TECHNICAL SPECIFICATION

ISO/TS 21815-2

First edition
2021-07

Earth-moving machinery — Collision warning and avoidance —

Part 2: On-board J1939 communication interface

Engins de terrassement — Avertissement et évitement de collision —

Partie 2: Interface de communication embarquée

Reference number
ISO/TS 21815-2:2021(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 127, *Earth-moving machinery*, Subcommittee SC 2, *Safety, ergonomics and general requirements*.

A list of all parts in the ISO 21815 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The increasing use of detection systems and avoidance technology has been supporting operators to safely operate machines in the field of mining and construction. At the same time, there are demands to set standards for machines and systems detecting, alerting and intervening to mitigate collision risk.

There are currently two existing standards in the field: ISO 16001 and ISO 17757. These standards provide guidance for visibility aids and object detection systems and for autonomous and semi-autonomous machines, however, there is currently no standard that describes collision risk awareness, warning signals and collision avoidance actions of the machinery operated by humans where there is a risk of collision.

Collision warning and avoidance systems are developing technologies; and the algorithms are not yet mature and well understood. This document is intended to foster innovation and accelerate the pace of improvements in new collision warning and avoidance technologies. The performance requirements of this document are technology neutral and do not specify technologies to meet the requirements.

The systems described in this document are intended to assist the operator of the machine. As current technologies are unable to achieve full collision warning/avoidance in every situation, the responsibility for safe operation of the machine remains with the operator of the machine.

This document defines a protocol for communication between a machine and a connected device to allow the connected device to command the machine to slow down, stop or to maintain a stationary state where the machine can move in a linear (i.e. forwards-backwards) direction along a travel path. Machines with rotational movements (e.g. excavators) and machines with compound movements (e.g. machines with booms) are only considered to the extent of the linear component of their travel.

The machine manufacturer may be flexible in deciding which method is most appropriate for their machine. Some applications can be delivered with basic functionality (e.g. without the use of registers). Regardless of which approach is selected, the connected device has a means to discover the capabilities of the machine.

Annex B outlines a mechanism for establishing trust between the machine and the connected device based on the exchange of certificates at the session layer as defined by the machine manufacturer. The message structure for the session layer can be different to the message structure defined in this document.

The specification of the J1939 protocol in this document does not preclude the development of other communication interfaces that can support collision warning and avoidance functionality. At the time of publishing this document, protocols have only been defined for SAE J1939 due to the general availability of CAN 2.0 interfaces on machinery and devices providing collision warning and avoidance functions.

Earth-moving machinery — Collision warning and avoidance —

Part 2: On-board J1939 communication interface

1 Scope

This document describes the on-board J1939 communication interface between a connected device and mobile machines for use in earth-moving, mining and road construction applications to enable interventional collision avoidance actions defined in ISO 21815-1 based on the SAE J1939 protocol. This interface is intended for use by a collision avoidance system (CAS) device integrated independently from the original machine providing intervention signals to slow down, stop or prevent motion of the machine. The protocol defined by this document can also be used to provide input information for a collision warning system (CWS).

This document is not intended for plug-and-play implementation of CAS or CWS on the machine. Additional details not fully described in this document can be negotiated by the CAS or CWS manufacturer and the machine manufacturer to enable functionality.

This document does not preclude the possibility of the machine manufacturer or the CxD manufacturer developing alternative on-board communication interfaces.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

ISO 19014-3, *Earth-moving machinery — Functional safety — Part 3: Environmental performance and test requirements of electronic and electrical components used in safety-related parts of the control system*

SAE J1939-15, *Reduced Physical Layer, 250 kbits/sec, UN-Shielded Twisted Pair (UTP)*

3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
collision warning system
CWS
system which detects intended objects in the collision risk area, evaluates the collision risk level and provides a warning to the operator

[SOURCE: ISO 21815-1:—, 3.8]

3.2
collision avoidance system
CAS
system which detects intended objects in the collision risk area, evaluates the collision risk level and provides interventional collision avoidance *action* (3.9)

[SOURCE: ISO 21815-1:—, 3.9]

3.3
CxS
CWS (3.1) or *CAS* (3.2) or both

[SOURCE: ISO 21815-1:—, 3.10]

3.4
CxS device
CxD
proximity detection system
device with sensors providing *CxS* (3.3) functions to detect objects in the proximity of the machine, assess the collision risk level, warn the operator of the presence of object(s) for *CWS* (3.1), and/or provide signals to the machine control system to initiate the appropriate interventional collision avoidance *action* (3.9) on the machine for *CAS* (3.2)

Note 1 to entry: Proximity detection system (PDS) is a colloquial industry term for a physical device providing CWS or CAS functionality.

3.5
on-board communication interface
bi-directional connection between a *CxD* (3.4) and the machine in a *CWS* (3.1) or *CAS* (3.2)

Note 1 to entry: The *CxS* (3.3) may utilise information sent from the machine via the on-board communication interface to improve the estimation of the collision risk level. Only a CAS can initiate interventional collision avoidance *action* (3.9) over the on-board communication interface.

3.6
register
storage location on the machine side of the *on-board communication interface* (3.5) that may be read and optionally written to by the *CxD* (3.4)

Note 1 to entry: Changing the value of a register does not immediately initiate an interventional collision avoidance *action* (3.9).

3.7
parameter
type of *register* (3.6) that is used to store configuration information

EXAMPLE Software revision, timeout, max speed for emergency stop.

3.8
setpoint
type of *register* (3.6) that is used by the machine to respond to an interventional collision avoidance *action* (3.9)

EXAMPLE Minimum braking, maximum throttle, maximum speed.

3.9
action
message sent from the *CxD* (3.4) to the machine to change an internal *register* (3.6), or to initiate a machine function

EXAMPLE Reduce speed, apply brakes, inhibit motion.

3.10
enquiry
message sent from the *CxD* (3.4) to the machine to read an internal *register* (3.6), or request a machine capabilityEXAMPLE Slow down, emergency stop, controlled stop.

3.11
instruction
action (3.9) or *enquiry* (3.10) issued by the *CxD* (3.4)

3.12
reply
response (3.35) of the machine to an *instruction* (3.11) from the *CxD* (3.4)

3.13
logical group
grouping of related information or *instruction* (3.11) elements into a coherent message, sent from the *CxD* (3.4) to the machine or from the machine to the CxD over the *on-board communication interface* (3.5)

3.14
+bat
system voltage of the machine as defined by the machine manufacturer

Note 1 to entry: Typical voltages are 12 V or 24 V DC.

3.15
key switch
device used by the operator to turn on or turn off the machine

3.16
isolator switch
disconnect switch
device used by the operator to isolate the batteries or electrical supply to the machine

3.17
+bat(switched)
machine system level voltage that is turned on or off through the *key switch* (3.15)

3.18
+bat(un-switched)
voltage that is not affected by the state of the *key switch* (3.15), but is affected by the *isolator switch* (3.16)

3.19
CxD harness
auxiliary wiring between the machine connector and the *CxD* (3.4) connector

3.20
CxD bus
CAN-bus communication path between the machine and the *CxD* (3.4) terminated by 120 Ohm resistors at each end

3.21
CxD branch
machine-to-CxD bus or CxD-to-CxD bus wiring connection

3.22
PowerOn()
startup sequence for the machine that enables *CxD* (3.4) operation

3.23
doNegotiation()
automatic or semi-automatic process that verifies the credentials of the *CxD* (3.4) attached to the machine and returns permissions for the CxD to send commands to the machine and receive machine information

3.24
enableTimeout()
automatic process that enables an automatic timer that counts down to zero and sets a flag that there has been a communications error on the J1939 *on-board communication interface* (3.5)

3.25
resetTimeout()
automatic process that resets the automatic timer and clears the error flag indicating that the J1939 *on-board communication interface* (3.5) is functioning normally

3.26
doEmergencyStop()
automatic process that initiates an emergency stop on the machine

3.27
doControlledStop()
automatic process that initiates a controlled stop on the machine

3.28
doSlowDown()
automatic process that slows down the machine

3.29
doStandDown()
automatic process that brings the machine to a halted state

3.30
doBypassPropulsion()
instruction to bypass the propulsion system

3.31
doApplyPropulsionSetpoints()
activation of braking, throttle, speed *setpoint* (3.8) *registers* (3.6)

3.32
doMotionInhibit()
automatic process that prevents a machine from moving while stationary

3.33
doNormalOperation()
instruction (3.11) for the machine to continue with normal operation or return to normal operation

Note 1 to entry: This instruction has the effect of cancelling any other interventional collision avoidance *action* (3.9) that is already in progress.

3.34
challenge
unique message issued by the machine to the *CxD* (3.4) to which a valid *reply* (3.12) is expected

3.35
response
reply (3.12) by the *CxD* (3.4) to the *challenge* (3.34) issued by the machine to establish trust

4 Symbols and abbreviated terms

PGN	parameter group number (see SAE J1939)
SPN	suspect parameter number (see SAE J1939)
0xHH	8-bit hexadecimal value in the range 0x00 to 0xFF
0bB	1-bit binary value in the range 0b0 to 0b1
0bBB…	N-bit binary value

5 Logical interface

5.1 Logical groups

The logical connections for the on-board J1939 communication interface between the CxD and the machine are shown in Figure 1.

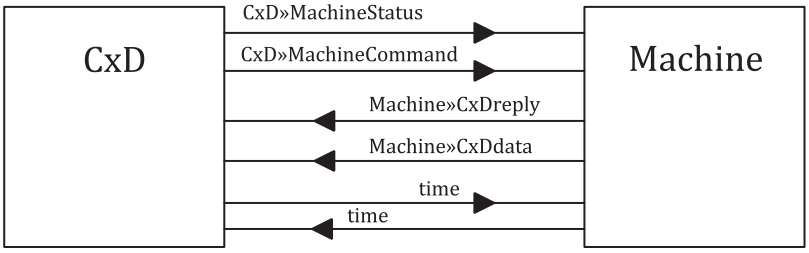


Figure 1 — Overview of logical CxD-machine interface

The logical groups defined for the on-board J1939 communication interface are:

— `CxD»MachineStatus` – this logical group of instructions allows the CxD to read or write to machine registers and detect the health of the communication interface.

NOTE 1 This logical group is used to provide machine status information to the CxD and allow the CxD to inspect and modify machine registers.

Information sent over this logical group may be sent at the maximum data rate supported by the machine.

— `CxD»MachineCommand` – instructions sent by CxD to machine to confirm or activate machine functions (e.g. slow down, stop, inhibit motion).

NOTE 2 This logical group is used to initiate interventional collision avoidance actions at the specified broadcast rate (see 7.3).

— `Machine»CxDreply` – response of machine to instructions sent by the CxD over `CxD»MachineStatus` or `CxD»MachineCommand`, which may include:

  — successful execution of the instruction;

  — an error was encountered while executing the instruction;

  — the instruction is not supported by the machine.

  The machine may limit the maximum data rate over the communication interface by delaying the response.

— `Machine»CxDdata` – data provided by machine to the CxD.

— `Time/Date` – information sent by machine to CxD or from CxD to machine to synchronise system clocks (see 7.10).

The enquiries and actions within each of these logical groups are described more fully in 5.2 to 5.4.

5.2 Negotiation

A higher level of sequence of negotiation and credentials may be used to protect the machine against access from an unauthorized CxD or connection of an unauthorised device to the interface. The negotiation sequence may be independently developed by the machine manufacturer and CxD supplier for exclusive use on a specific machine and may include:

— protocol version;

— machine model ID;

— machine generation / revision / series;

— other information defined by machine manufacturer and CxD manufacturer.

The CxD may pass credential information to the machine in a predefined sequence agreed between the CxD manufacturer and the machine manufacturer. Basic authentication methods are described in 7.2.3.1 (refer to NEGOTIATE_NOP description in Table 8).

A mechanism for establishing trust between the machine and the CxD is described in Annex B.

The machine may refuse to reply to or acknowledge all other instructions until after the negotiation sequence has been completed successfully.

Once negotiation has been successfully completed, the CxD shall send a PROTOCOL_NOP instruction within the specified maximum interval to avoid a timeout of the communication link.

5.3 Initialisation

After successful completion of the negotiation sequence, the CxD should read the contents of all registers defined on the machine and discover the capabilities of the machine using the mechanisms described in 7.2.

The CxD may read or write to registers after startup of the machine or at any time while the machine is running.

5.4 Operation

After negotiation and initialisation have been completed, the CxD may initiate interventional collision avoidance actions which are supported by the machine, including:

— motion inhibit;

— emergency stop;

— controlled stop;

— slow down;

— stand down;

— bypass propulsion;

— apply propulsion setpoints;

— no operation (NOP) – do nothing.

The machine may not support all interventional collision avoidance actions listed here. The CxD should discover the capabilities of the machine during initialisation.

Some interventional collision avoidance actions may require pre-conditions to be met, e.g. the machine is stationary before motion inhibit is applied, maximum machine speed for emergency stop, valid setpoint values provided by CxD.

Examples are shown for the PROPULSION subsystem only. Additional interventional collision avoidance actions may be defined in other subsystems.

6 Physical interface

6.1 General

The connection between the machine and the CxD is defined in 6.2 to 6.5.

The connectors specified in 6.2 to 6.5 can be unsuitable for the specific requirements of machines working in hazardous atmospheres. Alternative connector and connection arrangements may be used in these cases.

6.2 Machine connector

The physical connector on the machine shall be Deutsch DT-Series 12-pin plug part DT06-12SC-EP06 (Key C) shown in Figure 2 or equivalent[1)]. The pin connections and pin definitions are shown in Table 1.

Dimensions in millimetres (inches)

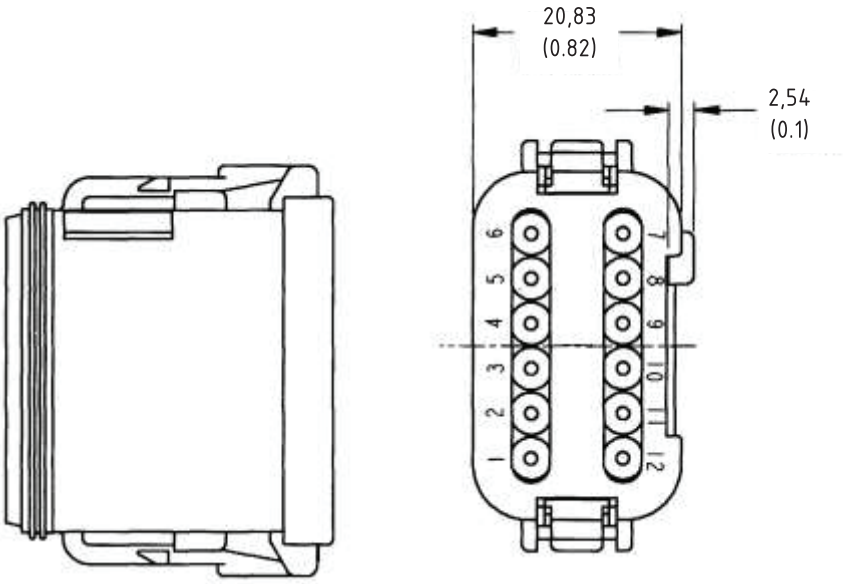


Figure 2 — Machine physical connector - Deutsch DT-series 12 pin, part DT06-12SC-EP06 (Key C)

1) Deutsch DT-Series 12-pin plug part DT06-12SC-EP06 (Key C) is an example of a suitable connector that is available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

Table 1 — Machine pin connections

Pin	Machine	Comment
1	n/a	Reserved for future use by the ISO 21815 series
2	+bat(un-switched)	10A (maximum)
3	-bat 0V/GND	Common ground, 0V from battery supply
4	+bat(switched)	10 A (maximum) switched from ignition key
5	CAN HI	SAE J1939 CAN-bus (High)
6	CAN LO	SAE J1939 CAN-bus (Low)
7	n/a	Reserved for future use by the ISO 21815 series
8	n/a	Reserved for future use by the ISO 21815 series
9	n/a	Reserved for future use by the ISO 21815 series
10	n/a	Reserved for future use by the ISO 21815 series
11	Override A	Override switch (see Table 2)
12	Override B	Override switch (see Table 2)

The maximum combined load from pin 2 and pin 4 shall not exceed 10 A. An alternative source of power should be obtained from the machine for CxS devices that exceed the combined requirements of pin 2 and pin 4.

NOTE For some machines the additional power requirement of the CxD can require modification to the machine electrical system (e.g. larger alternator, reduction in electrical loads, changes to wiring harness).

The +bat(switched) or +bat(un-switched) battery lines should not be routed around the isolator switch to connect directly to the battery.

The machine should provide protection for short circuit or overcurrent fault conditions on pin 2 and pin 4, e.g. circuit breaker, resettable fuse, fuseable link.

Alternate compatible connector body styles may be used depending on the preferred method of connection, e.g. bulkhead / chassis, in-line connection.

All unused cable entries and cavities should be plugged to preserve the IP rating of the connector.

6.3 CxD connector

The physical connector on the CxD shall be Deutsch DT-series 12-pin receptacle part DT04-12PC-BE02 (Key C) shown in Figure 3 or equivalent[2)]. The pin connections are identical to the machine connector (see Table 1).

2) Deutsch DT-series 12-pin receptacle part DT04-12PC-BE02 (Key C) is an example of a suitable connector that is available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

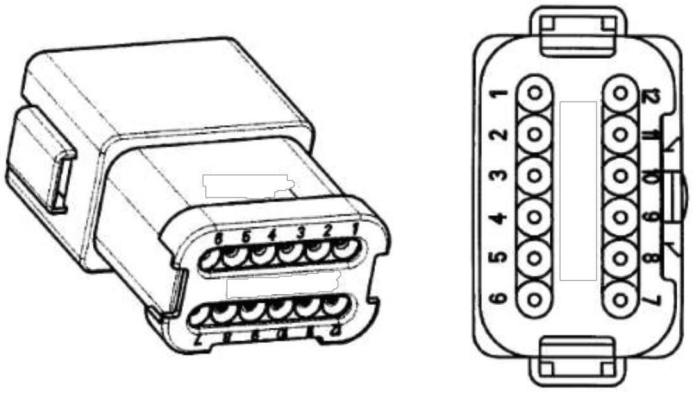


Figure 3 — CxD physical connector - Deutsch DT-series 12 pin, part DT04-12PC-BE02 (Key C)

The maximum combined load from pin 2 and pin 4 shall not exceed 10 A.

Alternate compatible connector body styles may be used depending on the preferred method of connection, e.g. bulkhead / chassis, in-line connection.

6.4 Override switch

An override signal may be implemented on the machine to notify the CxD that the operator has determined that the collision avoidance action may be overridden.

If provided by the machine, the override signal shall be implemented by parity switches connected to override A (pin 11) and override B (pin 12) with the states defined in Table 2.

Table 2 — CxD Override switch

Override A Pin 11	Override B Pin 12	Value
Open	Open	Connection fault
Open	Closed	Override enabled
Closed	Open	Override disabled
Closed	Closed	External fault
Key Closed short to 0V/GND Open machine voltage (e.g. +bat)		

If provided by the machine, the maximum sinking load on either override A or override B shall be 1 A at the system voltage of the machine as defined by the machine manufacturer.

NOTE Typical voltages for +bat (un-switched) and +bat (switched) are 12 V or 24 V DC.

If provided by the machine, the override A and override B contacts shall be maintained in the closed state when the override signal is indeterminate (e.g. during the startup sequence).

The override signal can be provided by a momentary action change-over switch, relay contacts, low impedance electronic switch with de-bounce provisions, or other means. Refer to Annex C for example implementations of the override functionality.

Transient states lasting less than 100 ms shall be ignored by the CxD.

6.5 Physical layer

The physical layer shall comply with SAE J1939-15. Both the machine connector and the CxD connector shall be CxD branches from the CxD harness as shown in Figure 4.

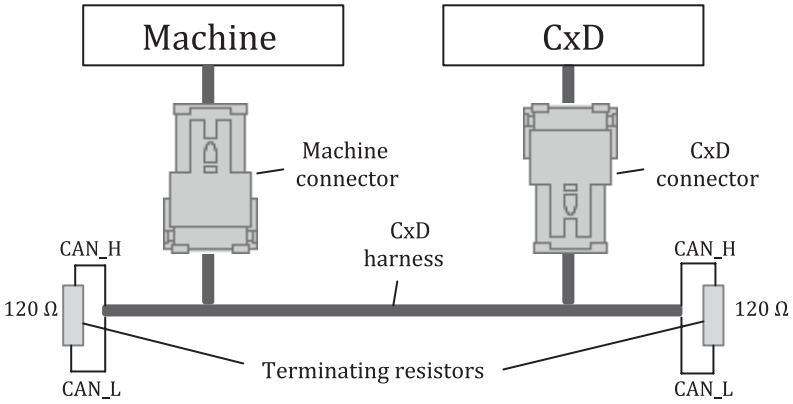


Figure 4 — Physical layer

The topology for the CxD harness should be a direct point to point connection with termination resistors at each end.

The CxD harness should be separated from the machine control bus by a suitable gateway.

The use of unshielded twisted pair (UTP) for the physical layer of the CxD harness shall meet the requirements defined in SAE J1939-15, including:

a) Connection of the CxD device to the CxD bus shall not exceed the maximum permitted number of nodes – typically 10 nodes maximum. The machine manufacturer shall state the number of nodes attached to the aux CxD bus in the factory default configuration of the machine.

b) Connection of the CxD device to the CxD bus shall not exceed the maximum physical bus length (40 m) and CxD branch limits (3 m). If the physical length of the CxD harness in the factory default configuration of the machine exceeds 1 m, the machine manufacturer shall state any physical limitations that should be taken into account before connection of the CxD.

The physical layer consisting of the CxD and CxD harness (e.g. conduit, braid, cable ways) shall meet the environmental protection requirements of ISO 19014-3.

7 J1939 communication protocol

7.1 General

The connections for the on-board communication interface between the CxD and machine shown in Figure 1 allows information to be passed between the CxD and the machine within the logical groups defined in Table 3. The functions provided within each logical group (for a specific subsystem, if specified) are shown in the right columns.

Table 3 — Information passed via logical interface

Logical group	Description	Registers
CxD»MachineStatus	Actions and enquiries sent by CxD to machine that modify machine configuration	Status Register index Register select Register value
CxD»MachineCommand	Actions and enquiries sent by CxD to machine that affect machine state	Command Register index Register select Register value
Machine»CxDreply	Response of machine to CxD actions or enquiries	Status / Command Register index Register format Register value
Machine»CxDdata	Data provided by machine to CxD	PROPULSION subsystem: — Speed — Direction — Gear position — Payload status — Traction control — Rollback status — Manual override — Machine pitch — Machine roll
Machine»CxDstatus	Actions and enquiries sent by machine to CxD that modify CxD configuration	Reserved for future use by the ISO 21815 series
Machine»CxDcommand	Actions and enquiries sent by machine to CxD that affect CxD state	Reserved for future use by the ISO 21815 series
CxD»MachineReply	Response of CxD to machine actions or enquiries	Reserved for future use by the ISO 21815 series
CxD»MachineData	Data provided by the CxD to the machine	Reserved for future use by the ISO 21815 series
TD	Time/Date	SAE J1939

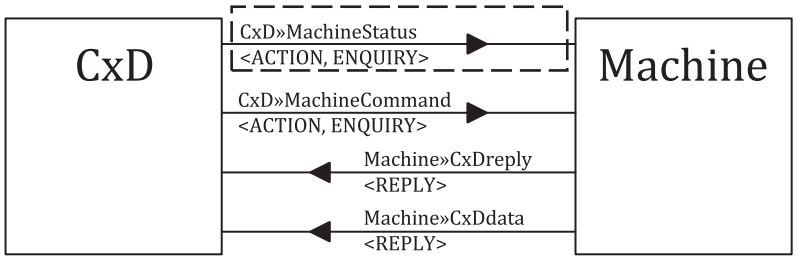
7.2 PGN:CxD»machine status

7.2.1 General

The structure of PGN:CxD»MachineStatus is shown in Figure 5. Additional details are provided in 7.2.2 and 7.2.3.

