	Required	Set*		Controller Time Stamp	LINT	This element represents the 64-bit System Time value at the beginning of the Controller Update Period when the controller's update timer event occurred. It is calculated by the controller as the sum of the controller's local clock value when update timer event occurred and the controller's System Time Offset value given by Controller Time Offset. The Controller Time Stamp is therefore directly associated with the command data contained in the connection. The time stamp format is absolute and follows the CIP Sync standard with 0 corresponding to 1970-01-01.	Nanoseconds (CIP Sync absolute)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
21	Optional	Set		Controller Update Delay High Limit	USINT	Represents high limit delay threshold for a Controller-to-Device (C-to-D) Connection update. This delay is specified in units of Controller Update Periods. Exceeding this limit results in a Control Connection Update Fault.	Number of Controller Update Periods
22	Optional	Set		Controller Update Delay Low Limit	USINT	Represents low limit delay threshold allowed for a Controller-to-Device (C-to-D) Connection update. This delay is specified in units of Controller Update Periods. Exceeding this limit results in a Control Connection Update Alarm.	Number of Controller Update Periods
28	Optional	Get		Sync Variance	UDINT	Represents the current statistical variation of the System Time Offset value from the mean System Time Offset value.	Nanoseconds
29	Optional	Set		Sync Threshold	UDINT	Determines the threshold for variation of System Time used to determine if the CIP Motion device is considered synchronized to a PTP master clock. This value is passed to the device's PTP subsystem to establish the sychronization criteria.	Nanoseconds Default: device dependent minimum value.
30	Optional	Get		Sync Update Delay	REAL	This value is the product of the number of networks hops the device is away from the grandmaster, as determined by the Time Sync Object attribute, Steps Removed, and the time synchronization interval, as determined by the Time Sync Object attribute, Sync Interval.	Seconds
31	Optional	Set*		Time Data Set	BYTE	This bit-mapped byte contains flags that determine the usage and format of the controller and device timing information. This value is derived from Time Configuration element sent by the controller in the Controller-to-Device (C-to-D) Connection and becomes the Time Configuration value sent by the device in the Device-to-Controller (D-to-C) Connection. Time Stamps bit: Time Stamp included in D-to-C connection. Time Offset bit: Time Offset included in D-to-C connection. Update Diagnostics bit: Device update diagnostic data included in D-to-C connection. Includes number of lost C-to-D updates, and late C-to-D updates. Time Diagnostics bit: Device timing diagnostic data included in D-to-C connection. Includes C-to-D data received time stamp, and D-to-C data transmit time stamp. Time Offset and Update Diagnostic bits need only be set when the associated data has changed value since the last D-to-C update.	Bit Field: 0 = Update Period 1 = Time Stamps 2 = Update Diagnostics 3 = Time Diagnostics 4 to 7 = (Reserved)
32	Optional	Set		Lost Controller Updates	BYTE	This byte value represents the accumulated number of lost Controller-to-Device Connection packets detected since the connection was opened and synchronized. Lost packets are detected by examining the Update ID of received packets from the controller. A packet is determined to be lost when its expected Update ID is skipped. The Lost Controller Updates attribute is incremented for every Update ID that is skipped.	Number of Lost Packets

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
33	Optional	Set		Late Controller Updates	BYTE	This byte value represents the number of late Controller-to-Device Connection packets detected since the connection was opened and synchronized. Late packets are detected by examining the Time Stamp of received packet from the controller. A packet is determined to be late when the difference between its Controller Time Stamp and the device's Data Received Time Stamp exceeds the Controller Update Period.	Number of Late Packets
34	Required	Get		Drive Power Structure Class ID	UDINT	This value represents an ID assigned by the drive vendor that uniquely identifies the power structure associated with the class instance. As a class instance attribute, the ID reflects power structure hardware which is common to all axis instances or identical for all axis instances (excluding master feedback axes). Power structure hardware that is unique to the axis instance shall be identified using the Drive Power Structure Axis ID attribute for that instance. In that case, the Drive Power Structure Class ID is simply set to 0. A single axis drive, however, shall use the Drive Power Structure Class ID to identify the power structure of the single axis instance. The Drive Power Structure Class ID is included as part of the Forward_Open service to the Motion Device Axis Object class to confirm that the drive's power structure matches the configuration in the controller.	ID number
35	Conditional [a]	Set		Step Threshold	UDINT	Determines the threshold for step changes in System Time as it applies to the Step Compensation algorithm. This value is generally set by the controller to the RPI of the CIP Motion I/O Connection. For more details, see the Step Compensation section of the Drive Device profile specification. The Step Threshold attribute value is also used by the Group_Sync service to identify clock skew. For more details on clock skew detection during device initialization refer to the Group_Sync service. Step Threshold is required for devices that implement the Group_Sync service with clock skew detection introduced in Revision 3 of this object. The controller may use successful access to this attribute to determine if the device supports the Group_Sync algorithm with clock skew detection.	Nanoseconds Default: 1 000 000

[a] The Step Threshold attribute is required when the device implements the Group_Sync service, otherwise it is not allowed.

7.2.2 Semantics

7.2.2.1 Attribute No. 14 – Node Control

Contains bits used to control the behavior of the associated motion device communications node, as specified in Table 31.

Table 31 – Node Control bit definitions

Bit	Req./Opt.	Name	Description
0	R	Remote Control	This bit is used to request that the device node turn over complete control of the device to the controller. If this bit is clear, the device is under the exclusive control of the local interface; the CIP Motion controller cannot affect the behavior of the device in any way other than to switch the device back to Remote Control mode. If the CIP Motion Connection is lost while the device is under remote control, the device may then be controlled via the local interface.
1	O	Sync Control	This bit is used by the controller to request synchronous operation of the device. Synchronous operation is defined as having the device node's local timer synchronized with System Time and that the device node start using the Controller Time Stamp and Time Offset to process the connection data and schedule future Device-to-Controller Connection updates. The Synchronous Control bit implies that the Controller Time Stamp and Time Offset values are valid. The bit shall remain set for the duration of synchronous operation. In asynchronous mode, there is no requirement for the local timer to be synchronized nor is time-stamping necessary or the associated time data even valid. Generally, this bit is set by the controller when all CIP Motion devices associated with the controller have given a successful Group_Sync response.
2	R	Controller Data Valid	This bit shall be set for the device node to process the Instance Data Blocks. During the connection initialization sequence there may be a period of time where the connection data in the Instance Data Blocks is not yet initialized. This condition can be recognized by the device by first checking this bit and, if clear, it is not necessary for the device to process any of the connection data beyond the connection header.
3	R	Node Fault Reset	This bit is used to request that the device perform a reset of the communications node and attempt to clear the Node Fault Code. When the controller detects that the Node Fault Code is clear, the Node Fault Reset bit is cleared. If the communication fault persists, the Node Fault Code may persist, in which case a power cycle or a complete reconfiguration process shall be performed.
4 to 5			(Reserved)
6 to 7			(Vendor Specific)

7.2.2.2 Attribute No. 15 – Node Status

Contains bits used to indicate the status of the associated CIP Motion device's communications node, as specified in Table 32.

Table 32 – Node Status bit definitions

Bit	Req./Opt.	Name	Description
0	R	Remote Mode	The Remote Mode bit is used to indicate that the device node has turned over total control of the device to the controller. If this bit is clear, the device does not act on any data contained in the Controller-to-Device cyclic or service channels.
1	O	Sync Mode	The Sync Mode bit indicates whether or not the device node is synchronized. Synchronous operation is defined as having the device node's local timer synchronized with System Time and that the device node is using the connection time stamps to process the connection data. The Sync Mode bit being set also implies that the Device Time Stamp is valid. The bit shall remain set for the duration of synchronous operation. If the Sync Mode bit is clear, the device is said to be in Asynchronous mode. In Asynchronous mode, there is no requirement for the local timer to be synchronized nor is time-stamping necessary or even valid. Generally, this bit is set in response to the Sync Control bit being set by the controller.
2	R	Device Data Valid	The Device Data Valid bit shall be set for the controller to process the Instance Data Blocks from the device node. During the connection initialization sequence, there may be a period of time where the connection data in the Instance Data Blocks is not yet initialized. This condition can be recognized by the controller by first checking this bit and if set, it is not necessary for the controller to process any of the connection data beyond the connection header.
3	R	Device Node Fault	The Device Node Fault bit is used to indicate that the device has detected one or more fault conditions related to the communications node. Specific fault conditions can be determined by the Node Fault Code attribute. If this bit is clear, there are no Node Fault conditions present. The Device Node Fault bit may be cleared by setting the Node Fault Reset bit in the Node Control word. All associated axes are disabled when a Node Fault condition is present.
4 to 5			(Reserved)
6 to 7			(Vendor Specific)

7.2.2.3 Attribute No. 16 – Node Fault Code

This attribute is a 4-bit fault code used to indicate the presence of specific fault condition associated with the CIP Motion device's communications node, as specified in Table 33

Table 33 – Node Fault Code definitions

Code	Name	Description
0	No Fault	Indicates that there is no fault condition currently present at the device communications node.
1	Control Connection Update Fault	The Control Connection Update Fault code is used to indicate that updates from the controller over the C-to-D connection have been excessively late or consecutively lost as determined by the Controller Update Delay High Limit attribute value.
2	Processor Watchdog Fault	The Processor Watchdog Fault code indicates that the processor associated with the device node has experienced an excessive overload condition that has tripped the associated processor watchdog mechanism.
3	Hardware Fault	The Hardware Fault code indicates that the critical support hardware (FPGA, ASIC, etc.) associated with the device node has experienced a fault condition.
4	Data Format Error	This fault code indicates that an error has occurred in the data format between the controller and the device, e.g. a Format Revision mismatch.
5	Clock Skew Fault	Clock Skew Fault code indicates that the drive has detected significant difference between the drive’s System Time and the controller’s System Time that prevented the drive from switching to synchronous operation after a time out period.
6	Control Connection Loss Fault	The Control Connection Loss fault code indicates that the CIP Motion C-to-D connection from the controller has timed out.
7	Clock Sync Fault	The fault condition is an indication that the local IEC 61588 clock has lost synchronization with the master and was not able to resynchronize within the allotted timeout (e.g. 40 s to 60 s).
8	Logic Watchdog	The Logic Watchdog Fault code indicates that an auxiliary logic component (e.g. FPGA, or ASIC) associated with the device node has experienced an excessive overload condition that has tripped the associated logic watchdog mechanism.
9	Duplicate Address	The Duplicate Address Fault code indicates that a device node has been detected on the network that using the same Node Address as this device node. For Ethernet, this address would be the IP Address of the device.
10 to 15	(Reserved)	

7.2.2.4 Attribute No. 17 – Node Alarm Code

This attribute is a 4-bit alarm code used to indicate specific alarm conditions of the associated CIP Motion device’s communications node, as specified in Table 34. Alarms do not result in any direct action. Individual Node Alarms are automatically cleared by the drive device after 10 s. If the alarm condition persists, the associated Node Alarm is re-posted.

Table 34 – Node Alarm Code definitions

Code	Name	Description
0	No Alarm	Indicates that there is no alarm condition currently present at the device communications node.
1	Control Connection Update Alarm	The Control Connection Update Alarm code is used to indicate that updates from the controller over the C-to-D connection have been late or consecutively lost as determined by the Controller Update Delay Low Limit attribute value.
2	Processor Overload Alarm	The Processor Overload Alarm code indicates that the processor associated with device is experiencing overload conditions that could eventually lead to a fault.
3	Clock Jitter Alarm	Clock Jitter Alarm code indicates that the Sync Variance has exceeded the Sync Threshold while the device is running in Sync Mode.
4	Clock Skew Alarm	Clock Skew Alarm code indicates that the drive has detected significant difference between the drive's System Time and the controller's System Time that is preventing the drive from switching to synchronous operation.
5	Clock Sync Alarm	The Clock Sync Alarm code indicates that the device's local clock has lost synchronization with the master clock for a short period of time (e.g. 10 s to 20 s) during synchronous operation. This alarm condition can also occur when a change in the master clock source has been detected. The Clock Sync Alarm is an indication that the local IEC 61588 clock has shifted back to its start-up mode to quickly synchronize to the master clock.
6	Node Address Alarm	The Node Address Alarm code indicates that the Node Address setting of the device has been changed during device operation and may no longer be valid.
7 to 15	(Reserved)	

7.3 Instance attributes

7.3.1 General

Subclause 7.3 lists all the supported attributes of a Motion Device Axis Object instance. Because of the large number of attributes listed in 7.3, attributes have been organised by functional category. Each functional grouping may be further organised by first listing the object Status and Signal attributes, followed by the object Configuration attributes.

NOTE 1 Due to the large number of instance attributes supported by this object, 16-bit Attribute IDs are used.

NOTE 2 No attributes associated with this object require Non-Volatile storage, so a "No" is implied for all attributes under the NV column. The Non-Volatile storage function is provided by the motion controller and specifically, the Motion Control Axis Object.

Unless otherwise specified, all object attributes default to 0 at device power-up. Since it is the motion controller working in conjunction with user driven configuration software that sets device attribute values, it is the responsibility of the controller to determine appropriate default values. For this reason, specification of default attribute values is addressed in the Motion Control Axis Object and are not within the scope of this specification.

Range limits associated with configuration attributes are specified by the device manufacturer. Attempting to set an attribute to a value that is out of range shall result in a General Status Error Code 9, for 'Invalid Attribute Value'. Since axis configuration tools interface directly to the motion controller, not to the CIP Motion device, it is the responsibility of the motion controller to enforce the attribute range limits of the device. For this reason, specification of attribute range limits is addressed in the Motion Control Axis Object and are not within the scope of this specification.

The one exception to erring an out of range configuration attribute value is when the out of range value does not impact the behavior of the device. An example of this condition would be the Torque LP Filter Bandwidth attribute. If the drive's maximum Torque LP Filter Bandwidth is 2 000 Hz based on sample rate limitations and the controller tries to set the attribute to 5 000 Hz, the drive can simply apply 2 000 Hz with the same resultant behavior. In this case, the vendor may choose to clamp the value to the device limit or simply disable the filter.

All reserved and otherwise unused bits and enumerations are set to 0.

Key to Tables: (see 7.1.6 and 7.1.7 for details)
(R) – Required bit or enumeration – shall be supported in the implementation
(O) – Optional bit or enumeration – support is left to vendor's discretion

Device Function Codes:

B – Bus Power Converter

E – Encoder, Feedback Only

F – Frequency Control (V/Hz or VFD)

P – Position Control Loop

V – Velocity Control Loop

T – Torque Control Loop

N – All Device Functions using No Control Method

O – All Device Functions using Open Loop Control Methods (Frequency Control)

C – All Device Functions using Closed Loop Vector Control (PI Vector Control Method)

D – All Device Functions using (Control Method != No Control)

All – All Device Functions

Conditional implementation rules:

E = Encoder-based Feedback channel present

!E = Encoderless or Sensorless device operation, no Feedback channel present

All – All Control Modes

Set* – Indicates the attribute is normally set by the CIP Motion Connection Header and not by a Set service.

% Device Rated Units – defined as the percentage of the continuous rating of the device with 100 % implying operation at the continuous rated specification for the device. This unit can be applied to attributes related to speed, torque, force, current, voltage, and power. Applicable "Devices" can be Motor, Inverter, Converter, or Bus Regulator. This unit can be used independently of whether the attribute value represents an instantaneous level or a time-averaged level; the appropriate unit for the device rating is implied. As with all attributes that are in units of %, an attribute value of 100 means 100 %.

Dynamic Units: Attributes that relate to motion dynamics typically express displacement in terms of the selected feedback or motor device units. Since the CIP Motion device can use different feedback channels for different control loops depending on the Feedback Mode , the determination of which feedback device applies, if any, depends on the Feedback Mode.

Table 35 provides a cross-reference table to determine the appropriate feedback counts or units to apply to an attribute based on its Dynamic Unit type (position, velocity, or acceleration), and the configured Feedback Mode . If the CIP Motion device supports Scaling functionality, then the Position Control Unit may be expressed in Motion Counts and Motion Units, independent of the Feedback Mode. The Scaling function, if enabled, converts Motion Counts to/from Feedback Counts, and Motion Units to/from Feedback Units. When an axis instance is configured for No Feedback, i.e. sensorless/encoderless operation (V/Hz, sensorless velocity control, etc.), feedback counts do not apply, so motion dynamics shall be expressed in terms of motor displacement.

Table 35 – Dynamic Units vs. Feedback Mode

Dynamic Units	Scaling Enabled	Feedback Mode				
		No Feedback	Master Feedback	Motor Feedback	Load Feedback	Dual Feedback
Position Control Units	Motion Counts	Motor Counts	Feedback n Counts [a]	Feedback 1 Counts	Feedback 2 Counts	Feedback 2 Counts
Velocity Control Units	Motion Units	Motor Units	Feedback n Units [a]	Feedback 1 Units	Feedback 2 Units	Feedback 1 Units
Accel Control Units	Motion Units	Motor Units	Feedback n Units [a]	Feedback 1 Units	Feedback 2 Units	Feedback 1 Units
[a] Feedback Channel determined by Master Feedback Select attribute						

7.3.2 Motion Control configuration attributes

7.3.2.1 General

The following attribute table in Table 36 contains basic motion control configuration attributes associated with a Motion Device Axis Object instance. These attributes govern aspects of the overall behavior of the Motion Device Axis Object.

Set* ➔ These attributes are generally updated via the cyclic Command Data Set of the CIP Motion C-to-D Connection. When included as cyclic command data, these attributes cannot be updated via a Set service.

Table 36 – Motion Control configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
40	Required – All	Set*		Control Mode	USINT	Attribute whose lower 4-bit enumeration determines the general dynamic control behavior of the motion axis instance.	Bits 0 to 3: Enumeration: Bits 4 to 7: (Reserved) See semantics in 7.3.2.2.1
41	Required – All	Set		Control Method	USINT	Enumerated code that determines the basic motor control algorithm applied by the device to control the dynamic behavior of the motor.	See semantics in 7.3.2.2.2

7.3.2.2 Semantics

7.3.2.2.1 Attribute No. 40 – Control Mode

The Control Mode attribute is a 4-bit enumeration that determines the specific dynamic behavior of the motor that the device is to control for this axis instance. The system view of these control modes are described in detail in 4.2. Table 37 shows a brief summary of the Control Mode enumerations.

Table 37 –Control Mode enumeration definitions

Enum.	Req./Opt.	Name	Description
0	R/N	No Control	No motor control is provided in this mode but interface to a specific feedback device as a master feedback source is possible via the Feedback Mode attribute.
1	R/P	Position Control	Device seeks to control the position, or orientation, of the motor.
2	R/PV	Velocity Control	Device seeks to control the velocity of the motor.
3	O	Acceleration Control	Device seeks to control the acceleration of the motor.
4	R/C	Torque Control	Device seeks to control the torque output of the motor.
5 to 15		(Reserved)	-

7.3.2.2.2 Attribute No. 41 – Control Method

The Control Method attribute is an 8-bit enumerated code that determines the basic control algorithm applied by the device to control the dynamic behavior of the motor associated with an axis instance, as specified in Table 38.

Table 38 – Control Method enumeration definitions

Enum.	Req./Opt.	Name	Description
0	R/N	No Control	No Control is associated with a Control Mode of No Control where there is no explicit motor control provided by the device for this axis instance.
1	R/F	Frequency Control	Frequency Control is an “open loop” control method that applies voltage to the motor, generally in proportion to the commanded frequency or speed. This control method is associated with variable frequency drives (VFDs) or so-called Volts/Hertz drives.
2	R/C	PI Vector Control	PI Vector Control is a “closed loop” control method that uses actual or estimated feedback for closed loop cascaded PI control of motor dynamics, i.e. position, velocity, acceleration, and torque, and always includes independent closed loop PI control of Iq and Id components of the motor current vector.
3 to 127		(Reserved)	-
128 to 255		(Vendor specific)	-

7.3.3 Motion Scaling attributes

7.3.3.1 General

The attribute table in Table 39 contains basic motion scaling configuration attributes associated with a Motion Device Axis Object instance. The Scaling function, if enabled, converts Motion Counts to/from Feedback Counts, and Motion Units to/from Feedback Units. Scaling functionality also encompasses cyclic unwind operation and motion polarity.

Table 39 – Motion Scaling attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
45	Required – Scaling	Set		Motion Scaling Configuration	USINT	Enumerated attribute used to determine whether the scaling function is performed by the controller or the drive. The Control Scaling Configuration selection configures the control system to perform the scaling calculations in the controller. In this mode, the controller interacts with the drive in terms of Feedback Counts or Motor Units, hence no scaling operations are required by the drive. Also, in Control Scaling mode the controller is responsible for Cyclic Unwind operations associated with Cyclic Travel Mode. The Drive Scaling selection configures the system to perform the scaling calculations in the drive device. In this mode, the controller interacts with the drive in terms of Motion Counts or Motion Units and the drive is responsible for conversion to equivalent Feedback Counts and Motor Units. Also, in Drive Scaling mode the drive is responsible for Cyclic Unwind operations associated with Cyclic Travel Mode.	Enumeration: 0 = Control Scaling (R) 1 = Drive Scaling (O) 2 to 255 = reserved
77	Required – Scaling	Set		Motion Unit	USINT	This enumerated value determines the unit of measure used to express the Motion Resolution used by motion planner functions. See semantics in 7.3.3.2.1	Enumeration: 0 = Motor Rev 1 = Load Rev 2 = Feedback Rev 3 = Motor mm 4 = Load mm 5 = Feedback mm 6 = Motor inch 7 = Load inch 8 = Feedback inch 9 = Motor r/s 10 = Load r/s 11 = Motor m/s 12 = Load m/s 13 = Motor inch/s 14 = Load inch/s 15 to 255 = (reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
78	Required – Scaling	Set		Motion Resolution	DINT	An integer value that determines the number of Motion Counts per Motion Unit used by the scaling function to convert between Motion Counts and Feedback Counts. See semantics in 7.3.3.2.1	Motion Counts / Motion Unit
79	Required – Scaling	Set		Motion Polarity	USINT	When Motion Scaling Configuration is set for Drive Scaling, Motion Polarity can be used to switch the directional sense of the motion control system. (see semantics in 7.3.3.2.2 for details)	Enumeration: 0 = Normal Polarity 1 = Inverted Polarity 2 to 255 = (Reserved)
83	Required – Scaling	Set		Cyclic Unwind Control	BOOL	When this Boolean attribute is set true (1), it enables the cyclic unwind capability of the scaling function. This feature provides infinite positioning range by “unwinding” the axis position whenever the axis moves through a complete machine cycle. The number of Motion Counts per machine cycle of the axis is specified by the Cyclic Unwind Position attribute. Thus, if the axis is configured Cyclic Operation, implementation of the Cyclic Unwind Position attribute is required. If the Cyclic Unwind attribute is false (0), indicating non-cyclic operation, no “unwind” operation is performed, and unidirectional motion continues to accumulate position and can eventually exceed the range of the position representation resulting in a roll-over. For this reason, non-cyclic travel is typically limited for position control applications.	0 = Non-Cyclic Operation (R) 1 = Cyclic Operation (E)
84	Required – Scaling E	Set		Cyclic Unwind Position	DINT	This integer value is used to “unwind” command and actual position values every machine cycle. Cyclic unwind functionality provides infinite position range for cyclic axes by subtracting the Cyclic Unwind Position value from both the actual and command position whenever the axis reaches or crosses the Cyclic Unwind Position. Similarly when moving in the reverse direction, the Cyclic Unwind Position value is added to both the actual and command position whenever the axis reaches or crosses zero. Thus command and actual position values shall never be outside the range of 0 and the Cyclic Unwind Position. To avoid accumulated error due to round-off with irrational conversion constants, the unwind value is represented as an integer number of Motion Counts per Cycle.	Motion Counts / Unwind Cycle

7.3.3.2 Semantics

7.3.3.2.1 Attribute No. 77 and 78 – Motion Unit/Resolution

The Motion Resolution attribute determines how many Motion Counts there are in a Motion Unit. A Motion Count is the fundamental unit of displacement used by the Motion Planner and a Motion Unit is the standard engineering unit of measure for motion displacement. Motion Units may be configured as Revs, inches, or millimeters depending on the specific application. In general, the Motion Resolution value may be may be configured in Motion Counts per Motion Unit independent of the resolution of the feedback device(s) used. The drive's scaling function takes care of scaling between Feedback Counts and Motion Counts. Providing a configurable Motion Resolution value is particularly useful for addressing Fractional Unwind applications where it is necessary to have an integer number of Motion Counts per Unwind Cycle.

Valid Motion Unit attribute selections are determined by the Feedback Mode, Load Type, and Linear Actuator Unit (Lead Unit or Diameter Unit) values according to Table 40.

Table 40 – Motion Unit selection rules

Feedback mode	Load type	Linear actuator unit	Motion unit
No Feedback	Direct Rotary	-	Motor r/s
No Feedback	Rotary Transmission	-	Load r/s
No Feedback	Linear Actuator	mm/r or mm	Load m/s
No Feedback	Linear Actuator	inch/r or inch	Load inch/s
Master Feedback	Direct Rotary	-	Feedback Rev
Master Feedback	Direct Linear	-	Feedback mm or Feedback inch
Master Feedback	Rotary Transmission	-	Load Rev
Master Feedback	Linear Actuator	mm/r or mm	Load mm
Master Feedback	Linear Actuator	inch/r or inch	Load inch
Motor Feedback	Direct Rotary	-	Motor Rev
Motor Feedback	Direct Linear	-	Motor mm or Motor inch
Motor Feedback	Rotary Transmission	-	Load Rev
Motor Feedback	Linear Actuator	mm/r or mm	Load mm
Motor Feedback	Linear Actuator	inch/r or inch	Load inch
Load or Dual Feedback	Direct Rotary	-	Load Rev
Load or Dual Feedback	Direct Linear	-	Load mm or Motor inch
Load or Dual Feedback	Rotary Transmission	-	Load Rev
Load or Dual Feedback	Linear Actuator	mm/r or mm	Load mm
Load or Dual Feedback	Linear Actuator	inch/r or inch	Load inch

7.3.3.2.2 Attribute No. 79 – Motion Polarity

When Motion Scaling Configuration is set for Drive Scaling, Motion Polarity can be used to switch the directional sense of the motion control system. A Normal setting leaves the sign of the motion control command and actual signal values unchanged from their values in the drive control structure. An Inverted setting flips the sign of the command signal values to the drive control structure and flips the sign of the actual signal values coming from the drive control structure. Motion Polarity can therefore be used to adjust the sense of positive direction of the motion control system to agree with the positive direction on the machine. When the Motion Scaling Configuration is set to Drive Scaling, the Motion Polarity inversion is performed between the CIP Motion Connection interface and the drive control structure. When the

Motion Scaling Configuration is set to Controller Scaling, the Motion Polarity inversion is performed exclusively by the controller. To maintain directional consistency, the signs of all Signal Attribute values read from the drive control structure or being written to the drive control structure are determined by Motion Polarity. A comprehensive list of these Signal Attributes is defined in Table 41.

Table 41 – Signal attributes affected by Motion Polarity

Attr. ID	Access rule	Signal attribute name
1402 + o	Get	Feedback n Position
1403 + o	Get	Feedback n Velocity
1404 + o	Get	Feedback n Acceleration
62	Get	Registration 1 Positive Edge Position
63	Get	Registration 1 Negative Edge Position
64	Get	Registration 2 Positive Edge Position
65	Get	Registration 2 Negative Edge Position
70	Get	Home Event Position
360	Set*	Controller Position Command – Integer
361	Set*	Controller Position Command – Float
362	Set*	Controller Velocity Command
363	Set*	Controller Acceleration Command
364	Set*	Controller Torque Command
365	Get	Fine Command Position
366	Get	Fine Command Velocity
367	Get	Fine Command Acceleration
370	Set	Skip Speed 1
371	Set	Skip Speed 2
372	Set	Skip Speed 3
430	Get	Position Command
431	Set*	Position Trim
432	Get	Position Reference
433	Get	Velocity Feedforward Command
434	Get	Position Feedback
436	Get	Position Error
437	Get	Position Integrator Output
438	Get	Position Loop Output
450	Get	Velocity Command
451	Set*	Velocity Trim
452	Get	Acceleration Feedforward Command
453	Get	Velocity Reference
454	Get	Velocity Feedback
455	Get	Velocity Error
456	Get	Velocity Integrator Output
457	Get	Velocity Loop Output
480	Get	Acceleration Command
481	Set*	Acceleration Trim
482	Get	Acceleration Reference

Attr. ID	Access rule	Signal attribute name
483	Get	Acceleration Feedback
490	Get	Torque Command
491	Set*	Torque Trim
492	Get	Torque Reference
493	Get	Torque Reference – Filtered
494	Get	Torque Reference – Limited
520	Get	Iq Current Command
521	Get	Operative Current Limit
523	Get	Motor Electrical Angle
524	Get	Iq Current Reference
525	Get	Id Current Reference
527	Get	Iq Current Error
528	Get	Id Current Error
529	Get	Iq Current Feedback
530	Get	Id Current Feedback
531	Get	Vq Decoupling
532	Get	Vd Decoupling
533	Get	Vq Voltage Output
534	Get	Vd Voltage Output
535	Get	U Voltage Output
536	Get	V Voltage Output
537	Get	W Voltage Output
538	Get	U Current Feedback
539	Get	V Current Feedback
540	Get	W Current Feedback
541	Get	U Current Offset
542	Get	V Current Offset
543	Get	W Current Offset
565	Get	Slip Compensation
600	Get	Output Frequency
601	Get	Output Current
602	Get	Output Voltage
603	Get	Output Power

Motion Polarity can also have an impact on directional position, velocity, acceleration, and torque limit attributes. When the Motion Scaling Configuration is set to Drive Scaling, inverting Motion Polarity requires that positive and negative position, velocity, acceleration, and torque limit values be swapped between the CIP Motion Connection interface and the drive's internal control structure. When the Motion Scaling Configuration is set to Controller Scaling, inverting Motion Polarity requires that positive and negative position, velocity, acceleration, and torque limit attribute values in Motion Control Axis Object be swapped with the corresponding attributes in the Motion Device Axis Object. For example the Velocity Limit – Positive value in the controller would be mapped to the Velocity Limit – Negative value in the drive device. A comprehensive list of these Directional Limit Attributes is defined in Table 42.

Table 42 – Directional Limit attributes affected by Motion Polarity

Attr. ID	Access rule	Signal attribute name
374	Set	Ramp Velocity – Positive
375	Set	Ramp Velocity – Negative
376	Set	Ramp Acceleration
377	Set	Ramp Deceleration
448	Set	Position Limit – Positive
449	Set	Position Limit – Negative
473	Set	Velocity Limit – Positive
474	Set	Velocity Limit – Negative
485	Set	Acceleration Limit
486	Set	Deceleration Limit
504	Set	Torque Limit – Positive
505	Set	Torque Limit – Negative

7.3.4 Connection Data attributes

7.3.4.1 General

The attribute table in Table 43 contains connection data related attributes associated with a Motion Device Axis Object instance. These attributes are elements contained in the header of the CIP Motion Connection data structure that govern the format and interpretation of the connection data.

Table 43 – Connection Data attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
89	Optional – All	Set*		Control Status	BYTE	Bit mapped field that contains status bits from the controller. See semantics in 7.3.4.2.1	Bit Field: 0 = Configuration Complete 1 = (reserved) 2 = Converter Bus Up 3 = Converter Bus Unload 4 = Converter AC Power Loss 5 = (reserved) 6 = (reserved) 7 = (reserved)
90	Required – All	Set*		Actual Data Set	BYTE	Bit mapped field that determines what actual feedback data values are being transmitted to the controller in the Device-to-Controller Connection. See semantics in 7.3.4.2.1.	Bit Field: 0 = Position Feedback (R/C) 1 = Velocity Feedback (O/C) 2 = Accel Feedback (O) 3 = (reserved) 4 = (reserved) 5 = (reserved) 6 = Unwind Cycle Count [a] 7 = Position Displacement [a]
91	Required – D	Set*		Command Data Set	BYTE	Bit mapped field that determines what command data values are being updated by the controller. See semantics in 7.3.4.2.3.	Bit Field: 0 = Position Command (R/P) 1 = Velocity Command (R/V) 2 = Accel Command (O) 3 = Torque Command (R/T) 4 = (reserved) 5 = (reserved) 6 = Unwind Cycle Count [a] 7 = Position Displacement [a]
92	Required – PVT Optional – F	Set*		Command Control	BYTE	Controls the behavior of cyclic command data, including the format of the command position data and the interpolation or extrapolation method applied to position, velocity, and acceleration command data from the motion planner based on the associated time stamp.	See semantics in 7.3.4.2.4
93	Required – P Optional – FVT	Get		Command Target Time	LINT	Time Stamp associated with command data from the motion planner and used by the command fine interpolators. The Command Target Time is the sum of the Controller Time Stamp, the Device Time Stamp value, and the Command Target Update associated with this axis instance. The time stamp format is absolute and follows the CIP Sync standard with 0 corresponding to 1970-01-01.	Nanoseconds (CIP Sync absolute)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
94	Required – All	Set*		Status Data Set	BYTE	Bit mapped field that determines what status data values are to be transmitted to the controller via the Device-to-Controller Connection.	See semantics in 7.3.4.2.5
95	Required – Scaling E	Set*		Registration Data Set	BYTE	Bit mapped field that determines what Registration Event Data values are being updated by the controller over the Controller-to-Device Connection. Windowed Registration functionality is only applicable when Scaling Configuration is set to Drive Scaling. The format of the Registration Event Data is defined in 7.3.4.2.6	Bit Field: 0 = Reg 1 Pos Edge Window Position (O) 1 = Reg 1 Neg Edge Window Position (O) 2 = Reg 1 Pos Edge Window Position (O) 3 = Reg 1 Neg Edge Window Position (O) 4 = (reserved) 5 = (reserved) 6 = (reserved) 7 = (reserved)
96	Required – Scaling E	Set*		Home Data Set	BYTE	Bit mapped field that determines what Home Event Data values are being updated by the controller over the Controller-to-Device Connection. The format of the Home Event Data is defined in 7.3.4.2.6.	Bit Field: 0 = Home Torque Threshold (O) 1 = Home Torque Time (O) 2 = (reserved) 3 = (reserved) 4 = (reserved) 5 = (reserved) 6 = (reserved) 7 = (reserved)
97	Required – Scaling E	Set*		Watch Data Set	BYTE	Bit mapped field that determines what Watch Event Data values are being updated by the controller over the Controller-to-Device Connection. Watch Position functionality is only applicable when Scaling Configuration is set to Drive Scaling. The format of the Watch Event Data is defined in 7.3.4.2.6.	Bit Field: 0 = Watch 1 Position (R) 1 = Watch 2 Position (O) 2 = (reserved) 3 = (reserved) 4 = (reserved) 5 = (reserved) 6 = (reserved) 7 = (reserved)

[a] Required in E with Drive Scaling.

7.3.4.2 Semantics

7.3.4.2.1 Attribute No. 89 – Control Status

The individual bits in the Control Status Bit Field are used as indicated below.

Configuration Complete: This bit is set when the controller has completed configuration of all axis instance attributes during Initialization phase.

Converter Bus Up: This bit, when applicable, is set by the controller when all associated Converters, or CIP Motion drives with integral converters, or CIP Motion drives with external non-CIP converters, supplying DC Bus power to this device have indicated to the controller that DC Bus voltage has reached an operational level as indicated by the Converter(s) setting of the Axis Status bit, Bus Up. This bit is only applicable to drives that support DC Bus Sharing functionality and the ability to qualify the DC Bus Up status of the drive based on this bit.

Converter Bus Unload: This bit, when applicable, is set by the controller when an associated Converter, or CIP Motion drive containing an integral converter, or CIP Motion drive connected to an external non-CIP converter, supplying DC Bus power to this device has requested that this device stops drawing DC Bus power. When the Converter Bus Unload bit is set, the device shall generate a Bus Power Sharing exception if the device's power structure is enabled and drawing DC Bus power. This bit is only applicable to drives that support DC Bus Sharing functionality and have the ability to generate a Bus Power Sharing exception based on this bit.

Converter AC Power Loss: This bit, when applicable, is set by the controller when an associated Converter, or CIP Motion drive containing an integral converter, or CIP Motion drive connected to an external non-CIP converter, has detected a loss of AC input power. When the Converter AC Power Loss bit is set, the device shall generate a Converter AC Power Loss exception if the device's power structure is enabled. This bit is cleared when the controller has determined that all associated Converter(s) supplying DC Bus power to this device have sufficient AC input power for converter operation. This bit is only applicable to drives that support DC Bus Sharing functionality and have the ability to generate a Converter AC Power Loss exception based on this bit.

7.3.4.2.2 Attribute No. 90 – Actual Data Set

Generally, the Actual Data Set value is determined by the operative Control Mode as defined in Table 44.

Table 44 – Actual Data Set value determination

Bit	Actual Data	N	F	P	V	T
0	Position	X		X	()[a]	()[a]
1	Velocity		X		()[a]	()[a]
2	Acceleration					
3	Torque					X

[a] Position Feedback is selected when feedback device is present, Velocity Feedback is selected when configured for sensorless or encoderless operation.

7.3.4.2.3 Attribute No. 91 – Command Data Set

Generally, the Command Data Set value is determined by the operative Control Mode as defined in Table 45.

Table 45 – Command Data Set value determination

Bit	Cmd Data	N	F	P	V	T
0	Position			()[a]		
1	Velocity		X		X	
2	Acceleration					
3	Torque					X

[a] Controller sends either Float or Integer Command Position data, but never both. Integer is only sent when drive does not support double precision floating point position command data.

7.3.4.2.4 Attribute No. 92 – Command Control

The Command Control attribute governs the interpolation/extrapolation method applied to position, velocity, and acceleration command data from the motion planner based on the associated time stamp, as specified in Figure 61.

← 8-bit BYTE →		
Command Control Word format		
Bits 4-7: (Reserved)	Bits 2-3: Enumeration Command Position Data Type	Bits 0-1: Enumeration Command Target Update

IEC

Figure 61 – Command Control Word field

The first two bits of this 8-bit attribute represent the Command Target Update that determines the number of Update Periods (UP) added to the command data Time Stamp to determine the absolute System Time that the command data value is targeted for, i.e. the Command Target Time (see Table 46). For more details on interpolation and extrapolation control, see 7.6.6.2.

Table 46 – Command Target Update enumeration definition

Enum.	Req./Opt.	Name	Description
0	R/P	Immediate	A Command Target Update of 0 implies that the command data is to be applied to the device control structure immediately.
1	R/P	Extrapolate (+1 UP)	A Command Target Update of 1 implies that extrapolation is to be used to apply the command data to the device control structure based on adding 1 Update Period to the command Time Stamp.
2	R/P	Interpolate (+2 UP)	A Command Target Update of 2 implies that fine interpolation is to be used to apply the command data based on adding 2 Update Periods to the command Time Stamp.
3		(Reserved)	-

Table 47 specifies the values for the Command Position Data Type field.

Table 47 – Command Position Data Type enumeration definition

Enum.	Req./Opt.	Name	Description
0	O/P [a]	LREAL (64-bit Float)	A Command Position Data Type of 0 corresponds to the LREAL, or double precision floating point data type. Support for this data type is preferred in the implementation since it provides fraction command count information to the drive resulting in smoother motion.
1	O/P [a]	DINT (32-bit Integer)	A Command Position Data Type of 1 corresponds to the DINT, or 32-bit signed integer data type. This data type is applicable to simple drive devices that either do not require the precision of a floating point command position value or do not have sufficient hardware support for double precision floating point math.
3-4		(Reserved)	-

[a] If axis is configured for Position Control Mode operation, at least one of the defined Command Position Data Types shall be supported by the device.

7.3.4.2.5 Attribute No. 94 – Status Data Set

The Status Data Set attribute is an 8-bit collection of bits indicating which Status attributes are to be transmitted to the controller over the Device-to-Controller Connection (see Table 48). Status data appears in the same order as the bit numbers, so Axis Fault Type/Code data would appear before Axis Fault Time Stamp data in the Status Data Set structure. Multiple attributes comprising a selected Status Data Element are transmitted in the order listed from top to bottom, so the Axis Fault Type is transmitted before Axis Fault Code.

Table 48 – Status Data Set bit definitions

Bit	Status Data Element produced	Data type
0	Axis Fault Type Axis Fault Code Axis Fault Sub Code Axis Fault Action Axis Fault Time Stamp	USINT USINT USINT USINT LINT
1	Axis Alarm Type Axis Alarm Code Axis Alarm Sub Code Axis Alarm State Axis Alarm Time Stamp	USINT USINT USINT USINT LINT
2	Axis Status Axis Status – Mfg	DWORD DWORD
3	Axis I/O Status Axis I/O Status – Mfg	DWORD DWORD
4	Axis Safety Status Axis Safety Status – Mfg Axis Safety State Pad[3]	DWORD DWORD USINT USINT[3]
5	(Reserved)	(Reserved)
6	(Vendor Specific)	(Vendor Specific)
7	(Vendor Specific)	(Vendor Specific)

7.3.4.2.6 Attribute No. 95 to 97 – Registration, Home and Watch Event Data format

The format of the Registration Event Data is defined Table 49.

Table 49 – Registration Event Data format

Bit	Reg Data	Format
0	Reg 1 Pos Window	DINT (max) DINT (min)
1	Reg 1 Neg Window	DINT (max) DINT (min)
2	Reg 2 Pos Window	DINT (max) DINT (min)
3	Reg 2 Neg Window	DINT (max) DINT (min)

The format of the Home Event Data is defined in Table 50.

Table 50 – Home Event Data format

Bit	Home Data	Format
0	Home Torque Threshold	REAL
1	Home Torque Time	REAL

The format of the Watch Event Data is defined in Table 51.

Table 51 – Watch Event Data format

Bit	Home Data	Format
0	Watch 1 Position	DINT
1	Watch 2 Position	DINT

7.3.5 Motor attributes

7.3.5.1 General

The following attribute tables contain motor configuration attributes associated with a Motion Device Axis Object instance that apply to various motor technologies. These motor technologies include three-phase motor rotary, linear, permanent magnet and induction motors. Motor attributes are, therefore, organised according to the various motor types. The Need in Implementation category for an attribute is based on the context of the Motor Type and, thus, to the context of the table in which the attribute appears. Where needed, Standard vs. Optional can be further differentiated by abbreviations for PM (Permanent Magnet) and IM (Induction Motors). Within the PM Motor family, there is further differentiation for SPM (Surface PM) and IPM (Interior PM) motors. It is implied that these motor attributes are applicable to all drive modes, F, P, V and T, but not applicable for N, or No Control axis configurations where there is no active motor control function.

The goal of 7.3.5 is to define the minimal set of required attributes to support CIP Motion device interchangeability. This guarantees that there is sufficient parametric data provided by the controller for any CIP Motion compliant device, i.e. drive, to effectively control a given motor.

For induction motors, the Motion Device Axis Object leverages the IEEE Std 112 recommended phase-neutral equivalent circuit motor model based on "Wye" configuration (see Figure 62). Reactance values, X, are related to their corresponding Inductance values, L, by $X = \omega \times L$, where ω is the rated frequency of the motor. The prime notation, for example X_2', R_2', indicates that the actual rotor component values X_2, and R_2 are referenced to the stator side of the stator-to-rotor winding ratio.

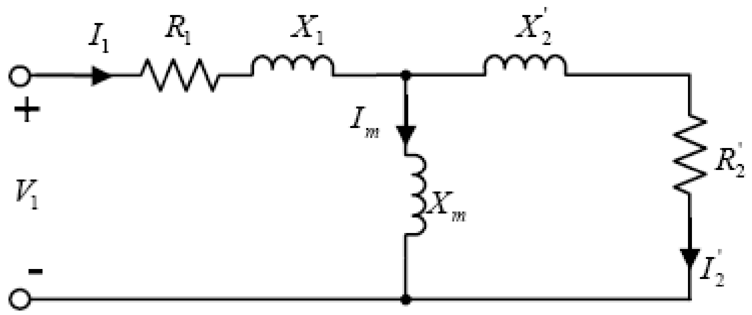

Figure 62 – IEEE Std 112 per phase motor model

For Permanent Magnet motors, the Motion Device Axis Object assumes all motor parameters are defined in the context of a phase-to-phase motor model.

7.3.5.2 General motor attributes

The attribute tables in Table 52 and Table 53 contain general motor attributes that apply to all motor technologies.

Table 52 – General Motor Info attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1310	Optional	Get		Motor Catalog Number	SHORT STRING	A 32-character string that specifies the motor catalog number. If the Motor Catalog Number is not available, the drive sets this attribute to a Null string.	For example MPL-B310F
1311	Optional	Get		Motor Serial Number	SHORT STRING	A 16-character string that specifies the serial number of the motor. If the Motor Serial Number is not available, the drive sets this attribute to a Null string.	For example 0012003400560078
1312	Optional (Motor NV)	Get		Motor Date Code	SHORT STRING	A 16-character string that specifies the manufacturing date of the motor. If the Motor Date Code is not available, the drive sets this attribute to a Null string.	For example Jan-01-2005

Table 53 – General Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1313	Required	Set		Motor Data Source	USINT	An enumeration that specifies the source of motor data for the drive. Controller NV implies that the motor attributes are derived from the controller's non-volatile memory during the drive configuration process. Drive NV implies that the motor attributes are derived directly from the drive's non-volatile memory. In this mode, only a minimal set of motor and motor feedback (Feedback 1) are required to configure the drive. Motor NV implies that the motor attributes are derived from non-volatile memory of a motor-mounted smart feedback device equipped with a serial interface. Again, in this mode, only a minimal set of motor and motor feedback (Feedback 1) are required to configure the drive. Motor and motor feedback attributes sent to the drive device in Drive NV or Motor NV are merely to confirm that the controller and the drive have the agreement on the values of attributes critical to scaling operation. If the NV attribute values in the drive differ from the values set by the controller, the drive shall reject the set service with General Status indicating an Invalid Attribute Value. The current list of motor and motor feedback attributes sent to the drive in the NV modes are as follows: 1. Motor Unit 2. Feedback 1 Unit 3. Feedback 1 Type 4. Feedback 1 Startup Method 5. Feedback 1 Cycle Resolution 6. Feedback 1 Cycle Interpolation 7. Feedback 1 Turns 8. Feedback 1 Length	Bits 0 to 3: Enum 0 = Controller NV (R) 1 = Drive NV (O) 2 = Motor NV (O) 3 to 127 = (reserved) 128 to 255 = (vendor specific)
1314	Required	Set		Motor Device Code	UDINT	A unique number assigned to a motor catalogue. This value is used to ensure that the motor and integral motor mounted feedback device configuration data delivered from the controller matches the actual motor and feedback data connected to the drive. This comparison is only valid in the case where the Motor Data Source is Controller NV and the motor is equipped with a smart feedback device that positively identifies the motor. If the codes do not match, a negative acknowledge is given by the drive. Motor Device Codes are assigned by the motor manufacturer. A value of 0 for the Motor Device Code shall be accepted by the drive without comparison.	

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1315	Required	Set		Motor Type	USINT	An enumeration that specifies the motor technology. If Motor Data Source is Motor NV or Drive NV, the Motor Type may not be know to the controller but is known by the drive, so the drive can operate in this case without the controller specifying the Motor Type. In this mode, Motor Type shall not be sent by the controller. If received, it shall be rejected with General Status indicating an Invalid Attribute Value. If Motor Data Source is Datasheet or Database, an unspecified Motor Type, when received by the drive device during configuration, indicates that the motor configuration has not been defined and therefore shall be rejected with General Status indicating an Invalid Attribute Value.	Enumeration: 0 = not specified (R) 1 = rotary permanent magnet (O) 2 = rotary induction (O) 3 = linear permanent magnet (O) 4 = linear induction (O) 5 to 127 = (reserved) 128 to 255 = (vendor specific)
1316	Required	Set		Motor Unit	USINT	Unit of measure for motor displacement. This attribute is also used for encoderless or sensorless operation since the Feedback Unit in that case is not applicable.	Enumeration: 0 = Rev (R for rotary motors) 1 = Meter (R for linear motors) 2 to 127 = (reserved) 128 to 255 = (vendor specific)
1317	Optional	Set		Motor Polarity	USINT	An enumerated value used to establish the direction of motor motion when the windings are phased according to factory specification. Normal polarity is defined as the direction of motor travel when the ABC motor winding leads are hooked up according to the drives published specifications. Inverted polarity effectively switches the ABC phasing to ACB so that the motor moves in the opposite direction in response to a positive drive output. This attribute can be used to make the direction of travel agree with the user's definition of positive travel and can be used in conjunction with the Feedback Polarity bit to provide negative feedback, when closed loop control is required. When commutating a PM motor it is imperative that the commutation phase sequencing match the motor phase sequencing to properly control the motor.	Enumeration: 0 = Normal Polarity 1 = Inverted Polarity 2 to 255 = (reserved)
1318	Required	Set		Motor Rated Voltage	REAL	A float that specifies the nameplate AC voltage rating of the motor. This represents the phase-to-phase voltage applied to the motor to reach rated speed at full load.	Volts (RMS)
1319	Required	Set		Motor Rated Continuous Current	REAL	A float that specifies the nameplate AC continuous current rating of the motor. This represents the current applied to the motor under full load conditions at rated speed and voltage.	Amperes (RMS)
1320	Required – PM Optional – IM	Set		Motor Rated Peak Current	REAL	A float that specifies the peak or intermittent current rating of the motor. The peak current rating of the motor is often determined by either the thermal constraints of the stator winding or the saturation limits of PM motor magnetic material.	Amperes (RMS)
1321	Required – IM Optional – PM	Set		Motor Rated Output Power	REAL	A float that specifies the nameplate rated output power rating of the motor. This represents the power output of motor under full load conditions at rated current, speed and voltage.	Kilowatts

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1322	Optional	Set		Motor Overload Limit	REAL	A float that specifies the maximum thermal overload limit for the motor. This value is typically 100 %, corresponding to the power dissipated when operating at the continuous current rating of the motor, but can be significantly higher if cooling options are applied. For induction motors, this attribute is related to the Service Factor of the motor. When the Motor Capacity attribute value associated with the motor thermal model exceeds the Motor Overload Limit, the drive can optionally trigger a predetermined Motor Overload action. The Motor Overload Limit can also used by the drive to determine the absolute thermal capacity limit of the motor, i.e. the Motor Thermal Overload Factory Limit, that if exceeded, generates a Motor Thermal Overload FL exception.	% Motor Rated
1323	Optional	Set		Motor Integral Thermal Switch	BOOL	A boolean that specifies if the motor has an integral thermal switch.	0 = No 1 = Yes
1324	Optional	Set		Motor Max Winding Temperature	REAL	A float that specifies the maximum winding temperature of the motor.	°C
1325	Optional	Set		Motor Winding-to-Ambient Capacitance	REAL	A float that specifies the winding-to-ambient thermal capacitance.	J/°C
1326	Optional	Set		Motor Winding-to-Ambient Resistance	REAL	A float that specifies the winding-to-ambient thermal resistance.	°C/W

7.3.5.3 General PM motor attributes

The attribute table in Table 54 contains motor configuration attributes that apply to Permanent Magnet motor types in general.

