

DIN EN 61800-7-202

ICS 29.160.30; 29.200

**Elektrische Leistungsantriebssysteme mit einstellbarer Drehzahl –
Teil 7-202: Generisches Interface und Nutzung von Profilen für
Leistungsantriebssysteme (PDS) –
Spezifikation von Profil-Typ 2 (IEC 61800-7-202:2007);
Englische Fassung EN 61800-7-202:2008**

Adjustable speed electrical power drive systems –
Part 7-202: Generic interface and use of profiles for power drive systems –
Profile type 2 specification (IEC 61800-7-202:2007);
English version EN 61800-7-202:2008

Entraînements électriques de puissance à vitesse variable –
Partie 7-202: Interface et utilisation génériques de profils pour les entraînements
électriques de puissance –
Spécifications des profils de type 2 (CEI 61800-7-202:2007);
Version anglaise EN 61800-7-202:2008

Gesamtumfang 187 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2008-02-01 angenommene EN 61800-7-202 gilt als DIN-Norm ab 2008-11-01.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN IEC 61800-7-2:2006-06.

Für diese Norm ist das nationale Arbeitsgremium K 226 „Ausrüstung von Starkstromgeräten und -anlagen mit elektronischen Betriebsmitteln“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Die enthaltene IEC-Publikation wurde vom SC 22G „Semiconductor power converters for adjustable speed electric drive systems“ erarbeitet.

Diese Norm erscheint ausschließlich in der englischsprachigen Originalfassung.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (maintenance result date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

**Adjustable speed electrical power drive systems –
Part 7-202: Generic interface and use of profiles for power drive systems –
Profile type 2 specification
(IEC 61800-7-202:2007)**

Entraînements électriques de puissance
à vitesse variable –
Partie 7-202: Interface et utilisation
génériques de profils pour les
entraînements électriques de puissance –
Spécifications des profils de type 2
(CEI 61800-7-202:2007)

Elektrische Leistungsantriebssysteme
mit einstellbarer Drehzahl –
Teil 7-202: Generisches Interface
und Nutzung von Profilen
für Leistungsantriebssysteme (PDS) –
Spezifikation von Profil-Typ 2
(IEC 61800-7-202:2007)

This European Standard was approved by CENELEC on 2008-02-01. 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 Central Secretariat 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 Central Secretariat has the same status as the official versions.

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

CENELEC

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

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 22G/184/FDIS, future edition 1 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 was approved by CENELEC as EN 61800-7-202 on 2008-02-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2008-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-02-01

The International Electrotechnical Commission (IEC) and CENELEC draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning:

Publication/ Application serial number	Holder	Title
US 11/241,539	[RA]	Time Stamped Motion Control Network Protocol That Enables Balanced Single Cycle Timing and Utilization of Dynamic Data Structures

The IEC and CENELEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the IEC. Information may be obtained from:

[RA] Rockwell Automation, Inc.
1201 S. Second Street
Milwaukee, WI 53204
USA
Attention: Intellectual Property Dept.

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 and CENELEC shall not be held responsible for identifying any or all such patent rights.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61800-7-202:2007 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 61158	NOTE	Harmonized in EN 61158 series (not modified).
IEC 61158-2	NOTE	Harmonized as EN 61158-2:2008 (not modified).
IEC 61158-3-2	NOTE	Harmonized as EN 61158-3-2:2008 (not modified).
IEC 61158-4-2	NOTE	Harmonized as EN 61158-4-2:2008 (not modified).
IEC 61499-1	NOTE	Harmonized as EN 61499-1:2005 (not modified).
IEC 61784-1	NOTE	Harmonized as EN 61784-1:2008 (not modified).
IEC 61784-2	NOTE	Harmonized as EN 61784-2:2008 (not modified).
IEC 61800	NOTE	Harmonized in EN 61800 series (not modified).
IEC 61800-7-201	NOTE	Harmonized as EN 61800-7-201:2008 (not modified).
IEC 61800-7-203	NOTE	Harmonized as EN 61800-7-203:2008 (not modified).
IEC 61800-7-204	NOTE	Harmonized as EN 61800-7-204:2008 (not modified).
IEC 61800-7-301	NOTE	Harmonized as EN 61800-7-301:2008 (not modified).
IEC 61800-7-302	NOTE	Harmonized as EN 61800-7-302:2008 (not modified).
IEC 61800-7-303	NOTE	Harmonized as EN 61800-7-303:2008 (not modified).
IEC 61800-7-304	NOTE	Harmonized as EN 61800-7-304:2008 (not modified).

CONTENTS

INTRODUCTION	11
1 Scope	14
2 Normative references	14
3 Terms, definitions and abbreviated terms	15
3.1 Terms and definitions	15
3.2 Abbreviated terms	21
4 General	21
4.1 General	21
4.2 Control modes	22
4.2.1 General	22
4.2.2 Control methods	22
4.2.3 Control nomenclature	22
4.2.4 Position Control	23
4.2.5 Velocity Control	24
4.2.6 Acceleration Control	26
4.2.7 Torque Control	26
4.2.8 No Control	27
5 Data types	28
5.1 Data type overview	28
5.2 Conventions	29
6 CIP Drive Profile	29
6.1 Object model	29
6.1.1 Object overview	29
6.1.2 Object description	30
6.2 How objects affect behaviour	30
6.3 Defining object interfaces	30
6.4 I/O Connection messages	31
6.4.1 General	31
6.4.2 CIP Motion I/O connection	31
6.4.3 Controller-to-Device connection	34
6.4.4 Device-to-Controller connection	45
6.4.5 Fixed Device connection format	52
6.4.6 CIP Motion connection timing	53
6.5 Drive startup procedure	63
6.5.1 General	63
6.5.2 Motion I/O Connection creation	63
6.5.3 Motion Axis Object configuration	63
6.5.4 Time Synchronisation	63
6.6 Device visualisation	65
6.7 Ethernet Quality of Service (QoS)	65
7 Motion Axis Object	66
7.1 General considerations	66
7.1.1 General	66

7.1.2	Object overview	66
7.1.3	Motion Axis Object abstraction	66
7.1.4	Motion Control Axis Object	67
7.1.5	Drive control classification.....	67
7.1.6	Required vs. Optional in implementation.....	68
7.2	Class attributes	76
7.2.1	General	76
7.2.2	Semantics.....	79
7.3	Instance attributes	82
7.3.1	General	82
7.3.2	Motion Control configuration attributes	83
7.3.3	Motor attributes.....	85
7.3.4	Feedback attributes	93
7.3.5	Event Capture attributes.....	99
7.3.6	Command reference generation attributes	103
7.3.7	Control mode attributes	107
7.3.8	Stopping & Braking attributes	117
7.3.9	DC Bus Control attributes.....	121
7.3.10	Power and thermal management attributes	122
7.3.11	Axis Status Attributes	124
7.3.12	Exception, fault, and alarm status attributes	128
7.3.13	Exception limit attributes	133
7.3.14	Exception action configuration attribute	134
7.3.15	Initialisation fault status attributes	136
7.3.16	Start inhibit status attributes.....	138
7.3.17	Axis statistical attributes.....	139
7.3.18	Axis info attributes	140
7.3.19	General purpose I/O attributes	140
7.3.20	Local Mode Attributes	141
7.4	Common services	141
7.4.1	Supported services	141
7.4.2	Service specific data	142
7.5	Object specific services	143
7.5.1	Supported services	143
7.5.2	Service specific data	144
7.6	Behaviour.....	153
7.6.1	State model	153
7.6.2	State behaviour.....	157
7.6.3	Fault and alarm behaviour.....	159
7.6.4	Start Inhibit behaviour	160
7.6.5	Visualization behaviour	160
7.6.6	Command generation behaviour.....	164
7.6.7	Feedback Interface Behavior.....	168
7.6.8	Event Capture Behavior	171
7.6.9	Control Mode Behaviour.....	172
	Bibliography	183

Figure 1 – Structure of IEC 61800-7	13
Figure 2 – Open loop position control	23
Figure 3 – Closed loop position control	24
Figure 4 – Open loop velocity control	25
Figure 5 – Closed loop velocity control	25
Figure 6 – Acceleration control	26
Figure 7 – Torque control	27
Figure 8 – No control	28
Figure 9 – Object Model for a CIP Motion drive	29
Figure 10 – CIP Motion I/O connection model	31
Figure 11 – Controller-to-Device Connection Format	32
Figure 12 – Device-to-Controller Connection Format	33
Figure 13 – CIP Motion I/O Connection Channels	34
Figure 14 – CIP Motion Controller-to-Device Connection Format	34
Figure 15 – Connection Header	35
Figure 16 – Connection Format	35
Figure 17 – Connection Header	36
Figure 18 – Instance Data Block	37
Figure 19 – Instance Data Header	38
Figure 20 – Cyclic Data Header	38
Figure 21 – Cyclic Write Data Block	42
Figure 22 – Event Data Block	43
Figure 23 – Service Data Block	45
Figure 24 – CIP Motion Controller-to-Device Connection Format	45
Figure 25 – Connection Header	46
Figure 26 – Connection Header	47
Figure 27 – Adjustment of actual position data based on device time stamp	48
Figure 28 – Instance Data Block	48
Figure 29 – Instance Data Header	49
Figure 30 – Cyclic Data Block	49
Figure 31 – Cyclic Read Data Block	50
Figure 32 – Event Data Block	51
Figure 33 – Service Data Block	52
Figure 34 – Fixed Controller to Device Connection Format (fixed size = 16 bytes)	53
Figure 35 – Fixed Device to Controller Connection Format (fixed size = 16 bytes)	53
Figure 36 – CIP Motion I/O Connection timing model	54
Figure 37 – Controller-to-Drive Connection timing with fine interpolation	55
Figure 38 – Controller-to-Drive Connection timing with unequal update periods	57
Figure 39 – Use of Time Stamp to adjust actual position to the controller's timebase	58
Figure 40 – Coordination of two drives with different Update Periods.	59
Figure 41 – Coordination of multiple drive axes in case of delayed Controller-to-Device Connection packets	60