PGN

CxD»MachineStatus



SPNs for **PGN:CxD»MachineStatus** (see Table 5)

SPN	Offset (byte.bit)	SPN length
Status	1.1	8 bits
Register Index[a]	2.1	8 bits
Register Select[b]	3.1	8 bits
Register Value_0	4.1	8 bits
Register Value_1	5.1	8 bits
Register Value_2	6.1	8 bits
Register Value_3	7.1	8 bits
Message Identifier	8.1	8 bits

Notes: (a) see 7.2.3.2
(b) see 7.2.3.3

Reserved SPN values (see Table 6)

Code	MESSAGE_STATUS
0xFA	GENERAL_FAULT
0xFE	GENERAL_ERROR
0xFF	IGNORE

Note: Reserved SPN values may be used in bytes 2, 3 and only for registers with J1939 data format in bytes 4, 5, 6, 7. Reserved SPN values should not be used for the Message Identifier (byte 8).

Status

7	6	5	4	3	2	1	0

ENQUIRY

1	SUB_SYSTEM

ACTION

0	SUB_SYSTEM

SUB_SYSTEM

0	0	0	PROPULSION
0	0	1	RESERVED_1
0	1	0	RESERVED_2
0	1	1	RESERVED_3
1	0	0	RESERVED_4
1	0	1	RESERVED_5
1	1	0	USER_DEFINED
1	1	1	PROTOCOL

Status (SUB_SYSTEM=PROTOCOL)

7	6	5	4	3	2	1	0

ENQUIRY

1	1	1	1	0	ENQ7

ACTION

0	1	1	1	0	ACT7

Status (SUB_SYSTEM=PROPULSION)

7	6	5	4	3	2	1	0

ENQUIRY

1	0	0	0	RSV	ENQ0

ACTION

0	0	0	0	RSV	ACT0

RSV - Reserved for future use

ENQ7 bitfield codes in PROTOCOL subsystem – **SPN:Status** (see Table 8)

ENQ7	Name
0b000	PROTOCOL_NOP
0b001	NEGOTIATE_NOP
0b010	NEGOTIATE_ENQ
0b101	GET_EXTENDED_REGISTER
0b110	GET_PROTOCOL_REGISTER

ACT7 bitfield codes in PROTOCOL subsystem – **SPN:Status** (see Table 9)

ACT7	Name
0b101	RESET_REGISTERS
0b110	SET_PROTOCOL_REGISTER

ENQ0 bitfield codes in PROPULSION subsystem – **SPN:Status** (see Table 10)

ENQ0	Name
0b110	GET_PROPULSION_REGISTER

ACT0 bitfield codes in PROPULSION subsystem – **SPN:Status** (see Table 11)

ACT0	Name
0b100	LOAD_PROPULSION_SETPOINTS
0b110	SET_PROPULSION_REGISTER

Figure 5 — Structure of `PGN:CxD»MachineStatus`

Refer to Annex A for typical communication sequences and Table A.1 and Table A.2 for examples of CxD instruction and machine responses.

7.2.2 PGN description

The CxD shall send status information to the machine via the PGN with the parameters shown in Table 4 and as defined in 7.2.3.

Table 4 — `PGN:CxD»MachineStatus` parameters

Parameter	Description
PGN number	61968
PGN type	PDU2_FAST
Acronym	CXD1
Data length	8 bytes
Multipacket message	No
Broadcast rate	10 ms[a]
Message priority	3

[a] The machine manufacturer may limit the rate that instructions are executed over this PGN by delaying the reply via `PGN:Machine»CxDreply`.

This PGN is intended to be used by the CxD to transmit a burst of information to the machine not exceeding the specified broadcast rate.

The CxD may remain silent between bursts.

The CxD shall maintain the communication link in an active state by periodically sending a PROTOCOL_NOP instruction (see Table 8) to avoid a timeout when no other instructions are being sent.

The SPN's associated with `PGN:CxD»MachineStatus` are shown in Table 5.

Table 5 — SPNs for `PGN:CxD»MachineStatus`

SPN	Offset (byte.bit)	SPN length	SPN Description
Status	1.1	8 bits	Action or enquiry sent from CxD to the machine
Register index	2.1	8 bits	Index to machine register
Register select	3.1	8 bits	Used in conjunction with `SPN:RegisterIndex` to set TAG values that enable multiple matching setpoints to be executed via `PGN:CxD»MachineCommand`.
Register value_0	4.1	8 bits	Least significant byte of register value specified by `SPN:RegisterIndex`
Register value_1	5.1	8 bits	Next significant byte of register value specified by `SPN:RegisterIndex`
Register value_2	6.1	8 bits	Next significant byte of register value specified by `SPN:RegisterIndex`
Register value_3	7.1	8 bits	Most significant byte of register value specified by `SPN:RegisterIndex`
Message identifier	8.1	8 bits	Identifier for message originated by the CxD e.g. incrementing counter

NOTE Unless specified otherwise, least-significant-first byte ordering is used for multi-byte transfers with individual bytes preserving J1939 bit ordering (big-endian).

The standard SAE J1939 reserved codes (see Table 6) may be used in `SPN:RegisterIndex` (offset = 2), `SPN:RegisterSelect` (offset = 3) and where noted in 7.2.3. Generally, the reserved codes are used to indicate missing data or an error. The use of these reserved codes in `SPN:Status` (offset = 1) and `SPN:MessageIdentifier` (offset = 8) is not permitted.

Table 6 — Reserved SPN values

Code	MESSAGE_STATUS
0xFA	GENERAL_FAULT
0xFE	GENERAL_ERROR
0xFF	IGNORE

Each SPN within `PGN:CxD»MachineStatus` is defined in more detail in 7.2.3.

7.2.3 SPN structure

7.2.3.1 SPN:Status

7.2.3.1.1 General

This SPN is used by the CxD to send an enquiry or action to a specified subsystem of the machine. Refer to Figure 6 for a description of the SUB_SYSTEM bitfield.

The typical negotiation, initialisation and operation message sequences for this SPN are described in Annex A.

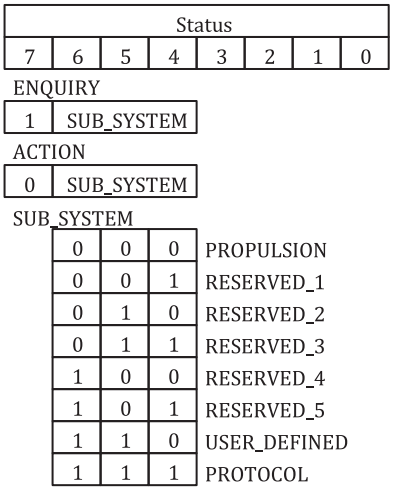


Figure 6 — Subsystem bitfields in `SPN:Status`

The valid enumerations for the SUB_SYSTEM bitfield are listed in Table 7.

NOTE The least significant 4-bits of `SPN:Status` are defined separately for each subsystem.

Table 7 — SUB_SYSTEM bitfield codes

SUB_ SYS-TEM	Name	Description
0b000	PROPULSION	Propulsion subsystem
0b001	RESERVED_1	Reserved for future use by the ISO 21815 series
0b010	RESERVED_2	Reserved for future use by the ISO 21815 series
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series
0b101	RESERVED_5	Reserved for future use by the ISO 21815 series
0b110	USER_DEFINED	Available for customisation by the end user or application
0b111	PROTOCOL	Protocol subsystem

7.2.3.1.2 SUB_SYSTEM: PROTOCOL

The format of `SPN:Status` for the PROTOCOL subsystem is shown in Figure 7.

Status (SUB_SYSTEM=PROTOCOL)							
7	6	5	4	3	2	1	0
ENQUIRY							
1	1	1	1	0	ENQ7		
ACTION							
0	1	1	1	0	ACT7		

Figure 7 — PROTOCOL subsystem bitfields – `SPN:Status`

The valid enumerations of the bitfields in `SPN:Status` for the PROTOCOL subsystem are listed in Table 8 and Table 9.

Table 8 — ENQ7 bitfield codes in PROTOCOL subsystem – `SPN:Status`

ENQ7	Name	Description
0b000	PROTOCOL_ NOP	This enquiry is used by the CxD after successful completion of the negotiation sequence to indicate to the machine that the CxD is healthy when no other active instructions are being sent to the machine. The machine may silently ignore this enquiry if negotiation has not been completed successfully. Use the NEGOTIATE_NOP enquiry to verify the connection between the CxD and the machine if negotiation has not been completed. The machine may force the CxD to reinitiate the negotiation sequence if no valid instructions are sent within the time specified in register INSTRUCTION_TIMEOUT of the protocol subsystem.
0b001	NEGOTIATE_ NOP	This enquiry is used by the CxD to indicate to the machine that the CxD is active and ready to perform a negotiation with the machine and the machine shall provide information to the CxD on the current state of the interface. The values in `SPN:RegisterIndex` and `SPN_RegisterSelect` and `SPN:RegisterValue0..3` are ignored for this enquiry and should be set to zero (0x00). The machine shall always respond to a NEGOTIATE_NOP enquiry with the following information in `SPN:RegisterValue0..3`: `RegisterValue0`: J1939 = INTERFACE_STATE `RegisterValue1`: REG_FORMAT = format of a non-zero NEGOTIATION_SEED determined by the machine and NEGOTIATION_KEY to be provided by the CxD `RegisterValue2`: UINT8 = Authentication method: — 0x00: no authentication; — 0x01: SEED/KEY combination; — 0x02: SEED/KEY with tokenisation of commands; — 0xE0-0xEF: user defined; — other values reserved for future use by this document. `RegisterValue3`: Reserved for future use by this document.

Table 8 *(continued)*

ENQ7	Name	Description
0b010	NEGOTIATE_ENQ	A higher level of sequence of negotiation and credentials may be used to protect the machine against access from an unauthorized CxD or connection of an unauthorised device to the interface. The negotiation sequence may be independently developed by the machine manufacturer and CxD supplier for exclusive use on a specific machine. This enquiry may be used by the CxD to pass credential information to the machine in a predefined sequence agreed between the CxD and machine manufacturer. The negotiation sequence may pass up to 32-bits per message using one of the formats specified in Table 36. A generic example negotiation sequence starts with the CxD requesting the seed value for the negotiation from the machine by specifying `SPN:RegisterIndex` = NEGOTIATION_SEED. The machine should respond with the non-zero seed value in `SPN:RegisterValue0..3` in the format indicated in the reply to the NEGOTIATE_NOP enquiry. The CxD can then specify the matching key value expected by the machine in `SPN:RegisterValue0..3`. The machine should indicate if an error occurs by replying with the appropriate value in the REG_FORMAT bitfield (see Table 36). The machine should prevent a CxD attempting to discover a valid key using trial and error by refusing to reply to the NEGOTIATE_ENQ for a suitable period of time and returning a zero seed in response to the enquiry by the CxD with SPN:Register Index = NEGOTIATION_SEED. Other variations of the negotiation sequence may pass the following information between the machine and the CxD: — protocol version; — machine model ID; — machine generation / revision / series; — other information defined by machine manufacturer and CxD supplier. An example of a negotiation sequence based on exchange of certificates is described in Annex B. The machine may refuse to reply to or acknowledge all actions or enquiries other than NEGOTIATE_NOP or NEGOTIATE_ENQ until the negotiation sequence has been completed successfully. The machine may force the CxD to restart negotiation of its credentials if the negotiation sequence is not completed within the time specified in register NEGOTIATION_TIMEOUT of the protocol subsystem. Where required by the machine manufacturer a token should be provided by the CxD for every command sent to the machine in a pseudo-random sequence via `PGN:CxD»MachineCommand`. The token sequence should be randomised by the initial seed provided by machine during the negotiation and ordered by a shared secret code that is known independently by the machine and the CxD.
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series.
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series.
0b101	GET_EXTENDED_REGISTER	This enquiry is used to access additional registers outside the 8-bit register index space accessible via GET_PROTOCOL_REGISTER (see below). The primary use of extended registers is to allow the full J1939 address space to be accessed indirectly as read-only registers if implemented by the machine manufacturer. Extended registers can be selected by the 32-bit index range specified by:

Table 8 *(continued)*

ENQ7	Name	Description
		— SPN:RegisterValue0 – least significant byte — SPN:RegisterValue1 — SPN:RegisterValue2 — SPN:RegisterValue3 – most significant byte The machine can indicate if this capability is available by returning the appropriate SPN:Status/Command in PGN:Machine»CxDreply (see Table 28 for responses). NOTE 1: Registers in the extended index space have REG_TYPE = PARAMETER and REG_FORMAT = READ_ONLY and are common for all subsystems. NOTE 2: The first 2048 registers in the 32-bit extended index range are mapped to the registers in subsystem address space: — 0x00000000 – 0x000000FF (SUB_SYSTEM = 0) — 0x00000100 – 0x000001FF (SUB_SYSTEM = 1) — . — 0x00011100 – 0x000111FF (SUB_SYSTEM = 7)
0b110	GET_PROTOCOL_REGISTER	This enquiry is used to read the protocol register at SPN:RegisterIndex. The machine can indicate if this capability is available by returning the appropriate SPN:Status/Command in PGN:Machine»CxDreply.
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

Table 9 — ACT7 bitfield codes in PROTOCOL subsystem – SPN:Status

ACT7	Name	Description
0b000	RESERVED_0	Reserved for future use by the ISO 21815 series.
0b001	RESERVED_1	Reserved for future use by the ISO 21815 series.
0b010	RESERVED_2	Reserved for future use by the ISO 21815 series.
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series.
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series.
0b101	RESET_REGISTERS	This action is used to reset all registers in the specified subsystem to the factory default settings specified by the manufacturer. The subsystem is defined by the SUB_SYSTEM bitfield in SPN:RegisterSelect (SELECT=SELECT_SUBSYSTEM). The machine can indicate if this capability is available by returning the appropriate SPN:Status/Command in PGN:Machine»CxDreply.
0b110	SET_PROTOCOL_REGISTER	This action is used to set the value of the protocol register specified in SPN:RegisterIndex to SPN:RegisterValue0..3. The machine can indicate if this capability is available by returning the appropriate SPN:Status/Command in PGN:Machine»CxDreply. NOTE: The value in SPN:RegisterSelect is ignored for registers in the PROTOCOL subsystem.
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

The machine shall reply to a NEGOTIATE_NOP enquiry issued by the CxD at the broadcast rate specified in Table 4. The machine may refuse to reply or acknowledge other enquiries or actions until negotiation has been completed successfully.

EXAMPLE

`SPN:Status` = 0xF2 (ENQUIRY, SUB_SYSTEM=PROTOCOL, ENQ7=NEGOTIATE_ENQ)

7.2.3.1.3 SUB_SYSTEM: PROPULSION

The format of `SPN:Status` for the PROPULSION subsystem is shown in Figure 8.

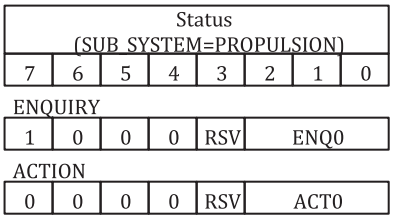


Figure 8 — PROPULSION subsystem bitfields – `SPN:Status`

The valid enumerations of the ENQ0, ACT0 and RSV bitfields in `SPN:Status` for the PROPULSION subsystem are shown in Table 10 to Table 12.

Table 10 — ENQ0 bitfield codes in PROPULSION subsystem – `SPN:Status`

ENQ0	Name	Description
0b000	RESERVED_0	Reserved for future use by the ISO 21815 series.
0b001	RESERVED_1	Reserved for future use by the ISO 21815 series.
0b010	RESERVED_2	Reserved for future use by the ISO 21815 series.
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series.
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series.
0b101	RESERVED_5	Reserved for future use by the ISO 21815 series.
0b110	GET_ PROPULSION_ REGISTER	This enquiry is used to read the propulsion register at `SPN:RegisterIndex`. The machine can indicate if this capability is available.
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

Table 11 — ACT0 bitfield codes in PROPULSION subsystem – `SPN:Status`

ACT0	Name	Description
0b000	RESERVED_0	Reserved for future use by the ISO 21815 series.
0b001	RESERVED_1	Reserved for future use by the ISO 21815 series.
0b010	RESERVED_2	Reserved for future use by the ISO 21815 series.
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series.

Table 11 *(continued)*

ACT0	Name	Description
0b100	LOAD_ PROPULSION_ SETPOINTS	This action is used in conjunction with `SPN:RegisterSelect` (SELECT= IMMEDIATE, UPDATE_AND_APPLY, APPLY_FROM_LIST, LOOKUP_INDIRECT, MATCH_TAG) to set the value of one or more propulsion setpoints as specified by the values of `SPN:RegisterValue0..3` (see Figure 12). This action does not result in setpoints being immediately applied on the machine. The setpoints affected by this action are temporarily flagged by the machine in preparation for an APPLY_PROPULSION_ SETPOINTS_CONFIRM enquiry and an APPLY_PROPULSION_ SETPOINTS action to be issued by the CxD. The capability of the machine to apply the intended intervention action may be tested using the CHK0=APPLY_PROPULSION_SETPOINTS_ CONFIRM enquiry (see Table 24) and then executed using the XEQ0=APPLY_ PROPULSION_ SETPOINTS action (see Table 25). The temporary flag on the affected setpoints is removed by any of the following actions issued by the CxD: — a new ACT0=LOAD_PROPULSION_SETPOINTS action; — a ACT0=SET_PROPULSION_REGISTER action; — a ACT7=RESET_REGISTERS action on the PROPULSION subsystem (see Table 9); — an XEQ0=APPLY_PROPULSION_SETPOINTS action (see Table 25). The machine has the opportunity to validate any particular combination of setpoints that are specified by the CxD. The 2-step sequence is for the CxD to load the setpoints via `SPN:Status` using the action ACT0= LOAD_PROPULSION_SETPOINTS (see Table 11) then checking if the values are valid via `SPN:Command` using the enquiry CHK0=APPLY_PROPULSION_SETPOINTS_CONFIRM (see Table 24). The machine may reply to the enquiry with APPLY_PROPULSION_SETPOINTS_CONFIRM_ACK or ENQUIRY_ERROR. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b101	RESERVED_5	Reserved for future use by the ISO 21815 series.
0b110	SET_ PROPULSION_ REGISTER	This action is used in conjunction with `SPN:RegisterSelect` (SELECT= SELECT_REGISTER, SELECT_AND_TAG) to set the value of the propulsion register specified in `SPN:RegisterIndex` to `SPN:RegisterValue0..3`. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`. The TAG bitfield (see 7.2.3.3) may optionally be specified for registers with REG_TYPE = SET_POINT (see Table 34) to allow multiple setpoints to be executed via `PGN:CxD»MachineCommand`.
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

Table 12 — RSV bitfield codes in PROPULSION subsystem – `SPN:Status`

RSV	Name	Description
0b0	-	Reserved for future use by the ISO 21815 series.
0b1	-	Reserved for future use by the ISO 21815 series.

7.2.3.2 SPN:RegisterIndex

7.2.3.1.4 General

This SPN contains the index of a machine register to be referenced by `SPN:RegisterSelect` and `SPN:RegisterValue0..3` (see 7.3.3.3 and 7.3.3.4). The interpretation of this SPN is controlled by the SUB_SYSTEM bitfield in `SPN:Status`.

The reserved message status codes (see Table 6) may be used for this SPN if registers have not been implemented on the machine.

The common index values including the protocol revision register and several registers to allow a CxD to sequentially discover the index values of registers defined in each subsystem are shown in Table 13.

Table 13 — Register index (all subsystems)

INDEX	Name	Description
0xF0	PROTOCOL_ REVISION	Index of register identifying the revision of the ISO/TS 21815-2 protocol implemented by the CxD in `SPN:RegisterValue0..3` in the following format: — `SPN:RegisterValue0` – revision number of this document — `SPN:RegisterValue1` – amendment number of this document — `SPN:RegisterValue2` – major revision (user defined) — `SPN:RegisterValue3` – minor revision (user defined) The format should correspond to the format information returned in `SPN:RegisterFormat` in the reply from the machine: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_WRITE — REG_FORMAT = CHAR4 The initial revision and amendment numbers on publication of this document are: — `SPN:RegisterValue0` = 0x01 (revision) — `SPN:RegisterValue1` = 0x00 (amendment) The machine should reply with the version of the ISO/TS 21815-2 protocol implemented on the machine in `SPN:RegisterValue0..3`. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF1	REGISTER_ COUNT	Total number of registers in the subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status` supplied by the CxD and the machine replying with the value in `SPN:RegisterValue0..1` using the format in `SPN:RegisterFormat`: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF2	FIRST_ REGISTER	Index of the first register in subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status` supplied by the CxD and the machine replying with the value in `SPN:RegisterValue0..1` using the format in `SPN:RegisterFormat`:

Table 13 *(continued)*

INDEX	Name	Description
		— REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF3	NEXT_ REGISTER	Index of the next register in subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status` supplied by the CxD and the machine replying with the value in `SPN:RegisterValue0..1` using the format in `SPN:RegisterFormat`: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF4	PREV_ REGISTER	Index of the previous register in subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status` supplied by the CxD and the machine replying with the value in `SPN:RegisterValue0..1` using the format in `SPN:RegisterFormat`: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF5	LAST_ REGISTER	Index of the last register in subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status` supplied by the CxD and the machine replying with the value in `SPN:RegisterValue0..1` using the format in `SPN:RegisterFormat`: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0xF6-0xF9	-	Reserved for future use by the ISO 21815 series.

7.2.3.1.5 SUB_SYSTEM: PROTOCOL

The register index values used by the PROTOCOL subsystem are shown in Table 14.

Table 14 — Register index (PROTOCOL subsystem)

INDEX	Name	Description
0x00	SUBSYSTEM_ MCAPS	Index of register identifying the subsystems that are supported by the machine. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 A description of the bitfields and the codes used for this register are shown in Figure 9. The SUBSYSTEM_MCAPS register is provided for information only. If this register is not defined (REG_DEF = UNDEFINED) the CxD should use other discovery methods described in this technical specification to discover the capabilities of the machine. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x01	EXTENDED_ INDEX	Index of registers in the 32-bit extended index range specified by: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` – most significant byte The extended index space is reserved for future use by the ISO 21815 series. The registers for each of the eight (8) subsystems are mapped as blocks of 256 consecutive registers occupying the first 2048 registers in the EXTENDED_REGISTER index space with SUB_SYSTEM=0 commencing at extended index 0. Setpoint registers are only accessible via the subsystem register space. All registers in the extended index space are treated as REG_TYE = PARAMETER, REG_FORMAT = J1939 and RAG_ATTRIB=READ_ONLY.
0x02	INTERFACE_ STATE	Index of register defining the current state of the interface between the CxD and the machine: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 A description of the bitfields and the codes used for this register are shown in Figure 10 and Table 15 to Table 17. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x03	-	Reserved for future use by the ISO 21815 series.