Table 54 – General PM Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1327	Required	Set		PM Motor Resistance	REAL	A float that specifies the phase-to-phase, resistance of a permanent magnet motor.	Ohms
1328	Required (SPM only)	Set		PM Motor Inductance	REAL	A float that specifies the phase-to-phase, inductance of a permanent magnet motor.	Henries
1353	Required (IPM only)	Set		PM Motor Lq Inductance	REAL	A float that specifies the phase-to-phase, q-axis, inductance of an interior permanent magnet motor.	Henries
1354	Required (IPM only)	Set		PM Motor Ld Inductance	REAL	A float that specifies the phase-to-phase, d-axis, inductance of an interior permanent magnet motor.	Henries

7.3.5.4 General rotary motor attributes

The attribute table in Table 55 contains motor configuration attributes that apply specifically to rotary motor types.

Table 55 – General Rotary Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1329	Required	Set		Rotary Motor Poles	UINT	An integer that specifies the number of poles per revolution for rotary motors. This value is always an even number, as poles always exist in pairs.	
1330	Optional	Set		Rotary Motor Inertia	REAL	A float that specifies the unloaded inertia of a rotary motor.	kg•m^2
1331	Required	Set		Rotary Motor Rated Speed	REAL	A float that specifies the nameplate rated speed of a rotary motor. For PM motors, this is generally specified at rated voltage based on either rated current, rated torque, or rated power. For induction motors this value is the speed of the motor driven at rated frequency under rated torque load. This value is synonymous with the term base speed.	r/min
1332	Optional	Set		Rotary Motor Max Speed	REAL	A float that specifies the absolute maximum operating speed of a rotary motor in units of r/min. This speed may be determined by the limitations of the motor or by limitations of the mechanical system. Specifically, this value can represent the maximum safe operating speed, maximum continuous “no-load” speed, maximum continuous encoder speed, or maximum continuous bearing speed of the motor. This value can be used by the drive to determine the Rotary Motor Overspeed Factory Limit.	r/min
1333	Optional	Set		Rotary Motor Damping Coefficient	REAL	A float that specifies the damping, or viscous friction, associated with a rotary motor.	N•m/ (rad/s)

7.3.5.5 General linear motor attributes

The attribute table in Table 56 contains motor configuration attributes that apply specifically to linear motor types.

Table 56 – General Linear Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1334	Required	Set		Linear Motor Pole Pitch	REAL	A float that specifies the pole pitch of a linear motor in units of meters, and is equivalent to the electrical cycle length.	m
1335	Required	Set		Linear Motor Rated Speed	REAL	A float that specifies the nameplate rated speed of a linear motor. For PM motors, this is generally specified at rated voltage based on either rated current, rated force, or rated power. For induction motors this value is the speed of the motor driven at rated frequency under rated force load. This value is synonymous with the term base speed.	m/s
1336	Optional	Set		Linear Motor Mass	REAL	A float that specifies the unloaded moving mass of a linear motor.	kg
1337	Optional	Set		Linear Motor Max Speed	REAL	A float that specifies the absolute maximum operating speed of a linear motor in units of m/s. This speed may be determined by the limitations of the motor or by limitations of the mechanical system. Specifically, this value can represent the maximum safe operating speed, maximum continuous “no-load” speed, maximum continuous encoder speed, or maximum continuous bearing speed of the motor. This value can be used by the drive to determine the Linear Motor Overspeed Factory Limit.	m/s
1338	Optional	Set		Linear Motor Damping Coefficient	REAL	A float that specifies the damping, or viscous friction, associated with a linear motor.	N/(m/s)

7.3.5.6 Rotary PM motor attributes

The attribute table in Table 57 contains motor configuration attributes that apply specifically to rotary motor types.

Table 57 – Rotary PM Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1339	Optional	Set		PM Motor Rated Torque	REAL	A float that specifies the nameplate continuous torque rating of a rotary permanent magnet motor.	N•m
1340	Optional	Set		PM Motor Torque Constant	REAL	A float that specifies the torque constant of a rotary permanent magnet motor in newton meters per RMS ampere.	N•m/ A (RMS)
1341	Required	Set		PM Motor Rotary Voltage Constant	REAL	A float that specifies the voltage, or back-EMF, constant of a rotary permanent magnet motor in phase-to-phase RMS Volts per kr/min. If the optional PM Motor Torque Constant, *Kt*, is not explicitly supported in the implementation, the value may be computed from the PM Motor Rotary Voltage Constant, *Ke*, according to the following equation: *Kt* (N•m/A) = 0,016 54 × *Ke* (V/kr/min)	Volts (RMS)/ kr/min

7.3.5.7 Linear PM motor attributes

The attribute table in Table 58 contains motor configuration attributes that apply specifically to linear PM motor types.

Table 58 – Linear PM Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1342	Optional	Set		PM Motor Rated Force	REAL	A float that specifies the nameplate continuous force rating of a linear permanent magnet motor.	N
1343	Optional	Set		PM Motor Force Constant	REAL	A float that specifies the force constant of a linear permanent magnet motor in Newtons per RMS ampere.	N/A (RMS)
1344	Required	Set		PM Motor Linear Voltage Constant	REAL	A float that specifies the voltage, or back-EMF, constant of a linear permanent magnet motor in phase-to-phase RMS Volts per m/s. If the optional PM Motor Force Constant, *Kf*, is not explicitly supported in the implementation, the value may be computed from the PM Motor Linear Voltage Constant, *Ke*, according to the following equation: *Kf* (N/A) = 1,732 × *Ke* (V/(m/s))	Volts (RMS) / (m/s)

7.3.5.8 Induction motor attributes

The attribute table in Table 59 contains motor configuration attributes that apply specifically to induction motor types.

Table 59 – Induction Motor Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1345	Required	Set		Induction Motor Rated Frequency	REAL	A float that specifies the nameplate frequency rating of an induction motor.	Hertz
1346	Required	Set		Induction Motor Flux Current	REAL	Id Current Reference that required to generate full motor flux. This value is closely approximated by the No Load Motor Rated Current commonly found in Induction Motor data sheets.	Amperes (RMS)
1347	Required	Set		Induction Motor Stator Resistance	REAL	A float that specifies the Y circuit, phase-neutral, winding resistance of the stator as shown as R1 in the IEEE Std 112 motor model.	Ohms
1348	Required	Set		Induction Motor Stator Leakage Reactance	REAL	A float that specifies the Y circuit, phase-neutral, leakage reactance of the stator winding, at rated frequency, as shown as X_1 in the IEEE Std 112 motor model.	Ohms
1349	Optional [a]	Set		Induction Motor Magnetization Reactance	REAL	A float that specifies the Y circuit, phase-neutral, magnetizing reactance of the motor, at rated frequency, as shown as Xm in the IEEE Std 112 motor model.	Ohms
1350	Optional [a]	Set		Induction Motor Rotor Resistance	REAL	A float that specifies the phase-neutral equivalent stator-referenced winding resistance of the rotor as shown as R_2’ in the IEEE Std 112 motor model.	Ohms
1351	Required	Set		Induction Motor Rotor Leakage Reactance	REAL	A float that specifies the Y circuit, phase-neutral, equivalent stator-referenced leakage inductance of the rotor winding, at rated frequency, as shown as X_2’ in the IEEE Std 112 motor model.	Ohms
1352	Optional – D	Set		Induction Motor Rated Slip Speed	REAL	Represents the amount of slip at motor rated current (full load) and motor rated frequency.	r/min (rotary motor) m/s (linear motor)

[a] These parameters have a strong motor temperature component that some drives circumvent through various adaption or compensation techniques.

7.3.5.9 Load transmission and actuator attributes

The attribute table in Table 60 contains motor configuration attributes that apply specifically to rotary transmission and linear actuator mechanisms associated with the axis.

Table 60 – Load Transmission and Actuator Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1370	Optional – Scaling	Set		Load Type	USINT	Enumerated attribute used to determine how the load is mechanically linked to the motor. “Direct” enumerations indicate that the motor is directly coupled to the load. “Rotary” enumerations indicate that the load is rotating and load dynamics are measured using a rotary system of units. “Linear” enumerations indicate that the load is moving linearly and load dynamics are measured using a linear system of units.	Enumeration: 0 = Direct Rotary 1 = Direct Linear 2 = Rotary Transmission 3 = Linear Actuator 4 to 255 = reserved
1371	Optional – Scaling	Set		Transmission Ratio Input	DINT	Integer number of input shaft revolutions per transmission cycle associated with the rotary transmission.	Input Shaft Revs
1372	Optional – Scaling	Set		Transmission Ratio Output	DINT	Integer number of output shaft revolutions per transmission associated with the rotary transmission.	Output Shaft Revs
1373	Optional – Scaling	Set		Actuator Type	USINT	This enumerated value indicates the type of mechanism used for linear actuation.	Enumeration: 0 = None (R) 1 = Screw (O) 2 = Belt and Pulley (O) 3 = Chain and Sprocket (O) 4 = Rack and Pinion (O) 5 to 255 = reserved
1374	Optional – Scaling NP only	Set		Actuator Lead	REAL	This floating point value represents the lead or pitch of a screw actuator that is a measure of the linear movement of the screw mechanism per revolution of the screw shaft.	Actuator Lead Units
1375	Optional – Scaling NP only	Set		Actuator Lead Unit	USINT	This enumerated value indicates the units of the Actuator Lead attribute.	Enumeration: 0 = mm/r 1 = inch/r 2 to 255 = reserved
1376	Optional – Scaling	Set		Actuator Diameter	REAL	This floating point value represents the diameter of the pulley, sprocket, or pinion used to convert rotary motion into tangential linear displacement of the load. The Actuator Diameter is internally converted to circumference of the pulley, sprocket, or pinion to determine the amount of tangential displacement per revolution.	Actuator Diameter Units
1377	Optional-Scaling	Set		Actuator Diameter Unit	USINT	This enumerated value indicates the units of the Actuator Diameter attribute.	Enumeration: 0 = mm 1 = inch 2 to 255 = reserved

7.3.6 Feedback attributes

7.3.6.1 General

The attribute tables in Table 64, Table 65 and Table 66 contain all position feedback related attributes associated with a Motion Device Axis Object instance that apply to various feedback device and feedback interface technologies.

NOTE These feedback interface technologies include Digital AqB (digital A quad B signals), Sine/Cosine (analog A quad B signals), Digital Parallel (parallel digital bit interface), SSI (Synchronous Serial Interface), LDT (Linear Displacement Transducer) and Resolver. Other modern feedback interfaces supported are Hiperface® (by Stegmann) and EnDat 2.1® & EnDat 2.2® (by Heidenhain).

The “Need in Implementation” specification for a feedback attribute is often based on the context of the Feedback Type. To facilitate this, abbreviations for the various Feedback Types are defined in Table 61.

Table 61 – Feedback Types abbreviations

Abbreviation	Feedback type
TT	Digital AqB
TP	Digital Parallel
SC	Sine/Cosine
HI	Hiperface®
E21	EnDat 2.1®
E22	EnDat 2.2®
RS	Resolver
SS	SSI
LT	LDT

The goal of 7.3.6 is to define the minimal set of required attributes to support CIP Motion device interchangeability. This guarantees that there is sufficient parametric data provided by the controller for any CIP Motion compliant drive to effectively interface to a wide range of feedback device types.

Multiple feedback device interfaces are currently defined by the Motion Device Axis Object per axis instance to serve specific control or master feedback functions. These feedback devices are accessed via their assigned logical channels, for example, Feedback 1 and Feedback 2. Each logical feedback channel is mapped to a physical feedback interface port of the device, for example Port 1, and Port 2.

Table 62 lists Logical Feedback Channel Control functions.

Table 62 – Logical Feedback Channel Control functions

Logical Feedback Channel	Motion Control function	Master Feedback function
Feedback 1	Motor Feedback & Commutation	Master Feedback 1
Feedback 2	Load-side Feedback	Master Feedback 2

When the Control Mode is set to something other than No Control, Feedback 1 is generally associated with the motor mounted feedback device while Feedback 2 is associated with the load-side or machine mounted feedback device. Feedback 1 is always required for Permanent Magnet motor commutation.

When Control Mode is set to No Control for a Motion Device Axis Object instance, different logical feedback channels can be used as a master feedback source, for example Feedback 1, Feedback 2, etc. Generally, Feedback 1 is used.

To minimize the length of the feedback attribute tables, the letter “n” in the generic “Feedback n” attribute name is used to specify the associated feedback channel number. Valid channel numbers for open standard feedback attributes of the Motion Device Axis Object are 1, 2, 3, and 4. Attribute IDs are assigned based on the channel number. Support for feedback interface channels 1, 2, 3, and 4 are optional in the device implementation. If no feedback interface channel is present in the device, the associated set of feedback channel attributes are not applicable. However, if hardware support for any of these feedback channels is available in a given device, these attributes are clearly applicable in the implementation and shall follow the Need in Implementation rules. An implementation rule of “Req – E” or “Opt – E” indicates that the attribute is generally applicable to all Device Function Codes where the feedback channel itself is applicable, hence the “E” for "Encoder". If a specific logical feedback channel is not applicable based on the current Feedback Mode, then attributes for feedback n are not applicable; no feedback configuration attributes for that channel are set by configuration software, nor are any such attributes sent to the drive device.

Table 63 outlines these rules.

Table 63 – Logical Feedback Channel rules

Feedback Mode	Feedback 1	Feedback 2
No Feedback	No	No
Master Feedback	Yes	No [a]
Motor Feedback	Yes	No
Load Feedback	Yes [b]	Yes
Dual Feedback	Yes	Yes

[a] Feedback 2 channel only needed if Feedback Master Select supports Feedback 2 channel option.

[b] Feedback 1 channel needed for commutation of PM Motors.

Table 64, Table 65 and Table 66 specify feedback related attributes.

Table 64 – General Feedback Info attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1400 + (n-1)x50	Optional – E	Get		Feedback n Catalog Number	SHORT STRING	A 32-character string that specifies the catalog number of the device associated with Feedback n. If the Feedback Catalog Number is not available, the drive sets this attribute to a Null string.	for example SRM-50
1401 + (n-1)x50	Optional – E	Get		Feedback n Serial Number	SHORT STRING	A 16-character string that specifies the serial number of the device associated with Feedback n. If the Feedback Serial Number is not available, the drive sets this attribute to a Null string.	for example 001200340 0560078

Table 65 – General Feedback Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1402 + (n-1)x50	Required – E	Get		Feedback n Position	DINT	Actual position of the axis based on Feedback n.	Feedback n Counts
1403 + (n-1)x50	Required – E	Get		Feedback n Velocity	REAL	Actual filtered velocity of the axis based on Feedback n.	Feedback n Units/s
1404 + (n-1)x50	Required – E	Get		Feedback n Acceleratio n	REAL	Actual filtered acceleration of the axis based on Feedback n.	Feedback n Units/s^2

Table 66 – Feedback Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
42	Required – All	Set*		Feedback Mode	BYTE	This attribute determines how the various available feedback channels are used to implement the selected Control Mode. Currently only bits 0 to 3 are used to enumerate the Feedback Mode configuration. Bit 4 to 7 are reserved for future use.	Bits 0 to 3: Feedback Mode Enumeration: 0 = No Feedback 1 = Master Feedback 2 = Motor Feedback 3 = Load Feedback 4 = Dual Feedback 5 to 7 = (Reserved) 8 to 15 = (Vendor Specific) Bits 4 to 7: Reserved See semantics in 7.3.6.2
43	Optional – N	Set		Feedback Master Select	USINT	This attribute determines what Logical channel is assigned to this axis instance when the Feedback Mode is set to Master Feedback. Default is Feedback 1.	Enumeration: 0 = (Reserved) 1 = Feedback 1 2 = Feedback 2 2 to 255 = (reserved)
44	Optional – PV	Set		Feedback Unit Ratio	REAL	Number of Feedback 1 Units per Feedback 2 Units. This value is used to convert from feedback 2 units or counts to feedback 1 units or counts when configured for dual loop or load side feedback operation. The Feedback Ratio block that applies the Feedback Unit Ratio scaling factor appears in the Position Loop and Torque Reference block diagrams.	Feedback 1 Units per Feedback 2 Units

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1410 + (n-1)×50	Optional – E	Set		Feedback n Resolution Unit	USINT	Unit of measure for feedback resolution used by Feedback n Cycle Resolution attribute. Default selection is Cycles/Unit where resolution is expressed in feedback cycles per revolution for rotary feedback devices or per meter for linear feedback devices. If Units/Cycle is selected then Feedback n Cycle Resolution shall be expressed in Nanometers/Cycle for linear feedback devices. This selection is not applicable to rotary devices. If Bits/Unit is selected then Feedback n Cycle Resolution is expressed as 2n Cycles per revolution of a rotary feedback device, where n is the number of bits in the binary position representation of the device. This selection is not applicable for linear devices.	Enumeration: 0 = Cycles/Unit (R) 1 = Units/Cycle (O) (linear only) 2 = Bits/Unit (O) (rotary only) 3 to 127 = (reserved) 128 to 255 = (vendor specific)
1411 + (n-1)×50	Required -E	Set		Feedback n Unit	USINT	Unit of measure for the designated feedback device. The Feedback Unit for Feedback 1 shall be the same as the configured Motor Unit; if the Motor Unit is set to Rev, Feedback 1 Unit shall be set to Rev; if Motor Unit is set to Meter, Feedback 1 Unit shall be set to Meter. Feedback devices with a Feedback Unit of Rev are considered “rotary” devices, while Feedback devices with a Feedback Unit of Meter are considered “linear” devices	Enumeration: 0 = Rev 1 = Meter 2 to 127 = (reserved) 128 to 255 = (vendor specific)
1412 + (n-1)×50	Optional – E	Set		Feedback n Port Select	USINT	Maps the logical Feedback Channel “n” to a physical Feedback Port ID. A Feedback Port ID is assigned to each feedback interface port of the device by the drive vendor. If this attribute is not supported by the drive, the drive vendor shall hardcode the feedback port mapping to the logical Feedback Channels for each axis instance. Supporting the Feedback n Port Select attribute allows flexibility to map the logical Feedback Channels to different Feedback Ports. Default Feedback n Port Select = 0. A value of 0 indicates that Feedback n Channel is unmapped, hence unused.	Enumeration: 0 = Unused (R) 1 to 255 = Feedback Port ID (O)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1413 + (n-1)×50	Required – E	Set		Feedback n Type	USINT	Identifies the type of feedback interface of the associated Feedback device channel. Drive support for any individual feedback type is left to the discretion of the device manufacturer. However if a specific feedback type is supported, attributes associated with that type are generally required in the implementation. The “Integrated” Feedback Type is specified for CIP Motion compliant devices with an integral feedback transducer function, e.g. a CIP Motion encoder. In the case of a motor mounted feedback device, if Motor Data Source is Motor NV or Drive NV, the Feedback 1 Type may not be known to the controller but is known by the drive, so the drive can operate in this case without specifying the Feedback 1 Type. If the optional Commutation Startup Method attribute is not supported by the device, or the Commutation Startup Method is set to From Feedback Type, the Feedback 1 Type can be used to implicitly specify the commutation startup method. For example, by selecting the Feedback 1 Type with or without UVW commutation signals the device applies the UVW commutation startup method or the Self-Sense startup method, respectively. In this case, UVW commutation signals can be derived from UVW tracks integral to the feedback device or via separate Hall sensors in the motor. All other Feedback 1 Type selections would apply the Digital commutation startup method. In the case of a motor mounted feedback device, if the Motor Data Source is Datasheet or Database, an unspecified Feedback 1 Type, when received by the drive device during configuration, indicates that the motor feedback configuration has not been defined and therefore results in a General Status error of Invalid Attribute Value.	Enumeration: 0 = Not Specified (R) 1 = Digital AqB (O) 2 = Digital AqB with UVW (O) 3 = Digital Parallel (O) 4 = Sine/Cosine (O) 5 = Sine/Cosine with UVW (O) 6 = Hiperface (O) 7 = EnDat 2.1 (O) 8 = EnDat 2.2 (O) 9 = Resolver (O) 10 = SSI (O) 11 = LDT (O) 12 = Hiperface DSL (O) 13 = BiSS (O) 14 = Integrated (O) 15 to 127 = (Reserved) 128 to 255 = (Vendor Specific)
1414 + (n-1)×50	Optional – E	Set		Feedback n Polarity	USINT	An enumerated value used to establish the direction of change in the feedback counter in response to positive motion of the associated feedback device. Normal polarity is defined as that which results in increasing feedback counts when the feedback device is hooked up and moved in the positive direction according to the devices published specifications. Inverted polarity internally switches the polarity of the feedback accumulator so that the feedback counts decrease when the feedback device moves in the positive direction. This attribute can be used to make the direction of travel agree with the user’s definition of positive travel and can be used in conjunction with the Motor Polarity bit to provide negative feedback, when this feedback channel is used for closed loop control.	Enumeration: 0 = Normal Polarity 1 = Inverted Polarity 2 to 255 = (reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1415 + (n-1)×50	Required -E	Set		Feedback n Startup Method	USINT	Determine how the device applies the feedback count value during drive startup. When configured for Incremental mode, the device zeroes the feedback count accumulator at power-up. The first Actual Position value sent to the controller in the Cyclic Data Block of the Device-to-Controller connection at power-up shall be zero. This is an indication to the controller that the drive has been power-cycled and the drive axis needs to be homed to establish a machine reference position. When configured for Absolute mode, the device initializes the feedback count accumulator at power-up to the absolute feedback position value read from the feedback device. When the feedback device's absolute position range is less than the 32-bit signed integer representation of the feedback count accumulator, the absolute position is sign extended to a 32-bit signed value. While there are many Feedback Types that support Absolute startup, there are a few strictly incremental types that do not, such as Digital AqB, and Sine/Cosine. Some device vendors tie the Feedback Start-up Method to the Feedback Type selection. In these cases, an attempt by the controller to incorrectly configured Feedback Startup Method shall generate an General Status error of Invalid Attribute Value.	Enumeration: 0 = Incremental (R) 1 = Absolute (O) 2 to 255 = (reserved)
1416 + (n-1)×50	Required – E Not LT	Set		Feedback n Cycle Resolution	UDINT	Determines the resolution capability of the associated feedback device. Units for this attribute are determined by the Feedback n Resolution Unit and the rotary or linear Feedback n Unit as shown in the Semantics column. For rotary feedback devices, this value is expressed as the number of Feedback Cycles per Revolution of the device, or alternatively by the number of bits in the binary position representation of the device per Revolution. For linear feedback devices, this value represents either the number of Feedback Cycles per Meter (m), or the number of nanometers (nm) per Feedback Cycle. Feedback Cycles for a Digital AqB device represents the "line" resolution of the encoder. Cycles for a Sin/Cos device represents the sinusoidal "cycle" resolution of the encoder. Cycles for a Resolver is the "pole" count of the device. For digital serial (e.g. SSI) or parallel absolute feedback devices Cycle represent the "step" or "count" resolution of the device.	Cycles/Unit (Rotary): Feedback Cycles / Rev (Rotary) Cycles/Unit (Linear): Feedback Cycles / m Unit/Cycle (Linear): nm / Feedback Cycle (Linear) Bits/Unit (Rotary): 2^n Cycles / Rev
1417 + (n-1)×50	Required – E Not LT	Set		Feedback n Cycle Interpolation	UDINT	Number of interpolated Feedback Counts per Feedback Cycle. For a Digital AqB device the device's feedback interface hardware can generally support interpolation values of 1, 2, or 4. For a Sin/Cos, Hiperface, Endat 2.1, or Resolver feedback device the number is generally much larger and determined by the interpolation capability of the device feedback interface hardware. A value of 1 024 is typical in this case. For digital serial (e.g. SSI) or parallel absolute feedback device interfaces, this value is always 1 since there is no opportunity of device-based interpolation. The effective resolution of the feedback device in Feedback Counts per Feedback Unit is determined by combination of Feedback Cycle Resolution and Feedback Cycle Interpolation attribute values.	Feedback Counts / Feedback Cycle

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1418 + (n-1)×50	Required – E (Rotary Absolute)	Set		Feedback n Turns	UDINT	Maximum number of shaft turns specified for a rotary absolute feedback device to maintain its absolute position reference. Typical rotary absolute feedback devices specify an absolute number of turns that typically range from 1 to 4 096 in powers of 2. This attribute can be used by the control system to determine the maximum Feedback Count range of the absolute feedback device, this being the product of the feedback cycle resolution, interpolation, and turns.	Feedback Units (Rev)
1419 + (n-1)×50	Required – E (Linear Absolute)	Set		Feedback n Length	REAL	Specified length of a linear absolute feedback device. Typical linear absolute feedback devices specify length in Meters. This attribute can be used by the control system to determine the maximum travel range of absolute feedback device in Feedback Counts, this being the combination of the feedback cycle resolution, interpolation, and length.	Feedback Units (m)
1420 + (n-1)×50	Optional – E TP, SS	Set		Feedback n Data Length	USINT	Number of feedback data bits transferred over the digital serial or parallel data interface channel of a feedback device.	Number of bits
1421 + (n-1)×50	Optional – E TP, SS	Set		Feedback n Data Code	USINT	Type of feedback data bit encoding used by designated serial or parallel data interface channel of a feedback device.	Enumeration: 0 = Binary 1 = Gray 2 to 255 = (reserved)
1422 + (n-1)×50	Optional – E RS	Set		Feedback n Resolver Transformer Ratio	REAL	Transformer Ratio specification of the designated resolver feedback device.	
1423 + (n-1)×50	Optional – E RS	Set		Feedback n Resolver Excitation Voltage	REAL	Sets the sinusoidal excitation voltage applied to the rotor of the designated resolver feedback device.	Volts (RMS)
1424 + (n-1)×50	Optional – E RS	Set		Feedback n Resolver Excitation Frequency	REAL	Frequency of sinusoidal excitation signal applied to the designated resolver feedback device. Valid frequency range or values for this attribute depends on the specific device hardware interface.	Hertz
1425 + (n-1)×50	Optional – E RS	Set		Feedback n Resolver Cable Balance	REAL	Adjusts the relative amplitude of the Sine and Cosine signals from the resolver to compensate for impact of resolver cable.	%

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
1426 + (n-1)×50	Required – E LT	Set		Feedback n LDT Type	USINT	Determines the LDT type. Options are Start/Stop and PWM. Start/Stop transducers accept an input (interrogate) signal to start the measurement cycle and respond with two pulses on the Return line. Timing can be based on either the Rising or Falling edge. The time between the pulses is proportional to the position. PWM transducers respond to the interrogate signal with a single long pulse on the Return line. The pulse width is proportional to the position.	Enumeration: 0 = PWM 1 = Start/Stop Rising 2 = Start/Stop Falling 3 to 255 = (reserved)
1427 + (n-1)×50	Required – E LT	Set		Feedback n LDT Recirculations	USINT	Determines the number of recirculations associated with a PWM type LDT transducer. Multiple recirculations can be used to increase the resolution of the LDT at the expense of increasing the sample period.	Number of Recirculations
1434 + (n-1)×50	Optional – E	Set		Feedback n Velocity Filter Bandwidth.	REAL	Controls the bandwidth of the Low Pass Filter applied to the raw velocity signal from Feedback n. A value of 0 for this attribute disabled this feature.	rad/s
1435 + (n-1)×50	Optional – E	Set		Feedback n Accel Filter Bandwidth	REAL	Controls the bandwidth of the Low Pass Filter applied to the raw acceleration signal from Feedback n. A value of 0 for this attribute disabled this feature.	rad/s

7.3.6.2 Semantics: Attribute No. 42 – Feedback Mode

This attribute contains a 4-bit enumerated value that determines how the various logical feedback channels are used to implement the selected Control Mode for this axis instance.

The Feedback Mode enumeration (see Table 67) provides support for multi-feedback device control functionality for the various active device Control Modes, i.e. where the device is actively controlling the motor based on feedback. In these active device Control Modes it is assumed that logical channel, Feedback 1, is attached directly to the motor while Feedback 2 is attached to the load side of the mechanical transmission. Commutation signals for a PM motor are always derived from the Feedback 1.

Table 67 – Feedback Mode enumeration definitions

Enum.	Req./Opt.	Name	Description
0	R/S	No Feedback	No Feedback is selected when sensorless/encoderless open loop or closed loop control is desired. When performing open loop control, no feedback signal is required. In closed loop control, the required feedback signal is estimated by a sensorless control algorithm based on motor phase voltage and current signals.
1	R/N	Master Feedback	Master Feedback assigns an uncommitted feedback channel, as specified by the Feedback Master Select attribute, to this device axis instance to serve as a “master feedback” source when the device is configured for No Control mode
2	R/C	Motor Feedback	When Motor Feedback is selected, then commutation, acceleration, velocity, and position feedback signals are all derived from motor mounted Feedback 1
3	O/C	Load Feedback	When Load Feedback is selected, then motor-mounted Feedback 1 is only used for PM motor commutation while load-side Feedback 2 is used for position, velocity, and acceleration.
4	O/P	Dual Feedback	When Dual Feedback is selected, then motor mounted Feedback 1 is used for commutation, acceleration, and velocity, and load-side Feedback 2 is used strictly for position.
5 to 7	-	Reserved	
8 to 15	-	Vendor Specific	

7.3.7 Event Capture attributes

7.3.7.1 General

The attribute table in Table 68 contains all event related attributes associated with a Motion Device Axis Object instance. These include registration, marker, and homing events. The Event Capture attributes are designed to support the possibility of up to 16 active events per controller update period. The format of all Time Stamp attributes is absolute System Time and follows the CIP Sync specification with 0 corresponding to 1970-01-01.

The Motion Device Axis Object currently supports two independent registration input channels per axis instance that can be triggered on either the rising or falling edges of the signal. In other words, Registration 1 for axis instance 1 is not generally the same signal as Registration 1 for axis instance 2. If the device hardware implementation allows, event time and position data can be captured for all four event conditions simultaneously. The Event Capture attributes also support Auto-rearm for registration events. This allows for controller implementation of important features like Windowed Registration and Registration Pattern Recognition.

The Motion Device Axis Object also supports Home Switch, Marker and Switch-Marker events for homing functionality on a per axis basis. The Marker events are typically generated by the configured position feedback device for the associated axis instance.

When the Motion Scaling Configuration is set to Drive Scaling, the Event Capture functionality extends to support Windowed Registration and Watch Events.

Set* ➔ These attributes are generally updated via the cyclic Command Data Set of the CIP Motion C-to-D connection. When included as cyclic command data, these attributes cannot be updated via a Set service.

Table 68 – Event attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
60	Optional – E	Set*		Event Checking Control	DWORD	This attribute is passed to the device by the controller as part of the Drive to Controller connection for the purpose of arming various device inputs, for example marker, home switch, and registration inputs, to generate events to the controller. When these enabled events occur, the device captures both the time and exact axis position when the event occurred. This attribute also manages the format and content of the C-to-D Event Data Block.	See Semantics in 7.3.7.2.1
61	Optional – E	Get		Event Checking Status	DWORD	This attribute is passed by the device to the controller as part of the Drive to Controller connection to indicate if the device is currently checking for events based on various device inputs, for example marker, home, and registration inputs. Event checking is initiated when the corresponding Event Checking Control bit is set in the Controller-to-Device connection. This attribute also manages the format and content of the D-to-C Event Data Block.	See Semantics in 7.3.7.2.2
62	Optional – E	Get		Registration 1 Positive Edge Position	DINT	Feedback position latched on the rising edge of the Registration Input 1.	Position Control Units
63	Optional – E	Get		Registration 1 Negative Edge Position	DINT	Feedback Position latched on the falling edge of the Registration Input 1.	Position Control Units
64	Optional – E	Get		Registration 2 Positive Edge Position	DINT	Feedback position latched on the rising edge of the Registration Input 2.	Position Control Units
65	Optional – E	Get		Registration 2 Negative Edge Position	DINT	Feedback Position latched on the falling edge of the Registration Input 2.	Position Control Units
66	Optional – E	Get		Registration 1 Positive Edge Time	LINT	System Time stamp on the rising edge of the Registration Input 1.	Nanoseconds (CIP Sync absolute)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
67	Optional – E	Get		Registration 1 Negative Edge Time	LINT	System Time stamp on the falling edge of the Registration Input 1.	Nanoseconds (CIP Sync absolute)
68	Optional – E	Get		Registration 2 Positive Edge Time	LINT	System Time stamp on the rising edge of the Registration Input 2.	Nanoseconds (CIP Sync absolute)
69	Optional – E	Get		Registration 2 Negative Edge Time	LINT	System Time stamp on the falling edge of the Registration Input 2.	Nanoseconds (CIP Sync absolute)
70	Required – NP Optional – VT E	Get		Home Event Position	DINT	Feedback Position latched on the specified home event.	Position Control Units
71	Optional – NP Optional – VT E	Get		Home Event Time	LINT	System Time stamp latched on the specified home event.	Nanoseconds (CIP Sync absolute)
72	Required – NP Optional – VT E (Drive Scaling)	Get		Watch 1 Event Time	LINT	System Time stamp latched on the specified watch event.	Nanoseconds (CIP Sync absolute)
73	Required – NP Optional – VT E (Drive Scaling)	Get		Watch 2 Event Time	LINT	System Time stamp latched on the specified watch event.	Nanoseconds (CIP Sync absolute)

7.3.7.2 Semantics

7.3.7.2.1 Attribute No. 60 – Event Checking Control

The first 28 bits of this 32-bit attribute (see Figure 63) are used to enable various device inputs, for example marker, home switch, and registration inputs, to generate events to the controller. When these enabled events occur, the device captures both the time and exact position of the associated motion axis instance. Three of the most significant 4 bits represent the number of Event Acknowledgement messages in the Event Data Block in this Controller-to-Device Connection update, with the final most, significant bit, indicating whether the Event Data Block format is Extended to support additional events associated with Drive Scaling functionality (see Motion Scaling Configuration). Support for Extended Format is optional in the device implementation. For a detailed description of the event handling protocol between the device and the controller, refer to Connection section in the appropriate CIP Motion device profile.

← 32-bit DWORD →		
Event Checking Control format		
Bit 31: Boolean Extended format	Bits 28-30: Integer Event Block Count	Bits 0-27: Bit Field Event Checking Control Bits

IEC

Figure 63 – Event Checking Control Word field

For the Home Switch-Marker events specified in Table 69, the device first looks for level of home switch input to transition according to the first + or – symbol and then immediately look for the transition of the marker input according to the second + or – symbol.

Table 69 – Event Checking Control bit definitions

Bit	Extended format only	Name	Description
0		Reg 1 Pos Edge	Enable checking for positive edge transition of Registration 1 Input
1		Reg 1 Neg Edge	Enable checking for negative edge transition of Registration 1 Input
2		Reg 2 Pos Edge	Enable checking for positive edge transition of Registration 2 Input
3		Reg 2 Neg Edge	Enable checking for negative edge transition of Registration 2 Input
4	*	Reg 1 Pos Edge Window	Enable Window check after positive edge transition of Registration 1 Input
5	*	Reg 1 Neg Edge Window	Enable Window check after negative edge transition of Registration 1 Input
6	*	Reg 2 Pos Edge Window	Enable Window check after positive edge transition of Registration 2 Input
7	*	Reg 2 Neg Edge Window	Enable Window check after negative edge transition of Registration 2 Input
8		Reg 1 Pos Auto-rearm	Enable auto-rearm after positive edge transition of Registration 1 Input
9		Reg 1 Neg Auto-rearm	Enable auto-rearm after negative edge transition of Registration 1 Input
10		Reg 2 Pos Auto-rearm	Enable auto-rearm after positive edge transition of Registration 2 Input
11		Reg 2 Neg Auto-rearm	Enable auto-rearm after negative edge transition of Registration 2 Input
12	*	Watch Position 1 Fwd	Enable checking for axis position crossing Watch Position 1 in the forward (positive) direction.
13	*	Watch Position 1 Rev	Enable checking for axis position crossing Watch Position 1 in the reverse (negative) direction.
14	*	Watch Position 2 Fwd	Enable checking for axis position crossing Watch Position 2 in the forward (positive) direction.
15	*	Watch Position 2 Rev	Enable checking for axis position crossing Watch Position 2 in the reverse (negative) direction.
16		Marker Pos Edge	Enable checking for positive edge of Marker input of associated position feedback channel.
17		Marker Neg Edge	Enable checking for negative edge of Marker input of associated position feedback channel.
18		Home Switch Pos Edge	Enable checking for positive edge of Home input associated with this axis instance.
19		Home Switch Neg Edge	Enable checking for positive edge of Home input associated with this axis instance.
20		Home Switch-Marker ++	Enable checking for event sequence specified as a positive edge transition of the Home Switch input followed by a positive edge of the Marker input.

Bit	Extended format only	Name	Description
21		Home Switch-Marker +-	Enable checking for event sequence specified as a positive edge transition of the Home Switch input followed by a negative edge of the Marker input.
22		Home Switch-Marker -+	Enable checking for event sequence specified as a negative edge transition of the Home Switch input followed by a positive edge of the Marker input.
23		Home Switch-Marker --	Enable checking for event sequence specified as a negative edge transition of the Home Switch input followed by a negative edge of the Marker input.
24	*	Home Torque Threshold	Enable checking for home torque event specified as a torque level that exceeds the Home Torque Threshold for a period of time given by the Home Torque Time.
25 to 27		(Reserved)	

7.3.7.2.2 Attribute No. 61 – Event Checking Status

The first 28 bits of this 32-bit attribute (see Figure 64) are used to indicate if the device is currently checking for events based on various device inputs, for example marker and registration inputs (see Table 70). Event checking is initiated when the corresponding Event Checking Control bit is set in the Controller-to-Device Connection. When an event occurs, the device captures both the time and exact axis position and passes this information to the controller in event notification data blocks. But for the controller to process the event data, the corresponding Event Checking Status bit shall be set. Three of the most significant 4 bits of this attribute represent the number of Event Notification messages in the Event Data Block in this Device-to-Controller Connection update, with the final, most significant, bit indicating whether the Event Data Block format is Extended to support additional events associated with Drive Scaling functionality (see Motion Scaling Configuration). Support for Extended Format is optional in the device implementation. For a detailed description of the event handling protocol between the device and the controller, refer to Connection section.

← 32-bit DWORD →		
Event Checking Status format		
Bit 31: Boolean Extended format	Bits 28-30: Integer Event Block Count	Bits 0-27: Bit Field Event Checking Status Bits

IEC

Figure 64 – Event Checking Status word field

Table 70 – Event Checking Status bit definitions

Bit	Extended format only	Name	Description
0		Reg 1 Pos Edge	Checking for positive edge transition of Registration 1 Input
1		Reg 1 Neg Edge	Checking for negative edge transition of Registration 1 Input
2		Reg 2 Pos Edge	Checking for positive edge transition of Registration 2 Input
3		Reg 2 Neg Edge	Checking for negative edge transition of Registration 2 Input
4	*	Reg 1 Pos Edge Window	Apply Window check after positive edge transition of Registration 1 Input
5	*	Reg 1 Neg Edge Window	Apply Window check after negative edge transition of Registration 1 Input
6	*	Reg 2 Pos Edge Window	Apply Window check after positive edge transition of Registration 2 Input
7	*	Reg 2 Neg Edge Window	Apply Window check after negative edge transition of Registration 2 Input
8		Reg 1 Pos Auto-rearm	Auto-rearm after positive edge transition of Registration 1 Input active
9		Reg 1 Neg Auto-rearm	Auto-rearm after negative edge transition of Registration 1 Input active
10		Reg 2 Pos Auto-rearm	Auto-rearm after positive edge transition of Registration 2 Input active
11		Reg 2 Neg Auto-rearm	Auto-rearm after negative edge transition of Registration 2 Input active
12	*	Watch Position 1 Fwd	Checking for watch event with actual position crossing Watch Position 1 in the forward (positive) direction.
13	*	Watch Position 1 Rev	Checking for watch event with actual position crossing Watch Position 1 in the reverse (negative) direction.
14	*	Watch Position 2 Fwd	Checking for watch event with actual position crossing Watch Position 2 in the forward (positive) direction.
15	*	Watch Position 2 Rev	Checking for watch event with actual position crossing Watch Position 2 in the reverse (negative) direction.
16		Marker Pos Edge	Checking for positive edge of Marker input of associated position feedback channel
17		Marker Neg Edge	Checking for negative edge of Marker input of associated position feedback channel
18		Home Switch Pos Edge	Checking for positive edge of Home input associated with this axis instance
19		Home Switch Neg Edge	Checking for positive edge of Home input associated with this axis instance
20		Home Switch-Marker ++	Checking for event sequence specified as a positive edge transition of the Home Switch input followed by a positive edge of the Marker input
21		Home Switch-Marker +-	Checking for event sequence specified as a positive edge transition of the Home Switch input followed by a negative edge of the Marker input
22		Home Switch-Marker -+	Checking for event sequence specified as a negative edge transition of the Home Switch input followed by a positive edge of the Marker input
23		Home Switch-Marker --	Checking for event sequence specified as a negative edge transition of the Home Switch input followed by a negative edge of the Marker input
24	*	Home Torque Threshold	Checking for home torque event specified as a torque level that exceeds the Home Torque Threshold for a period of time given by the Home Torque Time.
25 to 27		(Reserved)	

7.3.8 Command reference generation attributes

The lists of attributes in Table 71 and Table 72 apply to the command reference generation functionality of the device that converts command position, velocity, acceleration, and torque data output from a controller-based planner into corresponding command references signals to the device's motor control structures. The command reference generator functionality includes fine interpolators, ramp generators, signal selector switches.

Set* ➔ These attributes are generally updated via the cyclic Command Data Set of the CIP Motion C-to-D connection. When included as cyclic command data, these attributes shall not be updated via a Set service.

Table 71 – Command Generator Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
360	Optional – P (If not supported, then Controller Position Command – Float is Required.)	Set*		Controller Position Command – Integer	DINT	Integer command position data value from controller, updated at the Controller Update Period, and feeding into the Position Fine Command Generator. This integer representation of command position is appropriate for simple positioning devices where real time processing of double precision floating point values is too time consuming and the smoothness of motion is not a driving factor.	Position Control Units
361	Optional – P (If not supported, then Controller Position Command – Integer is Required.)	Set*		Controller Position Command – Float	LREAL	Floating point command position data value from controller, updated at the Controller Update Period, and feeding into the Position Fine Command Generator. The range of this value is limited to that of a DINT by the motion planner but, but unlike a DINT, it carries a fractional component that translates to smoother motion especially for feedforward operation.	Position Control Units
362	Required – VF Optional – P	Set*		Controller Velocity Command	REAL	Command velocity data value from controller, updated at the Controller Update Period, and feeding into the Velocity Fine Command Generator.	Velocity Control Units / s
363	Optional – PVT	Set*		Controller Acceleration Command	REAL	Command acceleration data value from controller, updated at the Controller Update Period, and feeding into the Acceleration Fine Command Generator.	Accel Control Units / s^2
364	Required – T	Set*		Controller Torque Command	REAL	Commanded torque data value from controller used in torque control mode, updated at the Controller Update Period, and feeding into the Torque Fine Command Generator.	% Motor Rated

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
365	Optional – P	Get		Fine Command Position	DINT	Output value from the Position Fine Command Generator.	Position Control Units
366	Optional – PV	Get		Fine Command Velocity	REAL	Output value from the Command Velocity Fine Command Generator. When no Command Velocity signal is present when performing position control, this signal can be derived by scaling the Differential Position output value of the Position Fine Command Generator.	Velocity Control Units / s
367	Optional – C	Get		Fine Command Acceleration	REAL	Output value from the Acceleration Fine Command Generator. When no Command Acceleration signal is present when performing position or velocity control, this signal can be derived by scaling the Differential Velocity output value of the Velocity Fine Command Generator. If no Command Velocity signal is present, the Fine Command Acceleration signal can be derived by scaling the 2nd Differential Position output value of the Position Fine Command Generator.	Accel Control Units / s^2

Table 72 – Command Generator Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
370	Optional – F	Set		Skip Speed 1	REAL	Sets the central speed of a skip speed band within which the device does not operate. The skip speed value is signed.	Velocity Control Units / s
371	Optional – F	Set		Skip Speed 2	REAL	Sets the central speed of a skip speed band within which the device does not operate. The skip speed value is signed.	Velocity Control Units / s
372	Optional – F	Set		Skip Speed 3	REAL	Sets the central speed of a skip speed band within which the device does not operate. The skip speed value is signed.	Velocity Control Units / s
373	Optional – F	Set		Skip Speed Band	REAL	Determines the speed window around a skip speed that cannot be commanded. Any command set-point within this window is adjusted by the Skip Speed block to fall at either the upper or lower Skip Speed Band boundary value. The device can smoothly accelerate or decelerate through the skip speed band due to the Ramp Generator but may not operate at a set speed within the band. The Skip Speed Band is distributed ½ above and ½ below the skip speed. This Skip Speed Band attribute applies to all skip speeds supported in the device. A value of 0 for this attribute disabled this feature.	Velocity Control Units / s
374	Optional – FV	Set		Ramp Velocity – Positive	REAL	This positive value defines the maximum positive velocity command output of the Ramp Generator.	Velocity Control Units / s
375	Optional – FV	Set		Ramp Velocity – Negative	REAL	This negative value defines the maximum negative velocity command output of the Ramp Generator.	Velocity Control Units / s
376	Optional – FV	Set		Ramp Acceleration	REAL	This positive value defines the maximum acceleration (increasing speed) of the velocity command output by the Ramp Generator.	Velocity Control Units / s^2
377	Optional – FV	Set		Ramp Deceleration	REAL	This positive value defines the maximum deceleration (decreasing speed) of the velocity command output by the Ramp Generator.	Velocity Control Units / s^2

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
378	Optional – FV	Set		Ramp Jerk Control	REAL	This value sets the percentage of accel or decel time that is applied to the speed ramp as jerk limited S Curve based on a step change in velocity. The S Curve time is added ½ at the beginning and ½ at the end of the ramp. A value of 0 results in no S-Curve, i.e. a linear acceleration or deceleration ramp. A value of 100 % results in a triangular acceleration profile with the peak being the configured ramp acceleration or deceleration. As the Jerk Control value increases the derived accelerating jerk value decreases based on (0,5 × 0,01 × Jerk Control × Ramp Vel Positive / Ramp Accel), and the decelerating Jerk limit value also decreases according to (0,5 × 0,01 × Jerk Control × Ramp Vel Negative / Ramp Decel).	%
380	Optional – FV	Set		Flying Start Enable	BOOL	This Boolean value is used to enable or disable the Flying Start feature of the device's Ramp Generator. When Flying Start Enabled is true and the motion axis is enabled, the device determines the current velocity of the motor as part of the Starting State initialization activities. Just prior to transitioning to the Running state, the device presets the output of the Ramp Generator to the current velocity. In this way, the motor seamlessly ramps from its current velocity to the commanded velocity. When Flying Start Enabled is false, the motor velocity is irrelevant and a preset of 0 is applied to the Ramp Generator output.	0 = Flying Start Disabled 1 = Flying Start Enabled

7.3.9 Control mode attributes

7.3.9.1 Position Loop attributes

The attributes tables in Table 73 and Table 74 contain all position control related attributes associated with a Motion Device Axis Object instance.

Set* ➔ this attribute is generally updated via the cyclic Command Data Set of the CIP Motion C-to-D Connection. When included as cyclic command data, this attribute shall not be updated via a Set service.

Table 73 – Position Loop Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
430	Required – P	Get		Position Command	DINT	Command position output from the Fine Command Generator (if active) into position loop when configured for position loop control.	Position Control Units
431	Required – P	Set*		Position Trim	DINT	Additional position command added to the Position Command to generate the Position Reference signal into the position loop summing junction.	Position Control Units
432	Required – P	Get		Position Reference	DINT	Command position reference signal into the position loop summing junction to be compared with a position feedback signal.	Position Control Units
433	Required – P	Get		Velocity Feedforward Command	REAL	Velocity feedforward command signal that represents a scaled version of the command velocity profile. This signal is the Velocity Fine Command signal scaled by Kvff and applied to the output of the position loop.	Velocity Control Units/s
434	Required – E	Get		Position Feedback	DINT	Position feedback value channeled into the position control summing junction. In most cases the Position Feedback signal is derived directly from the feedback device specified by the Feedback Mode selection. If the axis configured for “No Control” mode, Position Feedback represents the actual position of the feedback device specified by the Feedback Master Select.	Position Control Units
436	Required – P	Get		Position Error	REAL	Error between commanded and actual position that is the output of the position loop summing junction.	Position Control Units
437	Required – P	Get		Position Integrator Output	REAL	Output of position integrator representing the contribution of the position integrator to Position Loop Output.	Velocity Control Units/s
438	Required – P	Get		Position Loop Output	REAL	Output of the position loop forward path representing the total control effort of the position loop.	Velocity Control Units/s

Table 74 – Position Loop Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
440	Required – P	Set		Kvff	REAL	Velocity Feedforward Gain value that multiplies the Velocity Feedforward Command signal to form the Velocity Feeforward Command that is applied to the output of the position control loop. 100 % Velocity Feedforward applies the full Velocity Fine Command signal to the velocity loop.	%
441	Required – P	Set		Kpp	REAL	Position Proportional Gain value that multiplies the Position Error signal to form the proportional control signal that summed together with the integral control signal to generate the output of the position control loop. This value directly determines the bandwidth of the position loop.	rad/s
442	Required – P	Set		Kpi	REAL	Position Integral Gain value that, together with the Kpp, multiplies the Position Integrator Error signal to form the integral control signal that is summed together with the proportional control signal to generate the output of the position control loop. The reciprocal of this value, 1/Kpi, represents the integrator time constant for the position loop. A value of 0 for this attribute disables the integrator.	rad/s
443	Required – P	Set		Position Lock Tolerance	REAL	This value establishes a window around the current command position. When the actual position is within this window the Position Lock status bit is set. When actual position falls outside this window, the Position Lock status bit is cleared.	Position Control Units
444	Required – P	Set		Position Error Tolerance	REAL	Determines the absolute maximum Position Error value that can be tolerated without causing an Excessive Position Error exception.	Position Control Units
445	Optional – P	Set		Position Error Tolerance Time	REAL	Determines the maximum amount of time that the Position Error Tolerance can be exceeded without generating an exception.	Seconds

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
446	Required – P	Set		Position Integrator Control	BYTE	This bit field controls the behavior of the position loop integrator while commanding motion via the controller. When the integrator hold enable bit is set, the integrator is held while motion is being commanded with a non-zero velocity. When clear, the integrator runs without qualification. When the auto-preset bit is set, the integrator preload value is automatically loaded with the current velocity command when there is a control mode change between velocity control and position control. If clear, the integrator is loaded with the configured position integrator preload value.	Bit Field: 0: Integrator Hold Enable (R) 1: Auto-Preset (O) 2-7: (Reserved)
447	Optional – P	Set		Position Integrator Preload	REAL	Value assigned to the position integrator when the position control loop is enabled.	Velocity Control Units/s
448	Optional – E (Drive Scaling)	Set		Position Limit – Positive	REAL	This value defines the most positive position feedback value that when exceeded in the positive direction, generates a Position Overtravel Positive exception.	Position Control Units
449	Optional – E (Drive Scaling)	Set		Position Limit – Negative	REAL	This value defines the most negative position feedback value that when exceeded in the negative direction, generates a Position Overtravel Negative exception.	Position Control Units

7.3.9.2 Velocity Loop attributes

The attributes tables in Table 75 and Table 76 contain all velocity control related attributes associated with a Motion Device Axis Object instance.

