Ford Motor Company

Global Diagnostic Specification

Part One

Worldwide Requirements

Version 2003.0
Date Issued: 25 April 2003

Research & Vehicle Technology Office
Electrical/Electronic Systems Engineering
Core Network Communications

REVISION HISTORY

Previous Version	Current Release	Current Version	Version Description	Author	Date
v4.2	DS-3N1T-1A294-AA	v5.0	Addition of CAN requirements and other updates.	S. Dilodovico (SDILODOV) 24-88815	20-December-1999
v5.0	N/A	v2001.0	Removal of all CAN specific references. Clarifications and additions of diagnostic strategies. CAN requirements are located in the Ford CAN Generic Diagnostic Specification.	J. Miller (JMILLE72) 24-88815	16-February-2001
v2001.0	N/A	v2001.1	Miscellaneous Clarifications	J. Miller (JMILLE72) 24-88815	30-November-2001
v2001.1	N/A	v2003.0	Miscellaneous Clarifications	J. Miller (JMILLE72) 24-88815	25-April-2003

PLEASE READ

This is a dynamic document within FORD Motor Company that will be periodically updated as required to make recommended and/or necessary improvements. Any recommendations for improving this document would be greatly appreciated. Please forward any and all suggestions, comments, and/or questions regarding this specification, via email, to the author(s) listed for the most recent revision date.

CHANGE LOG

Release	Previous Section	Current Section	Change Description
v2003.0	All	All	Inclusion of Errata sheet details from the Global Diagnostic Specification Part 1 v2001.1 Errata Sheet dated 03/05/2002.
v2003.0	1.3	1.3	Added clarification in regards to the deviation process.
v2003.0	3.2.3	3.2.3	Deleted reference to EESYS SDS EY-0105.
v2003.0	Table 16-8	Table 16-8	Added examples for 2 character Basic Designs in Part Number suffix.
v2003.0	Appendix A.2	Appendix A.2	Clarified the meaning of the affirmative ($00) response code.
v2003.0	Appendix A.2	Appendix A.2	Clarified the reporting of the General Response ($7F).
v2003.0	Appendix B	Appendix B	Modified reference to WCR to reference EESYS SDS instead.

Table of Contents

1 Introduction 13
1.1 Purpose / Scope 13
1.2 WWDiag Documentation Structure 13
1.2.1 Core Specifications Used to Implement Diagnostics and Configuration 13
1.3 Deviations 13

2 Diagnostic Development Process 14
2.1 Purpose / Scope 14
2.2 Benefits of Diagnostics 14
2.3 Subsystem Design Specification and FMEAs 14
2.3.1 I/O Fault Probability Ranking 14
2.4 Diagnostic Trouble Codes (DTCs) & Fault Correlation 14

3 Link-Based Diagnostic System Overview 16
3.1 Purpose / Scope 16
3.2 ECU Diagnostic System Components 16
3.2.1 Electronics Control Unit (ECU) 16
3.2.2 Vehicle Serial Communications Link 17
3.2.2.1 ISO-9141-Ford Serial Communications Link 17
3.2.2.2 SCP Serial Communications Link 17
3.2.2.3 Interaction of SCP-DIAG and SCP-CARB 17
3.2.2.4 UBP Serial Communications Link 17
3.2.3 Data Link Connector (DLC) 18
3.2.4 Tester 19
3.2.5 Service/Repair Technician 19
3.2.6 Shop Manual 19

4 Diagnostic Structure and Modes 20
4.1 ECU Diagnostic Structure 20
4.1.1 Operational State - Mandatory 20
4.1.2 Diagnostic State - Mandatory 20
4.1.3 Secure Diagnostic State - Optional 20
4.1.4 Execution Routine State - Mandatory 20
4.1.5 No Stored Codes Logging State - Mandatory for UBP and SCP 20
4.1.6 Information Transfer State - Optional 20
4.1.7 Gateway State - Optional 20
4.1.8 Input Integrity Test (IIT) State - Optional 20
4.2 Diagnostic Modes 21
4.2.1 General Response Message (Mode $7F) 21
4.2.2 Diagnostic State Entry Request (Mode $10) 22
4.2.3 Request Diagnostic Freeze Frame Data (Mode $12) 22
4.2.4 Report Diagnostic Freeze Frame Data (Mode $52) 22
4.2.5 Request Stored Codes (Mode $13) 22
4.2.6 Report Stored Codes (Mode $53) 22
4.2.7 Request Clear Stored Codes (Mode $14) 22
4.2.8 Operational State Entry Request (Mode $20) 22
4.2.9 Request Parameter By PID (Mode $22) 22
4.2.10 Report Parameter By PID (Mode $62) 23
4.2.11 Request Parameter By DMR (Mode $23) 23
4.2.12 Report Parameter By DMR (Mode $63) 23
4.2.13 Request Parameter Scaling/Masking (Mode $24) 23
4.2.14 Report Parameter Scaling/Masking (Mode $64) 23
4.2.15 Stop Transmitting Requested Data (Mode $25) 23

4.2.16 Request Security Access (Mode $27) ... 23
4.2.17 Report Security Access (Mode $67) ... 24
4.2.18 Request Disable Normal Message Transmission (Mode $28) ... 24
4.2.19 Request Diagnostic Data Packet(s) (Mode $2A) ... 24
4.2.20 Report Data Packet(s) (Mode $6A) ... 24
4.2.21 Dynamically Define Diagnostic Data Packet (Mode $2C) ... 24
4.2.22 Input/Output Control By PID (Mode $2F) ... 24
4.2.23 Request Diagnostic Routine Entry (Mode $31) ... 24
4.2.24 Request Diagnostic Routine Exit (Mode $32) ... 24
4.2.25 Request Diagnostic Routine Results (Mode $33) ... 24
4.2.26 Report Diagnostic Routine Results (Mode $73) ... 25
4.2.27 Request Download Routine Entry (Mode $34) ... 25
4.2.28 Request Upload Routine Entry (Mode $35) ... 25
4.2.29 Block Data Transfer Message (Mode $36) ... 25
4.2.30 Request Block Transfer Exit (Mode $37) ... 25
4.2.31 Request Write Memory Block (Mode $3B) ... 25
4.2.32 Request Read Memory Block (Mode $3C) ... 25
4.2.32.1 Report Contents Of Memory Block (Mode $7C) ... 25
4.2.33 Tester Present Diagnostic Message (Mode $3F) ... 25
4.2.34 No Stored Codes Logging State Entry Request (Mode $B0) ... 26
4.2.35 Diagnostic Command (Mode $B1) ... 26
4.2.36 Input Integrity Test State Entry Request (Mode $B2) ... 26
4.2.37 Request Manufacturer State Entry (Mode $B4) ... 26
4.3 Diagnostic Response Messages ... 26

5 General Requirements ... 27
5.1 Purpose / Scope ... 27
5.2 Diagnostic Message Definition ... 27
5.3 Diagnostic Data Byte Format ... 27
5.4 ECU Power-up Requirements ... 27
5.5 Communication Protocol Information ... 28
5.5.1 Master/Slave Configuration ... 28
5.5.2 Rapid-Data Diagnostic Message Priority ... 28
5.5.3 Unsupported Message Strategy ... 28
5.5.4 SCP, UBP And ISO-9141-Ford Target/Source Address Assignments ... 28
5.6 Tester Re-transmission of Diagnostic Messages ... 28
5.7 MRDB Implementation Requirements ... 28
5.8 Diagnostic Session Requirements ... 29
5.8.1 Normal Mode Messages ... 29
5.8.2 Continuous Code Management in the Diagnostic State ... 29
5.8.3 I/O Relating to Diagnostic Sessions (Excluding IIT and NSCL States) ... 29
5.8.4 Return to Operational State Timing ... 29
5.8.5 On-demand DTC Clearing ... 29
5.8.6 Diagnostic Session Timer ... 29
5.8.7 Vehicle Speed Timeout ... 30
5.9 Safety and Damage Prevention Requirements ... 30
5.10 OBD Compliance ... 30

6 Operational State Diagnostics ... 31
6.1 Purpose / Scope ... 31
6.2 Operational State Modes ... 31
6.3 DTC Requirements ... 31
6.3.1 DTC Classifications ... 32
6.3.2 Continuous DTCs ... 32
6.3.2.1 Continuous DTC Clearing Counters ... 32
6.3.2.2 Continuous DTC Memory Requirements ... 32
6.3.2.2.1 Guidelines for Continuous DTC Updating of All DTCs ... 32

6.3.2.3 Reporting Continuous DTCs (Ford-9141, SCP and UBP) 34
6.3.2.4 Clearing Continuous DTCs 34
6.3.2.4.1 Manual Clearing of Continuous DTCs 34
6.3.2.4.2 Selective Clearing of DTCs 35
6.3.2.4.3 Automatic Clearing of Continuous DTCs 35
6.3.2.5 Freeze Frame Associated With Continuous DTCs - Optional 35
6.3.2.6 Mandatory Continuous DTCs 36
6.3.2.6.1 ECU is Faulted ($9342) 36
6.3.2.6.2 Module Configuration Failure 36
6.4 Network Communication Diagnostic Fault Strategy 36
6.4.1 Network Fault Types 36
6.4.2 Network Communication DTCs 36
6.4.2.1 SCP Invalid Data DTCs 36
6.4.2.1.1 Example: 36
6.4.2.2 UBP Invalid Data DTCs 37
6.4.2.2.1 Example: 37
6.4.2.3 Invalid Data Network Faults 37
6.4.2.3.1 Digital Invalid Data Network Fault Strategy (All protocols) 37
6.4.2.3.2 Digital Invalid Data Network Fault Strategy (SCP and UBP) 37
6.4.2.3.3 Analog Invalid Data Network Fault Strategy (SCP and UBP) 38
6.4.2.3.4 Combined Digital and Analog Invalid Data Network Fault Strategy 38
6.4.2.4 Missing Message Network Faults (Non ISO-9141-FORD) 38
6.4.2.4.1 Missing Message Fault Code Logging 39
6.4.2.4.2 Optional Missing Message Fault PID Pointer Strategy (UBP and SCP Only) 39
6.4.2.4.3 Missing Message Fault PID Strategy for SCP 39
6.4.2.4.4 Missing Message Fault PID Strategy for UBP 39
6.4.2.5 Test Tool Missing Message Network Test 40

7 No Stored Codes Logging (NSCL) State Requirements 41
7.1 Purpose / Scope 41
7.2 NSCL State Diagnostic Requirements 41
7.2.1 Entering the No Stored Codes Logging State 42
7.2.2 NSCL State to Operational State Timing 42
7.2.3 Exiting the No Stored Codes Logging State to the Operational State 42
7.2.4 Normal Message Transmission Disabling 42
7.2.5 NSCL State Tester Requirements 42

8 Diagnostic State Requirements 43
8.1 Purpose / Scope 43
8.2 Entering the Diagnostic State 43
8.2.1 Entering Secure Diagnostic State From Diagnostic State 43
8.3 Diagnostic State Modes 43

9 Secure Diagnostic State Requirements 45
9.1 Purpose / Scope 45
9.2 Entering the Secure Diagnostic State 45
9.3 Secure Diagnostic State Modes 45
9.4 Security Levels 46
9.4.1 Active Security Level 46
9.4.2 ECU Response to a Lack of Security Access 46
9.5 Seed and Key Procedure 46

10 Execution Routine State Requirements 48
10.1 Purpose / Scope 48
10.2 Execution Routine State Entry Requirements 48
10.3 Execution Routine State Exit Requirements 49
10.3.1 Exiting a Diagnostic Test to the Diagnostic State 49
10.4 On-demand Diagnostic Trouble Codes 49
10.4.1 Reporting On-demand DTCs 49

10.4.2 Maintaining On-demand DTCs ... 50
10.4.3 Clearing On-demand DTCs ... 50
10.4.4 On-demand Fault PIDs ... 50
10.5 Diagnostic Tests ... 51
10.5.1 On-demand Self-test ($02) - Mandatory ... 51
10.5.1.1 On-Demand Self-Test Guidelines ... 51
10.5.1.2 On-demand Self-test Execution Time ... 52
10.5.1.3 On-demand Self-test I/O Circuit Omission ... 52
10.5.2 Assembly Self-test ($11) - Optional ... 52
10.5.3 ECU Specific Functional Tests - Optional ... 52
10.5.4 Execute Downloaded Routine ($05) - Optional ... 52
10.5.5 Logging Diagnostic Test Results ... 52
10.5.6 ECU Inputs Not in Expected State ... 53

11 Information Transfer State Requirements ... 54
11.1 Purpose / Scope ... 54
11.2 Information Transfer State Modes ... 54
11.3 Entering the Information Transfer State ... 54
11.4 Data Block Download Procedure - Optional ... 54
11.5 Exiting the Information Transfer State ... 55
11.6 Data Block Upload Procedure - Optional ... 55

12 Input Integrity Test State ... 58
12.1 Purpose / Scope ... 58
12.2 Input Integrity Test State Summary ... 58
12.3 Input Integrity Test State Modes ... 58

13 Manufacturer State Requirements ... 60
13.1 Purpose / Scope ... 60
13.2 Entering the Manufacturer State ... 60
13.3 Manufacturer State Operation ... 60
13.4 Exiting the Manufacturer State ... 60
13.4.1 Diagnostic Session Timeout ... 60

14 Gateway State Diagnostic Requirements ... 61
14.1 Purpose / Scope ... 61
14.2 Gateway Module ... 61
14.3 Gateway State ... 61
14.3.1 Entering Gateway State ... 61
14.3.1.1 Gateway Module in Gateway State ... 62
14.3.1.2 Gateway State Message Handling ... 62
14.3.1.2.1 Messages: Tester to Gateway ... 62
14.3.1.2.2 Messages: Gateway to Satellite ... 62
14.3.1.2.3 Messages: Satellite to Gateway ... 63
14.4 Gateway State Timing Requirements ... 63
14.5 Satellite Modules ... 63
14.5.1 Satellite Module in Diagnostic State ... 63
14.5.2 Satellite Module Mandatory PIDs ... 63
14.6 Redirecting Communication ... 63

15 Diagnostic Command Requirements ... 64
15.1 Purpose / Scope ... 64
15.2 Diagnostic Command Definition ... 64
15.3 Historical Command Usage and Definition ... 64
15.3.1 Diagnostic Command Message Information ... 64
15.4 Diagnostic Command Classifications ... 64
15.4.1 Direct Commands ... 64
15.4.2 State-Encoded (SED) Commands ... 64
15.4.3 Bit-Mapped (BMP) Commands ... 65

15.4.4 Numeric (NUM) Commands ... 65
15.5 Diagnostic Command Entry Criteria ... 65
15.6 Diagnostic Command Exit Criteria ... 65
15.7 Supplier Reserved Command Range ... 65

16 Parameter Identifier (PID) and Data Packet Requirements ... 66
16.1 Purpose / Scope ... 66
16.2 PID Types and Classifications ... 66
16.2.1 ASCII (ASC) PIDs ... 66
16.2.2 Binary Coded Decimal (BCD) PIDs ... 66
16.2.3 State Encoded (SED) PIDs ... 66
16.2.4 Bit-Mapped (BMP) PIDs ... 66
16.2.5 Numeric (NUM) PIDs ... 66
16.2.5.1 Unsigned Numeric PIDs ... 67
16.2.5.2 Signed Numeric PIDs ... 67
16.2.6 Packeted (PKT) PIDs ... 67
16.2.7 Output PIDs ... 67
16.2.7.1 Output State Monitoring ... 67
16.2.7.1.1 Example of a Single PID Supported for Multiple Outputs ... 67
16.2.7.2 Output Feedback PIDs ... 68
16.2.8 Input PIDs ... 68
16.3 PID ACCESS ... 68
16.3.1 PID Reading ... 68
16.3.2 PID Control ... 68
16.4 PID State Support ... 68
16.5 PID Scaling and Masking Capability - Optional ... 68
16.6 Supplier Reserved PID Range ... 69
16.7 Mandatory PIDs ... 69
16.7.1 Number of Continuous DTCs (PID $0200) ... 70
16.7.2 Number of Trouble Codes Set Due to Diagnostic Test (PID $0202) ... 70
16.7.3 ECU Operating State (PID $D100) ... 70
16.7.4 Software Version Number (PID $E200) ... 70
16.7.5 Part Number Identification Base (PID $E217) ... 71
16.7.6 Part Number Identification Prefix (PID $E21A) ... 73
16.7.7 Part Number Identification Suffix (PID $E219) ... 74
16.8 Miscellaneous Identification PIDs ... 75
16.8.1 Serial Number (PIDs $E221 THRU $E226 & $E231 THRU $E236) ... 75
16.8.2 Vehicle Identification Number (PIDs $E300 THRU $E307) ... 76
16.9 Fault PID Description ... 76
16.9.1 Automatic Clearing of Continuous Fault PID Fault Bits ... 77
16.9.2 Tester Initiated Clearing of Continuous Fault PID Fault Bits ... 77
16.9.3 On-demand Fault PID Fault Bits ... 77
16.10 Define/Request/Stop Data Packets ... 77
16.10.1 Dynamically Define Diagnostic Data Packet (Mode $2C) ... 78
16.10.2 Request Diagnostic Data Packets(s) (Mode $2A) ... 78
16.10.3 Stop Transmitting Requested Data (Mode $25) ... 78

17 UART - Based Diagnostic Protocol (UBP) ... 79
17.1 Purpose / Scope ... 79
17.2 UBP Enhanced Diagnostic Protocol Description ... 79
17.3 UBP Serial Communications Link ... 79
17.4 Grounding Requirements ... 79
17.5 Message Structure ... 79
17.5.1 Generic Message Format ... 80
17.5.1.1 Format Byte ... 80
17.5.1.1.1 Message Type ... 80
17.5.1.1.2 Message Length ... 81

17.5.1.2 Target Byte 81
17.5.1.2.1 Functionally Addressed Diagnostic Messages 81
17.5.1.2.2 Physically Addressed Diagnostic Messages 82
17.5.1.3 Source Byte 82
17.5.1.4 Data Byte(s) 82
17.5.1.4.1 Mode Encoding 83
17.5.1.4.2 Other Encoding 83
17.5.1.5 Checksum Byte 83
17.6 Legislative OBD Emission Related Diagnostics on UBP 83
17.7 Message Timing Requirements 83
17.7.1 Interbyte Gap 83
17.7.2 Intermessage Gap 83

18 ISO-9141-Ford Serial Communications Link Protocol 85
18.1 Purpose / Scope 85
18.2 ISO-9141-Ford Serial Communications Link 85
18.3 ISO-9141-Ford Message Structure 85
18.3.1 ISO-9141-Ford Message Type Byte 86
18.3.2 ISO-9141-Ford Target Address Byte 86
18.3.3 ISO-9141-Ford Source Address Byte 86
18.3.4 ISO-9141-Ford Mode Byte 86
18.3.5 ISO-9141-Ford Optional Data Bytes 87
18.3.6 ISO-9141-Ford Checksum Byte 87
18.3.7 ISO-9141-Ford Message Timing Requirements 87
18.3.8 Interbyte Gap 87
18.3.9 Intermessage Gap 88

19 Serial Communications Protocol (SCP) 90
19.1 Purpose / Scope 90
19.2 Ford SCP Protocol Description 90
19.2.1 Overview 90
19.2.2 SCP Serial Communications Link 90
19.2.3 SCP Message Structure 90
19.2.3.1 SCP Target Address Byte 91
19.2.3.2 SCP Source Address Byte 91
19.2.3.3 SCP Data Byte 1 91
19.2.3.4 SCP Optional Data Bytes 92
19.2.3.5 SCP Cyclic Redundancy Code (CRC) Byte 92
19.2.3.6 SCP Acknowledgment Byte 92
19.2.4 SCP Diagnostic Description 92
19.2.4.1 Diagnostic Message Characteristics 92
19.2.5 SCP-CARB (California Air Resources Board) Capabilities 92
19.2.6 Ford SCP Message Timing Requirements 92
19.2.7 Interbyte Gap 92
19.2.8 Intermessage Gap 92

Appendix A – Diagnostic Messages 94
A.1 Table Descriptions 94
A.2 General Response [7F] 94
A.3 Request Diagnostic State Entry [10] 96
A.4 Request Diagnostic Freeze Frame Data [12] 97
A.5 Report Diagnostic Freeze Frame Data [52] 98
A.6 Request stored codes [13] (Continuous DTCs) 99
A.7 Report Stored Codes [53] 100
A.8 Request Clear Stored Codes [14] 101
A.9 Request Operational State Entry [20] 102
A.10 Request Parameter by PID [22] 103
A.11 Report Parameter by PID [62] 104

A.12 Request Parameter by DMR [23] 105
A.13 Report Parameter by DMR [63] 106
A.14 Request Parameter Scaling/Masking [24] 107
A.15 Report Parameter Scaling/Masking [64] 108
A.16 Stop Transmitting Requested Data [25] 109
A.17 Request Security Access [27] 110
A.18 Report Security Seed [67] 112
A.19 Request Disable Normal Message Transmission [28] 113
A.20 Request Diagnostic Data Packets [2A] 114
A.21 Report Data Packet [6A] 116
A.22 Dynamically Define Diagnostic Data Packet [2C] 117
A.23 Input Output Control by PID [2F] 119
A.24 Request Diagnostic Routine Entry [31] 120
A.25 Request Diagnostic Routine Exit [32] 121
A.26 Request Diagnostic Routine Results [33] (On-demand DTCs) 122
A.27 Report Diagnostic Routine Results [73] 123
A.28 Request Download Routine Entry [34] 124
A.29 Request Upload Routine Entry [35] 126
A.30 Block Data Transfer Message [36] 128
A.31 Request Block Transfer Exit [37] 129
A.32 Request Write Memory Block [3B] 130
A.33 Request Read Memory Block [3C] 131
A.34 Report Contents of Memory Block [7C] 132
A.35 Tester Present [3F] 133
A.36 Request No Stored Codes Logging State [B0] 134
A.37 Diagnostic Command [B1] 135
A.38 Request Input Integrity Test State [B2] 136
A.39 Request Manufacturer State Entry [B4] 137

Appendix B – References 138
B.1 General 138

Appendix C – Abbreviations, Acronyms and Glossary 140
C.1 Purpose 140
C.2 Abbreviations 140
C.3 Acronyms 147
C.4 Terms 148

List of Figures

Figure 3.1 Vehicle Diagnostic Setup 16
Figure 3.2 Data Link Connector Pins 18
Figure 4.1 State Model Diagram 21
Figure 6.1 Example of Memory when Storing all Continuous DTCs 33
Figure 6.2 Example of Memory when Storing Only 15 of 24 Continuous DTCs 33
Figure 9.1 Security Access Procedure 47
Figure 11.1 Data Block Upload Procedure with Handshake 56
Figure 11.2 Data Block Upload Procedure without Handshake 57
Figure 14.1 Example of Multi-Module Gateway System. 61

List of Tables

Table 1.1 Core Diagnostic Specifications 13
Table 3.1 Data Link Connector Pin Numbers and Descriptions 19
Table 5.1 Message Data Byte Descriptions 27
Table 5.2 Diagnostic Data Byte Format 27
Table 5.3 Diagnostic Message Byte Layout 27
Table 6.1 Mandatory Diagnostic Modes for Operational State 31
Table 6.2 Optional Diagnostic Modes for Operational State 31
Table 6.3 Example of Memory Usage When Storing All Continuous DTCs Vs Storing Only 15 Continuous DTCs For an ECU With 24 Continuous DTCs 34
Table 6.4 Consecutive Messages Returned When Reporting DTCs 34
Table 6.5 Mandatory DTCs. 36
Table 6.6 Missing Message DTCs 39
Table 7.1 Mandatory Diagnostic Modes for the No Stored Codes Logging State 41
Table 7.2 Optional Diagnostic Modes for No Stored Codes Logging State 41
Table 8.1 Mandatory Diagnostic Modes for Diagnostic State 43
Table 8.2 Optional Diagnostic Modes for Diagnostic State 44
Table 9.1 Mandatory Diagnostic Modes for Secure Diagnostic State 45
Table 9.2 Optional Diagnostic Modes for Secure Diagnostic State 46
Table 10.1 Mandatory Diagnostic Modes for Execution Routine State 48
Table 10.2 Optional Diagnostic Modes for Execution Routine State 48
Table 10.3 Example of “sticky” behavior of on-demand fault PIDs 51
Table 11.1 Mandatory Diagnostic Modes for Information Transfer State 54
Table 11.2 Optional Diagnostic Modes for Information Transfer State 54
Table 12.1 Mandatory Diagnostic Modes for Input Integrity Test State 58
Table 12.2 Optional Diagnostic Modes for Input Integrity Test State 59
Table 15.1 Command Data Type Classifications 64
Table 16.1 PID Data Type Classifications 66
Table 16.2 Encoding of Parameter Information Byte (Data Byte 4, Mode $64) 69
Table 16.3 Mandatory Supported PIDs 70
Table 16.4 Applicable ECU Operating States (PID $D100) 70
Table 16.5 Software Version Number (PID $E200) 71
Table 16.6 Base Part Number Structure (PID $E217) 72
Table 16.7 Part Number Identification Prefix (PID $E21A) 73
Table 16.8 Part Number Identification Suffix (PID $E219) 75
Table 16.9 Serial Number and VIN PIDs 75
Table 16.10 Example of ASCII Module Serial Number Storage 76
Table 16.11 Example of BCD Module Serial Number Storage 76
Table 16.12 Example of VIN Storage 76

Table 17.1 Generic Message Format 80
Table 17.2 Format Byte Encoding 80
Table 17.3 Type Field Encoding 81
Table 17.4 Length Field Encoding 81
Table 17.5 Format of Functional UBP Request No Stored Codes Logging Message 82
Table 17.6 Physically Addressed Diagnostic Message Format 83
Table 18.1 ISO-9141-FORD Message Format 86
Table 19.1 SCP Message Structure 91

1 Introduction

1.1 Purpose / Scope

The purpose of this specification is to define requirements and to provide guidelines for the development of electrical system diagnostics that will be used to functionally evaluate serial communications link Electronic Control Units (ECUs) and their associated subsystems. This specification also defines requirements and provides general information necessary for the development of diagnostic service tools.

This document supports recommended diagnostic practices specified by the Society of Automotive Engineers SAE and ISO committees *1930[2], SAE J1962[3], SAE J2012[4], J2190[5], J2186[6],* and *ISO 15031[25]*.

1.2 WWDiag Documentation Structure

This document is the main specification required for the implementation of vehicle serial communications link ECU diagnostics on UBP, SCP, and Ford-9141 protocols. Other related diagnostic specifications are detailed below.

1.2.1 Core Specifications Used to Implement Diagnostics and Configuration

Specification	***Description***
Global Diagnostic Specification (Part 1)	Definition of requirements and guidelines for ECU diagnostic services on serial communication links.
CAN Generic Diagnostic Specification	Definition of requirements and guidelines for ECU diagnostic services on CAN based links.
Module Programming and Configuration Specification	Lists the various methods of configuration and programming available to all ECUs. It utilizes much of the information specified in this document such as standard diagnostic modes and services. Any configuration feature or variable calibration parameter associated with a module is governed by the *Module Programming and Configuration Specification.*
Subsystem Specific Diagnostic Specification	Commonly referred to as SSDS – Part II, it provides specific details about the diagnostic capabilities (commands, execution routines, PIDs, DTCs, etc.) that an ECU shall support. In addition, strategies used by service tools such as security access and ABS bleed procedures are included. All manufacturer or supplier specific services that are supported shall be included in this specification. An ECU shall return a general response $7F with a response code of $11 (mode not supported) for any service not listed as supported.
Vehicle Specific Configuration Specification	Contains specifics for configuration of all ECUs for a given vehicle line.

Table 1.1 Core Diagnostic Specifications

1.3 Deviations

Any non-compliance to this specification, for any ECU, is a deviation and shall follow the appropriate process for resolution. Any deviation shall be fully documented and adequately explained in the Subsystem Specific Diagnostic Specification (Part 2) for the ECU. In addition to the normal deviation process, any deviations must be approved by FCSD / DSP. Note that failure to comply with the requirements outlined in this specification may result in lack of diagnostic availability in time to sell/ship vehicles.

2 Diagnostic Development Process

2.1 Purpose / Scope

This section explains the process for developing accurate, serial link ECU diagnostics. It also explains their relationship with the development of service manual procedures.

2.2 Benefits of Diagnostics

A partial list of benefits for implementing serial link diagnostics in an ECU are as follows:

- Customer satisfaction by achieving the *Dealer to Customer Standard - Fix it Right the First Time* repairs (eliminating repeat repairs).
- Warranty Cost Reduction by:
  1. Reducing Trouble Not Identified (TNI) rates
  2. Reducing repeat repairs
  3. Reducing the time required to isolate failures
  4. Reducing the time needed to access hard to reach areas

2.3 Subsystem Design Specification and FMEAs

Each Subsystem Design Specification (SDS) explains the application of a vehicle's subsystem, and is generated by a subsystem developer from the functional requirements of the subsystem's features. The subsystem developer is responsible for the development of the subsystem's Failure Modes & Effects Analysis (FMEA) and diagnostics.

2.3.1 I/O Fault Probability Ranking

The following I/O fault list shows the ranking of the most to least probable occurrence, for the faults identified:

1. Open Circuit
2. Intermittent Open Circuit
3. Intermittent Short-to-Ground
4. Short-to-Ground
5. Intermittent Short-to-Battery
6. Signal Faults
7. Wiring Routing Faults

NOTE: An Open Circuit is the most prevalent type of I/O fault and shall be given a higher priority consideration for diagnostic coverage.

2.4 Diagnostic Trouble Codes (DTCs) & Fault Correlation

ECU fault symptoms recorded and identified in a FMEA summary document shall be correlated to Diagnostic Trouble Codes (DTCs) either directly or through the use of fault PIDs. See Section 16.9 for more information regarding fault PIDs.

DTC information may be further correlated to I/O specific Parameter Identifier (PID) and command information in the Subsystem Specific Diagnostic Specification for each ECU. The Subsystem Specific

Diagnostic Specification provides an arranged snapshot of subsystem specific diagnostic capabilities that should be used as a reference in the development of service manual pinpoint tests and test tool diagnostic flow diagrams.

3 Link-Based Diagnostic System Overview

3.1 Purpose / Scope

This section provides an overview of the diagnostic system (vehicle, ECU, service technician, tester and service manual) to serve as an introduction to the detailed information presented in subsequent sections.

3.2 ECU Diagnostic System Components

The following subsections discuss how each element fits into the diagnostic evaluation process. Refer to Figure 3.1 for a typical setup of a vehicle undergoing diagnostic evaluation.