Figure 42 – Propagation of a step change in time	61
Figure 43 – Group Sync of CIP Motion drives	64
Figure 44 – Object components for CIP motion architecture	66
Figure 45 – Controller Consumed Connection Data Format.....	79
Figure 46 – Controller Produced Connection Data Format.....	80
Figure 47 – Control Mode bit field	84
Figure 48 – IEEE per phase motor model	85
Figure 49 – Feedback Configuration bit field.....	99
Figure 50 – Event Checking Control word field	101
Figure 51 – Event Checking Status word field.....	102
Figure 52 – Interpolation Control word field	106
Figure 53 – Get Axis Attribute List Request Format	144
Figure 54 – Get Axis Attribute List Response Format.....	145
Figure 55 – Get Axis Attribute List Response – Example 1.....	146
Figure 56 – Get Axis Attribute List Response – Example 2.....	146
Figure 57 – Get Axis Attribute List Response – Example 3.....	146
Figure 58 – Get Axis Attribute List Response – Example 4.....	146
Figure 59 – Set Axis Attribute List Request Format.....	147
Figure 60 – Set Axis Attribute List Request – Example 1.....	148
Figure 61 – Set Axis Attribute List Request – Example 2.....	148
Figure 62 – Set Axis Attribute List Request – Example 3.....	148
Figure 63 – Set Axis Attribute List Response Format	148
Figure 64 – Set Cyclic Write List Request Format	149
Figure 65 – Set Cyclic Write List Response Format	149
Figure 66 – Set Cyclic Read List Request Format	150
Figure 67 – Set Cyclic Read List Response Format	150
Figure 68 – Motion Axis Object State Model	154
Figure 69 – Command Generator	165
Figure 70 – Feedback Channels 1 and 2	169
Figure 71 – Feedback Channels 3 and 4	170
Figure 72 – Event Capture Functionality	171
Figure 73 – No Control (Feedback Only).....	173
Figure 74 – Closed Loop Position Control.....	174
Figure 75 – Closed Loop Velocity Control.....	176
Figure 76 – Open Loop Frequency Control	178
Figure 77 – Acceleration Control	179
Figure 78 – Torque Control	180
Figure 79 – Closed Loop Current Vector Control.....	181
Table 1 – Data types.....	28
Table 2 – Objects present in a CIP Motion drive device	30
Table 3 – Object effect on behavior	30

Table 4 – Object interfaces	30
Table 5 – Time Data Set	36
Table 6 – Axis Control	39
Table 7 – Command Data Set	39
Table 8 – Command Data Element to Motion Axis Object Attribute mapping.....	40
Table 9 – Actual Data Set	40
Table 10 – Actual Data Element to Motion Axis Object Attribute Mapping.....	41
Table 11 – Status Data Set configuration	41
Table 12 – Interpolation Control	42
Table 13 – Axis Response	49
Table 14 – Event Type.....	51
Table 15 – Propagation of a step change in time (example 1)	61
Table 16 – Propagation of a step change in time (example 2)	62
Table 17 – CIP Motion visualisation components.....	65
Table 18 – Instance attribute implementation vs. Device Control Code.....	68
Table 19 – Class attributes for the Motion Axis Object.....	77
Table 20 – Node Control bit definitions.....	80
Table 21 – Node Status bit definitions	81
Table 22 – Node Faults code definitions	81
Table 23 – Node Alarms code definitions	82
Table 24 – Dynamic Unit vs. Feedback Configuration	83
Table 25 – Motion Control configuration attributes.....	84
Table 26 – Motor Control field enumeration definitions	84
Table 27 –Control Method field enumeration definitions	85
Table 28 – General Motor Info Attributes	87
Table 29 – General Motor Configuration Attributes	87
Table 30 – General PM Motor Configuration Attributes	90
Table 31 – General Rotary Motor Configuration Attributes	90
Table 32 – General Linear Motor Configuration Attributes.....	91
Table 33 – Rotary PM Motor Configuration Attributes	91
Table 34 – Linear PM Motor Configuration Attributes.....	92
Table 35 – Induction Motor Configuration Attributes	92
Table 36 – General Feedback Info Attributes.....	93
Table 37 – Feedback Configuration Attributes	94
Table 38 – Feedback Selection field enumeration definitions	99
Table 39 – Event Attributes.....	100
Table 40 – Event Checking Control bit definitions.....	101
Table 41 – Event Checking Status bit definitions	102
Table 42 – Command Generator Signal Attributes	103
Table 43 – Command Generator Configuration Attributes	104
Table 44 – Command Target Update enumeration definition	107
Table 45 – Position Loop Signal Attributes	107

Table 46 – Position Loop Configuration Attributes.....	108
Table 47 – Velocity Loop Signal Attributes.....	109
Table 48 – Velocity Loop Configuration Attributes.....	109
Table 49 – Acceleration Signal Attributes	111
Table 50 – Torque/Force Reference Signal Attributes	112
Table 51 – Torque/Force Reference Configuration Attributes	112
Table 52 – Current Loop Signal Attributes	113
Table 53 – Current Loop Configuration Attributes	115
Table 54 – Frequency Control Signal Attributes	116
Table 55 – Frequency Control Configuration Attributes	116
Table 56 – Drive Output Attributes	117
Table 57 – Stopping/Braking Attributes.....	117
Table 58 – Stopping Mode enumeration definitions	119
Table 59 – DC Bus Control Attributes	121
Table 60 – Power and Thermal Management Status Attributes.....	123
Table 61 – Power and Thermal Management Configuration Attributes	124
Table 62 – Axis Status Attributes	125
Table 63 – Axis Status bit definitions.....	126
Table 64 – Axis I/O Status bit definitions	126
Table 65 –Status Data Set bit definitions	127
Table 66 – Exception, Fault and Alarm Status Attributes.....	128
Table 67 – Standard Exception Table	131
Table 68 – Exception Factory Limit Info Attributes	133
Table 69 – Exception User Limit Configuration Attributes.....	134
Table 70 – Exception Action Configuration Attribute	135
Table 71 – Exception Action Bit Definitions.....	135
Table 72 – Initialisation Fault Status Attributes	136
Table 73 – Standard Initialisation Fault Table	138
Table 74 – Start Inhibit Status Attributes	138
Table 75 – Standard Start Inhibit Table	139
Table 76 –Statistical Attributes.....	139
Table 77 – Axis Info Attributes	140
Table 78 – General Purpose I/O Attributes	140
Table 79 – Local Mode Configuration Attributes.....	141
Table 80 – Motion Axis Object – Common Services	141
Table 81 – Group_Sync Request Data Structure.....	142
Table 82 – Group_Sync Response Data Structure	142
Table 83 – Motion Axis Object – Object Specific Services	143
Table 84 – Run Motor Test Request Data Structure	151
Table 85 – Get Motor Test Response Data Structure	151
Table 86 – Run Inertia Test Request Data Structure	152
Table 87 – Get Inertia Test Response Data Structure	152

Table 88 – Run Hookup Test Request Data Structure	153
Table 89 – Get Hookup Test Response Data Structure	153
Table 90 – Axis Control Request Code	154
Table 91 – Axis Response Acknowledge Codes	155
Table 92 – Completion Criteria for Requested Operation	155
Table 93 – Successful Axis Control Request Cycle	156
Table 94 – Unsuccessful Axis Control Request Cycle	156
Table 95 – Axis State Mapping to Identity Object with LED Behavior.....	161
Table 96 – CIP Motion Device Seven-Segment Display Behavior	162
Table 97 – CIP Motion Multi-Character Alphanumeric Display Behavior	163
Table 98 – Multi-Axis Multi-Character Alphanumeric Display Behavior	164

INTRODUCTION

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

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 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 (builds modules, machines, plants etc.)

- 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 specified with incompatibilities. It is up to the systems integrator to write interfaces to the application software to handle that which 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 adopt 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 profile types for CiA 402¹, CIP Motion^{TM2}, PROFIdrive³ and SERCOS Interface^{TM4} 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 MotionTM).

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

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 CANopen⁵, EtherCAT^{TM6}, Ethernet Powerlink^{TM7}, DeviceNet^{TM8}, ControlNet^{TM9}, EtherNet/IP^{TM10}, PROFIBUS¹¹, PROFINET¹² and SERCOS Interface).

-
- 1 CiA 402 is a trade name of CAN in Automation, 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 name holder or any of its products. Compliance to this profile does not require use of the trade name CiA 402.
 - 2 CIP MotionTM is a trade name of Open DeviceNet Vendor Association, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade name CIP MotionTM. Use of the trade name CIP MotionTM requires permission of Open DeviceNet Vendor Association, Inc.
 - 3 PROFIdrive is a trade name of PROFIBUS 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 International.
 - 4 SERCOSTM and SERCOS InterfaceTM are trade names 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 name holder or any of its products. Compliance to this profile does not require use of the trade name SERCOS and SERCOS interface. Use of the trade name SERCOS and SERCOS interface requires permission of the trade name holder.
 - 5 CANopen is an acronym for Controller Area Network *open* and is used to refer to EN 50325-4.
 - 6 EtherCATTM is a trade name 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 trademark holder or any of its products. Compliance to this profile does not require use of the trade name EtherCATTM. Use of the trade name EtherCATTM requires permission of the trade name holder.
 - 7 Ethernet PowerlinkTM is a trade name of B&R, control of trade name use is given to the non profit organisation EPSG. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade name Ethernet PowerlinkTM. Use of the trade name Ethernet PowerlinkTM requires permission of the trade name holder.
 - 8 DeviceNetTM is a trade name of Open DeviceNet Vendor Association, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade name DeviceNetTM. Use of the trade name DeviceNetTM requires permission of Open DeviceNet Vendor Association, Inc.
 - 9 ControlNetTM is a trade name of ControlNet International, Ltd. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade name ControlNetTM. Use of the trade name ControlNetTM requires permission of ControlNet International, Ltd.
 - 10 EtherNet/IPTM is a trade name of ControlNet International, Ltd. and Open DeviceNet Vendor Association, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade name EtherNet/IPTM. Use of the trade name EtherNet/IPTM requires permission of either ControlNet International, Ltd. or Open DeviceNet Vendor Association, Inc.
 - 11 PROFIBUS is a trade name of PROFIBUS 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 International.
 - 12 PROFINET is a trade name of PROFIBUS 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 International.