Table 14 *(continued)*

INDEX	Name	Description
0x04	NEGOTIATION_ SEED	Index of the initial seed value provided by the machine for the negotiation sequence: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = <specified by machine manufacturer> The machine responds with a value in `SPN:RegisterValue0..3`. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. The machine may exchange an arbitrary length seed (e.g. a public key) with the CxD by specifying REG_FORMAT = MULTI_BYTE and initiating a multi-byte data transfer using the J1939 broadcast address. A CRC-32 checksum should be provided by the machine in `SPN:RegisterValue0..3` to allow the CxD to verify correct transmission. If implemented, the machine should reset the seed with a new random value for every negotiation sequence.
0x05	NEGOTIATION_ KEY	Index of the key provided by the CxD in response to the machine-provided seed to complete the negotiation sequence: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_WRITE — REG_FORMAT = <specified by machine manufacturer> The machine responds with the expected bit length of the key in `SPN:RegisterValue0..3` The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. The machine may accept an arbitrary length key (e.g. a certificate) from the CxD by specifying REG_FORMAT = MULTI_BYTE. The CxD initiates the multi-byte data transfer using the J1939 broadcast address immediately after setting the value of this register to the CRC-32 checksum of the key in `SPN:RegisterValue0..3`. The machine should calculate the CRC-32 of the received key and compare with the value provided by the CxD to verify correct transmission. The value of the key required by the machine should be unique for every negotiation sequence.
0x06	MACHINE_ SOFTWARE_ REVISION	Index of the machine software version: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the value in `SPN:RegisterValue0..3`: MACHINE_SOFTWARE_BUILD_0 [LSB] MACHINE_SOFTWARE_BUILD_1 [MSB] MACHINE_SOFTWARE_MINOR MACHINE_SOFTWARE_MAJOR The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.

Table 14 *(continued)*

INDEX	Name	Description
0x07	MACHINE_ID_0	Index of the machine id subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status`. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the machine id in `SPN:RegisterValue0..3`: MACHINE_ID_BYTE_0 [LSB] MACHINE_ID_BYTE_1 MACHINE_ID_BYTE_2 MACHINE_ID_BYTE_3 [MSB] The Machine ID should be a unique number assigned by the end user or allocated by the machine manufacturer for a specific machine, e.g. a vehicle identification number (VIN) defined in ISO 3779 or a product identification number (PIN) as defined in ISO 10261. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x08	MACHINE_ID_1	Index of the machine id subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status`. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the machine id in `SPN:RegisterValue0..3`: MACHINE_ID_BYTE_4 [LSB] MACHINE_ID_BYTE_5 MACHINE_ID_BYTE_6 MACHINE_ID_BYTE_7 [MSB] Refer to note for MACHINE_ID_0 above. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x09	MACHINE_ID_2	Index of the machine id subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status`. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the machine id in `SPN:RegisterValue0..3`: MACHINE_ID_BYTE_8 [LSB] MACHINE_ID_BYTE_9 MACHINE_ID_BYTE_10 MACHINE_ID_BYTE_11 [MSB] Refer to note for MACHINE_ID_0 above. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.

Table 14 *(continued)*

INDEX	Name	Description
0x0A	MACHINE_ID_3	Index of the machine id subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status`. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the machine id in `SPN:RegisterValue0..3`: MACHINE_ID_BYTE_12 [LSB] MACHINE_ID_BYTE_13 MACHINE_ID_BYTE_14 MACHINE_ID_BYTE_15 [MSB] Refer to note for MACHINE_ID_0 above. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x0B	MACHINE_ID_4	Index of the machine id subsystem specified by the SUB_SYSTEM bitfield of `SPN:Status`. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = CHAR4 The machine responds with the machine id in `SPN:RegisterValue0..3`: MACHINE_ID_BYTE_16 [LSB] SPARE_17 SPARE_18 SPARE_19 [MSB] Refer to note for MACHINE_ID_0 above. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x0C-0x0F	-	Reserved for future use by the ISO 21815 series.
0x10	CXD_SOFTWARE_REVISION	Index of the CxD software version: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_WRITE — REG_FORMAT = CHAR4 The machine responds with the value in `SPN:RegisterValue0..3`: CXD_SOFTWARE_MAJOR [LSB] CXD _SOFTWARE_MINOR CXD _SOFTWARE_BUILD_0 CXD _SOFTWARE_BUILD_1 [MSB] The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.

Table 14 *(continued)*

INDEX	Name	Description
0x11	CXD_HARDWARE_REVISION	Index of the CxD hardware version: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_WRITE — REG_FORMAT = CHAR4 The machine responds with the value in `SPN:RegisterValue0..3`: CXD _HARDWARE_BUILD_0 [LSB] CXD _HARDWARE_BUILD_1 [MSB] CXD _HARDWARE_MINOR CXD _HARDWARE_MAJOR The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x12	CXD_HARDWARE_ID	Index of the CxD hardware identity: — REG_TYPE = PARAMETER — REG_ATTRIB = READ_WRITE — REG_FORMAT = CHAR4 The machine responds with the value in `SPN:RegisterValue_0..3`: CXD _ID_BYTE_0 [LSB] CXD _ID_BYTE_1 CXD _ID_BYTE_2 CXD _ID_BYTE_3 [MSB] The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x13-0x1F	-	Reserved for future use by the ISO 21815 series.
0x20	INSTRUCTION_TIMEOUT	The maximum elapsed time in milliseconds between successive instructions (`PGN:CxD»MachineStatus` or `PGN:CxD»MachineCommand`) sent by the CxD to the machine. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine responds with the timeout in `SPN:RegisterValue0..1`. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x21	NEGOTIATION_TIMEOUT	The maximum elapsed time in milliseconds between initiation of the negotiation sequence and successful completion of negotiation. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine responds with the timeout in `SPN:RegisterValue0..1`. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. The machine may reset the negotiation sequence if this timeout value is exceeded.

Table 14 *(continued)*

INDEX	Name	Description
0x22	RENEGOTIATION_ TIMEOUT	The maximum elapsed time in milliseconds between initiation of a renegotiation request by the machine and initiation of a new negotiation sequence. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = USHORT16 The machine responds with the timeout in `SPN:RegisterValue0..1`. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. The machine may reset to the pre-negotiation state if this timeout value is exceeded.
0x23-0xDF	-	Reserved for future use by the ISO 21815 series.
0xE0-0xEF	-	User defined (available for use).
0xF0-0xF9	-	Common registers (see Table 13).

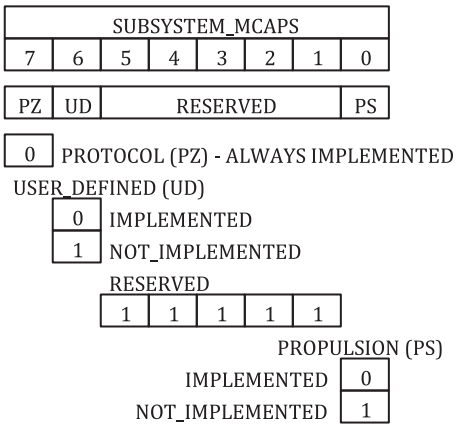


Figure 9 — bitfield codes in register SUBSYSTEM _MCAPS

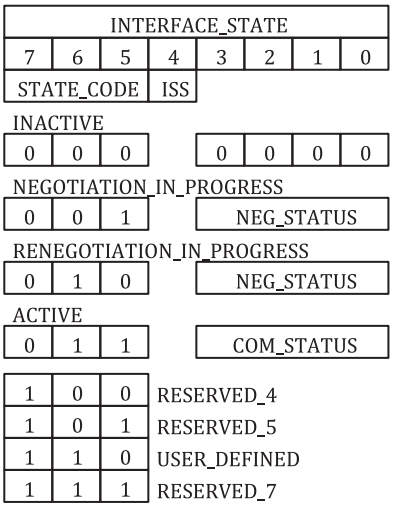


Figure 10 — bitfield codes in register INTERFACE_STATE

Refer to Table 15, Table 16 and Table 17 for a description of the interface state codes and status codes.

The ISS bitfield of the INTERFACE_STATE register is reserved for future use by the ISO 21815 series.

Table 15 — STATE_CODE bitfield in register INTERFACE_STATE

STATE_ CODE	Name	Description
0b000	INACTIVE	Interface in an uninitialized state
0b001	NEGOTIATION_ IN_PROGRESS	Negotiation in progress
0b010	RENEGOTIATION_ IN_PROGRESS	Renegotiation in progress
0b011	ACTIVE	Interface between the CxD and the machine is ready to accept instructions
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series
0b101	RESERVED_5	Reserved for future use by the ISO 21815 series
0b110	USER_DEFINED	Available for customisation by the end user or application
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series

Table 16 — NEG_STATUS bitfield in register INTERFACE_STATE

NEG_ STATUS	Name	Description
0b0000	RESERVED_0	Reserved for future use by the ISO 21815 series
0b0001	NEG_INITIATED	Negotiation / renegotiation in progress
0b0010	NEG_ERROR	Negotiation / renegotiation error (sequence failure)
0b0011	NEG_TIMEOUT	Negotiation / renegotiation timeout (CxD taking too long to complete negotiation sequence) – cleared by CxD starting a new negotiation sequence
0b0100	NEG_CREDFAIL	Negotiation / renegotiation failed (credentials provided in incorrect format) – cleared by CxD starting a new negotiation sequence
0b0101	NEG_CREDINVALID	Negotiation / renegotiation failed (invalid credentials) – cleared by CxD starting a new negotiation sequence
0b0110	NEG_CREDEXPIRED	Negotiation / renegotiation failed (expired credentials) – cleared by CxD starting a new negotiation sequence
0b0111	RESERVED_7	Reserved for future use by the ISO 21815 series
0b1000	RESERVED_8	Reserved for future use by the ISO 21815 series
0b1001	RESERVED_9	Reserved for future use by the ISO 21815 series
0b1010	RESERVED_10	Reserved for future use by the ISO 21815 series
0b1011	RESERVED_11	Reserved for future use by the ISO 21815 series
0b1100	RESERVED_12	Reserved for future use by the ISO 21815 series
0b1101	RESERVED_13	Reserved for future use by the ISO 21815 series
0b1110	RESERVED_14	Reserved for future use by the ISO 21815 series
0b1111	NEG_COMPLETE	Negotiation completed (valid credentials) – waiting for PROTOCOL_NOP to keep interface alive and change state of interface to ACTIVE

Table 17 — COM_STATUS bitfield in register INTERFACE_STATE

COM_ STATUS	Name	Description
0b0000	READY	The machine is ready to accept instructions from the CxD

Table 17 *(continued)*

COM_ STATUS	Name	Description
0b0001	INSTRUCTION_ TIMEOUT	CxD taking too long to issue another instruction – cleared by CxD starting a new negotiation sequence
0b0010	RESERVED_2	Reserved for future use by the ISO 21815 series
0b0011	RESERVED_3	Reserved for future use by the ISO 21815 series
0b0100	RENEG_PENDING	Renegotiation requested by machine (interface still fully operational) – cleared by CxD starting a new negotiation sequence
0b0101	RENEG_TIMEOUT	Renegotiation timeout (CxD taking too long to initiate a new negotiation sequence) – cleared by CxD starting a new negotiation sequence
0b0110	RESERVED_6	Reserved for future use by the ISO 21815 series
0b0111	RESERVED_7	Reserved for future use by the ISO 21815 series
0b1000	COM_CHECKSUM	Checksum failure – cleared by CxD issuing a new instruction without a checksum error The error rate on the CAN-bus should be monitored and flagged by this code if the Bit Error Rate (BER) is unacceptable.
0b1001	COM_INVALID	Invalid message – cleared by CxD issuing a valid new instruction
0b1010	COM_UNSEQ	Out of sequence message – cleared by CxD issuing a valid new instruction
0b1011	COM_DUPLICATE	Duplicate message identifier – cleared by CxD issuing a valid new instruction
0b1100	COM_CONFLICT	Multiple CxD master devices detected – cleared by CxD starting a new negotiation sequence
0b1101	RESERVED_13	Reserved for future use by the ISO 21815 series
0b1110	RESERVED_14	Reserved for future use by the ISO 21815 series
0b1111	COM_ERROR	General communication error – cleared by CxD issuing a valid new instruction

EXAMPLE

`SPN:RegisterIndex` = 0xF0 (INDEX=PROTOCOL_REVISION)

– index of register containing the revision number of the protocol implemented on the machine.

7.2.3.1.6 SUB_SYSTEM: PROPULSION

The register index values used by the PROPULSION subsystem are shown in Table 18.

Table 18 — Register Index (PROPULSION subsystem)

INDEX	Name	Description
0x00	PROPULSION_ MCAPS	Register identifying the machine capabilities that are supported by the PROPULSION subsystem. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 (see bitfields below) A description of the bitfields and the codes used for this register are shown in Figure 11. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
INDEX values in the range 0x00-0xEF may return different values for each subsystem specified in SPN:Status.		

Table 18 *(continued)*

INDEX	Name	Description
0x01	MIN_BRAKING	Setpoint register for desired brake application 0-249 encoded onto range 0x00-0xF9 with zero corresponding to no braking and 249 corresponding to full application of brakes. — REG_TYPE = SET_POINT — REG_ATTRIB = READ_WRITE — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. The braking characteristics of the machine should be considered in the configuration of the CxD.
0x02	MAX_THROTTLE	Setpoint register for desired throttle position 0-100 % encoded onto range 0x00-0xC8 with a scaling of 0,5 % per bit and zero offset. Values in the range 0xC9-0xF9 are interpreted as 100 %. — REG_TYPE = SET_POINT — REG_ATTRIB = READ_WRITE — REG_FORMAT = J1939 The operation of this setpoint is limited to reduction in throttle only and excludes functionality to maintain a certain throttle position which requires acceleration or automatic release of braking. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`. NOTE: The machine response corresponding to 100 % throttle is defined by the machine manufacturer (e.g. engine rpm, applied torque).
0x03	MAX_SPEED	Setpoint register for maximum permitted ground speed of machine 0-124,5 km/h encoded onto range 0x00-0xF9 with a scaling of 0,5 km/h per bit and zero offset. — REG_TYPE = SET_POINT — REG_ATTRIB = READ_WRITE — REG_FORMAT = J1939 The machine manufacturer may limit the range of speeds that are available using this setpoint register. Operation of this setpoint is limited to reduction in speed only by braking or other means, and excludes functionality to maintain certain speed which requires acceleration or automatic release of braking. The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x04-0x80	-	Reserved for future use by the ISO 21815 series.
0x81	EMERGENCY_ STOP_ MAX_SPEED	Register containing maximum speed for EMERGENCY_STOP action with a range 0-124,5 km/h encoded onto 0x00-0xF9, scaling of 0,5 km/h per bit and zero offset. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
INDEX values in the range 0x00-0xEF may return different values for each subsystem specified in SPN:Status.		

Table 18 *(continued)*

INDEX	Name	Description
0x82	CONTROLLED_ STOP_ MAX_SPEED	Register containing maximum speed for CONTROLLED_STOP action with a range of 0-124,5 km/h encoded onto 0x00-0xF9, scaling of 0,5 km/h per bit and zero offset. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in the `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x83	SLOW_DOWN_ MAX_SPEED	Register containing maximum speed for SLOW_DOWN action with a range of 0-124,5 km/h encoded onto 0x00-0xF9, scaling of 0,5 km/h per bit and zero offset. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x84	MAX_FORWARD_ GEAR	Register containing maximum forward gear in the range 0-12 corresponding to the enumeration values in Table 47. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x85	MAX_REVERSE_ GEAR	Register containing maximum reverse gear in the range 0-12 corresponding to the enumeration values in Table 47. — REG_TYPE = PARAMETER — REG_ATTRIB = READ_ONLY — REG_FORMAT = J1939 The machine can indicate if this register is available by returning REG_DEF = DEFINED in `SPN:RegisterFormat` in `PGN:Machine»CxDreply`.
0x86-0xDF	-	Reserved for future use by the ISO 21815 series.
0xE0-0xEF	-	User defined (available for use).
0xF0-0xF9	-	Common registers (see Table 13).
INDEX values in the range 0x00-0xEF may return different values for each subsystem specified in SPN:Status.		

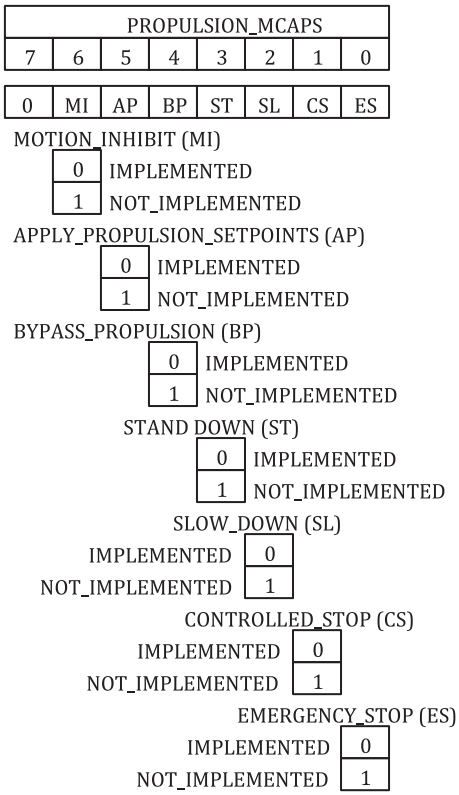


Figure 11 — bitfield codes in register PROPULSION_MCAPS

EXAMPLE

`SPN:Status` = 0x00 (SUB_SYSTEM=PROPULSION)

`SPN:RegisterIndex` = 0x81 (INDEX=EMERGENCY_STOP_MAX_SPEED)

– index of register containing maximum speed for the EMERGENCY_STOP action in the PROPULSION subsystem.

7.2.3.3 SPN:RegisterSelect

This SPN defines how registers are to be individually addressed or for multiple setpoint registers with REG_TYPE = SET_POINT to be applied simultaneously. The selection of multiple registers is enabled by the SELECT bitfield to allow a range of matching criteria to be applied to registers.

NOTE This mechanism allows for an arbitrary number of setpoints to be activated using a single command (see description for bitfield code LOAD_PROPULSION_SETPOINTS in Table 11 for the PROPULSION subsystem).

The structure of `SPN:RegisterSelect` is shown in Figure 12.

The TAG bitfield is a 4-bit unsigned value used in conjunction with the SELECT bitfield to:

a) specify or modify a TAG value for a setpoint register (REG_TYPE = SET_POINT) with supported 8-bit or 16-bit numeric data types (REG_FORMAT = J1939, INT8, UINT8, SHORT16, USHORT16);

b) select multiple setpoints with a matching non-zero TAG value to be applied by `SPN:Command` (SELECT= MATCH_TAG) in `PGN:CxD»MachineCommand`;

c) remove the setpoint from being selected by `SPN:Command` (SELECT= MATCH_TAG) by specifying a zero TAG value.

The TAG value for each of the supported data types should be provided by the machine in the `SPN:RegisterValue3` when the setpoint register value is read (see Table 36).

The valid enumerations for the SELECT, TAG, COUNT, OFFSET and RSZ bitfields in `SPN:RegisterSelect` are defined in Table 19 and Table 20.

Table 19 — RSZ bitfield codes in `SPN:RegisterSelect`

RSZ	Name	Description
0b0	-	Reserved for future use by the ISO 21815 series
0b1	-	Reserved for future use by the ISO 21815 series

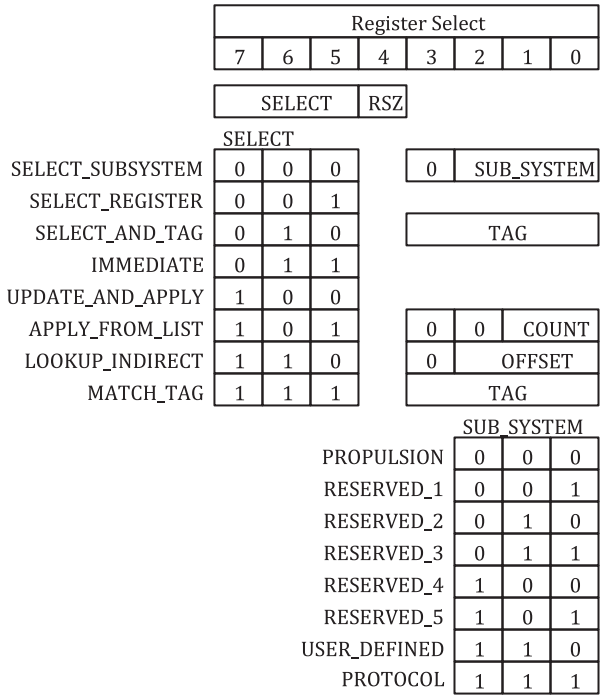


Figure 12 — Structure of `SPN:RegisterSelect` byte

Table 20 — SELECT bitfield codes in `SPN:RegisterSelect`

SELECT	Name	Description
0b000	SELECT_ SUBSYSTEM	Use this code to specify an operation on the subsystem specified in the SUB_SYSTEM bitfield of `SPN:RegisterSelect`. EXAMPLE: `SPN:Status` (ACT7 = RESET_REGISTERS) `SPN:RegisterSelect` (SELECT = SELECT_SUBSYSTEM, SUB_SYSTEM=PROPULSION) — resets the registers in the PROPULSION subsystem to factory default values
0b001	SELECT_ REGISTER	Use this code to specify a read operation on the register index specified in `SPN:RegisterIndex`. Refer to 7.2.3.4 to 7.2.3.7 for a description of register values and the corresponding format.

Table 20 *(continued)*

SELECT	Name	Description
0b010	SELECT_AND_TAG	Use this code to select a single register specified by `SPN:RegisterIndex` and set the TAG value for registers with REG_TYPE = SET_POINT (see Table 34) for certain data formats (see Table 36) to enable execution of multiple matching setpoints using `PGN:CxD»MachineCommand` in conjunction with `SPN:Command`=(CMD7 = APPLY_PROPULSION_SETPOINTS). Setting the TAG bitfield to zero disables matching functionality for the register. The value in `SPN:RegisterValue0..3` is used for registers in the extended index range. NOTE: Setpoints cannot be applied to registers in the extended index range.
0b011	IMMEDIATE	Use this code to perform an immediate operation via `SPN:Command`.
0b100	UPDATE_AND_ APPLY	Use this code to prepare to apply the setpoint associated with the register index specified in `SPN:RegisterIndex` to the value in: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` – most significant byte See 7.2.3.4 to 7.2.3.7 for a description of register values and the corresponding format. The APPLY_PROPULSION_SETPOINTS_CONFIRM enquiry in `SPN:Command` may be used to confirm proper execution of the setpoint. Use the APPLY_PROPULSION_SETPOINTS action to execute the setpoint. NOTE: This select code can only be used for registers with REG_TYPE = SET_POINT.
0b101	APPLY_FROM_ LIST	Use this code to prepare the selection of up to four (4) setpoints associated with the register indexes specified in: — `SPN:RegisterValue0` – first register index — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` See 7.2.3.4 to 7.2.3.7 for a description of register values. The number of setpoints to apply is COUNT+1 in the least significant two (2) bits of `SPN:RegisterSelect`. EXAMPLE: `SPN:RegisterSelect` = 0xA2 (COUNT=2) `SPN:RegisterValue0` = 0x01 (MIN_BRAKING) `SPN:RegisterValue1` = 0x02 (MAX_THROTTLE) `SPN:RegisterValue2` = 0x03 (MAX_SPEED) `SPN:RegisterValue3` = ignore prepares the three (3) specified setpoints

Table 20 *(continued)*

SELECT	Name	Description
		The APPLY_PROPULSION_SETPOINTS_CONFIRM enquiry in PGN:CxD»MachineCommand may be used to confirm proper execution of these setpoints. Use the APPLY_PROPULSION_SETPOINTS action to execute these setpoints. NOTE: This select code can only be used for registers with REG_TYPE = SET_POINT.
0b110	LOOKUP_ INDIRECT	Use this code to prepare the selection of multiple setpoints associated with a contiguous range of registers in the range [OFFSET*32,OFFSET*32+31] and the 32-bit pattern in: — SPN:RegisterValue0 – least significant byte — SPN:RegisterValue1 — SPN:RegisterValue2 — SPN:RegisterValue3 – most significant byte where a '1' bit value applies the corresponding setpoint and a '0' bit value leaves the setpoint unchanged and unapplied. The APPLY_PROPULSION_SETPOINTS_CONFIRM enquiry in PGN:CxD»MachineCommand may be used to confirm proper execution of these setpoints. Use the APPLY_PROPULSION_SETPOINTS action to execute these setpoints. This code may be used to apply up to thirty two (32) setpoints at once within a contiguous range of thirty two (32) index registers. Example: — SPN:RegisterSelect = 0xC3 (OFFSET=3) — SPN:RegisterValue0 – 0b0100 0000 (bit 6 set) — SPN:RegisterValue1 – 0b1000 0100 (bits 2,7 set) — SPN:RegisterValue2 – 0b0000 0000 (no bits set) — SPN:RegisterValue3 – 0b0010 0000 (bit 5 set) applies setpoints associated with up to 32 registers in the index range 96…127 corresponding to 32-bit positions: Register Value_0: 3*32+(0*8)+6 = 102 = 0x66 (index range 96-103) Register Value_1: 3*32+(1*8)+2 =106 = 0x6A (index range 104-111) Register Value_2: 3*32+(1*8)+7 =111 = 0x6F (index range 112-119) Register Value_3: 3*32+(3*8)+5 =125 = 0x7D (index range 120-127) NOTE: This select code can only be used for registers with REG_TYPE = SET_POINT.