Set* ➔ this attribute is generally updated via the cyclic Command Data Set of the CIP Motion C-to-D Connection. When included as cyclic command data, this attribute shall not be updated via a Set service.

Table 75 – Velocity Loop Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
450	Required – FPV	Get		Velocity Command	REAL	Command velocity output from Fine Command Generator (if active) into the Velocity Loop control or the frequency controller when configured for Frequency Control operation.	Velocity Control Units/s
451	Required – FPV	Set*		Velocity Trim	REAL	Additional velocity command added to the velocity loop or frequency control summing junction.	Velocity Control Units/s
452	Required – PV	Get		Acceleration Feedforward Command	REAL	Acceleration feedforward command signal that represents a scaled version of the command acceleration profile. This signal is the Accelleration Fine Command signal scaled by Kaff and applied to the output of the velocity loop.	Velocity Control Units/s^2
453	Required – FPV	Get		Velocity Reference	REAL	Command velocity reference into velocity loop summing junction, or in the case of Frequency Control, the signal that is scaled to become the Frequency Reference.	Velocity Control Units/s
454	Required – All	Get		Velocity Feedback	REAL	Actual velocity of the axis that is applied to the velocity summing junction, if applicable, based on Control Mode selection. In most cases the Velocity Feedback signal is derived directly from the feedback device specified by the Feedback Mode selection. If the axis is configured for “No Control” mode, Velocity Feedback represents the actual velocity of the feedback device specified by the Feedback Master Select. If axis is configured for Frequency Control, the Velocity Feedback signal is derived from the Velocity Command signal. If configured for Sensorless Velocity Loop operation, Velocity Feedback is estimated by the sensorless control algorithm.	Velocity Control Units/s
455	Required – PV	Get		Velocity Error	REAL	Error between the Velocity Reference and Velocity Feedback value that is the output of the velocity loop summing junction.	Velocity Control Units/s
456	Required – PV	Get		Velocity Integrator Output	REAL	Output of velocity integrator representing the contribution of the velocity integrator to Velocity Loop Output.	Accel Control Units/s^2
457	Required – PV	Get		Velocity Loop Output	REAL	Output of velocity forward path representing the total control effort of the velocity loop.	Accel Control Units/ s^2

Table 76 – Velocity Loop Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
460	Required – PV	Set		Kaff	REAL	Acceleration Feedforward Gain value that multiplies the Acceleration Fine Command signal to form the Acceleration Feedforward Command before applying it to the output of the velocity control loop. 100 % Acceleration Fine applies the full Acceleration Feedforward Command signal to the output of the velocity loop.	%
461	Required – PV	Set		Kvp	REAL	Velocity Proportional Gain value that multiplies the Velocity Error signal to form the proportional control signal that summed together with the integral control signal to generate the output of the velocity control loop. This value directly determines the bandwidth of the velocity loop.	rad/s
462	Required – PV	Set		Kvi	REAL	Velocity Integral Gain value that, together with the Kvp, multiplies the Velocity Integrator Error signal to form the integral control signal that is summed together with the proportional control signal to generate the output of the velocity control loop. The reciprocal of this value, 1/Kvi, represents the integrator time constant for the velocity loop. A value of 0 for this attribute disables the integrator.	rad/s
464	Optional – FPV	Set		Kdr	REAL	Velocity Droop value that provides compliance to the velocity integrator by subtracting a portion of the velocity loop effort from the velocity error input to the velocity integrator. The presence of the Torque/Force scaling gain, Kj, in the droop signal path allows Velocity Droop to be specified in velocity units per % rated torque output. This parameter is also valid for V/Hz devices and its behavior is nearly identical, but instead of % rated being related to torque, % rated is related to current.	Velocity Control Units/ s/ % Rated
465	Optional – PV	Set		Velocity Error Tolerance	REAL	Determines the absolute maximum Velocity Error value that can be tolerated without causing an Excessive Velocity Error exception.	Velocity Control Units/ s
466	Optional – PV	Set		Velocity Error Tolerance Time	REAL	Determines the maximum amount of time that the Velocity Error Tolerance can be exceeded without generating an exception.	seconds
467	Required – PV	Set		Velocity Integrator Control	BYTE	This bit field controls the behavior of the velocity loop integrator while commanding motion via the controller. When the integrator hold enable bit is set, the integrator is held while motion is being commanded with a non-zero velocity. When clear, the integrator runs without qualification. When the auto-preset bit is set, the integrator preload value is automatically loaded with the current torque command when there is a control mode change between torque control and velocity control. If clear, the integrator is loaded with the configured velocity integrator preload value.	Bit Field: 0: Integrator Hold Enable (R) 1: Auto-Preset (O) 2-7: (Reserved)
468	Optional – PV	Set		Velocity Integrator Preload	REAL	Value assigned to the velocity integrator when the velocity control loop is enabled.	Accel Control Units/s^2
469	Optional – PV	Set		Velocity Low Pass Filter Bandwidth	REAL	Controls the bandwidth of the Low Pass Filter applied to the Velocity Error signal. Recommended implementation is a two pole IIR filter. A value of 0 for this attribute disables this feature.	rad/s
470	Optional – ED	Set		Velocity Threshold	REAL	Defines a minimum absolute velocity. If the magnitude of the Velocity Feedback signal is less than this value, the Velocity Threshold status bit is set.	Velocity Control Units/s

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
471	Optional – FPV	Set		Velocity Lock Tolerance	REAL	This value establishes a window around the unlimited velocity reference signal. When the Velocity Feedback signal is within this window the Velocity Lock status bit is set. When Velocity Feedback signal falls outside this window, the Velocity Lock status bit is cleared.	Velocity Control Units/s
472	Required – ED	Set		Velocity Standstill Window	REAL	This value establishes a window around zero speed. When the Velocity Feedback signal is within this window the Velocity Standstill status bit is set. When Velocity Feedback signal falls outside this window, the Velocity Standstill status bit is cleared.	Velocity Control Units/s
473	Optional – FPV	Set		Velocity Limit – Positive	REAL	This positive value defines the most positive velocity reference value into the velocity summing junction. If this velocity limit value is exceeded, the device responds by clamping the velocity reference to this limit and setting the Velocity Limit status bit.	Velocity Control Units/s
474	Optional – FPV	Set		Velocity Limit – Negative	REAL	This negative value defines the most negative velocity reference value allowed into the velocity summing junction. If this velocity limit value is exceeded, the device responds by clamping the velocity reference to this limit and setting the Velocity Limit status bit.	Velocity Control Units/s

7.3.9.3 Acceleration Control attributes

The attributes tables in Table 77 and Table 78 contain all accelerations related attributes associated with a Motion Device Axis Object instance.

Set* ➔ this attribute is generally updated via the cyclic Command Data Set of the CIP Motion C-to-D Connection. When included as cyclic command data, this attribute shall not be updated via a Set service.

Table 77 – Acceleration Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
480	Optional – C	Get		Acceleration Command	REAL	Command acceleration output from Fine Command Generator (if active) into acceleration loop when configured for acceleration control.	Accel Control Units/s^2
481	Optional – C	Set*		Acceleration Trim	REAL	Additional acceleration command added to the acceleration loop summing junction.	Accel Control Units/s^2
482	Optional – C	Get		Acceleration Reference	REAL	Command acceleration reference into acceleration loop summing junction.	Accel Control Units/s^2
483	Required – E	Get		Acceleration Feedback	REAL	Actual acceleration of the axis based on the selected feedback device. In most cases the Acceleration Feedback signal is derived directly from the feedback device specified by the Feedback Mode selection. If the axis configured for "No Control" mode Acceleration Feedback represents the actual acceleration of the feedback device specified by the Feedback Master Select.	Accel Control Units/s^2

Table 78 – Acceleration Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
485	Optional – D	Set		Acceleration Limit	REAL	This value defines the maximum acceleration value (increasing speed) into the acceleration summing junction. If this acceleration limit value is exceeded, the device responds by clamping the acceleration reference to this limit and setting the Acceleration Limit status bit.	Accel Control Units / s^2
486	Optional – D	Set		Deceleration Limit	REAL	This value defines the maximum deceleration value (decreasing speed) into the acceleration summing junction. If this deceleration limit value is exceeded, the device responds by clamping the acceleration reference to this limit and setting the Deceleration Limit status bit.	Accel Control Units / s^2

7.3.9.4 Torque/Force Control attributes

The attributes tables in Table 79 and Table 80 contain all torque/force related attributes associated with a Motion Device Axis Object instance.

Set* ➔ this attribute is generally updated via the cyclic Command Data Set of the CIP Motion C-to-D Connection. When included as cyclic command data, this attribute shall not be updated via a Set service.

Table 79 – Torque/Force Control Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
490	Required – C	Get		Torque Command	REAL	Command torque output from Fine Command Generator (if active) into torque input summing junction when configured for torque control.	% Motor Rated
491	Required – C	Set*		Torque Trim	REAL	Additional torque command added to the torque input summing junction.	% Motor Rated
492	Required – C	Get		Torque Reference	REAL	Commanded torque reference input signal before torque filter section representing the sum of the Torque Command and Torque Trim signal inputs.	% Motor Rated
493	Required – C	Get		Torque Reference – Filtered	REAL	Commanded torque reference input signal after torque filter section.	% Motor Rated
494	Required – C	Get		Torque Reference – Limited	REAL	Commanded torque reference input signal after torque limiter section.	% Motor Rated

Table 80 – Torque/Force Control Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
496	Required – PV Optional – T	Set		Kj	REAL	Acceleration to Torque/Force scaling gain value that converts commanded acceleration into equivalent rated torque/force. Properly set, this value represents the total system inertia or mass.	% Motor Rated/ (Motor Units/ s^2)
498	Optional – C	Set		Friction Compensation – Sliding	REAL	Value added to the current/torque command to offset the effects of coulomb, or sliding friction.	% Motor Rated
499	Optional – C	Set		Friction Compensation – Static	REAL	Value added to the current/torque command to offset the effects of static friction, sometimes referred to as “sticktion”.	% Motor Rated
500	Optional – C	Set		Friction Compensation – Viscous	REAL	Value added to the current/torque command to offset the effects of viscous friction, i.e. friction that is proportional to speed.	% Motor Rated / (Motor Units/ s)
502	Optional – C	Set		Torque Low Pass Filter Bandwidth	REAL	Break frequency for the low pass filter applied to torque reference signal.	rad/s
503	Optional – C	Set		Torque Notch Filter Frequency	REAL	Center frequency of the notch filter applied to the torque reference signal. A value of 0 for this attribute disables this feature.	rad/s
504	Required – C	Set		Torque Limit – Positive	REAL	This positive value determines the maximum positive torque that can be applied to the motor. If the device attempts to exceed this value, the torque command is clamped to this value.	% Motor Rated
505	Required – C	Set		Torque Limit – Negative	REAL	This negative value determines the maximum negative torque that can be applied to the motor. If the device attempts to exceed this value, the torque command is clamped to this value.	% Motor Rated
506	Optional – C	Set		Torque Rate Limit	REAL	Limits the rate of change of the torque reference signal.	% Motor Rated/s
507	Optional – C	Set		Torque Threshold	REAL	Specifies the threshold for the Filtered Torque Reference signal magnitude that, when exceeded, results in the Torque Threshold status bit being set.	% Motor Rated

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
508	Optional – D	Set		Overtorque Limit	REAL	Maximum threshold for the torque producing Iq Current Feedback signal magnitude. When the Iq Current Feedback signal is greater than this value for the duration specified by Overtorque Limit Time attribute, the results is an Overtorque Limit exception. This feature allows the device to generate an exception if there is a sudden increase in load torque magnitude during operation. This condition could occur if a bearing fails, a hard stop is reached, or there is some other mechanical failure.	% Motor Rated
509	Optional – D	Set		Overtorque Limit Time	REAL	Specifies the amount of time allowed in an Overtorque Limit condition before generating an Overtorque Limit exception. A value of 0 for this attribute disables the Overtorque feature.	seconds
510	Optional – D	Set		Undertorque Limit	REAL	Minimum threshold for the torque producing Iq Current Feedback signal magnitude. When the Iq Current Feedback signal is less than this value for the duration specified by Undertorque Limit Time attribute, the result is an Undertorque Limit exception. This feature allows the device to generate an exception if there is a sudden decrease in load torque magnitude during operation. This condition could occur if a load coupling breaks, tensioned web material breaks, etc.	% Motor Rated
511	Optional – D	Set		Undertorque Limit Time	REAL	Specifies the amount of time allowed in an Undertorque Limit condition before generating an Undertorque Limit exception. A value of 0 for this attribute disables the Undertorque feature.	seconds

7.3.9.5 Current Control attributes

The attributes tables in Table 81 and Table 82 contain all current control related attributes associated with a Motion Device Axis Object instance.

Table 81 – Current Control Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
520	Required – C	Get		Iq Current Command	REAL	Represents the instantaneous commanded torque producing current signal, Iq, prior to passing through the Current Vector Limiter. It is tied directly to the output of torque reference path after the 1/Kt scaling that represents the torque effort to be applied to the drive's torque producing Iq current loop. The nominal value for 1/Kt is 1 based on 100 % rated torque being produced by 100 % rated current.	% Motor Rated
521	Optional – C	Get		Operative Current Limit	REAL	Represents the operative current limit based on multiple limit sources.	% Motor Rated
522	Optional – C	Get		Current Limit Source	USINT	Represents the operative source of a current limit shall a current limit condition occur.	Enumeration: 0 = Not Limited 1 = Inverter Peak Current Limit 2 = Motor Peak Current Limit 3 = Inverter Thermal Current Limit 4 = Motor Thermal Current Limit 5 = Shunt Regulator Limit 6 = Current Vector Limit 7 = Brake Test Limit 8 to 127 = reserved 128 to 255 = vendor specific
523	Required – C (PM Motor)	Get		Motor Electrical Angle	REAL	Calculated electrical angle of the motor based on motor pole count, commutation offset, and selected feedback device.	Degrees
524	Optional – C	Get		Iq Current Reference	REAL	Current reference signal, Iq, into the torque producing current loop summing junction.	% Motor Rated
525	Optional – C	Get		Id Current Reference	REAL	Command current reference, Id, into the flux producing current loop summing junction.	% Motor Rated
527	Optional – C	Get		Iq Current Error	REAL	Error between commanded and actual current that is the output of the torque producing, q-axis, current loop summing junction.	% Motor Rated

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
528	Optional – C	Get		Id Current Error	REAL	Error between commanded and actual current that is the output of the flux producing, d-axis, current loop summing junction.	% Motor Rated
529	Optional – C	Get		Iq Current Feedback	REAL	Actual torque current applied to the axis based on current sensor feedback.	% Motor Rated
530	Optional – C	Get		Id Current Feedback	REAL	Actual flux current applied to the axis based on current sensor feedback.	% Motor Rated
531	Optional – C	Get		Vq Decoupling	REAL	Instantaneous voltage signal added to the Iq control loop output to compensate for the effects of Id.	Volts
532	Optional – C	Get		Vd Decoupling	REAL	Instantaneous voltage signal added to the Id control loop output to compensate for the effects of Iq.	Volts
533	Optional – C	Get		Vq Voltage Output	REAL	Instantaneous Torque/Force producing output voltage.	Volts
534	Optional – C	Get		Vd Voltage Output	REAL	Instantaneous Flux producing output voltage.	Volts
535	Optional – C	Get		U Voltage Output	REAL	Instantaneous voltage applied to U phase.	Volts
536	Optional – C	Get		V Voltage Output	REAL	Instantaneous voltage applied to V phase.	Volts
537	Optional – C	Get		W Voltage Output	REAL	Instantaneous voltage applied to W phase.	Volts
538	Optional – C	Get		U Current Feedback	REAL	Instantaneous current measured on U phase.	Amperes
539	Optional – C	Get		V Current Feedback	REAL	Instantaneous current measured on V phase.	Amperes
540	Optional – C	Get		W Current Feedback	REAL	Instantaneous current measured on W phase.	Amperes
541	Optional – C	Get		U Current Offset	REAL	Offset for U Phase current transducer.	Amperes
542	Optional – C	Get		V Current Offset	REAL	Offset for V Phase current transducer.	Amperes
543	Optional – C	Get		W Current Offset	REAL	Offset for W Phase current transducer.	Amperes

Table 82 – Current Control Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
553	Optional – D	Set		Current Vector Limit	REAL	Current Vector Limit value applied to current vector limiter to provide a configurable limit the magnitude of the current vector.	% Motor Rated
554	Optional – C	Set		Kqp	REAL	Iq Proportional Gain value that multiplies the Iq Current Error signal. This value directly determines the bandwidth of the torque producing current loop.	rad/s
555	Optional – C	Set		Kqi	REAL	Iq Integral Gain value that, together with Kdp, multiplies the Iq Current Error signal before applying it to the Iq Integrator Error accumulator. The reciprocal of this value, 1/Kdi, represents the integrator time constant for the flux current loop. A value of 0 for this attribute disables the integrator.	rad/s
556	Optional – C	Set		Kdp	REAL	Id Proportional Gain value that multiplies the Id Current Error signal. This value directly determines the bandwidth of the flux producing current loop.	rad/s
557	Optional – C	Set		Kdi	REAL	Id Integral Gain value that, together with Kdp, multiplies the Id Current Error signal before applying it to the Id Integrator Error accumulator. The reciprocal of this value, 1/Kdi, represents the integrator time constant for the flux current loop. A value of 0 for this attribute disables the integrator.	rad/s
558	Optional – D (Induction Motor)	Set		Flux Up Control	USINT	When the motion axis is enabled, DC current is applied to an induction motor to build stator flux before transitioning to the Running state. This attribute controls how an induction motor is to be fluxed in the Starting state prior to transitioning to the Running state. If No Delay is selected the axis transitions immediately to the Running state while the motor flux is building. With Manual Delay, the axis remains in the Starting state for the Flux Up Time to allow time for the motor to be fully fluxed. With Automatic Delay, the drive device determines the amount of time to delay to fully flux the motor based on motor configuration attribute data or measurements.	Enumeration: 0 = No Delay (R) 1 = Manual Delay (O) 2 = Automatic Delay (O) 3 to 255 = (reserved)
559	Optional – D (Induction Motor)	Set		Flux Up Time	REAL	Sets the amount of time the drive device allows to build full motor flux before transitioning to the Running state.	seconds

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
560	Optional – CE (PM Motor)	Set		Commutation Startup Method	USINT	Specifies the method used by the drive to establish absolute rotor (or moving coil) alignment relative to stator windings for the purposes of PM motor commutation when starting up the drive. If this attribute is not supported, the commutation startup method is determined implicitly by the Feedback 1 Type selection. Likewise, if this attribute is supported and set to From Feedback Type, the commutation startup method is also determined implicitly by the Feedback 1 Type selection. The UVW startup method uses UVW signals from motor mounted encoder tracks or Hall sensors together with the Commutation Offset to properly align the rotor with stator windings or, in the case of a linear motor, the moving coil with magnet track. Once aligned, commutation is maintained via position signals from the motor mounted feedback device, i.e. Feedback 1. The Digital startup method uses Digital signals from a motor mounted absolute feedback device together with the Commutation Offset to align the rotor with stator windings or, in the case of a linear motor, the moving coil with the magnet track. The Self-Sensing start-up method applies current to the motor stator during the initial Starting state to force the rotor to the Null position and thereby achieve proper commutation alignment. Once aligned, commutation is maintained via position signals from the motor mounted feedback device, Feedback 1. This method is used when there is no absolute feedback available to align the motor, e.g. a motor equipped with an incremental encoder.	Enumeration: 0 = From Feedback Type (R) 1 = UVW (O) 2 = Digital (O) 3 = Self-Sense (O) 4 to 255 = (Reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
561	Required – CE (PM Motor)	Set		Commutation Offset	REAL	A value that specifies the commutation offset of the PM motor mounted feedback device in units of electrical degrees. This attribute specifies the offset from a commutation reference position defined by applying DC current into the A terminal and out of the shorted B and C terminals of the motor and allowing the rotor to move to its magnetic null position relative to the stator. On an absolute encoder or resolver, the offset is the difference from the device's zero absolute position and the commutation reference position. On an incremental encoder or Hall sensor with UVW signals, the offset is the difference between the position corresponding to a transition of the commutation device's W (S3) channel with the U (S1) channel high and the V (S2) channel low, and the commutation reference position. The commutation offset is only applicable to the motor mounted Feedback 1 device. When the optional Commutation Alignment attribute is supported and set to Controller Offset, the drive shall apply the Commutation Offset value from the controller to determine the electrical angle of the motor. In this case, a valid Commutation Offset value shall be established by the controller. In the unusual case where the commutation offset is also stored in the motor, and differs significantly from Commutation Offset value from the controller, the drive shall transition to the Start Inhibited state. If the Commutation Alignment attribute is not set to Controller Offset, the Commutation Offset value from the controller is ignored by the drive and the drive shall determine its internal commutation offset value by other means. Without a valid commutation offset, the drive shall be Start Inhibited.	Electrical Degrees
562	Optional – CE (PM Motor)	Set		Commutation Self-Sensing Current	REAL	When a PM motor feedback drive device is an incremental encoder without UVW tracks for commutation, a Self-Sensing algorithm shall be run during the Starting state that determines the Commutation Offset to apply to the position feedback. This algorithm applies a current to the motor stator to orient the rotor to establish the motor commutation phasing.	% Motor Rated
563	Optional – CE (PM Motor)	Set		Commutation Polarity	USINT	When a PM motor is using UVW signals for commutation start up, it is critical that the UVW phases of the commutation device follow the phasing of the motor. Normal polarity implies UVW phasing according to factory specification when the commutation device is moving in the factory defined positive direction. Inverted polarity effectively switches the UVW phasing to UWV thus reversing the directional sense of the commutation device. If it is determined via a Commutation Test that the phasing of the motor and the phasing of the commutation device have opposite polarity, this attribute can be used to compensate for the mismatch.	Enumeration: 0 = Normal Polarity 1 = Inverted Polarity 2 to 255 = (reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
564	Optional – CE (PM Motor)	Set		Commutation Alignment	USINT	This enumerated parameter is set to Controller Offset (1) when the motor mounted absolute feedback device is to be aligned with the stator windings of the PM motor according to the Commutation Offset value. In some cases the Commutation Offset can be preset to a value established by factory alignment of the motor feedback device relative to the motor stator windings. A setting of Not Aligned (0) indicates that the motor is not aligned, and that the Commutation Offset value is not valid. If the Commutation Offset is not valid, it cannot be used by the drive to determine the commutation angle. Any attempt to enable the drive with an invalid commutation angle shall result in a Start Inhibit condition. Alignment can be achieved via a Commutation Test that measures and sets the Commutation Offset for the motor or by direct user entry. If this attribute is set to Motor Offset (2) the drive derives the commutation offset directly from the motor. If set to Self-Sense (4) the drive automatically measures the commutation offset when it transitions to the Starting state for the first time after a power cycle. This generally applies to a PM motor equipped with a simple incremental feedback device.	Enumeration: 0 = Not Aligned (R) 1 = Controller Offset (R) 2 = Motor Offset (O) 3 = Self-Sense (O) 4 to 255 = Reserved

7.3.9.6 Frequency Control attributes

The attributes tables in Table 83 and Table 84 contain all related attributes associated with the Frequency Control method of operation of a Motion Device Axis Object instance.

Table 83 – Frequency Control Signal attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
565	Required – F	Get		Slip Compensation	REAL	Indicates the actual amount of slip compensation currently being applied.	r/min

Table 84 – Frequency Control Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
570	Required – F	Set		Frequency Control Method	SINT	The Basic Volts/Hertz control method applies voltage to the motor generally in direct proportion to the commanded frequency or speed.	Enumeration 0 = Basic Volts/Hertz (R) 1 to 127 = reserved 128 to 255 = vendor specific
572	Required – F	Set		Maximum Voltage	REAL	Sets the highest phase-to-phase voltage the drive device can output.	Volts (RMS)
573	Required – F	Set		Maximum Frequency	REAL	Sets the highest frequency the drive device can output.	Hertz
575	Required – F	Set		Break Voltage	REAL	Sets the phase-to-phase output voltage of the drive device at the Break Frequency where boost ends.	Volts (RMS)
576	Required – F	Set		Break Frequency	REAL	Sets the output frequency of the drive device at the Break Voltage where boost ends.	Hertz
577	Required – F	Set		Start Boost	REAL	Sets phase-to-phase voltage boost level for starting and accelerating.	Volts (RMS)
578	Required – F	Set		Run Boost	REAL	Sets the phase-to-phase voltage boost level for steady-state speed or deceleration.	Volts (RMS)

7.3.9.7 Drive Output attributes

The attributes table in Table 85 contains all inverter output related attributes associated with a Motion Device Axis Object instance.

Table 85 – Drive Output attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
600	Required – F Optional – C	Get		Output Frequency	REAL	Output frequency applied to motor. The frequency value is in terms of electrical cycles.	Hertz
601	Required – D	Get		Output Current	REAL	Output current applied to motor.	Amperes (RMS)
602	Required – D	Get		Output Voltage	REAL	Phase-to-phase output voltage applied to motor.	Volts (RMS)
603	Required – D	Get		Output Power	REAL	Output power of the motor. This value is based on the product of the Torque Reference signal and the Velocity Feedback.	kW
604	Optional – D	Set		PWM Frequency	UINT	Sets the carrier frequency for the Pulse Width Modulation output to the motor. Drive derating is required at higher PWM frequencies due to switching losses. Current loop update time is tied directly to the PWM frequency so loop performance generally increases with increasing PWM rate.	Hertz

7.3.10 Stopping & Braking attributes

7.3.10.1 General

The attributes table in Table 86 contains all active stopping and braking related attributes associated with a Motion Device Axis Object instance.

Table 86 – Stopping/Braking attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
610	Required – D	Set		Stopping Action	USINT	When disabling or aborting an axis, via a Disable Request or an Abort Request, this value determines the stopping method to apply to the motor. The final state after the stopping method is applied is the Stopped state. In the Stopped state, the device's inverter power structure shall either be Disabled (Disable selection) and free of torque or actively held (Hold selection) in a static condition. This attribute does not, in any way, determine the stopping actions applied in response to fault conditions.	See Semantics in 7.3.10.2.1
611	Required – C	Set		Stopping Torque	REAL	When disabling or aborting an axis, this value determines the maximum amount of torque producing current available to stop the motor when the Stopping Action is set to Current Decel.	% Motor Rated
612	Optional – C	Set		Stopping Time Limit	REAL	When disabling or aborting an axis, this parameter determines the maximum amount of time the drive allows to reach zero speed.	seconds
613	Optional – D (PM Motor)	Set		Resistive Brake Contact Delay	REAL	When an external resistive brake is used, the Resistive Brake Contact Delay can be set to delay the enabling of the device power structure until after the resistive brake has had time to connect the motor to the drive device.	seconds See Semantics in 7.3.10.2.2 for details.
614	Optional – D	Set		Mechanical Brake Control	USINT	This attribute governs the operation of the drive's Mechanical Brake Output that controls the mechanical brake mechanism. When set to Automatic, the Mechanical Brake is under the control of the axis state machine. The sequencing for the brake is described in detail by the Mechanical Brake Engage Delay and Mechanical Brake Release Delay attributes. When set to Brake Release, the brake is unconditionally released, and no longer under the control of the axis state machine.	Enumeration: 0 = Automatic 1 = Brake Release 2 to 255 = (reserved)
615	Optional – D	Set		Mechanical Brake Release Delay	REAL	When enabling the axis, the Mechanical Brake Release Delay value determines the amount of time the device shall delay transition from the Starting state to the Running or Testing states. This delay prevents any commanded motion of the motion axis until the external mechanical brake has had enough time to disengage.	seconds See Semantics in 7.3.10.2.3 for details.
616	Optional – D	Set		Mechanical Brake Engage Delay	REAL	When disabling the motion axis, the Mechanical Brake Engage Delay value determines the amount of time the device power structure shall remain enabled after the axis has decelerated to stop. This allows time for an external mechanical brake to engage. The configured Stopping Action determines the type of stopping sequence applied.	seconds See Semantics in 7.3.10.2.4 for details.
608	Optional – D	Set		Zero Speed	REAL	This attribute sets the speed threshold associated with the zero speed criteria of the stop sequence. Zero Speed is specified as a percent of motor rated speed. When Zero Speed Time attribute is supported, this attribute sets the speed threshold where the zero speed timer starts. When the axis speed has been below the Zero Speed threshold for Zero Speed Time the axis has satisfied the zero speed criteria. In all but Category 2 stops, this results in action to engage the mechanical brake. If this attribute is not supported, the zero speed threshold is left to the vendor's discretion and typically set to 1% of motor rated speed. Axis speed in the above description is based on the Velocity Feedback signal, or in the case of a Frequency Control drive, axis speed is based on Velocity Reference signal.	% Motor Rated

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
609	Optional – D	Set		Zero Speed Time	REAL	This attribute sets the amount of time that the axis speed shall be below the zero speed threshold, set by the Zero Speed attribute or established by the drive vendor, before satisfying the zero speed criteria. In all but Category 2 stops, this results in action to engage the mechanical brake. If this attribute is not supported, the amount of time needed to satisfy the zero speed criteria is left to the vendor's discretion and typically immediate (0). Axis speed in the above description is based on the Velocity Feedback signal, or in the case of a Frequency Control drive, axis speed is based on Velocity Reference signal.	seconds
590	Optional – D	Set		Proving Configuration	USINT	This attribute enables the operation of the drive's Torque Proving and Brake Proving functions that work in conjunction with mechanical brake control. When Proving is enabled, the mechanical brake shall be set as soon as the drive is disabled. When the brake is under the control of the axis state machine this is automatic. But when controlled externally, failure to set the brake when the drive is disabled can cause a free fall condition on a vertical application. When enabled, the drive performs a Torque Prove test of the motor current while in the Starting state to "prove" that current is properly flowing through each of the motor phases before releasing the brake. Should the Torque Prove test fail, a Motor Phase Loss exception is generated. While Torque Proving functionality is applicable to drive Control Modes that are not capable of generating reliable holding torque based on a feedback device, e.g. Frequency Control and Sensorless Velocity Control, it should not be used in applications where holding torque is critical to safe operation, such as in a typical lift or crane application. If the optional Brake Test Torque attribute is supported, the Torque Prove test also includes a proactive Brake Test to insure the mechanical brake is functioning properly. Should the Brake Test detect brake slip, a Brake Slip exception is generated. When Proving is enabled, the drive also performs a Brake Prove test while in the Stopping or Aborting states to "prove" proper mechanical brake function before the drive power structure is disabled. Should the Brake Prove test detect brake slip a Brake Slip exception is generated. Unless another vendor specific method is used to address a Brake Slip condition in the Stopping or Aborting state, the appropriate Fault Action for the Brake Slip exception is Torque Limited Stop and Hold. This Fault Action applies holding torque to arrest the brake slip and transitions the axis to the Major Faulted state. In general, Brake Proving functionality is only applicable to drive Control Modes that are capable of generating holding torque based on a feedback device. Brake Proving is therefore is not applicable to Frequency Control or Sensorless Velocity Control modes. The sequencing of the torque and brake "prove" tests are described in detail by the Mechanical Brake Engage Delay and Mechanical Brake Release Delay attribute semantic sections below.	Enumeration: 0 = Disabled 1 = Enabled 2 to 255 = (reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
591	Optional – D	Set		Torque Prove Current	REAL	This attribute sets the percent of motor rated current applied to the motor by the Torque Prove test as part of the Torque Proving function executed in the Starting state. The Torque Prove test applies current to the motor to "prove" that current is properly flowing through each of the motor phases before releasing the brake.	% Motor Rated
592	Optional – D E	Set		Brake Test Torque	REAL	This attribute sets the percent of motor rated torque applied to the motor by the Brake Test as part of the Torque Proving function executed in the Starting state. This Brake Test proactively tests the ability of the mechanical brake to hold the maximum anticipated load before releasing the brake and allowing operation. Should the Brake Test detect brake slip, a Brake Slip exception is generated. If the Brake Test Torque attribute value is 0 the Brake Test shall not be performed in the Starting state.	% Motor Rated
593	Optional – D E	Set		Brake Prove Ramp Time	REAL	This attribute determines the amount of time the drive shall take to ramp the applied torque of the motor down to zero during the Brake Proving test in the Stopping or Aborting state. The Brake Prove Ramp Time determines the ramp down rate of the applied torque output by dividing the Torque Limit by the Brake Prove Ramp Time. The Torque Limit in this case is the maximum of the configured Torque Limit Positive and Torque Limit Negative values. The Brake Prove test is performed to check for brake slip before the power structure is disabled.	seconds
594	Optional – D E	Set		Brake Slip Tolerance	REAL	This attribute determines the amount of brake slip allowed after the brake is engaged. If this tolerance is exceeded while the brake is engaged, a Brake Slip exception is generated. Brake slip can therefore be monitored in any axis state where the brake is engaged.	Position Control Units

7.3.10.2 Semantics

7.3.10.2.1 Attribute No. 610 – Stopping Action

When disabling or aborting an axis, this attribute determines the stopping method to apply to the motor (see Table 87). The final state after the stopping method is applied is the Stopped state. In the Stopped state the device's inverter power structure shall either be Disabled (Disable selection) and free of torque or actively held (Hold selection) in a static condition. This attribute has no impact or relationship to the planner generated acceleration and deceleration profiles.

Table 87 – Stopping Action enumeration definitions

Enum.	Req./Opt.	Name	Description
0	R/D	Disable & Coast	Disable & Coast immediately disables the device power structure and active control loops, which causes the motor to coast unless some form of external braking is applied. This is equivalent to an IEC 60204-1 Category 0 Stop.
1	R/C O/F	Current Decel & Disable	Current Decel & Disable leaves the power structure and any active control loops enabled while stopping. If configured for position control mode, the drive forces the position reference to hold its current value until the axis reaches zero speed. Once at zero speed the position reference is immediately set equal to the actual position to hold the axis at standstill. If in velocity control mode, the drive forces the velocity reference to zero. In either case, forcing the position or velocity reference signals to a fixed value results in a rapid increase in torque reference of the moving axis that saturates the torque producing current of the drive to the configured Stopping Torque that brings the motor to a stop. Once stopped, or the configured Stopping Time limit expires, the drive disables the power structure and control loops. This stop mode complies with the IEC 60204-1 Category 1 Stop. In frequency control mode the Current Vector Limit attribute, rather than the Stopping Torque attribute, is used to regulate the stopping current.
2	O/FV	Ramped Decel & Disable	Ramped Decel & Disable also leaves the power structure and any active control loops enabled while stopping but uses the Ramp Generator associated with the Velocity Fine Command Generator block to decelerate the motor to a stop. When initiating a Ramped Decel & Disable Stop, the Ramp Generator is immediately activated and the drive no longer follows command from the controller. The Ramp Generator input is initialized to zero and the output is initialized to the current speed of the motor, thus causing the Ramp Generator output to ramp the motor from its current speed down to zero according to the ramp control parameters. Once stopped, or the configured Stopping Time or factory timeout limit expires, the device disables the power structure and control loops. This stop mode also complies with the IEC 60204-1 Category 1 Stop.
3	O/PV	Current Decel & Hold	Current Decel & Hold behaves like Current Decel & Disable, but leaves the power structure active with holding torque to maintain the stopped condition. The method for generating holding torque is left to the drive vendor's discretion. This stop mode complies with the IEC 60204-1 Category 2 Stop.
4	O/PV	Ramped Decel & Hold	Ramped Decel & Hold behaves like Ramped Decel & Disable, but leaves the power structure active with holding torque to maintain the stopped condition. The method for generating holding torque is left to the drive vendor's discretion. This stop modes also complies with the IEC 60204-1 Category 2 Stop.
5-127		(Reserved)	-
128-255		(Vendor Specific)	-

7.3.10.2.2 Attribute No. 613 – Resistive Brake Contact Delay

When an external resistive brake is used, an external contactor switches the ABC motor leads from the inverter power structure to an energy dissipating resistor to stop the motor. This switching does not occur instantaneously and so enabling the power structure too early can cause electrical arcing across the contactor. To prevent this condition, the Resistive Brake Contact Delay can be set to the maximum time that it takes to fully close the contactor across the ABC motor lines so when the axis is enabled, the inverter power structure is not enabled until after the Resistive Brake Contact Delay Time has expired. Resistive Brake operation is only applicable to PM Motor types. The following sequence further defines how the Resistive Brake Contact Delay factors into the overall Enable Sequence that may also include the operation of a Mechanical Brake.

Enable Sequence:

1) Switch to Starting state.
2) Activate Resistive Brake contactor to connect motor to inverter power structure.
3) Wait for “Resistive Brake Contact Delay” while Resistive Brake contacts close.
4) Enable inverter power structure.
5) Activate Mechanical Brake output to release brake.
6) Wait for “Mechanical Brake Release Delay” while brake releases.
7) Transition to Running state.

7.3.10.2.3 Attribute No. 615 – Mechanical Brake Release Delay

When enabling the axis with an engaged mechanical brake, the Mechanical Brake Release Delay value determines the amount of time the drive shall delay transition from the Starting state to the Running or Testing states. This delay prevents any commanded motion of the motion axis until the external mechanical brake has had enough time to disengage. If supported, a Torque Proving operation is included in this sequence prior to releasing the brake.

Enable Sequence:

1) Switch to Starting state.
2) Activate Resistive Brake contactor to connect motor to inverter power structure.
3) Wait for “Resistive Brake Contact Delay” while Resistive Brake contacts close.
4) Enable inverter power structure.
5) Perform (optional) Torque Proving operation to verify motor control of load.
6) Activate Mechanical Brake output to release brake.
7) Wait for “Mechanical Brake Release Delay” while brake releases.
8) Transition to Running (or Testing) state.

7.3.10.2.4 Attribute No. 616 – Mechanical Brake Engage Delay

7.3.10.2.4.1 General

When disabling the motion axis using a Category 1 Stopping Action, the Mechanical Brake Engage Delay value determines the amount of time the inverter power structure shall remain enabled after the axis has decelerated to a stop. This allows time for an external mechanical brake to engage. The configured Stopping Action determines the type of stopping sequence applied. If supported, a Brake Proving operation is included in the Category 1 stopping sequence prior to disabling the power structure.

7.3.10.2.4.2 Disable Sequence (Category 0 Stop)

Inverter is immediately disabled. Brake Proving is not applicable. The corresponding sequence is specified below and illustrated in Figure 65.

1) Switch to Stopping state.
2) Disable inverter power structure.
3) Wait for zero speed (see 7.3.10.2.4.5) or "Stopping Time Limit" or a factory set timeout, whichever occurs first.
4) Transition to Stopped state.
5) Deactivate Mechanical Brake output to engage brake.
6) Deactivate Resistive Brake contactor to disconnect motor from inverter power structure.

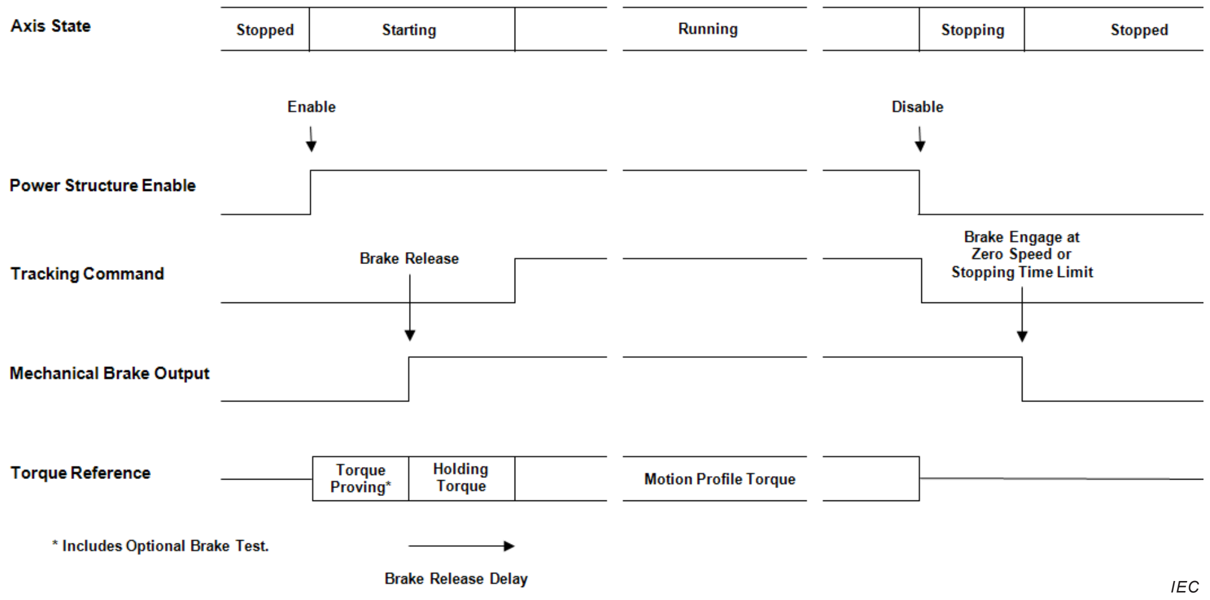

Figure 65 – Brake Control Sequence (Category 0 Stop)

7.3.10.2.4.3 Disable Sequence (Category 1 Stop)

Torque is applied to stop motor. Brake Proving is applicable. The corresponding sequence is specified below and illustrated in Figure 66.

1) Switch to Stopping state.
2) Apply “Current Decel” or “Ramp Decel” method to stop motor.
3) Wait for zero speed (see 7.3.10.2.4.5) or “Stopping Time Limit” or factory set timeout, whichever occurs first.
4) Deactivate Mechanical Brake output to engage brake.
5) Wait for “Mechanical Brake Engage Delay” while brake engages.
6) Perform (optional) Brake Proving operation to verify brake control of load.
7) Disable inverter power structure.
8) Transition to Stopped state.
9) Deactivate Resistive Brake contactor to disconnect motor from inverter power structure.

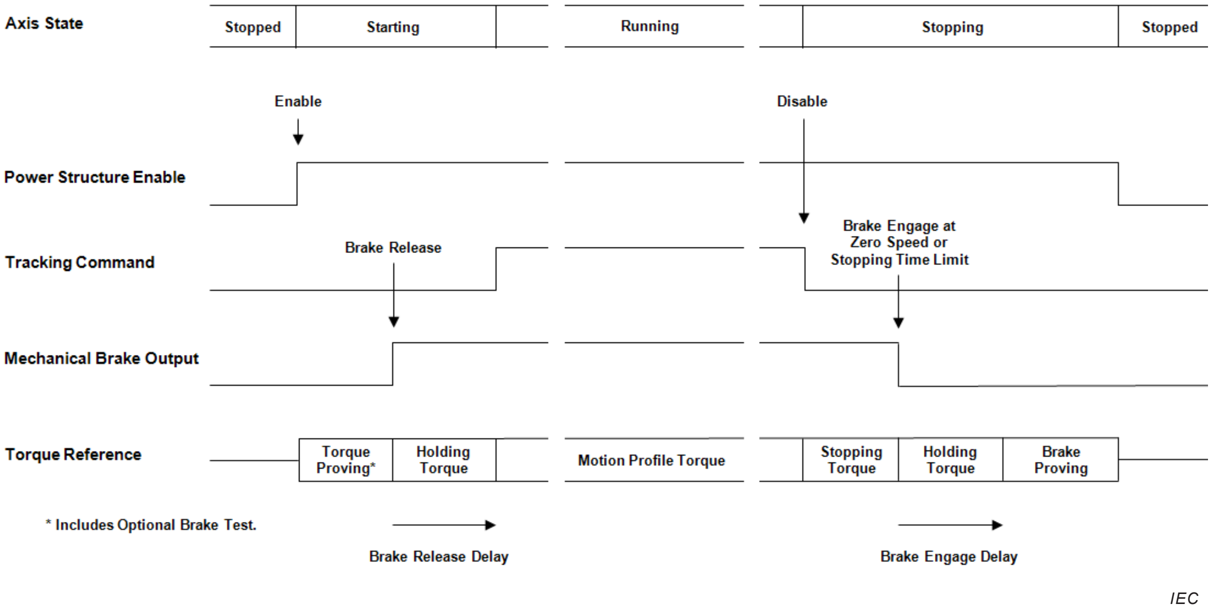

Figure 66 – Brake Control Sequence (Category 1 Stop)

7.3.10.2.4.4 Disable Sequence (Category 2 Stop)

The mechanical brake is not used. Brake Proving is not applicable. The corresponding sequence is specified below and illustrated in Figure 67.

1) Switch to Stopping state.
2) Apply “Current Decel” or “Ramp Decel” method to stop motor.
3) Wait for zero speed (see 7.3.10.2.4.5) or “Stopping Time Limit" or factory set timeout, whichever occurs first.
4) Transition to Stopped state.

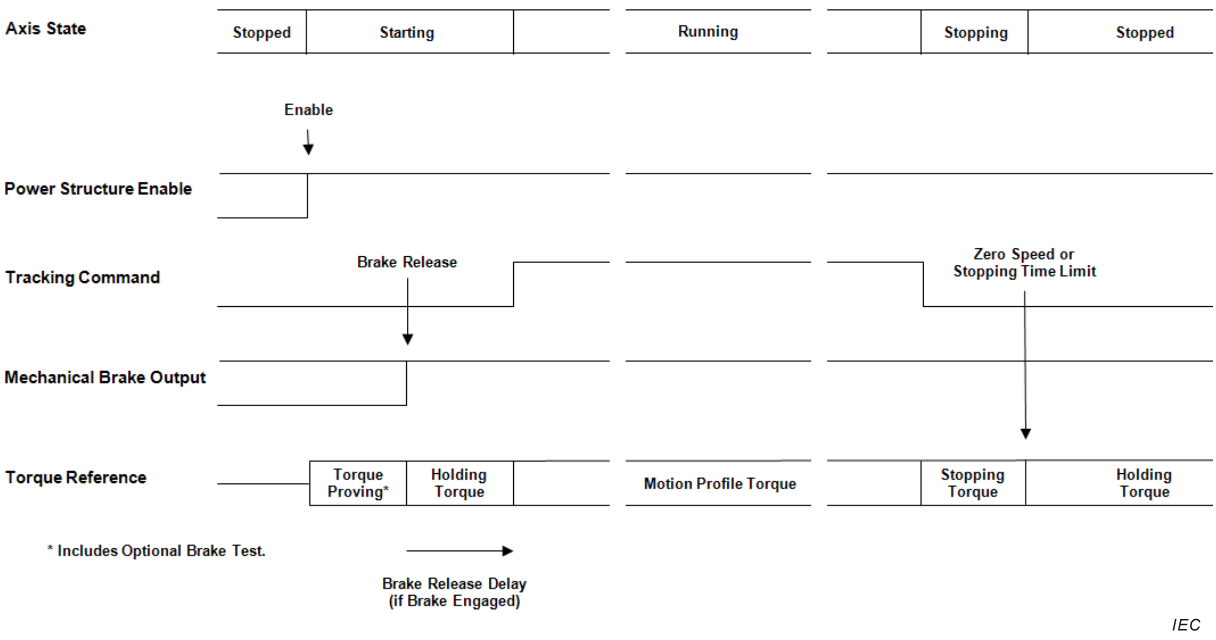

Figure 67 – Brake Control Sequence (Category 2 Stop)

7.3.10.2.4.5 Zero speed criteria

Recommended criteria for zero speed is based on Velocity Feedback, or in the case of a Frequency Control drive, based on Velocity Reference. Zero speed criteria can be established explicitly via optional Zero Speed and Zero Speed Time attributes or implicitly as 1 % of motor rated speed or left to the drive vendor’s discretion.

7.3.10.2.5 Attribute No. 590 – Proving Configuration

The Proving feature includes a number of optional sub-features, many of which depend on support of other Proving feature attributes. Table 88 defines these attribute dependencies.

Table 88 – Proving sub-feature attribute dependencies

Proving sub-feature	Controlling attributes	Attribute prerequisites
Torque Prove		Proving Configuration
Brake Test	Brake Test Torque Brake Slip Tolerance	Proving Configuration
Brake Prove	Brake Prove Ramp Time Brake Slip Tolerance	Proving Configuration

Proving tests are performed when enabling or disabling the drive axis. During these state transitions a series of operations are performed by the drive to insure the proper function of the motor (Torque Proving) and the brake (Brake Proving). The flow charts in Figure 68 Figure 67and Figure 69 define these operational sequences in the context of a drive enable transition and a drive disable or abort transition.