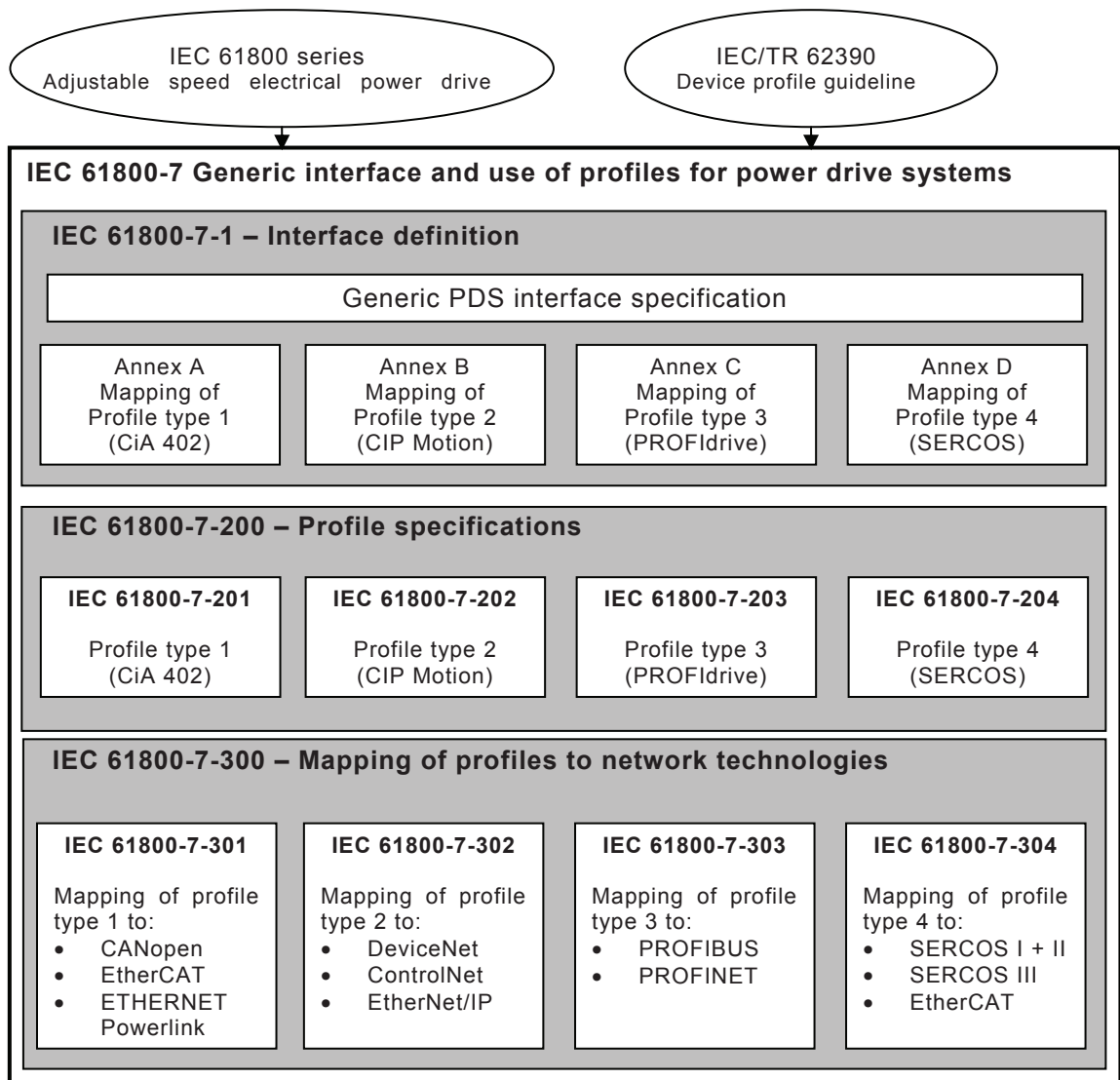


Figure 1 – Structure of IEC 61800-7

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

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

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.

This part of IEC 61800-7 specifies profile type 2 (CIP Motion™) for Power Drive Systems (PDS). Profile type 2 can be mapped onto different communication network technologies.

2 Normative references

The following referenced documents are indispensable for the application of this document. 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 61131-3, *Programmable controllers – Part 3: Programming languages*

IEC 61158-5-2, *Industrial communication networks – Fieldbus specifications – Part 5-2 (Ed.1.0): Application layer service definition – Type 2 elements*

IEC 61158-6-2, *Industrial communication networks – Fieldbus specifications – Part 6-2 (Ed.1.0): Application layer protocol specification – Type 2 elements*

IEC 61588:2004, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61800-7 (all parts), *Adjustable speed electrical power drive systems – Generic interface and use of profiles for power drive systems*

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

IEEE 802.1Q, *IEEE standards for local and metropolitan area networks – Virtual bridged local area networks*

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

[IEV 351-21-37]

3.1.2

application

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

NOTE An application may be distributed among resources, and may communicate with other applications.

[IEC/TR 62390:2005, 3.1.2, modified]

3.1.3

application mode

type of application that can be requested from a PDS

NOTE The different application modes reflect the control loop for torque control, velocity control, position control or other applications like homing.

3.1.4

attribute

property or characteristic of an entity

[IEC/TR 62390:2005, 3.1.3]

3.1.5

axis

logical element of a motion control system that exhibits some form of movement

NOTE 1 Axes can be rotary or linear, physical or virtual, controlled or simply observed.

NOTE 2 A physical axis may include one or more of the following components: a motion sensor, a motion control structure, a power amplifier, and a motion actuator.

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™¹³

extensions to the CIP services and protocol to support motion control over CIP networks

¹³ CIP Motion™ and CIP Sync™ are trade names of Open DeviceNet Vendor Association, Inc. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this profile does not require use of the trade names CIP Motion™ or CIP Sync™. Use of the trade names CIP Motion™ or CIP Sync™ requires permission of Open DeviceNet Vendor Association, Inc.

3.1.8

CIP (Motion) drive

drive that complies with the CIP (Motion) specification

3.1.9

CIP Motion drive profile

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

3.1.10

CIP Motion I/O connection

periodic bi-directional, class 1, CIP connection between a controller and a drive that is defined as part of the CIP Motion specification

3.1.11

CIP Sync™¹³

extensions to the CIP services and protocol to encapsulate IEC 61588:2004 time synchronisation functionality over a CIP Network (see Time Sync Object in IEC 61158-5-2 and IEC 61158-6-2)

3.1.12

class

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

[ISO/IEC 19501, modified]

3.1.13

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 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).

3.1.14

commands

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

NOTE 1 The behaviour is reflected by states or operating modes.

NOTE 2 The different commands may be represented by one bit each.

3.1.15

control

purposeful action on or in a process to meet specified objectives

[IEV 351-21-29, modified]

3.1.16

control device

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

3.1.17

converter

device that generally converts AC input to DC output

NOTE 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.

3.1.18

cyclic data block

high priority real-time data block that is transferred by a CIP Motion connection on a periodic basis

3.1.19

data type

set of values together with a set of permitted operations

[ISO/IEC 2382-15:1999, 15.04.01, modified]

3.1.20

device

field device

networked independent physical entity of an industrial automation system capable of performing specified functions in a particular context and delimited by its interfaces

[IEC 61499-1:2005, 3.30, modified]

entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system

[ISO 15745-1:2003, 3.11]

3.1.21

device profile

representation of a device in terms of its parameters, parameter assemblies and behaviour according to a device model that describes the device's data and behaviour as viewed through a network

NOTE This is a definition from IEC/TS 61915 which is extended by the addition of the device functional structure.

[IEC/TR 62390:2005, 3.19, modified]

3.1.22

drive

device designed to control the dynamics of a motor

3.1.23

event data block

medium priority real-time data block that is transferred by a CIP Motion connection only after a specified event occurs

NOTE Registration and marker input transitions are typical drive events.

3.1.24

functional element

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

NOTE 1 A functional element has an interface, associations to other functional elements and functions.

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

[IEC/TR 62390:2005, 3.1.12]

3.1.25

get/read

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

3.1.26

input data

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

[IEC/TR 62390:2005, 3.1.14]

3.1.27

interface

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

[IEV 351-21-35, modified]

3.1.28

inverter

device that generally converts DC input to AC output

NOTE 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.

3.1.29

model

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

[IEV 351-21-36]

3.1.30

motion

any aspect of the dynamics of an axis

NOTE 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.

3.1.31

Motion Axis Object

object that defines the attributes, services, and behaviour of a motion device based axis according to the CIP Motion specification

3.1.32

Motion Control Axis Object

object that defines the attributes, services, and behaviour of a controller based axis according to the CIP Motion specification

3.1.33

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.

3.1.34
operating mode

characterisation of the way and the extent to which the human operator intervenes in the control equipment

[IEV 351-31-01]

3.1.35
output data

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

[IEC/TR 62390:2005, 3.1.21]

3.1.36
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 Adapted from IEC/TS 61915.

NOTE 2 A parameter is typically characterised by a parameter name, data type and access direction.

[IEC/TR 62390:2005, 3.1.22, modified]

3.1.37
profile

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

3.1.38
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 Service data includes service request messages to access Motion Axis Object attributes or perform various drive diagnostics.

3.1.39
set-point

(in the context of a device) value sent to the drive that is used to directly control some aspect of the motor dynamics, which includes (but is not limited to) position, velocity, acceleration, and torque

3.1.40
set/write

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

3.1.41
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

3.1.42
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 The different status information may be coded with one bit each.

3.1.43

synchronised

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

NOTE When synchronised, the drive and controller devices may utilise time stamps associated with CIP Motion connection data.

3.1.44

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 synchronised with a common master clock

NOTE 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.

3.1.45

thermal model

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

3.1.46

time offset

System Time offset value associated with the CIP Motion connection data that is associated with source device

NOTE 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

3.1.47

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

3.1.48

type

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

[IEC/TR 62390:2005, 3.1.25]

3.1.49

variable

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

[IEC/TR 62390:2005, 3.1.27]

NOTE The values of a variable as well as of a parameter are usually restricted to a certain data type.

3.1.50

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 Variable Frequency Drives are therefore sometimes referred to as a Volts/Hertz drives.

3.1.51

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

3.2 Abbreviated terms

Ack	Acknowledge
Act	Actual
Blk	Block
CIP™	Common Industrial Protocol (see IEC 61158 Type 2, IEC 61784-1 and IEC 61784-2 CPF2)
Cmd	Command
Cyc	Cyclic
HMI	Human Machine Interface
ID	Identifier
I/O	Input/Output
Opt.	Optional
PDS	Power Drive System
Req.	Required
RMS	Root mean square
RPM	Revolutions per minute
Sync'd	Synchronized
TCP	Transmission Control Protocol (see RFC 793)
UDP	User Datagram Protocol (see RFC 768)
VFD	Variable Frequency Drive

4 General

4.1 General

A CIP Motion drive device controls the motion of one or more drive axes, typically consisting of a rotary or linear motor actuator and one or more optional feedback devices. Each associated motor is driven by an associated drive power structure that is also part of the device. Motor control functionality supported by the CIP Motion drive device profile can be applied to “open loop” Variable Frequency Drives or “closed loop” Vector Controlled Servo Drives. In either case, axis 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 use of additional drive feedback channels as master feedback sources.