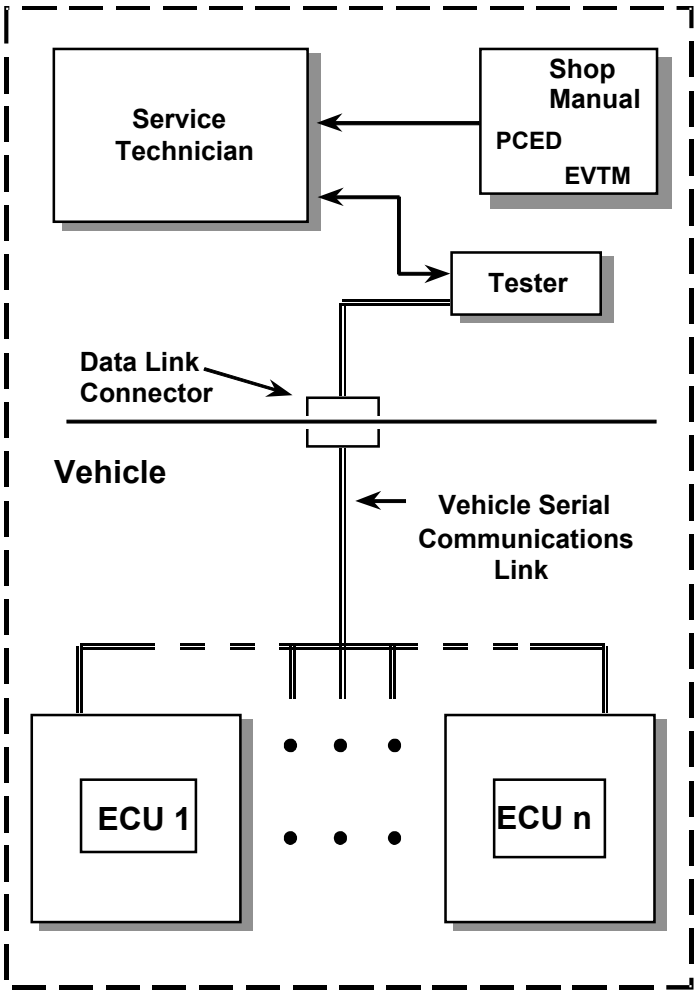


Figure 3.1 Vehicle Diagnostic Setup

3.2.1 Electronics Control Unit (ECU)

An ECU can be the microprocessor-based "brains" of all or a portion of each of the vehicle subsystems. Each ECU receives input data from various switches, sensors, etc. and processes that data to control the ECU's outputs. Each ECU may support communication over the communications link and may also receive data from other ECUs on the link.

Each ECU contains on-board diagnostics (as specified in the Subsystem Specific Diagnostic Specification for that ECU) that allows the ECU and a set of its subsystem components to be evaluated via the vehicle's serial communication link.

3.2.2 Vehicle Serial Communications Link

The vehicle serial communications link is a physical interface between each of the vehicle's ECUs and the vehicle data link connector (DLC). It provides the capability for tester-to-ECU and potentially ECU-to-ECU communications. Actual communication occurs in accordance with a communications protocol uniquely defined for each of the Ford approved serial links.

- This specification provides diagnostic information for ECUs which incorporate one of the Ford approved communication protocols specified by the Multiplex Standards Organization. A listing of the currently approved and defined protocols can be obtained at the *Multiplex Standards web-page*[10].

NOTE: It is possible, and even probable, for more than one vehicle serial communications link to be on a vehicle simultaneously.

3.2.2.1 ISO-9141-Ford Serial Communications Link

The ISO-9141-Ford serial communications link, which is a Ford interpretation of the ISO-9141 specification, is achieved over a single wire connection between the tester and each of the vehicle's ECUs via the DLC. The ISO-9141-Ford serial communications link protocol operates node-to-node using a master-slave relationship. The tester (master) initiates all communications with the ECU(s) (slave) undergoing test.

Information pertaining to the diagnostic portions of the ISO-9141 protocol resides in Section 18 of this document. ECU physical layer information resides in the *Ford-9141 Protocol Definition and Interface Requirements Specification*[9].

NOTE: ISO-9141-Ford only permits tester-to-ECU communication; it does not support ECU-to-ECU communication.

3.2.2.2 SCP Serial Communications Link

The SCP serial communications link (based on *SAE J1850*[13]) uses a twisted wire pair to provide a physical communications path between each of the vehicle's SCP ECUs and the tester. The tester connects to the vehicle's DLC. The SCP link supports both ECU-to-ECU communication and tester-to-ECU communication.

Information pertaining to the diagnostic portions of the SCP protocol resides in Section 19 of this document. SCP vehicle network implementation and physical layer information can be found in *SCP Vehicle Network Implementation Requirements*[14] and the *SCP Protocol Definition and Interface Requirements*[15]

NOTE: For diagnostics, all SCP communications between a tester and ECU are initiated by the tester.

3.2.2.3 Interaction of SCP-DIAG and SCP-CARB

SCP-DIAG and SCP-CARB communication may co-exist without interference on the same network.

3.2.2.4 UBP Serial Communications Link

The UBP serial communications link uses a single wire to provide a physical communications path between each of the vehicle's UBP ECUs and the tester. The tester connects to the vehicle's DLC. The UBP link supports both ECU-to-ECU communication and tester-to-ECU communication.

Information pertaining to the diagnostic portions of the UBP protocol resides in Section 17 of this document. UBP vehicle network implementation and physical layer information can be found in *UBP Vehicle Network Implementation Requirements*[23] and the *UBP Protocol Definition and Interface Requirements*[24].

NOTE: For diagnostics, all UBP communications between a tester and ECU are initiated by the tester.

3.2.3 Data Link Connector (DLC)

The data link connector (or central diagnostic connector) is the mandatory electrical connector required for On Board Diagnostics (OBD-II) emissions compliance. The DLC consists of two mating connectors, the vehicle connector and the test equipment connector. Tester systems attach to the DLC so they may communicate with each of the vehicle's ECUs over the serial communications link. It is located within the vehicle under the instrument panel. The data link connector is a standard corporate diagnostic connector that complies with the recommended practices specified in *SAE J1962*[3].

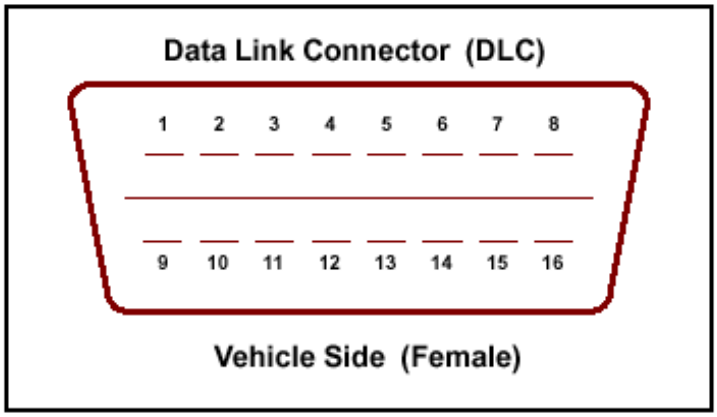


Figure 3.2 Data Link Connector Pins

Terminal Number	*Ford Terminal Assignment*	*Terminal Description*
1	Ignition Control	Activates low current switching device to power ignition current
2	SCP Bus (+)	Ford Standard Corporate Protocol (SCP) Bus (+)
3	UBP Network #1 **<OR>** Medium Speed CAN (High)	Ford UART Based Protocol (UBP) Network #1 **<OR>** Medium Speed Controller Area Network (CAN) data link (High)
4	Chassis Ground	
5	Signal Ground	
6	High Speed CAN (High)	High Speed Controller Area Network (CAN) data link (High)
7	K Line ISO-9141	ISO-9141-FORD and ISO-9141-CARB communication line
8	Trigger Signal	Multiple module trigger input controlled through the communication link to initiate / terminate an event.
9	Battery Power (Switched)	Vehicle battery power available via the ignition switch or ignition control (Pin 1)
10	SCP Bus (-)	Ford Standard Corporate Protocol (SCP) Bus (-)
11	UBP Network #2 **<OR>** Medium Speed CAN (Low)	Ford UART Based Protocol (UBP) Network #2 **<OR>** Medium Speed Controller Area Network (CAN) data link (Low)

12	Flash EEPROM	Reserved for Flash – Power programming and services up to 3 modules (Module assignment controlled by RVT EESE Network Communications section)
13	Flash EEPROM	Reserved for Flash – Power programming and services up to 3 modules (Module assignment controlled by RVT EESE Network Communications section)
14	High Speed CAN (Low)	High Speed Controller Area Network (CAN) data link (Low)
15	L Line ISO-9141	L Line is not used for Ford Motor Company
16	Battery Power (Unswitched)	Vehicle battery power available at all times (unswitched)

Table 3.1 Data Link Connector Pin Numbers and Descriptions

3.2.4 Tester

A tester provides the service technician with the user interface and control capabilities for non-intrusive testing of ECU subsystems that reside on the serial communications link. Non-intrusive testing is the ability to retrieve data from an ECU without having to physically disassemble a portion of the subsystem.

Example: Instead of using a voltmeter and performing a physical measurement, the voltage being supplied to an ECU may be requested by a technician using a tester and reported by the ECU, both over the serial communications link, with the tester displaying the ECU's voltage result to the technician.

3.2.5 Service/Repair Technician

Control of the diagnostic tester is performed by the service technician. Based on what symptoms are conveyed from the customer, etc., the technician can proficiently control the tester to perform required diagnostic tests to determine root cause information. Service manuals are often used in conjunction with the electronic tester, by the technician, during root cause analysis.

3.2.6 Shop Manual

The vehicle's service manual integrates the diagnostic capabilities of the ECU(s) and the tester into the procedures a service technician can follow to isolate and repair ECU failures.

The service manuals contain a section entitled *Section 418-00, Module Communications Network*$_{[16]}$ that provides the service technician with the diagnostic procedures necessary for isolating network communication faults.

Section 418-01, Module Configuration$_{[17]}$ in the service manual describes programmable module configuration, module configuration and lists all customer preference items.

4 Diagnostic Structure and Modes

4.1 ECU Diagnostic Structure

A state transition diagram that identifies the basic design structure (operating states and modes) of the ECU software is shown in Figure 4.1. Diagnostic states contain specific rules regarding which diagnostic services and functionality are supported. Diagnostic states aid in the partitioning of diagnostic functionality and help ensure diagnostic consistency between implementations.

4.1.1 Operational State - Mandatory

The operational state is the default state from which the ECU executes its normal operating strategy. A limited set of diagnostic functionality is supported in this state. See Section 6 for more detail.

4.1.2 Diagnostic State - Mandatory

A state the ECU enters, upon request from a tester, to execute its primary set of diagnostic functions as specified in the ECU's Subsystem Specific Diagnostic Specification. See Section 8 for more detail.

4.1.3 Secure Diagnostic State - Optional

A state the ECU enters, upon request from a tester only after a tester successfully performs the security access procedures to the ECU. The secure diagnostic state is a superset of the diagnostic state. See Section 9 for security access strategy.

4.1.4 Execution Routine State - Mandatory

A state the ECU enters, when requested by a tester, to execute a diagnostic routine (e.g., on-demand self-test). See Section 10 for more detail.

4.1.5 No Stored Codes Logging State - Mandatory for UBP and SCP

A state the ECU enters, when requested by the tester, to prevent logging of erroneous missing message continuous network DTCs. This state shall be supported only for UBP and SCP, which enable inter-module communications. A module in this state shall not log any DTCs and operate all functionality as if in Operational State. See Section 7 for more detail.

4.1.6 Information Transfer State - Optional

A state the ECU enters, when requested by the tester, to download or upload information. See Section 11 and the *Module Programming and Configuration Specification*[8] for more detail.

4.1.7 Gateway State - Optional

A state the ECU enters, when requested by the tester, to allow the tester to communicate with protocols that aren't accessible through the diagnostic link connector (DLC). See Section 12 (TBD) for more detail.

4.1.8 Input Integrity Test (IIT) State - Optional

A state the ECU enters, when requested by the tester, to allow the tester to obtain write access to a limited set of parameters while the ECU maintains normal functionality and run-time operations that are supported in the operational state. The IIT state provides additional functions to the tester such as input and output parameter control (e.g., modification of internal registers or control of output drivers). The intention of this state is to determine how the module reacts to these changes while performing the normal operational state functions.

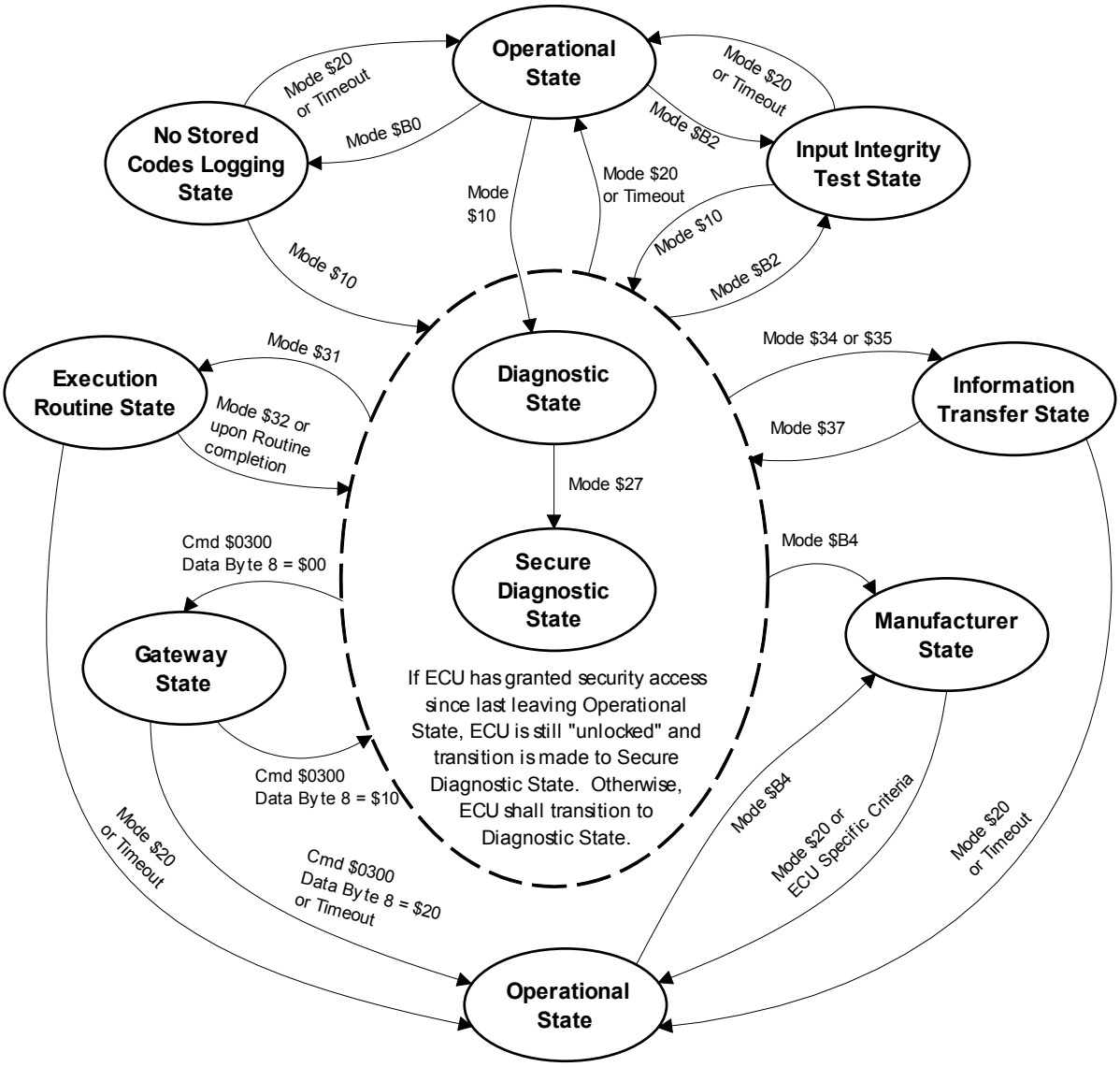


Figure 4.1 State Model Diagram

4.2 Diagnostic Modes

All diagnostic requests performed by a tester consist of diagnostic modes (services) and are based on the *Enhanced E/E Diagnostic Test Modes – J2190*[5] SAE standard. A listing of all of the Ford supported modes is described in this section. See "Appendix A" for a matrix of all of the defined requests and responses for each state of operation relating to each mode.

4.2.1 General Response Message (Mode $7F)

The general response message shall be used to provide an affirmative response to tester requests where no data is returned in the control module response. For example, a request to clear codes (mode $14) does not return any data, so the general response message is used to inform whether the codes were successfully cleared or not. Additionally, the general response message is used for all ECU negative responses to the tester. See "Appendix A" for detail on the specific formatting of this message.

4.2.2 Diagnostic State Entry Request (Mode $10)

This mode instructs an ECU to enter the diagnostic state, which is the state in which the majority of diagnostic functionality is enabled and where various normal mode functionality (e.g., DTC logging and input/output control) is disabled. The control module shall respond to the tester request with a general response message containing the affirmative response code. If the control module is unable to comply with the tester request the control module shall respond with a general response message containing the appropriate negative response code.

4.2.3 Request Diagnostic Freeze Frame Data (Mode $12)

This mode is used to request diagnostic freeze frame data from the ECU. It contains a one byte freeze frame number and an associated DTC number. Freeze frame data contains a snapshot of information on vehicle conditions (such as RPM, engine coolant temperature, etc.) that were present at the time the DTC was logged.

4.2.4 Report Diagnostic Freeze Frame Data (Mode $52)

This mode is used to report diagnostic freeze frame data from the ECU to a tester. It contains the freeze frame number, associated DTC number, and the freeze frame data requested with a corresponding mode $12.

4.2.5 Request Stored Codes (Mode $13)

This mode instructs an ECU to transfer all of its continuous DTCs to the tester. See Section 6.3.2 for a description of continuous codes. A module may respond to this message with one or more frames of information, depending on the number of codes present in a module.

4.2.6 Report Stored Codes (Mode $53)

This mode is transmitted from a control module, in response of a mode $13 request, to report all its continuous codes. Multiple mode $53 frames may be transmitted back-to-back in response to one mode $13 request.

4.2.7 Request Clear Stored Codes (Mode $14)

This mode instructs an ECU to clear all of its continuous codes.

The control module may erase/clear all of its continuous codes after receiving this request. Compliance to this request is dependent upon the operating conditions within the ECU.

4.2.8 Operational State Entry Request (Mode $20)

This mode instructs an ECU to return to the operational state (see Figure 4.1).

The control module shall respond to a mode $20 request with a general response message containing the affirmative response code. The control module must always comply with and respond to an operational state entry request with an affirmative response.

The control module is required to provide an affirmative response to the request operational state entry message if it is in the operational state at the time the request is received.

4.2.9 Request Parameter By PID (Mode $22)

Used to request module Parameter IDs (PIDs). A PID is a logical address assigned to specific module information that may be retrieved by a tester by specifying the logical PID address (i.e., the tester does not need to know the physical address of the information within the ECU to obtain the data). For example, PID $D100 is supported by all Ford ECUs and always contains a one byte state encoded value

that represents the current diagnostic state of the ECU (e.g., operational state, information transfer state, etc.). The method for obtaining this information is the same across all ECUs, although it is most likely stored in different physical locations on an engine controller versus an instrument cluster. Specific PID information shall be identified in the Subsystem Specific Diagnostic Specification document for each ECU and its subsystem. See Section 16 for more information.

4.2.10 Report Parameter By PID (Mode $62)

This mode is transmitted from a control module, in response of a mode $22 request, to report the current contents of the PID requested.

4.2.11 Request Parameter By DMR (Mode $23)

Used to request supported module DMR locations. Direct memory reference (DMR) is a service whereby information contained within an ECU can be retrieved by a tester by specifying the ECU's physical address of where the data is located in the module.

NOTE: DMRs are intended for development and manufacturing only.

4.2.12 Report Parameter By DMR (Mode $63)

This mode is transmitted from a control module, in response of a mode $23 request, to report the current contents of the DMR location requested.

4.2.13 Request Parameter Scaling/Masking (Mode $24)

This mode requests information about the type and format of the data stored in various PIDs.

Control modules that support scaling and masking shall support mode $24, see Section 16.5 for more information.

4.2.14 Report Parameter Scaling/Masking (Mode $64)

This mode is used to report information about the type and format of the data stored in various PIDs.

Control modules that support scaling and masking shall support mode $64, see Section 16.5 for more information.

4.2.15 Stop Transmitting Requested Data (Mode $25)

The mode $25 message is a request to discontinue the transmission of all repetitive data requested via any mode $2A requests. See Section 16.10.3 for more detail relating to this mode.

4.2.16 Request Security Access (Mode $27)

Used to gain access to various module functions that are either security, safety or other defined information sensitive systems. It employs capability for various levels of security access (e.g., one level for supplier access and another for Ford access).

Mode $27 performs two types of requests in addition to transferring the level of desired security access:

1. Requesting a seed from a control module.
2. Submitting a key to a control module.

Refer to Section 9 for further information relating to security access.

4.2.17 Report Security Access (Mode $67)

This mode is the positive response to a mode $27 message to obtain security access seeds and is only supported for control modules that support security access.

NOTE: Mode $67 isn't used as the response message for mode $27 key submission requests, only access seed requests are supported by this mode.

4.2.18 Request Disable Normal Message Transmission (Mode $28)

This mode is used to inhibit an ECU from transmitting normal operating data on the link. This allows more bandwidth for diagnostic messages, which is useful during programming. Another use is to allow the test equipment to emulate the ECU for diagnostic purposes. However, if an unsafe or undesirable vehicle operating condition would result from the lack of normal messages, then this mode should only cause all nonessential messages to be inhibited.

4.2.19 Request Diagnostic Data Packet(s) (Mode $2A)

This mode is used to request any defined data packet.

Various options can be specified to control either a one-shot report or a continuous stream of the packet(s) requested. Also, if a continuous stream of reporting of the parameters is specified, then three different rates of reporting may be defined for each module. See Section 16.10 for further detail.

4.2.20 Report Data Packet(s) (Mode $6A)

This mode reports defined Data Packet Identifications DPIDs requested with mode $2A.

A DPID shall always fit within a single, mode $6A, frame. See Section 16.10 for further detail.

4.2.21 Dynamically Define Diagnostic Data Packet (Mode $2C)

This mode is used to dynamically define data packets that are requested with a mode $2A message. See Section 16.10 for further detail

4.2.22 Input/Output Control By PID (Mode $2F)

This mode provides the capability to control various input and output parameters associated with PIDs.

It can be used to de-couple input parameter values to an ECU and substitute them with ones provided with this mode. Also, it can be used to directly write to an output parameter.

4.2.23 Request Diagnostic Routine Entry (Mode $31)

Instructs an ECU to run a pre-defined execution routine (e.g., test number 02 for self-test). See Section 10 for further detail.

4.2.24 Request Diagnostic Routine Exit (Mode $32)

Instructs an ECU to either conditionally or unconditionally terminate a diagnostic routine. See Section 10 for further detail.

4.2.25 Request Diagnostic Routine Results (Mode $33)

Requests an ECU to report on-demand DTCs that occurred during the most recent instance of an execution routine. There must be an active diagnostic session and the most recent instance of the execution routine must have occurred during the existing diagnostic session. See Section 10 for further detail.

4.2.26 Report Diagnostic Routine Results (Mode $73)

Reports on-demand DTCs that occurred during the most recent instance of an execution routine.

There must be an active diagnostic session and the most recent instance of the execution routine must have occurred during the existing diagnostic session. See Section 10 for further detail.

4.2.27 Request Download Routine Entry (Mode $34)

Instructs an ECU to enter the information transfer state and begin transferring information from a tester to an ECU with the control information specified in this mode.

Contains information regarding the amount of data to be downloaded, format of transfer and the address within the ECU to where the data is to be stored.

4.2.28 Request Upload Routine Entry (Mode $35)

Instructs an ECU to enter the information transfer state and begin transferring information from an ECU to a tester with the control information specified in this mode.

Contains information regarding the amount of data to be uploaded, format of transfer and the address within the ECU from where the data is to be read.

4.2.29 Block Data Transfer Message (Mode $36)

This mode contains the data transferred to and from an ECU during download and upload procedures.

4.2.30 Request Block Transfer Exit (Mode $37)

This mode is used to complete a data transfer by instructing an ECU to exit Information Transfer State (ITS).

4.2.31 Request Write Memory Block (Mode $3B)

This mode is used to request module memory writes associated with configuration and calibration of ECUs. It contains a logical, one byte, address and a small amount of data to be downloaded to an ECU (i.e. up to 5 bytes). Refer to the *Module Programming and Configuration Specification*[8] for more information on this mode.

4.2.32 Request Read Memory Block (Mode $3C)

This mode is used to request module memory reads associated with configuration and calibration of ECUs. It contains a logical, one byte, address of where to read small amounts of data from (i.e. up to 5 bytes). Refer to the *Module Programming and Configuration Specification*[8] for more information on this mode.

4.2.32.1 Report Contents Of Memory Block (Mode $7C)

This mode is used to report module memory data from an ECU to a tester. It contains the block number and data requested with a corresponding mode $3C.

4.2.33 Tester Present Diagnostic Message (Mode $3F)

This mode is used to indicate to an ECU that a tester is still connected to the vehicle and that certain diagnostic services and/or communication that have been previously activated are to remain active (e.g., current diagnostic state). The tester present message is intended to place minimum processing burden on the ECU. The control module is not required to take any action after receiving a tester present message other than to acknowledge that a diagnostic message has been received and to reset it's diagnostic session timer.

4.2.34 No Stored Codes Logging State Entry Request (Mode $B0)

Instructs an ECU to enter the no stored codes logging state, if supported, to suspend the logging of all DTCs. It is normally used to inhibit the incorrect logging of DTCs that may occur from another module performing self-test and not taking care of normal operations. See Section 7 for more detail.

4.2.35 Diagnostic Command (Mode $B1)

Instructs an ECU to perform an action that doesn't fit the scope of other defined diagnostic modes of operation. It is intended to be used for functions that are more specific in nature that don't warrant an entirely new mode definition. Historically, mode B1 was used for output actuation that worked in conjunction with a PID. Now, mode $2F is defined to perform output control and uses PID parameters as the target for output control requests. See Section 15 for more detail.

4.2.36 Input Integrity Test State Entry Request (Mode $B2)

Instructs an ECU to enter the input integrity test state from operational state.

The input integrity test state is intended to be used to perform output control and modification of input registers while the module is performing normal operational strategies. The items in this state shall be restricted to those that cannot compromise customer or user safety. All items associated with this state shall be fully documented in the ECU's Subsystem Specific Diagnostic Specification.

4.2.37 Request Manufacturer State Entry (Mode $B4)

This mode instructs an ECU to enter the manufacturer state (see section 13). The control module shall respond to the tester request with a general response message containing the affirmative response code. If the control module is unable to comply with the tester request the control module shall respond with a general response message containing the appropriate negative response code.

4.3 Diagnostic Response Messages

Diagnostic response messages can be either general response messages or messages with a mode number that is the request mode number with $40 added to it (data response message). The data response message is used when the response message has data associated with it to fulfill a tester diagnostic message request. For example, a request to report stored codes (mode $13) does return data (DTCs), so the data response message (request mode + $40, or $53 for this example) is used. The general response message is used for all positive responses that indicate the status of the request message without returning data and for any negative response. See "Appendix A" for a detailed list of all valid responses for a given diagnostic mode.

5 General Requirements

5.1 Purpose / Scope

This section details general requirements that apply to all control modules. All sections are considered required specifications unless stated otherwise

5.2 Diagnostic Message Definition

All diagnostic messages, request and response, will have up to seven data bytes available for use on SCP, Ford-9141 and UBP protocols.

Data Byte Number	SCP, Ford-9141 and UBP Description
1	Diagnostic mode byte
2	Optional diagnostic data byte 1
3	Optional diagnostic data byte 2
4	Optional diagnostic data byte 3
5	Optional diagnostic data byte 4
6	Optional diagnostic data byte 5
7	Optional diagnostic data byte 6

Table 5.1 Message Data Byte Descriptions

5.3 Diagnostic Data Byte Format

Each data byte in a diagnostic message shall have the bit format as shown in Table 5.2.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Most significant bit							Least significant bit

Table 5.2 Diagnostic Data Byte Format

Each diagnostic message shall have the byte layout as specified in Table 5.3.

Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	...	Data Byte N

Table 5.3 Diagnostic Message Byte Layout

5.4 ECU Power-up Requirements

An ECU shall respond to tester request messages within 1.0 second of module power-up.

NOTE: Test tools need to accommodate the potential inability of an ECU not able to communicate within the 1 second power-up period.

5.5 Communication Protocol Information

5.5.1 Master/Slave Configuration

Diagnostic communication for all of the Ford supported protocols are defined to use a master/slave configuration. Interrogating devices (testers) that communicate with an ECU using a diagnostic protocol will be considered the master on the network. As the master, the tester initiates all diagnostic communication on the link and each ECU is considered the slave that responds to the tester. An ECU shall never initiate diagnostic dialog between itself and a tester. In certain cases, an ECU may send multiple messages in response to a tester request, see "Appendix A" for a listing of the acceptable response for each protocol.

NOTE: Only one tester shall be active on the network at a time.

5.5.2 Rapid-Data Diagnostic Message Priority

The following applies only to modules while in the operational state.

Rapid-data or continuous response diagnostic messages shall not interfere with inter-module functional messages. For example, if a module is set up to periodically transmit a PID at 25 ms and the module needs to perform a functional strategy, it may temporarily suspend the transmission of the PID if transmission of the PID would interfere with the module's functional performance. A tester shall not interpret the extra delay for the module to transmit the parameter as a module error.

5.5.3 Unsupported Message Strategy

Any valid diagnostic message (correctly formatted with a valid checksum and targeted to an ECU) shall be responded to by the receiving ECU. A General Response (mode $7F) of mode not supported (code $11) shall be used as the response message for any received mode that isn't specified in the ECUs Subsystem Specific Diagnostic Specification.

5.5.4 SCP, UBP And ISO-9141-Ford Target/Source Address Assignments

Module and tester physical address assignments reflected in the target (byte 2) and source (byte 3) bytes of all request / response messages shall adhere to the ranges as outlined in the *Class B Data Communication Network Messages*[20] specification. EESE Core Network Communications allocates and maintains a list of physical addresses for module use. Also, the standard set of module acronyms in the MRDB shall be followed.