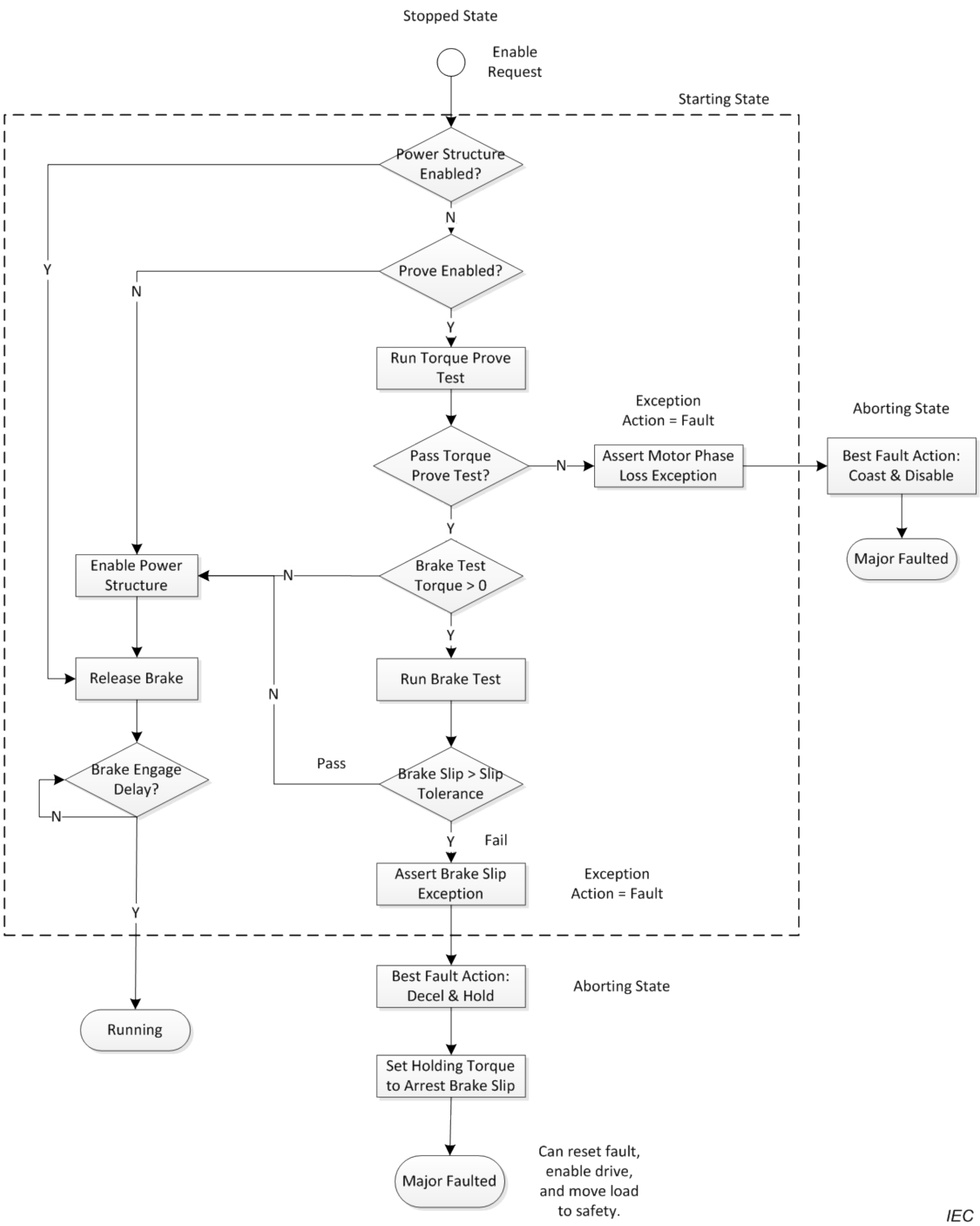

Figure 68 – Drive Enable sequence with Proving feature

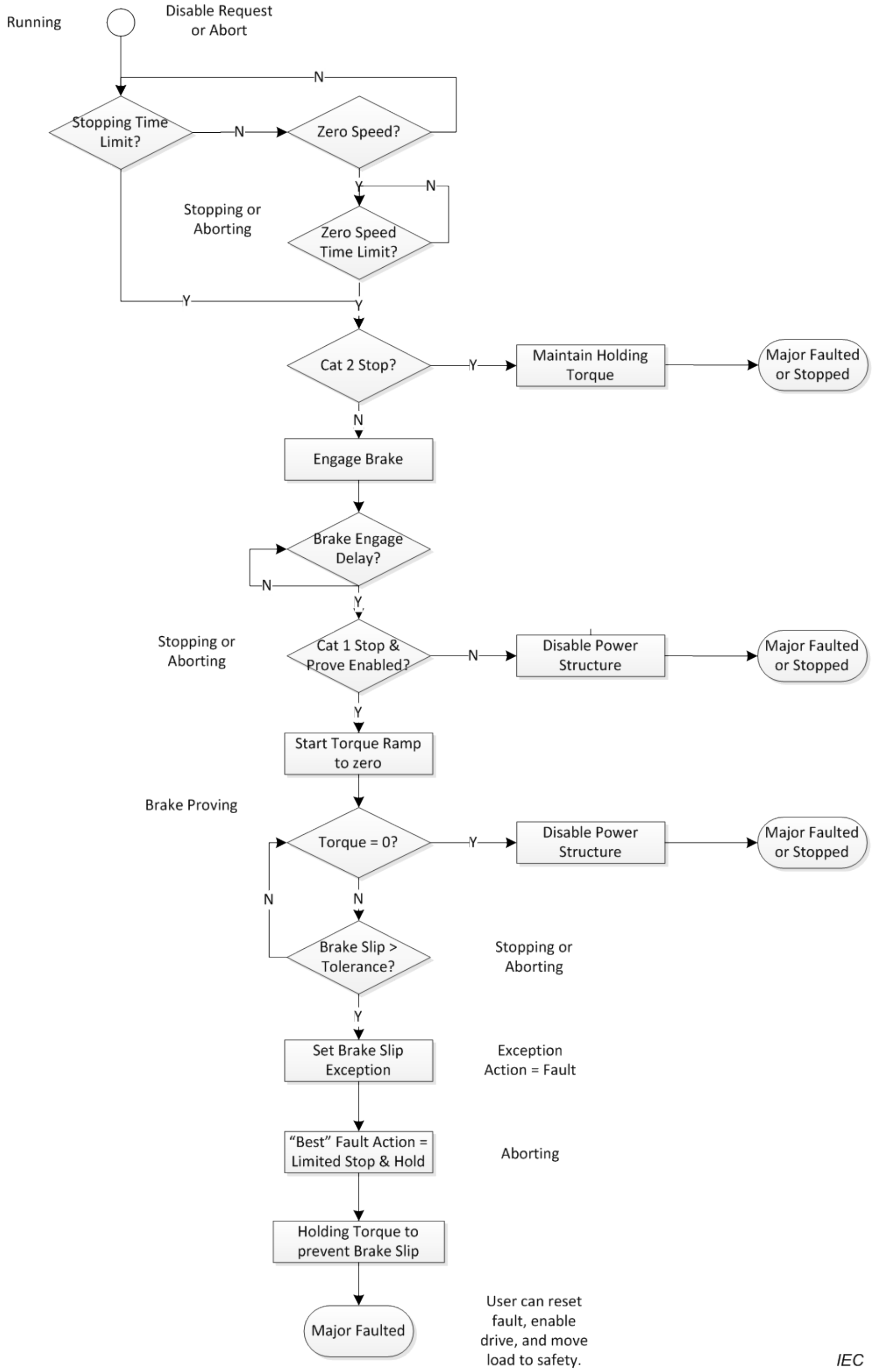

Figure 69 – Drive Disable sequence with Proving feature

7.3.11 DC Bus Control attributes

The attributes table in Table 89 contains Motion Device Axis Object instance attributes associated with DC Bus control including functionality to address both under-voltage and over-voltage conditions.

Table 89 – DC Bus Control attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
620	Required – BD	Get		DC Bus Voltage	REAL	Measured DC Bus Voltage.	Volts
621	Required – BD	Get		DC Bus Voltage – Nominal	REAL	Normal DC Bus Voltage during operation as determined by averaging the DC Bus Voltage over a device specific time interval. This value is used as the basis for Bus Overvoltage and Undervoltage limits. NOTE If the device does not support this bus voltage averaging concept, the alternative is to hard code this value.	Volts
622	Optional – BD	Set		Bus Configuration	USINT	This enumerated selection indicates how the DC Bus is going to be used. Standalone indicates that DC Bus power supplied by the drive's converter section is applied only to this drive's power structure. Shared AC/DC indicates that the converter associated with this CIP Motion device is to supply and share DC Bus power with other drives. This would typically result in de-rating of the continuous current rating for the Shared AC/DC drive. Shared DC indicates that this drive is sharing DC bus power generated by another Shared AC/DC drive. Shared DC – Non CIP Converter indicates that this drive is receiving DC bus power generated by an external AC/DC converter that is not CIP Motion compliant and distributing its DC bus power to other CIP Motion drives. A drive configured for Shared DC – Non CIP Converter is responsible for communicating the status of the external converter to the control system as if the external converter were integrated with the drive. Specifically, this communication includes the Bus Up and DC Bus Unload status bits reflecting the current state of associated external converter.	Enumeration: 0 = Standalone 1 = Shared AC/DC 2 = Shared DC 3 = Shared DC – Non-CIP Converter 4 to 255 = (reserved)
623	Optional – D	Set		Bus Voltage Select	USINT	This value indicates the expected bus voltage level of the drive application. High bus voltage selection is usually associated with drive running on the North American power grid, while operating in Europe a Low Bus Voltage selection would be appropriate. This parameter can be used to compensate for these different bus voltage levels in the current loop.	Enumeration: 0 = High (115 V, 230 V, 460 V) 1 = Low (100 V, 200 V, 400 V) 2 to 255 = (reserved)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
624	Required – BD (Integral Converter)	Set		Bus Regulator Action	USINT	Controls the method of operation of the DC Bus Regulator that addresses the regenerative over-voltage conditions that can occur when decelerating a motor. If Disabled, no regulation is applied to the DC Bus level to control regenerative energy sourced by the motor. This selection is only valid when applied to an Induction Motor where the motor may be allowed to coast to a stop. When Shunt Regulator is selected the associated shunt regulation hardware is applied to the DC Bus to dissipate regenerative energy via an internal or external resistor.	Enumeration: 0 = Disabled (R) 1 = Shunt Regulator (R) 3 to 127 = (reserved) 128 to 255 = (vendor specific)
625	Optional – D (Integral Converter)	Set		Regenerative Power Limit	REAL	Limits the amount of power allowed to transfer between the motor and the DC Bus during regenerative braking of the motor load. When using an external shunt resistor, set this value to its maximum value. Since this is regenerative power, the value of the limit is negative.	% Motor Rated
627	Optional – D	Set		Power Loss Action	USINT	Set the reaction to a DC Bus under-voltage condition when the DC Bus drops below a hard-coded threshold in the device or the configured Power Loss Threshold. This provides a specific (configured) response to an incoming power loss while the drive/motor is running. A continue action selection configures the drive to ignore the power loss condition and continue to run for as long as possible. A Bus Undervoltage exception may occur if the DC Bus Voltage falls below the Factory or User Limits. Otherwise, operation will continue until the low voltage power supplies drop out. There may be concerns operating the power structure below the point where the gate drives start to loose power where this could possibly result in drive damage. The Bus Undervoltage Exception Actions shall be set accordingly. A coast thru action selection configures the drive to zero the PWM output of the drive while leaving the axis in the Running state. If the incoming power returns before the timeout period, given by the Power Loss Time, the drive automatically starts to control the motor again. If, however, the power doesn't return before Power Loss Timeout period expires, a Bus Power Loss Exception is generated. The decel regen action selection configures the drive to regeneratively charge the DC bus by decelerating the motor using the bus regulator to regulate the bus voltage at a predetermined level. When incoming power is restored the drive returns to normal operation. If, however, the drive reaches zero speed or the Power Loss Time period expires before the incoming power has restored, the drive power structure is disabled and a Bus Power Loss exception is generated.	Enumeration: 0 = Continue (Ignore) (R) 1 = Coast Thru (R) 2 = Decel Regen (O) 3 to 127 = (reserved) 128 to 255 = (vendor specific)
628	Optional – BD	Set		Power Loss Threshold	REAL	Sets the Level for Power Loss as percent of nominal DC Bus Voltage. If this value is 0, the hard-coded threshold is used.	%

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
629	Optional – BD	Set		Shutdown Action	USINT	Selects the device's action when transitioning to the Shutdown state as a result of a Shutdown Request or in response to an Exception Action where the best available stopping action is Shutdown. Default action is to immediately disable the device's power structure. If Drop DC Bus is selected, action can be taken to drop the DC Bus voltage as well. This is generally done by opening an AC Contactor Enable output provided by the drive that controls power to the converter.	Enumeration: 0 = Disable (R) 1 = Drop DC Bus (O) 2 to 127 = (reserved) 128 to 255 = (vendor specific)
630	Optional – BD	Set		Power Loss Time	REAL	Sets the timeout value before a Bus Power Loss Exception is generated by the drive in response to a Power Loss condition. See Power Loss Action for details.	seconds

7.3.12 Power and thermal management attributes

The attribute tables in Table 90 and Table 91 contain all power and thermal related attributes associated with a Motion Device Axis Object instance.

Table 90 – Power and Thermal Management Status attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
634	Optional – BD	Get		Integral Converter	BOOL	This boolean value indicates whether the axis is associated with an integral converter (power supply) that provides DC Bus Power and Bus Regulation.	0 = No 1 = Yes
635	Required – D	Get		Motor Capacity	REAL	Real-time estimate of the continuous rated motor thermal capacity utilized during operation based on the motor thermal model. A value of 100 % would indicate that the motor is being used at 100 % of rated thermal capacity as determined by the continuous current rating of the motor.	% Motor Rated
636	Required – D	Get		Inverter Capacity	REAL	Real-time estimate of the continuous rated inverter thermal capacity utilized during operation based on the inverter thermal model. A value of 100 % would indicate that the inverter is being used at 100 % of rated thermal capacity as determined by the continuous current rating of the inverter.	% Inverter Rated
637	Optional – BD (Integral Converter)	Get		Converter Capacity	REAL	Real-time estimate of the continuous rated converter thermal capacity utilized during operation based on the converter thermal model. A value of 100 % would indicate that the converter is being used at 100 % of rated thermal capacity as determined by the continuous current rating of the converter.	% Converter Rated
638	Optional – BD (Integral Converter)	Get		Bus Regulator Capacity	REAL	Real-time estimate of the continuous rated bus regulator thermal capacity utilized during operation based on the bus regulator thermal model. A value of 100 % would indicate that the bus regulator is being used at 100 % of rated thermal capacity as determined by the continuous current rating of the bus regulator.	% Regulator Rated
639	Optional – BD	Get		Ambient Temperature	REAL	Current internal ambient temperature of the device enclosure.	°C
640	Optional – D	Get		Inverter Heatsink Temperature	REAL	Current temperature of the device’s inverter heatsink, typically based on an embedded temp sensor.	°C
641	Optional – D	Get		Inverter Temperature	REAL	Current temperature of the power block used in the inverter’s power structure, sometime referred to as the semiconductor junction temperature.	°C

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
642	Optional – D	Get		Motor Temperature	REAL	Current temperature of the motor stator, typically based on an embedded temp sensor.	°C
643	Optional – E	Get		Feedback 1 Temperature	REAL	Current temperature of the Feedback 1 device.	°C
644	Optional – E	Get		Feedback 2 Temperature	REAL	Current temperature of the Feedback 2 device.	°C
645	Optional – D	Get		Inverter Overload Limit	REAL	Factory set maximum limit for Inverter Capacity that when exceeded triggers the selected Inverter Overload action.	% Inverter Rated

Table 91 – Power and Thermal Management Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
646	Optional – D	Set		Motor Overload Action	USINT	Selects the device's response to a motor overload condition based on an I2t based motor thermal model. The motor overload condition occurs when the motor thermal model indicates that the Motor Capacity has exceeded the Motor Overload Limit. This motor overload action functionality is independent of the motor overload exception action functionality. No explicit action is taken by the device in the overload condition if None is the selected overload action. Selecting the Current Foldback action, however, results in a reduction of the current reference via the Motor Thermal Current Limit attribute value that is reduced in proportion to the percentage difference between Motor Capacity and the Motor Overload Limit.	Enumeration : 0 = None (R) 1 = Current Foldback (O) 2 to 127 = (reserved) 128 to 255 = (vendor specific)
647	Optional – D	Set		Inverter Overload Action	USINT	Selects the device's response to an inverter overload alarm condition based on an I2t based inverter thermal model. The inverter overload alarm condition occurs when the inverter thermal model indicates that the Inverter Capacity has exceeded the Inverter Overload Limit. The Inverter Overload Action provides opportunities to mitigate the overload condition without stopping operation. This inverter overload action functionality is independent of the motor overload exception action functionality. No explicit action is taken by the device in the overload condition if None is the selected overload action. Selecting the Current Foldback action, however, results in a reduction of the current command via the Inverter Thermal Current Limit attribute value that is reduced in proportion to the percentage difference between Inverter Capacity and the Inverter Overload Limit.	Enumeration : 0 = None (R) 1 = Current Foldback (O) 2 to 127 = (reserved) 128 to 255 = (vendor specific)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
648	Optional – D	Set		Duty Select	USINT	This value indicates the duty level of the drive application and balances the continuous and intermittent overload capacity of the drive and motor accordingly. Normal Duty provides nominal continuous rating and nominal overload capacity. Heavy Duty provides highest overload capacity at the expense of a lower continuous rating. Light Duty provides highest continuous rating at the expense of lower overload capacity Specification for the continuous and overload ratings understr Normal, Heavy and Light Duty are left to the discretion of the drive vendor. Duty Select is used to determine the level of thermal protection for the motor and the inverter during drive operation.	Enumeration : 0 = Normal (R) 1 = Heavy (O) 2 = Light (O) 3 to 255 = (reserved)

7.3.13 Axis Status attributes

7.3.13.1 General

The attributes table in Table 92 contains all status attributes associated with a Motion Device Axis Object instance.

Table 92 – Axis Status attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
650	Required – All	Get		Axis State	USINT	Enumerated value indicating the state of the motion axis.	Enumeration: 0 = Initializing 1 = Pre-Charge 2 = Stopped 3 = Starting 4 = Running 5 = Testing 6 = Stopping 7 = Aborting 8 = Major Faulted 9 = Start Inhibited 10 = Shutdown 11 to 255: Reserved
651	Required – All	Get		Axis Status	DWORD	Collection of standard bits indicating the internal status of the motion axis.	See Semantics in 7.3.13.2.1
652	Required – All	Get		Axis Status – Mfg	DWORD	Collection of vendor specific bits indicating the internal status of the motion axis.	Bitmap: 0 to 31: Vendor Specific (Published in Product Manual)
653	Required – All	Get		Axis I/O Status	DWORD	The Axis I/O Status attribute is a 32-bit collection of bits indicating the state of standard digital inputs and outputs associated with the operation of this motion axis. A value of zero for a given input bit indicates a logical 0 value, while a value of 1 indicates a logical 1 value.	See Semantics in 7.3.13.2.2
654	Required – All	Get		Axis I/O Status – Mfg	DWORD	Collection of bits indicating the state of vendor specific digital inputs associated with the operation of this motion axis. A value of zero for a given input bit indicates a logical 0 value, while a value of 1 indicates a logical 1 value.	Bitmap: 0 to 31: Vendor Specific (Published in Product Manual)

7.3.13.2 Semantics

7.3.13.2.1 Attribute No. 651 – Axis Status

The Axis Status attribute is a 32-bit collection of bits indicating various internal status conditions of the axis instance, as specified in Table 93. Any status bits that are not applicable shall be set to 0.

Table 93 – Axis Status bit definitions

Bit	Req./Opt.	Name	Description
0	R	Local Control	This bit is set if axis is taking command reference and services from local interface instead of the remote (CIP Motion) interface. This bit is based on the current state of the Remote Mode bit of the Node Status attribute (see 7.2.2.2).
1	R	Alarm	This bit is set if the axis has detected one or more exception conditions configured to generate an alarm. This bit is clear if there are no current axis alarm conditions (see 7.3.13).
2	R/BD	DC Bus Up	This bit is set if DC Bus has charged up to an operational voltage level based on direct measurement and, if applicable, the Converter Bus Up Status bit associated with external converter(s) supplying DC Bus power to this device. When an axis is in the Pre-charge state, transition of the DC Bus Up status bit from 0 to 1 initiates state transition to the Stopped State (drive axis) or to the Running state (converter axis). Once set, the DC Bus Up bit is cleared when the DC Bus voltage has dropped below an operational voltage level, independent of the state of the Converter Bus Up Status bit (see 7.6.2.5).
3	R/D	Power Structure Enabled	This bit is set if the axis amplifier is energized and capable of generating motor flux and torque. The value of the Power Structure Enabled bit is determined by the Axis State (see 7.6.1) and the configured Stopping Action attribute value (see 7.3.10.2.1).
4	R/D	Motor Flux Up	This bit is set if motor flux for an induction motor has reached an operational level. Transition of the Motor Flux Up bit is initiated in the Starting State (see 7.6.2.7) according to the configured Flux Up Control attribute value (see 7.3.9.5). This bit is only applicable to Induction Motor types.
5	R/D	Tracking Command	This bit is set if the axis control structure is actively tracking the command reference from motion planner. The Tracking Command bit is directly associated with the Running state of the Axis State Model (see 7.6.2.8).
6	R/P	Position Lock	This bit is set if the actual position is within Position Lock Tolerance of command position (see 7.6.9.3).
7	R/PV	Velocity Lock	This bit is set if the velocity feedback signal is within Velocity Lock Tolerance of the unlimited velocity reference (see 7.6.9.5).
8	R/ED	Velocity Standstill	This bit is set if the velocity feedback signal is within Velocity Standstill Window of 0. For a Frequency Control drive this bit is set if the velocity reference signal is within Velocity Standstill Window of 0 (see 7.6.9.5).
9	R/ED	Velocity Threshold	This bit is set if the absolute velocity feedback signal is below Velocity Threshold. For a Frequency Control drive this bit is set if the absolute velocity reference signal is below Velocity Threshold (see 7.6.9.5.
10	R/FPV	Velocity Limit	This bit is set if the velocity reference signal is currently being limited by the Velocity Limiter (see 7.6.9.6.2).
11	R/C	Acceleration Limit	This bit is set if the acceleration reference signal is currently being limited by the Acceleration Limiter (see 7.6.9.8.2).
12	R/C	Deceleration Limit	This bit is set if the acceleration reference signal is currently being limited by the Deceleration Limiter (see 7.6.9.8.2).
13	R/C	Torque Threshold	This bit is set if the absolute filtered torque reference is above the Torque Threshold (see 7.6.9.9).
14	R/C	Torque Limit	This bit is set if the filtered torque reference is currently being limited by the Torque Limiter (see 7.6.9.9.7).
15	R/C	Current Limit	This bit is set if the command current, Iq, is currently being limited by the Current Vector Limiter (see 7.6.9.10.2).
16	R/C	Thermal Limit	This bit is set if Current Vector Limit condition of the axis is being limited by any of the axis’s Thermal Models (see 7.6.9.10.2).

Bit	Req./Opt.	Name	Description
17	R/E	Feedback Integrity	This bit, when set, indicates that the feedback device is accurately reflecting changes to axis position, and there have been no conditions detected that would compromise the quality of the feedback position value. The bit is set at power-up assuming that the feedback device passes any power-up self test required. If during operation a feedback exception occurs that could impact the fidelity of axis position, the bit is immediately cleared. The bit remains clear until either a fault reset is executed by the drive or the drive is power cycled. Note that the Feedback Integrity bit behavior applies to both absolute and incremental feedback device operation.
18	R/BD	Shutdown	This bit is set when the axis is in the shutdown state or in the faulted state but would transition to the shutdown state if the faults were cleared. Therefore, the Shutdown bit is closely associated with the Shutdown State of the Axis State Model (see 7.6.2.14).
19	R	In Process	This bit is set for the duration of an active process. An example of active process would be an operation initiated by a service request such as, Run_Motor_Test, Run_Hookup_Test, or Run_Inertia_Test. The In Process bit should be set as soon as the process is initiated and should remain set until the process is complete. An active process that requires the enabling of the axis power structure results in a transition to the Testing State of the Axis State Model (see 7.6.2.9).
20	O/BD	DC Bus Unload	This bit is set by a CIP Motion converter or a CIP Motion drive, containing an integral converter, or a CIP Motion drive connected to an external non-CIP converter, to indicate that the converter cannot continue supplying DC Bus power to other drives on a shared DC Bus. This is usually the result of a shutdown fault action initiated by the drive or converter, or a shutdown request from the controller. Thus, when the DC Bus Unload bit is set, the Shutdown bit (bit 18) should also be set. When there is no AC Contactor Enable output to drop the DC Bus, a method is needed to unload the converter from all other drives sharing the DC Bus. By monitoring the DC Bus Unload status bit, the control system can propagate a Bus Power Sharing exception to all drives on the shared DC Bus that are configured for Shared AC/DC or Shared DC operation. This Bus Power Sharing exception invokes the configured Exception Action that, by default, disables the drive power structure, thereby unloading the bus. The Bus Power Sharing Fault on these drives is a persistent fault that cannot be cleared as long as the DC Bus Unload bit is set on this originating drive or converter. The control system shall simply regenerate the Bus Power Sharing Faults based on the DC Bus Unload status bit still being set. Note that only the originating drive or converter with the DC Bus Unload condition can cause Bus Power Sharing Faults on other shared drives. In other words, no device with a Bus Power Sharing Fault can cause a Bus Power Sharing exception on other shared drives by setting its DC Bus Unload bit. This qualification prevents DC Bus recovery deadlock. To recover full DC Bus operation, the originating drive with the DC Bus Unload condition shall first be reset via a Shutdown Reset Request. Once clear, the Bus Power Sharing Faults on the shared drives can then be successfully cleared by either a Fault Reset Request, or a Shutdown Reset Request, allowing these drives to become operational.

Bit	Req./Opt.	Name	Description
21	O/BD	AC Power Loss	This bit is set when a CIP Motion converter, or a CIP Motion drive containing an integral converter, or a CIP Motion drive connected to an external non-CIP converter, has detected a loss of AC input power. This bit is cleared when AC input power is determined to be sufficient for converter operation. When an AC Power Loss condition is detected by a converter supplying power to other drives over a shared DC Bus, a method is needed to generate a Converter AC Power Loss exception on any drive whose power structure is enabled. To accomplish this, the control system monitors the AC Power Loss status bits of converters supplying DC Bus power and propagates AC Power Loss status to all drives on the shared DC Bus, i.e. drives that are configured for Shared AC/DC or Shared DC operation. Upon notification of AC Power Loss, drives that have enabled power structures shall assert a Converter AC Power Loss exception and invoke the programmed exception action. Disabled drives shall not generate an exception action on AC Power Loss. Thus, no drive faults shall occur on removal of AC Power from a converter as long as no drive power structures drawing power from that converter are enabled.
22 to 31		Reserved	-

Many of the Axis Status bits defined in Table 93 are related to the current Axis State as shown in Table 94.

Table 94 – Axis Status bit vs. Axis State

Bit	Axis Status bit name	Initializing	Pre-Charge	Shutdown	Start Inhibit	Stopped	Starting	Running	Testing	Stopping	Aborting	Major Faulted
0	Local Control	x	x	x	x	x	x	x	x	x	x	x
1	Alarm	x	x	x	x	x	x	x	x	x	x	x
2	DC Bus Up	x	0	x	1	1	1	1	1	1	1	1
3	Power Structure Enabled	0	0	0	0	0/1	1	1	1	1	1	0/1
4	Motor Flux Up (IM only)	0	0	0	0	0/1	1	1	1	1	1	0/1
5	Tracking Command	0	0	0	0	0	0	1	0	0	0	0
6	Position Lock	0	0	0	0	0/x	x	x	0	x	x	0/x
7	Velocity Lock	0	0	0	0	0/x	x	x	0	x	x	0/x
8	Velocity Standstill	0	0	0	0	0/x	x	x	0	x	x	0/x
9	Velocity Threshold	0	0	0	0	0/x	x	x	0	x	x	0/x
10	Velocity Limit	0	0	0	0	0/x	x	x	0	x	x	0/x
11	Acceleration Limit	0	0	0	0	0/x	x	x	0	x	x	0/x
12	Deceleration Limit	0	0	0	0	0/x	x	x	0	x	x	0/x
13	Torque Threshold	0	0	0	0	0/x	x	x	0	x	x	0/x
14	Torque Limit	0	0	0	0	0/x	x	x	0	x	x	0/x
15	Current Limit	0	0	0	0	0/x	x	x	x	x	x	0/x
16	Thermal Limit	x	x	x	x	x	x	x	x	x	x	x
17	Feedback Integrity	x	x	x	x	x	x	x	x	x	x	x
18	Shutdown	0	0	1	0	0	0	0	0	0	x	x
19	In Process	0	x	x	x	x	0	0	1	0	0	0
20	DC Bus Unload	0	0	x	0	0	0	0	0	0	x	x

Key:

x = 0 or 1

0/1 = 0 for Category 0 or 1 Stop, 1 for Category 2 Stop

0/x = 0 for Category 0 or 1 Stop, x for Category 2 Stop

The correspondence between Stopping Action and Stop Category is given in Table 95.

Table 95 – Stopping Action vs. Stop Category

Stopping Action enum.	Stopping Action name	IEC-60204-1
0	Disable & Coast	Category 0 Stop.
1	Current Decel & Disable	Category 1 Stop.
2	Ramped Decel & Disable	Category 1 Stop.
3	Current Decel & Hold	Category 2 Stop.
4	Ramped Decel & Hold	Category 2 Stop.

7.3.13.2.2 Attribute No. 653 – Axis I/O Status

The Axis I/O Status attribute is a 32-bit collection of bits indicating the state of standard digital inputs and outputs associated with the operation of this motion axis, as specified in Table 96.

A value of zero for a given input bit indicates a logical 0 value, while a value of 1 indicates a logical 1 value. Any status bits that are not applicable shall be set to 0.

Table 96 – Axis I/O Status bit definitions

Bit	Req./Opt.	Name
0	R	Enable Input
1	R/E	Home Input
2	R/E	Registration 1 Input
3	O/E	Registration 2 Input
4	R/P	Positive Overtravel Input
5	R/P	Negative Overtravel Input
6	O/E	Feedback 1 Thermostat
7	O/D	Resistive Brake Output
8	O/D	Mechanical Brake Output
9 to 31		Reserved

7.3.14 Exception, fault, and alarm attributes

7.3.14.1 General

The attribute table in Table 97 contains all exception, fault, and alarm related attributes associated with a Motion Device Axis Object instance. Exceptions are conditions that can occur during motion axis operation that have the potential of generating faults or alarms.

Table 97 – Exception, Fault and Alarm attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
655	Required – All	Get		Axis Exceptions	LWORD	A bit map that represents the current state of all standard exception conditions. See the Standard Exception Table bit map definition in 7.3.14.2 for detail on the bit locations. Each exception has a corresponding Axis Exception Action. Exceptions that are configured to be Ignored are only visible in this attribute.	See Semantics in 7.3.14.2.
656	Required – All	Get		Axis Exceptions – Mfg	LWORD	A bit map that represents the current state of all manufacturer specific exception conditions. See the Mfg. Exception Table published in drive product manual. Each exception has a corresponding Axis Exception Action. Exceptions that are configured to be Ignored are only visible in this attribute.	See Mfg. Exception Table (Published in Product Manual)
657	Required – All	Get		Axis Faults	LWORD	A bit map that represents the state of all standard runtime faults. The bit map is identical to that of the Axis Exceptions attribute. Fault bits when set are latched until a fault reset occurs. A fault reset clears the runtime fault bits, but the bits set again immediately if the underlying exception condition is still present. Any exceptions whose Axis Exception Action is configured to ignore or report as alarms do not appear in this attribute.	See Semantics in 7.3.14.2.
658	Required – All	Get		Axis Faults – Mfg	LWORD	A bit map that represents the state of all manufacturer specific runtime faults. The bit map is identical to that of the Mfg. Axis Exceptions attribute. Fault bits when set are latched until a fault reset occurs. A fault reset clears the runtime fault bits, but the bits set again immediately if the underlying exception condition is still present. Any exceptions whose Axis Exception Action is configured to ignore or report as alarms do not appear in this attribute.	See Mfg. Exception Table (Published in Product Manual)
659	Required – All	Get		Axis Alarms	LWORD	A bit map that represents the current state of all standard alarm conditions. The bit map is identical to that of the Std. Axis Exceptions attribute. Only exception conditions whose Axis Exception Action is configured to report as an alarm appear in this attribute, and will not be reported in the Axis Faults attribute. Alarm bits when set are not latched and will clear as soon as the underlying exception condition is corrected.	See Semantics in 7.3.14.2.

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
660	Required – All	Get		Axis Alarms – Mfg	LWORD	A bit map that represents the current state of all manufacturer specific alarm conditions. The bit map is identical to that of the Mfg. Axis Exceptions attribute. Only exception conditions whose Axis Exception Action is configured to report as an alarm appear in this attribute, and will not be reported in the Axis Faults attribute. Alarm bits when set are not latched and will clear as soon as the underlying exception condition is corrected.	See Mfg. Exception Table. (Published in Product Manual)

7.3.14.2 Semantics: Standard Exception Table

Table 98 defines a list of standard exception conditions associated with the Axis Exceptions, Axis Faults, and Axis Alarms attributes. While the Axis Exceptions, Axis Faults, and Axis Alarms attributes are all Required in the CIP Motion device implementation, support for each of the individual exception conditions therein is left Optional. The Rule column in Table 98 indicates the Device Function Codes where the associated exception is applicable.

Table 98 – Standard Exception Table

Bit	Rule	Exception	Description
0	-	-- Reserved --	This bit cannot be used since the Alarm Codes and Fault Code are defined by the associated exception bit number and an Alarm Code or Fault Code of 0 means no alarm or fault condition is present.
1	D	Motor Overcurrent	Motor current has exceeded its rated peak or instantaneous current limit.
2	D	Motor Commutation	Permanent magnet motor commutation problem detected. Example would be an illegal state “111” or “000” for a UVW commutation device.
3	D	Motor Overspeed FL	Motor speed has exceeded its maximum limit given by the Motor Overspeed Factory Limit attribute associated with the motor type.
4	D	Motor Overspeed UL	Motor speed has exceeded the user defined speed limit given by Motor Overspeed User Limit.
5	D	Motor Overtemperature FL	Motor temperature has exceeded its factory set temperature limit given by Motor Overtemperature Factory Limit, or the integral motor thermal switch has tripped.
6	D	Motor Overtemperature UL	Motor temperature has exceeded the user defined temperature limit given by Motor Overtemperature User Limit.
7	D	Motor Thermal Overload FL	Motor thermal model has exceeded its factory set thermal capacity limit given by Motor Thermal Overload Factory Limit.
8	D	Motor Thermal Overload UL	Motor thermal model has exceeded its user defined thermal capacity given by Motor Thermal Overload User Limit.
9	D	Motor Phase Loss	The current in one or more motor phases has been lost or has been below a factory set threshold or, if supported, the configured Motor Phase Loss Limit. This exception is also associated with the optional Torque Prove function that tests motor current against an engaged mechanical brake. During normal operation in the Running state, the motor phase loss test cycles through the three motor currents checking that current in each motor phase exceeds the threshold level. When the phase being checked exceeds the level, the check is advanced to the next phase. If any phase fails to exceed the level within a vendor specific time period, e.g. one second, this exception is issued. The motor phase loss test only runs when the motor is running above a vendor specified speed. When Torque Proving is enabled, the motor phase current is checked during the Starting state. The current is applied to the motor at a fixed angle that is known to produce current in all three motor phases; hence this test takes very little time to execute. The Motor Phase Loss Limit is used to determine if the drive can produce torque. The measured current in all three phases need to exceed this level for a pass to occur.
10	D	Inverter Overcurrent	Inverter current has exceeded the factory set peak or instantaneous current limit.
11	D	Inverter Overtemperature FL	Inverter temperature has exceeded its factory set temperature limit given by Inverter Overtemperature Factory Limit.
12	D	Inverter Overtemperature UL	Inverter temperature has exceeded the user defined temperature limit given by Inverter Overtemperature User Limit.
13	D	Inverter Thermal Overload FL	Inverter thermal model has exceeded its factory set thermal capacity limit given by Inverter Thermal Overload Factory Limit.
14	D	Inverter Thermal Overload UL	Inverter thermal model has exceeded its user defined thermal capacity given by Inverter Thermal Overload User Limit.
15	BD	Converter Overcurrent	Converter current has exceeded the factory set peak or instantaneous current limit
16	BD	Converter Ground Current FL	Ground Current has exceeded its factory set current limit given by Converter Ground Current Factory Limit.
17	BD	Converter Ground Current UL	Ground Current has exceeded user defined limit given by Converter Ground Current User Limit.

Bit	Rule	Exception	Description
18	BD	Converter Overtemperature FL	Converter temperature has exceeded its factory set temperature limit given by Converter Overtemperature Factory Limit.
19	BD	Converter Overtemperature UL	Converter temperature has exceeded the user defined temperature limit given by Converter Overtemperature User Limit.
20	BD	Converter Thermal Overload FL	Converter thermal model has exceeded its factory set thermal capacity limit given by Converter Thermal Overload Factory Limit.
21	BD	Converter Thermal Overload UL	Converter thermal model has exceeded its user defined thermal capacity given by Converter Thermal Overload User Limit.
22	BD	Converter AC Power Loss	Multiple AC phases have been lost on the AC line to the converter, usually as a result of opening an AC line contactor. When associated with an external converter in a Shared AC/DC or Shared DC bus configuration, the AC Power Loss condition detected by the converter can be conveyed via the CIP Motion Connection’s Control Status element. Generally, this exception is not asserted unless the device’s power structure is enabled.
23	BD	Converter AC Single Phase Loss	One AC phase has been lost on the AC line to the converter.
24	BD	Converter AC Phase Short	A short has been detected between an AC phase and another AC phase or ground.
25	BD	Converter Pre-Charge Failure	A problem has been detected in the Converter’s Pre-Charge circuitry preventing the DC Bus from charging to an acceptable voltage level.
26	-	-- Reserved --	
27	BD	Bus Regulator Overtemperature FL	Bus Regulator temperature has exceeded its factory set temperature limit given by Bus Regulator Overtemperature Factory Limit.
28	BD	Bus Regulator Overtemperature UL	Bus Regulator temperature has exceeded the user defined temperature limit given by Bus Regulator Overtemperature User Limit.
29	BD	Bus Regulator Thermal Overload FL	Bus Regulator thermal model has exceeded its factory set thermal capacity limit given by Bus Regulator Thermal Overload Factory Limit.
30	BD	Bus Regulator Thermal Overload UL	Bus Regulator thermal model has exceeded its user defined thermal capacity given by Bus Regulator Thermal Overload User Limit.
31	BD	Bus Regulator Failure	The bus regulator (shunt) module in a multi-axis system has a failed.
32	-	-- Reserved --	
33	BD	Bus Undervoltage FL	DC Bus voltage level is below the factory set limit given by Bus Undervoltage Factory Limit.
34	BD	Bus Undervoltage UL	DC Bus voltage level is below user defined limit given by Bus Undervoltage User Limit.
35	BD	Bus Overvoltage FL	DC Bus voltage level is above the factory set limit given by Bus Overvoltage Factory Limit.
36	BD	Bus Overvoltage UL	DC Bus voltage level is above user defined limit given by Bus Overvoltage User Limit.
37	BD	Bus Power Loss	DC Bus voltage level is below the Bus Power Loss Threshold for more than the timeout period specified Bus Power Loss Time value.
38	BD	Bus Power Blown Fuse	DC Bus power loss due to blown fuse.
39	D	Bus Power Leakage	DC Bus power leak has been detected when configured for Standalone operation. This can occur when the drive, configured for Standalone operation, is incorrectly wired to share DC bus power.

Bit	Rule	Exception	Description
40	D	Bus Power Sharing	An external converter sharing DC Bus power with this drive in a Shared AC/DC or Shared DC configuration has requested that this drive stop consuming power from the shared DC Bus. This may require that the drive be disabled to remove its DC Bus Power load from the failed converter. When there is no communication link between this drive and the external converter, the controller can monitor the DC Bus Unload bit of the converter axis and, if set, it can initiate Bus Power Sharing exceptions on all drives associated with the converter. See the DC Bus Unload status bit definition associated with the Axis Status attribute for a detailed description of this behavior.
41	E	Feedback Signal Noise FL	Noise induced A/B channel state changes (illegal states) from a feedback device were detected by the drive. Specifically, the number of these noise events that have occurred on this channel has exceeded the Feedback Noise Factory Limit. The offending feedback channel number is encoded in the associated Fault/Alarm Sub Code.
42	E	Feedback Signal Noise UL	Noise induced A/B channel state changes (illegal states) from a feedback device were detected on a feedback channel. Specifically, the number of these noise events that have occurred on this channel has exceeded the Feedback Noise User Limit. The offending feedback channel is encoded in the associated Fault/Alarm Sub Code.
43	E	Feedback Signal Loss FL	One or more A/B channel signals from a feedback device are open, shorted, missing, or severely attenuated. Specifically, the detected voltage levels of the signals are below the Feedback Signal Loss Factory Limit. The offending feedback channel is encoded in the associated Fault/Alarm Sub Code.
44	E	Feedback Signal Loss UL	One or more A/B channel signals from a feedback device are open, shorted, missing, or severely attenuated. Specifically, the detected voltage levels of the signals are below the Feedback Signal Loss User Limit. The offending feedback channel is encoded in the associated Fault/Alarm Sub Code.
45	E	Feedback Data Loss FL	The number of consecutive missed or corrupted serial data packets over the serial data channel from a feedback device has exceeded the Feedback Data Loss Factory Limit. The offending feedback channel is encoded in the associated Fault/Alarm Sub Code.
46	E	Feedback Data Loss UL	The number of consecutive missed or corrupted serial data packets over the serial data channel from a feedback device has exceeded the Feedback Data Loss User Limit. The offending feedback channel is encoded in the associated Fault/Alarm Sub Code.
47	E	Feedback Device Failure	The feedback device has detected an internal error.
48	-	-- Reserved --	
49	D	Brake Slip	Motor displacement has exceeded the Brake Slip Tolerance while the mechanical brake is engaged.
50	D	Hardware Overtravel Positive	Axis moved beyond the physical travel limits in the positive direction and activated the Positive Overtravel limit switch.
51	D	Hardware Overtravel Negative	Axis moved beyond the physical travel limits in the negative direction and activated the Negative Overtravel limit switch.
52	E	Position Overtravel Positive	Axis actual position exceeded the configured Position Limit – Positive attribute value in the positive direction. (Drive Scaling only)
53	E	Position Overtravel Negative	Axis actual position exceeded the configured Position Limit – Negative attribute value in the negative direction. (Drive Scaling only)
54	D	Excessive Position Error	The Position Error value of the position control loop has exceeded the configured value for Position Error Tolerance.
55	D	Excessive Velocity Error	The Velocity Error value of the velocity control loop has exceeded the configured value for Velocity Error Tolerance.
56	D	Overtorque Limit	Motor torque has risen above user defined maximum torque level given by Overtorque Limit.

Bit	Rule	Exception	Description
57	D	Undertorque Limit	Motor torque has dropped below user defined minimum torque level given by Undertorque Limit.
58	-	-- Reserved --	
59	-	-- Reserved --	
60	All	Illegal Control Mode	Controller has specified an unsupported Control Mode or Feedback Mode
61	BD	Enable Input Deactivated	Enable Input has been deactivated while the axis power structure is enabled and supplying current to the DC Bus or motor.
62	All	Controller Initiated Exception	Exception generated specifically by controller.
63	All	External Exception Input	Exception generated by external input to device.

7.3.15 Fault and alarm Log attributes

The attributes in Table 99 are used in conjunction with device based fault and alarm logs. These logs are basically a list of data records with each record corresponding to a fault or alarm related condition detected by the device. The Motion Device Axis Object specification does not specifically define the Fault Log or Alarm Log structures; the log structure is left to the device vendor's discretion. The specification does, however, require that certain information exist in the Fault Log and Alarm Log records.

For the purposes of the discussion that follows, a representative structure for a Fault Log could look like the following:

```
Struct
{
Index;        // Index to the latest fault record element
Fault Log Record Struct    // Array of fault records
{
Fault Type;   // Type of fault
Fault Code;   // Specific fault identifier
Fault Sub Code;      // Extended fault diagnostic info
Fault Action; // Drives response to the fault condition
Time Stamp;   // Time when this fault occurred
} [25]               // Store last 25 fault records in this circular buffer.
}
```

Similarly an alarm log could look like the following:

```
Struct
{
Index;        // Index to the latest alarm record element
Alarm Log Record Struct    // Array of alarm records
{
Alarm Type;   // Type of alarm
Alarm Code;   // Specific alarm identifier
Alarm Sub Code;      // Extended alarm diagnostic info
Alarm State;  // Alarm turned on (1) or turned off (0)
Time Stamp;   // Time stamp when this alarm transition occurred.
} [25]               // Store last 25 alarm records in circular buffer
}
```

Fault and alarm logs serve several purposes in the CIP Motion device. First, they provide a mechanism to queue up multiple fault and alarm events with their associated time stamps for subsequent transfer to the controller for the purpose of building a user accessible replica of the fault and alarm log in the controller. Second, the log record elements themselves provide an efficient way to communicate fault and alarm info to the controller. Instead of sending 24 bytes of fault attributes to the controller every update, the device only needs to send a 1 byte Fault Code corresponding to a fault that has occurred.

To facilitate transfer of fault and alarm records to the controller, attributes are defined for each of the associated record elements that map directly to elements in the CIP Motion Device-to-Controller connection's Status Data Set and to corresponding attributes of the Motion Control Axis Object.

One fault log record and one alarm log record can be transferred to the controller per connection update. The Last Received ID is used by the drive to determine if the data that was sent was processed successfully, i.e. that the Fault and Alarm Codes passed as part of the Device-to-Controller connection's Status Data Block have been entered into the controller's fault and alarm logs. This condition allows the device to send the next Fault Code and Alarm Code in the sequence. When the device has successfully transmitted the most recent fault or alarm event, the device then sends the No Faults for no new faults or No Alarms for no new alarm conditions to report. Alternatively, the device may clear the associated Status Data Set bits indicating to the controller that there are no faults or alarms to report.

Table 99 – Fault and Alarm Log attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
661	Required – All	Get		Axis Fault Code	USINT	An 8-bit enumeration that, when the Axis Fault Type is non-zero, specifies the bit number of a fault condition in an axis fault log record. Possible sources of faults include standard and manufacturer specific Initialization Faults, Axis Faults and Axis Safety Faults. In the case of a Fault Type = 0, this value indicates the source of a fault clearing event. Possible fault clearing events are a device reset, a fault reset request, a shutdown reset request, or a reset resulting from opening a CIP Motion Connection. A Reset shall be recorded regardless of whether or not the axis is currently faulted. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' fault attributes and fault log. A value of 0 indicates that all faults have been cleared and there are no new fault log records to report.	If Axis Fault Type != 0 Enumeration: 0 to 255: Bit Number of Fault If Axis Fault Type = 0 Enumeration: 0 = No Faults 1 = Device Reset (Power Cycle) 2 = Fault Reset 3 = Shutdown Reset 4 = Connection Reset 5 to 255 = (reserved)
662	Required – All	Get		Axis Fault Type	USINT	An 8-bit enumeration that specifies the source of a fault condition or a fault clearing event in an axis fault log record. Possible sources of faults include standard and manufacturer specific Initialization Faults, Axis Faults and Axis Safety Faults. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' fault attributes and fault log. A value of 0 indicates that faults have been cleared. In this case the Axis Fault Code indicates what specific fault clearing event took place to clear the faults.	Enumeration: 0 = Faults Cleared 1 = Initialization Fault 2 = Initialization Fault – Mfg 3 = Axis Fault 4 = Axis Fault – Mfg 5 = APR Fault 6 = APR Fault – Mfg 7 = Axis Safety Fault 8 = Axis Safety Fault – Mfg 9 to 127: Reserved 128 to 255: Vendor Specific
663	Required – All	Get		Axis Fault Time Stamp	LINT	A 64-bit Time Stamp of the axis fault condition or fault clearing event in an axis fault log record. The time stamp format follows the CIP Sync standard with 0 corresponding to 1970-01-01. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' fault log. If the fault log record's Axis Fault Type and Axis Fault Code attributes are 0, the Axis Fault Time Stamp indicates time that the axis faults were last cleared.	Nanoseconds (CIP Sync)

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
664	Optional – All	Get		Axis Alarm Code	USINT	An 8-bit enumeration that specifies the bit number of an alarm condition in an axis alarm log record. Possible sources of alarms include standard and manufacturer specific Axis Alarms. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' alarm status and alarm log. A value of 0 indicates that no alarm conditions are currently present for this axis instance.	Enumeration: 0 to 255: Bit Number of Alarm
665	Optional – All	Get		Axis Alarm Type	USINT	An 8-bit enumeration that specifies the source of an alarm condition in an axis alarm log record. Possible sources of alarms include standard and manufacturer specific Axis Alarms. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' alarm attributes and alarm log. A value of 0 indicates that no alarm conditions are currently present for this axis instance.	Enumeration: 0 = No Alarms 1 = Start Inhibit 2 = Start Inhibit – Mfg 3 = Axis Alarm 4 = Axis Alarm – Mfg 5 to 127: Reserved 128 to 255: Vendor Specific
666	Optional – All	Get		Axis Fault Sub Code	USINT	An 8-bit enumeration that specifies the fault sub code of a fault condition in an axis fault log record. This attribute allows additional detail to be provided in the case of some faults, and has a different interpretation for each fault. Non-zero values indicate that additional detail is present. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' fault log.	Enumeration: specific to fault condition.
667	Optional – All	Get		Axis Alarm Sub Code	USINT	An 8-bit enumeration that specifies the alarm sub code of an alarm condition in an axis alarm log record. This attribute allows additional detail to be provided in the case of some alarms, and has a different interpretation for each alarm. Non-zero values indicate that additional detail is present. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' alarm log.	Enumeration: specific to alarm condition.