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

Applicable CIP Networks are DeviceNet™, ControlNet™, and EtherNet/IP™. EtherNet/IP is the network of choice for high performance, synchronised multi-axis control. ControlNet and DeviceNet would only be targeted for lower performance non-synchronised device applications such as simple variable frequency drives, velocity loop drives, and indexing drives.

4.2 Control modes

4.2.1 General

The Motion Axis Object covers the behavior of various motion control system devices that includes feedback devices and drive devices. For drive devices, the Motion Axis Object covers a wide range of drive types from simple variable frequency (V/Hz) drives, to sophisticated position control servo drives. Indeed, many commercial drive products have axes that can be configured to operate in any one of these different motion control modes depending on the specific application requirement.

The attributes of the Motion Axis Object are therefore organized to address this broad range of functionality, using the framework of “applications modes” defined in IEC 61800-7-1, and further detailed in the following subclauses.

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 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 behaviour 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.

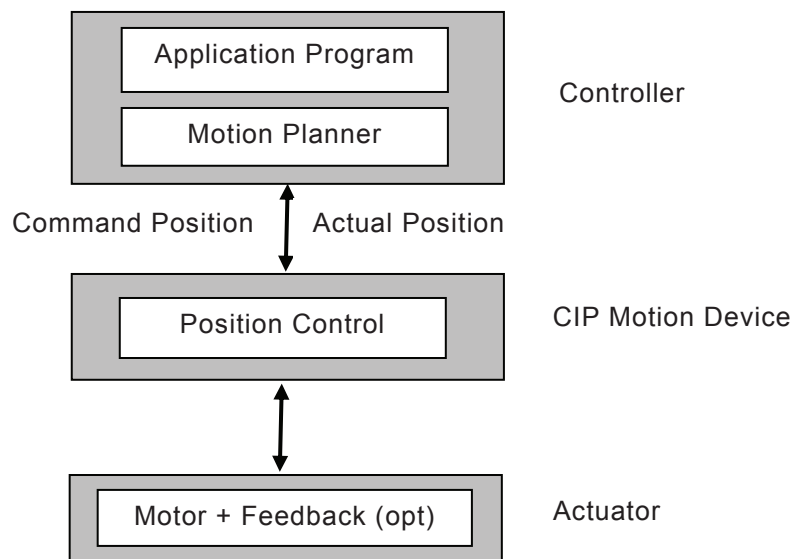


Figure 2 – Open loop position control

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 motion 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.

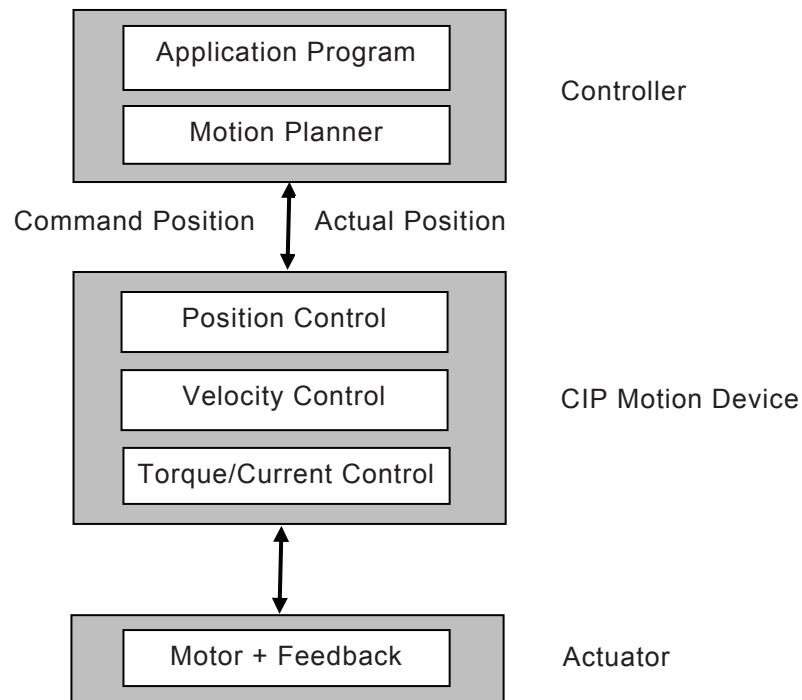


Figure 3 – Closed loop position control

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 motion 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.

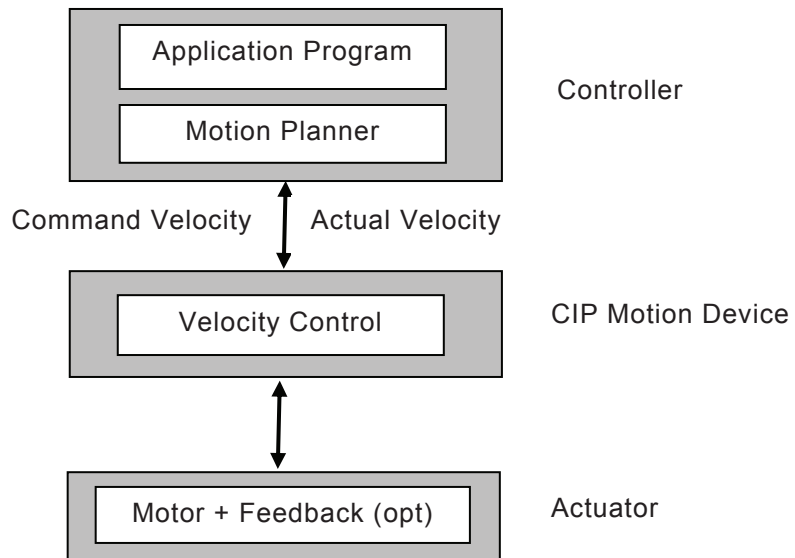


Figure 4 – Open loop velocity control

4.2.5.3 Closed Loop Velocity Control

A motion 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.

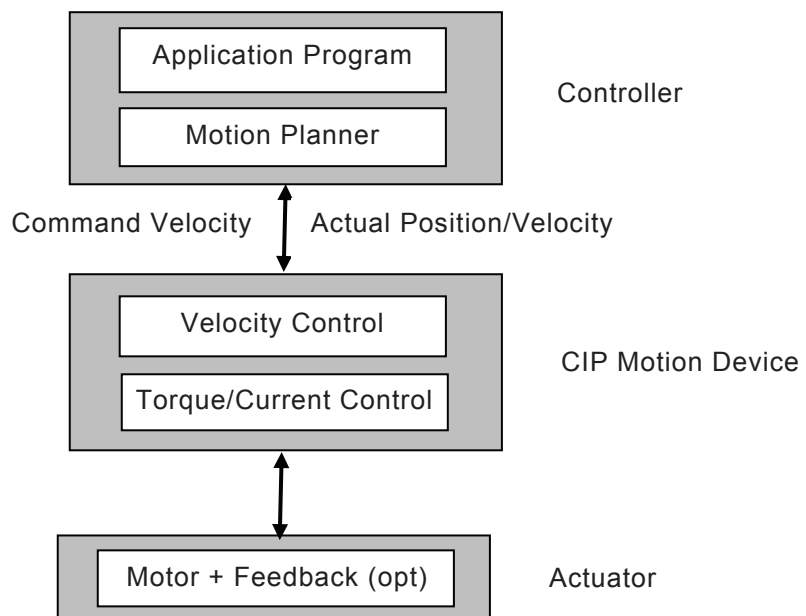


Figure 5 – Closed loop velocity control

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 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.

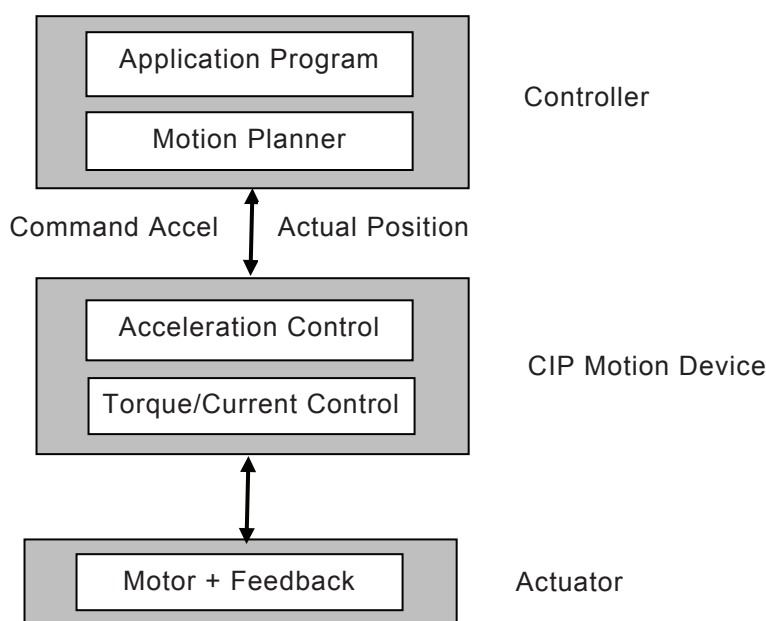


Figure 6 – Acceleration control

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, K_t , torque control is often synonymous with current control.

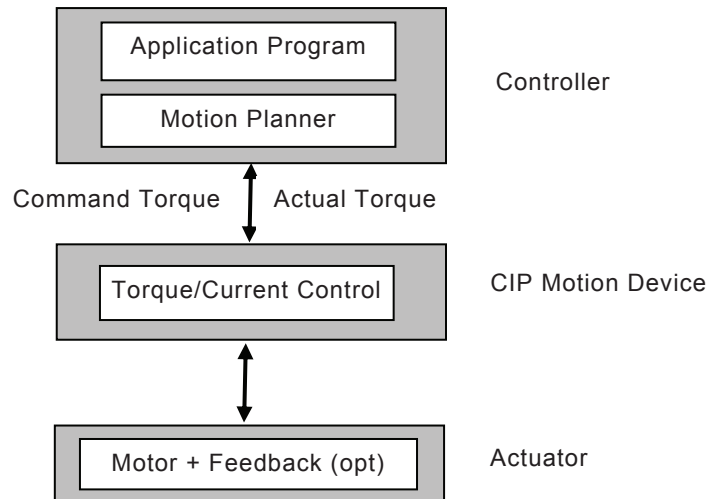


Figure 7 – Torque control

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 Axis Object supports a “No Control” application mode. This configuration, also referred to as “feedback only” or “master feedback”, can be of value when a particular feedback channel in the CIP Motion drive device is to serve simply as a master feedback source to the rest of the control system. This application mode can also be applied to CIP Motion feedback devices. In this “No Control” configuration, 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. The No Control drive configuration is illustrated in Figure 8.