NOTE: ECU shall respond to any address (i.e., no need to filter data to determine if address is within range).

5.6 Tester Re-transmission of Diagnostic Messages

Testers may attempt a re-transmission of a diagnostic message if there was no response from an ECU and the message response timer expires. See Sections 17, 18 and 19 for information relating to response timeout pertaining to each supported diagnostic protocol.

5.7 MRDB Implementation Requirements

All items defined in the Master Reference Database MRDB shall always be transmitted on the bus with the same size and format as defined in the MRDB. If only a portion of a parameter is utilized within an ECU (e.g., only the first 2 bytes are used for a parameter defined to be 3 bytes in the MRDB), then the ECU shall pad any of the unused bytes and bits with zero.

5.8 Diagnostic Session Requirements

A diagnostic session is defined as any diagnostic state other than operational state. Diagnostic sessions encompass all states other than operational state as defined in Figure 4.1.

5.8.1 Normal Mode Messages

A control module may either support or not support normal module communication during a diagnostic session. A control module that receives normal mode communications has to receive all of its normal mode messages while in a diagnostic session, but has the option to discard any received message. Furthermore, a control module shall transmit all or none of its normal mode messages while in a diagnostic session.

NOTE: Normal mode communications are always supported (active) in operational state.

5.8.2 Continuous Code Management in the Diagnostic State

Continuous DTCs shall not be logged by an ECU while it is executing in any diagnostic state besides operational state and input integrity test state. The only exception is in regards to DTC $A477 (see the *Module Programming and Configuration Specification*[8] for more information). However, continuous DTCs previously logged shall be maintained (i.e., not overwritten) by the ECU while it is executing in a diagnostic session, unless otherwise commanded by the tester.

5.8.3 I/O Relating to Diagnostic Sessions (Excluding IIT and NSCL States)

Upon entering a diagnostic session, all ECU controlled outputs shall be made inactive, unless specified otherwise in the ECUs Subsystem Specific Diagnostic Specification. Also all ECU inputs shall be isolated from directly controlling related ECU outputs (e.g., switch input will not activate output).

Upon returning to the operational state from a diagnostic session, an ECU shall return all I/O to the state that they were in before entry into the diagnostic session and resume normal operation.

NOTE: The restraints module shall be exempt from the I/O requirements relating to diagnostic sessions as specified in section 5.8.3.

5.8.4 Return to Operational State Timing

An ECU shall return to the operational state and resume normal operation within 750 milliseconds from the time it initiates a return to operational state. During the 750 millisecond re-initialization period, the ECU is not required to respond to any diagnostic messages.

All modules are to respond affirmatively to the Request Return to Operational State Entry message and exit to the operational state as soon as possible. If the module is performing diagnostic operations, it is to suspend them as quickly as possible and comply with the request. For example, if a module is performing flash re-programming and a request is received to exit to operational state, then the module may need to delay by completing the current memory transfer and clearing its receive buffer before complying with the request.

5.8.5 On-demand DTC Clearing

Upon returning to the operational state, all on-demand DTCs logged during execution of a diagnostic test shall be cleared.

5.8.6 Diagnostic Session Timer

A 5.0-second (-0.0, +1.0) diagnostic session timer shall be supported and reset each time the control module successfully receives or transmits ANY diagnostic message (node-to-node or functional) in every

state other than operational state. The control module shall drop out of the state it may be in and return to the operational state if the timer expires. Upon entering each state, other than operational state, the timer shall be reset and begin running.

5.8.7 Vehicle Speed Timeout

If vehicle speed is available to an ECU, it may return to operational state when vehicle speed exceeds a value specified in the Subsystem Specific Diagnostic Specification, regardless of which diagnostic session is currently active. This shall only be utilized in situations where safety is the primary need for the timeout.

5.9 Safety and Damage Prevention Requirements

All diagnostic operations shall have safety and damage protection associated with them. Any operation that can cause a safety hazard or damage the ECU or any of the vehicle systems shall have protection associated with it.

For example, performing a service $2F to drive ABS coils too long can cause damage to the ABS system by burning the coils. For conditions such as this, the ABS module shall only allow the coils to be energized for a finite amount of time to avoid any damage to the ABS subsystem.

5.10 OBD Compliance

Refer to www.pcse.poee.ford.com/ds/ for information relating to OBD compliance. Note that failure of the serial communications link could result in illumination of the “Check Engine” MIL.

6 Operational State Diagnostics

6.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements that all serial communications link control modules shall support while in the operational state.

6.2 Operational State Modes

While in the operational state, an ECU shall support all of the diagnostic-supported modes shown in Table 6.1.

Mode #	*Description*
$10	Diagnostic state entry request
$20	Operational state entry request
$13	Request stored codes
$53	Report stored codes
$14	Request clear stored codes
$22	Request parameter by PID
$62	Report parameter by PID

Table 6.1 Mandatory Diagnostic Modes for Operational State

While in the operational state, an ECU may support any of the diagnostic modes shown in Table 6.2.

Mode #	*Description*
$12	Request Freeze Frame Data
$52	Report Freeze Frame Data. Required if mode $12 is supported
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$2F	Input/Output Control by PID
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop Transmitting Requested Data
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2C	Dynamically define diagnostic data packet.
$3F	Tester present
$B0	Request No Stored Codes Logging state
$B2	Request Input Integrity Test state
$B4	Request manufacturer state

Table 6.2 Optional Diagnostic Modes for Operational State

6.3 DTC Requirements

This section defines the requirements associated with Diagnostic Trouble Codes (DTCs).

Each DTC is a 2-byte hexadecimal number that is used to report an ECU failure. The assignment of a DTC shall attempt to isolate an ECU failure to a single field replaceable unit (i.e., air compressor, shock, switch, etc.).

NOTE: The definitions of all DTCs must be approved by the R&VT EESE Network Communication Application Engineer. All approved DTCs and their descriptions are maintained in the Diagnostic Master Reference Database - MRDB.

6.3.1 DTC Classifications

There are two classifications of DTCs that an ECU shall support:

- Continuous DTCs
- On-demand DTCs, see Section 10.4 for more information regarding on-demand DTCs.

6.3.2 Continuous DTCs

Continuous DTCs are diagnostic trouble codes logged by an ECU upon the detection of faults (e.g., I/O or ROM memory faults) while functioning in the operational state. All continuous DTCs shall be documented in an appropriate Subsystem Specific Diagnostic Specification.

NOTE: These strategies don't apply to Government Legislative requirements.

If any fault telltale is turned on by an ECU, it is mandatory for the ECU to log a continuous DTC relating to the cause of the fault telltale illumination.

To prevent the logging of "false" DTCs, it is recommended that an ECU suspend DTC logging when it's supply voltage is not within the range of 10 volts to 16 volts. In addition, it is recommended that an ECU log DTC $9317 when it's supply voltage exceeds 16 volts and log DTC $9318 when it's supply voltage drops beneath 10 volts.

6.3.2.1 Continuous DTC Clearing Counters

When an ECU detects a fault and logs a continuous DTC, it shall clear the DTC's "clearing counter". Clearing counters are used to track the number of error free ignition cycles from when the ECU last logged a particular continuous DTC. Each time the vehicle's ignition cycles, the clearing counters for all logged continuous DTCs shall be incremented only if their faults aren't present anymore. Whenever an ECU detects a fault, the clearing counter for that DTC shall be reset to zero.

6.3.2.2 Continuous DTC Memory Requirements

An ECU shall be capable of simultaneously logging all continuous DTCs and their associated clearing counters when the total number of DTCs supported by an ECU is less than 15. An ECU shall be capable of simultaneously logging a minimum of 15 continuous DTCs and their associated clearing counters when the total number of DTCs supported by an ECU is 15 or more.

6.3.2.2.1 Guidelines for Continuous DTC Updating of All DTCs

Each continuous DTC should be defined in ROM, the aging counter should be stored in Non Volatile Memory (NVM - e.g., EEPROM), and the DTC logged indicator (1 bit / DTC) should be stored in NVM.

Each DTC should be mapped to a 2-bit counter (modulo 3 DTC counter) and a 1-bit indicator in NVM. A modulo 32 counter (main counter) should be used to increment the individual 2-bit DTC counters. On each ignition cycle or key cycle, the main counter should increment. When the main counter rolls over from 31 to 0, the DTC counters should be incremented. When the DTC counter rolls over from 3 (11 binary) to 0 (zero), the DTC indicator bit in NVM should be cleared. Upon receipt of a clear codes request, all DTC counters and NVM indicators are to be cleared.

Upon detection of a new fault, the corresponding DTC will be logged as follows:

1. The DTC counter (2 bit counter in NVM) will be reset
2. The logged indicator bit in NVM will be set

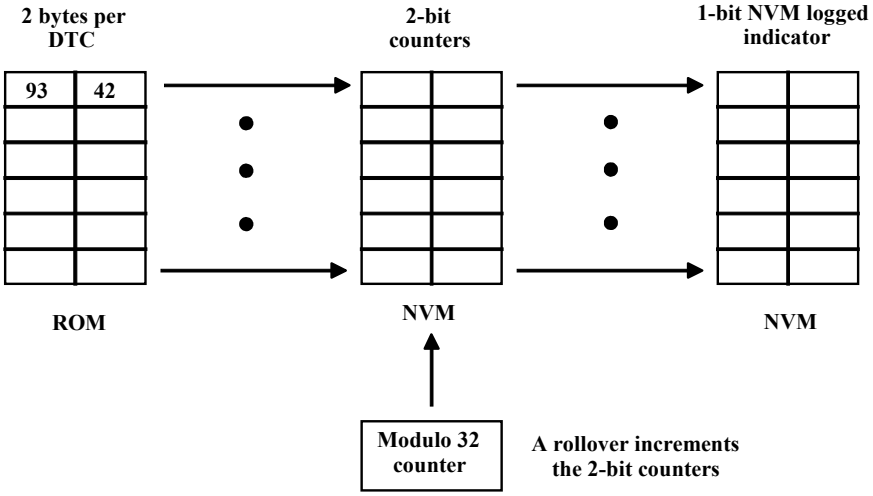


Figure 6.1 Example of Memory when Storing all Continuous DTCs

With the above method, the DTCs can be arranged in the ECU such that the DTC with the highest priority, or most critical, is read out and displayed in descending order. If all supported DTCs are not capable of being logged simultaneously in NVM, then there is no longer a one to one relationship between a given NVM bit and a given DTC. Therefore, more information on which DTC is logged needs to be stored in NVM as shown in the following figure.

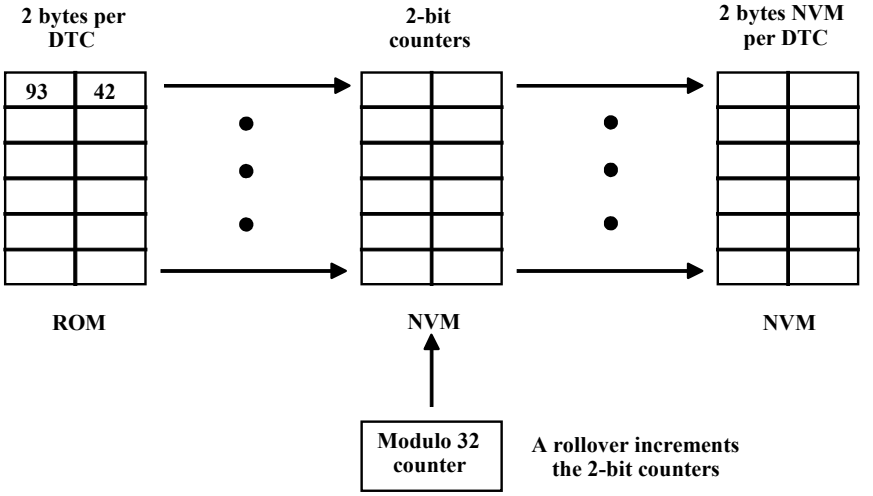


Figure 6.2 Example of Memory when Storing Only 15 of 24 Continuous DTCs

Resource Requirements	Store All	Store only 15
NVM (bytes for 2-bit counters)	6 Bytes (24*2 bits)	4 Bytes (15*2 bits)
NVM (bytes for DTC storage)	3 Bytes (24*1 bits)	30 Bytes (15*16 bits)
Search Routine	None	x bytes of RAM & ROM

Table 6.3 Example of Memory Usage When Storing All Continuous DTCs Vs Storing Only 15 Continuous DTCs For an ECU With 24 Continuous DTCs

6.3.2.3 Reporting Continuous DTCs (Ford-9141, SCP and UBP)

In response to a Request Parameter by PID (mode $22) message, with the address set to $0200 - Number of Continuous DTCs**,** an ECU shall return a Report Parameter by PID (mode $62) message with the number of continuous DTCs currently logged. In response to a single Request Stored Codes (mode $13) message, an ECU shall report all continuous DTCs logged by returning as many consecutive Report Stored Codes (mode $53) messages as is required. Refer to Table 6.4 for an example of the number of consecutive messages returned in relation to the number of continuous codes in a module.

NOTE: There is no provision for a tester to request, or for an ECU to report, less than all of the continuous DTCs logged by a module.

Continuous DTCs Logged	Messages Returned	Method
0	1	All six Bytes of the message reserved for three DTCs shall be padded with $00
1 or 2	1	DTC(s) in Data Bytes 2 & 3 and/or Data Bytes 4 & 5; the remaining Bytes shall be padded with $00
3	1	All three DTCs shall be reported in Data Bytes 2 to 7 of the message
4 or 5	2	The first message shall return three DTCs. The second message shall follow the method used for one or two DTCs as explained above
6	2	Each message shall return three DTCs
n	m	Follow the same method presented above for four to six DTCs

Table 6.4 Consecutive Messages Returned When Reporting DTCs

6.3.2.4 Clearing Continuous DTCs

Each ECU shall support the capability to clear continuous DTCs via two methods:

- Manually
- Automatically

6.3.2.4.1 Manual Clearing of Continuous DTCs

An ECU shall clear all of its logged continuous DTCs upon receiving a Request Clear Stored Codes *(mode $14)* message. DTC $A477 has unique clearing requirements (refer to the *Module Programming and Configuration Specification*[8] for more information on clearing requirements for DTC $A477). Once the DTCs have been cleared, the ECU shall respond to the request with an affirmative general response (mode $7F with response code $00)**.**

NOTE: Based upon FMEA determined criticality, it is allowable for ECUs with safety critical enhancements to not clear the associated safety critical DTCs in operational state only. An affirmative general response shall still be transmitted and all non-safety critical DTCs cleared. This approach should be used only when absolutely necessary, and all safety critical DTCs that are not cleared in operational state shall be clearly documented in the ECU's Subsystem Specific Diagnostic Specification.

6.3.2.4.2 Selective Clearing of DTCs

ECUs shall support selective clearing of continuous DTCs only on Jaguar control modules.

6.3.2.4.3 Automatic Clearing of Continuous DTCs

An ECU shall automatically clear a continuous DTC when its corresponding clearing counter reaches between 80 and 128 counts, refer to Section 6.3.2.1 for information on clearing counters. DTC $A477 shall not be automatically cleared, refer to the *Module Programming and Configuration Specification*[8] for more information on clearing requirements for DTC $A477. The exact value of the clearing counter required before a continuous DTC is cleared shall be specified in the ECUs Subsystem Specific Diagnostic Specification.

NOTE: The restraints module shall have the option of using time based clearing of DTCs versus the clearing counter. The exact method including time values, if applicable, for each DTC shall be specified in the ECUs Subsystem Specific Diagnostic Specification.

6.3.2.5 Freeze Frame Associated With Continuous DTCs - Optional

This is a strategy that ECUs may employ to gain additional information related to a DTC. Freeze frame information can define information relating to a DTC such as the value of certain module parameters at the time the DTC was logged or other running counter information such as the number of times the fault has appeared and disappeared.

"Appendix A" lists the structure of this operation within mode $12. A one byte frame number is listed in the mode $12 message. Multiple frames can be defined for each module, with each frame associated with a different set of data parameters. All information associated with each frame defined for each ECU shall be fully documented in its corresponding Subsystem Specific Diagnostic Specification. The format of any implemented freeze frame number shall be the same for all DTCs on an ECU that support this freeze frame number. For example, DTC $5213 may support two freeze frames $00 and $01 (where freeze frame number $00 contains engine speed and freeze frame number $01 contains engine coolant temperature). If another DTC (e.g., $5214) needs to support the storage of engine speed and intake air temperature, it may support only two freeze frames $00 and $02 (where freeze frame number $00 contains engine speed and freeze frame number $02 contains intake air temperature). A request to read freeze frame number $01 for DTC $5214 shall return a general response of sub-function not supported ($7F-$12) since storing engine coolant temperature is not supported for this DTC. Once a freeze frame number is assigned in a given ECU, its format shall be the same for any DTC on that ECU that chooses to implement that freeze frame number (i.e., if freeze frame number $00 contains engine coolant temperature for one DTC on an ECU, it must contain engine coolant temperature for any DTC on the ECU supporting freeze frame number $00.)

The response message associated with the mode $12 request for freeze frame data is mode $52 to report the freeze frame and is listed in "Appendix A". The mode $52 message echoes the freeze frame and DTC numbers respectively and then has up to 3 data bytes of freeze frame data.

6.3.2.6 Mandatory Continuous DTCs

Table 6.5 defines the mandatory DTCs that shall be supported by all ECUs which communicate via the DLC.

DTC	Description
$9342	ECU is Faulted
$A477	Module Configuration Failure (If configuration is supported by module).

Table 6.5 Mandatory DTCs.

6.3.2.6.1 ECU is Faulted ($9342)

DTC **$9342** shall be logged by an ECU when it detects a failure with its internal circuitry while in the operational state or while executing a self-test in execution routine state.

6.3.2.6.2 Module Configuration Failure

DTC **$A477** shall be logged by an ECU per the Module Programming and Configuration Specification[8].

6.4 Network Communication Diagnostic Fault Strategy

This section defines the diagnostic strategies that an ECU shall implement to handle fault conditions relating to inter-module communication.

6.4.1 Network Fault Types

Network faults can be categorized into two types, as follows:

- Invalid data network faults
- Missing message network faults

6.4.2 Network Communication DTCs

The SAE standard *Diagnostic Trouble Code Definitions*[4] classifies and partitions DTCs into various categories. the range from 1100-1111 lists the range of DTCs that are assigned to faults which may occur during inter-module network communication.

6.4.2.1 SCP Invalid Data DTCs

The procedure to determine the DTC that shall be logged by a module when a SCP network message containing invalid data is received:

1. Convert the hexadecimal primary ID of the message received to a BCD number (hex to decimal conversion).
2. Add this BCD number to the base DTC number, D000, to determine the invalid data SCP DTC number.

6.4.2.1.1 Example:

ECU (A) transmits a status request message to ECU (B) and ECU (B) responds with a message that contains “Invalid data”.

The primary ID is of the desired information is $73.

What DTC should be logged by the ECU (A)?

1. Convert $73 to BCD : (7 x 16) + (3 x 1) = 115

2. Add 115 to the Base DTC : D000 + 115 = D115

6.4.2.2 UBP Invalid Data DTCs

The procedure to determine the DTC that shall be logged by a module when a UBP network message containing invalid data is received:

1. Convert the hexadecimal node ID of the module that transmitted the invalid data to a BCD number (hex to decimal conversion).
2. Add this BCD number to the base DTC Number, E210, to determine the invalid data UBP DTC number.

6.4.2.2.1 Example:

ECU transmits a status request message to ECU (B) and ECU (B) responds with a message that contains ”Invalid data”.

The node ID is of the desired information is $26.

What DTC should be logged by the ECU?

1. Convert $26 to BCD : (2 x 16) + (6 x 1) = 38
2. Add 38 to the Base DTC : E210 + 38 = E248

6.4.2.3 Invalid Data Network Faults

Data transferred within normal inter-module messages can be incorrect or invalid at the source of the data. When the source data is known to be incorrect, for whatever reason, then invalid data identifiers shall be placed in the data field. Invalid data network faults are mainly caused by incorrect input data.

6.4.2.3.1 Digital Invalid Data Network Fault Strategy (All protocols)

Any module transmitting known invalid data to another module shall log a DTC relating to the invalid data fault. Modules transmitting periodic or response messages to other modules shall always send their messages when the message is expected by another on the network, even if the data may be invalid; this, at a minimum, lets the receiver know that communications are still operational.

6.4.2.3.2 Digital Invalid Data Network Fault Strategy (SCP and UBP)

If a network message’s valid functional data is normally digital information, such as enabled/disabled or status, the transmitting ECU shall place ”Invalid data” information into the network message according to the following procedure.

6.4.2.3.2.1 Procedure for SCP:

1. Shift all data that is located to the right of the message’s source address (byte 3) one byte position to the right
2. Place the digital “Invalid data” value - $3F into the message to the immediate right of the source address; this is the location previously held by the message’s secondary ID.

6.4.2.3.2.2 Procedure for UBP:

1. Shift all data that is located to the right of the message’s operation byte (byte 4) one byte position to the right
2. Place the digital “Invalid data” value - $3F into the message to the immediate right of the operation byte; this is the location previously held by the message’s secondary ID.

The following example provides a visual indication of how a SCP network message is modified to provide digital ”Invalid data” information.

6.4.2.3.2.3 Digital Invalid Data Example on SCP:

For Brake Lamp Pedal Switch Status - Active

Normal Message: 41 33 70 A2

“Invalid data” Message: 41 33 70 **3F** A2

6.4.2.3.3 Analog Invalid Data Network Fault Strategy (SCP and UBP)

If an ECU that is responsible for providing analog data in a network message, detects a fault that would render all or a portion of that analog data invalid, it shall replace each affected byte of normally valid information with ”Invalid data” value of $FF. The following example provides a visual indication of how a SCP network message is modified to provide analog “Invalid data” information. Other than modifying the required data bytes, the structure of the message that gets transmitted is the same as that sent with ”Valid data”. A network message’s valid functional data may consist of one to six bytes of analog information (such as a temperature or voltage value).

6.4.2.3.3.1 *Analog Invalid Data Examples on SCP:*

For Engine Coolant Fluid Temperature Status;

Normal Message: 81 49 10 10 xx

“Invalid data” Message: 81 49 10 10 FF

NOTE: ”FF” is defined as the ”Invalid data” value

6.4.2.3.3.2 For Actual Axle Torque w/min. & max. Available:

Normal Message: A1 09 10 22 xx xx yy yy zz zz

“Invalid data” Message: A1 09 10 22 FF FF yy yy zz zz (FF is the invalid data)

6.4.2.3.4 Combined Digital and Analog Invalid Data Network Fault Strategy

If an ECU that is responsible for providing digital and analog data in a network message, detects a fault that would render all or a portion of that data invalid, it shall follow the requirements defined above for digital invalid data and analog invalid data. The following example provides a visual indication of how a SCP network message is modified to provide digital and analog ”Invalid data” information.

6.4.2.3.4.1 Digital and Analog Invalid Data Examples:

For Backlighting Intensity & Dimming Curve w/ Headlamps Command - On;

Normal Message: 41 DA 60 12 xx xx

“Invalid data” Message: 41 DA 60 **3F** 12 xx xx (Digital data)

OR

Normal Message: 41 DA 60 12 xx xx

“Invalid data” Message: 41 DA 60 12 **FF** xx (Analog data)

6.4.2.4 Missing Message Network Faults (Non ISO-9141-FORD)

A missing message network fault may be logged by an ECU upon failure to receive a message expected from another ECU within a defined retry period. There is only one continuous DTC per protocol (See

Table 6.6) that is allocated for an ECU to identify all possible missing message network faults. The missing message network continuous DTCs are optional and shall NOT be supported by an ECU unless the missing message is deemed a critical failure to the function of the ECU. Implementation of missing message network continuous DTCs shall be defined in the appropriate Subsystem Specific Diagnostic Specification for the module logging the code. Some causes of missing message network faults include the following:

- ECU transmitter failures
- ECU receiver failures
- Transmission wire opens or shorts
- Initialization timing differences
- Power up and/or power down timing differences
- ECU placed into diagnostic state and other modules not placed in the no stored codes logging state
- ECU disconnected by service technician

Protocol	Missing Message DTC
SCP	$D262
UBP	$D950

Table 6.6 Missing Message DTCs

6.4.2.4.1 Missing Message Fault Code Logging

A missing message DTC may be logged if either of the following conditions occur:

1. The message is missed for at least 5 periods in which the message was supposed to be received.
2. The message is missed consecutively for 5 seconds or after a retry strategy has expired.

6.4.2.4.2 Optional Missing Message Fault PID Pointer Strategy (UBP and SCP Only)

Missing message fault PIDs shall define the details relating to a missing message and shall be defined in the MRDB. Missing message fault PIDs shall only have fault bits set when the missing message DTC that the fault PID is associated with is logged. For more information regarding fault PIDs refer to section 16.9.

6.4.2.4.3 Missing Message Fault PID Strategy for SCP

SCP missing message fault PID bits shall refer to the primary ID of the missing message. Only the primary ID is listed due to the application software in the module having access to this information in the header and not the source address. The fault PIDs associated with the SCP missing message DTC $D262 are $B911 - $B918.

6.4.2.4.4 Missing Message Fault PID Strategy for UBP

UBP missing message fault PID bits shall refer to the node ID of the source module from which the message is missing; this is the ideal strategy for missing message network faults in that it directly identifies the source of the missing message. The fault PID associated with the UBP missing message DTC $D950 is $B903.

6.4.2.5 Test Tool Missing Message Network Test

To determine the location, or path, of an inter-module ”missing message” communication fault, all FORD test tools shall implement a network test in which the test tool attempts to establish communication with each of the ECUs on the link. If, during this network test, the test tool fails to establish communication with one or more ECU, it shall display information that indicates which ECU could not establish communication.

7 No Stored Codes Logging (NSCL) State Requirements

7.1 Purpose / Scope

All ECUs on SCP and UBP shall implement the No Stored Codes Logging (NSCL) State. NSCL prevents any DTC from being logged by an ECU and is especially useful when preventing erroneous network DTCs from being logged by ECUs that are involved in inter-module communication with other modules.

7.2 NSCL State Diagnostic Requirements

While in the no stored codes logging state, an ECU shall support all of the diagnostic supported modes shown in Table 7.1.

Mode #	*Description*
$10	Diagnostic state entry request
$20	Operational state entry request
$13	Request stored codes
$53	Report stored codes
$14	Request clear stored codes
$22	Request parameter by PID
$62	Report parameter by PID
$B0	Request No Stored Codes Logging state

Table 7.1 Mandatory Diagnostic Modes for the No Stored Codes Logging State

While in the no stored codes logging state, an ECU may support the diagnostic modes shown in Table 7.2.

Mode #	*Description*
$12	Request Freeze Frame Data
$52	Report Freeze Frame Data. Required if mode $12 is supported
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop transmitting requested data
$27	Request security access
$67	Report security access. Required if mode $27 is supported
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2C	Dynamically define diagnostic data packet
$2F	Input/output control by PID
$28	Disable normal message transmission (except ISO-9141-FORD)
$3F	Tester present

Table 7.2 Optional Diagnostic Modes for No Stored Codes Logging State

7.2.1 Entering the No Stored Codes Logging State

An ECU in the operational state shall enter the no stored codes logging state upon receiving a Request No Stored Codes Logging message. See "Appendix A" for information regarding the defined response messages for each state of operation with relation to the Request No Stored Codes Logging message.

7.2.2 NSCL State to Operational State Timing

Upon returning to the operational state, from the NSCL state, an ECU shall return to normal operation (i.e., continue logging continuous DTCs) within 750 milliseconds.

7.2.3 Exiting the No Stored Codes Logging State to the Operational State

An ECU shall exit the no stored codes logging state and enter the operational state if any of the following conditions occur:

- The ECU receives a Request Operational State Entry message
- The diagnostic session timer expires

7.2.4 Normal Message Transmission Disabling

Mode $28 (disable normal message transmission) is only allowable in the no stored codes logging state. A typical use would be to reduce the bus traffic during a download or upload to another ECU. The disabling of normal messages due to a mode $28 shall only remain active while no stored codes logging state is active. If no stored codes logging state is exited for any reason, the mode $28 functionality shall cease to be active, and the transmission of messages will default to the implementation of the state that was transitioned to.

7.2.5 NSCL State Tester Requirements

Testers shall adhere to the following requirements relating to diagnostic sessions and the NSCL:

- Prior to placing a module that performs inter-module communication in the diagnostic state, a tester shall place all of the other modules into the NSCL state that communicate on any of the protocols supported by the ECU being placed into the diagnostic state.
- Testers shall also keep all of the modules in NSCL for one second after the original module placed in diagnostic state terminates its diagnostic session and returns to the operational state.
- Testers shall command all applicable modules into the NSCL state at least 400 milliseconds before entering a diagnostic session.
- Modules that transmit messages to other modules may suspend periodic message transmission when placed in the diagnostic state. Suspended messages can cause the receivers of those messages to log DTCs. Therefore, when a module that is responsible for transmitting inter-module messages is placed in diagnostic state, then all of the other modules that are involved in inter-module communication, on the same protocol, need to be placed in NSCL to avoid logging erroneous network DTCs.

8 Diagnostic State Requirements

8.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements that all bi-directional serial communications link control modules shall support while in the diagnostic state.

8.2 Entering the Diagnostic State

To enter diagnostic state from operational, input integrity or no stored codes logging state, an ECU shall meet all entry conditions specified in its Subsystem Specific Diagnostic Specification. If any entry condition isn't met, the ECU shall not enter the diagnostic state and shall return a General Response message with the Response Code set to Conditions Not Correct - $22.

8.2.1 Entering Secure Diagnostic State From Diagnostic State

Secure diagnostic state is an optional state. The control module's Subsystem Specific Diagnostic Specification shall list if secure diagnostic state is supported. See Section 9 for more information regarding security access.

8.3 Diagnostic State Modes

While in the diagnostic state, an ECU shall support all of the mandatory supported diagnostic modes shown in Table 8.1. It is up to the discretion of the module designer to determine if the ECU will send and/or receive normal mode (non-diagnostic) messages during the diagnostic state.

Mode #	***Description***
$10	Diagnostic state entry request
$20	Operational state entry request
$13	Request stored codes
$53	Report stored codes
$14	Request clear stored codes
$22	Request parameter by PID
$62	Report parameter by PID
$31	Request diagnostic routine entry. Execution routine $02 shall be supported to accommodate self-test.
$33	Request diagnostic routine results
$73	Report diagnostic routine result

Table 8.1 Mandatory Diagnostic Modes for Diagnostic State

Other optional diagnostic modes may be supported in the diagnostic state as shown in Table 8.2.