Table 20 *(continued)*

SELECT	Name	Description
0b111	MATCH_TAG	Use this select code to prepare to apply all setpoints with a non-zero TAG matching the value in the TAG bitfield. The TAG values should be set by the CxD for all applicable setpoint registers in advance via the SELECT_AND_TAG code. The APPLY_PROPULSION_SETPOINTS_CONFIRM enquiry in `PGN:CxD»MachineCommand` may be used to confirm proper execution of these setpoints. Use the APPLY_PROPULSION_SETPOINTS action to execute these setpoints. The TAG value can be verified by the CxD by reading the least significant 4-bits of `SPN:RegisterValue3` via `SPN:Status` for supported numeric data formats (see Table 36). NOTE: This code can only be used for setpoint registers REG_TYPE = SET_POINT.

EXAMPLE

`SPN:Status` = 0x06 (ACTION, SUB_SYSTEM=PROPULSION, ACT0= SET_PROPULSION_REGISTER)

`SPN:RegisterIndex` = 0x02 (INDEX=MAX_THROTTLE)

`SPN:RegisterSelect` = 0x00 (SELECT=SELECT_AND_TAG, TAG=0b0001)

`SPN:RegisterValue0` = 0x64 (50 %)

– set the throttle register of the machine to a value of 100 (corresponding to 50 % throttle). The machine responds with the actual throttle position in `SPN:RegisterValue0`. Note that this instruction does not affect the actual throttle position until the setpoint is applied via `SPN:Command`.

7.2.3.4 SPN:RegisterValue0

This SPN contains the least significant byte of register value specified by `SPN:RegisterIndex` using one of the formats specified in Table 36.

NOTE The value ranges for the registers in the PROTOCOL and PROPULSION subsystems are defined in Table 13, Table 14 and Table 18.

7.2.3.5 SPN:RegisterValue1

This SPN contains the next significant byte of register value specified by `SPN:RegisterIndex` using one of the formats specified in Table 36.

7.2.3.6 SPN:RegisterValue2

This SPN contains the next significant byte of register value specified by `SPN:RegisterIndex` using one of the formats specified in Table 36.

7.2.3.7 SPN:RegisterValue3

This SPN contains the most significant byte of register value specified by `SPN:RegisterIndex` using one of the formats specified in Table 36.

7.2.3.8 SPN:MessageIdentifier

The message identifier shall be used to provide an 8-bit signature as a communication integrity check. The message identifier may be an incrementing counter 0-255 with rollover to 0 after 255 or a security signature.

The security signature should provide the same or higher level of integrity as an incrementing counter.

Messages shall be sent in sequence by the device connected to the J1939 interface issuing the action / enquiry instructions. The identifier provided by the CxD should be used as a reference in the reply PGN provided by the machine.

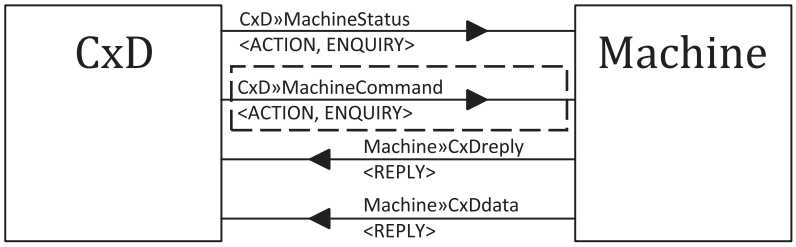
7.3 PGN:CxD»MachineCommand

7.3.1 General

The structure of `PGN:CxD»MachineCommand` is shown in Figure 13. Additional details are provided in 7.3.2 and 7.3.3.

PGN

CxD»Machine Command



SPNs for **PGN:CxD»MachineCommand** (see Table 22)

SPN	Offset (byte.bit)	SPN length
Command	1.1	8 bits
Register Index[a]	2.1	8 bits
Register Select[b]	3.1	8 bits
Register Value_0	4.1	8 bits
Register Value_1	5.1	8 bits
Register Value_2	6.1	8 bits
Register Value_3	7.1	8 bits
Message Identifier	8.1	8 bits

Notes: (a) see 7.2.3.2
(b) see 7.2.3.3

Reserved SPN values (see Table 6)

Code	MESSAGE_STATUS
0xFA	GENERAL_FAULT
0xFE	GENERAL_ERROR
0xFF	IGNORE

Note: Reserved SPN values may be used in bytes 2, 3 and only for registers with J1939 data format in bytes 4, 5, 6, 7. Reserved SPN values should not be used for the Message Identifier (byte 8).

Command

7	6	5	4	3	2	1	0

ENQUIRY

1	SUB_SYSTEM

ACTION

0	SUB_SYSTEM

SUB_SYSTEM

0	0	0	PROPULSION
0	0	1	RESERVED_1
0	1	0	RESERVED_2
0	1	1	RESERVED_3
1	0	0	RESERVED_4
1	0	1	RESERVED_5
1	1	0	USER_DEFINED
1	1	1	PROTOCOL

Command (SUB_SYSTEM=PROPULSION)

7	6	5	4	3	2	1	0

ENQUIRY

1	0	0	0	INH	CHK0

ACTION

0	0	0	0	INH	XEQ0

INHIBIT_COMMAND (INH)

0	OFF
1	ON

CHK0 bitfield codes in PROPULSION subsystem – **SPN:Command** (see Table 24)

CHK0	Name
0b001	EMERGENCY_STOP_CONFIRM
0b010	CONTROLLED_STOP_CONFIRM
0b011	SLOW_DOWN_CONFIRM
0b100	STAND_DOWN_CONFIRM
0b101	BYPASS_PROPULSION_CONFIRM
0b110	APPLY_PROPULSION_SETPOINTS _CONFIRM

XEQ0 bitfield codes in PROPULSION subsystem – **SPN:Command** (see Table 25)

XEQ0	Name
0b000	NORMAL_OPERATION
0b001	EMERGENCY_STOP
0b010	CONTROLLED_STOP
0b011	SLOW_DOWN
0b100	STAND_DOWN
0b101	BYPASS_PROPULSION
0b110	APPLY_PROPULSION_SETPOINTS

Figure 13 — Structure of `PGN:CxD»MachineCommand`

Refer to Annex A for typical communication sequences and Table A.1 and Table A.2 for examples of CxD instruction and machine responses.

7.3.2 PGN description

This CxD shall send actions to the machine via the PGN with the parameters shown in Table 21.

Table 21 — `PGN:CxD»MachineCommand` parameters

Parameter	Description
PGN number	61969
PGN type	PDU2_FAST
Acronym	CXD2
Data length	8 bytes
Multipacket message	No
Broadcast rate	100 ms
Message priority	1

The SPN's associated with `PGN:CxD»MachineCommand` are listed in Table 22.

Table 22 — SPNs for `PGN:CxD»MachineCommand`

SPN	Offset (byte.bit)	SPN length	SPN Description
Command	1.1	8 bits	Action or enquiry sent from CxD to the machine
Register index	2.1	8 bits	Index to machine register
Register select	3.1	8 bits	Used in association with `SPN:RegisterIndex` to allow multiple setpoints to be applied in a single instruction from the CxD to the machine
Register value_0	4.1	8 bits	Least significant byte of a unique 32-bit token
Register value_1	5.1	8 bits	Next significant byte of a unique 32-bit token
Register value_2	6.1	8 bits	Next significant byte of a unique 32-bit token
Register value_3	7.1	8 bits	Most significant byte of a unique 32-bit token
Message identifier	8.1	8 bits	Incrementing counter

NOTE Unless specified otherwise, least-significant-first byte ordering is used for multi-byte transfers with individual bytes preserving J1939 bit ordering (big-endian).

The use of tokens in `SPN:RegisterValue_0..3` to secure each command is described in Table 8. If tokens have not been implemented the machine manufacturer should define the expected contents of these registers to enable execution of the function selected by `SPN:Command`.

The standard SAE J1939 reserved codes (see Table 6) may be used in `SPN:RegisterIndex` (offset = 2), `SPN:RegisterSelect` (offset = 3) and where noted in 7.3.3. Generally, the reserved codes are used to indicate missing data or an error. The use of these reserved codes in `SPN:Command` (offset = 1) and `SPN:MessageIdentifier` (offset = 8) is not permitted.

The machine shall acknowledge all actions or enquiries issued by the CxD via `PGN:CxD»MachineCommand` within the period corresponding to the broadcast rate specified in Table 22 once negotiation has been successfully completed.

The actions issued by the CxD apply only for the duration of each message and are updated at the broadcast rate specified by the PGN parameters (see Table 21).

`PGN:CxD»MachineCommand` is not currently used by the PROTOCOL subsystem.

Each SPN within `PGN:CxD»MachineCommand` is defined in more detail in 7.3.3.

7.3.3 SPN structure

7.3.3.1 SPN:Command

7.3.3.1.1 General

This SPN is used by the CxD to initiate an action or enquiry to the machine. Refer to Figure 14 for a description of the bitfields in this value item.

The typical negotiation, initialisation and operation message sequences for this SPN are described in Annex A.

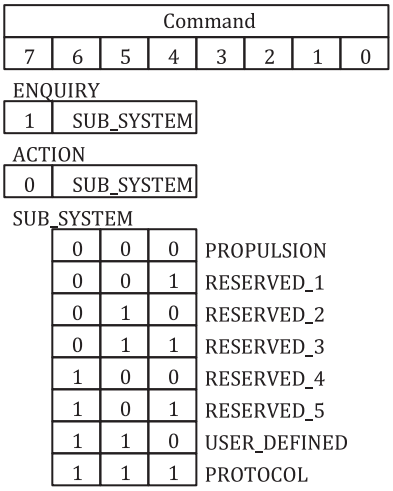


Figure 14 — Bit-fields in `SPN:Command`

The valid enumerations for the SUB_SYSTEM bitfield are listed in Table 7.

The least significant 4-bits of `SPN:Command` are defined separately for each subsystem.

7.3.3.1.2 SUB_SYSTEM: PROPULSION

The format of `SPN:Command` for the PROPULSION subsystem is shown in Figure 15.

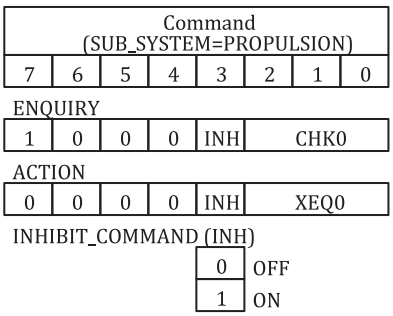


Figure 15 — PROPULSION subsystem bitfields – `SPN:Command`

The valid enumerations for `SPN:Command` in the PROPULSION subsystem are defined in Table 23 to Table 25. The reserved message status codes (see Table 6) should not be used for this SPN.

Table 23 — INHIBIT_COMMAND (INH) bitfield codes in PROPULSION subsystem – `SPN:Command`

INH	Name	Description
0b0	OFF	Instructs the machine to release motion inhibit and allow the propulsion system to re-engage.
0b1	ON	Instruct the machine to engage motion inhibit and prevent engagement of the propulsion system while the machine remains stationary. The CxD should check the INR flag of `SPN:Status/Command` in `PGN:Machine»CxDreply` to verify if motion inhibit functionality is available on the machine. The machine may refuse to respond to a motion inhibit instruction (INH=1) sent by the CxD when already moving.

The inhibit command (INH) may be applied independently of other enquiries or actions defined by the CHK0 or XEQ0 bitfields. For example, the machine may be held in a motion inhibit state while confirming the capability of the machine.

The current motion inhibit status of the machine may be provided by the machine manufacturer via the MI bitfield defined in Table 45.

Table 24 — CHK0 bitfield codes in PROPULSION subsystem – `SPN:Command`

CHK0	Name	Description
0b000	RESERVED_0	Reserved for future use by the ISO 21815 series.
0b001	EMERGENCY_STOP_CONFIRM	This enquiry is used by the CxD to verify if the machine can respond to an EMERGENCY_STOP command (see Table 25). NOTE: This enquiry does not initiate the EMERGENCY_STOP action on the machine. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)
0b010	CONTROLLED_STOP_CONFIRM	This enquiry is used by the CxD to verify if the machine can respond to a CONTROLLED_STOP command (see Table 25). NOTE: This enquiry does not initiate the CONTROLLED_STOP action on the machine. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)
0b011	SLOW_DOWN_CONFIRM	This enquiry is used by the CxD to verify if the machine can respond to an SLOW_DOWN command (see Table 25). NOTE: This enquiry does not initiate the SLOW_DOWN action on the machine). The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)
0b100	STAND_DOWN_CONFIRM	This enquiry is used by the CxD to verify if the machine can respond to a STAND_DOWN command (see Table 25). NOTE: This enquiry does not initiate the STAND_DOWN action on the machine). The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)

Table 24 *(continued)*

CHK0	Name	Description
0b101	BYPASS_ PROPULSION_ CONFIRM	This enquiry is used by the CxD to verify if the machine can respond to a BYPASS_PROPULSION action (see Table 25). NOTE: This enquiry does not initiate the BYPASS_PROPULSION action on the machine. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)
0b110	APPLY_ PROPULSION_ SETPOINTS_ CONFIRM	This enquiry is used by the CxD to verify if the machine can execute the APPLY_PROPULSION_SETPOINTS action based on the values in `SPN:RegisterIndex` and `SPN:RegisterSelect`. If the select condition matches multiple setpoints the machine shall verify that every matching setpoint can be applied simultaneously. NOTE: This enquiry does not initiate the APPLY_PROPULSION_SETPOINTS action on the machine. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply` (see Table 31 for responses)
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

Table 25 — XEQ0 bitfield codes in PROPULSION subsystem – `SPN:Command`

XEQ0	Name	Description
0b000	NORMAL_ OPERATION	The NORMAL_OPERATION action is sent by the CxD to instruct the machine to continue normal operation. This action also has the effect of cancelling any other action in the XEQ0 code group that is already in progress. NOTE: This action has no effect on any setpoints that have been loaded by the LOAD_PROPULSION_SETPOINTS action sent via `SPN:Status`. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b001	EMERGENCY_ STOP	The EMERGENCY_STOP action is sent by CxD to instruct the machine to implement the emergency stop sequence defined by the machine control system. The intent of this command is to stop the machine motion as rapidly as possible to reduce the consequence level, if the CxD logic determines that a collision is imminent. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b010	CONTROLLED_ STOP	The CONTROLLED_STOP action is sent by CxD to instruct the machine to implement the controlled stop sequence defined by the machine control system. The intent of this command is to stop the machine motion in a controlled / conventional manner when the CxD logic determines that a collision / interaction can be avoided by slowing down and stopping. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b011	SLOW_DOWN	The SLOW_DOWN action is sent by the CxD to reduce the speed of the machine in a controlled / conventional manner as defined by the machine control system. The intent of this command is to slow down the machine when the CxD logic determines that a collision / interaction can be avoided by reducing speed. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.

Table 25 *(continued)*

XEQ0	Name	Description
0b100	STAND_DOWN	This action is sent by CxD to instruct the machine to enter a state of limited functionality where the desired end state is a stationary machine. This command can be used when an immediate response by the machine to an EMERGENCY_STOP, CONTROLLED_STOP or SLOW_DOWN action is not suitable. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b101	BYPASS_ PROPULSION	The BYPASS_PROPULSION action is sent by the CxD when propulsion has been bypassed on the CxD side of the interface and CxD actions are inactive. This action also has the effect of cancelling any other action in the XEQ0 code group that is already in progress. This action should clear any setpoints that have been loaded by the LOAD_PROPULSION_SETPOINTS action sent via `SPN:Status`. Refer to Annex C for additional information on the implementation of this action for standby mode and the use of the dedicated override switch contacts described in 6.4. The machine can indicate if a capability to respond to this action is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`.
0b110	APPLY_ PROPULSION_ SETPOINTS	This action is used to initiate the application of all propulsion setpoints matching the criteria previously specified by the LOAD_PROPULSION_SETPOINTS action sent via `PGN:CxD»MachineStatus`. The machine can indicate if this capability is available by returning the appropriate `SPN:Status/Command` in `PGN:Machine»CxDreply`. NOTE: This action can result in multiple setpoints being applied simultaneously.
0b111	RESERVED_7	Reserved for future use by the ISO 21815 series.

EXAMPLE 1

`SPN:Command` = 0x02 (ACTION, SUB_SYSTEM=PROPULSION, INH=OFF, XEQ0=CONTROLLED_STOP)

– initiate a controlled stop action defined by the machine.

EXAMPLE 2

`SPN:Command` = 0x0E (ACTION, SUB_SYSTEM=PROPULSION, INH=ON, XEQ0=APPLY_PROPULSION_SETPOINTS)

`SPN:RegisterSelect` = 0xE1 (SELECT=MATCH_TAG, TAG=0b0001)

– apply all setpoint registers with the least significant bit of the mask set while machine is kept in a stationary state (motion inhibit).

EXAMPLE 3

`SPN:Status` = 0x83 (ENQUIRY, SUB_SYSTEM=PROPULSION, ENQ0=SLOW_DOWN_CONFIRM)

– verify if the PROPULSION subsystem of the machine can respond to a slow down action during initialisation (while machine is stationary) with motion inhibited.

7.3.3.2 SPN:RegisterIndex

See 7.2.3.2.

7.3.3.3 SPN:RegisterSelect

See 7.2.3.3.

7.3.3.4 SPN:RegisterValue0

All instructions sent by the CxD to the machine via the `SPN:command` channel should be secured with a 32-bit token in `SPN:RegisterValue0..3` that is synchronised between the machine and the CxD during negotiation.

Where required by the machine manufacturer, the token should change with every message sent by the CxD to the machine in a pseudo-random sequence. The token sequence should be randomised by the initial seed provided by machine and ordered by a shared secret code that is known independently of the interface by the machine and the CxD.

7.3.3.5 SPN:RegisterValue1

See 7.3.3.4.

7.3.3.6 SPN:RegisterValue2

See 7.3.3.4.

7.3.3.7 SPN:RegisterValue3

See 7.3.3.4.

7.3.3.8 SPN:MessageIdentifier

The message identifier shall be used to provide an 8-bit signature as a communication integrity check. The message identifier may be an incrementing counter 0-255 with rollover to 0 after 255 or a security signature.

The security signature should provide the same or higher level of integrity as an incrementing counter.

Messages shall be sent in sequence by the device connected to the J1939 interface issuing the action / enquiry instructions.

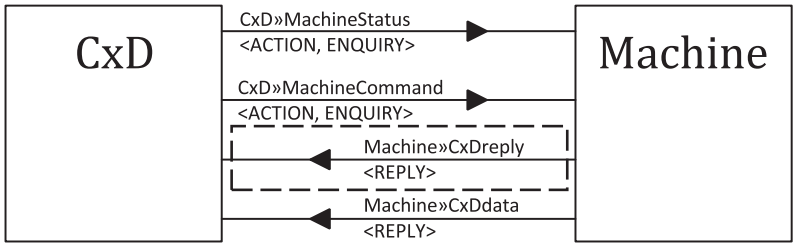
7.4 PGN:Machine»CxDreply

7.4.1 General

The structure of `PGN:CxD»MachineCommand` is shown in Figure 16. Additional details are provided in 7.4.2 and 7.4.3.

PGN

Machine»CxDreply



SPNs for **PGN:Machine»CxDreply** (see Table 27)

SPN	Offset (byte.bit)	SPN length
Status / Command	1.1	8 bits
Register Index[a]	2.1	8 bits
Register Format[b]	3.1	8 bits
Register Value_0	4.1	8 bits
Register Value_1	5.1	8 bits
Register Value_2	6.1	8 bits
Register Value_3	7.1	8 bits
Message Identifier	8.1	8 bits

Notes: (a) see 7.2.3.2
(b) see 7.4.3.3

Reserved SPN values (see Table 6)

Code	MESSAGE_STATUS
0xFA	GENERAL_FAULT
0xFE	GENERAL_ERROR
0xFF	IGNORE

Note: Reserved SPN values may be used in bytes 2, 3 and only for registers with J1939 data format in bytes 4, 5, 6, 7. Reserved SPN values should not be used for the Message Identifier (byte 8).

Status / Command

7	6	5	4	3	2	1	0

REPLY_TO_ENQUIRY

1	SUB_SYSTEM

REPLY_TO_ACTION

0	SUB_SYSTEM

SUB_SYSTEM

0	0	0	PROPULSION
0	0	1	RESERVED_1
0	1	0	RESERVED_2
0	1	1	RESERVED_3
1	0	0	RESERVED_4
1	0	1	RESERVED_5
1	1	0	USER_DEFINED
1	1	1	PROTOCOL

Status / Command (SUB_SYSTEM=PROTOCOL)

7	6	5	4	3	2	1	0

REPLY_TO_ENQUIRY

1	1	1	1	0	RTE7

REPLY_TO_ACTION

0	1	1	1	0	RTA7

Status / Command (SUB_SYSTEM=PROPULSION)

7	6	5	4	3	2	1	0

REPLY_TO_ENQUIRY

1	0	0	0	INR	RTE0

REPLY_TO_ACTION

0	0	0	0	INR	RTA0

INHIBIT_RESPONSE (INR)

0	OFF
1	ON

RTE7 bitfield enumerations - **SPN:Status/Command** (see Table 28)

RTE7	Name
0b000	PROTOCOL_ACK
0b001	NEGOTIATE_ACK
0b010	NEGOTIATE_REPLY
0b011	RENEGOTIATE_REPLY
0b101	GET_EXTENDED_REGISTER_OK
0b110	GET_REGISTER_OK
0b111	ENQUIRY_ERROR

RTA7 bitfield enumerations - **SPN:Status/Command** (see Table 29)

RTA7	Name
0b101	RESET_REGISTERS_OK
0b110	SET_REGISTER_OK
0b111	ACTION_ERROR

RTE0 bitfield codes in PROPULSION subsystem - **SPN:Status/Command** (see Table 31)

RTE0	Name
0b000	PROPULSION_ACK
0b001	EMERGENCY_STOP_CONFIRM_ACK
0b010	CONTROLLED_STOP_CONFIRM_ACK
0b011	SLOW_DOWN_CONFIRM_ACK
0b100	STAND_DOWN_CONFIRM_ACK
0b101	BYPASS_PROPULSION_CONFIRM_ACK
0b110	APPLY_PROPULSION_SETPOINTS _CONFIRM_ACK
0b111	ENQUIRY_ERROR

RTA0 bitfield codes in PROPULSION subsystem - **SPN:Status/Command** (see Table 32)

RTA0	Name
0b000	PROPULSION_ACK
0b001	EMERGENCY_STOP_ACK
0b010	CONTROLLED_STOP_ACK
0b011	SLOW_DOWN_ACK
0b100	STAND_DOWN_ACK
0b101	BYPASS_PROPULSION_ACK
0b110	APPLY_PROPULSION_SETPOINTS_ACK
0b111	ACTION_ERROR

Figure 16 — Structure of `PGN:Machine»CxDreply`

The machine shall respond to a message issued by the CxD in `PGN:CxD»MachineStatus` or `PGN:CxD»MachineCommand` with the PGN defined in 7.4.2. The contents of this PGN are a reflection of the information sent by the CxD with an indication of the success or failure of the action or enquiry.