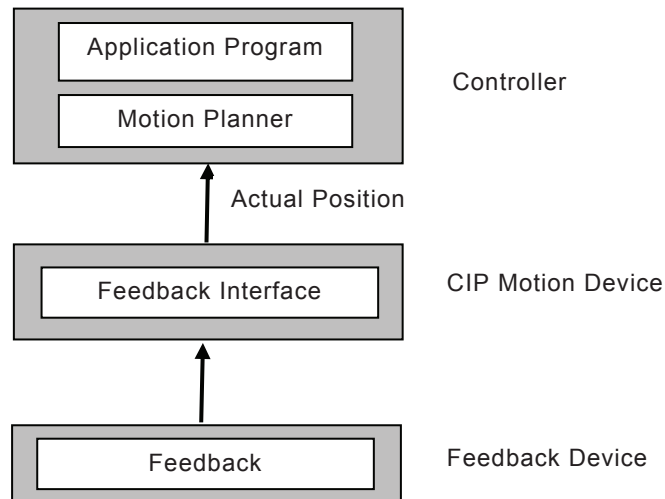


Figure 8 – No control

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)
SINT	Integer8 (see 5.3.1.4.2.2 of IEC 61158-5-2)
INT	Integer16 (see 5.3.1.4.2.4 of IEC 61158-5-2)
DINT	Integer32 (see 5.3.1.4.2.6 of IEC 61158-5-2)
LINT	Integer64 (see 5.3.1.4.2.8 of IEC 61158-5-2)
USINT	Unsigned8 (see 5.3.1.4.3.2 of IEC 61158-5-2)
UINT	Unsigned16 (see 5.3.1.4.3.4 of IEC 61158-5-2)
UDINT	Unsigned32 (see 5.3.1.4.3.6 of IEC 61158-5-2)
ULINT	Unsigned64 (see 5.3.1.4.3.8 of IEC 61158-5-2)
REAL	Float32 (see 5.3.1.4.1.2 of IEC 61158-5-2)
LREAL	Float64 (see 5.3.1.4.1.4 of IEC 61158-5-2)
SWORD/BYTE	Bitstring8 (see 5.3.1.2.2 of IEC 61158-5-2)
WORD	Bitstring16 (see 5.3.1.2.4 of IEC 61158-5-2)
DWORD	Bitstring32 (see 5.3.1.2.6 of IEC 61158-5-2)
LWORD	Bitstring64 (see 5.3.1.2.8 of IEC 61158-5-2)

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 $0xnn$, $0xn\text{nnn}$ (if their data type is specified).

6 CIP Drive Profile

6.1 Object model

6.1.1 Object overview

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

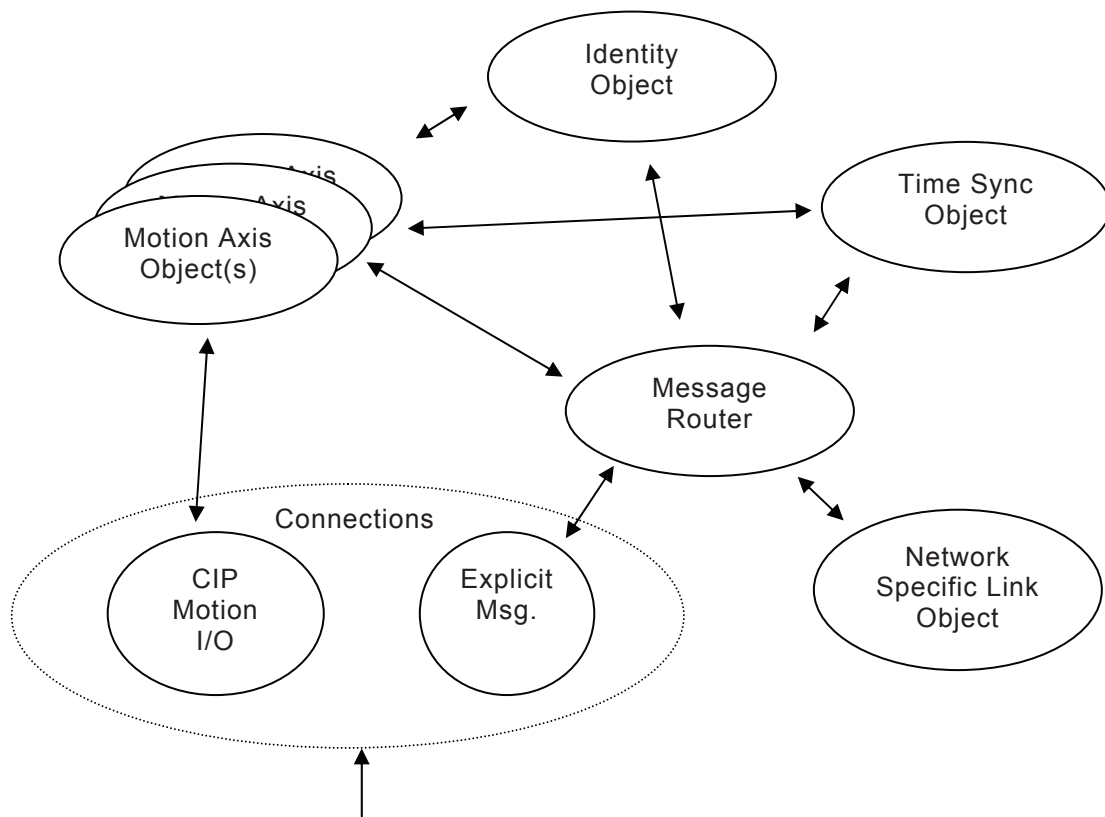


Figure 9 – Object Model for a CIP Motion drive

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 drive 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 Axis Object	Required	1 per axis
Time Sync Object	Optional	1

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 drive device profile defined as part of the CIP Motion extensions to CIP.

This object model also illustrates the use of multiple instances of a Motion Axis Object to implement a multi-axis drive device. Each Motion Axis Object instance governs the behavior of the associated drive axis. In this device profile, the term “drive axis” is synonymous with a “Motion Axis Object instance”.

A single bi-directional I/O connection to the Motion Axis Object class instance provides a cyclic data path between the controller and each individual Motion 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 or VFD drives, the Time Sync Object is not necessary for drive operation.

6.2 How objects affect behaviour

The objects for this device affect the device's behavior as shown in Table 3.

Table 3 – Object effect on behavior

Object class	Effect on behavior
CIP common required objects	See IEC 61158-5-2 and IEC 61158-6-2
Motion 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.

6.3 Defining object interfaces

Objects supported for the CIP Motion drive have the interfaces listed in Table 4.

Table 4 – Object interfaces

Object	Interface
CIP common required objects	See IEC 61158-5-2 and IEC 61158-6-2
Motion Axis Object	Message Router or Motion Axis Object class
Time Sync Object	Message Router or Motion Axis Object class

6.4 I/O Connection messages

6.4.1 General

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

The Motion Axis Object distributes the data in this connection to each instantiated Motion Axis Object instance. Therefore, the produced and consumed connection paths explicitly target Motion Axis Object class attributes (i.e. instance 0). The Consumed Connection path corresponds to the Controller-to-Device (C-to-D) connection, while the Produced Connection path corresponds to the Device-to-Controller (D-to-C) connection as detailed in the following subclauses.

The I/O Connection paths for this device are as follows:

Consumed Connection Path	=	Motion Axis Object class/Instance 0 / (Controller Consumed Connection Data attribute)
Produced Connection Path	=	Motion Axis Object class/Instance 0 / (Controller Produced Connection Data attribute)

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 drive, as shown in Figure 10.

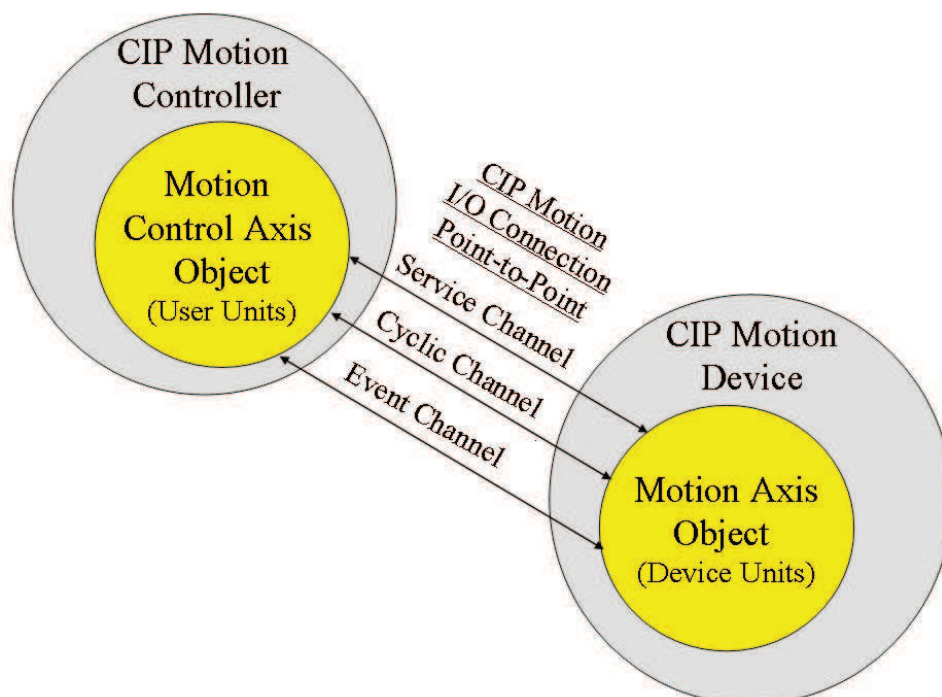


Figure 10 – CIP Motion I/O connection model

6.4.2.2 I/O connection formats

The Controller-to-Device connection format is shown in Figure 11 and the Device-to-Controller connection format is shown in Figure 12.