Mode #	***Description***
$12	Request Freeze Frame Data
$52	Report Freeze Frame Data. Required if mode $12 is supported
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported

$27	Request security access
$67	Report security access. Required if mode $27 is supported
$25	Stop transmitting requested data
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2F	Input/output control by PID
$34	Request download routine entry
$35	Request upload routine entry
$3B	Request write memory block
$3C	Request read memory block
$7C	Report contents of memory block
$B1	Diagnostic command
$B2	Request Input Integrity Test state
$B4	Request manufacturer state
$3F	Tester present
$2C	Dynamically define diagnostic data packet.

Table 8.2 Optional Diagnostic Modes for Diagnostic State

9 Secure Diagnostic State Requirements

9.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements that all bi-directional serial communications link control modules shall support while in the secure diagnostic state. This state is optional and is typically provided for sub-systems to avoid intentional modification of ECU data that could cause harm to the occupants of the vehicle, damage the ECU or to protect other proprietary interests.

9.2 Entering the Secure Diagnostic State

The security access procedure shown in Figure 9.1 shall be used to enter the secure diagnostic from the diagnostic state or the secure diagnostic state if multiple security levels exist. Once an ECU is "unlocked" it shall remain that way until a transition to operational state occurs (either with a valid mode $20 request or a diagnostic session timeout). A mode $10 request from any diagnostic state (with the exception of operational state) shall place the ECU into secure diagnostic versus diagnostic state if the ECU is currently "unlocked".

9.3 Secure Diagnostic State Modes

While in the secure diagnostic state, an ECU shall support all of the mandatory supported diagnostic modes shown in Table 9.1. It is up to the discretion of the module designer to determine if the ECU will send and/or receive normal mode (non-diagnostic) messages during the secure diagnostic state.

Mode #	***Description***
$10	Diagnostic state entry request
$20	Operational state entry request
$13	Request stored codes
$53	Report stored codes
$14	Request clear stored codes
$22	Request parameter by PID
$62	Report parameter by PID
$27	Request security access
$67	Report security access.
$31	Request diagnostic routine entry. Execution routine $02 shall be supported to accommodate self-test.
$33	Request diagnostic routine results
$73	Report diagnostic routine result

Table 9.1 Mandatory Diagnostic Modes for Secure Diagnostic State

Other optional diagnostic modes may be supported in the secure diagnostic state as shown in Table 9.2.

Mode #	***Description***
$12	Request Freeze Frame Data
$52	Report Freeze Frame Data. Required if mode $12 is supported
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$24	Request parameter scaling/masking

$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop transmitting requested data
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2F	Input/output control by PID
$34	Request download routine entry
$35	Request upload routine entry
$3B	Request write memory block
$3C	Request read memory block
$7C	Report contents of memory block
$3F	Tester present
$B1	Diagnostic command
$B2	Request Input Integrity Test state
$B4	Request manufacturer state
$2C	Dynamically define diagnostic data packet.

Table 9.2 Optional Diagnostic Modes for Secure Diagnostic State

9.4 Security Levels

Security levels supported by an ECU for seed requests shall only be odd numbers. In addition, security key submittal by the tester will always be the security level plus one, resulting in even numbers only for the key submittal. If an ECU is capable of being programmed or configured at end of line or in service, and security access is required for this functionality, then the recommended practice is for security level $01 to provide this level of security access that allows a tester the ability to program or configure the ECU to the extent needed at end of line or service.

9.4.1 Active Security Level

Only one security level shall be active at any instant of time. For example, if the active security level is $03, and a tester request is successful in transitioning the ECU to security level $01, then only the functionality supported by security level $01 shall be supported at that time. Any additional functionality that was supported in security level $03 shall no longer be active.

9.4.2 ECU Response to a Lack of Security Access

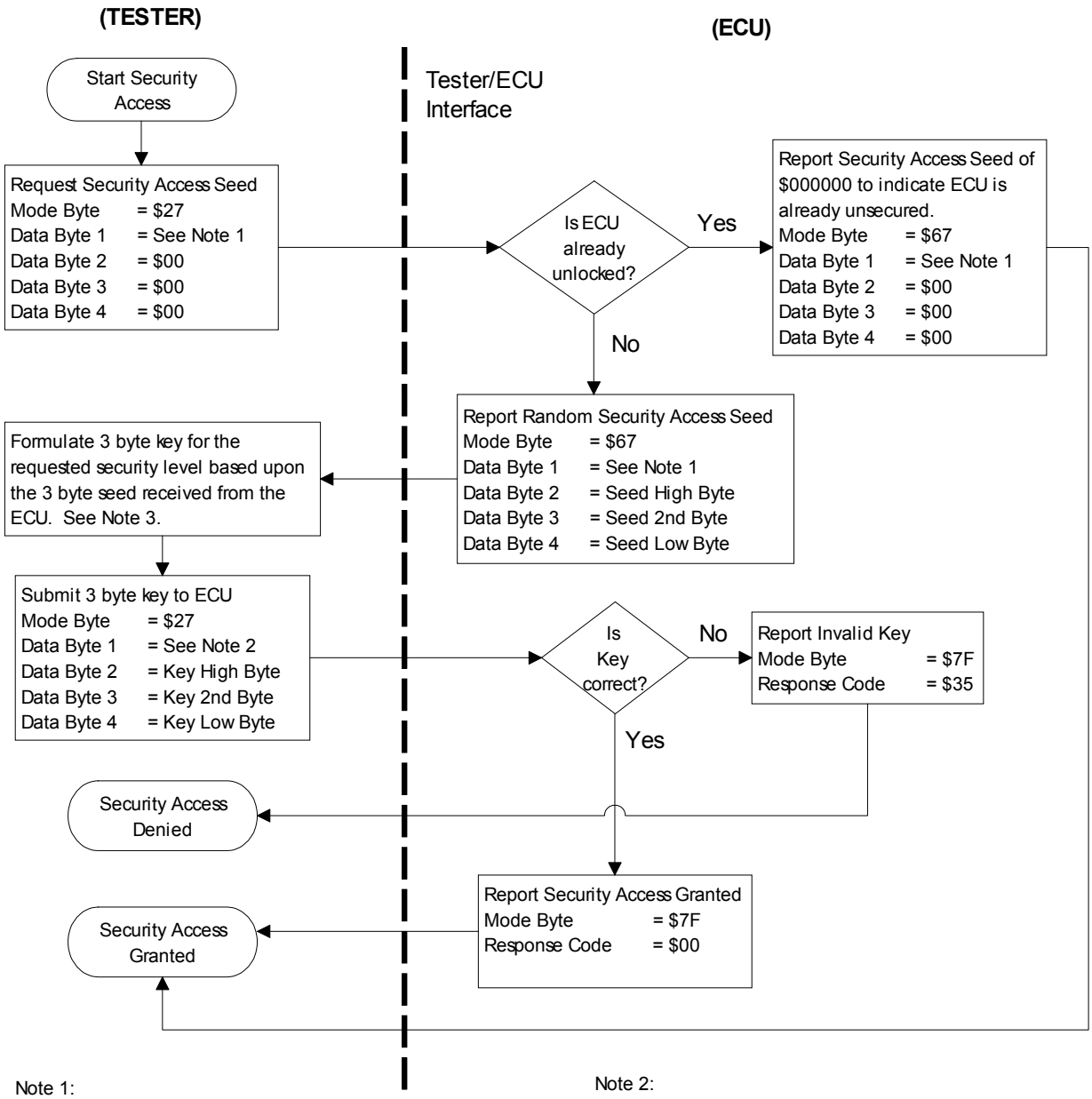
If any diagnostic mode or subset of a diagnostic mode needs to be in a certain level of secure diagnostic state to execute, and all other conditions are met besides the appropriate level of secure diagnostic state, then the ECU shall return a general response of mode $7F, code $33 (security access denied) to inform the tester of the need for a higher level of security access.

9.5 Seed and Key Procedure

The security access procedure shown in shall be used to "unlock" the ECU and grant security access. Upon receipt of a valid seed request, the ECU shall respond with a randomly chosen security seed. This seed shall remain valid or "active" in the ECU until one of the following conditions occur:

- A new valid seed request is received by the ECU (regardless of whether it is for the same or a different security level)
- A security key (correct or incorrect) is received by the ECU (regardless of whether it is for the same or a different security level)
- A transition to operational state is made (e.g., with a mode $20 or a diagnostic session timeout)

An ECU shall reject any security key sent by a test tool for a given security level if a valid or "active" seed does not exist for that security level, with a mode $7F and response code $22.



Note 1:
1st Level of Security: Data Byte 1 = $01
2nd Level of Security: Data Byte 1 = $03
X Level of Security: Data Byte 1 = $2X-1

Note 2:
1st Level of Security: Data Byte 1 = $02
2nd Level of Security: Data Byte 1 = $04
X Level of Security: Data Byte 1 = $2X

Note 3:
Security algorithm may be obtained upon request from Core Network Communications.
The module Part 2 specification shall define which security levels are supported as well as the constants used in the security algorithm for each security level supported. Constants used in the security algorithm shall all be non-zero values.

Figure 9.1 Security Access Procedure

10 Execution Routine State Requirements

10.1 Purpose / Scope

This section defines the requirements for the execution routine state and the execution routines (diagnostic tests) that shall reside within an ECU for the purpose of evaluating the ECU and its subsystem(s). It also describes the ability for an external tester to download an executable piece of code to the ECU and execute it; this functionality is primarily for manufacturer use (supplier and Ford).

10.2 Execution Routine State Entry Requirements

An ECU shall execute a diagnostic execution routine upon receiving a Request Diagnostic Routine Entry message from a tester if the following entry requirements are satisfied:

- The ECU is in the diagnostic state
- All entry requirements specified in the ECUs Subsystem Specific Diagnostic Specification have been satisfied
- ECU memory is allocated within the ECU for the storage of the diagnostic test results

While in the Execution Routine state, an ECU shall support all of the diagnostic supported modes shown in Table 10.1.

Mode #	***Description***
$20	Operational state entry request
$22	Request parameter by PID
$62	Report parameter by PID
$32	Request diagnostic routine exit

Table 10.1 Mandatory Diagnostic Modes for Execution Routine State

Other optional diagnostic modes may be supported in the Execution Routine state as shown in Table 10.2.

Mode #	***Description***
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop transmitting requested data
$27	Request security access
$67	Report security access. Required if mode $27 is supported
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2C	Dynamically define diagnostic data packet.
$3F	Tester present

Table 10.2 Optional Diagnostic Modes for Execution Routine State

10.3 Execution Routine State Exit Requirements

An ECU shall exit the execution routine state and return to either of the following states under the appropriate conditions:

- Diagnostic State
- Operational State

10.3.1 Exiting a Diagnostic Test to the Diagnostic State

An ECU shall exit an execution routine and return to the diagnostic state if any of the following occurs:

- The ECU receives a Request Diagnostic Routine Exit message over the serial communication link, where data byte 3 specifies the exit condition (***$00*** - exit if test is completed, ***$01*** - exit immediately).
    - Note: Upon completion of an execution routine (e.g., on-demand self-test), an ECU shall respond with a general responsive of affirmative ($7F-$00) to the first request diagnostic routine exit (mode $32, $00 - exit when complete) message received from a tester. For example, while a diagnostic test is running the test tool may continuously transmit request diagnostic routine exit (mode $32, $00 - exit when complete) message to determine when the test has completed. While the ECU is executing the routine, the module shall respond with general response of busy ($7F-$21) to the request diagnostic routine exit message. Once the ECU has completed executing the routine it shall respond to the next request diagnostic routine exit message from the tester with a general response of affirmative ($7F-$00).
- An exit condition, as specified in the ECU's Subsystem Specific Diagnostic Specification, that causes the ECU to return to the diagnostic state, is met. Example: If the ECU's Subsystem Specific Diagnostic Specification states that "all doors shall remain closed during test execution" and a door is opened while the test is executing, the module shall immediately exit from the test and return to the diagnostic state.

NOTE: A delay exiting execution routine state, to either the diagnostic or operational states, is permissible for tests involving safety related systems such as anti-lock braking systems and shall not exceed 500ms.

10.4 On-demand Diagnostic Trouble Codes

On-demand diagnostic trouble codes are fault codes temporarily logged (i.e., in RAM) by an ECU which identify failures detected during the execution of any diagnostic test (e.g., On-demand self-test) supported by the ECU.

The following on-demand DTC operations shall be supported:

- Logging on-demand DTCs that identify subsystem failures detected during the execution of a diagnostic test.
- Reporting on-demand DTCs over the serial communications link upon request.

10.4.1 Reporting On-demand DTCs

In response to a Request Parameter by PID message, with the PID number set to $0202 - Number of Trouble Codes Set Due to Diagnostic Test, an ECU shall return a Report Parameter by PID message with the number of on-demand DTCs generated during the most recent diagnostic test executed as the data. In response to a single Request Diagnostic Routine Results message, an ECU shall report all on-demand DTCs being logged by returning as many consecutive Report Diagnostic Routine Results messages as is required. Refer to Table 6.4 for an example of consecutive messages returned. Refer to "Appendix A" for

direction about responses to diagnostic routine result requests in the various operating states/modes. Note: There is no provision for a tester to request, or for an ECU to report, less than all of the on-demand DTCs being logged by the module.

10.4.2 Maintaining On-demand DTCs

An ECU shall maintain on-demand DTCs for the most recently run on-demand test for as long as the diagnostic session that the test was ran in is still active. An ECU shall not maintain the results for more than one diagnostic test at any given time. ECU memory allocated for the storage of diagnostic test results, shall be overwritten each time a diagnostic test is executed. The tester is responsible for obtaining the results to a diagnostic test prior to the execution of another test.

10.4.3 Clearing On-demand DTCs

On-demand DTCs shall be cleared when the ECU returns to the operational state or when the ECU executes another diagnostic test. An ECU shall NOT clear on-demand DTCs upon receiving a Request Clear Stored Codes message; this message is only used for clearing continuous DTCs. Note: There are no messages defined for clearing on-demand DTCs.

10.4.4 On-demand Fault PIDs

An on-demand fault PID always has an associated master on-demand DTC. An on-demand fault PID is only valid if a diagnostic execution routine that affects the setting of the on-demand fault PID bits has been executed within the last three key cycles (i.e., on-demand fault PID data is invalid following the fourth key cycle after the execution of the aforementioned routine). When the execution routine determines whether or not to set an on-demand DTC that has an associated on-demand fault PID, the corresponding fault PID(s) bits shall be set to '1' or '0' indicating a fault or no fault respectively for each bit. The fault PID bits shall remain locked in this state for three consecutive key cycles (assuming no additional diagnostic execution routines are performed which affect the values of the fault PID), after which the on-demand fault PID will become invalid. If any diagnostic execution routine is ran during this time that affects the logging of this same on-demand DTC and thus the on-demand fault PID(s), the fault PID shall be updated with new values corresponding to the results of the execution routine.

The on-demand fault PID information is read by transmitting a standard mode $22, Request Parameter by PID, message. If a requested on-demand fault PID is not valid (i.e., an execution routine that affects the values of the fault PID was not ran in the last three key cycles), the ECU shall respond with a mode $7F, response code $22 (conditions not correct) to a request to report the fault PID values.

NOTE: On-demand fault PIDs are always defined in separate PIDs than continuous fault PIDs.

Example of "sticky" behavior of on-demand fault PIDs. Assumption here is that the ECU is already in the diagnostic state and no prior execution routines have been executed in the past three key cycles.

Note: DTC $5724 has an associated on-demand fault PID $395D.

Message Transfer	***Event or Action***
Tester to ECU	Request PID $395D read (mode $22)
ECU to Tester	Negative response (mode $7F, response code $22 – conditions not correct) Note: No execution routine has been executed in the past three key cycles that would validate the data reported in the on-demand fault PID $395D.
Tester to ECU	Request begin execution routine $02 (mode $31)
ECU to Tester	Affirmative response
Tester to ECU	Request execution routine $02 exit (mode $32)

ECU to Tester	Affirmative response
Tester to ECU	Request execution routine $02 results (mode $33)
ECU to Tester	Affirmative response (mode $73) with on-demand DTC $5724
Tester to ECU	Request PID $395D read (mode $22) Note: PID $395D is the on-demand fault PID associated with DTC $5724
ECU to Tester	Affirmative response (mode $62) with PID data Note: At least one bit in the on-demand fault PID **must** be set due to DTC $5724
Tester to ECU	Request operational state entry (mode $10)
ECU to Tester	Affirmative response
Tester to ECU	Request execution routine $02 results (mode $33)
ECU to Tester	Negative response (mode $7F, response code $22 – conditions not correct) Note: On-demand DTC results are no longer available in operational state.
Tester to ECU	Request PID $395D read (mode $22)
ECU to Tester	Affirmative response (mode $62) with PID data Note: On-demand fault PID contains same data for three key cycles unless another execution routine is ran which effects the values.
	Key is cycled three times.
Tester to ECU	Request PID $395D read (mode $22)
ECU to Tester	Affirmative response (mode $62) with same PID data as earlier.
	Key is cycled one more time.
Tester to ECU	Request PID $395D read (mode $22)
ECU to Tester	Negative response (mode $7F, response code $22 – conditions not correct) Note: After fourth key cycle, the on-demand fault PID information is no longer valid, and an attempt to read the information results in a negative response.

Table 10.3 Example of "sticky" behavior of on-demand fault PIDs

10.5 Diagnostic Tests

Diagnostic tests are executable routines resident within an ECU or downloaded from a tester that may be invoked by a tester to evaluate an ECU and its associated components. The diagnostic tests that an ECU shall support will be specified in the ECU's Subsystem Specific Diagnostic Specification.

NOTE: An ECU is only required to support the execution of a single test at any given time.

10.5.1 On-demand Self-test ($02) - Mandatory

The on-demand self-test is a mandatory diagnostic test and should attempt to minimally detect and identify all of the ECU's internal I/O circuit faults (e.g., short to battery, short to ground, open circuit, etc.).

10.5.1.1 On-Demand Self-Test Guidelines

An ECU's on-demand self-test may use the following guidelines:

- All ECU inputs (e.g., door ajar, 4x4 low, etc.) should be physically placed in a defined state and verified to be in that state.

NOTE: Since open circuits are the predominant I/O circuit fault type, inputs should be set to a state where an "open" would be detected as a circuit fault.

- All ECU outputs that employ feedback capability should be energized long enough to detect a fault, and if one exists, to identify its type (open, short-to-Vbatt, short-to-ground, etc.).

- All ECU outputs that control subsystem functions that can be monitored should be actuated and the subsystem's functionality verified (e.g., energizing a shock damper motor output that cycles a shock's firm / soft ride type and monitoring a separate shock ride setting input).

NOTE: Upon completion of the on-demand self-test, the ECU shall return its outputs to their initial entry conditions (in existence prior to the ECU entering on-demand self-test). If the ECU returns to operational state from a diagnostic session timer timeout (See Section 5.8.6), then the outputs will be set to the states that were set prior to entering diagnostic state.

10.5.1.2 On-demand Self-test Execution Time

An ECU shall complete its on-demand self-test within a maximum of 30 seconds after receiving the request to perform the test.

10.5.1.3 On-demand Self-test I/O Circuit Omission

Omitting the evaluation of an ECU I/O circuit shall only be permitted for the following reasons:

- Testing a circuit would cause the 30-second on-demand self-test execution time limit to be exceeded.
- Testing a circuit would require operator interaction during the test.
- Testing a circuit could damage the ECU and/or any of its components.
- Testing the circuit could cause a hazardous condition.
- The reliability of the circuit is good enough such that it does not fall within the top 80 percent (highest probability of occurrence) of the faults listed in the FMEA for the ECU Subsystem.

10.5.2 Assembly Self-test ($11) - Optional

The assembly self-test is strongly recommended if the on-demand self-test ($02) is longer than 20 seconds. If the on-demand self-test is longer than 20 seconds, there is a risk it will not be performed at the assembly plant. The assembly on-demand self-test should attempt to verify correct assembly of an ECU and as many of its components as possible. The list of on-demand DTCs that can be logged by this test shall be a sub-set of the DTCs logged for the on-demand self-test. Also, the test shall complete execution of the entire test within 5 seconds.

10.5.3 ECU Specific Functional Tests - Optional

ECU specific functional tests are diagnostic tests used to determine the operating condition of the ECUs circuitry and components that aren't fully tested during the on-demand self-test. These tests are unique for each ECU and shall be defined in the ECUs Subsystem Specific Diagnostic Specification.

10.5.4 Execute Downloaded Routine ($05) - Optional

Test $05 is intended for computer driven test stands to allow verification and testing of module functionality per manufacturer defined tests and execution routines. The module's Subsystem Specific Diagnostic Specification shall specify the starting address of where the execution routine is to be downloaded and the amount of memory available for the routine (an appropriate download procedure detailed in the *Module Programming and Configuration Specification*[8] shall be used). The execution of the routine will use mode $31 and test number $05. This gives the flexibility to set up any needed tests for the module without re-design or re-allocation of module resources.

10.5.5 Logging Diagnostic Test Results

When a failure is detected during the execution of a diagnostic test, the ECU shall log an on-demand DTC that identifies the fault and shall be specified in the ECU's Subsystem Specific Diagnostic Specification.

NOTE: The data returned after the execution of a diagnostic test is only expected to be correct while in the diagnostic state.

10.5.6 ECU Inputs Not in Expected State

If an attempt to enter any execution routine is made and the ECU inputs are not in the required states to execute the test as defined by the ECU, then the ECU shall have two options:

- Respond to the request to enter execution routine state with a mode $7F, response code $22 (conditions not correct).
- Execute only the portions of the execution routine where the ECU inputs are in the required states. For ECU inputs that are not in the required states, if the ECU can detect that a valid fault is the cause of the problem then a valid on-demand DTC related to that fault shall be logged. If the ECU can not discriminate between a valid fault and operator error (e.g., failure to physically close the door when required inputs are all doors closed), then an on-demand DTC shall be logged related to the "unexpected" state of the inputs instead of a known fault against the input.

11 Information Transfer State Requirements

11.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements for an ECU implementing the optional information transfer state and methods for reading from ECU memory through the use of an upload routine.

11.2 Information Transfer State Modes

While in the information transfer state, an ECU shall support all of the mandatory supported diagnostic modes shown in Table 11.1.

Mode #	*Description*
$20	Operational state entry request
$22	Request parameter by PID
$62	Report parameter by PID
$36	Block data transfer message
$37	Request block transfer exit

Table 11.1 Mandatory Diagnostic Modes for Information Transfer State

Other optional diagnostic modes may be supported in the information transfer state as shown in Table 11.2.

Mode #	*Description*
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop transmitting requested data
$27	Request security access
$67	Report security access. Required if mode $27 is supported
$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2C	Dynamically define diagnostic data packet.
$3F	Tester present

Table 11.2 Optional Diagnostic Modes for Information Transfer State

11.3 Entering the Information Transfer State

To enter the information transfer state from diagnostic state or secure diagnostic state, an ECU shall meet any entry conditions specified in its Subsystem Specific Diagnostic Specification. The transition to information transfer state is triggered upon the ECU receiving a valid request download routine entry or request upload routine entry message.

11.4 Data Block Download Procedure - Optional

For detailed information on the download procedure refer to the Module Programming and Configuration Specification[8].

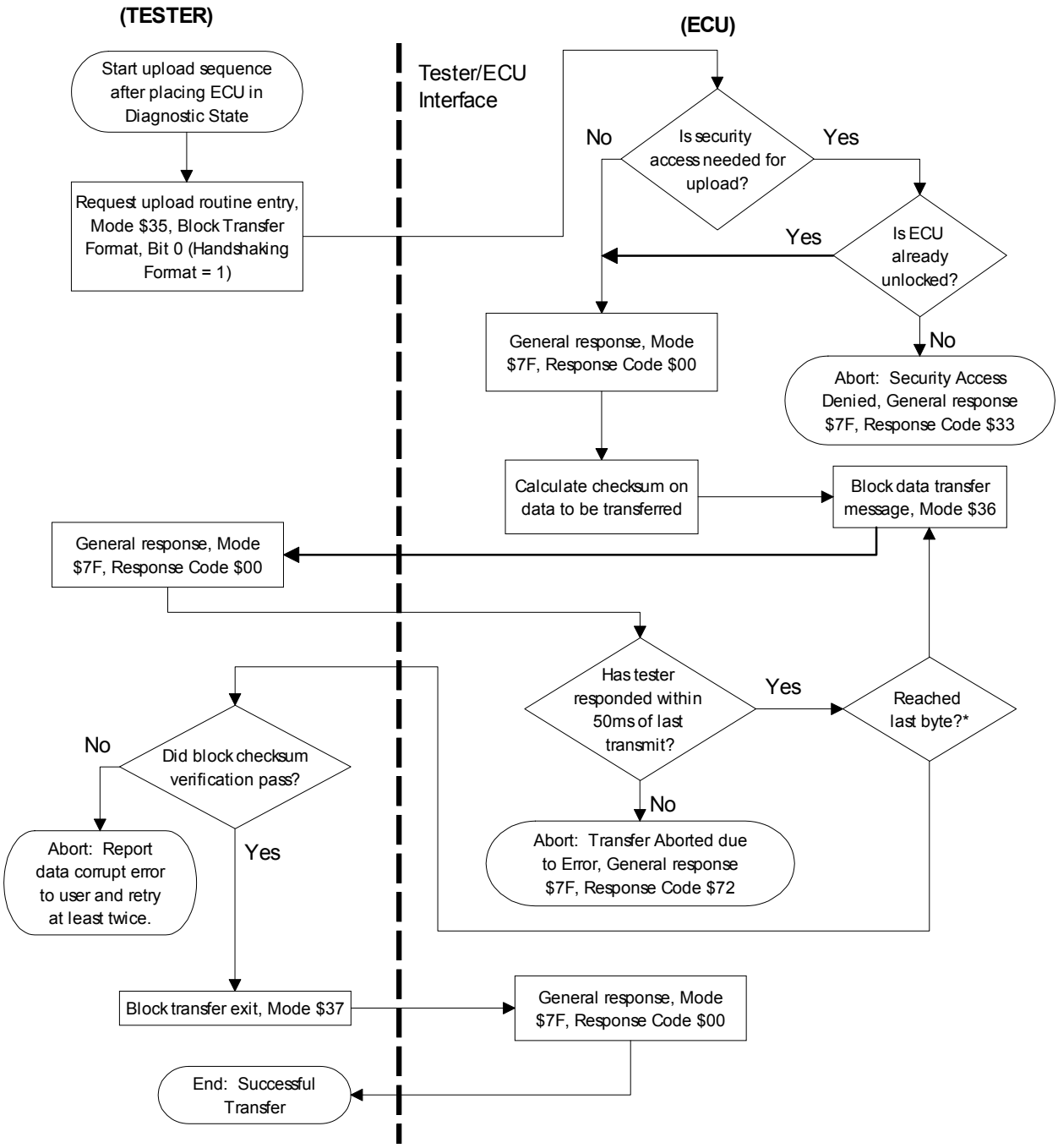
11.5 Exiting the Information Transfer State

The ECU shall exit the information transfer state if any of the following conditions occur:

- The ECU receives a valid request block transfer exit message
- The diagnostic session timer expires

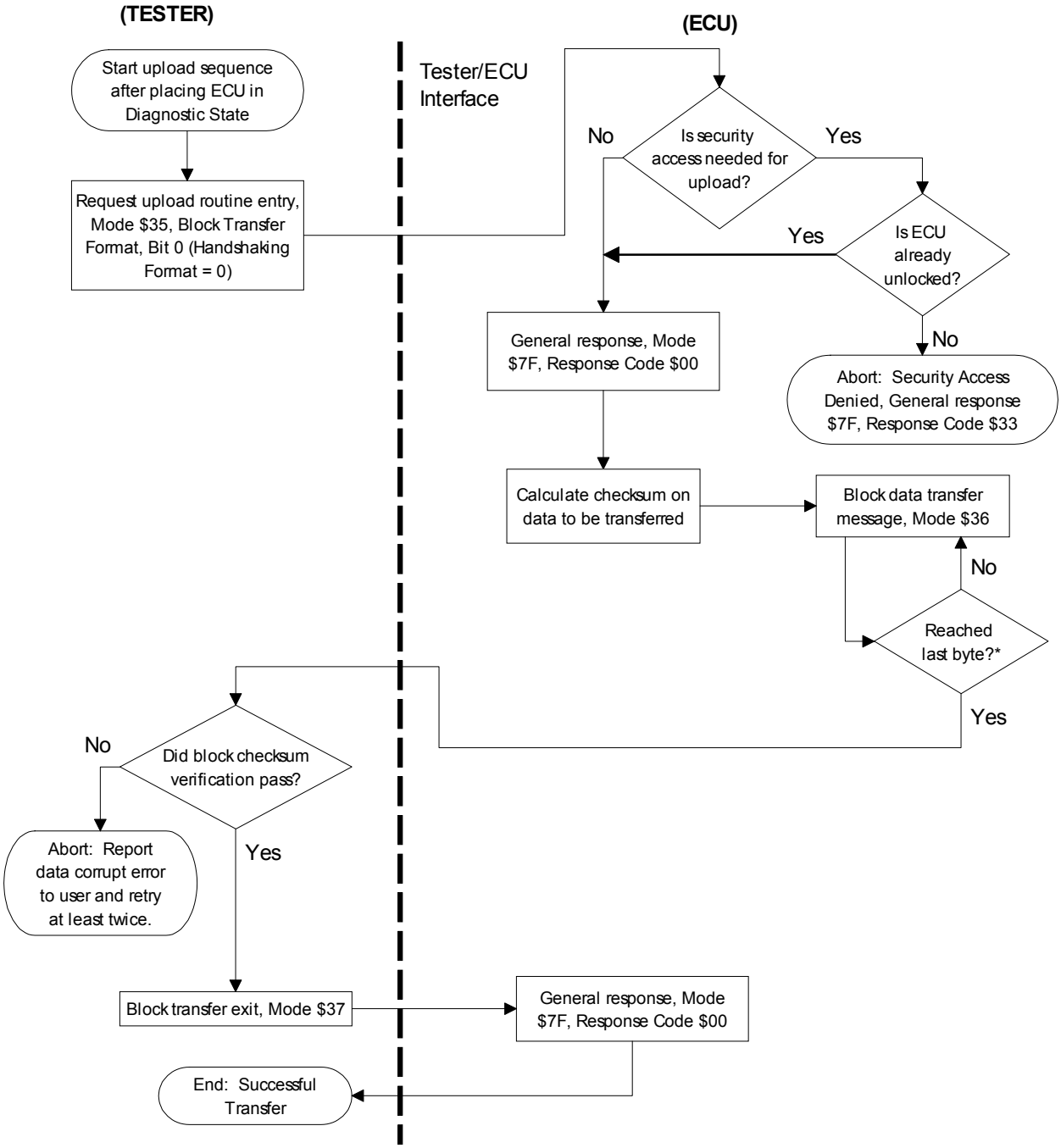
11.6 Data Block Upload Procedure - Optional

This upload procedure (shown in Figure 11.1 through Figure 11.2) utilizes the diagnostic block transfer mode to upload memory blocks from an ECU's memory to an off-board tester.