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
668	Required – All	Get		Axis Fault Action	USINT	An 8-bit value consisting of two nibbles whose enumerations indicate the action taken by the device in response to the fault condition listed in an axis fault log record. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' fault attributes and fault log. The action taken by the device in response to the fault may also be used by the controller to modify motion planner and motion instruction operation. The Fault Action applied by the device is defined by a Stop Action followed by a State Change Action.	Bit 0 to 3: Stopping Action Enumeration: 0 = Minor Fault (No Action) 1 = (reserved) 2 = Ramped Stop 3 = Torque Limited Stop 4 = Coast 5 to 15 = reserved. Bit 4 to 7: State Change Enumeration: 0 = Minor Fault (No Action) 1 = Hold 2 = Disable 3 = Shutdown 4 to 15 = reserved.
669	Optional – All	Get		Axis Alarm State	USINT	An 8-bit enumeration that indicates the current state of an alarm condition associated with an axis alarm log record. Alarm records are logged when an alarm condition occurs, i.e. when an alarm bit turns on, and when an alarm condition clears, i.e. when an alarm bit turns off. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' alarm attributes and alarm log. If the alarm log record's Axis Alarm Code is 0, i.e. no Alarm is active, the value of this attribute is also 0.	Enumeration: 0 = Alarm Bit Off – 0 1 = Alarm Bit On – 1 2 to 255 = reserved.
670	Optional – All	Get		Axis Alarm Time Stamp	LINT	A 64-bit Time Stamp of the axis alarm condition associated with an axis alarm log record. The time stamp format follows the CIP Sync standard with 0 corresponding to 1970-01-01. This attribute is designed to be passed to the controller via the CIP Motion Connection's Cyclic Data Block for the purpose of replicating the devices' alarm log. If the alarm log record's Axis Alarm Type and Axis Alarm Code attributes are 0, the Axis Alarm Time Stamp value is irrelevant and may also be set to 0.	Nanoseconds (CIP Sync)

7.3.16 Exception limit attributes

The attribute tables in Table 100 and Table 101 contain all exception limit related attributes associated with a Motion Device Axis Object instance. Exception Limit attributes define the conditions under which a corresponding exception is generated during motion axis operation that has the potential of generating either a fault or alarm. They are typically associated with temperature, current, and voltage conditions of the device that are continuous in nature. Factory Limits (FL) for exceptions are usually hard coded in the device and typically result in a major fault condition. User Limits (UL) for exceptions are configurable and typically used to generate a minor fault, or alarm condition. For this reason, the User Limits are generally set inside the corresponding Factory Limits. The triggering of a User Limit exception does not preclude triggering of the corresponding Factory Limit exception; the two exception trigger conditions are totally independent of one another.

Table 100 – Exception Factory Limit Info attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
678	Optional – D	Get		Rotary Motor Overspeed Factory Limit	REAL	Returns the Factory Limit for the Motor Overspeed FL exception based on a factory set value determined by the Rotary Motor Rated Speed or Rotary Motor Max Speed attribute values.	r/min
679	Optional – D	Get		Linear Motor Overspeed Factory Limit	REAL	Returns the Factory Limit for the Motor Overspeed FL exception based on a factory set value determined by the Linear Motor Rated Speed or Linear Motor Max Speed attribute values.	m/s
680	Optional – D	Get		Motor Overtemperature Factory Limit	REAL	Returns the Factory Limit for the Motor Overtemperature FL exception based on a factory set value related to the Motor Max Winding Temperature of the motor.	°C
681	Optional – D	Get		Motor Thermal Overload Factory Limit	REAL	Returns the Factory Limit for the Motor Thermal Overload FL exception based on a factory set value related to the Motor Thermal Overload rating of the motor.	% Motor Rated
682	Optional – D	Get		Inverter Overtemperature Factory Limit	REAL	Returns the Factory Limit for the Inverter Overtemperature FL exception.	°C
683	Optional – D	Get		Inverter Thermal Overload Factory Limit	REAL	Returns the Factory Limit for the Inverter Thermal Overload FL exception.	% Inverter Rated
684	Optional – BD	Get		Converter Overtemperature Factory Limit	REAL	Returns the Factory Limit for the Converter Overtemperature FL exception.	°C
685	Optional – BD	Get		Converter Thermal Overload Factory Limit	REAL	Returns the Factory Limit for the Converter Thermal Overload FL exception.	% Converter Rated
693	Optional – BD	Get		Converter Ground Current Factory Limit	REAL	Returns the Factory Limit for the Converter Ground Current FL exception	Amperes
686	Optional – BD	Get		Bus Regulator Overtemperature Factory Limit	REAL	Returns the Factory Limit for the Bus Regulator Overtemperature FL exception.	°C

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
687	Optional – BD	Get		Bus Regulator Thermal Overload Factory Limit	REAL	Returns the Factory Limit for the Bus Regulator Thermal Overload FL exception.	% Regulator Rated
688	Optional – BD	Get		Bus Overvoltage Factory Limit	REAL	Returns the Factory Limit for the Bus Overvoltage FL exception.	Volts
689	Optional – BD	Get		Bus Undervoltage Factory Limit	REAL	Returns the Factory Limit for the Bus Undervoltage FL exception.	Volts
690	Optional – E	Get		Feedback Noise Factory Limit	UDINT	Returns the Factory Limit for the Feedback Noise FL exception.	Noise Counts
691	Optional – E	Get		Feedback Signal Loss Factory Limit	REAL	Returns the Factory Limit for the Feedback Signal Loss FL exception.	% Nominal Voltage
692	Optional – E	Get		Feedback Data Loss Factory Limit	UDINT	Returns the Factory Limit for the Feedback Data Loss FL exception.	Consecutive Lost Data Packets

Table 101 – Exception User Limit Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
694	Optional – D	Set		Motor Phase Loss Limit	REAL	Sets the minimum motor phase current for the Motor Phase Loss exception. The current in each motor phase shall exceed this value during the motor phase loss test or a Motor Phase Loss exception occurs. Decreasing this attribute's value lowers sensitivity to phase loss conditions. A value of 0 shall effectively disable the motor phase loss test.	% Motor Rated
695	Optional – D	Set		Motor Overspeed User Limit	REAL	Sets the Overspeed User Limit relative to Rotary Motor Rated Speed or Linear Motor Max Speed that is allowable before throwing a Motor Overspeed UL exception.	% Motor Rated
696	Optional – D	Set		Motor Overtemperature User Limit	REAL	Sets User Limit for the Motor Overtemperature UL exception.	°C
697	Optional – D	Set		Motor Thermal Overload User Limit	REAL	Sets User Limit for the Motor Thermal Overload UL exception.	% Motor Rated
698	Optional – D	Set		Inverter Overtemperature User Limit	REAL	Sets User Limit for the Inverter Overtemperature UL exception.	°C
699	Optional – D	Set		Inverter Thermal Overload User Limit	REAL	Sets User Limit for the Inverter Thermal Overload UL exception.	% Inverter Rated
700	Optional – BD (Integral Converter only)	Set		Converter Overtemperature User Limit	REAL	Sets User Limit for the Converter Overtemperature UL exception.	°C

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
701	Optional – BD (Integral Converter only)	Set		Converter Thermal Overload User Limit	REAL	Sets User Limit for the Converter Thermal Overload UL exception.	% Converter Rated
709	Optional – BD (Integral Converter only)	Set		Converter Ground Current User Limit	REAL	Sets User Limit for the Converter Ground Current UL exception	% Factory Limit
702	Optional – BD (Integral Converter only)	Set		Bus Regulator Overtemperature User Limit	REAL	Sets User Limit for the Bus Regulator Overtemperature UL exception.	°C
703	Optional – BD (Integral Converter only)	Set		Bus Regulator Thermal Overload User Limit	REAL	Sets User Limit for the Bus Regulator Thermal UL exception.	% Regulator Rated
704	Optional – BD	Set		Bus Overvoltage User Limit	REAL	Sets User Limit for the Bus Overvoltage UL exception. Unlike the corresponding Factory Limit, which is specified in Volts, the User Limit is based on percent of Nominal Bus Voltage during operation.	% Nominal Bus Voltage
705	Optional – BD	Set		Bus Undervoltage User Limit	REAL	Sets User Limit for the Bus Undervoltage UL exception. Unlike the corresponding Factory Limit, which is specified in Volts, the User Limit is based on percent of Nominal Bus Voltage during operation.	% Nominal Bus Voltage
706	Optional – E	Set		Feedback Noise User Limit	UDINT	Sets User Limit for the Feedback Noise Overload UL exception. Example of Noise Counts would be simultaneous transitions of the A and B channel of a quadrature encoder feedback device.	Noise Counts
707	Optional – E	Set		Feedback Signal Loss User Limit	REAL	Sets User Limit for the Feedback Signal Loss UL exception. Feedback interface hardware typically monitor average voltage levels on incoming signals. Feedback Signal Loss conditions occur when the average voltage levels drop below a percentage of the voltage drop allowed by the Feedback Signal Loss Factory Limit.	% FL Voltage Drop

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
708	Optional – E	Set		Feedback Data Loss User Limit	UDINT	Sets User Limit for the Feedback Data Loss UL exception. For digital feedback devices, feedback interface hardware monitors the integrity of data transferred over the serial connection to the feedback device. Feedback Data Loss conditions occur when the two or more consecutive data packets are lost or corrupted.	Consecutive Lost Data Packets

7.3.17 Axis exception action configuration attribute

7.3.17.1 General

The configuration attribute in Table 102 controls the action performed by the device as a result of an exception condition. A unique exception action is defined for each supported exception condition.

Table 102 – Axis Exception Action Configuration attribute

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
672	Required – All	Set		Axis Exception Action	USINT [64]	The Axis Exception Action attribute is a 64-element array of enumerated bytes that specifies the action for the associated standard exception. See Semantics in 7.3.17.2 for details.	Enumeration: 0 = Ignore (O) 1 = Alarm (O) 2 = Minor Fault (O) 3 = Major Fault (R) 4 to 254 = reserved 255 = Unsupported
673	Required – All	Set		Axis Exception Action – Mfg	USINT [64]	The Axis Exception Action attribute is a 64-element array of enumerated bytes that specifies the action for the associated manufacturer specific exception. See Semantics in 7.3.17.2 for details	Enumeration: 0 = Ignore (O) 1 = Alarm (O) 2 = Minor Fault (O) 3 = Major Fault (R) 4 to 254 = reserved 255 = Unsupported

7.3.17.2 Semantics: Attribute No. 672 and 673 – Axis Exception Action

The Axis Exception Action and Axis Exception Action–Mfg. attributes are 64-element arrays of enumerated bytes that specifies the action for the associated standard or manufacturer specific exception, respectively (see Table 103). For a given exception, certain exception actions may not be supported. Attempting to do so shall result in an error. Each device product shall specify the available actions for each exception that is supported. If a specific exception is not supported by the device, the only valid exception action enumeration is “Unsupported”. Attempting to write any other value to the element associated with an unsupported exception results in an error.

Table 103 – Axis Exception Action definitions

Enum.	Need in implementation	Name	Description
0	Optional	Ignore	Ignore instructs the device to completely ignore the exception condition. For some exceptions that are fundamental to the operation of the axis, it may not be possible to Ignore the condition.
1	Optional	Alarm	Alarm action simply sets the associated bit in the Axis Alarms word but does not otherwise affect axis behavior. For some exceptions that are fundamental to the operation of the device it may not be possible to select this action or any other action that leaves device operation unaffected.
2	Optional	Minor Fault	Minor Fault action reports the exception by setting the associated bit in the Axis Faults word but does not affect axis behavior. Minor Faults allow the controller to take application specific action in response to the exception condition. For some exceptions that are fundamental to the operation of the device, it may not be possible to simply indicate the condition as a Minor Fault and leave device operation unaffected. Converters (B) executing a Minor Fault exception action continue to supply DC Bus Power and do not set the DC Bus Unload bit in Axis Status attribute.
3	Required	Major Fault	Major Fault action results in both setting the associated bit in the Faults word and bringing the axis to a stop based on the factory set "best" available stopping method. This "best" stopping method includes both the method of decelerating the motor to a stop and the final state of the axis given the expected level of control still available. The level of axis control available depends on the specific exception condition and on the configured control mode. The available deceleration methods are defined by the Stopping Action attribute. Standard stopping actions, listed in decreasing levels of deceleration control, are as follows: 1. Ramp Decel 2. Current Limit Decel 3. Coast In general, the "best" stopping action is the most controlled deceleration method still available given the exception condition. While the final Axis State after a Major Fault is always the Major Faulted state the final state of the power structure in response to the Major Fault exception action can be any one of the following states that are listed in decreasing levels of control functionality: 1. Hold (stopped with Holding Torque) 2. Disable (stopped with Power Structure Disabled) 3. Shutdown (stopped with Shutdown Action) The "best" final state of the power structure is the state with the most control functionality still available given the exception condition. The specific stopping action and final state associated with a given Major Fault exception action is captured in the Axis Fault Action attribute that is included in the Fault Log record. Axis Fault Action enumerations are as follows: Stop Action Enumerations: 0 = No Action 1 = (reserved) 2 = Ramped Stop 3 = Current Limited Stop 4 = Coast State Change Enumerations: 0 = No Action 1 = Hold 2 = Disable 3 = Shutdown

Enum.	Need in implementation	Name	Description
			If the application requires exception action that is a more severe stopping action than the factory set "best" method, the controller shall initiate that action. If the application requires exception action that is less severe than the factory set "best" method, the controller shall configure the device axis instance for a Minor Fault exception action and handle the fault directly. This may put device and motor components at risk and shall only be allowed by the device when there is an opportunity, albeit temporal, for the device to remain operational. This is important in applications where the value of the product is higher than the value of the motor or device. When the Major Fault exception action is applied to a converter device (B), stopping action is not applicable (0 = No Action). The final states of Disable or Shutdown for the converter are applicable, however, with Shutdown executing the configured Shutdown Action. In either case, the DC Bus Unload bit of the converter's Axis Status attribute is set allowing the control system to generate a Bus Sharing exception on all drives that consume DC Bus power from this converter. When multiple major faults occur with different stopping actions, the most severe of the associated stopping actions is applied, i.e. the stopping action that requires the lowest level of control functionality. This rule also applies to the stopping action associated with the Stopping Action associated with a Disable Request.
4 to 254		Reserved	-
255	Required	Unsupported	The Unsupported Exception Action is the value assigned to Exceptions that are not supported by the device. Trying to assign an Exception Action other than Unsupported to an exception that is not supported by the device results in an error.

7.3.18 Initialization fault attributes

7.3.18.1 General

The attribute table in Table 104 contains all initialization fault related attributes associated with a Motion Device Axis Object instance. Initialization Faults are conditions that can occur during the device initialization process that prevent normal operation of the device.

Table 104 – Initialization Fault attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
674	Required – All	Get		Initialization Faults	DWORD	A bit map that represents the state of all standard initialization faults. These faults prevent any motion, and do not have configurable fault actions. Examples of initialization faults are corrupted memory data, calibration errors, firmware startup problems, or an invalid configuration attribute value. Initialization faults cannot be cleared with a Fault Reset request, although a power-cycle provides a new attempt at initialization.	See Semantics in 7.3.18.2
675	Required – All	Get		Initialization Faults – Mfg	DWORD	A bit map that represents the state of all manufacturer specific initialization faults. These faults prevent any motion, and do not have configurable fault actions. Examples of initialization faults are corrupted memory data, calibration errors, firmware startup problems, or an invalid configuration attribute value. Initialization faults cannot be cleared with a Fault Reset request, although a power-cycle provides a new attempt at initialization.	See Mfg. Initialization Fault Table (Published in Product Manual)

7.3.18.2 Semantics: Standard Initialization Fault Table

Table 105 defines a list of standard faults associated with the Initialization Faults attribute and also applicable to the Initialization Fault Code attribute.

Table 105 – Standard Initialization Fault Table

Bit	Exception	Description
0	-- Reserved --	This bit cannot be used since the Fault Code is defined by the associated exception bit number and Fault Code of 0 means no fault condition is present.
1	Boot Block Checksum Fault	Checksum or CRC error for Boot Block of CIP Motion device detected as part of Self-Test.
2	Main Block Checksum Fault	Checksum or CRC error for Main Block of CIP Motion device detected as part of Self-Test.
3	Nonvolatile Memory Checksum Fault	Checksum or CRC error for NV Memory of CIP Motion device detected as part of Self-Test.
4 to 31	-- Reserved --	

7.3.19 Start inhibit attributes

7.3.19.1 General

The attribute table in Table 106 contains all Start Inhibit related attributes associated with a Motion Device Axis Object instance. Start Inhibits are conditions that prevent transition of the motion axis from the Stopped State into any of the operational states.

Table 106 – Start Inhibit attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
676	Required – D	Get		Start Inhibits	WORD	A bit map that specifies the current state of all standard conditions that inhibit starting of the axis.	See Semantics in 7.3.19.2
677	Required – D	Get		Start Inhibits – Mfg	WORD	A bit map that specifies the current state of all manufacturer specific conditions that inhibit starting of the axis.	See Mfg. Start Inhibit Table (Published in Product Manual)

7.3.19.2 Semantics: Standard Start Inhibit Table

Table 107 defines a list of standard start inhibits associated with the Start Inhibits attribute.

Table 107 – Standard Start Inhibit Table

Bit	Exception	Description
0	-- Reserved --	This bit cannot be used since the Start Inhibit Code is defined by the associated bit number and Start Inhibit Code of 0 means no fault condition is present.
1	Axis Enable Input	Axis Enable Input is not active.
2	Motor Not Configured	The associated motor has not been configured for use.
3	Feedback Not Configured	The associated feedback device has not been configured for use
4	Commutation Not Configured	The associated PM motor commutation function has not been configured for use.
5	Safe Torque Off Active	The integrated Safe Torque Off safety function is active based on the Safe Torque Off Active bit (bit 3) of the Axis Safety Status (Attribute No. 761) being set.
6 to 15	-- Reserved --	

7.3.20 APR fault attributes

7.3.20.1 General

The attribute table in Table 108 contains all APR (Absolute Position Recovery) fault related attributes associated with a Motion Device Axis Object instance. APR Faults are conditions that can occur during the device initialization process when trying to restore the absolute position of an axis. Unlike Initialization Faults, these faults are recoverable and may be cleared with a Fault Reset request.

Table 108 – APR Fault attributes

Attr id	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
756	Optional – E Drive Scaling	Get		APR Faults	WORD	A bit map that represents the state of all standard APR (Absolute Position Recovery) faults. An APR fault is generated when the system fails to recover the absolute position of the axis after power cycle, reset, or reconnection. APR faults are detected during the initial configuration or initialization of the drive axis. When an APR fault occurs, the actual position of the axis is no longer correlated to the position of the axis prior to the power cycle, reset, or reconnect. Examples of standard APR faults are feedback serial number mismatch, Axis ID mismatch, scaling configuration change, etc. APR faults are recoverable and can be cleared with a Fault Reset request.	See semantics in 7.3.20.2
757	Optional – E Drive Scaling	Get		APR Faults – Mfg	WORD	A bit map that represents the state of all manufacturer APR (Absolute Position Recovery) faults. An APR fault is generated when the system fails to recover the absolute position of the axis after power cycle, reset, or reconnection. APR faults are detected during the initial configuration or initialization of the drive axis. When an APR fault occurs, the actual position of the axis is no longer correlated to the position of the axis prior to the power cycle, reset, or reconnect. APR faults are recoverable and can be cleared with a Fault Reset request.	See Mfg. APR Fault Table (Published in Product Manual)

7.3.20.2 Semantics: Standard APR Fault Table

Table 109 defines a list of standard faults associated with the APR Faults attribute and also applicable to the APR Fault Code attribute.

Table 109 – Standard APR Fault Table

Bit	Exception	Description
0	-- Reserved --	This bit cannot be used since the Fault Code is defined by the associated exception bit number and Fault Code of 0 means no fault condition is present.
1	Memory Write Error	Error in saving absolute position data to NV memory.
2	Memory Read Error	Error in reading absolute position data from NV memory
3	Feedback Serial Number Mismatch	Position Feedback Serial Number does not match saved Feedback Serial Number
4	Buffer Allocation Fault	Caused when there is not enough RAM memory left to save APR data.
5	Scaling Configuration Changed	Scaling attribute configuration for this axis does not match the saved scaling configuration.
6	Feedback Mode Changed	Feedback Mode has changed and does not match the saved Feedback Mode configuration.
7 to 15	-- Reserved --	

7.3.21 Axis statistical attributes

Table 110 includes attributes that provide useful statistics on motion axis operation.

Table 110 – Axis Statistical attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
710	Optional – BD	Get		Control Power-up Time	REAL	Elapsed time since axis control power was last applied.	Seconds
711	Optional – BD	Get		Cumulative Run Time	REAL	Accumulated time that the axis has been powering the Running state.	Hours
712	Optional – BD	Get		Cumulative Energy Usage	REAL	Accumulated output energy of the axis.	kW•h
713	Optional – D	Get		Cumulative Motor Revs	LINT	Cumulative number of times motor shaft has turned. (Rotary Motors Only)	
714	Optional – BD	Get		Cumulative Main Power Cycles	DINT	Cumulative number of times AC Mains has been cycled.	
715	Optional – BD	Get		Cumulative Control Power Cycles	DINT	Cumulative number of times Control Power has been cycled.	

7.3.22 Axis info attributes

Table 111 includes attributes that provide information about the associated hardware capabilities of Motion Device Axis Object instance.

Table 111 – Axis Info attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
720	Required – D	Get		Inverter Rated Output Voltage	REAL	Drive inverter output voltage rating. This value is hard coded in the device.	Volts (RMS)
721	Required – D	Get		Inverter Rated Output Current	REAL	Drive inverter output current rating. This value is hard coded in the device.	Amperes (RMS)
722	Required – D	Get		Inverter Rated Output Power	REAL	Drive inverter output power rating. This value is hard coded in the device.	kW
723	Optional – BD (Integral Converter Only)	Get		Converter Rated Output Current	REAL	Drive converter output current rating. This value is determined by the motion axis from the associated converter.	Amperes
724	Optional – BD (Integral Converter Only)	Get		Converter Rated Output Power	REAL	Drive converter output power rating. This value is determined by the motion axis from the associated converter.	kW
725	Optional – D	Get		Drive Power Structure Axis ID	UDINT	This value represents an ID assigned by the drive vendor that uniquely identifies the power structure associated with this axis instance. By contrast, power structure hardware that is common or identical to all axis instances of the drive (excluding master feedback axes) can be identified using the Drive Power Structure Class ID attribute. The Drive Power Structure Axis ID can be included as part of the data segment in the CIP Motion Connection's Forward_Open service to confirm that the power structure for this axis instance matches the configuration in the controller. Axis instances with power structures that are not configured are set to 0 indicating to the device that the Drive Power Structure Axis ID for such an axis instance does not need to be checked.	ID#

7.3.23 General purpose I/O attributes

Table 112 includes attributes that provide to general purpose analog and digital I/O associated with the Motion Device Axis Object instance.

Table 112 – Drive General Purpose I/O attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
730	Optional – BD	Get		Digital Inputs	DWORD	A 32-bit word with whose bits can be assigned by the vendor to general purpose digital inputs.	Vendor Specific Bit Map
731	Optional – BD	Set		Digital Outputs	DWORD	A 32-bit word with whose bits can be assigned by the vendor to general purpose digital outputs.	Vendor Specific Bit Map
732	Optional – BD	Get		Analog Input 1	REAL	General purpose analogue input 1 level.	% Full Scale
733	Optional – BD	Get		Analog Input 2	REAL	General purpose analogue input 2 level.	% Full Scale
734	Optional – BD	Set		Analog Output 1	REAL	General purpose analogue output 2 level.	% Full Scale
735	Optional – BD	Set		Analog Output 2	REAL	General purpose analogue output 2 level.	% Full Scale
736	Optional – BD	Set		Drive Enable Input Checking	BOOL	Boolean value that controls whether or not the drive shall regularly check the state of the Drive Enable input. When Drive Enable Input Checking is enabled, an inactive Drive Enable input results in a Start Inhibit condition. If the Drive Enable input is deactivated while the drive is enabled, a Drive Enable Input Deactivated exception is generated. If Drive Enable Input Checking is disabled, the drive shall not check the state of the Drive Enable input.	0 = Disabled 1 = Enabled
737	Optional – D	Set		Hardware Overtravel Input Checking	BOOL	Boolean value that controls whether or not the drive shall regularly check the state of the positive and negative Hardware Overtravel inputs. When Hardware Overtravel Input Checking is enabled, an inactive Hardware Overtravel input results in an associated Hardware Overtravel Positive or Negative exception. If Drive Hardware Overtravel Checking is disabled, the drive shall not check the state of the Hardware Overtravel inputs.	0 = Disabled 1 = Enabled

7.3.24 Local Mode attributes

Table 113 contains all local mode attributes associated with a Motion Device Axis Object instance. These attributes govern the behavior of the Motion Device Axis Object instance when a local drive (i.e. non-CIP Motion) device interface requests control of axis behavior.

Table 113 – Local Mode Configuration attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
750	Optional	Set		Local Control	USINT	Mechanism for controller to allow a local interface to request local control. Control implies, in this case, the ability to change the state or behavior of the axis. Local Control Not Allowed configures the device to prevent a local interface from taking control of the drive. Local Control Conditionally Allowed configures the device to prevent a local interface from taking control of the axis while the axis is in an Operational State, such as Running or Testing. Local Control Allowed configures the device to allow a local interface to take control of the axis, even in an Operational State. Some devices may not support this state.	Enumeration: 0 = Not Allowed (R) 1 = Conditionally Allowed (O) 2 = Allowed (O) 3 to 255 = reserved

7.3.25 Axis Safety attributes

The following attribute table contains axis attributes associated with the “Networked” Safety functionality associated with a Motion Device Axis Object instance included in a CIP Motion Safety Drive. These attributes reflect the current state of an embedded Safety Core within the device that is designed to interoperate with an external Safety Controller via a CIP Safety™ [17] connection.

The Axis Safety State, Axis Safety Status, and Axis Safety Fault attributes defined in Table 114 are based on the values read from attributes resident in objects associated with the Safety Core and are used by the motion control system to monitor the behavior of the Safety Core via the CIP Motion Connection.

NOTE The CIP Motion Safety Drive, Safety Core, CIP Safety™ connection, as well as associated safety related objects and their attributes are outside the scope of this standard. Their specification can be found in [30].

[17] CIP Safety™ is a trade mark of ODVA, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trade mark holder or any of its products. Compliance to this profile does not require use of the trade mark CIP Safety™. Use of the trade mark CIP Safety™ requires permission of ODVA, Inc.

Table 114 – Axis Safety Status attributes

Attr ID	Need in impl.	Access rule	NV	Attribute name	Data type	Description of attribute	Semantics of values
760	Optional – DE Safety only	Get		Axis Safety State	USINT	8-bit enumeration that indicates the state of the associated Safety Supervisor object of the device. There is only one Safety Supervisor object servicing the CIP Motion device so its state applies to all applicable Axis instances of the device. This means that all instances of this object will have the same state for this attribute.	Enum
761	Optional – DE Safety only	Get		Axis Safety Status	DWORD	Collection of bits indicating the status of the standard safety functions for the axis as reported by the embedded Safety Core.	Bitmap
762	Optional – DE Safety only	Get		Axis Safety Status – Mfg	DWORD	Collection of bits indicating the status of the manufacturer specific safety functions for the axis as reported by the embedded Safety Core.	Bitmap: 0 to 31: Vendor Specific
763	Optional – DE Safety only	Get		Axis Safety Faults	DWORD	Collection of bits indicating the safety fault status of the axis associated with standard safety functionality as reported by the embedded Safety Core.	Bitmap
764	Optional – DE Safety only	Get		Axis Safety Faults – Mfg	DWORD	Collection of bits indicating the safety fault status of the axis associated with manufacturer specific safety functionality as reported by the embedded Safety Core.	Bitmap: 0 31: Vendor specific
NOTE See [30] for complete definition of enumerations, bitmaps, and corresponding state behavior.							

7.4 Common services

7.4.1 Supported services

The Motion Device Axis Object provides the CIP Common Services specified in Table 115.

Table 115 – Motion Device Axis Object – Common Services

Service code (Hex)	Need in implementation		Service name	Description of service
	Class	Instance		
03_{hex}	Required	Required	Get_Attributes_List	Returns the contents of the selected gettable attributes for given instance.
04_{hex}	Required	Required	Set_Attributes_List	Sets the content of the selected settable attributes for given instance.
$0E_{hex}$	Required	Required	Get_Attribute_Single	Returns the contents of the selected gettable attribute for given instance.
10_{hex}	Required	Required	Set_Attribute_Single	Sets the content of the selected settable attribute for given instance.
$1C_{hex}$	Conditional [a]	N/A	Group_Sync	Passes controller's Grand Master Clock ID to device and reports if this device can be synchronized with the controller as part of a group of associated devices.

[a] Group_Sync service is Required for CIP Motion devices that support Time Synchronization service and include the Time Sync Object.

7.4.2 Service specific data

7.4.2.1 General

Common Services supported by the Motion Device Axis Object have associated Request Data and Response Data. The format and syntax of the Service Specific Data depends on the specified Service Code. Descriptions of the Service Specific Data for the Set/Get_Attributes_List and Set/Get_Attribute_Single services are already defined in CIP (see IEC 61158-5-2 and IEC 61158-6-2). A description of the Service Specific Data associated with the Group_Sync service (part of the CIP Sync extensions to CIP specifications) is detailed in 7.4.2.2.

7.4.2.2 Group_Sync

The Group_Sync service is used to synchronize the Motion Device Axis Object associated with this device node to the controller issuing the service request and, ultimately, to all other devices comprising the controller's Motion Group. The service response indicates "device synchronized" if the device is presently synchronized to the same IEC 61588:2009 Grand Master as the controller.

The format and syntax of Group_Sync Request Data is specified in Table 116.

Table 116 – Group_Sync Request Data Structure

Name	Type	Description of request parameter	Semantics of values
Grand Master Clock ID	SINT[8]	Clock ID of Grandmaster Clock associated with the Controller.	See Time Sync Object in IEC 61158-5-2 and IEC 61158-6-2

The Grand Master Clock ID is used to verify that the IEC 61588:2009 PTP Grand Master Clock for the Controller is the same Grand Master Clock associated with the device.

The format and syntax of Group_Sync Response Data is specified in Table 117.

Table 117 – Group_Sync Response Data Structure

Name	Type	Description of response parameter	Semantics of values
Sync Status	SINT	Enumerated value that indicates whether or not the device is presently group synchronized or has different time master than controller.	0 = device synchronized 1 = device not synchronized 2 = different Grand Master 3 = clock skew detected 4 to 255 = reserved

When the Group_Sync service request is executed, the device first checks if the Grand Master associated with the controller matches the associated Grand Master of the device. This is an important prerequisite to successfully synchronizing device's local clock with the controller. If the Grand Master IDs do not match, the Group_Sync replies with a Sync Status of 2, indicating "different grand master". If the Grand Master Clock IDs match, then the Group_Sync service moves on to check the state of the device's clock synchronization.

Next, the device shall query the Time Sync Object to determine if the device is synchronized. This requires that the System Time Offset changes have been less than the Time Sync Threshold (typically, 10 µs) for a predetermined period of time. If the device's clock is synchronized with a common Grand Master clock, then the Group_Sync service moves on to check for possible clock skew. Otherwise the service returns a Sync Status of "device not synchronized" (1).

If the above conditions are met, the device then checks for possible time skew between the controller's System Time and device's System Time. This check is made possible by the controller's device configuration process insuring that regular Time Stamp and Time Offset values are sent to the device prior to sending a Group_Sync service request, which in turn requires that regular Time Stamp and Time Offset values are sent back to the controller prior sending a Group_Sync service response. The device performs a clock skew check by loading the Controller Time Stamp from the most recent Control-to-Device packet. If the sum of the Controller Time Stamp and the Controller Update Period (CUP) differs from the current device System Time by more than one Controller Update Period (CUP), the controller and device System Time clocks are skewed.

That is:

If |(Controller Time Stamp + CUP) – Current Device System Time| > 1 CUP

Controller and Device Clocks are Skewed

Else

Controller and Device Clocks are OK

If the device clock is not skewed, then the controller and device System Time clocks are considered fully synchronized at this instant in time. Based on this assumption, the Group_Sync service initializes the Controller's Last System Time Offset value with the Controller Time Stamp. Similarly, the service initializes the Device's Last System Time Offset value with the value associated with the Current Device Time Stamp. These two variables shall be used later when executing the Time Step Compensation algorithm. Finally, the Group_Sync returns with Sync Status of "device synchronized" (0). Otherwise, if clock skew is detected, the service returns a Sync Status of "clock skew detected" (3).

When the controller receives a "device synchronized" response from the device, it performs its own skew check and Time Offset initialization. If no clock skew is detected, the device is then considered Group Synchronized. If the controller receives a non-zero Sync Status response from the device or fails its skew test, the controller shall send another Group_Sync service request. This process shall continue for at least 1 min until the device is Group Synchronized or the process timeout occurs resulting in a controller initiated connection fault.

Once the controller and all the CIP Motion devices are group synchronized, the controller then sets the Synchronous Control bit in the Controller-to-Device Connection Header of the next cyclic packet sent to the device indicating that the device can schedule its next Device-to-Controller Connection update based on the Controller Time Stamp and Controller Update Period contained in the connection data. The device then indicates that it is sending scheduled connection data based on the Controller Update Period by setting the Sync Mode bit in the Device-to-Controller Connection Header of the next cyclic packet sent to the controller. Data exchange between the controller and the device proceed thereafter according the CIP Motion timing model.

7.5 Object specific services

7.5.1 Supported services

The Motion Device Axis Object provides the Object Specific Services specified in Table 118.

Table 118 – Motion Device Axis Object – Object Specific Services

Service code (Hex)	Need in implementation		Service name	Description of service
	Class	Instance		
$4B_{hex}$	n/a	Required	Get_Axis_Attributes_List	Returns the contents of the selected gettable Motion Device Axis Object attributes.
$4C_{hex}$	n/a	Required	Set_Axis_Attributes_List	Sets the content of the selected settable Motion Device Axis Object attributes.
$4D_{hex}$	n/a	Required	Set_Cyclic_Write_List	Sets the attribute contents of the Cyclic Write Data block.
$4E_{hex}$	n/a	Required	Set_Cyclic_Read_List	Sets the attribute contents of the Cyclic Read Data block.
$4F_{hex}$	n/a	Optional – D	Run_Motor_Test	Initiates the selected test on the motor to measure various motor parameters.
50_{hex}	n/a	Optional – D	Get_Motor_Test_Data	Returns the results of the preceding Run_Motor_Test service.
51_{hex}	n/a	Required – C	Run_Inertia_Test	Initiates the selected test on the motor to measure the inertia.
52_{hex}	n/a	Required – C	Get_Inertia_Test_Data	Returns the results of the preceding Run_Inertia_Test service.
53_{hex}	n/a	Required	Run_Hookup_Test	Initiates the selected test to determine the condition of the motor and feedback device connections.
54_{hex}	n/a	Required	Get_Hookup_Test_Data	Returns the results of the preceding Run_Hookup_Test service.
55 to 63_{hex}			Reserved	

7.5.2 Service specific data

7.5.2.1 General

Object Specific Services supported by the Motion Device Axis Object have associated Request Data and Response Data. The format and syntax of the Service Specific Data depends on the specified Service Code. All parameters contained in the Service Specific Data are word aligned. This is true regardless of whether the service specific request data is passed in the Controller-to-Device Connection or as part of an Explicit messaging connection. A description of the Service Specific Data associated with each service is shown in 7.5.2.2 to 7.5.2.11.

7.5.2.2 Get_Axis_Attributes_List

The Get_Axis_Attributes List service provides a mechanism to read the current value of one or more Motion Device Axis Object attributes, including attributes having a multi-dimensional array data type. The buffer/array transfer mechanism can be used to transfer large data log arrays from the device to facilitate features like high speed trending.

The format of the Request Data Block for this service is shown in Figure 70.

← 32-bit Word →		
Get_Axis_Attributes_List Request format		
Number of Attributes	-	
Attr ID 1	Attr 1 Dimension	Attr 1 Element Size
Attr Start Index 1 (array only)	Attr Data Elements 1 (array only)	
…	…	
Attr ID 2	Attr 2 Dimension	Attr 2 Element Size
Attr Start Index 2 (array only)	Attr Data Elements 2 (array only)	
…	…	
Attr ID n	Attr n Dimension	Attr n Element Size
Attr Start Index n (array only)	Attr Data Elements n (array only)	
…	…	

IEC

Figure 70 – Get_Axis_Attributes_List Request rormat

Definitions of the individual parameters in this data structure are as follows.

- Number of Attributes: represents the number of attributes contained in the Get Axis Attribute service request.
- Attr ID: identifies the targeted Motion Device Axis Object attribute to get.
- Attr Dimension: determines the dimension of the attribute array. A dimension of zero means the attribute is a singular data element and, therefore, not really an array at all. Multidimensional arrays (dimension > 1) are supported by adding additional Attr Start Index and Addr Data Elements values. If there is an error associated with a specific requested get attribute operation, the device shall indicate this by setting the Attr Dimension to 0xFF, in which case the Element Size field contains the specific error code. When an error code is present, neither the array index parameters nor the attribute data fields are returned.
- Attr Element Size: determines the size, in bytes, of the attribute data element. Data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding may be added to maintain word alignment. If there is an error associated with a specific requested get attribute operation, i.e. Attr Dimension = 0xFF, the Element Size field then contains the CIP Common General Status error code, as defined in IEC 61158-6. When an error code is present neither the array index parameters nor the attribute data fields are returned.
- Attr Start Index: identifies the starting index for the array of attribute values in the Attr Data section. This field is only present when the attribute data type is an array (i.e., dimension > 0).
- Attr Data Elements: determines the number of data element values in the Attr Data section for the associated index. This field is only present when the attribute data type is an array (i.e., dimension > 0).

The structure of Response Data Block for this service is as shown in Figure 71.

← 32-bit Word →		
Get_Axis_Attributes_List Response format		
Number of Attributes	-	
Attr ID 1	Attr 1 Dimension	Attr 1 Element Size
Attr Start Index 1 (array only)	Attr Data Elements 1 (array only)	
…	…	
Attr Data 1		
…		
Attr ID 2	Attr 2 Dimension	Attr 2 Element Size
Attr Start Index 2 (array only)	Attr Data Elements 2 (array only)	
…	…	
Attr Data 2		
…		
Attr ID n	Attr n Dimension	Attr n Element Size
Attr Start Index n (array only)	Attr Data Elements n (array only)	
…	…	
Attr Data n		
…		

IEC

Figure 71 – Get_Axis_Attributes_List Response format

Definitions of the individual parameters in this data structure are identical to the request data structure with the addition of the Attribute Data element, which is defined as follows.

- Attr Data: contains the current value(s) of the targeted Motion Device Axis Object attribute indicated by the Attr ID. If the attribute is an array, the value(s) are listed according to the Attr Start Index and the number of Attr Data Elements. For multidimensional arrays (dimension > 1), the sequence of data move sequentially through the indices from right to left. Examples are provided below.

EXAMPLE 1 Figure 72 defines a Get_Axis_Attributes_List Response for a simple 4-byte DINT scalar attribute. Since the Attr Dim is 0, the Attr Start Index and Attr Data Element fields are omitted.

Get_Axis_Attributes_List Response – 0 Dimensional Data Example		
Number of Attributes = 1	-	
Attr ID = 25	Attr Dim = 0	Attr Elem Size = 4
Attr Data = ?		

IEC

Figure 72 – Get_Axis_Attributes_List Response – Single 4-byte attribute

EXAMPLE 2 Figure 73 defines a Get_Axis_Attributes_List Response for a simple 2-byte scalar attribute showing the pad byte that is added to maintain 32-bit word alignment.

Get_Axis_Attributes_List Response – 0 Dimensional Data Example		
Number of Attributes = 1	-	
Attr ID = 20	Attr Dim = 0	Attr Elem Size = 2
Attr Data = ?	(Pad)	

IEC

Figure 73 – Get_Axis_Attributes_List Response – Single 2-byte attribute

EXAMPLE 3 Figure 74 defines a Get_Axis_Attributes_List Response for the first three elements of a one dimensional array attribute that is a UINT. Since the Attr Elem Size for this attribute is a 2-byte value, a Pad byte is added to maintain word alignment for any connection data to follow.

Get_Axis_Attributes_List Response – 1 Dimensional Array Example		
Number of Attributes = 1	-	
Attr ID = 43	Attr Dim = 1	Attr Elem Size = 2
Attr Start Index 1 = 0	Attr Data Elements 1 = 3	
Attr Data (0) = ?	Attr Data (1) = ?	
Attr Data (2) = ?	(Pad)	

IEC

Figure 74 – Get_Axis_Attributes_List Response – Byte attribute array

EXAMPLE 4 Figure 75 defines a Get_Axis_Attributes_List Response for six elements of a two dimensional array of REALs. The Attr Data sequences through the leftmost array index first beginning with the Attr Start Index 1 of 0.

Get_Axis_Attributes_List Response – 2 Dimensional Array Example		
Number of Attributes = 1	-	
Attr ID = 27	Attr Dim = 2	Attr Elem Size = 4
Attr Start Index 1 = 0	Attr Data Elements 1 = 2	
Attr Start Index 2 = 2	Attr Data Elements 2 = 3	
Attr Data (0, 2) = ?		
Attr Data (0, 3) = ?		
Attr Data (0, 4) = ?		
Attr Data (1, 2) = ?		
Attr Data (1, 3) = ?		
Attr Data (1, 4) = ?		

IEC

Figure 75 – Get_Axis_Attributes_List Response – Two Dimensional attribute array

If there is an error associated with a specific requested get attribute operation, the device shall indicate this by setting the Attr Dimension to 0xFF, in which case the Element Size field may contain the specific error code. When an error code is present, neither the array index parameters nor the attribute data fields are returned.

EXAMPLE 5 Figure 76 defines such a case, where Attribute 29 is not supported by the targeted motion axis, as indicated by the General Status Code of 0x14.

Get_Axis_Attributes_List Response – Attribute Error Example		
Number of Attributes = 3	-	
Attr ID = 25	Attr Dim = 0	Attr Elem Size = 4
Attr Data = 43,5		
Attr ID = 29	Attr Dim = 0xFF	Error Code = 0x14
Attr ID = 43	Attr Dim = 1	Attr Elem Size = 2
Attr Start Index 1 = 0	Attr Data Elements 1 = 3	
Attr Data [0] = 55	Attr Data [1] = 45	
Attr Data [2] = 35	(Pad)	

IEC

Figure 76 – Get_Axis_Attributes_List Response – Error example

7.5.2.3 Set_Axis_Attributes_List

The Set_Axis_Attributes_List service provides a mechanism to write a value to one or more settable Motion Device Axis Object attributes, including attributes having a multi-dimensional array data type. The buffer/array transfer mechanism can be used to build parameter tables in the device for indexing, or camming applications.

The format of the Request Data Block for this service is shown in Figure 77.

← 32-bit Word →		
Set_Axis_Attributes_List Request format		
Number of Attributes	-	
Attr ID 1	Attr 1 Dimension	Attr 1 Element Size
Attr Start Index 1 (array only)	Attr Data Elements 1 (array only)	
…	…	
Attr Data 1		
…		
Attr ID 2	Attr 2 Dimension	Attr 2 Element Size
Attr Start Index 2 (array only)	Attr Data Elements 2 (array only)	
…	…	
Attr Data 2		
…		
…		
Attr ID 3	Attr n Dimension	Attr n Element Size
Attr Start Index n (array only)	Attr Data Elements n (array only)	
…	…	
Attr Data 3		
…		

IEC

Figure 77 – Set_Axis_Attributes_List Request format

Definitions of the individual parameters in this data structure are as follows.

- Number of Attributes: represents the number of attributes contained in the Set Axis Attribute service request.
- Attr ID: identifies the targeted Motion Device Axis Object configuration attribute to set.
- Attr Dimension: determines the dimension of the attribute array. A dimension of zero means the attribute is a singular data element and, therefore, not really an array at all. Multidimensional arrays (dimension > 1) are supported by adding additional Attr Start Index and Addr Data Elements values prior to the Attr Data sequence.
- Attr Element Size: determines the size, in bytes, of the attribute data element. Data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding may be added to maintain word alignment.
- Attr Start Index: identifies the starting index for the array of attribute values in the Attr Data section. This field is only present when the attribute data type is an array (i.e., dimension > 0).
- Attr Data Elements: determines the number of data element values based on the start index that included in the Attr Data section. This field is only present when the attribute data type is an array (i.e., dimension > 0).
- Attr Data: contains the new value(s) that are to be applied to the targeted device configuration attribute indicated by the Attr ID. If the attribute is an array, the new value(s) are applied according to the Attr Start Index and the number of Attr Data Elements. For multidimensional arrays (dimension > 1), the list of Addr Data elements moves sequentially through the indices from left to right. The data type of the Attr Data shall match the data type of the attribute specified by the Attr ID.

EXAMPLE 1 Figure 78 defines a Set_Axis_Attributes_List Request for a simple 4-byte scalar attribute. Since the Attr Dim is 0, the Attr Start Index and Attr Data Element fields are omitted.

Set_Axis_Attributes_List Request – 0 Dimensional Data Example		
Number of Attributes = 1	-	
Attr ID = 25	Attr Dim = 0	Attr Elem Size = 4
Attr Data = ?		

IEC

Figure 78 – Set_Axis_Attributes_List Request – Single 4-byte attribute

EXAMPLE 2 Figure 79 defines a Set_Axis_Attributes_List Request for a simple 2-byte scalar attribute showing the pad byte that is added to maintain 32-bit word alignment.

Set_Axis_Attributes_List Request – 0 Dimensional Data Example		
Number of Attributes = 1	-	
Attr ID = 20	Attr Dim = 0	Attr Elem Size = 2
Attr Data = ?	(Pad)	

IEC

Figure 79 – Set_Axis_Attributes_List Request – Single 2-byte attribute

EXAMPLE 3 Figure 80 defines a Set_Axis_Attributes_List Request for the first three elements of a one dimensional array of UINTs. Since the Attr Elem Size is 2-bytes, a Pad byte is added to maintain word alignment for any connection data to follow.

Set_Axis_Attributes_List Request – 1 Dimensional Array Example		
Number of Attributes = 1	-	
Attr ID = 43	Attr Dim = 1	Attr Elem Size = 2
Attr Start Index 1 = 0	Attr Data Elements 1 = 3	
Attr Data (0) = ?	Attr Data (1) = ?	
Attr Data (2) = ?	(Pad)	

IEC

Figure 80 – Set_Axis_Attributes_List Request – 2-byte attribute array

EXAMPLE 4 Figure 81 defines a Set_Axis_Attributes_List Request for six elements of a two dimensional array of REALs. The Attr Data sequences through the leftmost array index first beginning with the Attr Start Index 1 of 0.

Set_Axis_Attributes_List Request – 2 Dimensional Array Example		
Number of Attributes = 1	-	
Attr ID = 27	Attr Dim = 2	Attr Elem Size = 4
Attr Start Index 1 = 0	Attr Data Elements 1 = 2	
Attr Start Index 2 = 2	Attr Data Elements 2 = 3	
Attr Data (0, 2) = ?		
Attr Data (0, 3) = ?		
Attr Data (0, 4) = ?		
Attr Data (1, 2) = ?		
Attr Data (1, 3) = ?		
Attr Data (1, 4) = ?		

IEC

Figure 81 – Set_Axis_Attributes_List Request – Two dimensional attribute array

The structure of Response Data Block for this service is as shown in Figure 82.

← 32-bit Word →		
Set_Axis_Attributes_List Response format		
Number of Attributes	-	
Attr ID 1	Attr Status 1	Attr Index 1 (array only)
Attr ID 2	Attr Status 2	Attr Index 2 (array only)
…		
Attr ID n	Attr Status n	Attr Index n (array only)

IEC

Figure 82 – Set_Axis_Attributes_List Response format

- Attr ID: identifies the targeted Motion Device Axis Object configuration attribute of the set list request.
- Attr Status: indicates whether the set list action for the targeted attribute was successful. An Attribute Status value is zero if the targeted attribute was successfully written. A non-zero Attr Status value indicates that an error occurred that prevented the attribute from being updated. The error codes follow the CIP Common General Status Codes (see

IEC 61158-6-2). A non-zero Attr Status value for any element in the list shall also be reflected in this service's General Status element by indicating an Attribute List Error, 0x0A.

- Attr Index: indicates the array index of the array element that is responsible for the non-zero Attr Status error code value. This value is zero when the Attr Status value is 0 (success) or when the attribute data type is not an array.

7.5.2.4 Set_Cyclic_Write_List

The Set_Cyclic_Write_List service provides a mechanism to determine the list of attributes to be passed as part of the Cyclic Write Data Block of the Controller-to-Device Connection.

The format of the Request Data Block for this service is shown in Figure 83.

← 32-bit Word →	
Set_Cyclic_Write_List Request format	
Number of Attributes	-
Attr ID 1	Attr ID 2
…	

IEC

Figure 83 – Set_Cyclic_Write_List Request format

- Number of Attributes: represents the number of attributes contained in the Cyclic Write List.
- Attr ID: identifies the specific device configuration attribute that is to be updated via the Cyclic Write Data Block of the Controller-to-Device Connection. The Attr ID determines the data type and implied semantics of the data based on the specifications for the associated attribute.

The ordering of the attribute data in the Cyclic Write Data Block is determined by the ordering of the Attr IDs in the Set_Cyclic_Write_List request. Attribute data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding may be added to maintain word alignment.

The structure of Response Data Block for this service is as shown in Figure 84.

← 32-bit Word →		
Set_Cyclic_Write_List Response format		
Number of Attributes	Cyclic Write Block ID	
Attr ID 1	Attr Status 1	-
Attr ID 2	Attr Status 2	-
…		
Attr ID n	Attr Status n	-

IEC

Figure 84 – Set_Cyclic_Write_List Response format

- Number of Attributes: represents the number of attributes contained in the requested Cyclic Write List.
- Cyclic Write Block ID: If all attributes in the requested list can be supported in the Set Cyclic Write Data Block, the Set_Cyclic_Write_List response provides a new Cyclic Write Block ID that is simply the increment of the current Cyclic Write Block ID. This new Cyclic Write Block ID can be used in the next Controller-to-Device Connection update. If the Set_Cyclic_Write_List is not successful, the Cyclic Write Block ID remains the current value.
- Attr ID: identifies the specific axis configuration attribute that was requested to be updated via the Cyclic Write Data Block of the Controller-to-Device Connection.

- Attr Status: The Attr Status value is zero if the associated attribute indicated by the Attr ID can be supported in the Set Cyclic Write Data Block. A non-zero Attr Status code indicates the specific reason why the associated attribute cannot be supported. These codes follow the CIP Common General Status Codes Codes (see IEC 61158-6-2). If the Attr Status of one or more Attribute IDs in the list indicates an error, the Set_Cyclic_Write_List request is considered unsuccessful.

7.5.2.5 Set_Cyclic_Read_List

The Set_Cyclic_Read_List service provides a mechanism to determine the list of attributes to be passed as part of the Cyclic Read Data Block of the Device-to-Controller Connection.

The format of the Request Data Block for this service is shown in Figure 85.

← 32-bit Word →	
Set_Cyclic_Read_List Request format	
Number of Attributes	-
Attr ID 1	Attr ID 2
…	

IEC

Figure 85 – Set_Cyclic_Read_List Request format

- Number of Attributes: represents the number of attributes contained in the Cyclic Read List.
- Attr ID: identifies the specific device status or signal attribute that is to be updated via the Cyclic Read Data Block of the Device-to-Controller Connection. The Attr ID determines the data type and implied semantics of the data based on the specifications for the associated attribute.