Refer to Annex A for typical communication sequences and Table A.1 and Table A.2 for examples of CxD instruction and machine responses.

Note that data fields may be modified to contain status conditions as per Table 6. Additionally, the machine may respond by indicating the control inputs it is capable of enacting, it is the responsibility of the CxD to react in an appropriate manner.

7.4.2 PGN description

The machine shall respond to actions or enquiries issued by the CxD over the J1939 on-board communication interface via the PGN with the parameters shown in Table 26.

Table 26 — `PGN:Machine»CxDreply` parameters

Parameter	Description
PGN number	64204
PGN type	PDU2_SLOW
Acronym	CXD3
Data length	8 bytes
Multipacket message	No
Broadcast rate	On request
Message priority	Current value

The machine may limit the rate of instructions sent by the CxD by delaying the response via this PGN.

The maximum delay shall not exceed 100 ms.

The SPN's associated with `PGN:Machine»CxDreply` are shown in Table 27.

Table 27 — SPNs for `PGN:Machine»CxDreply`

SPN	Offset (byte.bit)	SPN length	SPN Description
Status / command	1.1	8 bits	Response from machine to action or enquiry sent by CxD
Register index	2.1	8 bits	Index to machine register
Register format	3.1	8 bits	Format information for machine register at index specified in `SPN:Status` of `PGN:CxD»MachineStatus`
Register value_0	4.1	8 bits	Least significant byte of register value specified by `SPN:RegisterIndex`
Register value_1	5.1	8 bits	Next significant byte of register value specified by `SPN:RegisterIndex`
Register value_2	6.1	8 bits	Next significant byte of register value specified by `SPN:RegisterIndex`
Register value_3	7.1	8 bits	Most significant byte of register value specified by `SPN:RegisterIndex`
Message identifier	8.1	8 bits	Incrementing counter

NOTE Unless specified otherwise, least-significant-first byte ordering is used for multi-byte transfers with individual bytes preserving J1939 bit ordering (big-endian).

The standard SAE J1939 reserved codes (see Table 6) may be used in `SPN:RegisterIndex` (offset = 2) and `SPN:RegisterFormat` (offset = 3) and where noted in 7.4.3. Generally, the reserved codes are used to indicate missing data or an error. The use of these reserved codes in `SPN:Status/Command` (offset = 1) and `SPN:MessageIdentifier` (offset = 8) is not permitted.

Each SPN within `PGN:Machine»CxDreply` is defined in more detail in 7.4.3.

7.4.3 SPN structure

7.4.3.1 SPN:Status/Command

7.4.3.1.1 General

This SPN mirrors the structure of `SPN:Status` and `SPN:Command` with the format shown in Figure 17.

The typical negotiation, initialisation and operation message sequences for this SPN are described in Annex A.

Status / Command

7	6	5	4	3	2	1	0

REPLY_TO_ENQUIRY

1	SUB_SYSTEM

REPLY_TO_ACTION

0	SUB_SYSTEM

SUB_SYSTEM

0	0	0	PROPULSION
0	0	1	RESERVED_1
0	1	0	RESERVED_2
0	1	1	RESERVED_3
1	0	0	RESERVED_4
1	0	1	RESERVED_5
1	1	0	USER_DEFINED
1	1	1	PROTOCOL

Figure 17 — Bit-fields in status / command SPN byte

The valid enumerations for the SUB_SYSTEM bitfield are listed in Table 7.

The least significant 4-bits of `SPN:Status/Command` are defined separately for each subsystem.

7.4.3.1.2 SUB_SYSTEM: PROTOCOL

The format of `SPN:Status/Command` for the PROTOCOL subsystem is shown in Figure 18.

Status / Command (SUB_SYSTEM=PROTOCOL)

7	6	5	4	3	2	1	0

REPLY_TO_ENQUIRY

1	1	1	1	0	RTE7

REPLY_TO_ACTION

0	1	1	1	0	RTA7

Figure 18 — PROTOCOL subsystem structure

The valid enumerations of the RTE7 bitfield for the PROTOCOL subsystem are listed in Table 28 and Table 29.

Table 28 — RTE7 bitfield enumerations - `SPN:Status/Command`

RTE7	Name	Usage in reply to: `PGN:CxD»MachineStatus`
0b000	PROTOCOL_ACK	This reply is provided by the machine after completion of negotiation in response to the PROTOCOL_NOP enquiry issued by the CxD to indicate that both sides of the interface (CxD and machine) are operating normally and the CxD is active. The machine shall respond to a PROTOCOL_NOP enquiry with: a) PROTOCOL_ACK – to acknowledge that the machine side of the interface is healthy after the credentials of the CxD have been verified by the machine. b) NEGOTIATE_ REPLY – if the machine requires the CxD to negotiate its credentials via `SPN:Status` (ENQ7= NEGOTIATE_ENQ) c) RENEGOTIATE_ REPLY – if the credential information provided by the CxD should be renegotiated. Refer to Table 8 for typical negotiation sequences. The machine may force the CxD to reinitiate the negotiation sequence if no valid instructions are sent within the time specified in register INSTRUCTION_TIMEOUT of the protocol subsystem.
0b001	NEGOTIATE_ACK	This reply is provided by the machine as a response to the NEGOTIATE_NOP enquiry issued by the CxD. Refer to Table 8 for typical negotiation sequences. The exchange of these messages indicates that both sides of the communication interface (CxD and machine) are operating normally and the CxD is active. The machine shall always respond to the NEGOTIATE_NOP enquiry whether or not negotiation has been completed successfully by the CxD.
0b010	NEGOTIATE_REPLY	This reply is used by the machine to acknowledge CxD credentials or pass permission information back to the CxD. The negotiation sequence may pass up to 32-bits per message using one of the formats specified in Table 36. Refer to Table 8 for typical negotiation sequences. The machine may ignore all instructions issued by the CxD except NEGOTIATE_NOP and NEGOTIATE_ENQ if the negotiation sequence and credentials supplied by the CxD are not recognised.
0b011	RENEGOTIATE_ REPLY	This reply can be used by the machine in response to a NEGOTIATE_NOP enquiry or PROTOCOL_NOP enquiry from the CxD to indicate that the machine requires the CxD to negotiate or renegotiate its credentials. The machine may refuse to execute instructions issued by the CxD if any of the following apply: — if the CxD does not initiate a NEGOTIATE_ENQUIRY in response, or within a specified time period after the initial negotiation — when authentication is approved or declined, — when one side of the communication interface goes into reboot, — periodical re-authentication during operation to detect disconnection of a CxD and reconnection of a substitute The machine should avoid using the same bit patterns or codes for successive authentication or re-authentication sequences to prevent authorised access to machine control functions.
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series.
0b101	GET_EXTENDED_ REGISTER_OK	Enquiry to access an extended register completed successfully.

Table 28 *(continued)*

RTE7	Name	Usage in reply to: PGN:CxD»MachineStatus
0b110	GET_ REGISTER_OK	Enquiry to access a register completed successfully.
0b111	ENQUIRY_ERROR	Used if the machine encountered an error when responding to the enquiry issued by the CxD.

Table 29 — RTA7 bitfield enumerations - SPN:Status/Command

RTA7	Name	Usage in reply to: PGN:CxD»MachineStatus
0b000	RESERVED_0	Reserved for future use by the ISO 21815 series.
0b001	RESERVED_1	Reserved for future use by the ISO 21815 series.
0b010	RESERVED_2	Reserved for future use by the ISO 21815 series.
0b011	RESERVED_3	Reserved for future use by the ISO 21815 series.
0b100	RESERVED_4	Reserved for future use by the ISO 21815 series.
0b101	RESET_ REGISTERS_OK	Action to reset registers in the specified subsystem to factory default settings completed successfully.
0b110	SET_ REGISTER_OK	Action to set the value of a register completed successfully.
0b111	ACTION_ERROR	Used if the machine encountered an error when responding to the action issued by the CxD.

EXAMPLE 1

SPN:Status/Command = 0xF1 (REPLY_TO_ENQUIRY, SUB_SYSTEM=PROTOCOL, RTE7=NEGOTIATE_ACK)

EXAMPLE 2

SPN:Status/Command = 0x75 (REPLY_TO_ACTION, SUB_SYSTEM=PROTOCOL, RTA7=RESET_REGISTERS_OK)

7.4.3.1.3 SUB_SYSTEM: PROPULSION

The structure of the SPN:Status/Command byte for the PROPULSION subsystem is shown in Figure 19.

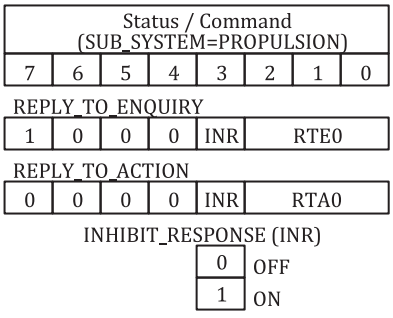


Figure 19 — Bitfields in PROPULSION subsystem - SPN:Status/Command

The valid enumerations for the INR, RTE0 and RTA0 bitfields of SPN:Status/Command in the PROPULSION subsystem are defined in Table 26 to Table 32.

Table 30 — INHBIT_RESPONSE (INR) bitfield codes in PROPULSION subsystem - `SPN:Status/Command`

INR	Name	reply to enquiry	reply to action
0b0	OFF	Machine does not have the capability to inhibit motion in response to an INHIBIT_COMMAND=ON/OFF action sent via the on-board communication interface. NOTE: It is possible that the motion inhibit capability is not available as a control function via the on-board communication interface on all machine types.	a) Machine acknowledges and executes INHIBIT_COMMAND=OFF action from the CxD, or b) Machine fails to execute a INHIBIT_COMMAND=ON action from the CxD If motion inhibit functionality is provided by the machine manufacturer the motion inhibit status of the machine should also be indicated (see 7.5.4.2)
0b1	ON	Machine has the capability to inhibit motion in response to an INHIBIT_COMMAND=ON/OFF action sent via the on-board communication interface.	Machine acknowledges and executes INHIBIT_COMMAND=ON instruction from the CxD and prevents motion of the machine while INR=1.
The motion inhibit status of the machine may be provided by the machine manufacturer via the MI bitfield defined in Table 46.			

Table 31 — RTE0 bitfield codes in PROPULSION subsystem - `SPN:Status/Command`

RTE0	Name	Description
0b000	PROPULSION_ACK	The machine acknowledges the enquiry from the CxD.
0b001	EMERGENCY_STOP_CONFIRM_ACK	The machine has the capability to perform an EMERGENCY_STOP to bring the machine to a stop by full application of brakes. NOTE: The machine response to an EMERGENCY STOP is defined by the machine manufacturer. The machine manufacturer may define a maximum speed where EMERGENCY_STOP can be applied (see Table 18).
0b010	CONTROLLED_STOP_CONFIRM_ACK	The machine has the capability to perform a CONTROLLED_STOP to reduce machine speed by application of a combination of slowing down and application of brakes. NOTE: The machine response to a CONTROLLED STOP is defined by the machine manufacturer. The machine manufacturer may define a maximum speed where CONTROLLED_STOP can be applied (see Table 18).
0b011	SLOW_DOWN_CONFIRM_ACK	The machine has the capability to perform a SLOW_DOWN to reduce the machine speed to a specified maximum speed. NOTE: The machine response to an SLOW_DOWN is defined by the machine manufacturer. The machine manufacturer may define a maximum speed where SLOW_DOWN can be applied (see Table 18).
0b100	STAND_DOWN_CONFIRM_ACK	The machine has the capability to perform a STAND_DOWN where the machine is brought to a stationary state within a specified period of time or while a specified condition exists (e.g. allow machine to relocate to parking area). NOTE: The specific machine response to a STAND_DOWN is defined by the machine manufacturer.
0b101	BYPASS_ PROPULSION_ CONFIRM_ACK	The machine has the capability to perform a BYPASS_ PROPULSION where the CxD instructions are ignored by the machine. NOTE: The machine response to a BYPASS_ PROPULSION is defined by the machine manufacturer.

Table 31 *(continued)*

RTE0	Name	Description
0b110	APPLY_ PROPULSION_ SETPOINTS_ CONFIRM_ACK	The machine has the capability to perform an APPLY_ PROPULSION_SETPOINTS based on the criteria specified by the CxD.
0b111	ENQUIRY_ERROR	Instruction issued by the CxD in `SPN:Status` is not available or cannot be executed.

Table 32 — RTA0 bitfield codes in PROPULSION subsystem - `SPN:Status/Command`

RTA0	Name	Description
0b000	PROPULSION_ACK	The machine acknowledges the action from the CxD.
0b001	EMERGENCY_ STOP_ACK	The machine acknowledges and executes the EMERGENCY_STOP command from the CxD.
0b010	CONTROLLED_ STOP_ACK	The machine acknowledges and executes the CONTROLLED_ STOP instruction from the CxD
0b011	SLOW_DOWN_ ACK	The machine acknowledges and executes the SLOW_DOWN instruction from the CxD.
0b100	STAND_DOWN_ACK	The machine acknowledges and executes the STAND_DOWN instruction from the CxD
0b101	BYPASS_ PROPULSION_ACK	The machine acknowledges and executes the BYPASS_ PROPULSION instruction from the CxD.
0b110	APPLY_ PROPULSION_ SETPOINTS_ ACK	The machine acknowledges and executes the APPLY_ PROPULSION_SETPOINTS instruction from the CxD
0b111	ACTION_ERROR	Instruction issued by the CxD in `SPN:Command` is not available or cannot be executed.

EXAMPLE 1

`SPN:Status/Command` = 0x82 (REPLY_TO_ENQUIRY, SUB_SYSTEM=PROPULSION, INR=OFF, RTE0=-CONTROLLED_STOP_CONFIRM_ACK)

– machine indicates it has a capability for controlled stop, but no capability for motion inhibit (INR=OFF).

EXAMPLE 2

`SPN:Status/Command` = 0x0E (REPLY_TO_ACTION, SUB_SYSTEM=PROPULSION, INH=ON, RTA0=APPLY_PROPULSION_SETPOINTS_ACK)

– apply propulsion setpoints configured previously by a LOAD_PROPULSION_SETPOINTS action; indicate the machine currently is in is motion inhibit state (INR=ON).

EXAMPLE 3

`SPN:Status` = 0x83 (ENQUIRY, SUB_SYSTEM=PROPULSION, CHK0=SLOW_DOWN_CONFIRM)

– verify if the PROPULSION subsystem of the machine can respond to a slow down action.

`SPN:Status/Command` = 0x83 (REPLY_TO_ENQUIRY, SUB_SYSTEM=PROPULSION, INR=OFF, RTE0=SLOW_DOWN_CONFIRM_ACK)

– indicate a capability for controlled stop Initiate a controlled stop action defined by the machine; the machine has no capability for motion inhibit (INR=OFF).

7.4.3.2 SPN:RegisterIndex

The machine shall reply to the CxD with the same information provided in `SPN:RegisterIndex` of either `PGN:CxD»MachineStatus` or `PGN:CxD»MachineCommand`.

Refer to 7.2.3.2 for a description of this SPN.

7.4.3.3 SPN:RegisterFormat

The structure of `SPN:RegisterFormat` is shown in Figure 20.

Register Format							
7	6	5	4	3	2	1	0
0	RD	RT	RA	REG_FORMAT			

Figure 20 — Bitfields in `SPN:RegisterFormat`

The valid enumerations for `SPN:Format` (SUB_SYSTEM=PROPULSION) are defined in Table 33 to Table 36.

Table 33 — REG_DEF (RD) bitfield - `SPN:RegisterFormat`

REG_DEF	Name	Description
0b0	DEFINED	Register index `SPN:RegisterIndex` is defined on the machine.
0b1	NOT_DEFINED	Register index `SPN:RegisterIndex` is not defined on the machine.
Enquiries or actions that refer to register addresses that are out of range should use the REG_DEF bit to indicate whether the register exists or not. The ENQUIRY_ERROR / ACTION_ERROR codes should only be used in the reply if the enquiry or action is improperly formed – e.g. with the wrong bitfield codes.		

Table 34 — REG_TYPE (RT) bitfield - `SPN:RegisterFormat`

REG_TYPE	Name	Description
0b0	PARAMETER	Register is a parameter on the machine.
0b1	SET_POINT	Register is a setpoint that can be modified by the CxD and applied simultaneously with other setpoints.

Table 35 — REG_ATTRIB (RA) bitfield - `SPN:RegisterFormat`

REG_ ATTRIB	Name	Description
0b0	READ_ONLY	Machine register can be read but cannot be modified by the CxD.
0b1	READ_WRITE	Machine register can be read and modified by the CxD.

Table 36 — REG_FORMAT bitfield - `SPN:RegisterFormat`

REG_ FORMAT	Name	Description
0x0	J1939	Data encoded as an unsigned byte in the range 0x00-0xF9 as an 8-bit value in: — `SPN:RegisterValue0` The reserved J1939 codes may be used to indicate the status of data contained in the register (see Table 6). J1939 registers with REG_TYPE = SET_POINT may optionally specify a TAG value (see 7.2.3.3) which may be accessed in the least significant four (4) bits of `SPN:RegisterValue3`.
The reserved SAE J1939 codes (see Table 6) should only be used for register values with REG_FORMAT = J1939.		

Table 36 *(continued)*

REG_ FORMAT	Name	Description
0x1	INT8	Data encoded as an 8-bit signed integer in: — `SPN:RegisterValue0` INT8 registers with REG_TYPE = SET_POINT may optionally specify a TAG value (see 7.2.3.3) which may be accessed in the least significant four (4) bits of `SPN:RegisterValue3`.
0x2	UINT8	Data encoded as an unsigned 8-bit signed integer in: — `SPN:RegisterValue0` UINT8 registers with REG_TYPE = SET_POINT may optionally specify a TAG value (see 7.2.3.3) which may be accessed in the least significant four (4) bits of `SPN:RegisterValue3`.
0x3	SHORT16	Data encoded as a signed 16-bit integer in SPNs: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` – most significant byte SHORT16 registers with REG_TYPE = SET_POINT may optionally specify a TAG value (see 7.2.3.3) which may be accessed in the least significant four (4) bits of `SPN:RegisterValue3`.
0x4	USHORT16	Data encoded as an unsigned 16-bit integer in SPNs: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` – most significant byte USHORT16 registers with REG_TYPE = SET_POINT may optionally specify a TAG value (see 7.2.3.3) which may be accessed in the most significant four (4) bits of `SPN:RegisterValue3`.
0x5	LONG32	Data encoded as a signed 32-bit integer in SPNs: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` – most significant byte NOTE: LONG32 registers do not support TAG values (see 7.2.3.3).
0x6	ULONG32	Data encoded as an unsigned 32-bit integer in SPNs: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` – most significant byte NOTE: ULONG32 registers do not support TAG values (see 7.2.3.3).
0x7	CHAR1	Data encoded as a 1-byte character in — `SPN:RegisterValue0` – byte offset 0 NOTE: CHAR1 registers with REG_TYPE = SET_POINT are not supported.
0x8	CHAR2	Data encoded as a 2-byte character — `SPN:RegisterValue0` – byte offset 0 (first character) — `SPN:RegisterValue1` – byte offset 1 NOTE: CHAR2 registers with REG_TYPE = SET_POINT are not supported.
The reserved SAE J1939 codes (see Table 6) should only be used for register values with REG_FORMAT = J1939.		

Table 36 *(continued)*

REG_ FORMAT	Name	Description
0x9	CHAR3	Data encoded as a 3-byte character — `SPN:RegisterValue0` – byte offset 0 (first character) — `SPN:RegisterValue1` – byte offset 1 — `SPN:RegisterValue2` – byte offset 2 NOTE: CHAR3 registers with REG_TYPE = SET_POINT are not supported.
0xA	CHAR4	Data encoded as a 4-byte character — `SPN:RegisterValue0` – byte offset 0 (first character) — `SPN:RegisterValue1` – byte offset 1 — `SPN:RegisterValue2` – byte offset 2 — `SPN:RegisterValue3` – byte offset 3 NOTE: CHAR4 registers with REG_TYPE = SET_POINT are not supported.
0xB	MULTI_BYTE	Data is encoded as a multi-byte array which is accessed via the J1939 multi-byte transfer mechanism with the 32-bit CRC value of the array (excluding header information) defined in: — `SPN:RegisterValue0` – least significant byte — `SPN:RegisterValue1` — `SPN:RegisterValue2` — `SPN:RegisterValue3` – most significant byte Examples: certificates, public keys, signatures, licence codes, XML data strings.
0xC	RESERVED_12	Reserved for future use by the ISO 21815 series.
0xD	RESERVED_13	Reserved for future use by the ISO 21815 series.
0xE	INVALID_DATA	Data for specified register is invalid (used with REG_DEF = DEFINED)
0xF	ERROR	Error in accessing specified register (used with REG_DEF = NOT_DEFINED).
The reserved SAE J1939 codes (see Table 6) should only be used for register values with REG_FORMAT = J1939.		

EXAMPLE

`SPN:RegisterIndex`= 0x01 (INDEX=MIN_BRAKING)

`SPN:RegisterFormat`= 0x30 (REG_DEF=DEFINED, REG_TYPE=SET_POINT, REG_ATTRIB= READ_WRITE, REG_FORMAT=J1939)

– confirm that the MIN_BRAKING setpoint register is defined, with READ_WRITE access attributes and has a J1939 format with the data provided in `SPN:RegisterValue0`.

7.4.3.4 SPN:RegisterValue0

The machine can reply to `PGN:CxD»MachineStatus` or `PGN:CxD»MachineCommand` instructions sent by the CxD with the following information in the `SPN:RegisterValue0..3` bytes:

a) confirmation of the register value provided by the CxD;

b) modified register value accepted by the machine (e.g. constrained to a value range).

In the case of an error or invalid data, the machine should return the ERROR or INVALID_DATA code in the REG_FORMAT bitfield of `SPN:RegisterFormat`.

7.4.3.5 SPN:RegisterValue1

This SPN is used for extended precision registers (16,32-bit) using one of the formats specified in Table 36.

7.4.3.6 SPN:RegisterValue2

This SPN is used for extended precision registers (32-bit) using one of the formats specified in Table 36.

7.4.3.7 SPN:RegisterValue3

This SPN is used for extended precision registers (32-bit) using one of the formats specified in Table 36.

7.4.3.8 SPN:MessageIdentifier

The message identifier shall be used to provide an 8-bit signature as a communication integrity check. The message identifier may be an incrementing counter 0-255 with rollover to 0 after 255 or a security signature.

The security signature should provide the same or higher level of integrity as an incrementing counter.

The response message counter SPN shall match the message counter SPN of the message sent by the device issuing the action / enquiry instruction via the J1939 interface.