Note:
The block checksum is an additive two-byte checksum and shall be included as the last two byes to be transferred by the ECU. The block checksum is NOT included in the total byte count as specified in the Mode $35 Request Upload Routine Entry message. The determination of "Reached last byte?" shall include the transmitting of the block checksum which shall be treated exactly the same as if it were two additional data bytes to be transferred by the ECU. (e.g., the MSB of the block checksum shall immediately follow the last data byte in the Mode $36 Block data transfer message, and the LSB of the block checksum shall immediately follow theMSB)

Figure 11.1 Data Block Upload Procedure with Handshake



Note:
The block checksum is an additive two-byte checksum and shall be included as the last two byes to be transferred by the ECU. The block checksum is NOT included in the total byte count as specified in the Mode $35 Request Upload Routine Entry message. The determination of "Reached last byte?" shall include the transmitting of the block checksum which shall be treated exactly the same as if it were two additional data bytes to be transferred by the ECU. (e.g., the MSB of the block checksum shall immediately follow the last data byte in the Mode $36 Block data transfer message, and the LSB of the block checksum shall immediately follow theMSB)

Figure 11.2 Data Block Upload Procedure without Handshake

12 Input Integrity Test State

12.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements for an ECU implementing the optional input integrity test state. This state allows the tester to obtain write access to a limited set of parameters while the ECU maintains normal functionality.

12.2 Input Integrity Test State Summary

To enter the input integrity test state from operational state, an ECU shall meet any entry conditions specified in its Substem Specific Diagnostic Specification. Input integrity test state shall support all of the normal run-time operations that are supported in operational state, yet provide additional functions to the tester such as input and output parameter control. The parameters can be items such as internal register values in the ECU or output drivers. The intention of this state is to determine how the module reacts to certain changes to its registers and other parameters, while performing the normal functions defined in the operational state.

12.3 Input Integrity Test State Modes

While in the input integrity test state, an ECU shall support all of the mandatory supported diagnostic modes shown in Table 12.1.

Mode #	***Description***
$10	Diagnostic state entry request
$20	Operational state entry request
$13	Request stored codes
$53	Report stored codes
$14	Request clear stored codes
$22	Request parameter by PID
$62	Report parameter by PID
$B2	Input integrity test state entry request

Table 12.1 Mandatory Diagnostic Modes for Input Integrity Test State

While in the input integrity test state, an ECU may support any of the diagnostic modes shown in Table 12.2.

Mode #	***Description***
$12	Request Freeze Frame Data
$52	Report Freeze Frame Data. Required if mode $12 is supported
$23	Request parameter by DMR (Powertrain control modules only)
$63	Report parameter by DMR. Required if mode $23 is supported
$2F	Input/Output Control by PID
$24	Request parameter scaling/masking
$64	Report parameter scaling/masking. Required if mode $24 is supported
$25	Stop transmitting requested data
$27	Request security access
$67	Report security access. Required if mode $27 is supported

$2A	Request diagnostic data packet(s)
$6A	Report data packet(s). Required if mode $2A is supported
$2C	Dynamically define diagnostic data packet.
$3F	Tester present

Table 12.2 Optional Diagnostic Modes for Input Integrity Test State

13 Manufacturer State Requirements

13.1 Purpose / Scope

This section defines the diagnostic guidelines and requirements for an ECU implementing the optional manufacturer state. This state allows the tester to place the ECU into a non-standard state of operation when required in manufacturing or service. Examples for the use of this state include placing an ECU into a low-power mode of operation for transport, placing an ECU into a state for in-plant testing that allows conditions beyond normal operation, or shipping an ECU to an assembly plant in an "unsecured" state to expedite end-of-line programming and configuration.

13.2 Entering the Manufacturer State

An ECU shall enter the manufacturer state upon receiving a valid request manufacturer state entry message. To enter manufacturer state from a given diagnostic state, an ECU shall meet any and all entry conditions specified in its Subsystem Specific Diagnostic Specification.

13.3 Manufacturer State Operation

While in the manufacturer state, all diagnostic modes are optional. The choice of supported modes shall be detailed in the ECUs Subsystem Specific Diagnostic Specification.

13.4 Exiting the Manufacturer State

Exiting of manufacturer state shall be ECU specific, based upon the individual requirements and circumstances that manufacturer mode is used for. For example, an ECU may need to determine that the correct electronic components have been installed in the vehicle relative to the option content programmed in the ECU before allowing an exit from manufacturer state. Manufacturer state may be exited upon reception of a valid request operational state entry message if all other exit criteria are met. Manufacturer state may be exited automatically by an ECU, if necessary, when all exit criteria are satisfied. All exit criteria shall be specified in its Subsystem Specific Diagnostic Specification.

13.4.1 Diagnostic Session Timeout

If necessary for a given ECU, the manufacturer state shall be immune from the 5 second diagnostic session timeout.

NOTE: Due to the diagnostic session timeout exemption, it is recommended that any ECU implementing the manufacturer state without the timeout, also implement a manufacturer enable counter. Upon transition to the manufacturer state, this counter shall be initialized to a predetermined value. Every key cycle, this counter shall be decremented and if the counter reaches zero while still in the manufacturer state a transition shall be made to operational state. The purpose of this is to prevent vehicles from being delivered to customers in a non-standard state of operation, such as "unsecured" modules which would not have security access protection against reprogramming.

14 Gateway State Diagnostic Requirements

14.1 Purpose / Scope

This section specifies the diagnostic requirements for the design of systems that utilize a gateway module(s). Diagnostic gateway modules translate diagnostic messages from one protocol to another to enable a tester to communicate to modules that only communicate on protocols not supported by a tester; see an example of a gateway topology in Figure 14.1.

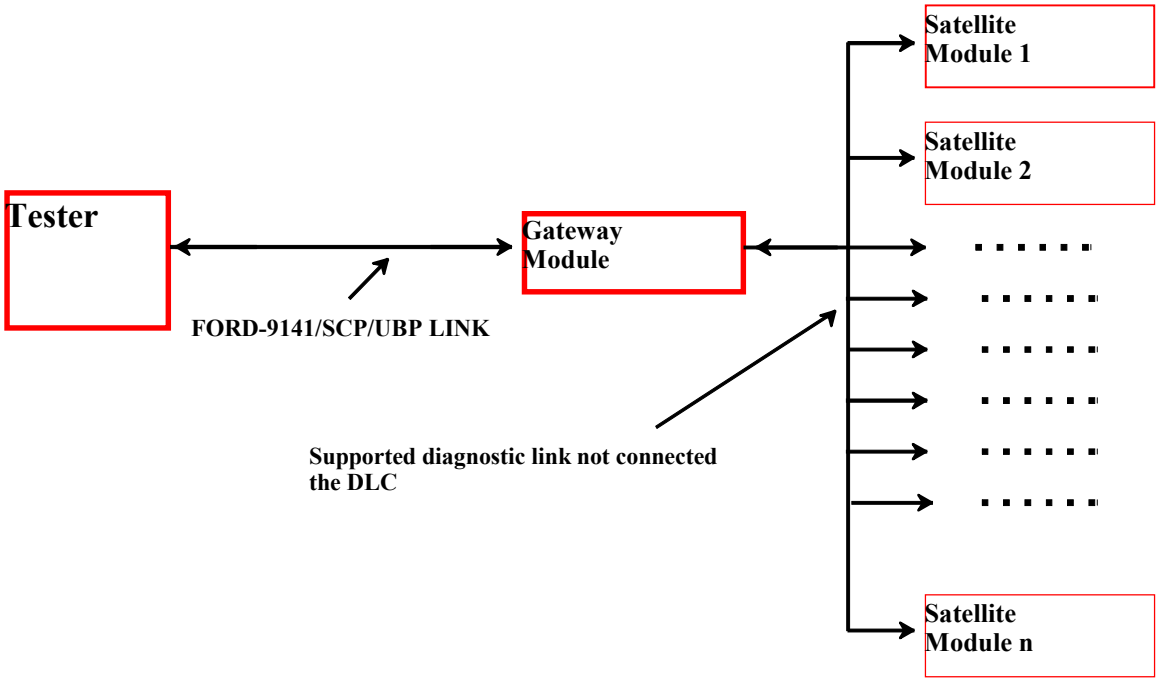


Figure 14.1 Example of Multi-Module Gateway System.

14.2 Gateway Module

While not operating in the gateway state, a gateway module shall respond directly to all request messages directed to it from a tester as defined in "Appendix A". Gateway modules are the interface through which a tester may communicate with satellite modules.

14.3 Gateway State

Gateway state is the operating state that a gateway module is directed to enter by a tester which allows the tester to indirectly communicate with the satellite modules.

14.3.1 Entering Gateway State

To enter the gateway state, the gateway module shall meet the following entry requirements:

- The gateway module shall be in the diagnostic state.

- No subsystem specific condition exists that prevents the gateway module from entering the gateway state.
- The gateway module receives a correctly formatted diagnostic command $0300 message with valid data.

Upon receipt of this message, the gateway module shall route all diagnostic messages, except for command $0300, to the satellite module. The satellite module is the target for diagnostic evaluation.

14.3.1.1 Gateway Module in Gateway State

After being placed in the gateway state, a gateway module shall translate request messages received from a tester into the protocol supported by the satellite module and forward those requests to the satellite module it points to. The gateway module's satellite module protocol translation algorithm shall be capable of translating the collective set (equivalent to a logical OR) of all the messages supported by the satellite module. The gateway module shall also translate response messages received from a satellite module into the tester supported protocol and return those responses to the tester.

14.3.1.2 Gateway State Message Handling

Once a gateway module is in the gateway state, the following rules for tester-gateway-satellite communications apply.

14.3.1.2.1 Messages: Tester to Gateway

- All request messages received by a gateway module from a tester that are formatted correctly, except for command $0300 "Gateway State Access," shall be translated and sent to the satellite module.
- If the gateway module receives command $0300 "Gateway State Access" with data byte 5 = $00, it shall return a General Response (mode $7F) of affirmative (code $00), and switch its pointer to the satellite module identified in data byte 4 of the command $0300 message.
- If the gateway module receives command $0300 "Gateway State Access" with data byte 5 = $10, it shall return a General Response (mode $7F) of affirmative (code $00) and exit the gateway state to the diagnostic state.
- If the gateway state receives command $0300 "Gateway State Access" with data byte 5 = $20, it shall return a General Response (mode $7F) of affirmative (code $00) and exit the gateway state to the operational state.

NOTE: Upon returning to the operational state, the gateway module shall have 500 milliseconds to reinitialize. During this time period, the gateway module is not required to respond to requests received from a tester.

- If the gateway state receives command $0300 "Gateway State Access" with data byte 5 not equal to $00, $10 or $20, it shall return a General Response (mode $7F) of invalid data (code $12).
- If the gateway state receives command $0300 "Gateway State Access" with data byte 4 containing a value that is not a recognized satellite module address, it shall return a General Response (mode $7F) invalid data (code $12).
- If the gateway module receives a request message that is not supported in the translation algorithm of the gateway module, it shall return a General Response (mode $7F) of mode not supported (code $11).

14.3.1.2.2 Messages: Gateway to Satellite

Satellite modules that receive translated requests from a gateway module shall handle the various translated requests as follows:

- If the satellite supports the request, it shall formulate a proper response and return it to the gateway module.
- If the satellite receives a mode that isn't supported, it shall respond with a General Response (mode $7F) of mode not supported (code $11) to the gateway module.
- If the satellite receives a data byte that isn't valid, it shall respond with a General Response (mode $7F) of invalid data (code $12) to the gateway module.

14.3.1.2.3 Messages: Satellite to Gateway

Gateway modules shall process satellite module responses as follows:

- Correctly formatted satellite responses shall be translated to the communication protocol linking the gateway module and the tester.
- Gateway modules shall ignore a satellite's response if it isn't pointed to by the gateway module at the time of the response.

14.4 Gateway State Timing Requirements

The response time between the trailing edge of the last bit of a request message issued by a tester and the leading edge of the response message returned to the tester shall not exceed 275 milliseconds; the tester shall wait 300 milliseconds before logging or indicating an error.

14.5 Satellite Modules

Satellite modules shall adhere to all requirements specified within this specification. Satellite modules are the indirect target of tester request messages when a gateway module is in the gateway state.

NOTE: All tester messages are sent to the gateway module address, not the satellite module address.

14.5.1 Satellite Module in Diagnostic State

A satellite module shall implement a diagnostic session timer as specified in Section 5.8.6.

14.5.2 Satellite Module Mandatory PIDs

In addition to supporting the mandatory PIDs defined in this specification, all satellite modules shall support PID $D12D "Node Address."

14.6 Redirecting Communication

To redirect communication to another satellite module when the gateway module is already in gateway state, the tester shall send command $0300 "Gateway State Access" message, with the new satellite module node ID specified in data byte 4 and data byte 5 = $00 for commanding the gateway module to either stay or enter into the gateway state.

15 Diagnostic Command Requirements

15.1 Purpose / Scope

This section explains what a diagnostic command is, reviews how it is used and identifies its message structure.

15.2 Diagnostic Command Definition

A diagnostic command is a message issued by a tester to instruct the ECU to perform a specific function. All diagnostic commands shall be defined in the ECU's Subsystem Specific Diagnostic Specification. Also, all commands, including supplier reserved commands, shall be listed in its corresponding Subsystem Diagnostic Specification.

NOTE: Definitions of all commands, except for those in the supplier reserved range, must be approved by the R&VT EESE Core Network Communication Section. All approved commands and their descriptions are maintained in the MRDB.

15.3 Historical Command Usage and Definition

Commands have been traditionally used to control ECU outputs for diagnostics. The Input/Output Control by PID message (mode $2F) is now the mechanism by which module outputs are to be controlled.

15.3.1 Diagnostic Command Message Information

A diagnostic command message is recognized by mode $B1 in data byte 1 of a diagnostic message.

15.4 Diagnostic Command Classifications

The following classifications of diagnostic commands listed in Table 15.1 may be used to control ECU functions.

Command	Classification	Data Size
Direct		0 Bytes
State Encoded	SED	1 Byte
Bit Mapped	BMP	1-4 Bytes
Numeric	NUM	1-4 Bytes
Binary Coded Decimal	BCD	1-4 Bytes

Table 15.1 Command Data Type Classifications

15.4.1 Direct Commands

Direct commands may be used to control specific ECU functions without specifying extra instructional data in the diagnostic command message's four optional command data bytes (Data bytes 4-7). A single direct command can only be used to toggle one ECU function (i.e., one direct command per function).

15.4.2 State-Encoded (SED) Commands

State-encoded commands use a hexadecimal value in the diagnostic command message's first optional command data byte (Data byte 4) to control up to 256 ($00 - $FF) unique functions.

15.4.3 Bit-Mapped (BMP) Commands

Bit-mapped commands may be used to control up to 32 ECU functions based on the state of the bits in each of the diagnostic command message's optional command data bytes.

15.4.4 Numeric (NUM) Commands

Numeric commands may be used to control specific ECU functions based on the hexadecimal value of data supplied in the diagnostic command message's optional command data bytes.

15.5 Diagnostic Command Entry Criteria

The ECU shall be in the diagnostic state or secure diagnostic state before a module shall be permitted to execute a diagnostic command.

NOTE: Any necessary protection shall be implemented within the ECU to eliminate any dangerous or damaging operations to occur.

15.6 Diagnostic Command Exit Criteria

The effect of a diagnostic command may remain active continuously while the ECU is in a diagnostic state other than operational state, or for a timed duration, depending on the chosen implementation. Once the ECU transitions to the operational state, any diagnostic commands that hold normal outputs in an uncontrollable position, shall cease to remain active. For example, if a diagnostic command is used to turn on the right turn signal indicator, the indicator may remain active for as long as the ECU remains in a diagnostic state other than operational state. When the ECU transitions to operational state, the right turn signal indicator shall resume it's normal operation of off or on depending on normal operating conditions.

15.7 Supplier Reserved Command Range

The range of commands from $F000 through $FFFF are reserved for use by ECU suppliers. Supplier reserved commands do not need to be approved by the R&VT EESE Network Communication Core Section. All supplier reserved commands shall be defined in the ECUs Subsystem Specific Diagnostic Specification. The exceptions to the supplier reserved range are commands $F001, $F004, $F010, and $FACE. These four commands are already assigned specific functionality by Ford Motor Company and are not available for use as supplier specific.

NOTE: Supplier reserved commands are not supported by Ford diagnostic service tools.

16 Parameter Identifier (PID) and Data Packet Requirements

16.1 Purpose / Scope

This section describes PID types, how PID data is accessed, how output PIDs work and defines mandatory PIDs that all serial communications link ECUs shall support.

NOTE: The definitions of all PIDs must be approved by the R&VT EESE Core Network Communication Section. All approved PIDs and their descriptions are maintained in the MRDB.

16.2 PID Types and Classifications

Parameter Identifiers (PIDs) can be of various types to be able to read and write different data types; they are listed in Table 16.1. Also, the amount of data that a PID can hold ranges from one to four bytes.

Type	*Classification*	*Data Size*
ASCII	ASC	1 - 4 Bytes
Binary Coded Decimal	BCD	1 - 4 Bytes
State Encoded	SED	1 Byte
Bit Mapped	BMP	1 - 4 Bytes
Numeric	NUM	1 - 4 Bytes
Packeted	PKT	1 - 4 Bytes

Table 16.1 PID Data Type Classifications

16.2.1 ASCII (ASC) PIDs

ASCII PIDs are used to provide information such as Vehicle Identification (VIN) and module serial numbers, and are encoded using standard ASCII characters between $00 and $7F. ASCII PIDs contain one character per byte.

16.2.2 Binary Coded Decimal (BCD) PIDs

BCD PIDs are used to provide information such as times, dates and serial numbers. BCD PIDs contain two numbers per byte, each number ranges from 0-9.

16.2.3 State Encoded (SED) PIDs

State encoded PIDs use hexadecimal values to represent unique states. SED PIDs are to be 8-bits in length when not used in packeted parameters. If more than one SED byte is defined for a PID, then the PID is to be a packeted one and each packet may be less than 8-bits in length.

16.2.4 Bit-Mapped (BMP) PIDs

Bit-mapped PIDs are used to provide information on the discrete state of ECU I/O signals (enabled or disabled, opened or closed, shorts to ground, shorts to battery, open circuits, etc.,). Bit-mapped PIDs specify information based on the value, ”1” or ”0”, of the bits in each PID data byte. Bit-mapped PIDs shall never be of the packeted type.

16.2.5 Numeric (NUM) PIDs

Numeric PIDs are used to represent signed or unsigned data that has a defined range of values and a specific resolution that can be represented through hexadecimal encoding. Each numeric element of a packeted PID shall be larger than 1 (if a parameter is 1 bit it shall be a bit-mapped parameter) and less

than 31 bits in length (if a numeric parameter is 32 bits it can not be a part of a packeted PID, and if a numeric parameter is 31 bits it also can not be part of a packeted PIDs since packeted PIDs must consist of groups of parameters of the same classification).

16.2.5.1 Unsigned Numeric PIDs

Unsigned numeric PIDs are used to represent data that takes on a range of increasing hexadecimal values starting at an offset value. Examples of unsigned numeric PIDs are vehicle speed and the number of continuous DTCs being stored by the ECU. Unsigned numeric PIDs have a defined range and are assigned a resolution per count and offset that is used to convert the PID value to an actual engineering value.

16.2.5.2 Signed Numeric PIDs

Signed numeric PIDs are used to represent data that is encoded using 2's complement arithmetic. If the most significant bit of the PID data is a "1", then the data is a negative value determined by computing the data's 2's complement. Signed numeric PIDs have a defined range of values and an assigned resolution per count that is used to convert the computed 2's complement value to an actual engineering value.

16.2.6 Packeted (PKT) PIDs

Packeted PIDs are groups of data that are a combination of parameters of the same classification, excluding the packeted type; this has not always been the case with previous versions of this specification (i.e. the Base Part Number PID doesn't comply with this rule). Packeted PIDs can be used to increase the retrieval rate of PID data that would otherwise be retrieved through a series of sequential requests for individual PIDs.

16.2.7 Output PIDs

Where unconditional tester control of an output could result in control module damage, time limits and other tester control sustaining criteria must be incorporated in the control module. It is the responsibility of the control module manufacturer along with the system developer to ensure that tester control is only allowed during safe conditions or time periods. All criteria must be documented in the control module's Subsystem Specific Diagnostic Specification.

16.2.7.1 Output State Monitoring

Two parameters can be associated with a control module output: the value or logic state the control module drives or writes to its output and the actual signal (value or state) that is present on the output line. PIDs shall be available for these output parameters. For the purpose of this document the parameter associated with the value written by the control module to its output is termed the output and the parameter associated with the actual signal produced is termed the output feedback.

16.2.7.1.1 Example of a Single PID Supported for Multiple Outputs

In cases where multiple control module output pins are activated for a single function, a single output PID may be provided to reflect the actual command issued by the control module functional software.

- **Example 1:** Side and tail lamps driven directly from the control module using four separate output drivers.
- **Example 2:** Directional indicators, each side may consist of front, rear, repeater and trailer outputs each fed from a separate output driver.

16.2.7.2 Output Feedback PIDs

Where appropriate, a control module may incorporate the ability to monitor the actual activity (feedback) on its output pins. Output feedback PIDs (current, voltage, frequency etc.) may be provided in association with the monitoring circuitry.

16.2.8 Input PIDs

Input PIDs point to parameters such as discrete I/O signals, analog I/O signals, time durations, activity counts etc. All input PIDs implemented within an ECU shall be defined in its Subsystem Specific Diagnostic Specification.

Each input PIDs shall provide the current value of its input, regardless of the state the ECU may be in.

NOTE: Input PIDs can be any of the defined PID types.

16.3 PID ACCESS

16.3.1 PID Reading

A tester shall retrieve PID information from an ECU through a Request Parameter by PID (mode $22) message. The ECU shall return the requested PID data via the Report Parameter by PID (mode $62) message. In the event that the ECU cannot concur with a PID request, the ECU shall respond according to the message response guidelines specified in "Appendix A".

16.3.2 PID Control

PID control shall be performed through the Input/Output Control by PID (mode $2F) message. The ECU shall respond to a mode $2F with the appropriate response message indicating whether the control took place or not. All valid responses to a mode $2F are listed in Appendix A. Input/Output Control by PID shall be used for temporarily bypassing inputs (e.g., real world sensors) and for directly controlling output devices only. The substituted value shall always revert back to the normal value as defined by the control system when there is no diagnostic dialog between the tester and ECU for five seconds. The value may revert back to the normal value sooner than this based upon a predefined timed duration.

16.4 PID State Support

An ECU shall be capable of performing PID reads while executing in the following states and modes:

- Operational State
- No Stored Codes Logging State
- Diagnostic State
- Execution Routine State
- Information Transfer State
- Secure Diagnostic State
- Input Integrity Test State

NOTE: For direction about responses to PID requests in the various operating states, refer to "Appendix A".

16.5 PID Scaling and Masking Capability - Optional

PID scaling and/or masking information is requested using mode $24. The response information is sent in mode $64 from the ECU. Scaling is used only for numeric PID types and masking is used only for bit-

packed types (See modes $24 and $64 of "Appendix A" for a table showing the message structure for these modes).

Table 16.2 describes the encoding of the "Parameter Information Byte" of mode $64 as follows:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
TYPE OF PARAMETER : 0000 = UNSIGNED NUMERIC 0001 = SIGNED NUMERIC 0010 = BIT PACKED PARAMETER 0011 = BIT MAP W/MASK 0100 = BCD 0101 = STATE ENCODED 0110 = ASCII CHARACTER 0111 = FLOATING POINT 1000 = PACKET 1001 THRU 1111 RESERVED				PID SIZE : NUMBER OF BYTES 0000 = RESERVED 0001 = 1 BYTE 0010 = 2 BYTES 0011 = 3 BYTES 0100 = 4 BYTES 0101 THRU 1111 = RESERVED			

Table 16.2 Encoding of Parameter Information Byte (Data Byte 4, Mode $64)

The high order nibble of the information byte defines the type of information encoding that is used to represent the parameter. The low order nibble can be read directly to determine the number of bytes used to represent the parameter; this is the parameter size. The size may be 1, 2, 3 or 4 bytes.

The "PID scaling / mask 1" byte of mode $64 contains either scaling or masking information. The number represented in this byte specifies scaling by indicating the binary point location. The binary point may be moved 127 places to the right ($FF) or 128 places to the left ($00). The binary point is initially assumed to be to the right of the LSB without this information.

The "PID scaling / mask 1" byte can also contain mask information that specifies which bits of the PID are supported for the current application. The two mask bytes at the end of the mode $64 message are used for bit-mapped parameters up to 2-bytes in length.

16.6 Supplier Reserved PID Range

The range of PIDs from $C900 through $C9F7 are reserved for use by ECU suppliers, with the exception of $C9F8 through $C9FF which may only be used according to the latest version of the *Module Programming and Configuration Specification*[8]. Supplier reserved PIDs do not need to be approved by the R&VT EESE Network Communication Core Section. All supplier reserved PIDs shall be defined in the Subsystem Specific Diagnostic Specification for that ECU.

NOTE: The service tools do not support supplier reserved PIDs.

16.7 Mandatory PIDs

Table 16.3 defines the mandatory PIDs that shall be supported by all ECUs connected to the diagnostic link connector.

PID	*Description*	*Classification*	*Size*
$0200	Number of Continuous DTCs	NUM	1 Byte
$0202	Number of DTCs from most recent test	NUM	1 Byte

$D100	ECU Operating State / Mode	SED	1 Byte
$E200	Software Version Number	PKT	3 Bytes
$E217	Part Number Identification Base	PKT	4 Bytes
$E219	Part Number Identification Suffix	PKT	2 Bytes
$E21A	Part Number Identification Prefix	PKT	4 Bytes

Table 16.3 Mandatory Supported PIDs

16.7.1 Number of Continuous DTCs (PID $0200)

The Number of Continuous DTCs PID contains the number of continuous DTCs currently being stored by the ECU.

16.7.2 Number of Trouble Codes Set Due to Diagnostic Test (PID $0202)

The number of trouble codes Set due to diagnostic test PID contains the number of on-demand DTCs generated during the most recent diagnostic test executed by an ECU.

16.7.3 ECU Operating State (PID $D100)

ECU Operating State PID contains the ECU's current operating state/mode, see Table 16.4.

Value	*Operational State*
$01	Operational State
$02	Diagnostic State
$03	On-Demand Self-Test / Execution Routine State
$07	Information Transfer State
$09	No Stored Codes Logging State
$0A	Input Integrity Test State
$0B	Secure Diagnostic State
$0D	Manufacturer State

Table 16.4 Applicable ECU Operating States (PID $D100)

16.7.4 Software Version Number (PID $E200)

The Software Version Number PID instructs an ECU to return the software revision level and release date, see Table 16.5.

Byte 1			Byte 2	Byte 3
Bits 7-4 Revision Level	**Bits 3-0 Month of Year**		**Day of Month**	**Year**
Num (no offset)	**State**	**State Value**	**Num with $00 offset**	**Num with #1900 decimal offset**
$00-$15	$1	January	$01-$1F	$00 = year 1900
	$2	February		...
	$3	March		$64 = year 2000
	$4	April		...
	$5	May		$FF = year 2155
	$6	June		
	$7	July		
	$8	August		
	$9	September		
	$A	October		
	$B	November		
	$C	December		

Table 16.5 Software Version Number (PID $E200)

16.7.5 Part Number Identification Base (PID $E217)

The base part number PID contains the control module's part number "group" and "detail" information in four bytes of data. The part number information supplied by this PID supports the worldwide FAO part number identification systems, see Table 16.6.

Byte One									Byte Two									Byte Three									Byte Four								
	Bits				Bits					Bits									Bits									Bits				Bits			
	7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0
Code	Hex				Hex				Code	Alphanumeric								Code	Alphanumeric								Code	Hex				Hex			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0	2	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	1	0	2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1	3	0	0	0	0	0	0	1	1	3	0	0	0	0	0	0	1	1	3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0	4	0	0	0	0	0	1	0	0	4	0	0	0	0	0	1	0	0	4	0	1	0	0	0	1	0	0
5	0	1	0	1	0	1	0	1	5	0	0	0	0	0	1	0	1	5	0	0	0	0	0	1	0	1	5	0	1	0	1	0	1	0	1
6	0	1	1	0	0	1	1	0	6	0	0	0	0	0	1	1	0	6	0	0	0	0	0	1	1	0	6	0	1	1	0	0	1	1	0
7	0	1	1	1	0	1	1	1	7	0	0	0	0	0	1	1	1	7	0	0	0	0	0	1	1	1	7	0	1	1	1	0	1	1	1
8	1	0	0	0	1	0	0	0	8	0	0	0	0	1	0	0	0	8	0	0	0	0	1	0	0	0	8	1	0	0	0	1	0	0	0
9	1	0	0	1	1	0	0	1	9	0	0	0	0	1	0	0	1	9	0	0	0	0	1	0	0	1	9	1	0	0	1	1	0	0	1
A	1	0	1	0	1	0	1	0	A	0	0	0	0	1	0	1	0	A	0	0	0	0	1	0	1	0	A	1	0	1	0	1	0	1	0
B	1	0	1	1	1	0	1	1	B	0	0	0	0	1	0	1	1	B	0	0	0	0	1	0	1	1	B	1	0	1	1	1	0	1	1
C	1	1	0	0	1	1	0	0	C	0	0	0	0	1	1	0	0	C	0	0	0	0	1	1	0	0	C	1	1	0	0	1	1	0	0
D	1	1	0	1	1	1	0	1	D	0	0	0	0	1	1	0	1	D	0	0	0	0	1	1	0	1	D	1	1	0	1	1	1	0	1
E	1	1	1	0	1	1	1	0	E	0	0	0	0	1	1	1	0	E	0	0	0	0	1	1	1	0	E	1	1	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1	F	0	0	0	0	1	1	1	1	F	0	0	0	0	1	1	1	1	F	1	1	1	1	1	1	1	1
									G	0	0	0	1	0	0	0	0	G	0	0	0	1	0	0	0	0									
									H	0	0	0	1	0	0	0	1	H	0	0	0	1	0	0	0	1									
									I	0	0	0	1	0	0	1	0	I	0	0	0	1	0	0	1	0									
									J	0	0	0	1	0	0	1	1	J	0	0	0	1	0	0	1	1									
									K	0	0	0	1	0	1	0	0	K	0	0	0	1	0	1	0	0									
									L	0	0	0	1	0	1	0	1	L	0	0	0	1	0	1	0	1									
									M	0	0	0	1	0	1	1	0	M	0	0	0	1	0	1	1	0									
									N	0	0	0	1	0	1	1	1	N	0	0	0	1	0	1	1	1									
									O	0	0	0	1	1	0	0	0	O	0	0	0	1	1	0	0	0									
									P	0	0	0	1	1	0	0	1	P	0	0	0	1	1	0	0	1									
									Q	0	0	0	1	1	0	1	0	Q	0	0	0	1	1	0	1	0									
									R	0	0	0	1	1	0	1	1	R	0	0	0	1	1	0	1	1									
									S	0	0	0	1	1	1	0	0	S	0	0	0	1	1	1	0	0									
									T	0	0	0	1	1	1	0	1	T	0	0	0	1	1	1	0	1									
									U	0	0	0	1	1	1	1	0	U	0	0	0	1	1	1	1	0									
									V	0	0	0	1	1	1	1	1	V	0	0	0	1	1	1	1	1									
									W	0	0	1	0	0	0	0	0	W	0	0	1	0	0	0	0	0									
									X	0	0	1	0	0	0	0	1	X	0	0	1	0	0	0	0	1									
									Y	0	0	1	0	0	0	1	0	Y	0	0	1	0	0	0	1	0									
									Z	0	0	1	0	0	0	1	1	Z	0	0	1	0	0	0	1	1									

Table 16.6 Base Part Number Structure (PID $E217)

EXAMPLE:

What is the encoding for an airbag module with the base part number 14B056?