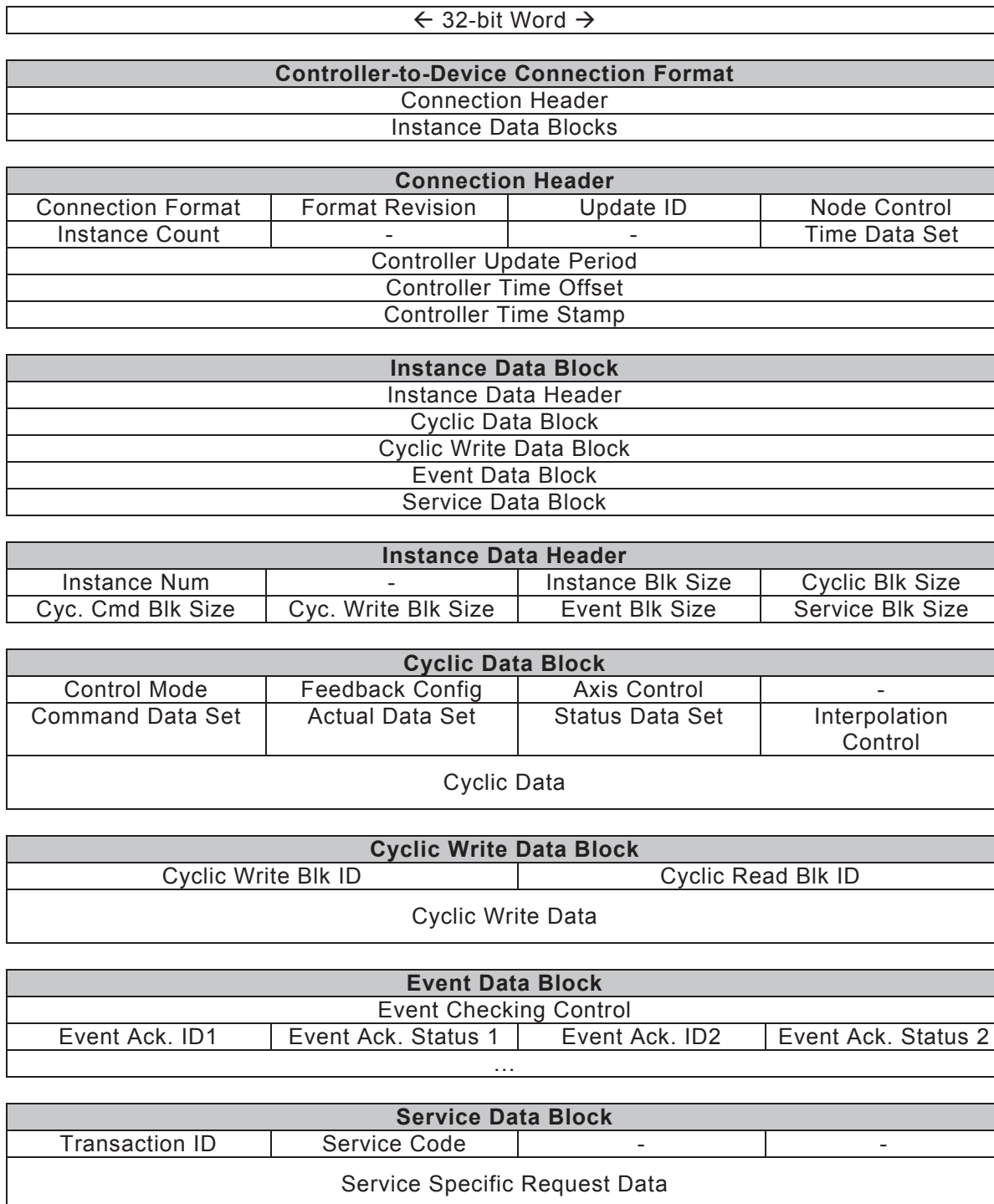


Figure 11 – Controller-to-Device Connection Format

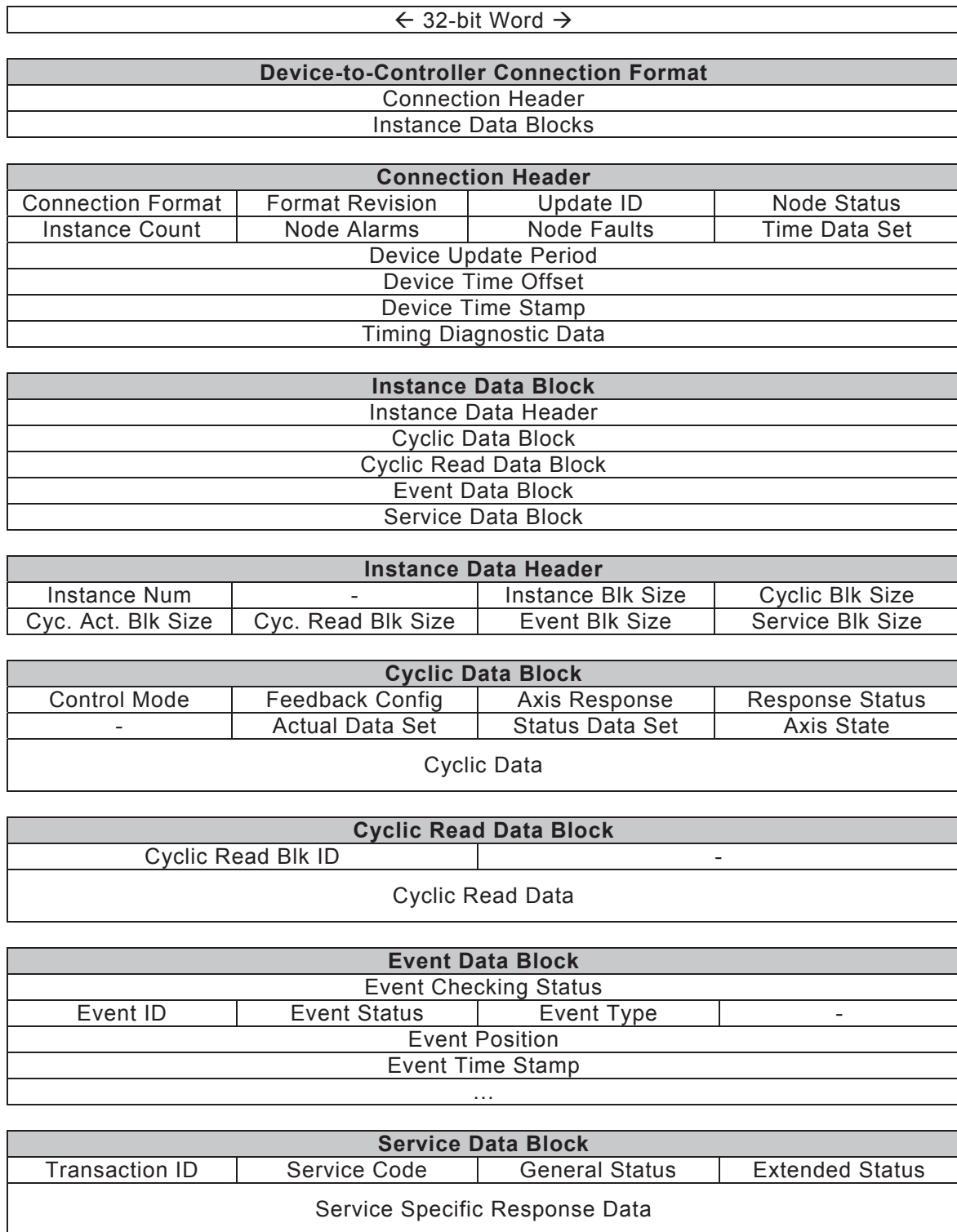


Figure 12 – Device-to-Controller Connection Format

6.4.2.3 CIP Motion connection structure

The CIP Motion I/O Connection actually consists of two unidirectional unicast connections; one passing data from the controller to the drive and the other passing data from the drive to the controller.

Both CIP Motion connection data structures (Controller-to-Device and Device-to-Controller) begin with a Connection Header that includes a 32-bit time stamp, followed by a series of data blocks for each axis instance supported by the drive node.

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. The size of these data blocks for a given update is variable and determined by the connection and instance data block headers. From the drive's perspective, these three distinct data channels have different data processing priorities as illustrated in Figure 13.

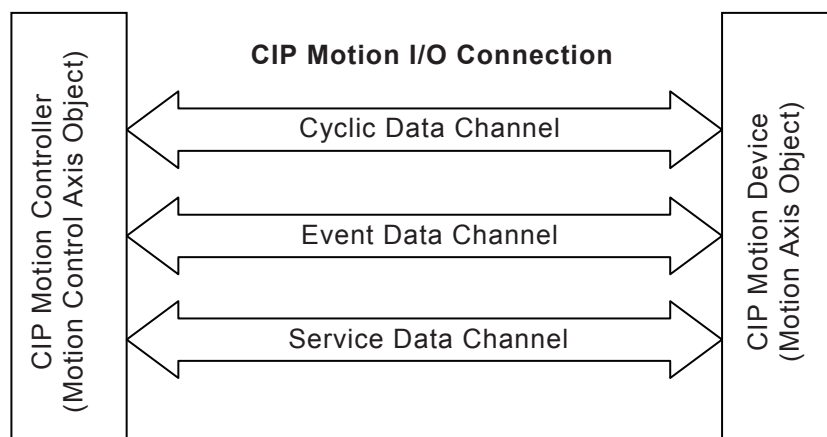


Figure 13 – CIP Motion I/O Connection Channels

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 synchronised 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 drive axis within one Device Update Period.
- **Event Data Channel** – carries event data associated with drive 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 Axis Object as part of on-line configuration and diagnostic functionality, as well as service requests to affect Motion Axis Object behaviour 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 drive data connection solution for industrial motion control.

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
Instance Data Blocks

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

Figure 15 – Connection Header

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

Bit 7	Connection Format		Bit 0
	(Reserved)	Connection Type	

Figure 16 – Connection Format

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

- 0 = Reserved
- 1 = Reserved
- 2 = Fixed Controller-to-Device Connection
- 3 = Fixed Device-to-Controller Connection
- 4 = Reserved
- 5 = Reserved
- 6 = Variable Controller-to-Device Connection
- 7 = Variable Device-to-Controller Connection
- 8 to 15 = Reserved.

- Format Revision: This edition of IEC 61800-7-202 defines Format Revision 1. Devices utilizing this edition of IEC 61800-7-202 will only recognize Format Revision 1. 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 drives to support the connection formats generated by an older controller. It also allows older drives to recognise a newer connection format from a controller that it cannot support and to generate a Node Fault Error Code 4 (Data Format Error).
- 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 drive to determine whether the connection buffer contains fresh data. If the Update ID has not changed, the drive attempts to ride through the missed update by extrapolation based on previous trajectory until fresh data arrives. In the case where the drive axis is not synchronised, or does not support time synchronisation services, the time stamp data is either not included or invalid, so the Update ID is the only way for the drive 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.
- Node Control: This element is applied to the Motion Axis Object class attribute "Node Control", which is used to control the state of the associated drive communications node. See 7.2.2.3 (Motion 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 drive addressing and time stamping, as shown in Figure 17.