7.5 PGN:Machine»CxDdata (PROPULSION)

7.5.1 General

This PGN provides information from the machine to the CxD. The structure of `PGN:Machine»CxDdata` for the PROPULSION subsystem is shown in Figure 21. Additional details are provided in 7.5.2 to 7.5.4.

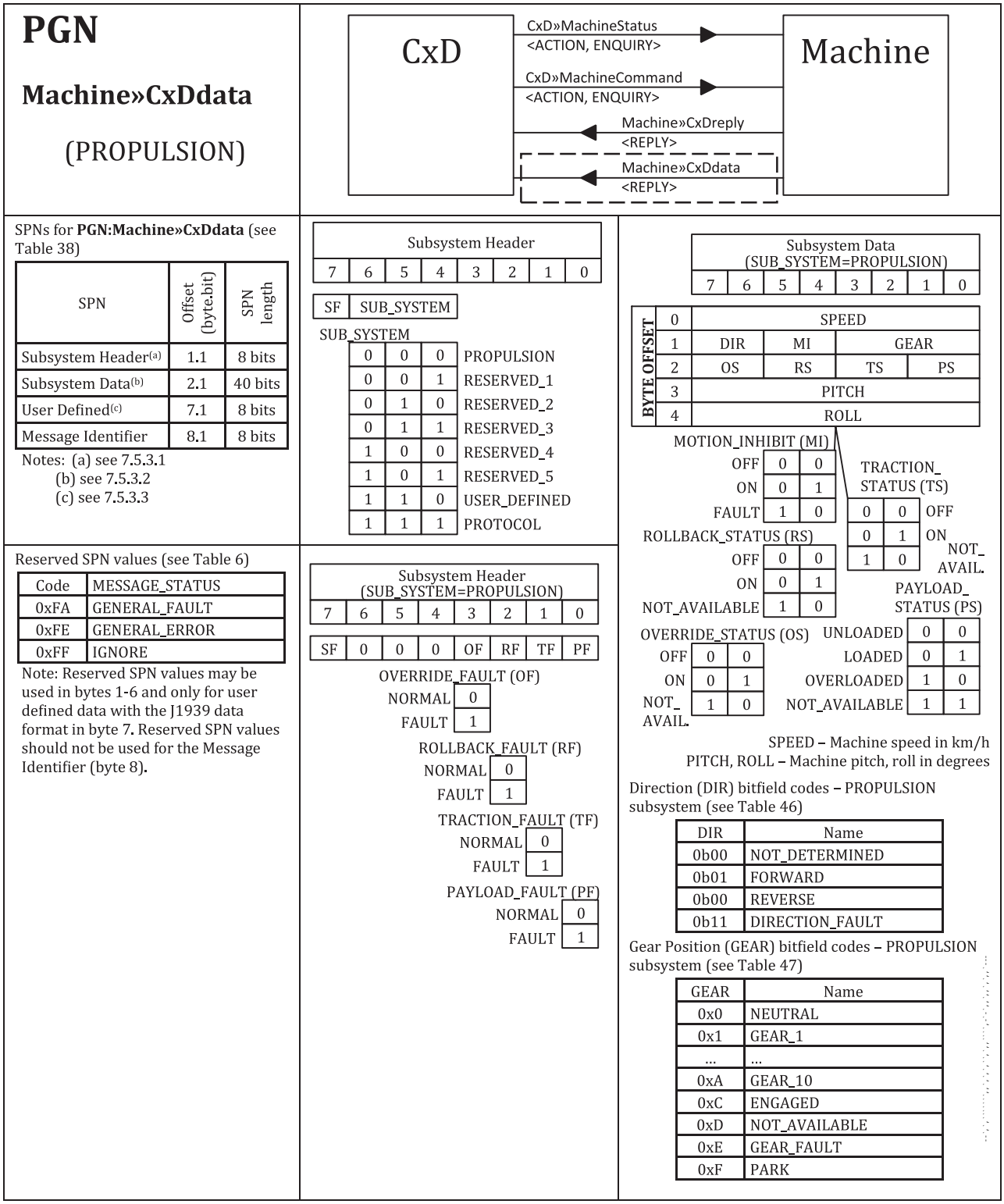


Figure 21 — Structure of `PGN:Machine»CxDdata` (PROPULSION subsystem)

7.5.2 PGN description

This parameter group provides information from the machine to the CxD over the J1939 on-board communication interface. The PGN parameters are shown in Table 37.

Table 37 — `PGN:Machine»CxDdata` parameters

Parameter	Description
PGN number	64203
PGN type	PDU2_SLOW
Acronym	CXD4
Data length	8 bytes
Multipacket message	No
Broadcast rate	100 ms
Message priority	5

The SPN's associated with `PGN:Machine»CxDdata` are listed in Table 38.

Table 38 — SPNs for `PGN:Machine»CxDdata`

SPN	Offset (byte.bit)	SPN length	SPN Description
Subsystem header	1.1	8 bits	Header information including subsystem and error indication flags
Subsystem data	2.1	40 bits	Information from machine for specified subsystem
User configurable	7.1	8 bits	Available for use
Message identifier	8.1	8 bits	Incrementing counter

NOTE Unless specified otherwise, least-significant-first byte ordering is used for multi-byte transfers with individual bytes preserving J1939 bit ordering (big-endian).

The standard SAE J1939 reserved message status codes (see Table 6) may be used to indicate status of each 8-bit SPN except for the message identifier (offset = 8) and where noted in 7.5.3.

Each SPN within `PGN:Machine»CxDdata` is defined in more detail in 7.5.3.

7.5.3 SPN structure

7.5.3.1 SPN:Subsystem Header

This SPN defines the subsystem providing data from the machine in the format shown in Figure 22. The least significant four (4) bits are defined separately for each subsystem.

Subsystem Header							
7	6	5	4	3	2	1	0
SF	SUB_SYSTEM						

SUB_SYSTEM

0	0	0	PROPULSION
0	0	1	RESERVED_1
0	1	0	RESERVED_2
0	1	1	RESERVED_3
1	0	0	RESERVED_4
1	0	1	RESERVED_5
1	1	0	USER_DEFINED
1	1	1	PROTOCOL

Figure 22 — Structure of `SPN:SubsystemHeader`

The bitfield enumerations for this SPN are listed in Table 39.

Table 39 — Subsystem Fault (SF) bitfield codes - `SPN:SubsystemHeader`

SF	Name	Description
0b0	NORMAL	The data stream from the specified subsystem is functioning normally
0b1	FAULT	There is a fault in the data stream from the specified subsystem (e.g. subsystem is offline, faulty, in an error state)

7.5.3.2 SPN:SubsystemData

This SPN provides 40 bits of information for the subsystem specified in `SPN:SubsystemHeader` in the format shown in Figure 23.

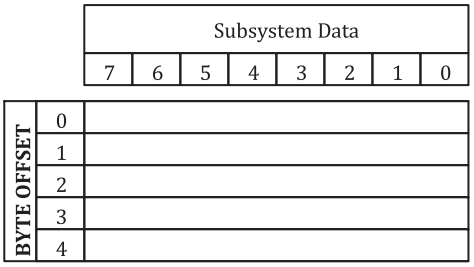


Figure 23 — Structure of `SPN:SubsystemData`

7.5.3.3 SPN:UserConfigurable

Available for use by end user.

7.5.3.4 SPN:MessageIdentifier

The message identifier shall be used to provide an 8-bit signature as a communication integrity check. The message identifier may be an incrementing counter 0-255 with rollover to 0 after 255 or a security signature.

The security signature should provide the same or higher level of integrity as an incrementing counter.

Messages shall be sent in sequence by the device connected to the J1939 interface.

7.5.4 PROPULSION subsystem

7.5.4.1 Propulsion Header

The structure of `SPN:SubsystemHeader` for the PROPULSION subsystem is shown in Figure 24.

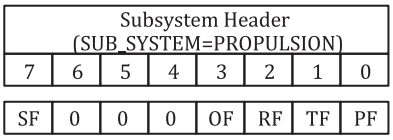


Figure 24 — Bit-fields in SPN:SubsystemHeader **– PROPULSION subsystem**

The valid bitfield enumerations are listed in Table 40 to Table 43.

Table 40 — Override fault (OF) bitfield codes – PROPULSION subsystem

OF	Name	Description
0b0	NORMAL	The manual override function is working normally
0b1	FAULT	The manual override function is in a fault state

Table 41 — Rollback fault (RF) bitfield codes – PROPULSION subsystem

RF	Name	Description
0b0	NORMAL	The rollback detection function is working normally
0b1	FAULT	The rollback detection function is in a fault state

Table 42 — Traction fault (TF) bitfield codes – PROPULSION subsystem

TF	Name	Description
0b0	NORMAL	The traction control function is working normally
0b1	FAULT	The traction control function is in a fault state

Table 43 — Payload fault (TF) bitfield codes – PROPULSION subsystem

PF	Name	Description
0b0	NORMAL	The payload monitoring function is working normally
0b1	FAULT	The payload monitoring function is in a fault state

7.5.4.2 Propulsion Data

The structure of `SPN:SubsystemData` for the PROPULSION subsystem is shown in Figure 25.

		Subsystem Data (SUB_SYSTEM=PROPULSION)							
		7	6	5	4	3	2	1	0
BYTE OFFSET	0	SPEED							
	1	DIR		MI		GEAR			
	2	OS		RS		TS		PS	
	3	PITCH							
	4	ROLL							

Figure 25 — Bit-fields in `SPN:SubsystemData` – PROPULSION subsystem

The valid bitfield enumerations for this SPN are listed in Table 44 to Table 53.

Table 44 — Speed (SPEED) encoding – PROPULSION subsystem

SPEED	Engineering Value	SPEED
0x00	0 km/h	0x00
0xF9	124,5 km/h	0xF9

The SAE J1939 message status codes (see Table 6) may be used to indicate if this SPN value is not available.

NOTE: This bitfield provides an indication of the current speed of machine registered by the speedometer / tachograph in the range 0 km/h to 124,5 km/h encoded onto 0x00-0xF9 with a scaling of 0,5 km/h per bit and zero offset.

Actual ground speed may vary due to measurement error in the tachograph or wheel spin. Consult machine manufacturer specifications or instructions for use.

EXAMPLE 1:

SPEED = 0x45 (34,5 km/h) – speedometer / tachograph reads 34,5 km/h.

EXAMPLE 2:

SPEED = 0xFF (IGNORE) – speed information not available from machine (see Table 6).

Table 45 — Motion inhibit (MI) bitfield codes – PROPULSION subsystem

MI	Name	Description
0b00	OFF	Motion inhibit status of the machine is OFF – i.e. machine is not prevented from moving from a stationary state. The CxD may determine if motion inhibit capability is available on the machine by reading the INR bitfield of `SPN:Status/Command` of `PGN:CxD»MachineStatus` in response to an enquiry sent by the CxD. The machine should return MI = OFF if motion inhibit capability is not available.
0b01	ON	Machine is stationary and motion inhibit status is ON – i.e. machine is prevented from moving from its stationary state.
0b10	FAULT	Motion inhibit indication is in a fault state.
0b11	RESERVED_3	Reserved for future use by the ISO 21815 series.

Table 46 — Direction (DIR) bitfield codes – PROPULSION subsystem

DIR	Name	Description
0b00	NOT_ DETERMINED	Operator has not selected either forward or reverse or is in park For tracked machines (e.g. skid steer loader), the indicated direction should be DIR=NOT_DETERMINED while the machine is slewing left or right, DIR=FORWARD when all tracks are moving the machine forward and DIR=REVERSE when all tracks are moving the machine in a backward direction.
0b01	FORWARD	Operator has selected a forward gear
0b10	REVERSE	Operator has selected a reverse gear
0b11	DIRECTION_FAULT	Direction function is in a fault state

Table 47 — Gear position (GEAR) bitfield codes – PROPULSION subsystem

GEAR	Name	Description
0x0	NEUTRAL	Operator has selected neutral (neither forward nor reverse)
0x1	GEAR_1	Operator has selected gear 1 / speed 1
0x2	GEAR_2	Operator has selected gear 2 / speed 2
0x3	GEAR_3	Operator has selected gear 3 / speed 3
0x4	GEAR_4	Operator has selected gear 4 / speed 4

NOTE: It is possible that the manual selection of gear by the operator does not reflect the physical engagement of a gear in the transmission or drive train of the machine.

Table 47 *(continued)*

GEAR	Name	Description
0x5	GEAR_5	Operator has selected gear 5 / speed 5
0x6	GEAR_6	Operator has selected gear 6 / speed 6
0x7	GEAR_7	Operator has selected gear 7 / speed 7
0x8	GEAR_8	Operator has selected gear 8 / speed 8
0x9	GEAR_9	Operator has selected gear 9 / speed 9
0xA	GEAR_10	Operator has selected gear 10 / speed 10
0xB	RESERVED_11	Reserved for future use by the ISO 21815 series
0xC	ENGAGED	Propulsion is engaged but no specific gear selected
0xD	NOT_AVAILABLE	Gear position is not available
0xE	GEAR_FAULT	Gear position function is in a fault state
0xF	PARK	Operator has selected parked position
NOTE: It is possible that the manual selection of gear by the operator does not reflect the physical engagement of a gear in the transmission or drive train of the machine.		

Table 48 — Payload status (PS) bitfield codes – PROPULSION subsystem

PS	Name	Description
0b00	UNLOADED	Machine is in unloaded state
0b01	LOADED	Machine is in loaded state
0b10	OVERLOADED	Machine is in overloaded state (e.g. >110 % payload, or the specific overload condition defined by the machine manufacturer)
0b11	NOT_AVAILABLE	Loaded / unloaded state is not available
This bitfield provides an indication of loaded / unloaded state for machines that can carry a payload (e.g. haul truck). The threshold for determining loaded / unloaded state should be specified by the machine manufacturer.		

Table 49 — Traction status (TS) bitfield codes – PROPULSION subsystem

TS	Name	Description
0b00	OFF	Traction control is in the OFF state (not active)
0b01	ON	Traction control is in the ON state (active)
0b10	NOT_AVAILABLE	Traction control status is not available
0b11	RESERVED_3	Reserved for future use by the ISO 21815 series
NOTE: This bit-field does not provide an indication that the traction control system is taking action.		

Table 50 — Rollback status (RS) bitfield codes – PROPULSION subsystem

RS	Name	Description
0b00	OFF	Rollback status is in the OFF state (inactive or not detected)
0b01	ON	Rollback status is in the ON state (rollback detected)
0b10	NOT_AVAILABLE	Rollback status is not available
0b11	RESERVED_3	Reserved for future use by the ISO 21815 series
NOTE 1: This bitfield indicates if movement of the machine in the reverse direction has been detected when either a forward gear or the neutral gear position has been selected by the operator. NOTE 2: Rollback detection is only active when a forward gear or the neutral gear position has been selected by the operator.		

Table 51 — Override status (OS) bitfield codes – PROPULSION subsystem

OS	Name	Description
0b00	OFF	Manual override is in the OFF state (inactive)
0b01	ON	Manual override is in the ON state
0b10	NOT_AVAILABLE	Override status is not available
0b11	RESERVED_3	Reserved for future use by the ISO 21815 series
The machine should return OFF state if the override function has not been implemented on the machine side of the interface. NOTE: This bitfield provides an indication that the operator has manually overridden the operation of the CxD from the machine side of the interface.		

Table 52 — Machine pitch (PITCH) encoding – PROPULSION subsystem

PITCH	Engineering value
0x00	-90°
0x22	-90°
0x7C	0°
0xD6	+90°
0xF9	+90°
The SAE J1939 message status codes (see Table 6) may be used to indicate if this SPN value is not available. NOTE: This bitfield provides an indication of the current pitch of the machine measured by the angle (-90° to +90°) between the machine x-axis (i.e. forward direction) and the horizontal plane as shown in Figure 26. Negative values indicate 'downhill' and a positive values 'uphill'. The value is encoded onto range 0x22-0xD6 with a scaling of 1° per bit. Values in the range 0x00 to 0x21 are interpreted as -90°. Values in the range 0xD7 to 0xF9 are interpreted as +90°. EXAMPLE 1: PITCH = 0x7C – pitch of machine is 0° (i.e. horizontal). EXAMPLE 2: PITCH = 0xFF (IGNORE) – pitch information not available from machine (see Table 6).		

Table 53 — Machine roll (ROLL) encoding – PROPULSION subsystem

ROLL	Engineering value
0x00	-124°
0x7C	0°
0xF9	+125°
The SAE J1939 message status codes (see Table 6) may be used to indicate if this SPN value is not available. NOTE: This bitfield provides an indication of the current roll of the machine measured by the angle (-124° to +125°) between the machine y-axis (i.e. lateral / side-to-side direction) and the ground plane in the direction of movement of the machine as shown in Figure 26. The ground plane is tilted from horizontal by the pitch angle. Negative values indicate 'roll left' and positive values indicate 'roll right'. The value is encoded onto range 0x00 to 0xF9 with a scaling of 1° per bit. EXAMPLE 1: ROLL = 0x73 – machine has rolled 9° to the left from the ground plane. EXAMPLE 2: ROLL = 0xFF (IGNORE) – roll information not available from machine (see Table 6).		

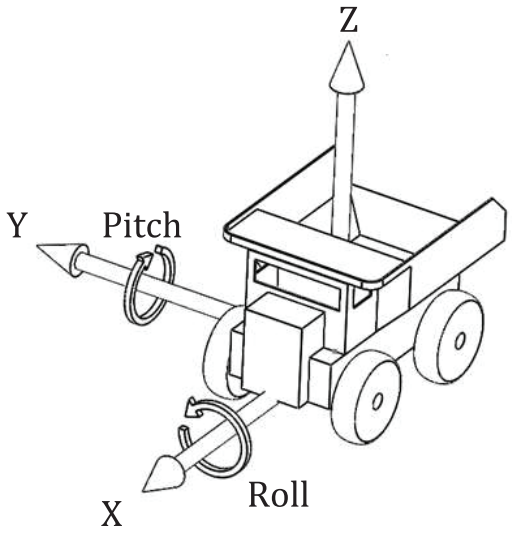


Figure 26 — SAE 2735 coordinates for pitch and roll

7.6 PGN:Machine»CxDstatus

This parameter group allows the machine to send enquiries or commands to the CxD. The PGN parameters are shown in Table 54.

Table 54 — `PGN:Machine»CxDstatus` parameters

Parameter	Description
PGN number	**to be assigned**
PGN type	PDU2_FAST
Acronym	CXD5
Data length	8 bytes
Multipacket message	No
Broadcast rate	10 ms
Message priority	4

This PGN is reserved for future use by the ISO 21815 series.

7.7 PGN:Machine»CxDcommand

This parameter group allows the machine to send enquiries or commands to the CxD. The PGN parameters are shown in Table 55.

Table 55 — `PGN:Machine»CxDcommand` parameters

Parameter	Description
PGN number	**to be assigned**
PGN type	PDU2_FAST
Acronym	CXD6
Data length	8 bytes
Multipacket message	No
Broadcast rate	100 ms
Message priority	2

This PGN is reserved for future use by the ISO 21815 series.

7.8 PGN:CxD»MachineReply

This parameter group allows the CxD to respond to enquiries or commands sent by the machine. The PGN parameters are shown in Table 56.

Table 56 — `PGN:CxD»MachineReply` parameters

Parameter	Description
PGN number	**to be assigned**
PGN type	PDU2_SLOW
Acronym	CXD7
Data length	8 bytes
Multipacket message	No
Broadcast rate	On request
Message priority	Current value

This PGN is reserved for future use by the ISO 21815 series.

7.9 PGN:CxD»MachineData

This PGN provides information from CxD to the machine. The PGN parameters are shown in Table 57.

Table 57 — `PGN:CxD»MachineData` parameters

Parameter	Description
PGN number	**to be assigned**
PGN type	PDU2_SLOW
Acronym	CXD8
Data length	8 bytes
Multipacket message	No
Broadcast rate	100 ms
Message priority	6

This PGN is reserved for future use by the ISO 21815 series.

7.10 PGN:Time/Date

This parameter group allows the machine to exchange time and date information with the CxD.

The PGN parameters are shown in Table 58.

Table 58 — `PGN:Time/Date` parameters

Parameter	Description
PGN number	65254
PGN type	Current Value
Acronym	TD
Data length	8 bytes
Multipacket message	No
Broadcast rate	On request

Table 58 *(continued)*

Parameter	Description
Message priority	Current value

This PGN is defined by SAE J1939-2. The structure of this PGN is shown in Figure 27 for reference.

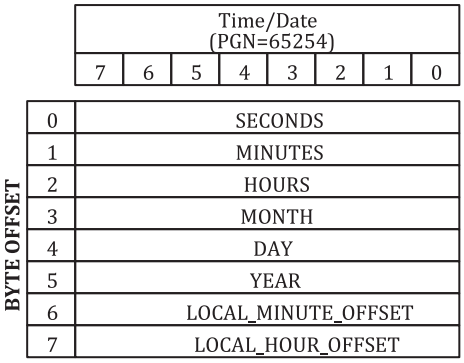


Figure 27 — Bit-fields in `PGN:Time/Date`

If supported by the machine, the CxD can request time and date information from the machine after negotiation has been completed (see 5.2).

If supported by the CxD, the machine can request time and date information from the CxD after negotiation has been completed (see 5.2).

8 Documentation

8.1 Machine documentation

The machine response and limitations for use shall be completed for each individual machine based on the format shown in Table 59.

Table 59 — Machine documentation (pro forma)

Command	Machine response	Limitations
EMERGENCY_STOP	Implemented = Y/N <what the machine does in response to this command>	Limitations for use
CONTROLLED_STOP		
SLOW_DOWN		
STAND_DOWN		
BYPASS_PROPULSION		
APPLY_PROPULSION_ SET-POINTS		

Additional details not fully described in this document should be provided by the machine manufacturer to enable implementation of the interface on CxD-ready machines – e.g. negotiation, installation, operation, user defined data structures.

8.2 System documentation

A general example of the type of information for use is provided in Table 60.

Table 60 — System documentation (pro forma)

Use case	Applicable command(s)	Limitations
e.g. usage scenario	e.g. EMERGENCY_STOP, CONTROLLED_STOP,…	Limitations of system (e.g. max speed)

Annex A
(informative)

Communication sequences

A.1 Instruction sequences

The CxD supplier and machine manufacturer should identify all instruction sequences implemented in the interface and expected behaviour of the machine in response to each CxD instruction using the format shown in Table A.1 and Table A.2 as a guide. The expected behaviour of the machine in response to each CxD instruction is outlined in Table A.1 and Table A.2.

The CxD column is used to indicate that a capability exists on the CxD-side of the communication interface.

The machine column is used to indicate that a capability exists on the machine-side of the communication interface.

The machine should respond to properly formed instructions sent by the CxD that are within the capabilities of the machine after negotiation has been completed. The machine should respond with ENQUIRY_ERROR or ACTION_ERROR for unsupported instructions.

Improperly formed messages sent by the CxD should be silently discarded by the machine.

Table A.1 — Instruction sequences (`PGN:CxD»MachineStatus`)

ID	CxD	CxD instruction	Machine response	Machine	Expected behaviour
S1	☑	ENQ7=NEGOTIATE_NOP	RTE7=NEGOTIATE_ACK	☑	Unconditional handshake, interface state (STATE code, see Figure 10) returned by machine in `RegisterValue0`
S2	☐	ENQ7=NEGOTIATE_ENQ	RTE7=NEGOTIATE_REPLY	☐	Reply to negotiation sequence initiated by CxD, interface state (STATE code, see Figure 10) returned by machine in `RegisterValue0`
S3	☐		RTE7=RENEGOTIATE_REPLY	☐	Machine is requesting the CxD to recommence the negotiation sequence
S4	☑	ENQ7=PROTOCOL_NOP	RTE7=PROTOCOL_ACK	☑	Conditional handshake after negotiation has been completed successfully, state of interface returned by machine `in RegisterValue0`
S5			RTE7=RENEGOTIATE_REPLY	☐	Machine is requesting the CxD to recommence the negotiation sequence

NOTE: Symbols to indicate implementation status for the CxD and the machine: ☑=implemented, ☒=not implemented, ☐=to be assigned.