The ordering of the attribute data in the Cyclic Read Data Block is determined by the ordering of the Attr IDs in the Set_Cyclic_Read_List request. Attribute data elements shall be word aligned; 32-bit words are 32-bit aligned and 16-bit words are 16-bit aligned. Padding may be added to maintain word alignment.

The structure of Response Data Block for this service is as shown in Figure 86.

← 32-bit Word →		
Set_Cyclic_Read_List Response format		
Number of Attributes	Cyclic Read Block ID	
Attr ID 1	Attr Status 1	-
Attr ID 2	Attr Status 2	-
…		
Attr ID n	Attr Status n	-

IEC

Figure 86 – Set_Cyclic_Read_List Response format

- Number of Attributes: represents the number of attributes contained in the requested Cyclic Read List.
- Cyclic Read Block ID: If all attributes in the requested list can be supported in the Set Cyclic Read Data Block, the Set_Cyclic_Read_List response provides a new Cyclic Read Block ID that is simply the increment of the current Cyclic Read Block ID. This new Cyclic Read Block ID can be used in the next Device-to-Controller Connection update. If the Set_Cyclic_Read_List is not successful, the Cyclic Read Block ID remains the current value.
- Attr ID: identifies the specific axis configuration attribute that was requested to be updated via the Cyclic Read Data Block of the Device-to-Controller Connection.

- Attr Status: The Attr Status value is zero if the associated attribute indicated by the Attr ID can be supported in the Set Cyclic Read Data Block. A non-zero Attr Status code indicates the specific reason why the associated attribute cannot be supported. These codes follow the CIP Common General Status Codes (see IEC 61158-6-2). If the Attr Status of one or more Attribute IDs in the list indicates an error, the Set_Cyclic_Read_List request is considered unsuccessful.

7.5.2.6 Run_Motor_Test

The Run_Motor_Test service initiates various test operations on the motor to determine important motor model parameters. For active tests that enable the power structure, the axis shall be in the Stopped state for this service request to be accepted. For passive tests that do not enable the power structure, the axis may be in the Stopped state or in the following Standby states: Shutdown, Pre-charge, or Start Inhibit. The Run_Motor_Test initiates the specified test process and sets the In Process bit in the Axis Status attribute. For active tests, the axis state transitions to the Testing state. Since an active test process can take a considerable amount of time to complete, a service response is sent back to the controller as soon as the test process is initiated. This frees the service channel of the CIP Motion Connection to be used for other service requests for the duration of the process. When an active test process completes, the axis state transitions from the Testing state to either the Stopped state or the Major Faulted state. Once this state transition is complete, the In Process bit is cleared.

There are several different Test Types supported by this service(see Table 119). The Static Test applies a directionally static flux to the motor, while the Dynamic Test performs dynamic tests on the motor that produce motion. When Test Type is Calculated, the drive estimates motor model parameters based on the assumption that the motor is well matched to the power rating of the drive. The parameters measured by these tests are retrieved by the controller via the Get_Motor_Test_Data service.

Table 119 – Run_Motor_Test Request structure

Name	Type	Description of request parameter	Semantics of values
Test Type	USINT	Enumeration that specifies the type of motor test to perform:	Enumeration: 0 = static test 1 = dynamic test 2 = calculated 3 to 255 = reserved

There are no specific response parameters required for this service.

7.5.2.7 Get_Motor_Test_Data

The Get_Motor_Test_Data service provides access to the motor parameter measurements generated by the last Run_Motor_Test command. The controller uses this data to update the device with the best estimates of critical motor parameters. Parameters that are returned depend on the configured Motor Type at the time of the last Run_Motor_Test service. The Get_Motor_Test_Data service returns a 0 for parameters that were not explicitly measured or calculated by the last Run_Motor_Test procedure. Table 120 lists motor parameters that are typically measured or calculated by the Run_Motor_Test process according to the Test Type.

Table 120 – Get_Motor_Test_Data measured by Test Type

Motor parameters	Test Type		
	Static	Dynamic	Calculated
Stator Resistance (All)	yes	yes	yes
Leakage Inductance (IM, SPM)	yes	yes	yes
Flux Current (IM)	no	yes	yes
Slip Speed (IM)	no	yes	yes
Counter EMF (PM)	no	yes	no
Max Speed (IPM)	no	yes	no
Lq Inductance (IPM)	no	yes	no
Ld Inductance (IPM)	no	yes	no

To segregate standard motor test data from vendor specific motor test data, a Motor Test Data Set parameter is optionally included in the Get_Motor_Test_Data service request. If the Motor Test Data Set value is not included in the service request, the service request is for standard motor test data.

The (optional) request parameters for this service are defined in Table 121.

Table 121 – Get_Motor_Test_Data Request structure (optional)

Name	Type	Description of request parameter	Semantics of values
Motor Test Data Set	USINT	Enumeration that specifies the motor test data set to the service shall return based on the configured Motor Type.	Enumeration: 0 = standard data 1 = vendor specific data 2 to 255 = reserved

The response parameters for this service are defined in Table 122, Table 123 and Table 124.

Should the Motor Test be aborted, time-out, or fail for whatever reason, the device shall return parameter values that have been successfully measured during the test and set any parameters that were not successfully measured to zero.

Table 122 – Get_Motor_Test_Data Response standard structure (Motor Type = Induction)

Name	Type	Description of response parameter	Semantics of values
Test Status	USINT	Indicates the result of the last Motor Test command.	Enumeration: 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted
(Reserved)	USINT[3]	Pad bytes for 32-bit word alignment	
Stator Resistance	REAL	Measured phase-to-phase stator resistance of the motor.	Ohms
Leakage Inductance	REAL	Measured phase-to-phase leakage inductance of the motor.	Henries
Flux Current	REAL	Measured current to achieve full induction motor flux.	Amperes
Slip Speed	REAL	Measured slip at rated induction motor current.	r/min (rotary motor) m/s (linear motor)
(Reserved)	REAL	Pad for backward compatibility.	

Table 123 – Get_Motor_Test_Data Response standard structure (Motor Type = SPM)

Name	Type	Description of response parameter	Semantics of values
Test Status	USINT	Indicates the result of the last Motor Test command.	Enumeration: 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted
(Reserved)	USINT[3]	Pad bytes for 32-bit word alignment	
Stator Resistance	REAL	Measured phase-to-phase stator resistance of the motor.	Ohms
Leakage Inductance	REAL	Measured phase-to-phase leakage inductance of the motor.	Henries
(Reserved)	REAL	Pad for backward compatibility.	
(Reserved)	REAL	Pad for backward compatibility.	
Counter EMF	REAL	Measured CEMF voltage drop at rated speed.	Volts

Table 124 – Get_Motor_Test_Data Response standard structure (Motor Type = IPM)

Name	Type	Description of response parameter	Semantics of values
Test Status	USINT	Indicates the result of the last Motor Test command.	Enumeration: 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted
(Reserved)	USINT[3]	Pad bytes for 32-bit word alignment	
Stator Resistance	REAL	Measured phase-to-phase stator resistance of the motor.	Ohms
Lq Inductance	REAL	Measured phase-to-phase q-axis stator inductance of the motor.	Henries
Ld Inductance	REAL	Measured phase-to-phase d-axis stator inductance of the motor.	Henries
Counter EMF	REAL	Measured CEMF voltage drop at rated speed.	Volts

7.5.2.8 Run_Inertia_Test

The Run_Inertia_Test service performs an acceleration and deceleration ramp on the axis and makes timing measurements in the process (see Table 125).

Since this is an active test that enables the power structure, the axis shall be in the Stopped state for this service request to be accepted. The Run_Inertia_Test initiates the specified test process and sets the In Process bit in the Axis Status attribute. For this active test, the axis state transitions to the Testing state. Since an active test process can take a considerable amount of time to complete, a service response is sent back to the controller as soon as the test process is initiated. This frees the service channel of the CIP Motion Connection to be used for other service requests for the duration of the process. When the active test process completes, the axis state transitions from the Testing state to either the Stopped state or the Major Faulted state. Once this state transition is complete, the In Process bit is cleared.

The resultant timing measurements are accessed by the controller via the Get_Inertia_Test_Data command and ultimately used to calculate an accurate inertia value for the motor and load.

Table 125 – Run_Inertia_Test Request structure

Name	Type	Description of request parameter	Semantics of values
Test Direction	USINT	Enumeration that selects test direction.	Enumeration: 0 = forward 1 = reverse 2 to 255 = reserved
(Reserved)	USINT[3]	Pad bytes for 32-bit word alignment	
Test Velocity	REAL	Determines the maximum velocity that will be reached during the test profile.	Motor Units / s For linear motors, Motor Unit = Meters, for rotary motors, Motor Unit = Revs.
Test Torque	REAL	Determines the maximum torque that is applied during the test profile.	% Motor Rated
Test Travel Limit	REAL	Establishes the maximum excursion of the axis in Motor Units allowed in the test direction.	Motor Units For linear motors, Motor Unit = Meters, for rotary motors, Motor Unit = Revs.

There are no specific response parameters required for this service.

7.5.2.9 Get_Inertia_Test_Data

The Get_Inertia_Test_Data service provides access to the timing measurements generated by the last Run_Inertia_Test service. The controller uses this data to ultimately calculate an accurate inertia value for the motor and load. The Get_Inertia_Test_Data service returns a 0 for parameters that are not successfully measured by the last Run_Inertia_Test procedure.

There are no specific request parameters required for this service.

The response parameters for this service are defined in Table 126. Should the Inertia Test be aborted, time-out, or fail for whatever reason, the device shall return parameter values that have been successfully measured during the test and set any parameters that were not successfully measured to zero.

Table 126 – Get_Inertia_Test_Data Response structure

Name	Type	Description of response parameter	Semantics of values
Test Status	USINT	Enumeration that indicates the result of the last Run_Inertia_Test service:	Enumeration: 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process fault 5 = test reached limit 6 = test polarity incorrect 7 = test measurement error 8 = test configuration error 9 to 255 = reserved
(Reserved)	USINT[3]	Pad bytes for 32-bit word alignment	
Accel Time	REAL	Measured acceleration time to reach test velocity.	Seconds
Decel Time	REAL	Measured deceleration time to reach zero speed from test velocity.	Seconds

7.5.2.10 Run_Hookup_Test

The Run_Hookup_Test service performs a number of different tests to check for proper interface to the motor and/or feedback device. For active tests that enable the drive power structure, the axis shall be in the Stopped state for this service request to be accepted. For passive tests that do not enable the power structure, the axis may be in the Stopped state or in the following Standby states: Shutdown, Pre-charge, or Start Inhibit. The Run_Hookup_Test initiates the specified test process and sets the In Process bit in the Axis Status attribute. For active tests, the axis state transitions to the Testing state. Since an active test process can take a considerable amount of time to complete, a service response is sent back to the controller as soon as the test process is initiated. This frees the service channel of the CIP Motion Connection to be used for other service requests for the duration of the process. When an active test process completes, the axis state transitions from the Testing state to either the Stopped state or the Major Faulted state. Once this state transition is complete, the In Process bit is cleared.

There are several different Test Types supported by this service. The following is a brief description of each of these Test Types.

- The Motor/Feedback test is an active test that attempts to move the motor in the forward direction for a specified distance given by the Test Distance, or for a specified time given by the Test Time. Forward direction is defined as the direction the motor moves when following the normal UVW motor phase sequencing. During the test the device monitors the change in feedback counts over the duration of the test for applicable feedback channels. When the Feedback Mode is set for Dual Feedback or Load Feedback, both feedback channels are tested.

- The Feedback test is a passive test that monitors the axis while an external agent moves it and indicates success if the axis feedback count exceeds the distance specified by the Test Distance. Which feedback channel to monitor is determined by the Feedback Channel parameter.
- The Marker test is a passive test that monitors the axis positioning feedback channel's marker signal while an external agent moves the axis through the marker location and indicates success if a marker pulse is detected. Which marker signal to test is determined by the Feedback Channel parameter.
- The Commutation Test, which applies only to PM motors, is an active test that applies current to the motor to align the rotor, check for proper phasing of a UVW encoder or Hall sensor if applicable, and measure the Commutation Offset. The Commutation test always targets Feedback 1, the motor feedback channel.

A time-out limit for the Run_Hookup_Test service is applied by the device vender to handle the case where the success criteria cannot be met. This is typically around 30 s.

The resultant data generated by these test can be accessed by the Get Test Data service and used by the controller to automatically set the proper polarities of the feedback interface and the motor interface.

The request parameters for this service are defined in Table 127.

Table 127 – Run_Hookup_Test Request structure

Name	Type	Description of request parameter	Semantics of values
Test Type	USINT	Determines the specific hookup test to run:	Enumeration: 0 = motor & feedback test 1 = feedback test 2 = marker test 3 = commutation test (O) 4 to 255 = reserved
Feedback Channel	USINT	Determines the logical channel number targeted for the feedback test. When not applicable, this parameter shall be set to 0.	Channel Number
(Reserved)	USINT[2]	Pad bytes for 32-bit word alignment	
Test Distance	REAL	Establishes the distance that the axis needs to travel, in Motor or Feedback n Units to indicate a successful test. Test Type determines units. A Test Type of "motor & feedback test" uses Motor Units. A Test Type of "feedback test" uses Feedback n Units where n is determined by Feedback Channel. Test Distance is valid if the axis is not configured Encoderless/Sensorless operation. When not applicable, this parameter shall be set to 0.	Motor or Feedback n Units
Test Time	REAL	Establishes the time duration of axis motion during the test. Test Time is valid if the axis is configured Encoderless/Sensorless operation. When not applicable, this value shall be set to 0.	seconds

There are no specific response parameters required for this service request.

7.5.2.11 Get_Hookup_Test_Data

The Get_Hookup_Test_Data service provides access to the hookup test results generated by the last Run_Hookup_Test service (see Table 128). The controller uses this data to flag a wiring problem to the user and to calculate polarity configuration bit parameters for the axis. The results of this test are independent of the configured values for motor, feedback, and commutation polarity. The Get_Hookup_Test_Data service returns a 0 for parameters that are not determined by the last Run_Hookup_Test procedure.

Table 128 – Get_Hookup_Test_Data measured by Test Type

Get_Hookup_Test parameters	Test Type			
	Motor/Feedback	Feedback	Marker	Commutation
Commutation Offset	no	no	no	yes
Commutation Polarity	no	no	no	yes
Feedback 1 Direction	Yes [a]	yes [b]	no	no
Feedback 2 Direction	yes [a]	yes [b]	no	no

[a] When running a Motor/Feedback Hookup Test, Feedback 1 Direction is always measured unless the Feedback Mode is set to No Feedback (encoderless/sensorless operation). Feedback 2 Direction is measured when the Feedback Mode is set to either Dual Feedback or Load Feedback.

[b] When running a Feedback Hookup Test, either Feedback 1 Direction or Feedback 2 Direction are measured based on Feedback Channel specified by last Run_Hookup_Test.

There are no specific request parameters required for this service.

The response parameters for this service are defined in Table 129. Should the Hookup Test be aborted, time-out, or fail for whatever reason, the device shall return parameter values that have been successfully measured during the test and set any parameters that were not successfully measured to zero.

Table 129 – Get_Hookup_Test_Data Response structure

Name	Type	Description of response parameter	Semantics of values
Test Status	USINT	Enumeration that indicates the result of the last Hookup Test command:	Enumeration: 0 = test process successful 1 = test in progress 2 = test process aborted 3 = test process timed-out 4 = test process faulted 5 = test failed no feedback 1 6 = test failed no feedback 2 7 to 255 = reserved
Feedback 1 Direction	USINT	Enumeration that indicates the direction of axis motion associated with the last Hookup Test command during a Feedback 1 or Motor/Feedback Test, if applicable. Positive direction indicates the test generated positive incrementing feedback 1 counts.	Enumeration: 0 = Positive 1 = Negative 2 to 255 = reserved
Feedback 2 Direction	USINT	Enumeration that indicates the direction of axis motion associated with the last Hookup Test command during a Feedback 2 or Motor/Feedback Test, if applicable. Positive direction indicates the test generated positive incrementing feedback 2 counts.	Enumeration: 0 = Positive 1 = Negative 2 to 255 = reserved
Commutation Polarity	USINT	When performing a Commutation Test on a PM motor the test checks if the UVW phasing of the Encoder or Hall Sensor match the phasing of the Motor. If the phasing matches, Commutation Polarity is Normal. If it is determined that the phasing for the motor and commutation device do not match, this parameter reports that the UVW phasing is Inverted. See Commutation Polarity attribute for further details.	Enumeration: 0 = Normal 1 = Inverted 2 to 255 = reserved
Commutation Offset	REAL	Offset measured during the Commutation Test that shall be applied to the motor position accumulator in order to align the Electrical Angle signal with motor stator windings. See Commutation Offset attribute.	Electrical Degrees

7.6 Behavior

7.6.1 State model

7.6.1.1 General

The Motion Device Axis Object State Model is based on elements of the S88 and Pack/ML standard state models. The current state of the Motion Device Axis Object instance is indicated by the Axis State attribute (Attribute ID = 650). State transitions can be initiated either directly via the Axis Control request mechanism or by conditions that occur during device operation. Figure 87 shows the basic operating states of the Motion Device Axis Object when actively controlling axis motion (Control Mode != No Control). Shaded regions show mapping of Axis States to corresponding Identity Object states.

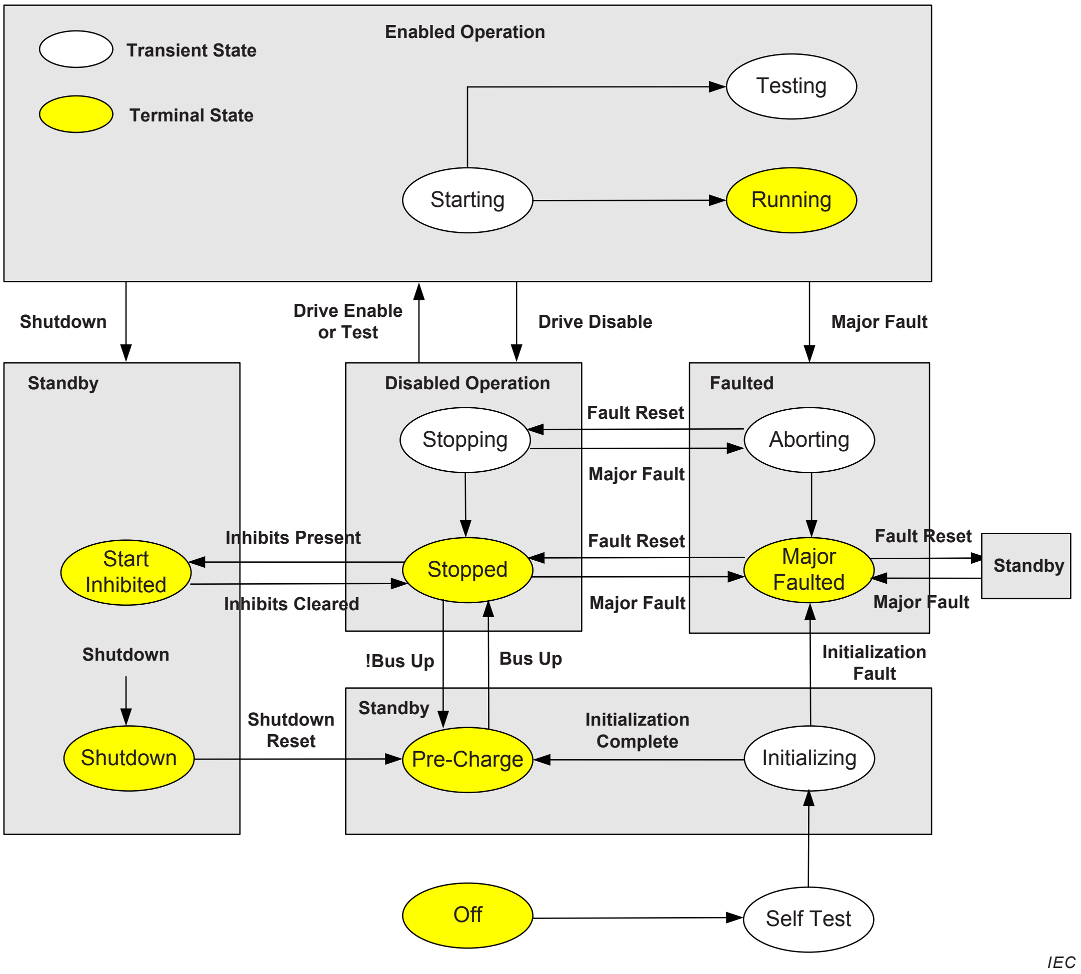

Figure 87 – Motion Device Axis Object State Model

Valid transitions for the Axis State Model are explicitly defined in Table 130.

Table 130 – Axis State Machine transitions

Current state	Event	Conditions	Next state
Off	Power Up		Self Test
Self Test	Self Test Complete		Initializing
Initializing	Initialization Fault		Major Faulted
Initializing	Initialization Complete		Pre-Charge
Shutdown	Major Fault or Abort		Major Faulted
Shutdown	Shutdown Reset		Pre-Charge
Pre-Charge	Shutdown		Shutdown
Pre-Charge	Major Fault		Major Faulted
Pre-Charge	Bus Up		Stopped
Start Inhibited	Shutdown		Shutdown
Start Inhibited	Major Fault		Major Faulted
Start Inhibited	Inhibits Cleared		Stopped
Major Faulted	Fault Reset	SD = 1	Shutdown
Major Faulted	Fault Reset	SD = 0, BU = 0	Pre-Charge
Major Faulted	Fault Reset	SD = 0, BU = 1, SI > 0	Start Inhibited
Major Faulted	Fault Reset	SD = 0, BU = 1, SI = 0	Stopped
Major Faulted	Reconnection		Initializing
Stopped	!Bus Up		Pre-Charge
Stopped	Shutdown		Shutdown
Stopped	Major Fault		Major Faulted
Stopped	Start Inhibit		Start Inhibit
Stopped	Enable		Starting
Stopped	Test		Starting
Starting	Shutdown		Shutdown
Starting	Major Fault or Abort		Aborting
Starting	Disable		Stopping
Starting	Start Complete	IP = 0	Running
Starting	Start Complete	IP = 1	Testing
Stopping	Shutdown		Shutdown
Stopping	Stop Complete		Stopped
Stopping	Major Fault		Aborting
Stopping	Enable	Flying Start Enabled	Starting
Aborting	Stop Complete		Major Faulted
Aborting	Fault Reset		Stopping
Testing	Shutdown		Shutdown
Testing	Major Fault or Abort		Aborting
Testing	Disable		Stopping
Running	Shutdown		Shutdown
Running	Major Fault or Abort		Aborting
Running	Disable		Stopping
Any State	Connection Close		Initializing
Any State	Connection Loss		Major Faulted

Some of the axis state machine transitions in Table 130 have dependencies on the current status conditions of the axis defined in Table 131.

Table 131 – Axis State Machine conditions

Condition	Symbol	Description
Bus Up	BU	This Axis Status bit is set if the DC bus is charged within the operating range.
Shutdown	SD	This Axis Status bit is set if there is an active shutdown. This is different than the shutdown state. The Shutdown status bit can be active in the faulted state.
In Process	IP	This Axis Status bit indicates a test process associated with service request is active. Note that the In Process bit may be active without the axis state machine being in the testing state.
Start Inhibit	SI	This represents the set of active Start Inhibits. An SI value of 0 indicates no active Start Inhibits. Note that start inhibits may be present without the axis state machine being in the start inhibit state.

When the Motion Device Axis Object is not actively controlling axis motion (Control Mode = No Control), the state diagram in Figure 87 reduces to the one shown in Figure 88 for a Feedback Only axis or CIP Motion Encoder device type.

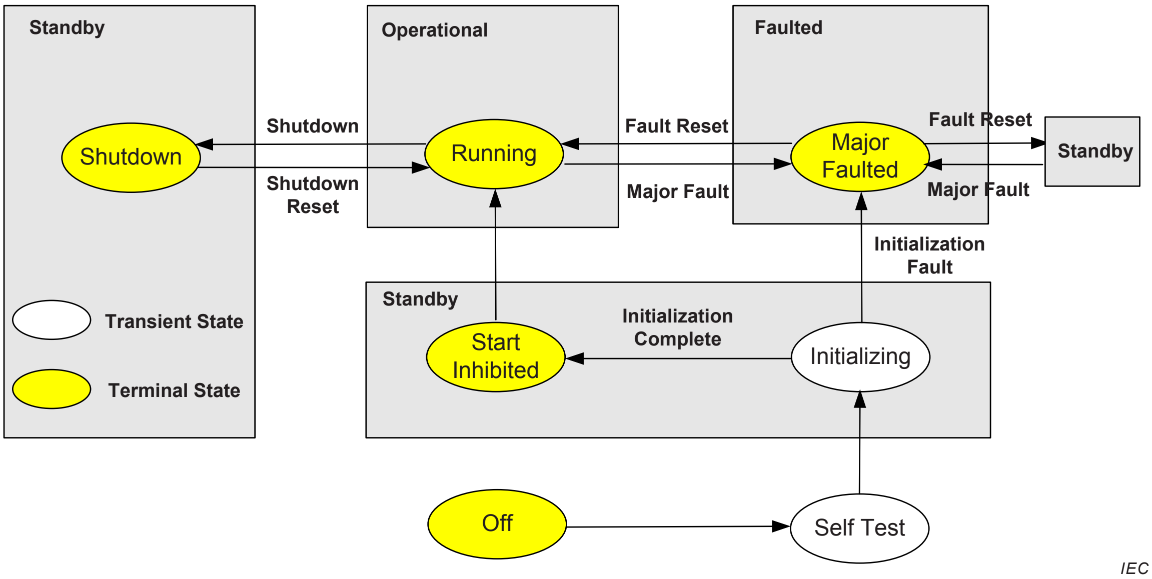

Figure 88 – Motion Device Axis Object State Model for Feedback Only

Valid transitions for the Axis State Model of a Feedback Only axis or CIP Motion are explicitly defined in Table 132.

Table 132 – Axis State Machine transitions (Feedback Only)

Current state	Event	Conditions	Next state
Off	Power Up		Self Test
Self Test	Self Test Complete		Initializing
Initializing	Initialization Fault		Major Faulted
Initializing	Initialization Complete		Start Inhibited
Shutdown	Major Fault		Major Faulted
Shutdown	Shutdown Reset		Running
Start Inhibited	Major Fault		Major Faulted
Start Inhibited	Inhibits Cleared		Running
Major Faulted	Fault Reset	SD = 1	Shutdown
Major Faulted	Fault Reset	SD = 0, SI > 0	Start Inhibited
Major Faulted	Fault Reset	SD = 0, SI = 0	Running
Major Faulted	Reconnection		Initializing
Running	Shutdown		Shutdown
Running	Major Fault		Major Faulted
Any State	Connection Close		Initializing
Any State	Connection Loss		Major Faulted

When the Motion Device Axis Object is associated with a CIP Motion Converter, the Active Control state diagram in Figure 87 reduces to the one shown in Figure 89.

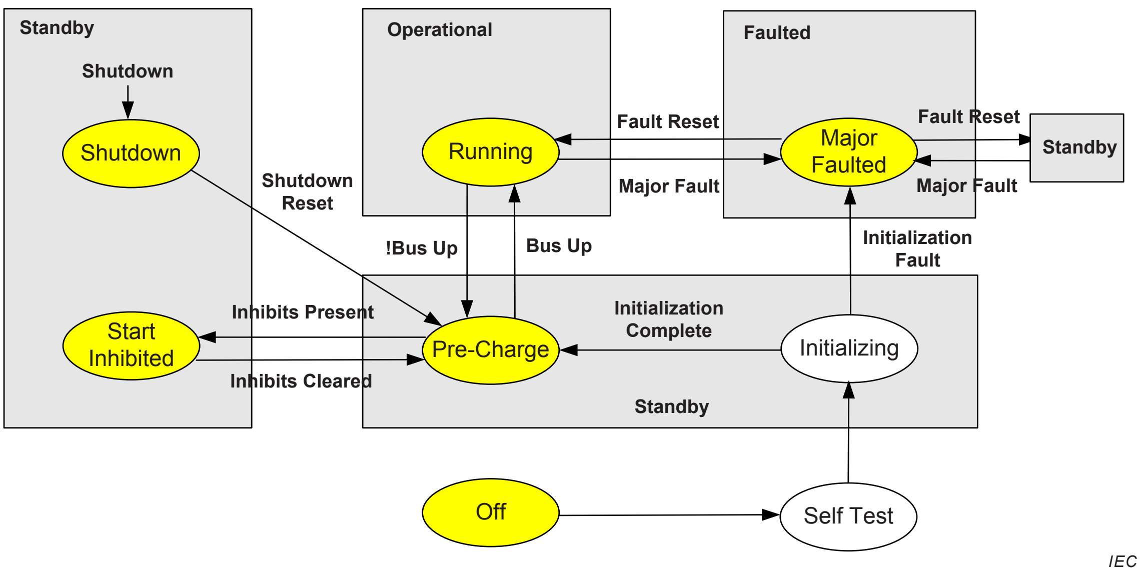

Figure 89 – Motion Device Axis Object State Model for Converter

Valid transitions for the Axis State Model of a CIP Motion Converter axis are explicitly defined in Table 133.

Table 133 – Axis State Machine transitions (Converter)

Current state	Event	Conditions	Next state
Off	Power Up		Self Test
Self Test	Self Test Complete		Initializing
Initializing	Initialization Fault		Major Faulted
Initializing	Initialization Complete		Pre-Charge
Shutdown	Major Fault		Major Faulted
Shutdown	Shutdown Reset		Pre-Charge
Start Inhibited	Inhibits Cleared		Pre-Charge
Pre-Charge	Start Inhibit		Start Inhibited
Pre-Charge	Shutdown		Shutdown
Pre-Charge	Major Fault		Major Faulted
Pre-Charge	Bus Up		Running
Major Faulted	Fault Reset	SD = 1	Shutdown
Major Faulted	Fault Reset	SD = 0	Pre-Charge
Major Faulted	Reconnection		Initializing
Running	!Bus Up		Pre-Charge
Running	Shutdown		Shutdown
Running	Major Fault		Major Faulted
Any State	Connection Close		Initializing
Any State	Connection Loss		Major Faulted

Fault Reset State transition precedence

In the Major Faulted state shown in the above Axis State Diagrams, the axis may transition to one of several different states in response to a Fault Reset event or the axis may transition right back to the Major Faulted state if there is a persistent fault condition present. Which state the axis transitions to depends on other state/status conditions of the axis. In fact, it may be possible for more than one state condition to be present at the same time, e.g. Shutdown, Start Inhibited, etc. Since the axis state model can only represent one state at any given time, the state of the axis is determined according to the following precedence:

- Major Faulted
- Shutdown
- Pre-Charge
- Start Inhibited
- Stopped

This and other state transition behavior is discussed in more detail in 7.6.1.2 and 7.6.2.

7.6.1.2 State Control

The primary method for changing the state of a Motion Device Axis Object instance is via the Axis Control/Axis Response mechanism that is built into the CIP Motion I/O Connection Header. Changing the state of the motion axis is simply performed by placing the appropriate Request Code in the Axis Control element of the Controller-to-Device Connection Header. The Control Request codes are defined in the CIP Motion device profile and reproduced in Table 134 for convenience.

Table 134 – Axis Control Request code

Request Code	Requested operation
0	No Request
1	Enable Request
2	Disable Request
3	Shutdown Request
4	Shutdown Reset Request
5	Abort Request
6	Fault Reset Request
7	Stop Process Request
8	Change Actual Position Reference Request
9	Change Command Position Reference Request
10 to 126	(Reserved)
127	Cancel Request
128 to 255	(Vendor Specific)

When a state transition is requested via the Axis Control element of the Controller-to-Device Connection, the device initiates the state transition and then acknowledges the transition request via the Axis Response attribute when the requested state transition completes or is determined unsuccessful. The Acknowledge Codes for the Axis Response element are the same as the Request Codes for the Axis Control element as defined in the CIP Motion device profile (see Table 135).

Table 135 – Axis Response Acknowledge codes

Acknowledge Code	Axis Response
0	No Acknowledge
1	Enable Acknowledge
2	Disable Acknowledge
3	Shutdown Acknowledge
4	Shutdown Reset Acknowledge
5	Abort Acknowledge
6	Fault Reset Acknowledge
7	Stop Process Acknowledge
8	Change Actual Position Reference Acknowledge
9	Change Command Position Reference Acknowledge
127	Cancel Acknowledge

The criteria for successful completion of the requested operation depend on the specific Request Code as shown in Table 136.

Table 136 – Completion criteria for requested operation

Request Code	Completion criteria
1	Axis successfully transitions to the Running state with the Tracking Command bit set in the Axis Status attribute.
2	Axis successfully transitions out of Enabled Operation with the Tracking Command bit clear in the Axis Status attribute.
3	Axis successfully transitions to Shutdown state or the Faulted state, with the Shutdown bit set in the Axis Status attribute.
4	Axis successfully transitions to Pre-charge, Stopped, or Inhibited state with the Shutdown bit clear in the Axis Status attribute and the Axis Faults attribute also momentarily clear. [a] If the axis is not in the Faulted or Shutdown states, the Shutdown Reset operation has no effect on the axis.
5	Axis successfully transitions to Aborting or Faulted state with the Controller Initiated Exception bit set in the Axis Faults attribute.
6	Axis successfully transitions to Stopped, Stopping, Inhibited, or Shutdown state with Axis Faults attribute momentarily clear. [a] If the axis is not in the Faulted state, the Fault Reset operation has no effect on the axis.
7	Axis successfully stops an active process, clearing the In Process bit in the Axis Status attribute.
8	Axis successfully applied Displacement to Actual Position to transition to new position reference system.
9	Axis successfully applied Displacement to Command Position to transition to new position reference system.
127	Always completes.

[a] In the case of a Fault Reset or a Shutdown Reset, if any fault generating exception conditions still exist, the axis transitions out of the Faulted state only momentarily since the axis shall immediately fault again and return to the Faulted state.

If the device determines that the requested state transition cannot be successfully completed, the device acknowledges the Control Request with a non-zero Error Code in the Response Status element. Possible error conditions are shown in Table 137.

Table 137 – Possible error conditions for requested operation

Request Code	Possible error conditions
1	Axis enters Faulted or Start Inhibited state after the request was received, before reaching the Running state.
2	None
3	None
4	None
5	None
6	None
7	None
8	None
127	None.

When the controller receives the Acknowledge Code, and there is no pending Control Request to initiate, it then zeroes the matching Request Code. When the device sees the Request Code zeroed, it clears the Acknowledge Code and the associated Response Status.

If the axis cannot transition to the requested state, the specific error condition is indicated by a non-zero Response Status attribute value that is passed along with the Axis Response at part of the Device-to-Controller Connection.

This behavior is further described in Table 138 and Table 139.

Table 138 – Successful Axis Control Request Cycle

Controller action	Connection data	Device action
Controller sets Request Code in the Axis Control element to request a state change.	→ Axis Control = x	Device sees non-zero Request Code, x, in Axis Control element and initiates the requested state change.
	← Axis Response = x ← Response Status = 0	If state change operation is successful, the device acknowledges the state change by setting the Acknowledge Code in the Axis Response element to the originating Request Code, x. The Response Status element is set to 0 indicating success.
Controller sees the non-zero Acknowledge Code in Axis Response element and zeroes the originating Request Code in the Axis Control element. The requested state change was successful as evidenced by the zero Response Status.	→ Axis Control = 0	
The state change request transaction is now complete. Another Axis Control request can be processed at this time.	← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.

Table 139 – Unsuccessful Axis Control Request Cycle

Controller action	Connection	Device action
Controller sets Request Code in the Axis Control element to request a state change.	→ Axis Control = x	Device sees non-zero Request Code, x, in Axis Control element and initiates the requested state change.
	← Axis Response = x ← Response Status = Error Code	If state change operation is unsuccessful, the device indicates the problem via a non-zero Response Status Error Code along with the Acknowledge Code matching the originating Request Code, x.
Controller sees the non-zero Response Status associated with the Request Code, x, in the Axis Response, handles the error condition based on the Error Code, and clears the originating Request Code in the Axis Control element.	→ Axis Control = 0	
The state change request transaction is now complete. Another Axis Control request can be processed at this time.	← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.

If there is a pending Control Request when the Acknowledge Code for the previous Control Request cycle is processed by the controller, the controller can immediately initiate the pending Control Request. In this way, the delay in executing the pending Control Request is minimized. This behavior is further described in Table 140.

Table 140 – Pending Axis Control Request Cycle

Controller action	Connection data	Device action
Controller sets Request Code in the Axis Control element to request a state change x.	→ Axis Control = x	Device sees non-zero Request Code, x, in Axis Control element and initiates the requested state change.
Controller has a pending state change request y as it receives the Acknowledge Code from state change request x	← Axis Response = x ← Response Status = 0	If state change operation is successful, the device acknowledges the state change by setting the Acknowledge Code, x, in the Axis Response element to the originating Request Code, x. The Response Status element is set to 0 indicating success.
Based on the non-zero Acknowledge Code in Axis Response element, the Controller immediately sets Request Code y in the Axis Control element to request the pending state change.	→ Axis Control = y	Device sees non-zero Request Code, y, in Axis Control element and initiates the requested state change.
	← Axis Response = y ← Response Status = 0	If state change operation is successful, the device acknowledges the state change by setting the Acknowledge Code, y, in the Axis Response element to the originating Request Code, x. The Response Status element is set to 0 indicating success.
Controller sees the non-zero Acknowledge Code in Axis Response element and, with no pending state change requests, zeroes the originating Request Code in the Axis Control element. The requested state change was successful as evidenced by the zero Response Status.	→ Axis Control = 0	
The last state change request transaction is now complete. Another Axis Control request can be processed at this time.	← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.

Object state transitions can also be initiated by service requests such as the Test service requests. In general, these state change requests use services because they require one or more parameters to be passed to the device in order to initiate the state change or are not real-time performance critical.

In some cases the Control Request operation can take a considerable amount of time to perform, for example an Enable Request. To prevent such a request from monopolizing the Control Request mechanism, a special Cancel Request code is defined. The Cancel Request is immediately acknowledged by the device to complete the handshaking cycle, thereby, allowing a subsequent Control Request (see Table 141).

Table 141 – Cancel Request Cycle

Controller action	Connection data	Device action
Controller sets Request Code in the Axis Control element to request a state change, an Enable Request in this example.	→ Axis Control = Enable Req	Device sees the Enable Request Code in Axis Control element and initiates the requested state change. This state change happens to require significant time to complete.
Meanwhile, the controller determines that it is necessary to override the last Enable state change request with a Disable Request. To free up the Control Request mechanism to send the Disable Request, the Axis Control element shall first be set to Cancel Request	→ Axis Control = Cancel Req ← Axis Response = Cancel Ack ← Response Status = 0	Device sees the Cancel Request Code in Axis Control element and immediately sends the Cancel Acknowledge Code in the Axis Response element. The device shall continue the original state change operation until another Control Request is received.
Controller sees the non-zero Acknowledge Code in Axis Response element and zeroes the originating Request Code in the Axis Control element. The Cancel Request transaction is now complete. Another Axis Control request can now be processed at this time.	→ Axis Control = 0 ← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.
Controller sets Request Code in the Axis Control element to request a Disable Request.	→ Axis Control = Disable Req	Device sees non-zero Request Code in Axis Control element and initiates the requested Disable state change. This may override any cancelled state change operation that is in process.
	← Axis Response = Disable Ack ← Response Status = 0	Once the Disable state change is successful, the device acknowledges the state change by setting the Acknowledge Code in the Axis Response element to the originating Request Code. The Response Status element is set to 0 indicating success.
Controller sees the non-zero Acknowledge Code in Axis Response element and zeroes the originating Request Code in the Axis Control element. The requested state change was successful as evidenced by the zero Response Status.	→ Axis Control = 0	
The state change request transaction is now complete. Another Axis Control request can be processed at this time.	← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.

7.6.1.3 Changing position reference system

When the scaling function is enabled in the drive (see Control Mode definition) the Axis Control/Axis Response mechanism is used to coordinate transition of Actual Position and Command Position in both the controller and the drive to a new absolute position reference system. The ability to change the absolute position reference system is fundamental to Homing and Redefine Position operations associated with Position Control Systems.

Homing and Redefine Position operations begin by determining a Displacement value between the current position reference system and the new, or desired, position reference system. The trick is to apply the Displacement value to the Actual Position and Command Position values in the drive and the controller in such a way that it doesn't disrupt motion in progress.

The process of coordinating the transition to the new position reference system begins by placing the Change Actual Position Request Code in the Axis Control element of the Controller-to-Device Connection and a Position Displacement element in the Cyclic Data. When the drive receives the Change Actual Position Request, it applies the Position Displacement to the Actual Position in the drive. The next Device-to-Controller update passes Actual Position based on the new position reference system and the Position Displacement to the controller along with the Change Actual Position Acknowledge Code in the Axis Response element. The controller recognizes the Change Actual Position Acknowledge Code, so when computing the actual position delta (used in gearing and camming), the Position Displacement value is applied to compensate for the position reference change.

After the Change Actual Position Acknowledge Code is received by the controller, it then applies the Position Displacement value to the Command Position and sends the Command Position based on the new position reference system to the drive in the next Controller-to-Device update along with the Change Command Position Request Code in the Axis Control element. Again the Position Displacement is included as an element of the Cyclic data. Based on receiving the Change Command Position Request, the drive applies the Position Displacement to compensate for the position reference change when computing the command position delta.

The change-over to the new position reference system is officially completed with the Change Command Position Acknowledge passed back to the controller in the next Drive-to-Controller update. No Position Displacement needs to be passed to the controller in this update. Table 142 further defines the complete redefine position reference cycle.

Table 142 – Redefine Position Reference Cycle

Controller action	Connection data	Device action
Controller sets Request Code in the Axis Control element to request a Change Actual Position Request, chg actpos, and updates drive with Displacement.	→ Axis Control = chg actpos → Cyclic Data = displacement	Device sees non-zero Request Code, a Change Actual Position Request, in Axis Control element applies Displacement to Actual Position in the drive device.
Controller receives the Change Actual Position Acknowledge from the drive and applies Displacement to delta actual position calculation to correct for position reference change. Controller applies Displacement to shift Command Position to the new position reference system.	← Axis Response = chg actpos ← Response Status = 0 ← Cyclic Data = new actpos ← Cyclic Data = displacement	The drive device acknowledges the request by passing the Change Actual Position Acknowledge Code, chg actpos, back to the controller along with the new Actual Position in the new position reference system. The Response Status element is set to 0 indicating success.
Controller sets the Request Code to Change Command Position Request, chg compos, and updates drive along with the new Command Position based on the new position reference system and the Displacement.	→ Axis Control = chg compos → Cyclic Data = new compos → Cyclic Data = displacement	Drive receives the Change Command Position Acknowledge from the controller and applies Displacement to delta command position calculation to correct for position reference change
Controller indicates that the originating Home or Redefine Position operation is complete.	← Axis Response = chg compos ← Response Status = 0	The drive device acknowledges the Change Command Position Request change by passing the Acknowledge Code, chg compos, back to the controller in the Axis Response element. The Response Status element is set to 0 indicating success.
Controller sees the non-zero Acknowledge Code in Axis Response element and, with no pending state change requests, zeroes the originating Request Code in the Axis Control element.	→ Axis Control = 0	
The last state change request transaction is now complete. Another Axis Control request can be processed at this time.	← Axis Response = 0 ← Response Status = 0	Device sees a zero Request Code in the Axis Control element and zeroes the associated Acknowledge Code in the Axis Response element.

7.6.2 State behavior

7.6.2.1 General

Subclauses 7.6.2.2 to 7.6.2.14 offer a detailed description of each of the states and state transitions of the Motion Device Axis Object state model.

7.6.2.2 Off state

This is the state of the Motion Device Axis Object with power off.

7.6.2.3 Self Test state

When power is applied to the device, or the device is reset, the device typically goes through a series of self-test diagnostics and internal device parameters are set to their power-up default values. Once completed successfully, the device and all its associated axis instances transition to the Initializing state and are ready for initialization by the associated controller. If unsuccessful, the device and all its associated axis instances transition immediately to the Major Faulted state by declaring an Initialization Fault that is classified as Unrecoverable according to the terminology defined by the Identity Object. Clearing this fault can only be accomplished through a power cycle and is most likely the result of a device hardware problem.

If the device supports stand-alone operation under local control with local configuration data, the device is free to transition from the Self-test state to the Pre-charge state and on to the Stopped state. If the device receives a subsequent Forward_Open service to open a CIP Motion Connection, the drive device shall disable all axes and transition back to the Initializing state, following the state sequence outlined below.

If the device does not support stand-alone operation and depends on remote configuration data to be supplied over a CIP connection, the device shall transition to the Initializing state and wait (Standby) for the Forward_Open service from the controller to open the CIP Motion Connection.

7.6.2.4 Initializing state

During the Initializing state, the device waits for the CIP Motion Connections to the device to be established by the controller via a Forward_Open service. Once the Forward_Open service is successfully processed, the device initializes all attributes to their factory default values, resets all active faults, resets applicable axis status conditions including the shutdown bit, in preparation for device attribute configuration.

Once connections are established, the controller sends Set services to the device to set the Motion Device Axis Object configuration attributes to values stored in the controller. Any configuration error encountered during this process, for example due to value out of range or value not applicable, shall be handled by erring the Set service response; not by generating an Initialization Fault. The controller shall not complete the configuration process unless all configuration attributes have been successfully acknowledged.

If the device supports synchronous operation, the controller then synchronizes with the device using the Group_Sync service. Once this entire process has been completed successfully, the device and all its associated axis instances transition to the Pre-charge state. If a problem is found during this initialization process that is beyond the scope of a Set service error, the device generates an Initialization Fault. An Initialization Fault is viewed as an unrecoverable fault so clearing the fault can only be accomplished through a power cycle or a device reset service to the associated Identity Object.

If the CIP Motion Connection is intentionally closed for any reason during operation via a Forward_Close service, the device clears all active faults and alarms and returns to the Initializing state. If the CIP Motion Connection is lost for any other reason during operation, the device generates a Node Fault and transitions to the Major Faulted state. In either case the device shall wait for the CIP Motion Connections to the device to be re-established by the controller via a Forward_Open service.

The Initializing state is classified as an Identity Object Standby state and, therefore, the device shall insure that all associated power structures are disabled.

7.6.2.5 Pre-Charge state

In the Pre-Charge state, when applicable, the device is waiting for the DC Bus to fully charge (DC Bus Up status bit is clear). Once the DC Bus reaches an operational voltage level (DC Bus Up status bit is set) the axis either transitions to the Stopped state (drive axis) or to the Running state (converter axis). The drive device's inverter power structure is always disabled in this state (Power Structure Enabled status bit clear). Any attempt for the controller to enable a drive via the Axis Control mechanism while it is in the Pre-charge state is reported back to the controller as an error in the Response Status and the axis remains in the Pre-charge state.

The Pre-Charge state is classified as an Identity Object Standby state and, therefore, requires that the associated inverter power structure, if applicable, is disabled.

7.6.2.6 Stopped state

In the Stopped state, the device's inverter power structure shall either be disabled and free of torque (Power Structure Enabled status bit clear) or held in a static condition via an active control loop (Power Structure Enabled status bit set). No motion can be initiated by the device in the Stopped State nor can the device respond to a planner generated command references (Tracking Command status bit clear). In general, the axis shall be at rest, but if an external force or torque is applied to the load, a brake may be needed to maintain the rest condition. In the Stopped state, main power is applied to the device and the DC Bus is at an operational voltage level. If there are any Start Inhibit conditions detected while in this state, the axis transitions to the Start Inhibited state. If an Enable request or one of the Run Test service requests is applied to an axis in the Stopped state, the motion axis transitions to the Starting state.

7.6.2.7 Starting state

When an Enable request is given to an axis in the Stopped, or Stopping state when executing a Flying Start, the axis immediately transitions to the Starting state. In this state, the device checks various conditions before transitioning to the Running state. These conditions can include Brake Release delay time and Induction Motor flux level. The device control and power structures are activated during the Starting state (Power Structure Enabled status bit set) but the command reference is set to a local static value and will not track the command reference derived from the motion planner (Tracking Command status bit clear). If all the starting conditions are met, the axis state transitions to either the Running state or the Testing state.

7.6.2.8 Running state

The Running state is where the work gets done. In this state, the device's power structure is active (Power Structure Enabled status bit set) and the selected Control Mode is enabled and actively tracking command data from the controller based motion planner output to affect axis motion (Tracking Command status bit set). The motion axis remains in the Running state until either a fault occurs or it is explicitly commanded to stop via an Axis Control request.

In the case of an axis with no active control function (Control Mode, = No Control), the Running state simply indicates that the device is fully operational (Power Structure Enabled status bit and the Tracking Command status bit are both clear). The motion axis remains in the Running state until either a fault occurs or it is explicitly commanded to Shutdown via an Axis Control request.

In the Running state, only a subset of all axis instance configuration attributes are allowed by the device to be modified. Table 143 lists all attributes that the device is required to support in the Running state. Access to all other configuration attributes in the Running state is left to the device vendor's discretion.