Connection Header			
<i>Connection Format</i>	<i>Format Revision</i>	<i>Update ID</i>	<i>Node Control</i>
Instance Count	-	-	Time Data Set
Controller Update Period			
Controller Time Offset			
Controller Time Stamp			

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.
- Time Data Set: This bit-mapped byte contains flags that determine the usage and format of the controller and drive timing information, as specified in Table 5. The Time Data Set value sent by the controller in the Controller-to-Device Connection becomes the Time Data Set value sent by the drive in the Device-to-Controller Connection. The value of this element is transferred to the Motion Axis Object attribute “Command Data Set”.

Table 5 – Time Data Set

Bit	Definition	Syntax
0	Update Period	0 = no update period 1 = update period included
1	Time Stamp	0 = no time stamp and offset 1 = time stamp and offset included
2	Time Diagnostic Data	0 = no diagnostic data 1 = diagnostic data included
3 to 7	(Reserved)	

- Controller Update Period: This 32-bit element specifies the update period of the Controller-to-Device Connection in nanoseconds. The value of this element is transferred to the Motion Axis Object class attribute of the same name. When the connection is synchronised, the Controller Update Period can be used by the drive along with the Controller Time Stamp to check for missed or late updates. If the difference between the last Controller Time Stamp and the current local drive time stamp exceeds the maximum value given by,

$$\text{Max Delay} = \text{Controller Update Delay High Limit} \times \text{Controller Update Period},$$
the Controller Update Fault bit is set in the Node Faults class attribute of the Motion 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 drive time stamp exceeds the maximum value given by,

$$\text{Max Delay} = \text{Controller Update Delay Low Limit} \times \text{Controller Update Period},$$
the Controller Update Alarm bit is set in the Node Alarm class attribute of the Motion Axis Object. When the connection is synchronised and fine interpolation is enabled, the Controller Update Period is also used by the drive along with the Controller Time Stamp value to compute new coefficients to the interpolation/extrapolation polynomial.

In the case where the connection is not synchronised, no late update checks are performed by the drive. In this case, the command data should be applied immediately, which means applying the data immediately as a command reference.

- **Controller Time Offset:** This element 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 Axis Object class attribute of the same name. The Controller Time Offset value is used by the drive to determine if System Time as defined in the controller is skewed relative to System Time in the drive. Normally, System Time between the drive and the controller are closely matched at any given time. But every few seconds, according to the IEC 61588:2004 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 drive are running with skewed System Time values. The Controller Time Offset value can be used together with the System Time Offset for the drive to both determine if System Time between the drive and the controller is skewed and if so, to compensate for the skew.
- **Controller Time Stamp:** 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. The Controller Time Stamp is therefore directly associated with the command data contained in the connection. The value of this element is transferred to the Motion Axis Object class attribute of the same name. Taken together with the Controller Update Period and the Command Target Time (discussed later), the drive has all the information it needs compute command interpolation/extrapolation polynomials to correct command data values for differences between the drive 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 drive updates are phase shifted relative to the controller.

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

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 drive applications. Information within the header can be used by the drive 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	(Reserved)	Instance Blk Size	Cyclic Blk Size
Cyc. Act. Blk Size	Cyc. Write Blk Size	Event Blk Size	Service Blk Size

Figure 19 – Instance Data Header

- **Instance Number:** This is the number that identifies the specific Motion Axis Object instance the following instance data block applies to. Motion 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. 1, 2, 3, etc. (instance 0 is defined as the class instance and not generally used as part of the connection data). Thus, in theory, up to 255 instances can be serviced by the CIP Motion I/O 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 drive wants to directly access the next instance data block without having to add the sizes of the cyclic, event, and service blocks.
- **Cyclic Block Size:** This value represents the size to the Cyclic Data Block in 32-bit word units including the header.
- **Cyclic Command Block Size:** This value represents the size to the Cyclic Command Data Block in 32-bit word units including the header.
- **Cyclic Write Block Size:** This value represents the size to the Cyclic Write Data Block in 32-bit word units including the header. If the Cyclic Write Block Size is 0, the drive does not need to support the Cyclic Read/Write functionality.
- **Event Block Size:** This value represents the size to the Event Data Block in 32-bit word units including the header. If the Event Block Size is 0, the drive does not need to support the event functionality.
- **Service Block Size:** This value represents the size to 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.

6.4.3.3.3 Cyclic data block

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 Configuration (see Figure 20). The header also provides a mechanism to control the state of the targeted drive axis.

Cyclic Data Header			
Control Mode	Feedback Config	Axis Control	-
Command Data Set	Actual Data Set	Status Data Set	Interpolation Control
Cyclic Data			

Figure 20 – Cyclic Data Header

- **Control Mode:** This 8-bit enumerated value determines the control mode context of the command data as presently configured in the motion controller. This value can be changed while on-line and even while the drive axis is in the Running state. If a particular Control Mode transition is not supported by the drive, an Illegal Mode Change 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 Axis Object attribute of the same name. The complete Control Mode attribute definition can be found in the attribute tables of the Motion Axis Object (see 7.3.2.2.1).

- **Feedback Configuration:** This 8-bit enumerated value indicates the feedback context of the command data as presently configured in the motion controller. Command 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 even while the drive axis is in the Running state. If a particular Feedback Configuration transition is not supported by the drive, an Illegal Mode Change 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 Axis Object attribute of the same name. The complete Feedback Configuration attribute definition can be found in the attribute tables of the Motion Axis Object (see 7.3.4.2).

- **Axis Control:** This 8-bit enumerated code can be used to directly initiate drive state change operations that do not require either passing or returning drive parameters, and therefore, do not require a CIP service to initiate. Valid enumerations for this data element are shown in Table 6.

Table 6 – 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 to 127	(Reserved)
128 to 255	(Vendor Specific)

- **Command Data Set:** This bit mapped value has a bit defined for each possible real-time command reference (see Table 7). Command data appears in the same order in the Command Data Set as the bit numbers, so Command Position 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 Axis Object attribute “Command Data Set”.

Table 7 – Command Data Set

Bit	Controller Command Data Element	Data Type
0	Command Position	DINT
1	Command Velocity	REAL
2	Command Acceleration	REAL
3	Command Torque	REAL
4	Position Trim	DINT
5	Velocity Trim	REAL
6	Acceleration Trim	REAL
7	Torque Trim	REAL

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 Axis Object as shown in Table 8.

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

Table 8 – Command Data Element to Motion Axis Object Attribute mapping

Bit	Command Data Element	Motion Axis Object Attribute
0	Command Position	Controller Position Command
1	Command Velocity	Controller Velocity Command
2	Command Acceleration	Controller Acceleration Command
3	Command Torque	Controller Torque Command
4	Position Trim	Position Trim
5	Velocity Trim	Velocity Trim
6	Acceleration Trim	Acceleration Trim
7	Torque Trim	Torque Trim

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 9). Actual data appears in the same order as the bit numbers, so Actual Position would appear before Actual Torque in the Actual Data Set structure. Using this mechanism, the contents of the Actual Data Set may be changed at any time during drive operation. The value of this element is transferred to the Motion Axis Object attribute "Feedback Data Set".

Table 9 – Actual Data Set

Bit	Actual Data Element Produced	Data Type
0	Actual Position	DINT
1	Actual Velocity	REAL
2	Actual Acceleration	REAL
3	Actual Torque	REAL
4	Actual Current	REAL
5	Actual Voltage	REAL
6	Actual Frequency	REAL
7	(Reserved)	

The above Actual Data Elements map to corresponding attributes in the Motion Axis Object as shown in Table 10. The units of the Actual Data Elements match the units defined for the associated Motion Axis Object attribute.

Table 10 – Actual Data Element to Motion Axis Object Attribute Mapping

Bit	Actual Data Element	Motion Axis Object Attribute
0	Actual Position	Position Feedback
1	Actual Velocity	Velocity Feedback
2	Actual Acceleration	Acceleration Feedback
3	Actual Torque	Torque Reference
4	Actual Current	Iq Current Reference
5	Actual Voltage	Output Voltage
6	Actual Frequency	Output Frequency
7	(Reserved)	

- **Status Data Set:** This bit-mapped byte contains flags that determine the contents of the Status Data Set of the Device-to-Controller Connection's Instance Data Block in the next update (see Table 11). Status data appears in the same order as the bit numbers, so Axis Fault Status would appear before, for example, Axis Alarm Status 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 Axis Fault Code is transmitted before Mfg Axis Fault Code. The definitions of each of these Status Data Elements can be found by looking up the corresponding Motion Axis Object attribute (see 7.3). The value of this element is transferred to the Motion Axis Object attribute "Status Data Set".

Table 11 – Status Data Set configuration

Bit	Status Data Element Produced	Data Type
0	Initialization Fault Code Initialization Fault Code – Mfg Axis Fault Code Axis Fault Code – Mfg Axis Alarm Code Axis Alarm Code – Mfg Start Inhibit Code Start Inhibit Code – Mfg Axis Fault Time Stamp	USINT USINT USINT USINT USINT USINT USINT USINT LINT
1	Axis Fault Status Axis Fault Status - Mfg	LWORD LWORD
2	Axis Alarm Status Axis Alarm Status - Mfg	LWORD LWORD
3	Start Inhibit Status Start Inhibit Status - Mfg	WORD WORD
4	Axis Status Axis Status - Mfg Axis I/O Status Axis I/O Status - Mfg	DWORD DWORD DWORD DWORD
5	(Reserved)	
6	(Reserved)	
7	(Vendor specific)	

- **Interpolation Control:** The byte contains information needed to control the fine interpolation algorithm and determine the target time of the command data to the drive control structure (see Table 12). The value of this element is transferred to the Motion Axis Object attribute of the same name.

Table 12 – Interpolation Control

Bit	Definition	Syntax
0 to 1	Command Target Time	0 = Immediate 1 = Extrapolate (+1 Update Period) 2 = Interpolate (+2 Update Periods)
3 to 5	(reserved)	
6 to 7	(vendor specific)	

Command Target Time – 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 time is the sum of the Controller Time Stamp from the controller and the product, Command Target Time x Controller Update Period.

- A Command Target Time 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 Time 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 Time 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.
- **Cyclic Command Data:** The Cyclic Command Data contains high priority data that needs to be applied to the associated drive axis during the next drive update. This block consists of command data elements that are applied as references to the drive'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 Axis Object configuration parameters within the drive. 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 drive update cycles to process, the Cyclic Write Data mechanism guarantees that the targeted parameter is applied at the next available drive update.

The Cyclic Write Data Block is only supported in the Variable Connection format (see Figure 21).

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

Figure 21 – Cyclic Write Data Block

The associated header for this block contains key elements related to the content of the Cyclic Write Data Block.