1	=	$1	Four bit hex encoding
4	=	$4	Four bit hex encoding
B	=	$0B	Ford specific
0	=	$00	Ford specific
5	=	$5	Four bit hex encoding
6	=	$6	Four bit hex encoding

What is the encoding for the anti-lock braking module with base part number 2M110?

0	=	$0	Four bit hex encoding
2	=	$2	Four bit hex encoding
M	=	$16	Ford specific
1	=	$01	Ford specific
1	=	$1	Four bit hex encoding
0	=	$0	Four bit hex encoding

16.7.6 Part Number Identification Prefix (PID $E21A)

The prefix part number PID contains the year/decade, product line, and design responsibility. The part number information supplied by this PID supports the worldwide FAO part number identification systems, see *Ford Automotive Procedures # FAP03-145* for official prefix codes. PID $E21A - Part Number Identification Number - replaces PID $E218 for model year 1999 and beyond vehicle programs. The part number identification prefix is a four position alpha-numeric code. PID $E21A is the ASCII equivalent of each of the individual alpha-numeric characters. See Table 16.7 for the format of PID $E21A. For a complete and updated list of year/decade codes, product line codes, and design responsibility, reference *Ford Automotive Procedures # FAP03-145*.

Byte 1: Year / Decade	Bytes 2 & 3: Product Line	Byte 4: Design Responsibility
ASCII equivalent of Official Year / Decade Code	ASCII equivalent of Official Product Line Code	ASCII equivalent of Official Design Responsibility

Table 16.7 Part Number Identification Prefix (PID $E21A)

EXAMPLE:

What is the Part Number Prefix for a fuel pump control module released by the Electrical and Fuel Handling Division for a Jaguar X200?

Year	-	2001	- **1**	(official year / decade code)
Product Line	-	Jaguar X200	- **R8**	(official product line code)
Design Resp.	-	Elec. Fuel & Handling Div.	- **U**	(official design responsibility)
Part Number Prefix			= **1R8U**	

The encoded data that shall be returned by an ECU in response to a tester request for PID $E21A is as follows:

1 - Byte 1 = $31 - ASCII value for '1'

R - Byte 2 = $52 - ASCII value for 'R'

8 - Byte 3 = $38 - ASCII value for '8'

U - Byte 4 = $55 - ASCII value for 'U'

16.7.7 Part Number Identification Suffix (PID $E219)

The suffix part number PID contains the ECU's basic design code, alternate part code, and change level. The information supplied by this PID supports all FAO (NAAO and IAO) part number identification systems, see Table 16.8 and the MRDB for details.

Byte 1									**Byte 2**									
	Bits									**Bits**					**Bits**			
	7	6	5	4	3	2	1		0	7	6	5		4	3	2	1	0
Code	**Basic Design**							**Code**	**Alternate Part**				**Code**	**Change Level (Revision)**				
A	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0	0	0
B	0	0	0	0	0	0	1	1	0	0	0	1	B	0	0	0	0	1
C	0	0	0	0	0	1	0	2	0	0	1	0	C	0	0	0	1	0
D	0	0	0	0	0	1	1	3	0	0	1	1	D	0	0	0	1	1
E	0	0	0	0	1	0	0	4	0	1	0	0	E	0	0	1	0	0
F	0	0	0	0	1	0	1	5	0	1	0	1	F	0	0	1	0	1
G	0	0	0	0	1	1	0	6	0	1	1	0	G	0	0	1	1	0
H	0	0	0	0	1	1	1	7	0	1	1	1	H	0	0	1	1	1
J	0	0	0	1	0	0	0	8	1	0	0	0	J	0	1	0	0	0
K	0	0	0	1	0	0	1	9	1	0	0	1	K	0	1	0	0	1
L	0	0	0	1	0	1	0						L	0	1	0	1	0
M	0	0	0	1	0	1	1						M	0	1	0	1	1
N	0	0	0	1	1	0	0						N	0	1	1	0	0
P	0	0	0	1	1	0	1						P	0	1	1	0	1
R	0	0	0	1	1	1	0						R	0	1	1	1	0
S	0	0	0	1	1	1	1						S	0	1	1	1	1
T	0	0	1	0	0	0	0						T	1	0	0	0	0
U	0	0	1	0	0	0	1						U	1	0	0	0	1
V	0	0	1	0	0	1	0						V	1	0	0	1	0
X	0	0	1	0	0	1	1						X	1	0	0	1	1
Y	0	0	1	0	1	0	0						Y	1	0	1	0	0
Z	0	0	1	0	1	0	1						Z	1	0	1	0	1
AA	0	0	1	0	1	1	0											

Byte 1									Byte 2									
	Bits								Bits						Bits			
	7	6	5	4	3	2	1		0	7	6	5		4	3	2	1	0
Code	**Basic Design**							**Code**	**Alternate Part**				**Code**	**Change Level (Revision)**				
AB	0	0	1	0	1	1	1											
AC	0	0	1	1	0	0	0											
AD	0	0	1	1	0	0	1											
AE	0	0	1	1	0	1	0											
AF	0	0	1	1	0	1	1											
AG	0	0	1	1	1	0	0											
AH	0	0	1	1	1	0	1											
AJ	0	0	1	1	1	1	0											
. . .	Refer to MRDB for complete list of values																	
BV	0	1	1	1	1	1	0											
BX	0	1	1	1	1	1	1											
BY	1	0	0	0	0	0	0											
BZ	1	0	0	0	0	0	1											

Table 16.8 Part Number Identification Suffix (PID $E219)

Examples:

```
AA = 0000000 0000 00000     A1B = 0000000 0001 00001
AZ = 0000000 0000 10101     C2Z = 0000010 0010 10101
ZZ = 0010101 0000 10101
```

16.8 Miscellaneous Identification PIDs

16.8.1 Serial Number (PIDs $E221 THRU $E226 & $E231 THRU $E236)

The Serial Number PIDs contain the ECU serial number as defined in the ECUs Subsystem Specific Diagnostic Specification.

PID	*Description*	*Classification*	*Size*
$E221 through $E226	Module Serial Number	ASC	4 Bytes each
$E231 through $E236	Module Serial Number	BCD	4 Byte each
$E300 through $E307	Vehicle Identification Number	ASC	4 Bytes each

Table 16.9 Serial Number and VIN PIDs

An ECU shall implement only as many of the serial number PIDs as needed to hold the module's serial number. PID $E221, byte 1 and PID $E231, byte 1 (high nibble) shall always contain the most

significant digit of the module's serial number when implemented. If the serial number is not a multiple of 4 for the ASCII serial number PID or a multiple of 8 for the BCD serial number PID, the PID shall be padded with $00 for the ASCII PIDs and with $F for the BCD PIDs.

An example serial number of 12345678901 is encoded in ASCII format as shown in Table 16.10.

PID $E221	PID $E221	PID $E221	PID $E221	PID $E222	PID $E222	PID $E222	PID $E222	PID $E223	PID $E223	PID $E223	PID $E223
Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2	Byte 3	Byte 4
0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x30	0x31	0x00

Table 16.10 Example of ASCII Module Serial Number Storage

An example serial number of 12345678901 is encoded in BCD format as shown in Table 16.11.

PID $E231	PID $E231	PID $E231	PID $E231	PID $E232	PID $E232	PID $E232	PID $E232
Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2	Byte 3	Byte 4
0x12	0x34	0x56	0x78	0x90	0x1F	0xFF	0xFF

Table 16.11 Example of BCD Module Serial Number Storage

16.8.2 Vehicle Identification Number (PIDs $E300 THRU $E307)

The Vehicle Identification Number PIDs contain the vehicle identification number. The entire VIN is to be stored in as many bytes as are needed in the above PID range. The VIN begins with PID E300 and progresses through PID $E304 to store the entire VIN. The VIN is to be read from left to right, for example: VIN - 1MEPM6046KH693300 is to be encoded as shown in Table 16.12.

All VIN numbers are currently set to a size of 17 alphanumeric characters. Padding of the first three bytes of PID $E300 is done to right justify the VIN throughout the range of PIDs defined to store the VIN. The hexadecimal numbers in Table 16.12 are to be translated into their ASCII representations when read.

PID $E300	PID $E300	PID $E300	PID $E300	PID $E301	PID $E301	PID $E301	PID $E301
Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2	Byte 3	Byte 4
0x00	0x00	0x00	0x31	0x4D	0x45	0x50	0x4D

PID $E302	PID $E302	PID $E302	PID $E302	PID $E303	PID $E303	PID $E303	PID $E303
Byte 1	Byte 2	Byte 3	Byte 4	Byte 1	Byte 2	Byte 3	Byte 4
0x36	0x30	0x34	0x36	0x4B	0x48	0x36	0x39

PID $E304	PID $E304	PID $E304	PID $E304
Byte 1	Byte 2	Byte 3	Byte 4
0x33	0x33	0x30	0x30

Table 16.12 Example of VIN Storage

16.9 Fault PID Description

Fault PIDs are used to report ECU faults through the use of bit-mapped parameters. Each bit-mapped PID parameter shall consist entirely of either continuous faults or on-demand faults, they shall not be mixed and there shall be an associated master DTC associated with all faults reported within a given fault PID.

A master DTC is associated with one or more fault PID bits and shall stay logged in the ECU as long as one of its fault PID bits is set.

For example, DTC $5724 “Air Suspension Height Sensor Power Circuit Failure” is a master DTC for the continuous fault PID $395C “Fault PID (CM): Air Suspension Height Sensor Power Circuit Failure”. The format of fault PID $395C is as follows (for each bit: 0 = no fault, 1 = fault):

Byte 1:
Bit 7: Air Suspension LR Height Sensor Supply Circuit Open
Bit 6: Air Suspension LR Height Sensor Supply Circuit Short to Battery
Bit 5: Air Suspension LR Height Sensor Supply Circuit Short to Ground
Bit 4: Air Suspension RR Height Sensor Supply Circuit Open
Bit 3: Air Suspension RR Height Sensor Supply Circuit Short to Battery
Bit 2: Air Suspension RR Height Sensor Supply Circuit Short to Ground
Byte 2:
Bit 7: Air Suspension LF Height Sensor Supply Circuit Open
Bit 6: Air Suspension LF Height Sensor Supply Circuit Short to Battery
Bit 5: Air Suspension LF Height Sensor Supply Circuit Short to Ground
Bit 4: Air Suspension RF Height Sensor Supply Circuit Open
Bit 3: Air Suspension RF Height Sensor Supply Circuit Short to Battery
Bit 2: Air Suspension RF Height Sensor Supply Circuit Short to Ground

Assuming PID $395C is the only fault PID associated with DTC $5724, then if DTC $5724 is logged, at least one of the twelve valid bits in PID $395C must be set equal to one. Similarly, if one of the bits in PID $395C is set equal to one, then DTC $5724 must be logged and if no bits in PID $395C are set, then DTC $5724 must not be logged.

16.9.1 Automatic Clearing of Continuous Fault PID Fault Bits

Each Fault PID bit shall have its own clearing counter and shall follow all of the rules for automatic clearing as specified in Section 6.3.2.1.

16.9.2 Tester Initiated Clearing of Continuous Fault PID Fault Bits

If a tester clears a master DTC, then all of the continuous fault PID bits associated with the master DTC shall also be cleared.

16.9.3 On-demand Fault PID Fault Bits

Please reference Section 10.4.4 for a description of the operation of on-demand fault PID fault bits.

16.10 Define/Request/Stop Data Packets

A data packet consists of PIDs, DMRs or other sets of manufacturer defined data types that can be repetitively reported (rapid data) to the tester at different rates as specified by an ECU. Reported in this way, the ECU may provide the tester with higher parameter sample rates than those achieved by the tester employing Read Parameter by PID (mode $22) or DMR (mode $23) requests.

The following constraint shall be adhered to pertaining to rapid data:

- Only one rate shall be used at any given time to report rapid data packets.

A data packet has a 1-byte identification number associated with it referred to as a DPID. A data packet (DPID) may consist of one or more different items. DPIDs are defined ”on-the-fly” before they are used by a test tool and remains defined in the module’s memory as long as the volatile memory remains intact or if it is cleared by the tester.

The off-board tester will request diagnostic data packets with a Request Diagnostic Data Packet(s) (mode $2A). The tester will set-up a dynamically defined DPID with a Dynamically Define Diagnostic Data Packet (mode $2C).

16.10.1 Dynamically Define Diagnostic Data Packet (Mode $2C)

A dynamically defined data packet is to be set up with mode $2C tester requests. A single mode $2C tester request may only be used to specify one item for one data packet (DPID). Multiple mode $2C requests will be needed to fully define a DPID containing more than one item. Each mode $2C request defines the type (i.e., by PID, DMR or other) and placement (the starting byte and the number of bytes used for the item in the DPID) of a DPID item, "See Appendix A" for the allowable parameters for this mode.

A DPID can be cleared by specifying zero as the third data byte of a mode $2C request.

NOTE: Data bytes four through seven are not used for clearing a DPID.

16.10.2 Request Diagnostic Data Packets(s) (Mode $2A)

Requesting a module to report one or multiple data packets is the operation of mode $2A. Each mode $2A request may specify up to five DPIDs to be reported by the module either continuously or once (single-shot). Each DPID within a mode $2A request will be reported with its own mode $6A response. A data rate parameter is to be specified that determines whether the response(s) are continuous or 'single-shot.' The description of the data rate parameter is specified in "Appendix A" for mode $2A. Slow, medium and fast data rates are to be supported, and the exact rate is to be specified in the module's Subsystem Specific Diagnostic Specification.

NOTE: If any packet is already being reported when this mode is issued, then the rate specified in the request must have the same rate of reporting then the one(s) currently being reported.

An ECU shall be capable of handling new mode $2A requests independently of any mode $2A requests the module may already be servicing, except for the note above.

Each continuously reported DPID shall continue to report data until it is either stopped with a mode $25, mode $2A, or mode $2C request, or upon the ECU transitioning to the operational state. The last condition prevents rapid data from being transmitted after a tester stops communicating and the ECU times out to the operational state.

16.10.3 Stop Transmitting Requested Data (Mode $25)

The mode $25 message is a request to discontinue the transmission of all repetitive data. See "Appendix A" for more detail relating to this mode.

17 UART - Based Diagnostic Protocol (UBP)

17.1 Purpose / Scope

The UBP serial communications link diagnostic protocol is described in this Section. Explanations pertaining to each of the features of this protocol are provided, ranging from message construction to message timing.

NOTE: Some of the specifications & requirements relating to enhanced diagnostics on the diagnostic supported protocols may reside in other specifications and are referenced appropriately throughout this section.

17.2 UBP Enhanced Diagnostic Protocol Description

UBP is used in vehicle applications for inter-module communication to share vehicle operational information. Diagnostics are implemented on modules equipped with UBP communications, utilizing a protocol already there for other functional purposes. Refer to the following specification for more information relating to UBP communications:

- *UBP Vehicle Network Implementation Requirements*[23]

17.3 UBP Serial Communications Link

The UBP serial communications link uses a single wire pair to provide a physical communications path between each of the vehicle's ECUs and the tester which is connected to the vehicle's DLC (SAE J1962 connector). The UBP protocol supports both ECU-to-ECU and tester-to-ECU communications.

17.4 Grounding Requirements

The UBP diagnostic communication bus shall require a single ground reference point from off-board to on-board; this connection shall be singularly provided through the DLC. The network shall operate in the presence of noise signals and voltage offsets of less than 1.0 volt between the off-board tester and any onboard ECU; this offset voltage can be partitioned to 0.350 volts between the battery ground and the off-board tester ground reference, and 0.650 volts between the battery ground and the onboard ECU ground reference.

17.5 Message Structure

Four message types exist in UBP. Only two of the four message types are defined in this document:

- ***Type 0*** This message type shall be used in Ford vehicles to accomplish inter-module communication between on-boards ECUs using a proprietary method for arbitration. Definition of the arbitration method and the message contents are <u>not</u> contained in this document; this information is specified in *UBP Protocol Definition and Interface Requirements*[24] and *UBP Vehicle Network Implementation Requirements*[23].
- ***Type 1*** This message type is not defined in this document.
- ***Type 2*** The format of this physically addressed message type is based on the FORD-9141 message structure; this message type is authorized for use in all physically addressed UBP module diagnostics and monitoring applications. Requirements for usage of this message type are contained within subsequent sections of this document.
- ***Type 3*** This message type is not defined in this document.

All messages are composed of a series of bytes that are transmitted within certain time boundaries such that the bytes are considered contiguous. Bytes are composed of individual bits arranged as described below.

17.5.1 Generic Message Format

All UBP diagnostic messages have a similar structure. All contain a three byte header consisting of message type, target identifier, and source identifier. These bytes are followed by a data field of 1 to 7 bytes and a checksum byte. There is no in-message acknowledgment byte in UBP. The general format for a message is shown below:

FORMAT BYTE	*BYTE #1*
TARGET ADDRESS	*BYTE #2*
SOURCE ADDRESS	*BYTE #3*
MODE BYTE	*BYTE #4*
OPTIONAL DATA BYTE #1	*BYTE #5*
OPTIONAL DATA BYTE #2	*BYTE #6*
OPTIONAL DATA BYTE #3	*BYTE #7*
OPTIONAL DATA BYTE #4	*BYTE #8*
OPTIONAL DATA BYTE #5	*BYTE #9*
OPTIONAL DATA BYTE #6	*BYTE #10*
CHECKSUM	*BYTE #11*

Table 17.1 Generic Message Format

17.5.1.1 Format Byte

The format byte shall be the first byte of the message header and the first byte of every UBP message; this byte contains both message type and length information. The two most significant bits shall encode the message type. The six least significant bits (transmitted first) specify length.

	<- Type Field ->		*<- - - - - - - Length Field - - - - - - ->*						
	A1	**A0**	**L5**	**L4**	**L3**	**L2**	**L1**	**L0**	
Stop Bit	*BIT 7*	*BIT 6*	*BIT 5*	*BIT 4*	*BIT 3*	*BIT 2*	*BIT 1*	*BIT 0*	***Start Bit***
	MSB							***LSB***	

Table 17.2 Format Byte Encoding

17.5.1.1.1 Message Type

The two most significant bits of the format byte, bits 7 and 6 shall encode the message type. This document defines two valid UBP message types: physically addressed and functionally addressed. The physically addressed type shall be known as type "10"; and the functionally addressed shall be known as type "00". Encoding of the message type shall be as follows:

	A1	A0
Message Type	**Bit 7**	**Bit 6**
Functional, Normal Operation	0	0
OBD Diagnostics (NOT DEFINED)	0	1
Physical, Diagnostics	1	0
(NOT DEFINED)	1	1

Table 17.3 Type Field Encoding

17.5.1.1.2 Message Length

The lower six bits (Bits 5, 4, 3, 2, 1, 0) shall encode the number of data bytes contained within the message; this field shall define the length of the message from the beginning of the data field to the checksum byte (not included). The length field shall enable a receiving module to determine the location of the checksum and end of the message. Length encoding shall be as shown in the following illustration; these are the only valid values.

Format Byte Value	***Data Bytes in Message***	Bit 3	Bit 2	Bit 1	Bit 0
--	*Reserved – Not Supported*	0	0	0	0
$81	*Mode*	0	0	0	1
$82	*Mode, Data 1*	0	0	1	0
$83	*Mode, Data 1 – 2*	0	0	1	1
$84	*Mode, Data 1 – 3*	0	1	0	0
$85	*Mode, Data 1 – 4*	0	1	0	1
$86	*Mode, Data 1 – 5*	0	1	1	0
$87	*Mode, Data 1 – 6*	0	1	1	1
--	*Not Supported for Diagnostics*	1	0	0	0
--	*Reserved -- Not Supported*	1	0	0	1
--	*Reserved -- Not Supported*	1	0	1	0
--	*Reserved -- Not Supported*	1	0	1	1
--	*Reserved -- Not Supported*	1	1	0	0
--	*Reserved -- Not Supported*	1	1	0	1
--	*Reserved -- Not Supported*	1	1	1	0
--	*Reserved -- Not Supported*	1	1	1	1

Table 17.4 Length Field Encoding

17.5.1.2 Target Byte

The second byte of the defined UBP diagnostic message types (also the second header byte) shall be the target specifier byte; this byte shall contain a 2-digit hex number that specifies 1 of 256 physical addresses. Physical addresses are uniquely defined for all modules. Module address assignment is the same across all car lines and model years.

As new modules are created, addresses will be assigned to them. Addresses will not be recycled, the assignments are controlled and maintained by the EESE Core Network Communications Section.

17.5.1.2.1 Functionally Addressed Diagnostic Messages

The only supported diagnostic functional (broadcast) message for UBP is request no stored codes logging state. The format for the functionally addressed UBP no stored codes logging message is shown in Table 17.5.

HEADER BYTES			DATA BYTES					CHECKSUM
FORMAT	PRIMARY ID	SOURCE ADDRESS	OPERATION	SECONDARY ID	DATA 1 (optional)	…	DATA 6 (optional)	CHECKSUM
$02	$B0	$XX (Tester ID)	$04	$B0	N/A	N/A	N/A	CHECKSUM

Table 17.5 Format of Functional UBP Request No Stored Codes Logging Message

For further clarification on definitions for functional "Type 0" UBP messages, please refer to the UBP Vehicle Network Implementation Requirements[23].

NOTE: An ECU shall not respond to a functional diagnostic UBP message.

17.5.1.2.2 Physically Addressed Diagnostic Messages

Physical addressing provides the ability to transfer information point to point between two modules on the network. The unique target address specified in this byte is assigned to only one module on the network. All physically addressed diagnostic messages are message type 10 binary (see Table 17.3). The first data byte of each message is dedicated to an operation code which is referred to as the MODE. Up to 256 distinct diagnostic operations may be specified by the operation mode byte. The number of data bytes in the remainder of the message is dependent upon the diagnostic operation (MODE). The format for a physically addressed diagnostic message is shown in Table 17.6.

17.5.1.3 Source Byte

The third byte of the defined UBP diagnostic message types (also the third header byte) shall be the source Identifier byte; this byte shall contain a 2 digit hex number that specifies either 1 of 256 physical addresses. Physical addresses are uniquely defined for all modules. Module address assignment is the same across all car lines and model years. The physical address assignment set used for source identifier is the same as that used for target specifier.

17.5.1.4 Data Byte(s)

The data field of every UBP diagnostic message shall contain from one to seven data bytes. Byte four of every UBP message is always the mode byte; this byte defines the diagnostic operation.

UBP - DIAG MESSAGE DEFINITION		
TYPE (2 bit) LENGTH (6 bit)	= 8?h	BYTE #1
TARGET ADDRESS	= ??h	BYTE #2
SOURCE ADDRESS	= ??h	BYTE #3
OPERATION MODE BYTE	= ??h	BYTE #4
OPTIONAL DATA BYTE #1	= ??h	BYTE #5
OPTIONAL DATA BYTE #2	= ??h	BYTE #6
OPTIONAL DATA BYTE #3	= ??h	BYTE #7
OPTIONAL DATA BYTE #4	= ??h	BYTE #8
OPTIONAL DATA BYTE #5	= ??h	BYTE #9
OPTIONAL DATA BYTE #6	= ??h	BYTE #10
CHECKSUM	= ??h	BYTE #11

Table 17.6 Physically Addressed Diagnostic Message Format

17.5.1.4.1 Mode Encoding

See Section 4 "Diagnostic Structure and Modes" and "Appendix A - Diagnostic Messages" for definitions of the different types of diagnostic mode definitions.

17.5.1.4.2 Other Encoding

Within the data field, various special encoding is are used to transfer information between the tester and the ECU. Definitions of the various types of encoding is described throughout this document. Also, encoding for PIDs, execution routine numbers, DTCs and commands are stored in the MRDB.

17.5.1.5 Checksum Byte

The last byte of every UBP message is the checksum byte; this byte contains an error check code that protects the receiving module in the case of receipt of corrupted information. The transmitter calculates the checksum using all bytes of the messages excluding the checksum byte itself. The checksum is defined as the simple 8-bit serial summation of all the bytes in the message excluding the checksum. Detail definition of the checksum algorithm is specified in the *UBP Protocol Definition and Interface Requirements*[24] specification.

17.6 Legislative OBD Emission Related Diagnostics on UBP

UBP does **not** support OBD diagnostic communications.

17.7 Message Timing Requirements

Boundary specifications are necessary to define timely and orderly flow of information on the communication bus. Maximum times are specified to provide a means to automatically terminate reception of aborted or faulty transmissions. Minimums are specified to prevent receiver overrun conditions.

17.7.1 Interbyte Gap

Messages are composed of sequentially transmitted bytes. For all messages, the timing between the completion of the stop bit of one byte and the first edge of the start bit of the next byte of that continuous message is defined in the *UBP Protocol Definition and Interface Requirements*[24] specification.

17.7.2 Intermessage Gap

Transactions are composed of messages that are sequentially transmitted. Specification of intermessage gap time sets the maximum and minimum transaction time with a 1 millisecond tolerance. The period between the completion of the last message and the start of the next message is dependent on the source of the messages. Limits are defined:

1. ECU RESPONSE FOLLOWING A TESTER REQUEST:

$0 \leq$ INTERMESSAGE GAP TIME $\leq$ 50 MILLISECONDS

This condition occurs when the ECU responds to any tester query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if an expected response is not returned within 200 milliseconds, the tester times out and may assume that none is forthcoming. The ECU shall respond to all tester requests as soon as the ECU's diagnostic response is ready.

2. ECU RESPONSE MESSAGE FOLLOWING ANOTHER ECU MESSAGE IN A SEQUENCE:

$0 \leq$ INTERMESSAGE GAP TIME $\leq$ 50 MILLISECONDS

This condition occurs when an ECU is responding with multiple messages to a single query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if the message is not returned within 200 milliseconds, the tester may time out and shall assume that no more messages will be sent.

3. TESTER REQUEST FOLLOWING AN ECU RESPONSE:

$55 \leq$ INTERMESSAGE GAP TIME $<$ 5,000 MILLISECONDS

If a response message is not received within 200 milliseconds, the tester may time out and shall assume that no more messages will be sent. A new request may be transmitted by the tester 200 milliseconds after transmission of the original request.

The upper limit of 5,000 milliseconds is set to allow an ECU to detect when no tester is connected to the communication bus. Whenever a tester is connected, it is required to send a message to each ECU at this minimum rate to continue the diagnostic state dialogue. When an ECU times out, it shall reset and revert to a non-diagnostic state. The 55 millisecond minimum time allows the ECU processing time and prevents the tester from overrunning the ECU receive buffer. Note that an ECU shall not implement any timers to ensure that the tester waits 55 milliseconds before transmitting the next request.

If an ECU, after receiving a tester request, receives one or more additional tester requests before the response to the initial tester request is transmitted on the bus, the ECU shall either:

1) Ignore the 2nd tester request, and all additional tester requests until the response to the initial tester request is processed and transmitted.
2) Process all tester requests and send a positive response to each. Note that the requests shall be responded to in the order they were received.

An ECU shall never perform actions based upon a tester request without giving an appropriate response.

NOTE: Intermessage gap specifications are derived from module response capabilities rather than protocol latency attributes. Increase in protocol speed has little influence on intermessage gap requirements.

18 ISO-9141-Ford Serial Communications Link Protocol

18.1 Purpose / Scope

The Ford-defined ISO-9141-FORD serial communications link diagnostic protocol is described in this section. Explanations pertaining to each of the features of this protocol are provided, ranging from message construction to message timing.

NOTE: Some of the specifications & requirements relating to enhanced diagnostics on the diagnostic supported protocols may reside in other specifications and are referenced appropriately throughout this section.

18.2 ISO-9141-Ford Serial Communications Link

The ISO-9141-FORD serial communications link, which is a FORD interpretation of ISO-9141, is achieved over a single wire connection between the tester and each of the vehicle's ECUs via the DLC. The ISO-9141-FORD serial communications link protocol operates node-to-node using a master-slave relationship. The tester is the master that initiates all communications with the ECU(s) undergoing testing and responding to the tester as the slave

Tester and ECU physical layer information resides in the Ford ISO-9141 document titled *Ford-9141 Protocol Definition and Interface Requirements*[9].

NOTE: ISO-9141-FORD only permits tester-to-ECU communication; it does not support ECU-to-ECU communication.

18.3 ISO-9141-Ford Message Structure

Information is transferred in groupings of bits, transmitted one byte at a time. The preferred method of implementation employs a Universal Asynchronous Receiver Transmitter (UART) "clockless" technique for transferring bits on a single wire. "Bit banging" is permissible but the bit detection algorithm shall be equivalent to the synchronization and sampling technique commonly applied with UARTs. In accordance with this technique, the receiver shall sample the bus at 16 times the 10.4k bit rate to detect transition of the line from idle and qualify bits with center bit sampling.