Table A.1 *(continued)*

ID	CxD	CxD instruction	Machine response	Machine	Expected behaviour
S6	☐	ENQ7=GET_PROTOCOL_REGISTER SELECT=SELECT_REGISTER	RTE7=GET_REGISTER_OK	☐	Successfully read the value of the register at `RegisterIndex` in the PROTOCOL subsystem
S7	☐		RTE7=ENQUIRY_ERROR	☑	Error encountered reading the value of the register at `RegisterIndex` in the PROTOCOL subsystem
S8	☐	ENQ7=GET_EXTENDED_REGISTER SELECT=SELECT_REGISTER	RTE7=GET_EXTENDED_REGISTER_OK	☐	Successfully read the value of the register at `RegisterIndex` in the extended register address space
S9	☐		RTE7=ENQUIRY_ERROR	☑	Error encountered reading the value of the register at `RegisterIndex` in the extended register address space
S10	☐	ACT7=SET_PROTOCOL_REGISTER	RTA7=SET_REGISTER_OK	☐	Successfully updated register at `RegisterIndex` in the PROTOCOL subsystem
S11	☐		RTA7= ACTION_ERROR	☑	Error updating register at `RegisterIndex` in the PROTOCOL subsystem
S12	☐	ACT7=RESET_REGISTERS SELECT=SELECT_SUBSYSTEM	RTA7=RESET_REGISTERS_OK	☐	Successful reset of registers in subsystem specified in SUB_SYSTEM bitfield of `RegisterSelect`
S13	☐		RTA7= ACTION_ERROR	☑	Failure to reset registers in specified in SUB_SYSTEM bitfield of `RegisterSelect`
S14	☐	ACT7=SET_PROPULSION_REGISTER SELECT=SELECT_REGISTER	RTA7=SET_REGISTER_OK	☐	Successfully set register at `RegisterIndex` in the PROPULSION subsystem to `RegisterValue0..3`.
S15	☐		RTA7= ACTION_ERROR	☑	Error updating register at `RegisterIndex` in the PROPULSION subsystem
S16	☐	ACT7=SET_PROPULSION_REGISTER SELECT=SELECT_AND_TAG	RTA7=SET_REGISTER_OK	☐	Successfully set register at `RegisterIndex` in the PROPULSION subsystem to `RegisterValue0..3` and applied TAG value.
S17	☐		RTA7= ACTION_ERROR	☑	Error encountered updating register at `RegisterIndex` in the PROPULSION subsystem and applying TAG value
S18	☐	ENQ0=GET_PROPULSION_REGISTER SELECT=SELECT_REGISTER	RTE0=PROPULSION_ACK	☐	Successfully read the value of the register at `RegisterIndex` in the PROPULSION subsystem
S19	☐		RTE0=ENQUIRY_ERROR	☑	Error encountered reading the value of the register at `RegisterIndex` in the PROPULSION subsystem
S20	☐	ACT0=LOAD_PROPULSION_SETPOINTS SELECT=UPDATE_AND_APPLY	RTA0=PROPULSION_ACK	☐	Machine is able to immediately update the specified register in the PROPULSION subsystem at `RegisterIndex` to `RegisterValue0..3` and apply the setpoint
S21	☐		RTA0=ACTION_ERROR	☑	Machine is not able to execute the action or the register does not have REG_TYPE=SET_POINT

NOTE: Symbols to indicate implementation status for the CxD and the machine: ☑=implemented, ☒=not implemented, ☐=to be assigned.

Table A.1 *(continued)*

ID	CxD	CxD instruction	Machine response	Machine	Expected behaviour
S22	☐	ACT0=LOAD_PROPULSION_SETPOINTS SELECT=APPLY_FROM_LIST	RTA0=PROPULSION_ACK	☐	Machine is able to apply the setpoints for up to four (4) registers in the PROPULSION subsystem specified in `RegisterValue_0..3`
S23	☐		RTA0=ACTION_ERROR	☑	Machine is not able to execute the action or at least one of the registers specified in `RegisterValue_0..3` does not have REG_TYPE=SET_POINT
S24	☐	ACT0=LOAD_PROPULSION_SETPOINTS SELECT=LOOKUP_INDIRECT	RTA0=PROPULSION_ACK	☐	Machine is able to apply the setpoints for up to thirty two (32) contiguous registers in the PROPULSION subsystem indirectly specified by the bit pattern in `RegisterValue_0..3` and the OFFSET value.
S25	☐		RTA0=ACTION_ERROR	☑	Machine is not able to execute the action or at least one of the registers indirectly specified does not have REG_TYPE=SET_POINT
S26	☐	ACT0=LOAD_PROPULSION_SETPOINTS SELECT=MATCH_TAG	RTA0=PROPULSION_ACK	☐	Machine is able to apply the setpoints for any register in the PROPULSION subsystem with a matching TAG value
S27	☐		RTA0=ACTION_ERROR	☑	Machine is not able to execute the action or at least one of the registers with a matching TAG value does not have REG_TYPE=SET_POINT

NOTE: Symbols to indicate implementation status for the CxD and the machine: ☑=implemented, ☒=not implemented, ☐=to be assigned.

Table A.2 — Instruction sequences (`PGN:CxD»MachineCommand`)

ID	CxD	CxD instruction	Machine response	Machine	Expected behaviour
C1	☐	XEQ0=NORMAL_OPERATION	RTA0=PROPULSION_ACK	☐	Machine acknowledges and executes the NORMAL_OPERATION action
C2	☐		RTA0=ACTION_ERROR	☑	NORMAL_OPERATION action not executed by machine
C3	☐	CHK0=EMERGENCY_STOP_CONFIRM	RTE0=EMERGENCY_STOP_CONFIRM_ACK	☐	Machine has the capability of performing an EMERGENCY_STOP
C4	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability of performing an EMERGENCY_STOP
C5	☐	CHK0=CONTROLLED_STOP_CONFIRM	RTE0=CONTROLLED_STOP_CONFIRM_ACK	☐	Machine has the capability of performing a CONTROLLED_STOP
C6	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability of performing a CONTROLLED_STOP

NOTE: Symbols to indicate implementation status for the CxD and the machine: ☑=implemented, ☒=not implemented, ☐=to be assigned.

Table A.2 *(continued)*

ID	CxD	CxD instruction	Machine response	Machine	Expected behaviour
C7	☐	CHK0=SLOW_DOWN_CONFIRM	RTE0=SLOW_DOWN_CONFIRM_ACK	☐	Machine has the capability of performing a SLOW_DOWN
C8	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability of performing a STAND_DOWN
C9	☐	CHK0=STAND_DOWN_CONFIRM	RTE0=STAND_DOWN_CONFIRM_ACK	☐	Machine has the capability of performing a STAND_DOWN
C10	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability of performing a STAND_DOWN
C11	☐	CHK0=BYPASS_PROPULSION_CONFIRM	RTE0=BYPASS_PROPULSION_ CONFIRM_ACK	☐	Machine has the capability of performing a BYPASS_PROPULSION
C12	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability of performing a BYPASS_PROPULSION
C13	☐	CHK0=APPLY_PROPULSION_SETPOINTS_CONFIRM	RTE0=APPLY_PROPULSION_SETPOINTS_CONFIRM_ACK	☐	Machine has the capability to apply the setpoint(s) specified by the LOAD_PROPULSION_ SETPOINTS action
C14	☐		RTE0=ENQUIRY_ERROR	☑	Machine does not have the capability to apply setpoints specified by the LOAD_PROPULSION_ SETPOINTS action
C15	☐	XEQ0=EMERGENCY_STOP	RTA0=EMERGENCY_STOP_ACK	☐	Machine acknowledges and executes the EMERGENCY_STOP action
C16	☐		RTA0=ACTION_ERROR	☑	EMERGENCY_STOP action not executed by machine
C17	☐	XEQ0=CONTROLLED_STOP	RTA0=CONTROLLED_STOP_ACK	☐	Machine acknowledges and executes the CONTROLLED_STOP action
C18	☐		RTA0=ACTION_ERROR	☑	CONTROLLED_STOP action not executed by machine
C19	☐	XEQ0=SLOW_DOWN	RTA0=SLOW_DOWN_ACK	☐	Machine acknowledges and executes the SLOW_DOWN action
C20	☐		RTA0=ACTION_ERROR	☑	SLOW_DOWN action not executed by machine
C21	☐	XEQ0=STAND_DOWN	RTA0=STAND _DOWN_ACK	☐	Machine acknowledges and executes the STAND_DOWN action
C22	☐		RTA0=ACTION_ERROR	☑	STAND_DOWN action not executed by machine
C23	☐	XEQ0=BYPASS_PROPULSION	RTA0=BYPASS_PROPULSION_ ACK	☐	Machine acknowledges and executes the BYPASS_PROPULSION action
C24	☐		RTA0=ACTION_ERROR	☑	BYPASS_PROPULSION action not executed by machine
C25	☐	XEQ0=APPLY_PROPULSION_SETPOINTS	RTA0=APPLY_PROPULSION_SETPOINTS_ACK	☐	Machine successfully applies the setpoint(s) specified by the LOAD_PROPULSION_ SETPOINTS action
C26	☐		RTA0=ACTION_ERROR	☑	Machine is unable to apply setpoints specified by the LOAD_PROPULSION_SETPOINTS action

NOTE: Symbols to indicate implementation status for the CxD and the machine: ☑=implemented, ☒=not implemented, ☐=to be assigned.

A.2 Message sequences

A.2.1 General

The message sequences are shown in ascending order with time running vertically down. Messages may appear in a different order to the examples shown in A.2.2 to A.2.4.

A.2.2 Negotiation

An example of the negotiation message sequences between the CxD and machine over the J1939 on-board communications interface is shown in Figure A.1.

The machine can implement the following actions:

— PowerOn();

— doNegotiation();

— enableTimeout().

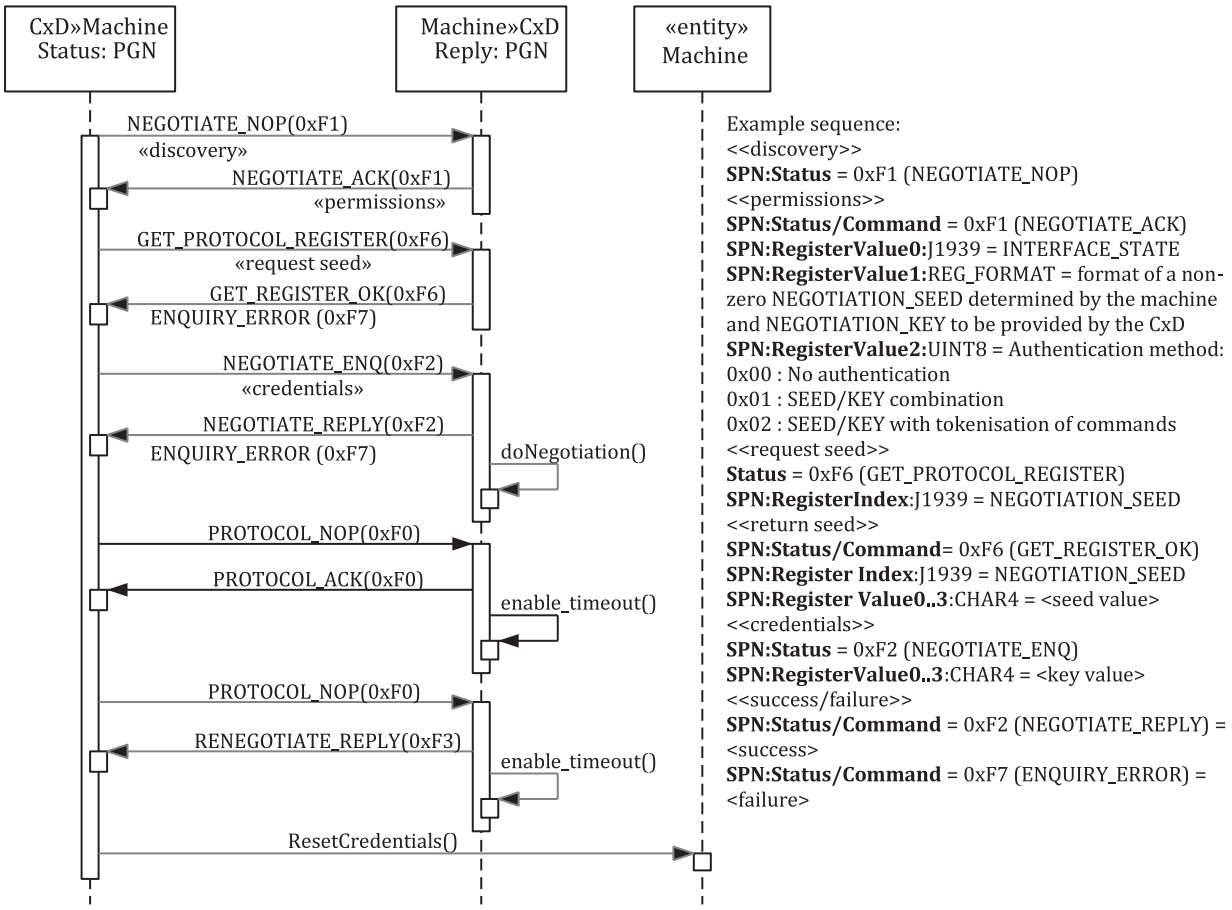


Figure A.1 — Status / command message sequence (PROTOCOL subsystem) – Negotiation

A.2.3 Initialisation

An example of the initialisation message sequences between the CxD and machine over the J1939 on-board communications interface are shown in Figure A.2.

The machine can implement the following actions:

— PowerOn();

— resetTimeout().

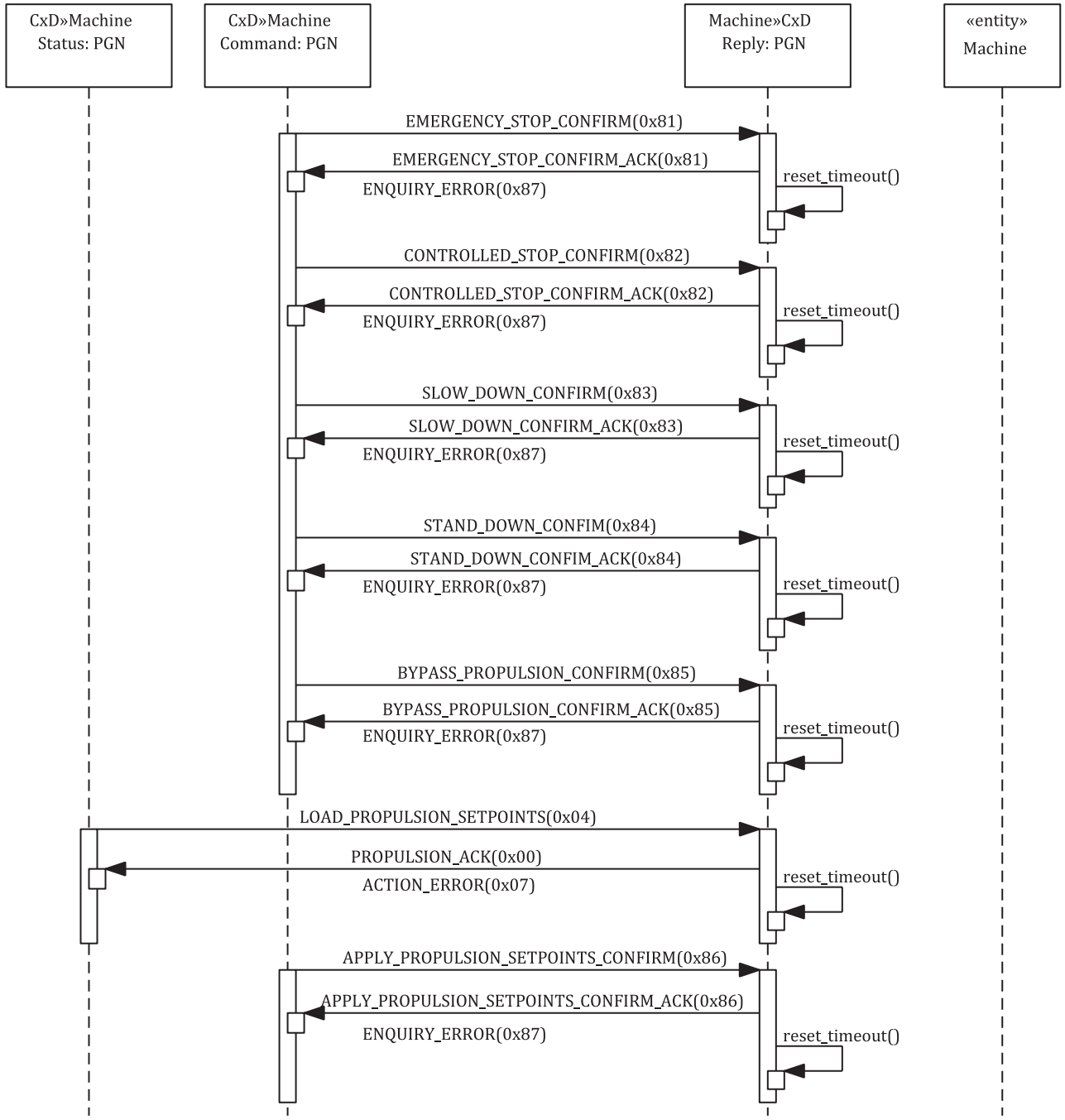


Figure A.2 — Status / command message sequence (PROPULSION subsystem) – Initialisation

A.2.4 Operation

An example of the operating message sequences between the CxD and machine over the J1939 on-board communications interface are shown in Figure A.3.

The machine can implement the following actions:

— resetTimeout();

— doEmergencyStop();

— doControlledStop();

— doSlowDown();

— doStandDown();

— doBypassPropulsion();

— doApplyPropulsionSetpoints();

— doMotionInhibit();

— doNormalOperation().

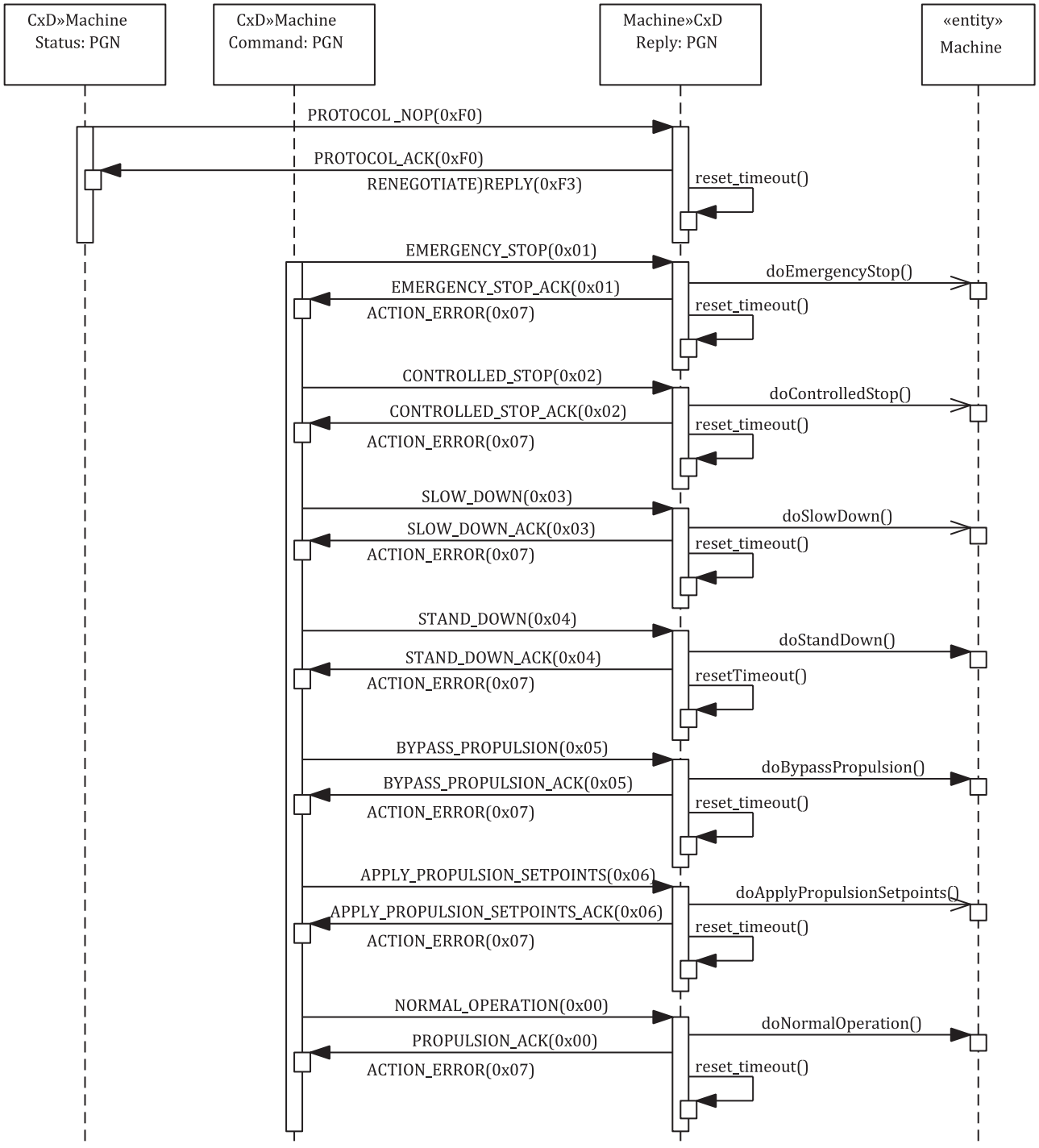


Figure A.3 — Status / command message sequence (PROPULSION subsystem) – Operation

The PROTOCOL_NOP message sequence should be used when the CxD is not issuing one of the other operational instructions.

Annex B
(informative)

Trust mechanisms

B.1 General

This annex outlines a mechanism for establishing trust between the machine and the CxD based on the exchange of certificates as an extension to the authentication methods referenced in 5.2. The structure of the messages used to establish trust using this mechanism are defined by the machine manufacturer and can be different from the message structure defined in this document.

B.2 Security goals

The goals for securing communication between the CxD and the machine are:

— authentication of CxD to the machine;

— message integrity between CxD and the machine.

NOTE The reservation of `PGN:Machine>>CxDCommand` in this document indicates that there can be a future requirement for the CxD to verify the signed credentials of the machine to achieve message integrity between the machine and the CxD.

B.3 Secure connection requirements

Authentication requirements include:

— mutually trusted authority;

— module unique identity (each module unique among all modules trusted by authority);

— module credentials from trusted authority.

Message integrity requirements include:

— message signature signed by sender and verified by receiver;

— symmetric signatures require a shared secret between sender and receiver.

B.4 Process steps

The establishment of trust between the machine and the CxD based on certificates include the following steps:

a) establish authority – create base authority;

b) enrolment – provision identities, keys and authorities;

c) pairing of CxD to the machine – handshake to authenticate CxD to machine and initialize a share secret;

d) secure session – exchange information securely.

B.5 Establish authority

The authority is a robust software package, with features of a public key infrastructure, trusted by the machine to issue CxD credentials with integrity. It may be supported by the machine manufacturer or CxD suppler or a mutually trusted third party. The authority should maintain exclusive and secure access of the authority private key.