- **Cyclic Write Block ID:** This 16-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. A successful response to this service includes a new Cyclic Write Blk ID that can be used in the next connection update to pass cyclic data in this format. The previous Cyclic Write Blk ID and its associated structure shall be maintained until the controller requests transmission of the new Cyclic Write Blk ID at which point the old Cyclic Write Blk ID and structure definition are no longer required.
- **Cyclic Read Block ID:** This 16-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. A successful response to this service includes a new Cyclic Read Blk ID that can be used in the next connection update to allow the drive 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 drive axis during the next drive update. This block consists of parametric data elements that are applied to Motion 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 Blk ID found in the Cyclic Write Data Header.

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 drive as well as acknowledge receipt of event notifications from the drive 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 22.

Event Data Block			
Event Checking Control			
Event Ack. ID1	Event Ack. Status 1	Event Ack. ID2	Event Ack. Status 2
...			

Figure 22 – Event Data Block

- **Event Checking Control:** This 32-bit word is copied into the Motion Axis Object attribute of the same name that is used to enable various drive inputs, for example marker and registration inputs, to generate events. When these events occur, the drive captures both the time and exact axis position when the event occurred. The last 4 bits of the Event Checking Control is a binary value that specifies the number of active events, which is literally the number of Event Acknowledge IDs listed in this Event Data Block. A complete definition for the Event Checking Control attribute can be found in 7.3.5.2.1.

The Event Control mechanism works as follows:

- 1) Controller sets the appropriate Event Checking 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.
- 2) Drive detects that the Event Checking Control request bit is set and initiates the requested event checking action.
- 3) Drive sets the corresponding Event Checking Status bit to acknowledge that the drive is now actively checking for the specified event condition.
- 4) Controller sees that the Event Checking Status bit from the drive is set indicating that specified event trigger is "armed" at the drive. Starting with this update, the controller can process any event notifications from the drive that match the specified event condition.
- 5) Drive detects the specified event condition, increments the Event ID, and sends an Event Notification Data Block with this ID to the controller. If the associated Auto-rearm bit is clear in the Event Checking Control word, the drive discontinues checking for the specified event. However, the Event Checking Status bit remains set until the Event

Checking Request bit is cleared by the controller. Notifications for other events can be sent in the same update.

- 6) Controller processes the Event Notification, sends the Event Acknowledge to the drive and, if Auto-rearm is not enabled, clears the associated Event Checking bit for the next Controller-to-Device Connection update.
- 7) Drive detects that the Event Checking Request bit is clear and clears the corresponding Event Checking Status bit prior to the next Device-to-Controller Connection update.
- 8) Controller is able to set the Event Checking bit again as soon as it sees the associated Event Checking Status bit has been reset. Therefore, the minimum time between when the Event Notification was received by the controller to the time when the drive is rearmed for the next event is one Controller Update Period.

In the case of a Registration event where Auto-rearm Event Checking is requested, the event handling sequence would be as follows:

- 1) Controller sets the appropriate Event Checking bit to look for a specific event condition to occur and also sets the Auto-rearm bit for that event condition.
 - 2) Drive detects that the Event Checking Control request bit is set and initiates the requested event checking action.
 - 3) Drive sets the corresponding Event Checking Status and Auto-rearm bits to acknowledge that the drive is now actively checking for the specified event condition.
 - 4) Controller sees that the Event Checking Status bit from the drive is set indicating that specified event trigger is “armed” at the drive. Starting with this update, the controller can process any event notifications from the drive that match the specified event condition.
 - 5) Drive detects the specified event condition, increments the Event ID, and sends an Event Notification Data Block with this ID to the controller. Since the corresponding Auto-rearm bit is set, the drive continues checking for the specified event. The Event Checking Status bit remains set until the Event Checking Request bit is cleared by the controller. If the Drive 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. Since the corresponding Auto-rearm bit is set, the drive continues checking for the specified event.
 - 6) Controller processes the Event Notification, sends the Event Acknowledge to the drive but, since the Auto-rearm bit is enabled, it leaves the associated Event Checking bit set for the next Controller-to-Device Connection update.
 - 7) Drive receives the Event Acknowledge from the controller. It can now send another Event Notification if the specified event has occurred again. Otherwise, go to the next step.
 - 8) Drive continues checking for the specified event condition and sends an Event Notification Data Block to the controller whenever the event condition occurs. The Event Checking Status bit remains set until the Event Checking Request bit is cleared by the controller. With the Auto-rearm feature, event checking is continuously enabled, ensuring 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 later case is how Windowed Registration functionality is implemented; the Windowed Registration feature checks each registration event that arrives from the drive 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 discarded. If the computed registration position is within the window, the registration position and registration time stamp is stored in the controller.
- 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 drive. Using the Event ID, the drive 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 drive in a given update. The service request requires a specific service response from the drive 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 drive.

Each service request is represented by a block of data organized as shown in Figure 23.

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			

Figure 23 – 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 Controller-to-Device Connection is organised as shown in Figure 24.

CIP Motion Controller-to-Device Connection Format
Connection Header
Instance Data Blocks

Figure 24 – CIP Motion Controller-to-Device 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 25.

Connection Header			
Connection Format	Format Revision	Update ID	Node Status

Figure 25 – 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 = Reserved
 - 1 = Reserved
 - 2 = Fixed Controller-to-Device Connection
 - 3 = Fixed Device-to-Controller Connection
 - 4 = Reserved
 - 5 = Reserved
 - 6 = Variable Controller-to-Device Connection
 - 7 = Variable Device-to-Controller Connection
 - 8 to 15 = Reserved.
- **Format Revision:** The Format Revision number is 1 for any device that utilizes this edition of IEC 61800-7-202. This value is to be 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 drives. It also allows older controllers to recognize a newer connection format from a drive that it cannot support and generate an appropriate error to its application.
- **Update ID:** The Device-to-Controller Connection Update ID should match the Update ID of the preceding Controller-to-Device Update ID and therefore shall be incremented every update period. 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 control system to ride through a lost or missed Controller-to-Device packet and maintain synchronisation 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 has not changed, the controller attempts to ride through the missed update by extrapolation based on the previous trajectory until a fresh update arrives. In the case where the axis is not synchronised, or does not support time synchronisation services, the time stamp data is either not included or invalid, so no ride through extrapolation is attempted. In that case, the Update ID is also the only way for the controller to detect new connection data.
- **Node Status:** Contains bits used to indicate the status of the associated drive device communications node. The value of this element is derived from the Motion Axis Object class attribute of the same name. See 7.2.2.4 (Motion 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 drive addressing and time stamping (see Figure 26).

Connection Header			
Connection Format	Format Revision	Update ID	Node Control
Instance Count	Node Alarms	Node Faults	Time Data Set
Device Update Period			
Device Time Offset			
Device Time Stamp			
Data Received Time Stamp			
Data Transmit Time Stamp			
Sync Variance			

Figure 26 – Connection Header

- Instance Count: Same as Controller-to-Device definition.
- Node Alarms: Contains bits used to indicate the alarm conditions of the associated drive communications node. The value of this element is derived from the Motion Axis Object class attribute of the same name. See 7.2.2.6 (Motion Axis Object) for details.
- Node Faults: Contains bits used to indicate the fault conditions of the associated drive communications node. The value of this element is derived from the Motion Axis Object class attribute of the same name. See 7.2.2.5 (Motion Axis Object) for details.
- Time Data Set: Same as Controller-to-Device definition.
- Device Update Period: This element represents the current period between drive control calculations that apply the command data. This value is used by the controller to configure the behaviour of the fine interpolation blocks that are applied to the command data. If the Controller Update Period is much longer than the Device Update Period, fine interpolation is generally applied. When the Controller Update Period is comparable to the Device Update Period, fine interpolation is unnecessary and only extrapolation is required to perform time stamp corrections.
- Device Time Offset: This 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 drive is skewed relative to System Time in the controller. Normally, System Time between the drive and the controller are closely matched at any given time. But every few seconds, according to the IEC 61588:2004 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 drive 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 drive and the controller is skewed and if so, to compensate for the skew.
- Device Time Stamp: This time stamp value represents the 64-bit System Time value, in nanoseconds, when the drive's update timer event that is associated with the actual data in the connection structure occurred, for example when the actual data was captured. It is calculated by the drive as the sum of the drive's local clock value when update timer event occurred and the drive'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 drive and controller update timing that result when the Controller Update Period is not an integer multiple of the Device Update Period or when the drive 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 drive 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 27 illustrates how axis position data from the drive is adjusted by the controller based on the relative time stamps between the drive and the controller.

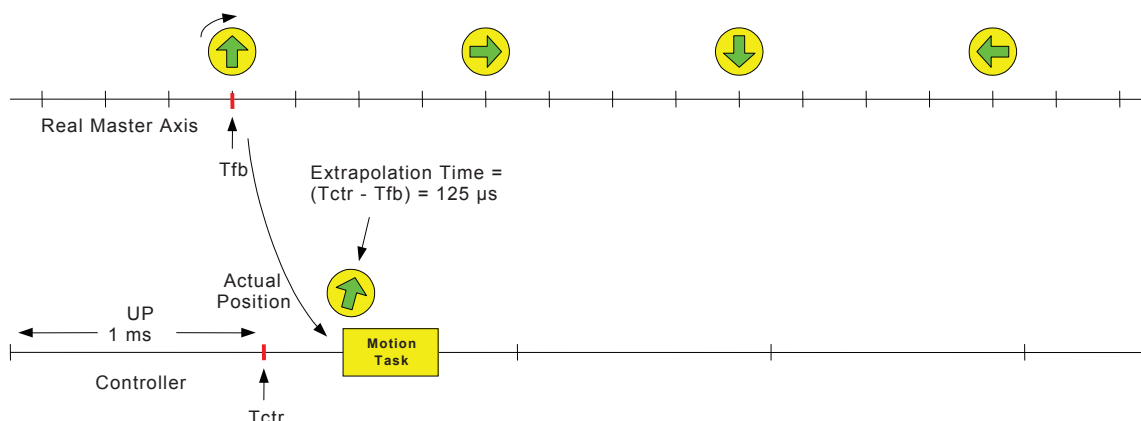


Figure 27 – Adjustment of actual position data based on device time stamp

- **Data Received Time Stamp:** This element represents the 64-bit System Time value at the moment that the last Controller-to-Device Connection data arrived at the drive device and was ready for processing. 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 represents the 64-bit System Time value at the moment that the Device-to-Controller Connection data is transmitted 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.
- **Sync Variance:** This element represents the current statistical variation of the System Time Offset value from the mean System Time Offset value.

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 28.

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

Figure 28 – 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 29). This header is only included in the Variable Connection format to accommodate multi-axis drive applications. Information within the header can be used by the drive 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.