All ISO-9141-FORD request and response messages are structured as shown in Table 18.1.

MESSAGE TYPE	***Byte 1***	
TARGET ADDRESS	***Byte 2***	
SOURCE ADDRESS	***Byte 3***	
Data Byte 1 / MODE	***Byte 4***	
Data Byte 2 / OPTIONAL		
Data Byte 3 / OPTIONAL		***0 < = N < = 6***
Data Byte 4 / OPTIONAL		***Optional Data Bytes***
Data Byte 5 / OPTIONAL		***May Be Used***
Data Byte 6 / OPTIONAL		
Data Byte 7 / OPTIONAL		
- - - - - - - - -		
Data Byte N / OPTIONAL		***Byte [4 - (N-6)]***
CHECKSUM		

Table 18.1 ISO-9141-FORD Message Format

The following subsections provide a brief byte-by-byte overview of the ISO-9141-FORD message structure.

18.3.1 ISO-9141-Ford Message Type Byte

The first byte of an ISO-9141-FORD message is the message type byte. The lower nibble identifies the message type. A value of $4 indicates that the message is physically addressed and a value of $8 indicates that the message is functionally addressed.

NOTE: Functionally addressed messages are not allowed.

The upper nibble of the first byte, specifies the length of the message. The length defines the number of bytes in the message that follow the message type byte including the checksum. Therefore, the value of the length nibble will be one less than the actual number of bytes that make up the message.

All ISO-9141-FORD diagnostic messages defined in “Appendix A” are physically addressed and have a type setting of $4.

NOTE: An ECU will ignore any ISO-9141-FORD message that has an invalid length or checksum.

18.3.2 ISO-9141-Ford Target Address Byte

Byte 2 of an ISO-9141-FORD message identifies the physical address of the target to which the message is being sent.

18.3.3 ISO-9141-Ford Source Address Byte

Byte 3 of an ISO-9141-FORD message identifies the physical address of the source from which the message originates.

18.3.4 ISO-9141-Ford Mode Byte

Byte 4 of an ISO-9141-FORD message is the mode byte. It uniquely identifies the purpose of the request or response message.

18.3.5 ISO-9141-Ford Optional Data Bytes

Following the mode byte are optional data bytes. The number of these optional data bytes (0 - 6) required per message and their meaning is specific to a particular message and are defined individually in each of the ISO-9141-FORD message descriptions.

18.3.6 ISO-9141-Ford Checksum Byte

The last byte of an ISO-9141-FORD message is the checksum which is used for message error detection. Its value is the least significant byte of the sum (with carry) of all of the bytes in the message excluding the checksum.

NOTE: An ECU will ignore any ISO-9141-FORD message that has an invalid length or checksum.

18.3.7 ISO-9141-Ford Message Timing Requirements

Boundary specifications are necessary to define timely and orderly flow of information on the communication bus. Maximum times are specified to provide a means to automatically terminate reception of aborted or faulty transmissions. Minimums are specified to prevent receiver overrun conditions.

18.3.8 Interbyte Gap

Messages are composed of sequentially transmitted bytes. For all messages, the period between the completion of the stop bit of one byte and the first edge of the start bit of the next byte is defined. All times specified below include receiver and transmitter tolerances:

1. **ECU TRANSMISSIONS: 0 ≤ INTERBYTE GAP TIME ≤ 20 milliseconds**

The ECU may send bytes end-to-end with no gaps. The ECU shall insure that bytes are not separated by more than 20 milliseconds. The ECU shall not transmit late if the byte cannot be sent on time. A maximum length message (11 bytes & 10 maximum gaps) requires approximately 210 milliseconds. The minimum length message (5 bytes and no gaps) requires approximately 5 milliseconds.

2. **ECU RECEPTION: 3 ≤ INTERBYTE GAP TIME ≤ 22 milliseconds**

The receiving ECU must receive bytes in the above range. Receiving ECU's shall time-out after 22 milliseconds and assume end-of-message or message abort. A receiving ECU may be capable of receiving messages with interbyte gaps of at least 3 milliseconds. An ECU may support PID $E210, which specifies an interbyte gap less than the 3 millisecond value specified above. PID $E210 is defined as two bytes:

- Byte 1: Minimum interbyte separation (0 - 255 ms)
- Byte 2: Tester intermessage time (0 - 255 ms)

3. **TESTER TRANSMISSIONS: 5 ≤ INTERBYTE GAP TIME ≤ 20 milliseconds**

The tester shall not send bytes end-to-end. In most cases, at least 5 milliseconds shall elapse between bytes to prevent overrun of the ECU receive buffer. However, modules may support PID $E210 which specifies the shortest interbyte gap the ECU may tolerate. The tester shall insure that bytes are not separated by more that 20 milliseconds. Interbyte gaps greater than 22 milliseconds will result in invalid messages. The tester shall not transmit late if the byte cannot be sent on time. A maximum length message (11 bytes & 10 maximum gaps) requires approximately 210 milliseconds. The minimum length message (5 bytes and 4 minimum gaps) requires approximately 25 milliseconds.

4. TESTER RECEPTION: 0 ≤ INTERBYTE GAP TIME ≤ 22 milliseconds

The tester shall be capable of receiving up to 11 bytes without an Interbyte gap. The tester receiver shall time-out after 22 milliseconds and assume end-of-message or message abort.

18.3.9 Intermessage Gap

Transactions are composed of messages that are sequentially transmitted. Specification of intermessage gap time sets the maximum and minimum transaction time with a 1 millisecond tolerance. The period between the completion of the last message and the start of the next message is dependent on the source of the messages. Limits are defined:

1. ECU RESPONSE FOLLOWING A TESTER REQUEST:

0 ≤ INTERMESSAGE GAP TIME ≤ 50 milliseconds

This condition occurs when the ECU responds to any tester query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if an expected response is not returned within 75 milliseconds (150% of 50 ms), the tester times out and may assume that none is forthcoming. The ECU shall respond to all tester requests as soon as the ECU's diagnostic response is ready.

2. ECU RESPONSE MESSAGE FOLLOWING ANOTHER ECU MESSAGE IN A SEQUENCE:

0 ≤ INTERMESSAGE GAP TIME ≤ 50 milliseconds

This condition occurs when an ECU is responding with multiple messages to a single query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if the message is not returned within 75 milliseconds (150% of 50 ms), the tester may time out and shall assume that no more messages will be sent. The time-out timer for the 75 milliseconds (150% of 50 ms) time-out period shall be reset after each message in the sequence form the ECU.

3. TESTER REQUEST FOLLOWING AN ECU RESPONSE:

55 ≤ INTERMESSAGE GAP TIME < 5,000 milliseconds

If a response message is not received within 75 milliseconds, the tester may time out and shall assume that no more messages will be sent. A new request may be transmitted by the tester 75 milliseconds (150% of 50 ms) after transmission of the original request.

The upper limit of 5,000 milliseconds is set to allow an ECU to detect when no tester is connected to the communication bus. Whenever a tester is connected, it is required to send a message to each ECU at this minimum rate to continue the diagnostic state dialogue. When an ECU times out, it shall reset and revert to a non-diagnostic state. The 55 millisecond minimum time allows the ECU processing time and prevents the tester from overrunning the ECU receive buffer. However, an ECU may support PID $E210, which specifies the shortest intermessage gap the ECU shall tolerate. A value of less than 55 milliseconds may be specified. Note that an ECU shall not implement any timers to ensure that the tester waits 55 milliseconds before transmitting the next request.

If an ECU, after receiving a tester request, receives one or more additional tester requests before the response to the initial tester request is transmitted on the bus, the ECU shall either:

1) Ignore the 2nd tester request, and all additional tester requests until the response to the initial tester request is processed and transmitted.
2) Process all tester requests and send a positive response to each. Note that the requests shall be responded to in the order they were received.

An ECU shall never perform actions based upon a tester request without giving an appropriate response.

NOTE: Intermessage gap specifications are derived from module response capabilities rather than protocol latency attributes. Increase in protocol speed has little influence on intermessage gap requirements.

19 Serial Communications Protocol (SCP)

19.1 Purpose / Scope

All Ford-defined diagnostic protocols are described in Sections 14 through 17 of this document. Explanations pertaining to each of the features of the protocols are provided, ranging from message construction to message timing.

NOTE: Some of the specifications & requirements relating to enhanced diagnostics on the diagnostic supported protocols may reside in other specifications and are referenced appropriately throughout this section.

19.2 Ford SCP Protocol Description

19.2.1 Overview

SCP is used in vehicle applications for inter-module communication to share vehicle operational information. Diagnostics are implemented on modules equipped with SCP communications, utilizing a protocol already there for other functional purposes. Refer to the following specifications for more information relating to SCP communications:

- SCP Vehicle Network Implementation Requirements[14]
- SCP Protocol Definition and Interface Requirements[15]

19.2.2 SCP Serial Communications Link

The SCP serial communications link, which is based on *Class B Data Communications Network Interface*[13], uses a twisted wire pair to provide a physical communications path between each of the vehicle's ECUs and the tester which is connected to the vehicle's DLC. The SCP serial communications link's protocol supports both ECU-to-ECU communication and tester-to-ECU communication.

Sections of "SCP Vehicle Network Implementation Requirements"[14] have the following information as follows:

- The length of the off-board extension of the SCP bus shall be limited to no more than the length specified in Section 5.3.
- Specification of the ground offset is noted in Section 5.9
- The interface circuitry for SCP network interfaces is to be found in Section 5.6.

19.2.3 SCP Message Structure

A SCP message is laid out as shown in Table 19.1.

All diagnostic SCP request and response messages are structured as shown in Table 19.1. The following subsections briefly define each of the bytes which make up a SCP diagnostic message.

PRIORITY/HEADER/TYPE	***Byte 1***	
TARGET ADDRESS	***Byte 2***	
SOURCE ADDRESS	***Byte 3***	
Data Byte 1 / MODE	***Byte 4***	
Data Byte 2 / OPTIONAL		
Data Byte 3 / OPTIONAL		***0 < = N < = 6***
Data Byte 4 / OPTIONAL		***Optional Data Bytes***
Data Byte 5 / OPTIONAL		***May Be Used***
Data Byte 6 / OPTIONAL		
Data Byte 7 / OPTIONAL		
CRC		***Byte [4 - (N-6)]***
TARGET'S ADDRESS		***Acknowledgement Byte***

Table 19.1 SCP Message Structure

All diagnostic messages (physically addressed) sent from a tester to ECU or from an ECU to tester shall have a type bit setting of [0100]. In addition, all diagnostic messages from an ECU to a tester shall have a priority of [110] (and therefore, the priority/header/type byte will always be $C4). An ECU shall be capable of responding to any properly formatted diagnostic request from a tester regardless of the priority level. The only supported diagnostic broadcast message for SCP is request no stored codes logging state, which has a type bit setting of [0001]. This broadcast format is mandatory for any ECU supporting the NSCL state. The SCP broadcast message for request NSCL state shall follow the format of all other diagnostic SCP requests, with the exception of the [0001] type bit setting, and the value of $5A used as the target address.

NOTE: An ECU shall not respond to a SCP diagnostic broadcast message.

19.2.3.1 SCP Target Address Byte

Byte 2 of a SCP message's header identifies the target destination of a message. For physically addressed messages, this byte contains the physical address of the target module. For functionally addressed messages, target byte contains the logical address (Primary ID) of the function to be executed.

19.2.3.2 SCP Source Address Byte

Header byte 3 of a SCP message identifies the physical address of the source from which a message originates.

19.2.3.3 SCP Data Byte 1

Byte 4 of a SCP diagnostic message is the mode byte and it uniquely identifies the purpose of the message; this byte is the secondary ID for a functional SCP message.

19.2.3.4 SCP Optional Data Bytes

Following byte 4 are up to six optional data bytes. The number of these bytes and their meaning is specific to each particular message and are defined individually in each message descriptions.

19.2.3.5 SCP Cyclic Redundancy Code (CRC) Byte

The last byte of a SCP message is the Cyclic Redundancy Code (CRC) byte which is used for message error detection. The algorithm for calculating the CRC can be found in the *Class B Data Communications Network Interface*[13] document.

19.2.3.6 SCP Acknowledgment Byte

The final byte shown in Figure 10 is the SCP acknowledgment byte. This byte is the physical address of the target and is returned by the target to acknowledge receipt of a message. The acknowledgement does not get transmitted by the source with the rest of the message.

NOTE: For simplicity, the acknowledgement byte will be omitted from the message definitions in this Appendix.

19.2.4 SCP Diagnostic Description

19.2.4.1 Diagnostic Message Characteristics

All module responses to diagnostic messages shall respond to the source module (usually tester) with it's request, meaning all SCP diagnostic response messages shall have the target address being the same as the source address of the request message.

19.2.5 SCP-CARB (California Air Resources Board) Capabilities

The capabilities of the Ford implementation of SCP-CARB are accomplished using only functionally addressed messages. The methodology is driven by the Society of Automotive Engineers in recommended practice *E/E Diagnostic Test Modes*[19]. The capabilities for SCP-CARB are as listed below:

- Parameter access capability
- Stored codes management capability
- Security access capability

SCP-CARB is intended to conform to government requirements.

19.2.6 Ford SCP Message Timing Requirements

Boundary specifications are necessary to define timely and orderly flow of information on the communication bus. Maximum times are specified to provide a means to automatically terminate reception of aborted or faulty transmissions. Minimum times are specified to prevent receiver overrun conditions.

19.2.7 Interbyte Gap

Messages are composed of bytes that are sequentially transmitted without interruption. Interbyte gaps are not allowed in SCP.

19.2.8 Intermessage Gap

Transactions are composed of messages that are sequentially transmitted. Specification of intermessage gap time sets the maximum and minimum transaction time with a 1 millisecond tolerance. The period

between the completion of the last message and the start of the next message is dependent on the source of the messages, limits are defined as follows:

1. ECU RESPONSE FOLLOWING A TESTER REQUEST:

$0 \leq$ INTERMESSAGE GAP TIME $\leq$ 50 milliseconds

This condition occurs when the ECU responds to any tester query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if an expected response is not returned within 75 milliseconds (150% of 50 ms), the tester times out and may assume that none is forthcoming. The ECU shall respond to all tester requests as soon as the ECU's diagnostic response is ready.

2. ECU RESPONSE MESSAGE FOLLOWING ANOTHER ECU MESSAGE IN A SEQUENCE:

$0 \leq$ INTERMESSAGE GAP TIME $\leq$ 50 milliseconds

This condition occurs when an ECU is responding with multiple messages to a single query. The tester must be able to receive messages back-to-back for an indefinite period. In addition, if each of the messages is not returned within 75 milliseconds (150% of 50 ms), the tester may time out and shall assume that no more messages will be sent. The timeout timer for the 75 milliseconds (150% of 50 ms) timeout period shall be reset after each message in the sequence from the ECU.

3. TESTER REQUEST FOLLOWING AN ECU RESPONSE:

$0 \leq$ INTERMESSAGE GAP TIME $<$ 5,000 milliseconds

If a response message is not received within 75 milliseconds, the tester may time out and shall assume that no more messages will be sent. A new request may be transmitted by the tester 75 milliseconds (150% of 50 ms) after transmission of the original request.

The upper limit of 5,000 milliseconds is set to allow an ECU to detect when no tester is connected to the communication bus. Whenever a tester is connected, it is required to send a message to each ECU at this minimum rate to continue the diagnostic state dialogue. When an ECU times out, it shall reset and revert to a non-diagnostic state. Once an ECU transmits a response, it shall be capable of immediately receiving a new tester request.

If an ECU, after receiving a tester request, receives one or more additional tester requests before the response to the initial tester request is transmitted on the bus, the ECU shall either:

1) Ignore the 2nd tester request, and all additional tester requests until the response to the initial tester request is processed and transmitted.
2) Process all tester requests and send a positive response to each. Note that the requests shall be responded to in the order they were received.

An ECU shall never perform actions based upon a tester request without giving an appropriate response.

NOTE: Intermessage gap specifications are derived from module response capabilities rather than protocol latency attributes. Increase in protocol speed has little influence on intermessage gap requirements.

Appendix A – Diagnostic Messages

A.1 Table Descriptions

The following tables describe all defined diagnostic messages and modes for every Ford supported protocol. Each table describes each diagnostic mode for all protocols.

NOTE: All values in this appendix are hexadecimal unless otherwise stated.

A.2 General Response [7F]

Byte-by-byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	7F	Mode Byte	General response
2	xx	Mode Byte of Request Msg (Data byte 1 of request msg)	Echo back the Mode Byte of the Request Message received
3	xx	Data Byte 2 of Request Msg	Echo back Data Byte 2 of Request Msg if Request Msg length > 4 bytes else 00
4	xx	Data Byte 3 of Request Msg	Echo back Data Byte 3 of Request Msg if Request Msg length > 5 bytes else 00
5	xx	Data Byte 4 of Request Msg	Echo back Data Byte 4 of Request Msg if Request Msg length > 6 bytes else 00
6	xx	Response Code	Identifies the Acceptance / Rejection of the Request (see definitions below)

Response Code (Data Byte 6) definitions:

CODE	DEFINITION	MEANING
00	AFFIRMATIVE	The Request received is supported by the module and all data was correct. The ECU executed the function as requested.
11	MODE NOT SUPPORTED	The ECU does not support the message (Mode) in any diagnostic state under any conditions.
12	SUB-FUNCTION NOT SUPPORTED, INVALID DATA OR INVALID FORMAT	The ECU supports the message, but arguments contained within the message are not supported, correct or in the proper format.
21	BUSY - REPEAT REQUEST	The ECU is currently Busy and won't process the request. Request needs to be re-sent.
22	CONDITIONS NOT CORRECT	The ECU could not service the request because one or more prerequisite conditions required for its support have not been satisfied (e.g., the mode is supported by the ECU but not in the current diagnostic state or the request is out of sequence). This response shall be used for all diagnostic requests in which the mode is never supported in a particular diagnostic state but is supported in others, even if the request is incorrectly formatted as would otherwise require a response code of $12.
33	SECURITY ACCESS DENIED	The requested action cannot be performed because the required level of security access hasn't been previously obtained.
35	INVALID KEY	Access was denied to enter secure diagnostic state due to an invalid key being submitted to the ECU.
72	TRANSFER ABORTED DUE TO ERROR	Block transfer operation was halted due to some fault (e.g., sequence or checksum error), and will not be completed later.
77	TRANSFER CHECKSUM ERROR	Error in block transfer due to disagreement between computed checksum and expected checksum.
78	REQUEST CORRECT RECEIVED – RESPONSE PENDING	This code can be used to indicate that the request message was properly received and does not need to be re-transmitted, but the response is pending and the ECU is not yet ready to receive another request.
79	INCORRECT BYTE COUNT DURING BLOCK TRANSFER	The number of bytes that was expected to be sent was not the same as the number of bytes received.

The following general response conditions shall be allowed as a response to a diagnostic request in any diagnostic state where a positive response (whether a $40 offset to the request such as for a mode $13 request or a general response $7F with a response code of $00 is used). Although the following responses may not appear in the valid responses table in this appendix for every mode and for every diagnostic state where the preceding condition is true, they shall be deemed as valid.

Mode $7F, Response Code $12 [Sub-Function Not Supported, Invalid Data or Invalid Format]

- This response shall be used when the data parameters in the diagnostic request are not supported by the ECU. For example, if a request for PID $D456 is received by the ECU, and the ECU does not support this specific PID.
- This response shall be used when the format of the diagnostic request does not conform to the format detailed in this document. Examples below:
    - The ECU is expecting only one data byte (e.g., the mode byte for mode $10) in a given diagnostic request and additional data bytes follow the mode byte.
    - The ECU is expecting two data bytes following the mode byte (e.g., the two byte PID for mode $22) and either more or less than this number of bytes follow the mode byte.

Mode $7F, Response Code $21 [Busy – Repeat Request]

- This response may be used when the ECU is busy performing some other task and cannot perform the action and thus reply within the time constraints for the given protocol. The tester shall be responsible for retransmitting the original request.

Mode $7F, Response Code $33 [Security Access Denied]

- This response shall be used when the diagnostic request is correctly formatted, yet the current level of security access previously obtained is insufficient to perform the request.

Mode $7F, Response Code $78 [Request Correctly Received – Response Pending]

- This response may be used when the diagnostic request is correctly formatted, but the action to be performed may not be completed yet. This shall indicate to the tester that the request message was properly received and does not need to be retransmitted, but the ECU is not yet ready to receive another request. This may be the case if the ECU does data processing or executes a command (e.g., Flash erase) which does not allow any attention to serial communication. The tester shall not repeat the request message after the reception of this response code. Response Code $78 is **only** approved for use in response to a Flash memory erase (diagnostic command $00B2) or in response to a write memory block (mode $3B) request. **Any other usage shall only be allowed if explicitly approved by EESE Core Network Communication.**
- As soon as the ECU has completed the task initiated by the request message it shall send the appropriate response based upon the original tester request message. The general response of $7F with response code $78 may be sent once or multiple times due to a single tester request if required by the ECU to prevent the tester from timing out. The time between each successive negative response from the ECU shall not exceed four seconds. The tester shall time out of waiting for an appropriate response if a response code $78 is not received at least every five seconds from the ECU, or if a valid positive response or general response (other than response code $78) is not received within one minute from the time of the original tester request.

NOTE: If a mode is never supported in a given diagnostic state for an ECU, the only valid general response codes are $11 (Mode Not Supported) and $22 (Conditions Not Correct) for that diagnostic state. Please refer to the previous table for appropriate usage.

A.3 Request Diagnostic State Entry [10]

Tester Request: Directs the ECU to enter diagnostic state

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	10	Mode Byte	Request Diagnostic State Entry

Valid ECU Responses to: Request Diagnostic State Entry			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its diagnostic state	7F	10	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	10	xx	xx	xx	21	
	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	10	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its diagnostic state	7F	10	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	10	xx	xx	xx	21	
	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	10	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its diagnostic state	7F	10	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	10	xx	xx	xx	21	
	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	10	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU is already in diagnostic state	7F	10	xx	xx	xx	00	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its diagnostic state	7F	10	xx	xx	xx	00	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	10	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	10	xx	xx	xx	22	

A.4 Request Diagnostic Freeze Frame Data [12]

Tester Request: Requests module freeze frame information

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	12	Mode Byte	Request freeze frame data
2	xx	Freeze frame number	Number indicating the type of frame requested
3	xx	High Byte DTC number	Most significant Byte of the DTC number requesting freeze frame on
4	xx	Low Byte DTC number	Least significant Byte of the DTC number requesting freeze frame on

Valid ECU Responses to: Request Freeze Frame Data			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report Freeze Frame Data [52]	- frame information will be provided	52	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	12	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	12	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	12	xx	xx	xx	21	
No Stored Codes Logging State	Report Freeze Frame Data [52]	- frame information will be provided	52	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	12	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	12	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	12	xx	xx	xx	21	
Input Integrity Test State	Report Freeze Frame Data [52]	- frame information will be provided	52	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	12	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	12	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy – try again	7F	12	xx	xx	xx	21	
Diagnostic State	Report Freeze Frame Data [52]	- frame information will be provided	52	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	12	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	12	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	12	xx	xx	xx	21	
Secure Diagnostic State	Report Freeze Frame Data [52]	- frame information will be provided	52	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	12	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	12	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	12	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	12	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	12	xx	xx	xx	22	

A.5 Report Diagnostic Freeze Frame Data [52]

ECU Response: Defines how enhanced freeze frame data is to be reported from an ECU.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	52	Mode Byte	Report freeze frame data
2	xx	Freeze frame number	Number indicating the type of frame requested
3	xx	High Byte DTC of request msg	High Byte DTC of requested freeze frame
4	xx	Low Byte DTC of request msg	Low Byte DTC of requested freeze frame
5	xx	Freeze frame data Byte 1	First data Byte of the freeze frame data returned from the ECU
6	xx	Optional freeze frame data Byte 2	Optional second data Byte of the freeze frame data returned from the ECU
7	xx	Optional freeze frame data Byte 3	Optional third data Byte of the freeze frame data returned from the ECU

A.6 Request stored codes [13] (Continuous DTCs)

Tester Request: Requests all of the continuous DTCs that may reside within an ECU.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	13	Mode Byte	Request stored codes

Valid ECU Responses to: Request Freeze Frame Data			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report Stored Codes [53]	- Reports all of the continuous DTCs residing within an ECU	53	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	13	xx	xx	xx	21	
No Stored Codes Logging State	Report Stored Codes [53]	- Reports all of the continuous DTCs residing within an ECU	53	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	13	xx	xx	xx	21	
Input Integrity Test State	Report Stored Codes [53]	- Reports all of the continuous DTCs residing within an ECU	53	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 21 (Busy)	- the ECU is busy – try again	7F	13	xx	xx	xx	21	
Diagnostic State	Report Stored Codes [53]	- Reports all of the continuous DTCs residing within an ECU	53	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	13	xx	xx	xx	21	
Secure Diagnostic State	Report Stored Codes [53]	- Reports all of the continuous DTCs residing within an ECU	53	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	13	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	13	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	13	xx	xx	xx	22	

A.7 Report Stored Codes [53]

ECU Response: Reports continuous DTCs stored in ECUs.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	53	Mode Byte	Report stored codes
2	xx or 00	1st DTC high Byte or Pad Byte	Most significant Byte
3	xx or 00	1st DTC low Byte or Pad Byte	Least significant Byte
4	xx or 00	2nd DTC high Byte or Pad Byte	Most significant Byte
5	xx or 00	2nd DTC low Byte or Pad Byte	Least significant Byte
6	xx or 00	3rd DTC high Byte or Pad Byte	Most significant Byte
7	xx or 00	3rd DTC low Byte or Pad Byte	Least significant Byte

NOTE: Send additional messages, each with 7 data bytes, if not all of the codes to be reported fit in the first message. Each mode $53 response must have 7 data bytes. The message shall be padded with zeros if necessary.

A.8 Request Clear Stored Codes [14]

Tester Request: Requests clearing of all of the continuous DTCs that may reside within an ECU.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	14	Mode Byte	Request clearing of continuous DTCs

Valid ECU Responses to: Request Clear Codes			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- continuous codes will be deleted	7F	14	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	14	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- continuous codes will be deleted	7F	14	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	14	xx	xx	xx	21	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- continuous codes will be deleted	7F	14	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	14	xx	xx	xx	21	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- continuous codes will be deleted	7F	14	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	14	xx	xx	xx	21	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- continuous codes will be deleted	7F	14	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	14	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	14	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	14	xx	xx	xx	22	

A.9 Request Operational State Entry [20]

Tester Request: Directs the ECU to enter operational state.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	20	Mode Byte	Request Operational State Entry

Valid ECU Responses to: Request Operational State Entry			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
Execution Routine State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
Information Transfer State	General Response [7F] of: 00 (Affirmative)	- the ECU will be placed into its Operational State	7F	20	xx	xx	xx	00	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	20	xx	xx	xx	21	

A.10 Request Parameter by PID [22]

Tester Request: Requests module parameter information by reading module Parameter Identifiers (PIDs)

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	22	Mode Byte	Request parameter by PID
2	xx	PID (MSB)	Most significant Byte of the logical PID address
3	xx	PID (LSB)	Least significant Byte of the logical PID address

Valid ECU Responses to: Request Parameter by PID			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
No Stored Codes Logging State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
Input Integrity Test State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
Diagnostic State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
Secure Diagnostic State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
Execution Routine State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	
Information Transfer State	Report Parameter by PID [62]	- returns the information identified by the logical PID address	62	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	22	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	22	xx	xx	xx	21	

A.11 Report Parameter by PID [62]

ECU Response: Reports information related to logical PID address memory locations.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	62	Mode Byte	Report parameter by PID
2	xx	MSB PID address	Most significant Byte of logical PID address of the request message
3	xx	LSB PID address	Least significant Byte of logical PID address of the request message
4	xx	PID data Byte 1	First data Byte of the PID data returned from the ECU
5	xx	Optional PID data Byte 2	Optional second data Byte of the PID data returned from the ECU
6	xx	Optional PID data Byte 3	Optional third data Byte of the PID data returned from the ECU
7	xx	Optional PID data Byte 4	Optional fourth data Byte of the PID data returned from the ECU

A.12 Request Parameter by DMR [23]

Tester Request: Requests module parameter information by reading module direct memory locations (DMRs)

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	23	Mode Byte	Request parameter by DMR
2	xx	Byte 3 of the memory address, MSB Byte	bits 31-24 (most significant Byte) of the 32-bit address
3	xx	Byte 2 of the memory address	bits 23-16 of the 32-bit address
4	xx	Byte 1 of the memory address	bits 15-8 of the 32-bit address
5	xx	Byte 0 of the memory address	bits 7-0 (least significant Byte) of the 32-bit address

Valid ECU Responses to: Request Parameter by DMR			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
No Stored Codes Logging State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
Input Integrity Test State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
Diagnostic State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
Secure Diagnostic State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
Execution Routine State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	
Information Transfer State	Report Parameter by DMR [63]	- returns the information identified by the DMR address	63	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	23	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	23	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	23	xx	xx	xx	21	

A.13 Report Parameter by DMR [63]

ECU Response: Reports information related to direct memory address locations.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	63	Mode Byte	Report parameter by DMR
2	xx	Byte 1 of the memory address	bits 8-15 of the 32-bit address, echoed from request message
3	xx	Byte 0 of the memory address	bits 0-7 (least significant Byte) of the 32-bit address, echoed from req. msg.
4	xx	DMR data Byte from DMR address	First data Byte of the DMR data returned from the ECU
5	xx	DMR data Byte from DMR address + 1	Second data Byte of the DMR data returned from the ECU
6	xx	DMR data Byte from DMR address + 2	Third data Byte of the DMR data returned from the ECU
7	xx	DMR data Byte from DMR address + 3	Forth data Byte of the DMR data returned from the ECU

A.14 Request Parameter Scaling/Masking [24]

Tester Request: Requests scaling or masking information relating to a supported PID

Byte-by-Byte message description for the data bytes of the message:

DATA Byte	VALUE	DEFINITION	COMMENTS
1	24	Mode Byte	Request parameter scaling/masking
2	xx	Offset ($FF if N/A)	Offset Parameter
3	xx	High Byte PID reference	MSB of the PID reference number
4	xx	Low Byte PID reference	LSB of the PID reference number

Valid ECU Responses to: Request Parameter Scaling/Masking			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
No Stored Codes Logging State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
Input Integrity Test State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
Diagnostic State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
Secure Diagnostic State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
Execution Routine State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	
Information Transfer State	Report Parameter by Scaling/Masking [64]	- returns parameter scaling/masking information	64	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	24	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	24	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	24	xx	xx	xx	21	

A.15 Report Parameter Scaling/Masking [64]

ECU Response: Reports parameter scaling/masking information.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	64	Mode Byte	Report parameter scaling/masking
2	xx	Offset	Offset parameter
3	xx	High Byte PID reference	MSB of PID number
4	xx	Low Byte PID reference	LSB of PID number
5	xx	Parameter information Byte	Specifies the parameter encoding type and number of bytes in parameter
6	xx	PID scaling / mask 1	Signed and unsigned scaling byte or bit packed mask byte. Set to $00 if scaling not supported for data type
7	xx	Mask 2	Used if more than 8 bit mapped data bits are associated with the parameter. All bits not used for masking are to be padded with $00

A.16 Stop Transmitting Requested Data [25]

Tester Request: Requests the termination of any ongoing repetitive data transfers (i.e. rapid data).