The primary functions of the authority are:

a) approve CxD identity;

b) issue signed credentials to only authentic CxD units;

c) distribute the authority public key to modules which verify credentials.

The authority may be based on an internet service, a proprietary back office hardware security module (HSM) or a smartcard on the manufacturing line.

B.6 Enrolment

Each CxD should be provisioned in a secure environment to initialize its unique identity (e.g. serial number), generate a public key pair and request the authority to issue signed credentials. The private key should be stored in secure ROM to avoid cloning. This function should be part of CxD end of line manufacturing. It may use CAN protocol or any other media. The authority should not issue credentials to unauthorized modules. The key management process is illustrated in Figure B.1.

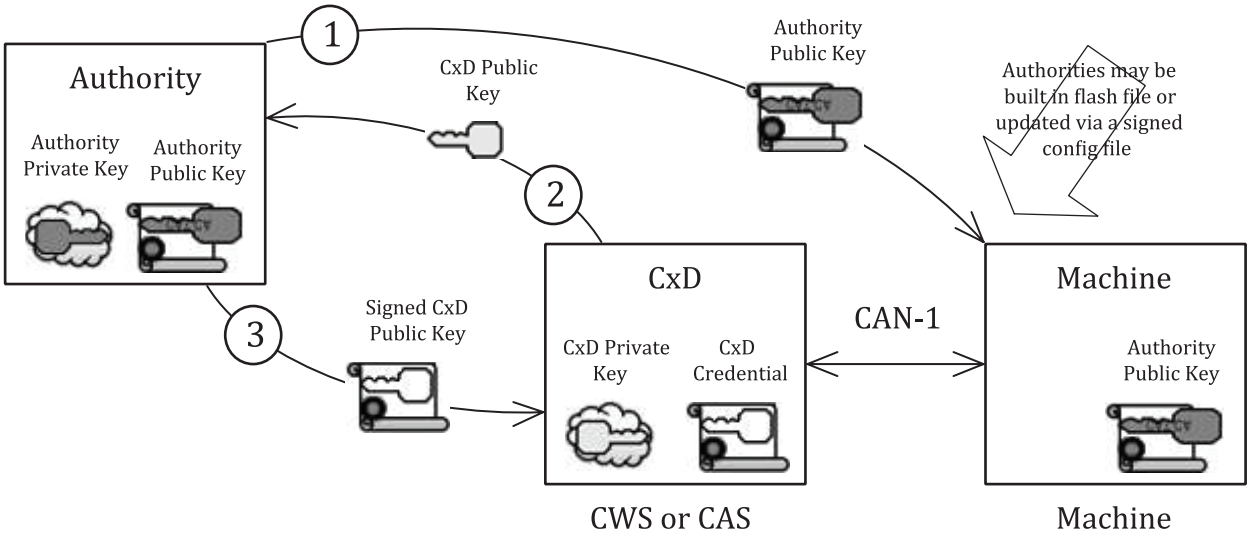


Figure B.1 — Key Management between machine and CxD

The authority provides its public key to machines as a file over any media. Each machine should be configured to trust the CxD authority by including the authority public key in a signed flash file or signed configuration file. The machine should keep the authority public key in immutable memory. A machine may be configured to trust multiple CxD authorities.

NOTE Enrolment does not associate any specific CxD identity to a machine, only to the CxD authority.

B.7 Pairing CxD to a machine

The pairing process between the CxD and the machine handshake has two results:

— authenticate CxD to the machine;

— initialize a shared secret.

Mutual authentication may be required by the CxD to authenticate the machine.

The general pairing process is illustrated in Figure B.2.

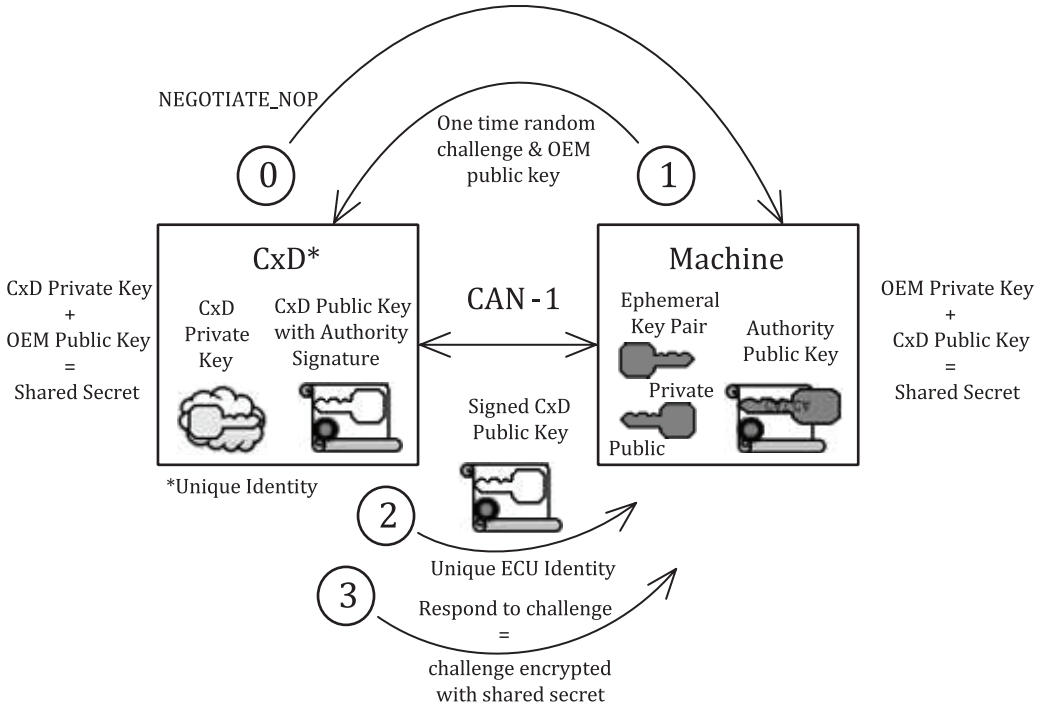


Figure B.2 — Pairing of CxD to machine

The negotiation sequence described for NEGOTIATE_ENQ in Table 8 can be implemented using certificates using the following steps:

a) CxD sends NEGOTIATE_NOP message.

b) Machine generates an ephemeral public key pair (ephemeral means temporary).

c) Machine sends CxD a one-time random challenge along with the machines ephemeral public key:

   — protocol version: 16 bits = 2 bytes (SHA256 / ECC-SECP224 / AES128ECB);

   — random challenge: 8 bytes;

   — ephemeral public key X & Y = 56 bytes.

d) CxD generates a shared secret using the challenge, machine public key and its own private key.

e) CxD calculates a response to the challenge as signature of CxD credential using the shared secret.

f) CxD returns the challenge response to the machine along with CxD credentials:

   — CxD credential: version + CxD_ID + pubic key + authority + signature = 128 bytes;

   — version = 2 bytes;

   — CxD_ID = 10 bytes;

   — public key X & Y = 56 bytes;

   — authority ID = 4 bytes;

— signature S & R = 56 bytes;

— handshake response: 8 bytes.

g) CxD securely stores shared secret for secure session communications with machine.

h) Machine verifies the CxD credential using the authority public key.

i) Machine generates a shared secret using the challenge, CxD public key and its own private key.

j) Machine verifies the challenge response using the shared secret.

k) Machine securely stores shared secret for secure session communications with CxD.

l) Machine sends a NEGOTIATE_ACK message (if everything success).

NOTE "Protocol version" defines cryptographic methods used which implies field lengths and algorithms needed. The suggested combination of SHA256 / ECC-SECP224 / AES128ECB are relatively compact and currently accepted cryptographically strong. Other methods can be used in combination.

CxD suppliers or machine manufacturers may use proprietary methods. A block of "protocol version" values should be reserved for standard definitions.

Authority identity conflicts should be avoided by machine manufacturers.

CxD unique identity should be controlled by the CxD manufacturer.

B.8 Secure session

The handshake dynamically pairs a specific CxD with one machine. This pairing may last a short or long time. It may expire at key-off or persist. A handshake may need to repeat after a max count of messages or may require re-negotiation for other reasons.

Authentication confirms that official devices are present and able to perform CxD functions. However, authentication alone does not prevent unauthorized devices from spoofing CxD messages. The handshake generates a shared secret between CxD and the machine which is useful to sign or encrypt data link messages using AES_CTR mode. The authentication itself uses the shared secret to verify the response to a challenge. The general arrangement is illustrated in Figure B.3.

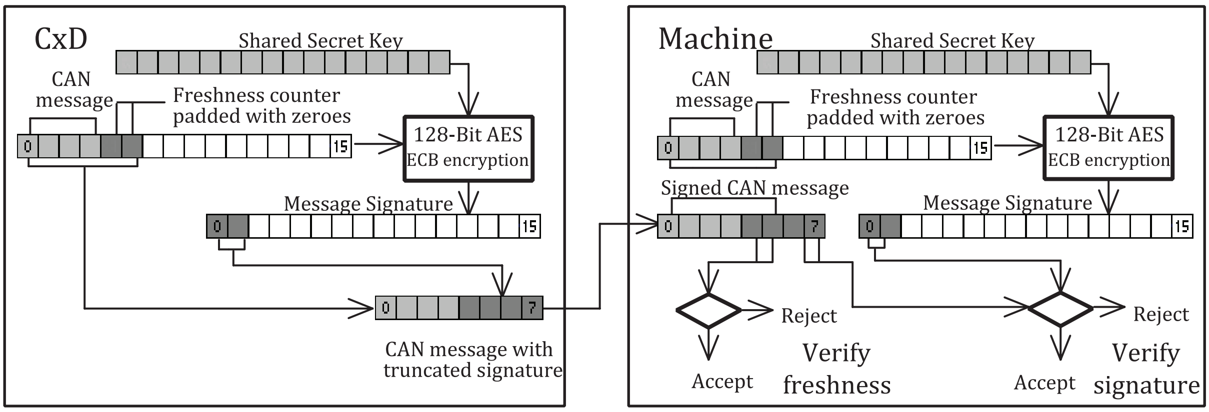


Figure B.3 — Signing of CAN messages

To assure data link integrity, messages should be unique and signed. Uniqueness is achieved by incrementing a freshness value. Without uniqueness, replaying recorded messages is easy. Without a signature authentic messages cannot be distinguished from malicious messages.

Compromises are made on data links to maintain integrity with minimum bandwidth. A suggested field size for CAN 8-byte payload for messages exchanged during the session are: maximum message length: 4 bytes; minimum freshness: 2 bytes; minimum signature: 2 bytes.

NOTE 1 The details of implementing the session layer are manufacturer specific and are not fully addressed here. The four byte session payload would necessitate two (2) session messages to deliver one PGN payload of eight bytes as defined in this document between the machine and the CxD.

NOTE 2 The existing “Message Identifier” sequence number can be used as part of freshness.

Rational: If freshness rolls over with the same key, an attacker can successfully replay all recorded messages. So, max freshness should be larger than number of messages sent in the lifetime of a key. 16 bits = 65535 permutations.

Signature brute force attack succeeds by cycling through all signature permutations. So, signature size should be larger than message retry limit. Pure chance allows unauthorized messages to succeed 1 out of 65535, or .0015 %. A diagnostic message should trigger an intrusion detection when a significant number of messages fail.

Assumed: crypto algorithm key size is accepted as strong.

Messages that result in machine movement should be signed. Messages that claim protocol health should be signed. The “Message Identifier” sequence number technique described in this document may enable CxD communications to detect mischief, but cannot prevent targeted attacks.

B.9 Interaction diagram

The high-level message sequences are shown in Figure B.4 in ascending order with time running vertically down. Messages may appear in a different order to the examples shown.

The following sequence illustrates message content exchanged to achieve the goals stated above using a protocol version: SHA256 / ECC224 / AES128ECB.

This information should be packaged as needed for the CxD and machine J1939 on-board communications interface.

Authority – given a unique identity, provisions the authority public and private key pair.

CxD – given a unique identity, provisions a public and private key pair and obtains a credential signature from the authority. CxD also generates a shared secret and challenge response with machine.

Machine – given a random challenge, provisions a public and private key pair, then verifies the CxD credential using authority, generates a shared secret and verifies the challenge response from CxD.

```
>> get-authority-public-key
Enter authority name: <authority name>
Authority private key = 632276187D11E80011AD1BAF60AA5FAC 1136EEFCF0373A3867F71326
Authority public key = 87EF50F767425B934C782A9A6A026584 2D1048459501E0AAD170465F
   A342FC85E1A0A661FEBA2E6AEE26EB94 BA9B601A8F9BD5A956392ED0
>> get-cxd-keys
Enter CxD name: <CxD name>
*CxD unique identity = 70707000FEFEFEFEFEFEFEFEFEFEFEFE
CxD private key = 51E10638434BE1171F1B598D63E6A064 BF8F6CBF3834C23E8A81AAED
*CxD public key = DC26E4BC9AFA3EEEE9800FD5B7BE0352 452A48D532F5392ECDE6E201
   1E2041497139251FAC101C1AE20E4A1F 7AADF8C24446E9EA02C9C219
*CxD authority signature = CC8FFEB837F1FD17D673D88E88ACD075 39E222AA59AE800C08A57A7B
   ABDE36198475E178E38C83716139ABB2 6EFFEC6282059E453CAA47E7
>> get-machine-keys
Enter random challenge: <random challenge>
Machine private key = 61D96B205DD36DA76AA2B350C37B9FE7 4030D8BF05BDAAC02E70AE6F
Machine public key = 47D0F0815B5787E1DD3072A1029481A0 CF3A440867D8E6DE024D4DB3
     BE0BCFFC6AEA7F87C8070E6156778B0B E074864C85FC04D3AB31D32C
CxD credentials verified!
Machine shared secret = B58E3C46A42BEED194D7C24EC982C5E2
```

```
CxD shared secret = B58E3C46A42BEED194D7C24EC982C5E2
Machine challenge = 6F6F6F00FEFEFEFE
CxD encrypted response = 05CDA3D6024CFEA5
Machine decrypted response = 6F6F6F00FEFEFEFE
```

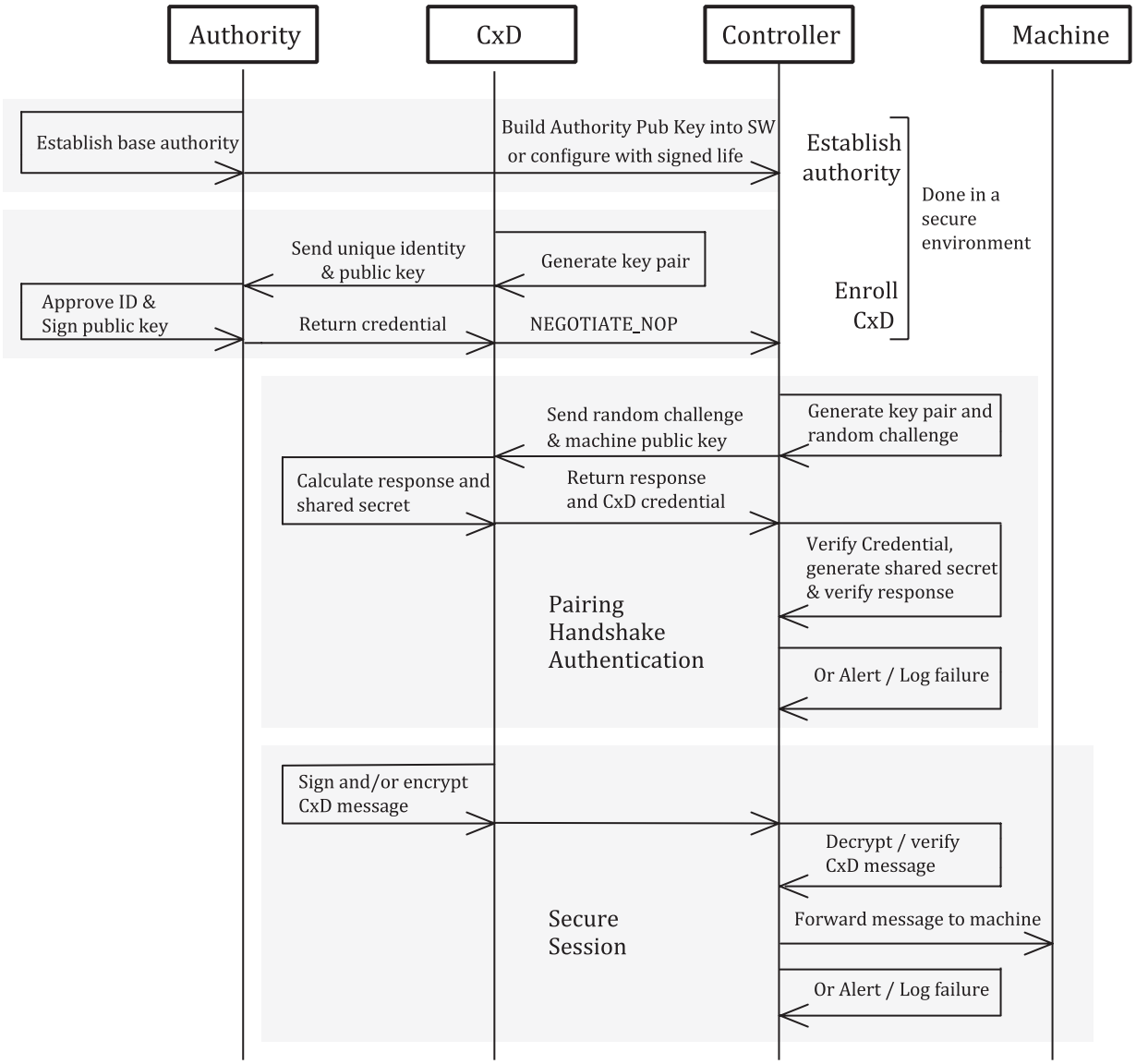


Figure B.4 — Message sequence for exchange of credentials

B.10 Diagnostics, conformance tests and audits

The proposed solution should be evaluated with a threat analysis and the implementation should be penetration tested for conformance.

A troubleshooting plan should be included to diagnose failures or misbehaviour between components with log files should be kept for auditing.

The hard part of security is staying ahead of attackers. The procedures above can be strong when released, but over time, attacks against the technology and specific implementation can produce findings that weaken the expected security. Prevention is best, detection is next, but either way a reaction plan is required for longevity of a secure solution. Revocation of rogue devices or authorities is very difficult for embedded devices. Security technology is very important; but be mindful that poor security looks very similar to good security, so choose wisely.

Annex C
(informative)

Implementation examples for override and standby modes

C.1 Interconnection of override switch

Example electrical connections are shown in Figure C.1 for applications where override mode is implemented by the override A and override B pins via the physical interface. Refer to 6.2 for additional information.

Resistors RA and RB (or equivalent components) should be provided by the CxD to pull up the override A and override B pins to a machine voltage. Other configurations can be implemented on the machine side or the CxD side of the connector consistent with the logic defined in Table 2.

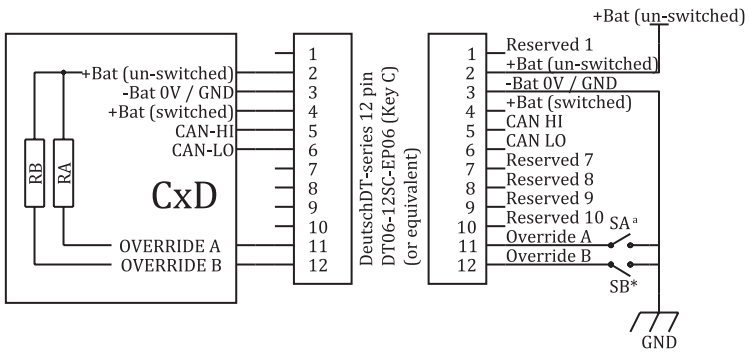


Figure C.1 — Example interconnection logic for override A and override B

C.2 Implementation examples for override and standby modes

C.2.1 Both on CxD-side of interface

Example electrical connections are shown in Figure C.2 for applications where standby and override modes are implemented by switch contacts on the CxD-side of the interface. During the period of time when the CxD is in the standby or override mode, the CxD should send the BYPASS_PROPULSION action via `PGN:CxD»MachineCommand`.

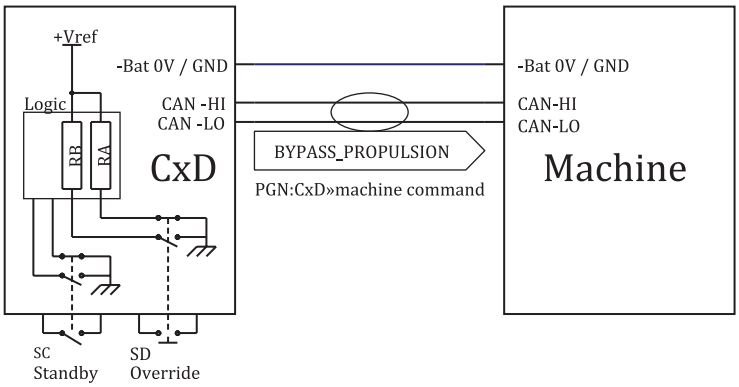


Figure C.2 — Example switch interconnection logic for standby mode and override mode implemented on CxD-side of interface

C.2.2 Override on machine-side of interface

Example electrical connections are shown in Figure C.3 for applications where standby mode is implemented by switch contacts on the CxD-side of the interface and override mode is implemented by switch contacts on the machine-side of the interface. During the period of time when the CxD is in the standby or override mode, the CxD should send the BYPASS_PROPULSION action via `PGN:CxD»MachineCommand`.

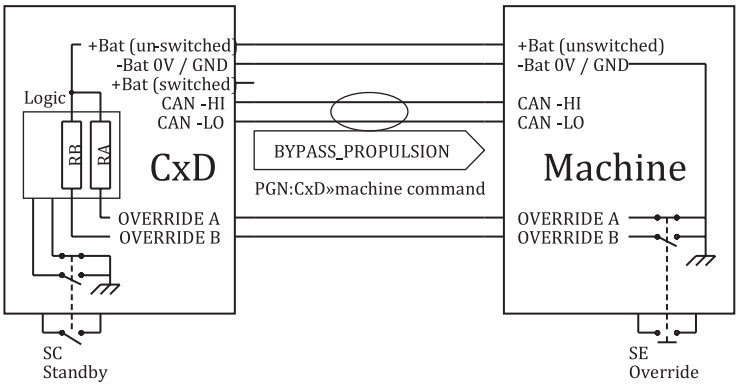


Figure C.3 — Example switch interconnection logic for standby mode implemented on CxD-side of interface and override mode implemented on machine-side of interface

Bibliography

[1] ISO 3779, *Road vehicles — Vehicle identification number (VIN) — Content and structure*

[2] ISO 6165, *Earth-moving machinery — Basic types — Identification and terms and definitions*

[3] ISO 10261, *Earth-moving machinery — Product identification numbering system*

[4] ISO 16001, *Earth-moving machinery — Object detection systems and visibility aids — Performance requirements and tests*

[5] ISO 17757, *Earth-moving machinery and mining — Autonomous and semi-autonomous machine system safety*

[6] ISO 19296, *Mining — Mobile machines working underground — Machine safety*

[7] ISO 21815-1:—[3)], *Earth-moving machinery — Collision warning and avoidance — Identification and terms and definitions*

[8] ISO 22242:2005, *Road construction and road maintenance machinery and equipment — Basic types — Identification and description*

[9] SAE J1939 — *Recommended Practice for a Serial Control and Communications Heavy Duty Vehicle Network*

[10] SAE J1939-02 — *Agricultural and Forestry Off-Road Machinery Control and Communication Network*

3) Under preparation. Stage at the time of publication: ISO/DIS 21815-1:2020.

ICS ISO ics

Price based on 82 pages