Table 143 – Running State – Configurable attributes

Attr. ID	Attribute name	B	E	F	P	V	T
370	Skip Speed 1	-	-	O	-	-	-
371	Skip Speed 2	-	-	O	-	-	-
372	Skip Speed 3	-	-	O	-	-	-
373	Skip Speed Band	-	-	O	-	-	-
374	Ramp Velocity – Positive	-	-	O	-	O	-
375	Ramp Velocity – Negative	-	-	O	-	O	-
376	Ramp Acceleration	-	-	O	-	O	-
377	Ramp Deceleration	-	-	O	-	O	-
378	Ramp Jerk Control	-	-	O	-	O	-
440	Kvff	-	-	-	R	-	-
441	Kpp	-	-	-	R	-	-
442	Kpi	-	-	-	R	-	-
443	Position Lock Tolerance	-	-	-	R	-	-
444	Position Error Tolerance	-	-	-	R	-	-
445	Position Error Tolerance Time	-	-	-	O	-	-
446	Position Integrator Control	-	-	-	R	-	-
447	Position Integrator Preload	-	-	-	O	-	-
460	Kaff	-	-	-	R	R	-
461	Kvp	-	-	-	R	R	-
462	Kvi	-	-	-	R	R	-
464	Kdr	-	-	O	O	O	-
465	Velocity Error Tolerance	-	-	-	O	O	-
466	Velocity Error Tolerance Time	-	-	-	O	O	-
467	Velocity Integrator Control	-	-	-	R	R	-
468	Velocity Integrator Preload	-	-	-	O	O	-
469	Velocity Low Pass Filter Bandwidth	-	-	-	O	O	-
470	Velocity Threshold	-	O	O	O	O	O
471	Velocity Lock Tolerance	-	-	O	O	O	-
472	Velocity Standstill Window	-	R	R	R	R	R
473	Velocity Limit – Positive	-	-	O	O	O	-
474	Velocity Limit – Negative	-	-	O	O	O	-
485	Acceleration Limit	-	-	-	O	O	O
486	Deceleration Limit	-	-	-	O	O	O
496	Kj	-	-	-	R	R	O
498	Friction Compensation – Sliding	-	-	-	O	O	O
499	Friction Compensation – Static	-	-	-	O	O	O
500	Friction Compensation – Viscous	-	-	-	O	O	O
502	Torque Low Pass Filter Bandwidth	-	-	-	O	O	O
503	Torque Notch Filter Frequency	-	-	-	O	O	O
504	Torque Limit – Positive	-	-	-	R	R	R
505	Torque Limit – Negative	-	-	-	R	R	R
506	Torque Rate Limit	-	-	-	O	O	O
507	Torque Threshold	-	-	-	O	O	O

Attr. ID	Attribute name	B	E	F	P	V	T
508	Overtorque Limit	-	-	O	O	O	O
509	Overtorque Limit Time	-	-	O	O	O	O
510	Undertorque Limit	-	-	O	O	O	O
511	Undertorque Limit Time	-	-	O	O	O	O
553	Current Vector Limit	-	-	O	O	O	O
554	Kqp	-	-	-	O	O	O
555	Kqi	-	-	-	O	O	O
556	Kdp	-	-	-	O	O	O
557	Kdi	-	-	-	O	O	O
731	Digital Outputs	-	-	O	O	O	O
734	Analog Output 1	-	-	O	O	O	O
735	Analog Output 2	-	-	O	O	O	O

7.6.2.9 Testing state

When any one of the Run Test request services is sent to the motion axis while in the Stopped state, i.e. services that require an active power structure to execute, the axis immediately transitions to the Starting State (Power Structure Enabled status bit set), and then once the Starting conditions are met, the axis transitions to the Testing state. This Testing state is like the Running state in that the device's power structure is active (Power Structure Enabled status bit set), but in the Testing state one of the device's built-in test algorithms is controlling the motor, not command data from a motion planner (Tracking Command status bit clear). In the Testing state, the device excites the motor in various ways while performing measurements to determine characteristics of the motor and load. The motion axis remains in this state for the duration of the requested test procedure and then returns to the Stopped state. The motion axis can also exit the Testing state by either a fault or an explicit Axis Control request.

7.6.2.10 Start Inhibited state

The Start Inhibited state is the same as the Stopped state with the exception that the axis has one or more “start inhibit” conditions that prevent it from successfully transitioning to the Starting state. These conditions can be found in the Start Inhibit attributes. Once corrected, the axis state automatically transitions back to the Stopped state.

For an axis with no active control function (Control Mode = No Control) the Start Inhibited axis state prevents transitioning to the Running state until specific Start Inhibit conditions are resolved, such as when the associated device is not fully configured for operation. Again, once these conditions are corrected, the axis state automatically transitions to the Running state.

The Start Inhibited state is classified as an Identity Object Standby state and, therefore, requires that the associated power structure, if applicable, is disabled.

7.6.2.11 Stopping state

When a Disable request is issued to the Motion Device Axis Object in the Starting, Running, or Testing states, the axis immediately transitions to the Stopping state. In this state, the axis is in the process of stopping and is no longer tracking command data from the motion planner (Tracking Command status bit clear). There are a number of different Stopping Actions supported by the Motion Device Axis Object. Most of these Stopping Actions actively decelerate the axis to a stop. The power structure may remain active (Power Structure Enabled status bit set) as long as the Stopping Action procedure takes to complete. Once the selected Stopping Action procedure has completed, the axis transitions to the Stopped state.

When the Stopping Action is Disable and Coast, however, the power structure is immediately disabled (Power Structure Enabled status bit clear) and the axis coasts to a stop while in the Stopping state. In any case, the drive device shall wait until the axis has reached zero speed before transitioning to the Stopped state. In some cases, such as when the axis is stationary, this transition can be immediate. Recommended criteria for zero speed is based on Velocity Feedback, or in the case of Frequency Control drive device, this implies Velocity Reference being less than 1 % of motor rated speed. Ultimately this criteria is left to the vendors discretion.

When an Enable Request is given to an axis in the Stopping state with Flying Start Enabled, the axis shall immediately transition to the Starting state.

7.6.2.12 Aborting state

When a Major Fault occurs in the motion device while the axis is in the Starting, Running, Testing, or Stopping states, the axis immediately transitions to the Aborting state. In this state, the axis is in the process of stopping and is no longer tracking command data from the motion planner (Tracking Command status bit clear). The Aborting state executes the appropriate stopping action as specified by the device vendor. When actively stopping the axis in the Aborting state, the power structure remains active (Power Structure Enabled status bit set) as long as the stopping action takes to complete. In some cases the power structure shall be immediately disabled so the axis may coast to a stop while in the Aborting state. In any case, the drive shall wait until the axis has reached zero speed before transitioning to the Major Faulted state. Once the stopping procedure is complete and the axis has reached zero speed, the axis transitions to the Major Faulted state. In some cases, such as when the axis is stationary, this transition can be immediate.

When an Abort Request is issued to the Motion Device Axis Object a Controller Initiated Exception is generated. If the associated Axis Exception Action is set to generate a Major Fault the drive stops the axis according to the configured Stopping Action before transitioning to the Major Faulted state. Recommended criteria for zero speed is based on Velocity Feedback, or in the case of Frequency Control drive, this implies Velocity Reference being less than 1 % of motor rated speed. Ultimately this criteria is left to vendor discretion and can be application specific.

7.6.2.13 Major Faulted state

The Major Faulted state is identical to the Stopped state (or, if a Shutdown fault action was initiated, the Shutdown state) with the exception that there are one or more Major Faults active. In other words, a Major Faulted axis is a Stopped (or Shutdown) axis with a Major Fault condition present. Since faults are latched conditions, a Fault Reset request from the controller is required to clear the fault and, assuming the original fault condition has been removed, the axis transitions to the Stopped (or Shutdown) state.

There are four different sources of Major Faults: Node Faults, Initialization Faults, Axis Faults and Axis Safety Faults.

Initialization Faults

This kind of faults can only occur during the Initializing state. You cannot generate an Initialization fault in any other state of the device, i.e. faults occurring during operation of the device after transitioning out of the Initializing state. Initialization Faults can apply to a specific axis or to the entire device, in which case all axis instances would indicate the Initialization Fault. The device power structure, if applicable, is disabled when there is an Initialization Fault present.

Node Faults

These Faults always apply to the entire device and affect all axes the same. These faults can occur at any time during device operation. Node Faults are primarily communication faults, but can include general hardware faults where these fault conditions are checked during run-time. A CPU watchdog fault would be an example of a Hardware Node Fault. The device power structure, if applicable, is disabled when there is a Node Fault present.

Axis Faults

As the name implies, Axis Faults apply to a specific axis instances. Axis Faults are the direct result of Axis Exceptions that are configured to generate a Fault response. These exception conditions may apply to individual axis instances or to all axis instances. In any case, applications may require the device be configured to handle these exceptions differently for different axes. Run time conditions related to Motor, Inverter, Converter, Bus Regulator, and Feedback components, in general, shall be handled as Axis Exceptions. The device power structure, if applicable, may or may not be disabled when there is an Axis Fault present depending on the specific stopping action applied by the device in response to the fault condition.

Axis Safety Faults

Axis Safety Faults also apply to specific axis instances. Safety Faults are reported by the embedded Safety Core of the device that is responsible for monitoring the condition of various critical safety functions associated with the axis.

NOTE See [30] for more information.

Faults that occur after the device's axis state has transitioned out of the Initializing state shall be defined as either Node Faults, Axis Faults or Axis Safety Faults.

7.6.2.14 Shutdown state

When a Shutdown request is executed by the device, the targeted axis transitions to the Shutdown state. In the case of a Shutdown request, the axis immediately transitions from whatever state it is currently in to the Shutdown state. The Shutdown state has the same basic characteristics as the Stopped state except that the device's inverter power structure shall be disabled and free of torque (Power Structure Enabled status bit clear), and the Shutdown Action attribute can be configured to drop the DC Bus power to the device's power structure (DC Bus Up status bit clear).

NOTE This is generally done by opening an AC Contactor Enable, if applicable, output provided by the device that controls power to the converter.

Regardless of whether or not DC Bus power is disconnected, this state requires an explicit Shutdown Reset request from the controller to transition to the Pre-Charge state. If the device is configured to keep the DC Bus power active while in the Shutdown state then the motion axis transitions through the Pre-Charge state to the Stopped state. The Shutdown state offers an extra level of safety against unexpected motion.

In the case where a Shutdown fault action is initiated by the device in response to an exception condition that is configured to be a Major Fault, the device executes the Shutdown action, but the axis goes to the Major Faulted state, not the Shutdown state. Similarly, when the axis is in the Shutdown state and a major fault condition occurs, the axis transitions to the Major Faulted state. In other words, the major fault condition has precedence over the shutdown condition and the shutdown condition can be considered a sub-state. In either of these cases a Fault Reset request from the controller clears the fault and, assuming the original fault condition has been removed, the axis then transitions to the Shutdown state. A Shutdown Reset request from the controller, however, both clears the fault and performs a shutdown reset so, assuming the original fault condition has been removed, the axis transitions to the Pre-Charge state as described above.

In addition to the Shutdown action functionality, the Shutdown state can also be used by the controller to disable any slave gearing or camming motion planner functions that reference this device axis as a master axis. For this reason, the Shutdown state is applicable to a Feedback Only where the axis instance is simply associated with a feedback device that has no active control function.

The Shutdown state is classified as an Identity Object Standby state and, therefore, requires that the associated drive power structure is disabled.

7.6.3 Fault and alarm behavior

7.6.3.1 General

The Motion Device Axis Object's Fault and Alarm handling functionality addresses both the need for a large and ever-expanding number of specific faults and alarms, the need for programmable actions, and the need for timely reporting of those faults and alarms to the controller. Additionally, no compromises are made to restrict the resolution of the reported faults and alarms, so that the controller always has access to the unique axis condition and a meaningful diagnosis. Numerous Fault and Alarm related attributes can be included in the fixed portion of the cyclic Device-to-Controller Connection so the controller can monitor the condition of the motion axis in real-time, without cumbersome polling.

The Axis Status attribute contains bits to indicate whether an alarm condition is present. The Axis State enumeration indicates when the axis has a major fault, which could be a regular runtime Axis Fault, or an Initialization Fault. The Axis Fault Code and related attributes are provided to report the specific fault condition, time stamp, and fault action to the controller for the purposes of building a fault log. But before going into detail on this, the terms used to describe the Fault and Alarm functionality of the Motion Device Axis Object need to be carefully defined.

7.6.3.2 Exceptions

Exceptions are runtime conditions that the device continually checks that might indicate improper behavior of the motion axis or operation outside of an allowable range. An exception can result in an alarm, a minor fault, or a major fault, depending on how the associated Axis Exception Action has been configured – an exception can even be configured to be ignored. Exceptions are automatically cleared by the device when the underlying exception condition is no longer present.

7.6.3.3 Exception Actions

For each exception, the motion axis can be programmed a variety of actions via the Axis Exception Action attribute. Exception Actions range from generating a major fault that results in the stopping of the motion axis all the way to taking no action at all. The Axis Faults attribute allows the controller to have immediate access to any exceptions that have been configured to generate a major or minor fault. The Axis Alarms attribute allows the controller to have immediate access to any exceptions that have been configured to be reported as alarms.

7.6.3.4 Alarms

Alarms are runtime exception conditions for which the device is to take no action other than to report as an alarm. Alarms and warnings, therefore, are basically synonymous. On a given device product, some exception conditions may not be able to simply be reported as an alarm without any associated action; for example an IPM fault in which the power module automatically shuts off without software intervention. Alarm conditions are automatically cleared when the underlying exception condition is no longer present.

7.6.3.5 Major Faults

Major Faults can be initialization faults, safety faults or runtime exception conditions that the device has been configured to regard as a major fault. If a runtime fault occurs during an operational state, for example Running or Testing, it results in the device stopping (or aborting) all axis motion. Major Faults ultimately transition the axis state to the Major Faulted state. A Major Fault that results from an exception condition is latched, and does not clear when the exception condition clears. A fault can only be cleared with a Fault Reset service request from the controller. If the fault condition is classified as an “unrecoverable fault”, only a power cycle or a device reset can clear the fault condition.

7.6.3.6 Minor Faults

Minor Faults are exception conditions that the device has been configured to report to the controller as a fault but not take any direct action. This provides the controller an opportunity to perform an application specific fault action that may not be supported by the device as one of the defined exception actions. Like alarms, Minor Faults do not initiate a state change nor does a Minor Faulted state even exist; a Minor Fault allows the motion axis to continue operation in the state that it is presently in. But unlike alarms, a Minor Fault is latched, i.e. the fault does not clear when the exception condition clears. Both Major and Minor Faults can only be cleared with a Fault Reset service request from the controller.

7.6.3.7 Initialization Faults

Initialization Faults are faults that are generated during the power-up or device reset procedure when the device detects a problem that prevents normal device operation. This could be a hardware or firmware problem detected as part of its self-diagnostic tests or a problem with the attribute configuration process. These faults are not sourced by exception conditions and, therefore, they do not have configurable actions. Examples of initialization faults are corrupted memory data, calibration errors, or firmware startup problems. Initialization Faults that result in the Faulted state are considered “unrecoverable faults” and cannot be cleared with a Fault Reset service request, so any kind of motion is impossible in this state; only a power-cycle or a Device Reset has a chance of clearing this kind of fault.

7.6.3.8 Safety Faults

Safety Faults are faults that are reported by the drive’s built-in Safety Core that is an integral part of a safety system that includes this device, a Safety Controller, and a CIP Safety network connection. When the safety system detects a problem that prevents normal safe operation of the drive it generates a safety fault. These faults are not sourced by exception conditions and, therefore, they do not have configurable actions. Like run-time faults, Safety Faults that result in the Faulted state are considered “recoverable faults” and can be cleared with a Fault Reset request.

EXAMPLE Safety feedback faults, safe stop faults, or safe speed faults.

7.6.3.9 Node Faults

Node Faults are faults that are generated by the drive’s communications interface. Such faults are not axis specific but apply to the Motion Device Axis Object class. If a Node Fault occurs during an operational state, for example Running or Testing, it results in the device stopping (or aborting) all controlled axes associated with the device. Node Faults ultimately transition the axis state to the Major Faulted state. A Major Fault that results from a Node Fault is latched. A fault can only be cleared with a Node Fault Reset service request from the controller. If the node fault condition is classified as an “unrecoverable fault”, only a power cycle or a device reset can clear the fault condition.

7.6.3.10 Fault codes

The Fault Code attribute is an enumeration that indicates the runtime exception condition, initialization condition or safety condition that generated the fault. The source of the condition is specified by the Fault Type attribute. Like the fault status bits, this attribute value is latched and does not change unless, 1) the fault is “recoverable” and a Fault Reset request is initiated, or 2) another fault condition occurs. The Fault Code value corresponds to the bit position of the associated fault attribute. A value of 0 indicates that no fault condition currently exists for any of the potential fault sources. Fault Codes are primarily used for Visualization purposes and to build a Fault Log in the associated controller.

7.6.4 Start Inhibit behavior

A Start Inhibit is a condition that inhibits the axis from starting, i.e. transitioning to the Starting state for enabled axis operation. This condition does not generate an exception if a start attempt is made. If the circumstances that led to the Start Inhibit are no longer present, the start inhibit condition is automatically cleared by the device, returning the axis to the Stopped State.

If the motion axis is in the Start Inhibit state, it indicates that one or more conditions are present that prevent the axis from transitioning to enabled operation. The Start Inhibits attribute reports the specific condition that is inhibiting the axis.

7.6.5 Visualization behavior

7.6.5.1 General

Motion Device Axis Object state behavior has a direct impact on motion device visualization components. These components range from bicolor LED, or LED equivalent indicators to multi-character alphanumeric displays. This section defines how the Motion Device Axis Object states affect the behavior of these visualization components.

7.6.5.2 Module Status LED

Motion Device Axis Object states have a relationship to the state behavior of the Identity Object of the CIP Motion device and to its associated Module Status LED. Table 144 maps the states of the primary Motion Device Axis Object instance to the appropriate states of the Identity Object. CIP Motion compliant devices are required to support the Module Status LED.

For more information regarding the Identity Object state model, refer to the Identity Object specification (see IEC 61158–5-2).

7.6.5.3 Axis Status LED

To further augment the visual information provided by the standard Module Status LED, the CIP Motion device profile also defines the behavior of a second LED. This so-called Axis Status LED provides visual indication of, for example, whether or not the DC Bus is energized, whether the axis is enabled or disabled, and even provides indication of active alarms or minor fault conditions (see Table 144). This LED is also required by CIP Motion compliant devices unless the device is equipped with a multi-character alphanumeric display.

The Axis Status LED uses three colors that can be generated by a standard bicolor Red/Green LED, namely Red, Green, and Amber. Amber (or Yellow) is the color produced when both the Red and Green junctions of the bicolor LED are on. The general meaning of these three colors are as follows:

green – indicates a normal power-up or operational state;

amber – indicates the presence of an alarm or start inhibiting condition;

red – indicates presence of some form of fault condition.

The normal power up or device reset, both the Module Status and Axis Status LEDs shall start in the Red state (under hardware control) while the device processor is booting and then switch to the Green state for approximately 1 s once the device begins executing its self test. Ideally, both LEDs shall be Red for approximately 1 s and then Green for approximately 1 s prior to transitioning to a Standby state indication. This Red-Green sequence confirms proper operation of the LED indicators.

Table 144 – Axis state mapping to Identity Object with LED behavior

Identity Object state	Module Status LED	Motion Device Axis Object state	Axis Status LED
Nonexistent – Power Off	Off	Off	Off
Device Self-Testing	Flash Red/Green	Self Test	Flash Red/Green
Standby	Flashing Green	Initialization – Bus not Up	Off
		Initialization – Bus Up	Flashing Green
		Shutdown – Bus not Up	Off
		Shutdown – Bus Up	Flashing Amber [a]
		Pre-Charge – Bus not Up	Off
		Start Inhibit	Flashing Amber [a]
Operational	Solid Green	Stopped	Flashing Green[a b]
		Stopping	Solid Green[a b]
		Starting	Solid Green[a b]
		Running	Solid Green[a b]
		Testing	Solid Green[a b]
Major Recoverable Fault	Flashing Red	Aborting	Flashing Red
		Major Faulted	Flashing Red
Major Unrecoverable Fault	Solid Red	Aborting	Solid Red
		Major Faulted	Solid Red

[a] The Motion Device Axis Object and the Identity Object define minor fault conditions. While a minor fault does not affect the Module Status LED, it does affect the Axis Status LED. When a minor fault condition is detected, a normally Solid Green LED indication changes to alternating Red-Green-Red-Green, a normally Flashing Green LED indication changes to alternating Red-Off-Green-Off, and a normally Flashing Amber indications changes to Red-Off-Amber-Off.

[b] The Motion Device Axis Object also defines alarm conditions. When an alarm condition is detected, a normally Solid Green LED indication changes to alternating Amber-Green-Amber-Green, while a normally Flashing Green LED indication changes to alternating Amber-Off-Green-Off.

7.6.5.4 Alphanumeric display

7.6.5.4.1 General

In addition to the required LED visualization provided by the Module Status LED and Axis Status LED, the Motion Device Axis Object also defines the behavior of an optional alphanumeric display to more explicitly indicate the condition of the device. Such a display is particularly useful for monitoring progress through the initialization process and providing detailed diagnostic information concerning fault, alarm, and inhibit conditions. Alphanumeric displays can range from simple seven-segment displays to multi-character alphanumeric displays.

7.6.5.4.2 Seven-segment display

If the device is equipped with a seven-segment display, the display can be used to indicate progress through the Initializing state and various fault, alarm, and inhibit conditions. As a minimum, the display shall support the mapping to various conditions of the device specified in Table 145.

Table 145 – CIP Motion Device seven-segment display behavior

Display digit	Device condition
8	Executing device Self-Test
0	Waiting for connection to controller
1	Configuring device attributes
2	Waiting for group synchronization
3	Waiting for DC Bus to charge
4	Device is operational
I##	Initialization Fault Code
Ic##	Initialization Fault Code – Manufacturer's custom extensions
F##	Axis Fault Code
Fc##	Axis Fault Code – Manufacturer's custom extensions
A##	Alarm Code
Ac##	Axis Alarm Code – Manufacturer's custom extensions
S##	Start Inhibit Code
Sc##	Start Inhibit Code – Manufacturer's custom extensions
SF##	Safety Fault Code
SFc##	Safety Fault Code – Manufacturer's custom extensions
nF##	Node Fault Code
nA##	Node Alarm Code

7.6.5.4.3 Multi-character alphanumeric display

If the device is equipped with a multi-character alphanumeric display, even more useful device information can be conveyed to the user via scrolling or static character fields. The capabilities of the display generally dictate the length of the character strings that can be effectively displayed. Nevertheless, for the purpose of enforcing consistent behavior among CIP Motion compliant device products, the Motion Device Axis Object dictates exactly what is displayed for the conditions outlined in Table 146.

Table 146 – CIP Motion multi-character alphanumeric display behavior

Display string	Device condition
SELF-TEST	Executing device Self-Test
STANDBY	Waiting for Forward_Open Service
CONNECTING	Waiting for 1st Set Attribute Service. (Refreshing)
CONFIGURING	Configuring Class & Axis Instance Attributes
SYNCING	Waiting for Successful Group_Sync Service.
PRE-CHARGE	Waiting for DC Bus Up
SHUTDOWN	Axis has been Shutdown
STOPPED	Axis has stopped
STARTING	Axis is Starting
RUNNING	Axis is Running
TESTING	Axis is executing a Test procedure
STOPPING	Decelerating to a stop as a result of a disable
ABORTING	Decelerating to a stop as a result of a fault
INIT FLT S##	Initialization Fault – Std Fault Code
INIT FLT M##	Initialization Fault – Mfg Fault Code
FLT S##	Axis Fault – Std. Fault Code
FLT M##	Axis Fault – Mfg Fault Code
ALARM S##	Axis Alarm – Std Alarm Code
ALARM M##	Axis Alarm – Mfg Alarm Code
INHIBIT S##	Start Inhibit – Std Inhibit Code
INHIBIT M##	Start Inhibit – Mfg Inhibit Code
SAFE FLT S##	Axis Safety Fault – Std Safety Fault Code
SAFE FLT M##	Axis Safety Fault – Mfg Safety Fault Code
NODE FLT ##	Node Fault Code
NODE ALARM ##	Node Alarm Code

In addition to this basic functionality, the alphanumeric display can also provide detailed text descriptions of fault, alarm, and inhibit conditions. Definition of these specific strings is well beyond the scope of this object and is left to the discretion of the device vendor.

7.6.5.4.4 Multi-axis device visualization

In the case where there are multiple motion axis instances supported by the device, the above behavior needs further explanation.

First of all, there is only one Module Status LED per device node, so its condition is a roll-up of the states of all the Motion Device Axis Object instances. By contrast, one Axis Status LED is associated with each Motion Device Axis Object instance in the device that has a power structure. A single multi-character alphanumeric display can easily manage multiple motion axis instances.

In the case of a motion axis instance configured with no active control function (Control Mode = No Control), there is very little state information to visualize; the motion axis is in the operational state called Running and remains in that state unless there is a fault that transitions the axis to Major Faulted. A No Control axis, therefore, has no effect on the Module Status LED unless a fault occurs, in which case the Module Status shows Flashing

Red (Major Recoverable Fault). A No Control axis does not require a separate Axis Status LED.

In the case of multiple motion axis instances, each with a separate power structure, the behavior of the Module Status LED is a roll-up of the states of the device axes. When the device axes are in disparate states, the Module Status LED condition is based on the following precedence:

1) Major Unrecoverable Fault;
2) Major Recoverable Fault;
3) Standby;
4) Operational.

In other words, as far as the Module Status LED is concerned, a Major Recoverable Fault on Axis 1 trumps Axis 2 that is in the Standby state, Start Inhibit.

NOTE Minor faults and alarms are not recognized by the Module Status LED.

Multi-character displays easily handle multiple axis instances by adding a “”X#” prefix to the display string to specify the associated motion axis instance number. For Master Feedback axis instances, no specific display string is shown unless the Master Axis has a fault or alarm condition.

Table 147 specifies the mapping between the display and the device condition in the case of multiple motion device axis instances.

Table 147 – Multi-axis multi-character alphanumeric display behavior

Display string [a]	Device condition
SELF-TEST	Executing device Self-Test
STANDBY	Waiting for Forward_Open Service
CONNECTING	Waiting for 1st Set Attribute Service. (Refreshing)
CONFIGURING	Configuring Class & Axis Instance Attributes
SYNCING	Waiting for Successful Group_Sync Service.
PRE-CHARGE	Waiting for DC Bus Up
X#:SHUTDOWN	Axis has been Shutdown
X#:STOPPED	Axis has stopped
X#:INHIBITED	Axis is Start Inhibited
X#:STARTING	Axis is Starting
X#:RUNNING	Axis is Running
X#:TESTING	Axis is executing a Test procedure
X#:STOPPING	Decelerating to a stop as a result of a disable
X#:ABORTING	Decelerating to a stop as a result of a fault
X#:INIT FLT S##	Initialization Fault – Std Fault Code
X#:INIT FLT M##	Initialization Fault – Mfg Fault Code
X#:FLT S##	Axis Fault – Std. Fault Code
X#:FLT M##	Axis Fault – Mfg Fault Code
X#:ALARM S##	Axis Alarm – Std Alarm Code
X#:ALARM M##	Axis Alarm – Mfg Alarm Code
X#:INHIBIT S##	Start Inhibit– Std Inhibit Code
X#:INHIBIT M##	Start Inhibit– Mfg Inhibit Code
X#:SAFE FLT S##	Axis Safety Fault – Std Inhibit Code
X#:SAFE FLT M##	Axis Safety Fault – Mfg Inhibit Code
NODE FLT ##	Node Fault Code
NODE ALARM ##	Node Alarm Code

[a] X# = Axis Instance Number (0, 1, 2, 3 …) → 0 = class instance

7.6.6 Command generation behavior

7.6.6.1 Command data sources

Command data that affects axis motion can come from a variety of sources. The most common command data source is from a controller-based motion planner via the CIP Motion C-to-D Connection. In this context, command data can take the form of Controller Position, Velocity, Acceleration, and Torque Commands generated by the motion planner (see Figure 90). The command data elements provided are specified by the Command Data Set attribute, which is based on the selected Control Mode. The primary command data element can be augmented by higher order command elements for the purposes of generating high quality feed-forward signals. Alternatively, these higher order command elements can be derived by the device from the primary command data. In either case, a Fine Command Interpolator is generally applied to the Command Data to generate command reference signals to the devices' control structure at the devices' update rate.

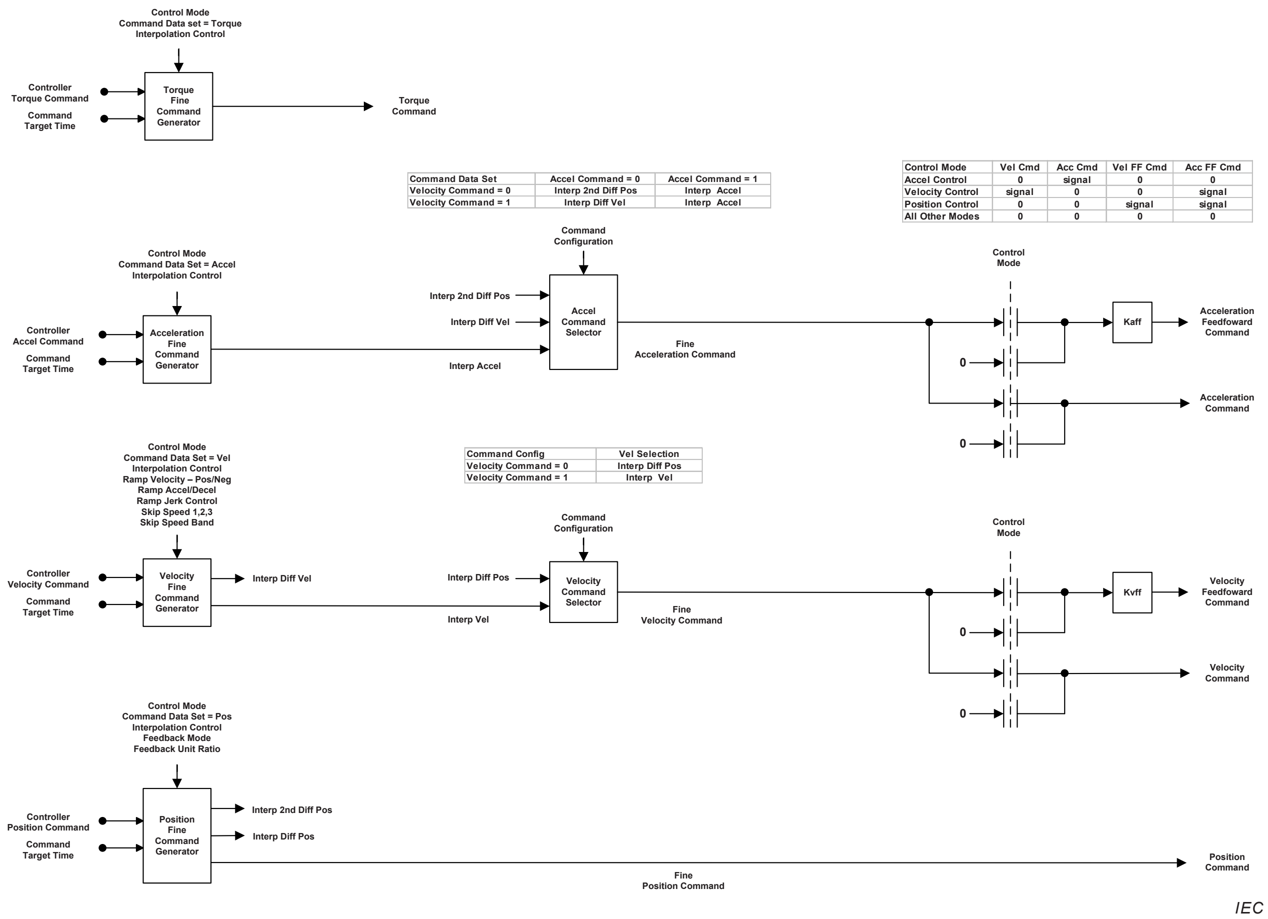

Figure 90 – Command Generator

7.6.6.2 Command fine interpolation

For synchronized, high-performance applications using CIP Motion, command data is received from the CIP Motion C-to-D Connection and based on the connection's Command Target Update element being set to "Interpolate", processed by the Fine Interpolator functionality of the Command Generator blocks. The job of the Fine Interpolator is to compute coefficients to a trajectory polynomial that is designed to reach the command data at its associated Command Target Time. Depending on the specific command data element, the trajectory can follow a 1st, 2nd, or 3rd order polynomial trajectory with initial conditions based on current axis dynamics. Since the polynomial is a function of time, a new fine command value can be calculated any time the CIP Motion device needs to perform a control calculation. As a result, it is not necessary that the device's control calculation period be integrally divisible into the Controller Update Period.

To improve device interchangeability, the Motion Device Axis Object recommends a minimum order for the fine interpolators. Since contemporary motion planners typically generate their trajectories based on 3rd order polynomials in position, it is important that the fine interpolators reproduce these trajectories with high fidelity. Therefore, the position fine interpolator is defined as 3rd order, the velocity interpolator is 2nd order, and the acceleration and torque interpolators are both 1st order. Higher order fine interpolators are possible and are left to the vendor's discretion.

Position Fine Interpolation Polynomial:

$$P(t) = a_0 + a_1 \times (t - t_0) + a_2 \times (t - t_0)^2 + a_3 \times (t - t_0)^3$$

Velocity Fine Interpolation Polynomial:

$V(t) = b_0 + b_1 \times (t - t_0) + b_2 \times (t - t_0)^2$

Acceleration Fine Interpolation Polynomial:

$A(t) = c_0 + c_1 \times (t - t_0)$

Torque Fine Interpolation Polynomial:

$T(t) = d_0 + d_1 \times (t - t_0)$

In these equations, time t_0 represents the Command Target Time for the previous motion planner update such that when $t = t_0$ the position, velocity, acceleration, and torque command values are equal to the values sent in the previous motion planner update, i.e. P_{-1}, V_{-1}, A_{-1}, and T_{-1}. This establishes the 0^{th} order coefficients of the polynomials.

$P(t_0) = P_{-1} = a_0$

$V(t_0) = V_{-1} = b_0$

$A(t_0) = A_{-1} = c_0$

$T(t_0) = T_{-1} = d_0$

The higher order polynomial coefficients are calculated such that by the next motion planner update, corresponding to Command Target Time, t_1, the position, velocity, acceleration, and torque command values are the values sent in the latest motion planner update, i.e. P_0, V_0, A_0, and T_0.

$P(t_1) = P_0$

$V(t_1) = V_0$

$A(t_1) = A_0$

$T(t_1) = T_0$

Using the above polynomial interpolation equations, the CIP Motion device can compute position, velocity, acceleration, and torque command values at any time by plugging in the current System Time value of the device into the variable, t. This allows the device's control calculation to be performed according to a schedule that is independent of the controller's update schedule.

One thing that shall be done, however, is to adjust the Command Target Time, t_0, should there be a shift in the System Time Offset for the device; t_0 and t shall always be based on the same System Time reference system. For example, assume the device's System Time Offset when the control command timestamp, t_0, was received is $Offset_0$. If the command interpolation equation is to be applied at $t = t_1$ and the current System Time Offset is defined as $Offset_1$ then t_0 shall be adjusted as follows before executing the polynomial:

Adjusted $t_0 = t_0 + (Offset_1 - Offset_0)$

Alternatively, the values for t, t_0 and t_1 can be based on local time rather than system time by using the current System Time Offset to convert between System Time to local time. This may be more convenient for the interpolator implementation and is left to the vendors discretion.

The polynomial coefficients are computed based on standard formulas that are a function of the history of command values over the last few updates. The number of historical command values used in the formula depends on the order of the polynomial. For example, the third order command position polynomial uses the three previous command position values. For convenience, the interpolator polynomial coefficient formulae are as follows:

Position Fine Interpolation Polynomial Coefficients:

$a_0 = P_{-1}$

$a_1 = 1/T \times (\Delta P_0 - 1/2 \times \Delta V_0 - 1/6 \times \Delta A_0)$

$a_2 = 1/T^2 \times (1/2 \times \Delta V_0)$

$a_3 = 1/T^3 \times (1/6 \times \Delta A_0)$

Velocity Fine Interpolation Polynomial Coefficients:

$b_0 = V_{-1}$

$b_1 = 1/T \times (\Delta V_0 - 1/2 \times \Delta A_0)$

$b_2 = 1/T^2 \times (1/2 \times \Delta A_0)$

Acceleration Fine Interpolation Polynomial Coefficients (Torque is same form as Accel):

$c_0 = A_{-1}$

$c_1 = 1/T \times \Delta A_0$

The above equations are based on the following nomenclature:

T = Controller Update Period

$\Delta P_0 = (P_0 - P_{-1})$

$\Delta V_0 = (V_0 - V_{-1}) = (P_0 - 2P_{-1} + P_{-2})$

$\Delta A_0 = (A_0 - A_{-1}) = (V_0 - 2V_{-1} + V_{-2}) = (P_0 - 3P_{-1} + 3P_{-2} - P_{-3})$

The above polynomial coefficients should be applied to the fine interpolator as soon possible after t is equal to or greater than t_0. Applying the new coefficients too early, i.e. with t significantly less than t_0, can create unnecessary error in the command trajectory when connecting the last fine interpolator segment to the new fine interpolator segment at t_0.

When $t > t_1$, the fine interpolation polynomial becomes an extrapolation polynomial. In the absence of a fresh update from the motion planner, the extrapolation polynomial can be used to provide estimated command data to the device control structure until fresh motion planner command data is available. Once fresh command data is made available, new polynomial coefficients shall be computed without delay. In this way, the motion control can be maintained ("ride-thru") despite occasional late or lost connection data packets resulting in a robust distributed motion control network solution. To be clear, late connection data is always applied and never thrown away; late data still represents the freshest data available from the controller and the extrapolation polynomial ensures that the command data is applied in such a way as to maintain a smooth motion trajectory despite variations in command data delivery.

When the update period of the motion planner is short enough relative to the dynamics of the command trajectory, or is comparable to the device control calculation period, fine interpolation may not be necessary. The motion planner can make this determination by comparing the planner update period to that of the device control calculation period. When fine interpolation is used, the planner shall add additional planner update periods to the planner time stamp, so it is advantageous to eliminate this planner update period delay if interpolation is not necessary.

Even though fine interpolation may not be necessary in some cases, it does not mean that the command data is to be applied directly to the device control structure. It still may be necessary to calculate the above polynomials so the device can extrapolate the command value when the device's control update occurs. That is because, in general, the device's control update time stamp does not need to match the time stamp of the command data.

Finally, there are applications and CIP Motion device types that do not require the dynamic accuracy that time-stamped interpolation and extrapolation provide. Various velocity and torque control applications, for example, may fall in this category. In general, command data can also be applied to the control structures of variable frequency drives without interpolation or extrapolation.

7.6.6.3 Command selectors

The Velocity and Acceleration Selectors are used to select the source of the Fine Velocity Command and Fine Acceleration Command signals. Selection is based on the Command Data Set value from the controller.

For example, if Velocity Command is provided by the controller, the Fine Velocity Command is sourced by the output of the Velocity Fine Command Generator. When configured for fine interpolation or extrapolation, the polynomial used to calculate the Velocity Fine Command is given by:

$$V(t) = b_0 + b_1 \times (t - t_0) + b_2 \times (t - t_0)^2$$

The polynomial coefficients are:

$$b_0 = V_{-1}$$

$$b_1 = 1/T \times (\Delta V_0 - 1/2 \times \Delta A_0)$$

$$b_2 = 1/T2 \times (1/2 \times \Delta A_0)$$

In this case, values for ΔV_0 and ΔA_0 are calculated in terms of Velocity Command values, V_0, V_{-1}, and V_{-2}:

$$\Delta V_0 = (V_0 - V_{-1})$$

$$\Delta A_0 = (V_0 - 2V_{-1} + V_{-2})$$

If the Velocity Command is not provided by the controller, then the Fine Velocity Command signal is the Interpolated Differential Position signal from the Position Fine Command Generator that is the time derivative of the Fine Position Command signal. When configured for fine interpolation or extrapolation, the polynomial used to calculate the Velocity Fine Command are the same as above, but the polynomial coefficients are calculated using ΔV_0 and ΔA_0 values derived from Position Command values, P_0, P_{-1}, P_{-2}, and P_{-3}:

$$\Delta V_0 = (P_0 - 2P_{-1} + P_{-2})$$

$$\Delta A_0 = (P_0 - 3P_{-1} + 3P_{-2} - P_{-3})$$

7.6.6.4 Command ramp generator

The Ramp Generator feature of the Command Generator block is applied to the Command Data value sent by the controller when the Command Target Update element of the connection is set to “Immediate” mode. In Immediate mode, the Command Data is applied immediately to the devices’ control structure. Since there is generally no motion planner generating the Command Data in this mode, the Command Data value from the controller can change drastically from one update to the next. To address this condition, a Ramp Generator function is needed to ramp the motor to the new Command Data value within the dynamic limitations of system. An example of if the Controller Velocity Command value suddenly changed from 0 to 30 revolutions per second in Immediate Mode, the Ramp Generator would produce a Fine Velocity Command signal that accelerates the motor to the Controller Velocity Command value based on the configured Ramp Acceleration and Jerk Control attribute values. The Ramp Jerk Control attribute determines what percentage of the acceleration or deceleration ramps is S-Curve with the remaining portion of the ramp governed by the fixed Ramp Acceleration or Deceleration attribute values.

While a Ramp Generator function could be included in each of the Fine Command Generator blocks for position, velocity, and acceleration commands, this version of the Motion Device Axis Object specification only supports a Ramp Generator in the Velocity Fine Command Generator block.

The Ramp Generator enforces directional velocity limits on the Command Data, insuring that the Velocity Command never exceeds the configured Maximum Velocity Pos/Neg values.

The Ramp Generator also supports Flying Start functionality. When enabling the drive while the motor is still moving, the Ramp Generator output is initialized to the current speed of the motor. From there, the Ramp Generator smoothly accelerates or decelerates the motor to the current Controller Velocity Command.

Finally, the Ramp Generator supports Skip Bands that are most frequently used in Frequency Control applications when certain speeds excite mechanical resonance frequencies of the motor and load. The Skip Band feature allows three separate Skip Speeds to be defined that shift the Velocity Command signal to avoid, or skip, these problematic speeds. The Skip Speed Band determines the range of speeds centred on the three Skip Speeds that the device avoids. If the Velocity Command falls is within the Skip Band but below the Skip Speed the Velocity Command output is set to the Skip Speed, minus ½ the Skip Speed Band. If the Fine Velocity Command falls is within the Skip Band but above the Skip Speed the Velocity Command output is set to the Skip Speed, plus ½ the Skip Speed Band.

7.6.6.5 Feed-Forward signal selection

The Fine Command Generators defined as part of the Motion Device Axis Object can generate higher derivatives of the command data input to serve as feedforward signals. The units for the velocity and acceleration feedforward signals are generally different than the derivative units, hence the derivative signals shall be scaled appropriately. Superior signal quality, however, can be provided by the motion planner trajectory generators. The feedforward selection blocks pick the best feed-forward signal to apply based on the bits set in the Command Data Set attribute. The best signal is defined as the signal derived using the fewest differencing operations. The fine command position is applied directly to the position control loop without any of the typical de-referencing and offsets. It is assumed that these operations are performed by the controller based motion planner.

7.6.7 Feedback interface behavior

7.6.7.1 Feedback sources

Feedback signals defined by the Motion Device Axis Object can be derived from any of 4 different feedback interface channels (see Figure 91). The two primary feedback channels employed by the various closed loop control modes are designated Feedback 1 and Feedback 2. This allows the control loops to operate with either a motor based feedback device that is typically attached to the Feedback 1 channel or a load-side feedback device that is connected to the Feedback 2 channel. Which feedback source is used by the loop is governed by the Feedback Mode attribute.

Each feedback interface is capable of supporting a number of different feedback device types as enumerated by the Feedback Type attribute. The feedback interface output is the number of feedback counts that the feedback device has moved since the last time the device was sampled. If the feedback device is an absolute device, the feedback interface also determines the absolute position of the feedback device at power-up and communicates that value to the Feedback Accumulator to preset the accumulator.

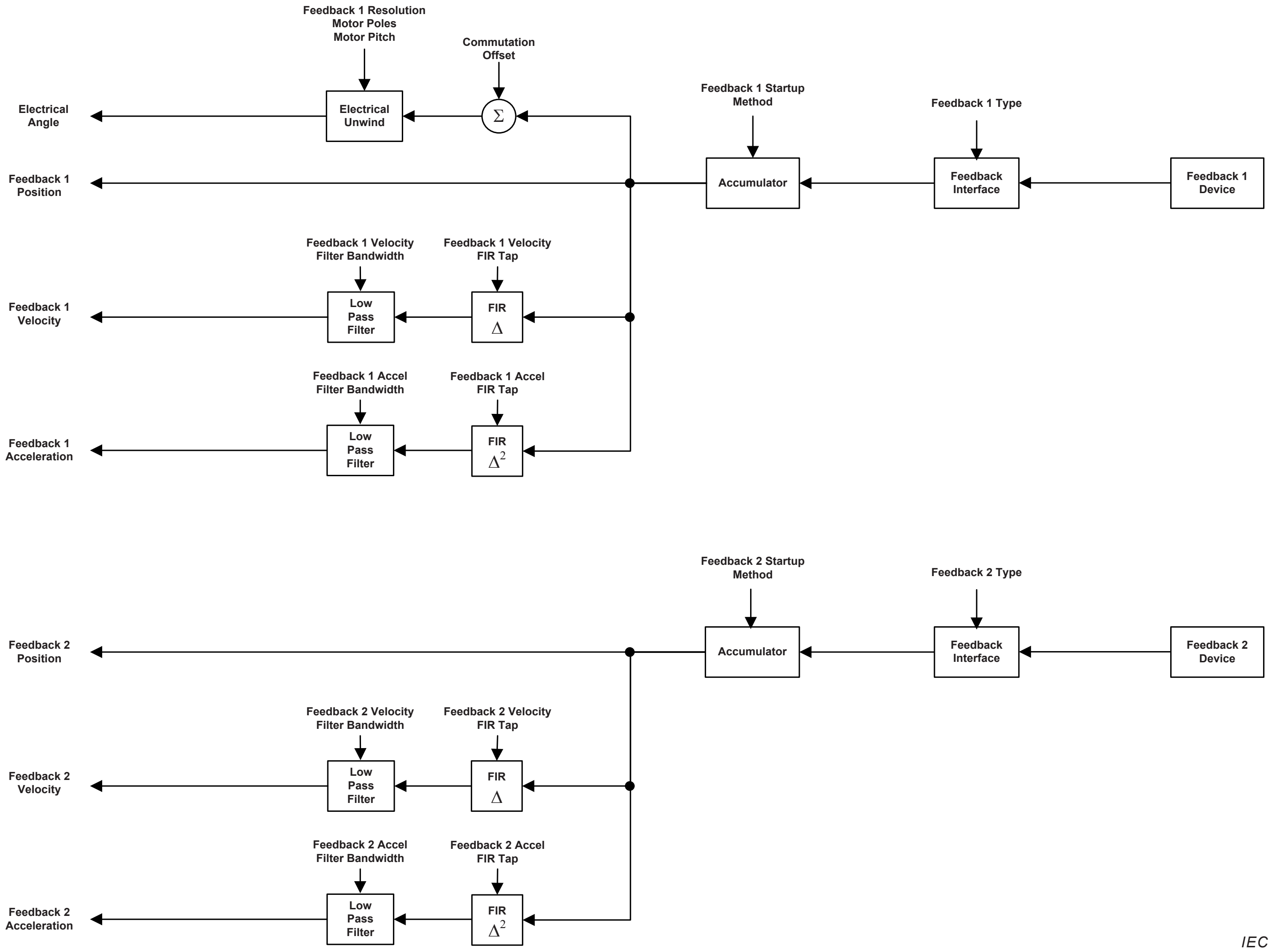

Figure 91 – Feedback Channels 1 and 2

7.6.7.2 Feedback accumulator

The role of the Feedback Accumulator depends on the configured Feedback n Startup Method which can be either Incremental or Absolute. If Incremental is selected, the accumulator simply accumulates changes to the feedback count value, a 32-bit signed integer, with every device update. If Absolute is selected, the Feedback Accumulator works basically the same way as in Incremental mode. The only difference is the initialization of the accumulator at device power-up. In Incremental mode, the Feedback Accumulator is set to zero, while in Absolute mode, the accumulator is initialized to the absolute position of the feedback device. This allows for the recovery of absolute position through a power-cycle as long as power-off movement of the absolute feedback device is limited to half of the absolute feedback range of the device. There is no device requirement to extend the absolute position range of the feedback device through non-volatile storage of the accumulator. This simple absolute feedback handling mechanism is due to the fact that the CIP Motion normally places the responsibility of extending the absolute position range of the axis, and establishing the absolute machine position reference, on the controller.

7.6.7.3 Commutation unwind and offset

An Electronic Unwind block is also connected to the Feedback 1 interface. This block is designed to unwind, or modulo, the position accumulator output to generate a signal that is proportional to the electrical angle of a Permanent Magnet motor based on the based on the Pole Count or Pole Pitch of the motor. To align this signal with the physical ABC windings of the motor rather than the zero of the feedback device, a configurable Commutation Offset is added prior to the Electrical Unwind block.

7.6.7.4 Feedback filtering

A configurable low-pass IIR filter is defined by the Motion Device Axis Object for filtering the velocity and acceleration estimates for each feedback channel. These filters can be used to reduce the level of quantization noise associated with differencing digital feedback signals. The bandwidth of the velocity and acceleration IIR filters for each feedback channel are individually programmable.

7.6.8 Event Capture Behavior

7.6.8.1 Event input sources

The Motion Device Axis Object defines a mechanism to capture both the feedback position and time stamp associated with specific state transitions of selected event input sources. Event input sources currently supported by the object are Registration 1, Registration 2, Marker, and Home Switch. These 4 event input sources apply to each supported feedback channel (see Figure 92).

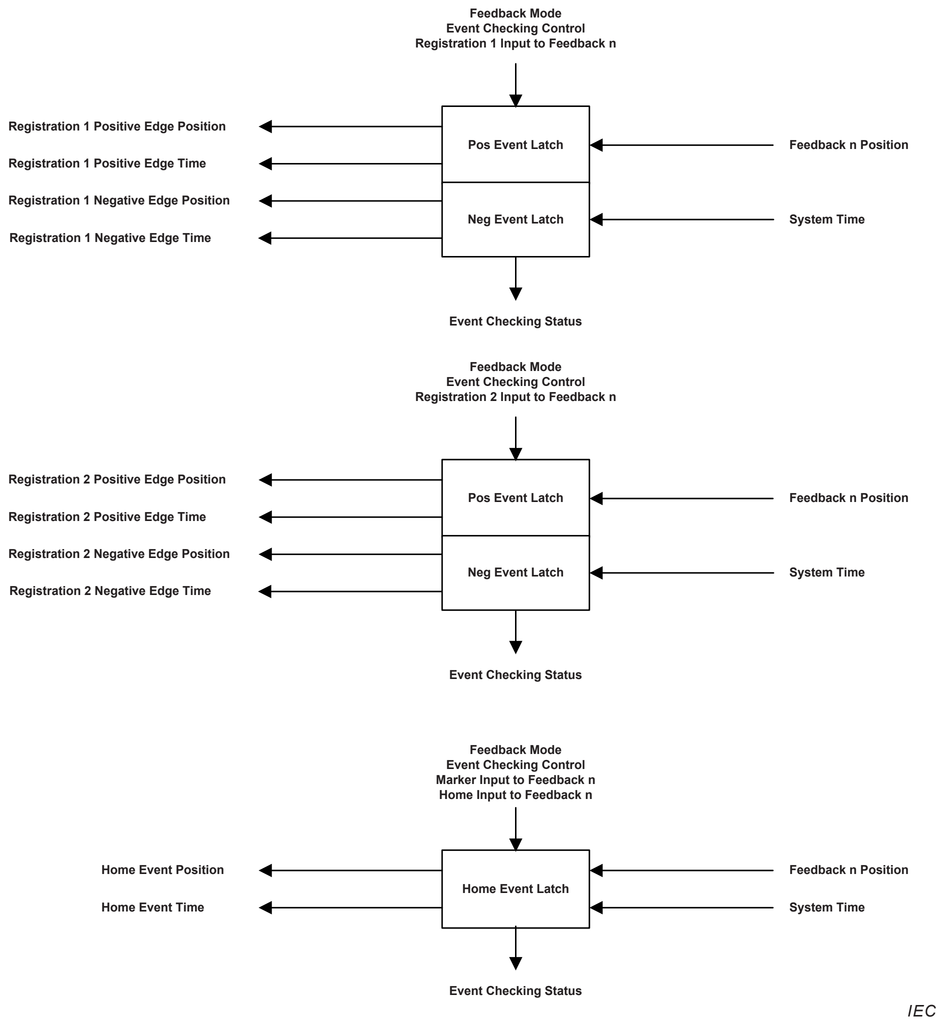

Figure 92 – Event Capture Functionality

7.6.8.2 Event latches

To facilitate accurate capture of both feedback position and time, hardware event latches are typically implemented as shown in the following block diagram. Two independent latches are defined for each registration input, one latch to capture positive edge transition events and one to capture negative edge transition events. This design enables capture of both registration events in applications with narrow registration pulses where the rising and falling edges occur nearly simultaneously. In addition to the registration latches, a separate latch is also defined for the home event capture. The home input event that triggers the Home Event Latch can be any of a number of different combinations of home switch and marker input events, i.e. marker transitions, switch transitions, or switch transitions followed by a marker transition.

With hardware based event latches, event capture accuracy is, in general, only limited by the latency of the associated event input. Registration and Marker event inputs are lightly filtered so event capture accuracy is in the order of 1 μs. In terms of position capture accuracy, that would be calculated as the product of the event capture accuracy and the speed of the axis. Home switch inputs are heavily filtered, in general, and therefore limited to an event capture accuracy of 1 ms to 10 ms. Thus, to get an accurate position capture based on a home switch input transition, a homing sequence with a slow homing speed is required.

7.6.8.3 Event time stamps

Since the registration time stamp is passed to the controller as part of the Event Notification data, the controller can apply the event time stamp to the position history of other axes in the system to interpolate their positions. This is particularly useful in applications where it is necessary to determine the location of several axes at the time of a single registration event. The more accurate the time stamp, the more accurately the controller can determine these positions.

One thing that shall be done, however, is to adjust the event time stamp, t_0, should there be a shift in the System Time Offset for the device prior to transmitting to the controller; the event time stamp shall always be based on the same System Time reference system at the time of transmission.For example, if $Offset_0$ is the device's System Time Offset when the event timestamp t_0 occurred, and t_1 is the System Time Offset at the time that the event is to be transmitted to the controller, then t_0 shall be adjusted to be t_1 prior to transmission with the rest of the associated event data to the controller, i.e.: $t_1 = t_0 + (Offset_1 – Offset_0)$.

7.6.9 Control Mode behavior

7.6.9.1 General

The instance attributes defined in 7.3 affect device behavior in the context of the Control Mode, Control Method, and Feedback Mode. As discussed in 4.2 concerning the scope of the Motion Device Axis Object, there are basically 4 Control Modes common to devices, position control, velocity control, torque control, and no control. A new control mode, acceleration control, is added to this list to complete the progression from velocity control to torque control. Subclause 7.6.9 provides a block diagram for each of the control modes in an effort to further define the collective behavior of the various Motion Device Axis Object attributes.

7.6.9.2 No Control (feedback only) mode

A Motion Device Axis Object instance can be configured for No Control mode. This Control Mode selection applies to several different Device Functions that include Bus Power Converters and Feedback Only devices. Feedback Only axis functionality allows the position, velocity, and acceleration of any one of four possible feedback channels to be accessed by the controller via the Device-to-Controller Connection. These signals can then be distributed across the motion control system as a master axis for gearing and camming operations. In this mode, the Feedback Master Select attribute, if supported, determines which feedback channel produces the Position, Velocity, and Acceleration Feedback signals (see Figure 93).

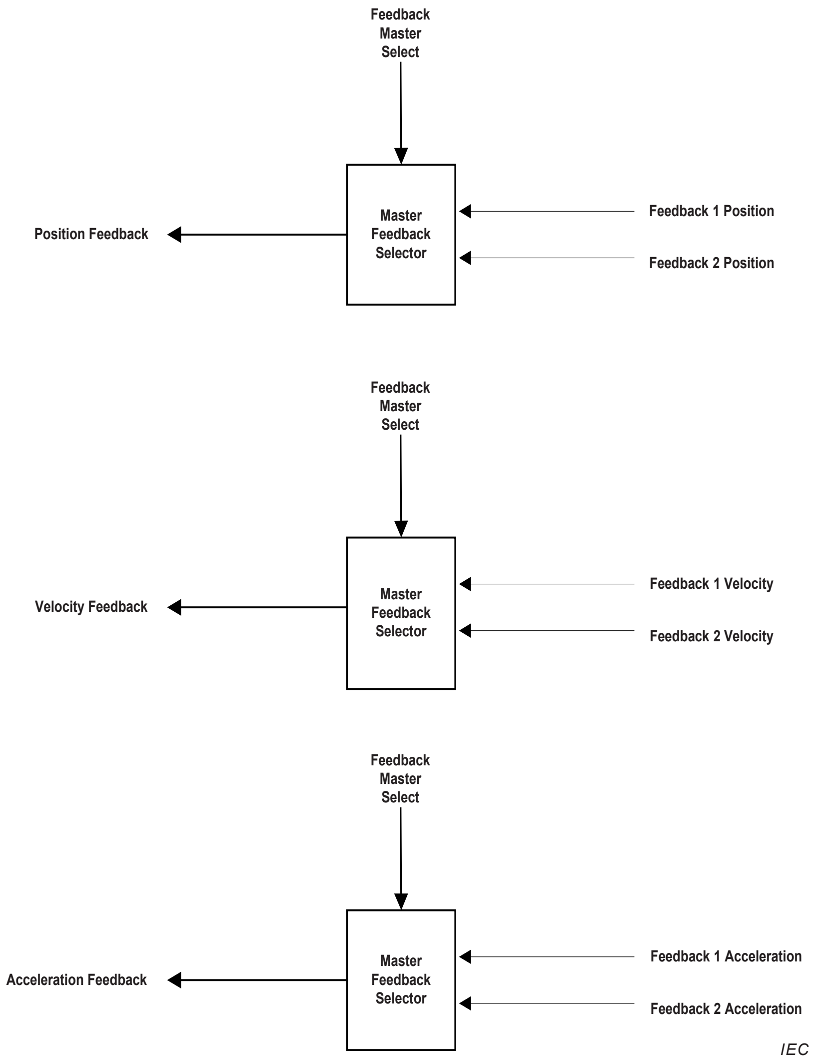

Figure 93 – No Control (Feedback Only)

7.6.9.3 Position Control

In Position Control mode, the only operative Control Method supported by the object currently is Closed Loop servo control. At a later date, the object could be expanded to include a Stepper based position Control Method.

7.6.9.4 Closed Loop Position Control

7.6.9.4.1 General

When performing closed loop position control, the device applies the Position Command signal output of the Command Generator to the position loop summing junction. In addition to the Position Command, a Position Trim input is provided which can be used to provide an offset to the position loop. The classic PI control loop generates a Position Loop Output signal to an inner velocity loop (see Figure 94).

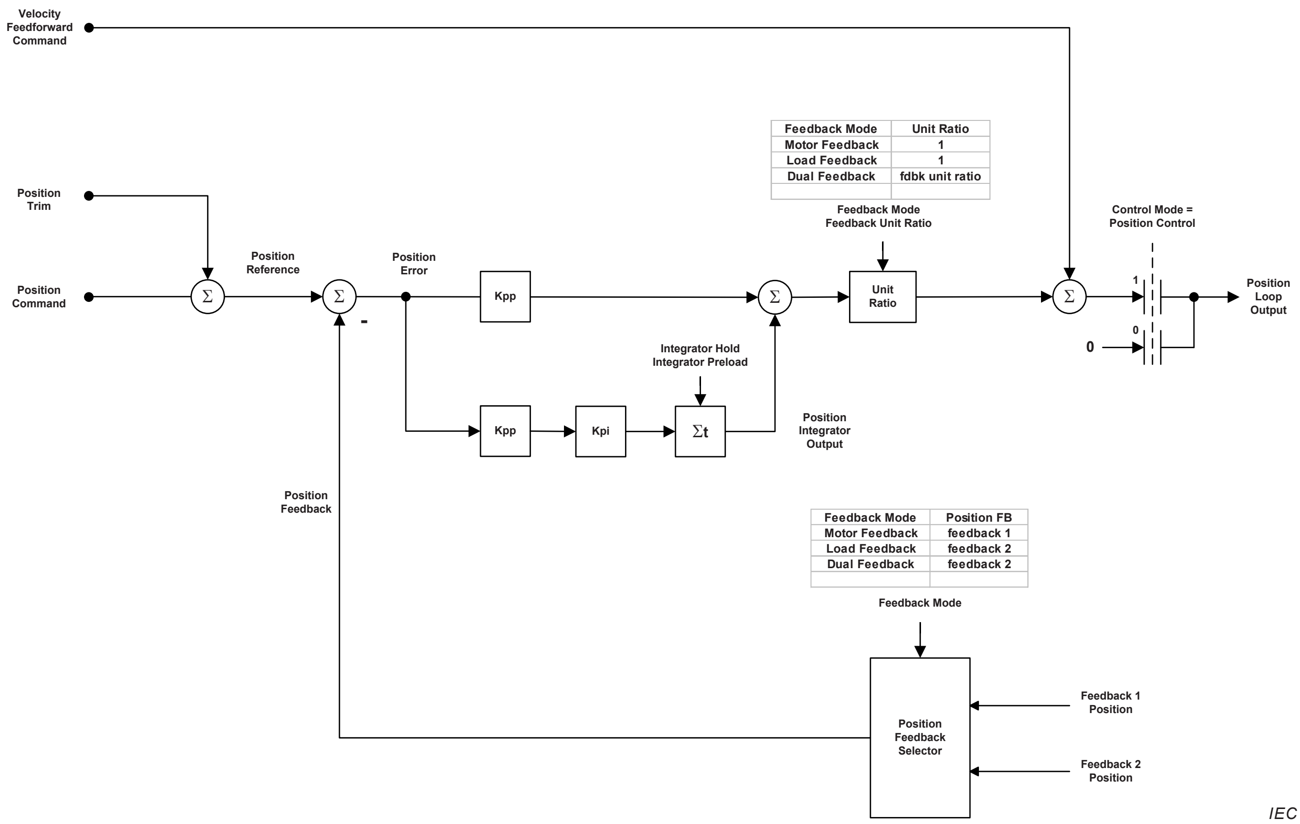

Figure 94 – Closed Loop Position Control

7.6.9.4.2 Position feedback selection

Feedback to the PI regulator can be derived from two different feedback channels. This flexibility allows the position loop to operate with either a motor based feedback device that is typically attached to the Feedback 1 channel or a load-side feedback device that is connected to the Feedback 2 channel. Which feedback source is used by the loop is governed by the Feedback Mode attribute.

When the Feedback Mode calls for Dual Feedback operation, the position loop utilizes the Feedback 2 channel and the velocity loop uses the Feedback 1 channel. Since the two feedback channels may not have the same feedback resolution, it is necessary to convert position loop output from Feedback 1 units to Feedback 2 units prior to applying the output to the velocity loop summing junction. This is done by scaling the position loop output via the Unit Scaling block using the Feedback Unit Ratio.

7.6.9.4.3 Position PI gains

The Proportional Gain of the classic PI controller sets the unity gain bandwidth of the position loop in rad/s, while the Integral Gain is used to drive the Position Error signal to zero to compensate for the effect of any static and quasi-static torque or forces applied to the load.

7.6.9.4.4 Velocity Feedforward

The inner velocity loop requires a non-zero command input to generate steady-state axis motor velocity. To provide the non-zero output from the device to the motor, a non-zero position loop output is required, which translates to a non-zero position error. This dynamic error between command position and actual position while moving is often called “following error”. Most closed loop motion control applications desire zero following error – all the time. This could be achieved to some extent through use of the position integral gain control as described above, but typically, the response time of the integrator action is too slow to be effective in high-performance motion control applications. An alternative approach that has superior dynamic response is to use Velocity Feedforward.

The Velocity Feedforward feature is used in Position Control Mode to provide the bulk of the Velocity Reference input necessary to generate the desired motor velocity. It does this by scaling the Fine Velocity Command signal output of the Command Generator by the Velocity Feedforward Gain and adding the resultant Velocity Feedforward Command signal to the Position Loop Output generated by the position loop to form the Velocity Reference signal. With this feature, the position loop does not need to generate much effort to produce the required velocity command level, hence the Position Error value is significantly reduced. The Velocity Feedforward Command signal allows the following error of the position control loop to be reduced to nearly zero when running at a constant velocity. This is important in applications such as electronic gearing and synchronization applications where it is necessary that the actual axis position does not significantly lag behind the commanded position at any time.

The optimal value for Velocity Feedforward Gain is theoretically 100 %. In reality, however, the value may need to be tweaked to accommodate velocity loops with finite loop gain. One aspect that may force a smaller Velocity Feedforward value is that increasing amounts of feedforward tends to exacerbate axis overshoot. For this reason, feedforward is not recommended for point-to-point positioning applications.

7.6.9.5 Velocity Control

In Velocity Control mode, there are two operative control methods supported by the object, Closed Loop Velocity Control and Open Loop Frequency Control.

7.6.9.6 Closed Loop Velocity Control

7.6.9.6.1 General

The Closed Loop velocity control method is targeted for applications that require tight speed regulation. The command input to the velocity loop can be derived directly from the Velocity Command of the Command Generator when configured for Velocity Control Mode or from the Position Loop Output when configured for Position Control Mode as described in 7.6.9.3 (see Figure 95).

When serving as an outer velocity loop in Velocity Control Mode, the device applies the Velocity Command input to the velocity command summing junction to generate the Velocity Reference signal into a classic PI regulator. Also contributing to the velocity command summing junction is the Velocity Trim input, which can be used in conjunction with an outer control loop to make minor adjustments to the velocity of the motor.

When serving as an inner velocity loop in Position Control Mode, the device applies the Position Loop Output signal to the input of the velocity command summing junction. Input signals that are not applicable to the configured control mode are generally set to zero.

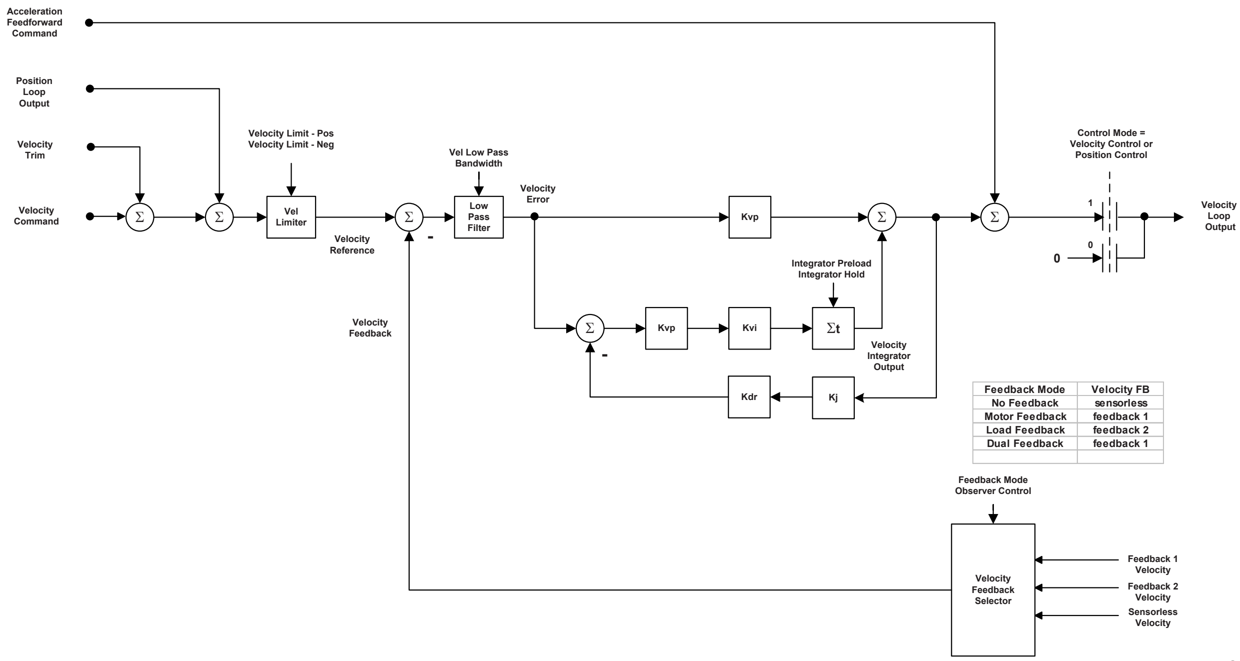

Figure 95 – Closed Loop Velocity Control

7.6.9.6.2 Velocity Limiter

The output of the velocity command summing junction signal passes through a classic limiter block to produce the Velocity Reference signal into the velocity loop. The Velocity Limiter block applies a directional velocity limit, either Velocity Limit – Pos or Velocity Limit – Neg, to the velocity command signal input that is based on the sign.

7.6.9.6.3 Velocity Feedback Selection

Feedback to the PI regulator can be derived from either of the two available feedback transducers, Feedback 1 or Feedback 2. Which feedback source is used by the loop is governed by the Feedback Mode enumeration. If Feedback Mode is No Feedback, indicating sensorless operation, the Velocity Feedback signal is the Sensorless Velocity signal generated by the sensorless control algorithm.

7.6.9.6.4 Velocity Error Filter

A low pass filter can be optionally applied to the velocity error signal generated by velocity loop summing junction. The output of this filter becomes the Velocity Error signal that is subsequently operated on by the velocity loop PI control algorithm. When used, the filter is typically set between 5 to 10 times the velocity loop bandwidth. It is recommended that this filter be a two pole IIR filter to maximum its effectiveness at quantization noise filtering.

7.6.9.6.5 Velocity PI gains

The velocity loop generates a Velocity Loop Output signal to the next inner loop via a classic PI control loop structure. The Proportional Gain of the controller sets the unity gain bandwidth of the velocity loop in rad/s, while the Integral Gain is used to drive the Velocity Error signal to zero to compensate for any static and quasi-static torque or forces applied to the load. The integrator path includes a Proportional Gain so that units of the Integral Gain represent the bandwidth of the integrator in rad/s.

The integral section of the velocity regulator includes an anti-windup feature. The anti-windup feature automatically holds the regulator's integral term when a limit condition is reached in the forward path. The anti-windup feature is conditioned by the arithmetic sign of the integrator's input. The integrator is held when the input's sign is such that the integrator output moves further into the active limit. In other words, the integrator is allowed to operate (not held) when the input would tend to bring the integrator output value off the active limit.

The integrator may also be configured for integrator hold operation. When the Integrator Hold attribute is set true, the regulator holds the integrator from accumulating while the axis is being commanded to move. This behavior is helpful in point-to-point positioning applications.

An automatic preset feature of the velocity regulator's integral term occurs when a transition is made from a torque control mode to speed control, using the Control Mode selection parameter. Upon transition to speed mode, the speed regulator's integral term is preset to the motor torque reference parameter. If the speed error is small, this provides a 'bumpless' transition from the last torque reference value present just prior to entering speed mode.

7.6.9.6.6 Velocity droop

Another feature of the velocity regulator is the velocity droop function. The velocity error input to the integral term is reduced by a fraction of the velocity regulator's output, as controlled by the droop gain setting, Kdr. As torque loading on the motor increases, actual motor speed is reduced in proportion to the droop gain. This is helpful when some level of compliance is required due to rigid mechanical coupling between two motors.

7.6.9.6.7 Acceleration Feedforward

The velocity loop requires a non-zero velocity loop output to generate steady-state axis motor acceleration. To provide the non-zero output from the drive to the motor, a non-zero velocity error is generally required. In position control applications, this non-zero velocity error translates to a non-zero position loop error. Since many closed loop motion control applications require near zero control loop error, this behavior is not desirable. Again, the position and velocity loop error could be reduced by applying the velocity integral gain control as described above, but the integrator action is still too slow to be very effective. The preferred approach with superior dynamic response is to use Acceleration Feedforward.

The Acceleration Feedforward feature is used to generate the bulk of the Acceleration Reference necessary to generate the commanded acceleration. It does this by scaling the Fine Acceleration Feedforward generated by the Command Generator by the Acceleration Feedforward Gain and adding the resultant Acceleration Feedforward Command Signal as an offset to the output of the velocity loop. With this feature, the velocity loop does not need to generate much control effort, thus reducing the amount of control loop error.

The optimal value for Acceleration Feedforward is 100 % theoretically. In reality, however, the value may need to be tweaked to accommodate variations in load inertia and the torque constant of the motor. Like Velocity Feedforward, Acceleration Feedforward can result in overshoot behavior and therefore shall not be used in point-to-point positioning applications.

When used in conjunction with the Velocity Feedforward, the Acceleration Feedforward allows the following error of the position or velocity control loop to be reduced to nearly zero during the acceleration and deceleration phases of motion. This is important in tracking applications that use electronic gearing and camming operations to precisely synchronize a slave axis to the movements of a master axis.

7.6.9.7 Open Loop Frequency Control

7.6.9.7.1 General

Another Velocity Control method is the open loop Frequency Control method associated with so called Volts/Hertz or variable frequency drives (VFDs) that do not have a current control loop and typically drive an induction motor. Velocity control with this method is achieved by controlling the voltage and frequency output of the drive device in some manner where voltage is generally proportional to frequency. For an induction motor, the velocity of the motor is determined by the Output Frequency of the drive device divided by the Motor Pole count. This control method is applicable to velocity control applications that do not require tight speed regulation and therefore do not require a feedback device. Figure 96 further defines this open loop velocity control method.

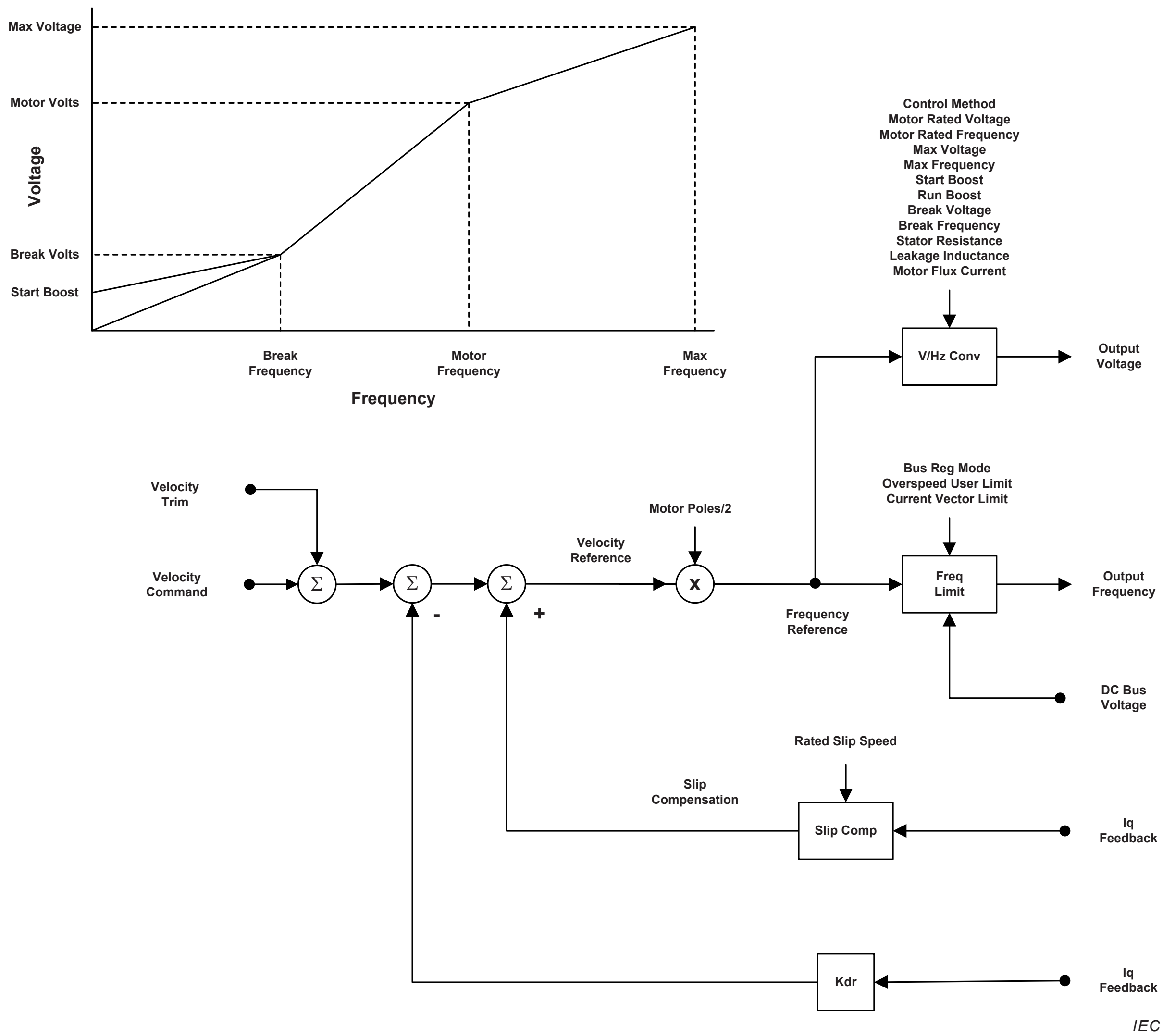

Figure 96 – Open Loop Frequency Control

7.6.9.7.2 Basic Volts/Hertz Operation

The Motion Device Axis Object provides a number attributes that are used to specify the relationship the drive device uses between output frequency (speed) and output voltage for a given (induction) motor. The Break Frequency and Break Voltage attributes define the point on the Volts/Hertz curve below which the Start Boost feature is applied. As the name indicates, Start Boost is used to provide a non-zero output voltage to the motor at stand-still to assist start-up. The contribution of the Start Boost to the output voltage of the drive device tapers off to zero when the motor reaches the Break Frequency. Above the break point, output voltage and output frequency follow a linear slope to the point defined by the Motor Rated Frequency and Motor Rated Voltage. From this point on, the Volts/Hertz curve follows another linear slope to the point defined by the Max Frequency and Max Voltage attributes. This segment of the Volts/Hertz curve allows for operation above the rated frequency and voltage of the motor in applications where that is required.

7.6.9.7.3 Sensorless Vector Operation

Sensorless Vector is an alternative Velocity Control Method that does not require configuration of a Volts/Hertz curve. Instead, by knowing the Stator Resistance and Leakage Inductance of the motor, the drive device can calculate the appropriate Output Voltage required for a given Output Frequency. This method provides better low speed velocity control behavior than using the Basic Volts/Hertz method.

7.6.9.7.4 Slip Compensation

When driving an induction motor at a specific frequency, the actual motor velocity is generally less than the command speed, given by the output frequency divided by the motor pole count, by an amount that is proportional to the load torque applied to the motor. This difference in speed is called “Slip” and is a configuration attribute associated with the motor. The Motion Device Axis Object supports a Slip Compensation feature that is common to variable frequency drives. The amount of Slip Compensation applied to the Velocity Reference is the product of the measured torque producing current, Iq, and the configured Induction Motor Rated Slip Speed.

7.6.9.7.5 Velocity Droop

Another feature defined for the Frequency Control method is the droop function. The droop function reduces the velocity reference by a scaled fraction of the torque producing current, Iq, as controlled by the droop gain setting, Kdr. As torque loading on the motor is increased, actual motor speed is reduced in proportion to the droop gain. This is helpful when some level of compliance is required when performing torque sharing between two motors on a common load.

7.6.9.8 Acceleration Control

7.6.9.8.1 General

While dynamic motor control via an acceleration command is not common in the industry, Acceleration Control was added to the Motion Device Axis Object to complete the dynamic progression from velocity control to torque control. The output of the velocity loop, Velocity Loop Output, also has units of acceleration. So, like the other control modes, the contributions of the Acceleration Command, Acceleration Trim, and Velocity Loop Output are summed to form the Acceleration Reference signal that serves as one of the primary inputs to the Torque Control section.

7.6.9.8.2 Acceleration Limiter

The output of the acceleration command summing junction signal passes through a classic limiter block to produce the Acceleration Reference signal. The Accel Limiter block applies a directional acceleration limit, either the Acceleration Limit and Deceleration Limit, to the input command signal based on the sign of the signal (see Figure 97).

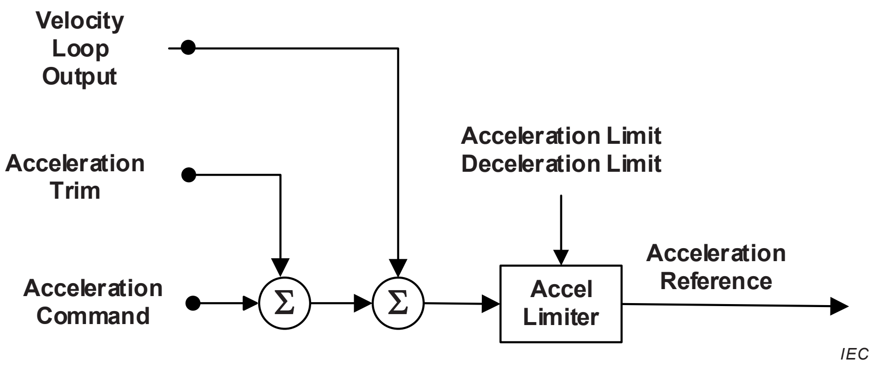

Figure 97 – Acceleration Control

7.6.9.9 Torque Control

7.6.9.9.1 General

Torque is generally proportional to acceleration and to the torque producing motor current, Iq. The purpose of the Torque Control structure is to combine input signals to create a Torque Reference from a number of different sources based on the Control Mode and apply various filters and compensation algorithms to the Torque Reference to create a Filtered Torque Reference. The Filtered Torque Reference signal is scaled by the reciprocal of the torque constant Kt of the motor to become the Iq Current Command input to the current loop. Since the motor current is also per unitized to the % Rated current of the motor, the torque constant Kt is nominally 1. In other words, in general it is assumed that 100 % rated current produces 100 % rated torque.

7.6.9.9.2 Torque input sources

The Torque Control section can take input from a variety of sources depending on the Control Mode. Input to the Torque Reference path can come via the cyclic Torque Command or Torque Trim signal in Torque Control mode. In Position or Velocity Control mode, torque input is derived from the outer velocity loop by bringing in the resulting acceleration signals and scaling these signals into equivalent torque (see Figure 98).

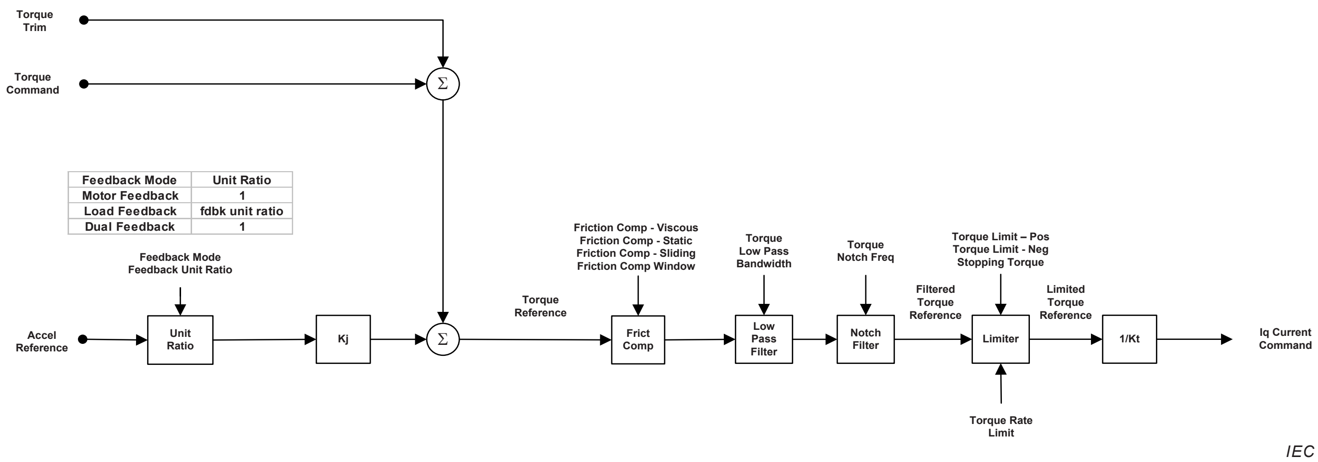

Figure 98 – Torque Control

7.6.9.9.3 Acceleration to torque scaling

Since the acceleration input signals into the Torque Control section are expressed in units of acceleration, a scaling factor Kj is needed to convert acceleration units to torque % Rated Torque units. This scaling factor, when properly configured, represents the total System

Inertia or mass of the system that includes the motor and the load and has the effect of cancelling the effects the system inertia/mass has on control loop response and loop gain settings. Since the torque units are expressed as % of Rated Torque of the motor, the units for the System Inertia attribute are % Rated per Motor Units/s^2. However, acceleration units can be expressed in Feedback 1 or Feedback 2 Units based on the Feedback Mode setting. Therefore, in the case where Feedback 2 applies, the acceleration signal needs to be scaled by the Feedback Unit Ratio as shown by the Unit Ratio block in Figure 98.

7.6.9.9.4 Friction compensation

7.6.9.9.4.1 General

Friction Compensation applies a compensating directional torque or force to the motor to overcome the effects of friction in the mechanical system, thus minimizing the amount of control effort required. Individual attributes have been defined to support compensation for static friction, sliding (Coulomb) friction, and viscous friction. A compensation window attribute is also provided to mitigate motor dithering associated with conventional friction compensation methods.

7.6.9.9.4.2 Static friction compensation

It is not unusual for an axis to have enough static friction, commonly called “sticktion”, in position control applications that even with a significant position error, the mechanical system refuses to budge. Of course, position integral gain can be used to generate enough output to the drive to correct the error, but this approach may not be responsive enough for the application. An alternative is to use Static Friction Compensation to break the sticktion in the presence of a non-zero position error. This is done by adding or subtracting a fixed torque level, as determined by the Static Friction Compensation attribute, to the Torque Reference signal value based on its current sign. This form of friction compensation shall only be applied when the axis is static, i.e. when there is no change in the position command.

The Static Friction Compensation value shall be just under the value that would overcome the sticktion. A larger value results in axis “dither”, a phenomena describing a rapid back and forth motion of the axis centred on the commanded position as it overcompensates for the sticktion.

To address the issue of dither when applying Static Friction Compensation, a Friction Compensation Window is applied around the current command position when the axis is at rest. If the actual position is within the Friction Compensation Window the Static Friction Compensation value is applied to the Servo Output but scaled by the ratio of the Position Error signal to the Friction Compensation Window. Within the window, the position loop and velocity loop integrators are also disabled to avoid the hunting effect that occurs when the integrators wind up. Thus, once the position error reaches or exceeds the value of the Friction Compensation Window attribute, the full Static Friction Compensation value is applied. Of course, shall the Friction Compensation Window be set to zero, this feature is effectively disabled.

A non-zero Friction Compensation Window has the effect of softening the Static Friction Compensation as its applied to the Torque Reference and reducing the dithering and hunting effects that it can create. This feature generally allows higher values of Static Friction Compensation to be applied resulting in better point-to-point positioning.

7.6.9.9.4.3 Sliding friction compensation

Sliding friction or Coulomb friction, by definition, is the component of friction that is independent of speed as long as the mechanical system is moving. Sliding friction is always less than static friction for a given mechanical system. The method of compensating for sliding friction is basically the same as that for static friction but the torque level added to the Torque Reference is less than that applied to overcome static friction and is determined by the Sliding Friction Compensation attribute. Sliding Friction Compensation is only applied when the axis is being commanded to move.

7.6.9.9.4.4 Viscous friction compensation

Viscous friction, by definition, is the component of friction that increases linearly with the speed of the mechanical system. The method of compensating for viscous friction is to multiply the configured Viscous Friction Compensation value by the speed of the motor and apply the result to the Torque Reference signal. Viscous Friction Compensation is only applied when the axis is being commanded to move.

7.6.9.9.5 Low Pass filter

The Low Pass Filter is effective in resonance control when the natural resonance frequency is much higher (>5x) than the control loop bandwidth. This filter works by reducing the amount of high-frequency energy in the device output that excite the natural resonance. The Low Pass Filter design can be single pole or multiple poles. Care shall be taken, however, to limit the amount of phase lag introduced by this filter to the control loop to avoid potential instability.

7.6.9.9.6 Notch filter

The Notch Filter is effective in resonance control when the natural resonance frequency is higher than the control loop bandwidth. Like the Low Pass filter, the notch filter works by significantly reducing the amount of energy in the device output that can excite the natural resonance. It can be used even when the natural resonance frequency is relatively close to the control loop bandwidth. That is because the phase lag introduced by the notch filter is localized around the notch frequency. For the notch filter to be effective, the Notch Filter Frequency has to be set very close to the natural resonance frequency of the load.

A typical equation for the notch filter is as follows:

$$\mathrm{G}(s) = \frac{s^2 + \omega_\mathrm{n}^2}{s^2 + s \times \omega_\mathrm{n}/\mathrm{Q} + \omega_\mathrm{n}^2}$$

In this equation, Q represents the sharpness of the notch. In most implementations, the sharpness, Q, is typically hard-coded in the device. The attenuation depth of the notch filter is infinite.

7.6.9.9.7 Torque limiter

The Filtered Torque Reference signal passes through a limiter block to produce the Limited Torque Reference signal. The Torque Limiter block applies a torque limit to the signal that is based on the sign of the torque reference signal input and the state of the axis. During normal operation it is the Torque Limit – Positive and Torque Limit – Negative attributes, set by the user, that are applied to the torque reference signal. When the axis is commanded to stop as part of a disable request or major fault condition, the device applies the Stopping Torque Limit.

Also included with the torque limit block is a built in Torque Rate of Change Limit. This feature limits the rate of change of the torque reference output.

7.6.9.9.8 Torque to current scaling

The final result of all this torque signal filtering, compensation, and limiting functionality is the Filtered Torque Reference signal, which when scaled by the reciprocal of the Torque Constant of the motor, 1/Kt, becomes the torque producing Iq Current Command signal to the current loop. Ideally, the relationship between motor torque and motor current is independent of position, time, current, and environmental conditions. In other words, the 1/Kt scaling in

Figure 98 has a nominal value of 1, i.e. 100 % rated torque translates to 100 % rated current. In practice, this may not be the case. Compensation can be applied to the 1/Kt value to address these issues at the drive vendors' discretion.

7.6.9.10 Current Control

7.6.9.10.1 General

In general, motor torque is controlled by controlling the orientation and magnitude of the motor stator current vector with respect to the rotor magnetic flux vector. The Current Control loop is responsible for providing this control and is actually composed of two PI loops, one that controls the torque producing current, Iq, and one that controls the flux producing current, Id. It is the quadrature component of current, Iq, that is used for dynamic torque control (see Figure 99).

In the case of an induction motor, the flux producing current, Id, is solely responsible for generating rotor flux. In the case of permanent magnet motors, rotor flux is generated by the rotor magnets and Id is only used to in some cases to extend the speed range of the motor by changing the angle of stator field relative to the rotor field. In this case, the angle of Iq relative to the rotor field remains the same, i.e. at quadrature. But since the vector combination of Iq and Id determines the stator flux angle relative to the rotor, increasing amounts of Id can shift the stator flux away from quadrature to extend the speed range of the motor at the expense of torque.

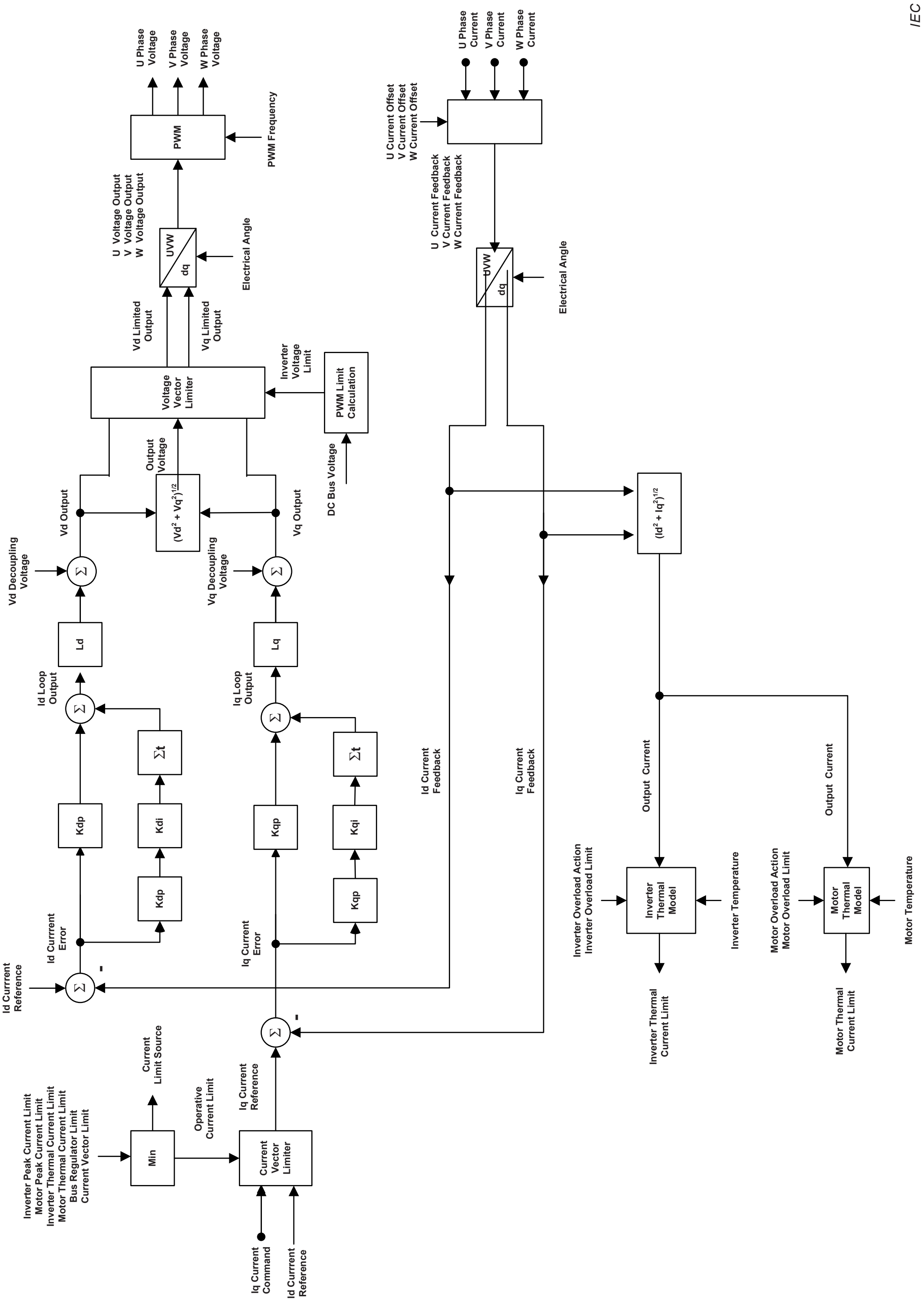

Figure 99 – Closed Loop Current Vector Control

7.6.9.10.2 Current vector limiter

The Iq Current Command passes through a Current Vector Limiter block before becoming the Iq Current Reference signal. This limiter block computes the combined vector magnitude of the Iq Current Reference and the Id Current Reference signals. The resultant current vector magnitude is compared to the Operative Current Limit that represents the minimum current limit from among a set of potential current limits of the drive device and motor. If the vector magnitude exceeds the Operative Current Limit, the Iq Current Reference is reduced so the vector magnitude equals the Operative Current Limit. Potential current limit sources can be the Peak Current Limit ratings as well as the Thermal Limits for the Motor and Drive Inverter. Another possible limit source is the user configurable Current Vector Limit attribute. Some of these limits are conditional and dynamic, such as the Motor and Inverter Thermal Current Limits derived from the thermal models for these devices. These limits are only active when the corresponding Motor and Inverter Overload Action attributes are set to provide current fold-back. The thermal current limits in this case would decrease as the simulated temperature of the modeled devices increases. The Bus Regulator Limit is only applied when the motor is regenerating power onto the DC Bus and is based on the Regenerative Power Limit.

With all these potential current limit sources that could be operative, a Current Limit Source attributes was included with the Motion Device Axis Object to identify the source of the active current limit.

7.6.9.10.3 Voltage output

The output of each current loop is scaled by the motor inductance to generate a voltage command to the vector transformation block. It is the job of the vector transformation block to transform the torque producing, Vq, and flux producing, Vd, command signals from the rotating synchronous reference frame to the stationary stator reference frame. The resultant U, V, and W Output Voltage values are then applied to the motor by Pulse Width Modulation (PWM). The PWM Frequency is also a configurable attribute of the Motion Device Axis Object.

The magnitude of the Vq, Vd vector is calculated in real time as the total Output Voltage signal. The maximum Output Voltage signal that can be applied to the motor is ultimately limited by the DC Bus Voltage and enforced by the Voltage Vector Limiter. Any attempt to exceed this value results in an Inverter Voltage Limit condition.

7.6.9.10.4 Current feedback

Current feedback signals to the current loop are provided by two or three current sensors. The signals from these sensors are conditioned and corrected for device specific offsets to become the U, V, and W Current Feedback signals associated with the stationary motor stator frame. These three signals are transformed back to the synchronous reference frame to generate the Iq and Id Current Feedback signals. The magnitude of the Iq, Id current vector is calculated in real-time and used as an input to the thermal models for the inverter and motor.

7.6.9.10.5 Motor commutation

Motor commutation is critical to closed loop motor control. The orientation of the motor rotor can be determined from a feedback source mounted to the motor. The actual commutation source is the motor feedback device assigned to Feedback 1. Once the feedback device is calibrated to the absolute orientation of the rotor using the Commutation Offset attribute, the commutation block can then generate the true Electrical Angle of the rotor. This signal is used to perform the vector transforms between the rotary and stationary motor frames and can also be used for any other algorithms that require knowledge of rotor position.

Bibliography

[1] IEC 60050-351:2013,*International Electrotechnical Vocabulary – Part 351: Control technology* (available at <http://www.electropedia.org>)

[2] IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

[3] IEC 61158 (all parts), *Industrial communication networks – Fieldbus specifications*

[4] IEC 61158-2:2014, *Industrial communication networks – Fieldbus specifications – Part 2 (Ed.4.0): Physical layer specification and service definition*

[5] IEC 61158-3-2:2014, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

[6] IEC 61499-1:2005, *Function blocks – Part 1: Architecture*

[7] IEC 61784-1:2014, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

[8] IEC 61784-2:2014, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

[9] IEC 61800 (all parts), *Adjustable speed electrical power drive systems*

[10] IEC 61800-7 (all parts), *Adjustable speed electrical power drive systems – Generic interface and use of profiles for power drive systems*

[11] IEC 61800-7-201, *Adjustable speed electrical power drive systems – Part 7-201: Generic interface and use of profiles for power drive systems – Profile type 1 specification*

[12] IEC 61800-7-203, *Adjustable speed electrical power drive systems – Part 7-203: Generic interface and use of profiles for power drive systems – Profile type 3 specification*

[13] IEC 61800-7-204, *Adjustable speed electrical power drive systems – Part 7-204: Generic interface and use of profiles for power drive systems – Profile type 4 specification*

[14] IEC 61800-7-301, *Adjustable speed electrical power drive systems – Part 7-301: Generic interface and use of profiles for power drive systems – Mapping of profile type 1 to network technologies*

[15] IEC 61800-7-302, *Adjustable speed electrical power drive systems – Part 7-302: Generic interface and use of profiles for power drive systems – Mapping of profile type 2 to network technologies*

[16] IEC 61800-7-303, *Adjustable speed electrical power drive systems – Part 7-303: Generic interface and use of profiles for power drive systems – Mapping of profile type 3 to network technologies*

[17] IEC 61800-7-304, *Adjustable speed electrical power drive systems – Part 7-304: Generic interface and use of profiles for power drive systems – Mapping of profile type 4 to network technologies*

[18] IEC 62026-3, *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet™*

[19] IEC TR 62390:2005, *Common Automation Device – Profile Guideline*

[20] ISO/IEC 2382-15:1999, *Information technology – Vocabulary – Part 15: Programming languages*

[21] ISO/IEC 19501, *Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2*

[22] ISO 15745-1:2003, *Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description*

[23] EN 50325-4, *Industrial communication subsystem based on ISO 11898 (CAN) for controller-device interfaces – Part 4: CANopen*

[24] IETF RFC 768, *User Datagram Protocol*, (available at http://www.ietf.org)

[25] IETF RFC 793, *Transmission Control Protocol*, (available at http://www.ietf.org)

[26] ODVA: *THE CIP NETWORKS LIBRARY – Volume 1: Common Industrial Protocol (CIP™) – Edition 3.16, April 2014,* (available at <http://www.odva.org>)

[27] ODVA: *THE CIP NETWORKS LIBRARY – Volume 2: EtherNet/IP™ Adaptation of CIP – Edition 1.17, April 2014,* (available at <http://www.odva.org>)

[28] ODVA: *THE CIP NETWORKS LIBRARY – Volume 3: DeviceNet™ Adaptation of CIP – Edition 1.14, November 2013,* (available at <http://www.odva.org>)

[29] ODVA: *THE CIP NETWORKS LIBRARY – Volume 4: ControlNet™ Adaptation of CIP – Edition 1.8, April 2013,* (available at <http://www.odva.org>)

[30] ODVA: *THE CIP NETWORKS LIBRARY – Volume 5: CIP Safety™ – Edition 2.9, April 2014,* (available at <http://www.odva.org>)

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.