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	25	Mode Byte	Stop transmitting requested data

Valid ECU Responses to: Stop Transmitting Requested Data			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	
Information Transfer State	General Response [7F] of: 00 (Affirmative)	- the ECU will terminate all repetitive data transfers	7F	25	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	25	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	25	xx	xx	xx	21	

A.17 Request Security Access [27]

Tester Request: Directs an ECU to return a security seed code or to grant or deny access based on a security key computed by the tester from a seed code supplied by an ECU.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	27	Mode byte	Request security access
2	xx	Request seed or submit key	Data Byte 2 = $(2X-1) for seed request or Data Byte 2 = $2X for key submission where X is the level of security desired. Odd numbers denote seed requests and even numbers denote key submission requests.
3	xx	Security key MSB or pad byte of $00	If data byte 2 is >0 and even (key submittal) this value is the most significant byte of security key. If data byte 2 is >0 and odd (seed request) this value is a pad byte of $00.
4	xx	Security key byte #2 or pad byte of $00	If data byte 2 is >0 and even (key submittal) this value is the second byte of security key. If data byte 2 is >0 and odd (seed request) this value is a pad byte of $00.
5	xx	Security key LSB or pad byte of $00	If data byte 2 is >0 and even (key submittal) this value is the least significant byte of security key. If data byte 2 is >0 and odd (seed request) this value is a pad byte of $00.

Note: Refer to Figure 9.1 for additional details relating to the security access procedure.

Valid ECU Responses to: Request Security Access			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	27	xx	xx	xx	22	
No Stored Codes Logging State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx		
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	
Input Integrity Test State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx	xx	
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	
Diagnostic State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx		
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	
Secure Diagnostic State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx		
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	
Execution Routine State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx		
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	
Information Transfer State	Report security access seed [67]	- responds with either a security seed or security status information	67	xx	xx	xx	xx		
	General Response [7F] of: 00 (Affirmative)	- security access granted for level of security requested	7F	27	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	27	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	27	xx	xx	xx	12	
	General Response [7F] of: 35 (Invalid key)	- security access denied due to reception of an invalid key	7F	27	xx	xx	xx	35	

A.18 Report Security Seed [67]

ECU Response: Reports the security seed

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	67	Mode Byte	Report security seed
2	xx	Security level	Level of security access = $2X-1
3	xx	High byte seed	MSB of the security seed
4	xx	Second byte seed	Second byte of the security seed
5	xx	Low byte seed	LSB of the security seed

A.19 Request Disable Normal Message Transmission [28]

Tester Request: Requests the ECU to disable normal message transmission (non-diagnostic messages) on the link.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	28	Mode Byte	Request Disable Normal Message Transmission

Valid ECU Responses to: Request Disable Normal Message Transmission			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	28	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 00 (affirmative)	- the command will be performed by the ECU	7F	28	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	28	xx	xx	xx	12	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	28	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	28	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- some condition within the ECU isn't correct for this operation to be performed	7F	28	xx	xx	xx	22	
Secure Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- some condition within the ECU isn't correct for this operation to be performed	7F	28	xx	xx	xx	22	
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	28	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	28	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	28	xx	xx	xx	22	

A.20 Request Diagnostic Data Packets [2A]

Tester Request: Requests data packets in the form of either PIDs or DMRs

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	2A	Mode Byte	Request diagnostic data packet(s)
2	xx	Data Rate / Clearing	00 = Stop sending data 01 = Send 1 response 02 = Repeat continuously at slow rate 03 = Repeat continuously at medium 04 = Repeat continuously at fast rate 0A = Stops sending and clears the DPID
3	xx	Data Packet Identification (DPID) Number 1	Identification number of the first or only data packet requested by the tester
4	xx	DPID Number 2	Used only if more than 1 DPID is defined
5	xx	DPID Number 3	Used only if more than 2 DPIDs are defined
6	xx	DPID Number 4	Used only if more than 3 DPIDs are defined
7	xx	DPID Number 5	Used only if more than 4 DPIDs are defined

Valid ECU Responses to: Request Diagnostic Data Packets			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
No Stored Codes Logging State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
Input Integrity Test State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
Diagnostic State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
Secure Diagnostic State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
Execution Routine State	Report data packet [6A]	- each DPID to be reported will have its own mode $6A response	6A	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode not supported)	- the ECU doesn't support this mode	7F	2A	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2A	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2A	xx	xx	xx	21	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	2A	xx	xx	xx	22	

A.21 Report Data Packet [6A]

ECU Response: Reports information related to Data Packet Identification parameter(s) [DPID(s)].

Byte-by-Byte message description for the data bytes of the message. Choose the correct table based upon the value of data Byte 2 of the report message:

If data byte 2 is $00, then a request to stop DPIDs was received and the first table is applicable.

If data byte 2 is not $00, then a request for DPID reporting was received, and the second table is applicable.

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	6A	Mode Byte	Report data packet
2	xx	Report DPID stopping information	Value of $00 indicates request was issued to stop DPIDs
3	xx	DPID stopped	DPID stopped from periodic transmissions
4	xx	Optional 2nd DPID commanded to be stopped	Optional 2nd DPID stopped from periodic transmissions
5	xx	Optional 3rd DPID commanded to be stopped	Optional 3rd DPID stopped from periodic transmissions
6	xx	Optional 4th DPID commanded to be stopped	Optional 4th DPID stopped from periodic transmissions
7	xx	Optional 5th DPID commanded to be stopped	Optional 5th DPID stopped from periodic transmissions

DATA Byte	VALUE	DEFINITION	COMMENTS
1	6A	Mode Byte	Report data packet
2	xx	DPID Number	DPID number the following data bytes apply to
3	xx	DPID data Byte 1	First data Byte of the DPID data returned from the ECU
4	xx	Optional DPID data Byte 2	Optional second data Byte of the DPID data returned from the ECU
5	xx	Optional DPID data Byte 3	Optional third data Byte of the DPID data returned from the ECU
6	xx	Optional DPID data Byte 4	Optional fourth data Byte of the DPID data returned from the ECU
7	xx	Optional DPID data Byte 5 (DMR only)	Optional fifth data Byte of the DPID data returned from the ECU (DMR only

A.22 Dynamically Define Diagnostic Data Packet [2C]

Tester Request: Requests definition of diagnostic data packets

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	2C	Mode Byte	Dynamically define diagnostic data packet
2	xx	Data Packet Identification (DPID) Number	The desired DPID number. Must be greater than or equal than one.
3	xx	DPID Format Byte	Bits 7,6 are defined as follows: **ONLY PIDs AND DMRs ARE ALLOWED TO BE DEFINED WITH THIS MODE** - 00: Reserved - 01: Define by PID (2 Byte PID numbers are used) - 10: Define by DMR (4 Byte DMR addresses are used) - 11: Reserved Bits 5,4 and 3 are defined as follows: - Position of defined DPID parameter in the message frame where 001 represents the data Byte position after the DPID #. Bits 2,1 and 0 are defined as follows: - The number of data bytes for this parameter (i.e. the data length of the DPID being defined). NOTE: If this entire Byte is set to zero, then the DPID shall be cleared and erased out of module memory.
4	xx	Reference Number 1	Identifies the first Byte of the number of the item being added to the DPID (i.e., MSB PID #, DMR address high byte or other)
5	xx	Reference Number 2	Identifies the second Byte of the number of the item being added to the DPID, if it exists (i.e., LSB PID #, DMR address or other).
6	xx	Reference Number 3	Identifies the third Byte of the number of the item being added to the DPID, if it exists (i.e., DMR address or other).
7	xx	Reference Number 4	Identifies the forth Byte of the number of the item being added to the DPID, if it exists (i.e., DMR address low byte or other).

NOTE: All DPIDs must be greater than or equal to 1.

Valid ECU Responses to: Dynamically Define Diagnostic Data Packet			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (affirmative)	- the request for addition of an item to a DPID was successful	7F	2C	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2C	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	2C	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 00 (affirmative)	- the request for addition of an item to a DPID was successful	7F	2C	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2C	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	2C	xx	xx	xx	21	
Input Integrity Test State	General Response [7F] of: 00 (affirmative)	- the request for addition of an item to a DPID was successful	7F	2C	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2C	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	2C	xx	xx	xx	21	
Diagnostic State	General Response [7F] of: 00 (affirmative)	- the request for addition of an item to a DPID was successful	7F	2C	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2C	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	2C	xx	xx	xx	21	
Secure Diagnostic State	General Response [7F] of: 00 (affirmative)	- the request for addition of an item to a DPID was successful	7F	2C	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2C	xx	xx	xx	11	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	2C	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform DPID addition during ERS	7F	2C	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform DPID addition during ITS	7F	2C	xx	xx	xx	22	

A.23 Input Output Control by PID [2F]

Tester Request: Requests the update of a Parameter Identifiers (PIDs) value

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	2F	Mode Byte	Input/Output Control By PID
2	xx	PID # (MSB)	Most significant Byte of the logical PID address
3	xx	PID # (LSB)	Least significant Byte of the logical PID address
4	xx	1st byte of PID to write to	Byte 1 of the PID that is written to
5	xx	Optional 2nd PID byte to write to	2nd PID byte to write to
6	xx	Optional 3rd PID byte to write to	3rd PID byte to write to
7	xx	Optional 4th PID byte to write to	4th PID byte to write to

Valid ECU Responses to: Input/Output Control by PID			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the PID control parameter has been received	7F	2F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2F	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2F	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- the PID control parameter has been received	7F	2F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2F	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2F	xx	xx	xx	21	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the PID control parameter has been received	7F	2F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2F	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2F	xx	xx	xx	21	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the PID control parameter has been received	7F	2F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2F	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2F	xx	xx	xx	21	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the PID control parameter has been received	7F	2F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	2F	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is busy with normal operation – try again	7F	2F	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this action during ERS	7F	2F	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	2F	xx	xx	xx	11	
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this action during ITS	7F	2F	xx	xx	xx	22	

A.24 Request Diagnostic Routine Entry [31]

Tester Request: Directs the ECU to execute a diagnostic execution routine

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	31	Mode Byte	Request diagnostic routine entry
2	xx	Execution routine number	The ID number of the execution routine the ECU is instructed to execute

Valid ECU Responses to: Request Diagnostic Routine Entry			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	31	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	31	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	31	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the execution routine has been initiated	7F	31	xx	xx	xx	00	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	31	xx	xx	xx	12	
	General Response [7F] of: 22 (Conditions Not Correct)	- some condition hasn't been met to enter the execution routine	7F	31	xx	xx	xx	22	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the execution routine has been initiated	7F	31	xx	xx	xx	00	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	31	xx	xx	xx	12	
	General Response [7F] of: 22 (Conditions Not Correct)	- some condition hasn't been met to enter the execution routine	7F	31	xx	xx	xx	22	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	31	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	31	xx	xx	xx	22	

A.25 Request Diagnostic Routine Exit [32]

Tester Request: Directs the ECU to exit a diagnostic execution routine

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	32	Mode Byte	Request diagnostic routine exit
2	xx	Execution routine number to exit	The ID number of the execution routine the ECU is instructed to exit
3	xx	Type of exit	$00 – exit if complete, $01 – exit immediately

Valid ECU Responses to: Request Diagnostic Routine Exit			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	
Secure Diagnostic State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	
Execution Routine State	General Response [7F] of: 00 (Affirmative)	- the execution routine has been exited	7F	32	xx	xx	xx	00	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	32	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy) - only for conditional exit	- the ECU is Busy - try again	7F	32	xx	xx	xx	21	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state	7F	32	xx	xx	xx	22	

A.26 Request Diagnostic Routine Results [33] (On-demand DTCs)

Tester Request: Requests all of the on-demand DTCs that may reside within an ECU for a particular execution routine

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	33	Mode Byte	Request diagnostic routine results
2	xx	Execution routine number of requested results	Report the diagnostic on-demand DTCs associated with this execution number

Valid ECU Responses to: Request Diagnostic Routine Results			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	33	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	33	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	33	xx	xx	xx	22	
Diagnostic State	Report Diagnostic Routine Results [73]	- Reports all of the on-demand DTCs residing within an ECU for a particular execution routine.	73	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Sub-function not supported)	- the Execution Routine number specified isn't supported.	7F	33	xx	xx	xx	12	
	General Response [7F] of: 22 (Conditions Not Correct)	- the Execution Routine number is supported but wasn't either the last execution routine ran or no execution routine was performed during the current Diagnostic session.	7F	33	xx	xx	xx	22	
Secure Diagnostic State	Report Diagnostic Routine Results [73]	- Reports all of the on-demand DTCs residing within an ECU for a particular execution routine.	73	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 12 (Sub-function not supported)	- the Execution Routine number specified isn't supported.	7F	33	xx	xx	xx	12	
	General Response [7F] of: 22 (Conditions Not Correct)	- the Execution Routine number is supported but wasn't either the last execution routine ran or no execution routine was performed during the current Diagnostic session.	7F	33	xx	xx	xx	22	
Execution Routine State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	33	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 22 (Conditions Not Correct)	- not all diagnostic state entry conditions have been satisfied	7F	33	xx	xx	xx	22	

A.27 Report Diagnostic Routine Results [73]

ECU Response: Reports on-demand DTCs associated with execution routines

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	73	Mode Byte	Report diagnostic routine results
2	xx or 00	1st on-demand DTC high Byte	Most significant Byte
3	xx or 00	1st on-demand DTC low Byte	Least significant Byte
4	xx or 00	2nd on-demand DTC high Byte	Most significant Byte
5	xx or 00	2nd on-demand DTC low Byte	Least significant Byte
6	xx or 00	3rd on-demand DTC high Byte	Most significant Byte
7	xx or 00	3rd on-demand DTC low Byte	Least significant Byte

NOTE: Send additional messages, each with 7 data bytes, if not all of the codes to be reported fit in the first message. Each mode $73 response must have 7 data bytes. The message shall be padded with zeros if necessary.

A.28 Request Download Routine Entry [34]

Tester Request: Directs the ECU initiate a memory block download to an ECU

Byte-by-Byte message description for the data bytes of the message. Choose the correct table based upon the addressing size supported within the ECU:

Table specifying a 3-Byte ECU address for a download operation

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	34	Mode Byte	Request download routine entry
2	xx	Block transfer format	Specifies the format of the data to be transferred (i.e., handshake and no handshake), byte count multiplier and the size of the address. Refer to the "Module Programming and Configuration Specification"[8] for detailed information relating to the format Byte.
3	xx	Byte count - MSB	Multiplied by the Byte count multiplier in the format Byte to formulate the # of bytes to download - most significant Byte.
4	xx	Byte count - LSB	Multiplied by the Byte count multiplier in the format Byte to formulate the # of bytes to download - Least significant Byte.
5	xx	MSB of the ECU's address – Byte 2	Most significant Byte of the ECU's address to where the block is to be downloaded
6	xx	ECU's address – Byte 1	Byte 2 of the ECU's address to where the block is to be downloaded
7	xx	LSB of the ECU's address – Byte 0	Least significant Byte of the ECU's address to where the block is to be downloaded

Table specifying a 4-Byte ECU address for a download operation

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	34	Mode Byte	Request download routine entry
2	xx	Block transfer format	Specifies the format of the data to be transferred (i.e. handshake and no handshake), byte count multiplier and the size of the address. Refer to the "Module Programming and Configuration Specification"[8] for detailed information relating to the format Byte.
3	xx	Byte count	Multiplied by the Byte count multiplier in the format Byte to formulate the # of bytes to download
4	xx	MSB of the ECU's address – Byte 3	Most significant Byte of the ECU's address to where the block is to be downloaded
5	xx	Byte 2	Byte 2 of the ECU's address to where the block is to be downloaded
6	xx	Byte 1	Byte 1 of the ECU's address to where the block is to be downloaded
7	xx	LSB of the ECU's address – Byte 0	Least significant Byte of the ECU's address to where the block is to be downloaded

NOTE: Refer to the *Module Programming and Configuration Specification[8]* for more information.

Valid ECU Responses to: Request Download Routine Entry:			RESPONSE MESSAGE DATA BYTES					
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- Cannot perform download in this state	7F	34	xx	xx	xx	22
No Stored Codes Logging State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- Cannot perform download in this state	7F	34	xx	xx	xx	22
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- Cannot perform download in this state	7F	34	xx	xx	xx	22
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU is ready to receive data	7F	34	xx	xx	xx	00
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 12 (Subfunction Not Supported)	- some of the data in the request was out of range or not supported	7F	34	xx	xx	xx	12
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	34	xx	xx	xx	33
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU is ready to receive data	7F	34	xx	xx	xx	00
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	34	xx	xx	xx	12
	General Response [7F] of: 33 (Security access denied)	- the requested action cannot be performed due to lack of security access	7F	34	xx	xx	xx	33
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- Cannot perform download in this state	7F	34	xx	xx	xx	22
Information Transfer State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	34	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- Cannot perform download in this state	7F	34	xx	xx	xx	22

A.29 Request Upload Routine Entry [35]

Tester Request: Directs the ECU to initiate a memory block upload from an ECU to the tester

Byte-by-Byte message description for the data bytes of the message. Choose the correct table based upon the addressing size supported within the ECU:

Table specifying a 3-Byte ECU address for an upload operation

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	35	Mode Byte	Request upload routine entry
2	xx	Block transfer format	Specifies the format of the data to be transferred (i.e. handshake and no handshake), byte count multiplier and the size of the address. Refer to the *Module Programming and Configuration Specification*[8] for detailed information relating to the format byte.
3	xx	Block size - MSB	Multiplied by the Byte count multiplier in the format byte to formulate the # of bytes to download - most significant Byte.
4	xx	Block size - LSB	Multiplied by the Byte count multiplier in the format byte to formulate the # of bytes to download - least significant Byte.
5	xx	MSB of the ECU's address – Byte 2	Most significant Byte of the ECU's address to where the block is to be uploaded from
6	xx	ECU's address – Byte 1	Byte 2 of the ECU's address to where the block is to be uploaded from
7	xx	LSB of the ECU's address – Byte 0	Least significant Byte of the ECU's address to where the block is to be uploaded from

Table specifying a 4-Byte ECU address for an upload operation

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	35	Mode Byte	Request upload routine entry
2	xx	Block transfer format	Specifies the format of the data to be transferred (i.e. handshake and no handshake), byte count multiplier and the size of the address. Refer to the *Module Programming and Configuration Specification*[8] for detailed information relating to the format byte.
3	xx	Block size to upload	Multiplied by the Byte count multiplier in the format byte to formulate the # of bytes to download
4	xx	MSB of the ECU's address – Byte 3	Most significant Byte of the ECU's address to where the block is to be uploaded from
5	xx	ECU's address – Byte 2	Byte 2 of the ECU's address to where the block is to be uploaded from
6	xx	ECU's address – Byte 1	Byte 1 of the ECU's address to where the block is to be uploaded from
7	xx	LSB of the ECU's address – Byte 0	Least significant Byte of the ECU's address to where the block is to be uploaded from

Valid ECU Responses to: Request Upload Routine Entry:			RESPONSE MESSAGE DATA BYTES					
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state of operation	7F	35	xx	xx	xx	22
No Stored Codes Logging State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state of operation	7F	35	xx	xx	xx	22
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state of operation	7F	35	xx	xx	xx	22
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- ECU is ready to transfer data	7F	35	xx	xx	xx	00
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	35	xx	xx	xx	12
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	35	xx	xx	xx	33
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU is ready to receive data	7F	35	xx	xx	xx	00
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	35	xx	xx	xx	12
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	35	xx	xx	xx	33
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state of operation	7F	35	xx	xx	xx	22
Information Transfer State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	35	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform the requested action in this state of operation	7F	35	xx	xx	xx	22

A.30 Block Data Transfer Message [36]

NOTE: This message is used for data messages that are transmitted and received from testers and ECUs

Tester message: Contains data to be downloaded from the tester to the ECU

ECU message: Contains data to be uploaded from the ECU to the tester

Byte-by-Byte message description for the data bytes of the message.

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	36	Mode Byte	Block data transfer message
2	xx	Data Byte 1	1st Byte of data
3	xx or 00	Data Byte 2	2nd Byte of data or zero padding
4	xx or 00	Data Byte 3	3rd Byte of data or zero padding
5	xx or 00	Data Byte 4	4th Byte of data or zero padding
6	xx or 00	Data Byte 5	5th Byte of data or zero padding
7	xx or 00	Data Byte 6	6th Byte of data or zero padding

Valid ECU Responses to: Block Data Transfer Message			RESPONSE MESSAGE DATA BYTES					
CURRENT ECU STATE/MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
No Stored Codes Logging State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
Secure Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	36	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	36	xx	xx	xx	22
Information Transfer State	General Response [7F] of: 00 (Affirmative)	- acknowledging that the message was received. Used for handshake mode.	7F	36	xx	xx	xx	00
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	36	xx	xx	xx	12
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	36	xx	xx	xx	21
	General Response [7F] of: 79 (Incorrect Byte count during block transfer)	- and error occurred due to an incorrect number of bytes transferred	7F	36	xx	xx	xx	79

A.31 Request Block Transfer Exit [37]

Tester Request: Directs the ECU to exit information transfer state

Byte-by-Byte message description for the data bytes of the message.

DATA Byte	VALUE	DEFINITION	Comments
1	37	Mode Byte	Request block transfer exit

Valid ECU Responses to: ***Request Block Transfer Exit***			RESPONSE MESSAGE DATA Byte					
CURRENT ECU STATE/MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
No Stored Codes Logging State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
Secure Diagnostic State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	37	xx	xx	xx	11
	General Response [7F] of: 22 (Conditions Not Correct)	- cannot perform this operation in this state	7F	37	xx	xx	xx	22
Information Transfer State	General Response [7F] of: 00 (Affirmative)	- Information transfer state will be exited	7F	37	xx	xx	xx	00
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	37	xx	xx	xx	12
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	37	xx	xx	xx	21
	General Response [7F] of: 77 (Transfer Checksum Error)	- an error occurred due to an incorrect checksum	7F	37	xx	xx	xx	21
	General Response [7F] of: 79 (Incorrect Byte count during block transfer)	- an error occurred due to an incorrect number of bytes transferred	7F	37	xx	xx	xx	79

A.32 Request Write Memory Block [3B]

Tester Request: Requests writing to a logical block of memory within an ECU. Used for Block (Method 2) configuration, see the *Module Programming and Configuration Specification*[8] for more details.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	3B	Mode Byte	Request write memory block
2	xx	Block number	Pointer to a logical ECU address location
3	xx	Data Byte 1	1st Byte of data to be written
4	xx or 00	Optional Data Byte 2 or 00 pad (all protocols)	2nd Byte of data to be written
5	xx or 00	Optional Data Byte 3 or 00 pad (all protocols)	3rd Byte of data to be written
6	xx or 00	Optional Data Byte 4 or 00 pad (all protocols)	4th Byte of data to be written
7	xx or 00	Optional Data Byte 5 or 00 pad (all protocols)	5th Byte of data to be written

Valid ECU Responses to: Request Write Memory Block			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3B	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
No Stored Codes Logging State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3B	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
Input Integrity Test State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3B	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
Diagnostic State	General Response [7F] of: 00 (affirmative)	- correct data has been received and will be written to memory	7F	3B	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	3B	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	3B	xx	xx	xx	21	
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	3B	xx	xx	xx	33	
Secure Diagnostic State	General Response [7F] of: 00 (affirmative)	- correct data has been received and will be written to memory	7F	3B	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	3B	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	3B	xx	xx	xx	21	
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	3B	xx	xx	xx	33	
Execution Routine State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3B	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	
Information Transfer State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3B	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3B	xx	xx	xx	11	

A.33 Request Read Memory Block [3C]

Tester Request: Requests reading a logical block of memory within an ECU. Used for Block (Method 2) configuration, see the *Module Programming and Configuration Specification*[8] for more details.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	3C	Mode Byte	Request read memory block
2	xx	Block number	Pointer to a logical ECU address location

Valid ECU Responses to: Request Read Memory Block			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3C	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
No Stored Codes Logging State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3C	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
Input Integrity Test State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3C	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
Diagnostic State	Report contents of memory block [7C]	- report the 5 bytes of memory associated with the logical block	7C	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	3C	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	3C	xx	xx	xx	21	
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	3C	xx	xx	xx	33	
Secure Diagnostic State	Report contents of memory block [7C]	- report the 5 bytes of memory associated with the logical block	7C	xx	xx	xx	xx	xx	xx
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	3C	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	3C	xx	xx	xx	21	
	General Response [7F] of: 33 (Security Access Denied)	- the requested action cannot be performed due to lack of security access	7F	3C	xx	xx	xx	33	
Execution Routine State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3C	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	
Information Transfer State	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	3C	xx	xx	xx	22	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	3C	xx	xx	xx	11	

A.34 Report Contents of Memory Block [7C]

ECU Response: Reports the five bytes of data associated with the block requested

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	7C	Mode Byte	Report contents of memory block
2	xx	Block number	Block number requested
3	xx	Block data Byte 1	1st Byte of the requested logical block number
4	xx or 00	Block data Byte 2 or Pad Byte	2nd Byte of the requested logical block number
5	xx or 00	Block data Byte 3 or Pad Byte	3rd Byte of the requested logical block number
6	xx or 00	Block data Byte 4 or Pad Byte	4th Byte of the requested logical block number
7	xx or 00	Block data Byte 5 or Pad Byte	5th Byte of the requested logical block number

A.35 Tester Present [3F]

Tester Request: Used to reset the diagnostic session timer

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	3F	Mode Byte	Tester present

Valid ECU Responses to: Report Stored Codes			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall perform no actions.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
No Stored Codes Logging State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
Execution Routine State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	
Information Transfer State	General Response [7F] of: 00 (Affirmative)	- the ECU acknowledges that a tester is present and shall reset its state timer.	7F	3F	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- this mode isn't supported by the ECU	7F	3F	xx	xx	xx	11	

A.36 Request No Stored Codes Logging State [B0]

Tester Request: Request entry into no stored codes logging state to suspend logging of ALL continuous DTCs

Byte-by-Byte message description for the data bytes of the message:

SCP: Only the functionally addressed (broadcast – type bit = [0001]) request no stored codes logging message shall be used.

The format for the SCP functional request is as follows. Note that no response is allowed to this SCP diagnostic broadcast message.

Priority/Header	PPP00001b	b represents binary and PPP represents the priority level
Target Address	$5A	Diagnostic functional address
Source Address	$XX	Tester Address
Mode Byte	$B0	Request No Stored Codes Logging State

UBP: Only the functionally addressed request no stored codes logging message shall be used.

The format for the UBP functional request is as follows. Note that no response is allowed to this UBP functional diagnostic message.

Format	$02	Format message type and length
Primary ID	$B0	Diagnostic functional address
Source Address	$XX	Tester Address
Operation	$04	Operation Byte
Secondary ID	$B0	Corresponding to request No Stored Codes Logging State

A.37 Diagnostic Command [B1]

Tester Request: Initiates diagnostic commands to perform internal ECU operations

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	B1	Mode Byte	Diagnostic command
2	xx	Diagnostic command MSB	MSB of the diagnostic command number
3	xx	Diagnostic command LSB	LSB of the diagnostic command number
4	xx	Command data Byte 1	To be used as defined in the MRDB and SSDS
5	xx	Command data Byte 2	To be used as defined in the MRDB and SSDS
6	xx	Command data Byte 3	To be used as defined in the MRDB and SSDS
7	xx	Command data Byte 4	To be used as defined in the MRDB and SSDS

Valid ECU Responses to: Diagnostic Command			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B1	xx	xx	xx	22	
No Stored Codes Logging State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B1	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B1	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 00 (affirmative)	- the command will be performed by the ECU	7F	B1	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B1	xx	xx	xx	12	
	General Response [7F] of: 22 (conditions not correct)	- some condition within the ECU isn't correct for this operation to be performed	7F	B1	xx	xx	xx	22	
Secure Diagnostic State	General Response [7F] of: 00 (affirmative)	- the command will be performed by the ECU	7F	B1	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B1	xx	xx	xx	12	
	General Response [7F] of: 22 (conditions not correct)	- some condition within the ECU isn't correct for this operation to be performed	7F	B1	xx	xx	xx	22	
Execution Routine State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B1	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 11 (Mode Not Supported)	- the ECU does not support this Mode (message)	7F	B1	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B1	xx	xx	xx	22	

A.38 Request Input Integrity Test State [B2]

Tester Request: Request entry into the input integrity test state to allow additional input and output parameter control

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	B2	Mode Byte	Request input integrity test state

Valid ECU Responses to: Request input integrity test state			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B2	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B2	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	B2	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B2	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 00 (Affirmative)	- the ECU is already in input integrity test state	7F	B2	xx	xx	xx	00	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B2	xx	xx	xx	12	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B2	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B2	xx	xx	xx	12	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B2	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B2	xx	xx	xx	12	
Execution Routine State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B2	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B2	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B2	xx	xx	xx	22	

A.39 Request Manufacturer State Entry [B4]

Tester Request: Request entry into the manufacturer state to allow a non-standard state of operation when required in manufacturing or service.

Byte-by-Byte message description for the data bytes of the message:

DATA BYTE	VALUE	DEFINITION	COMMENTS
1	B4	Mode Byte	Request manufacturer state

Valid ECU Responses to: Request manufacturer state			RESPONSE MESSAGE DATA BYTES						
CURRENT ECU STATE / MODE	VALID RESPONSES	- REASON FOR THE RESPONSE	1	2	3	4	5	6	7
Operational State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B4	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B4	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	B4	xx	xx	xx	21	
No Stored Codes Logging State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B4	xx	xx	xx	22	
Input Integrity Test State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B4	xx	xx	xx	22	
Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B4	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B4	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	B4	xx	xx	xx	21	
Secure Diagnostic State	General Response [7F] of: 00 (Affirmative)	- the ECU will enter input integrity test state	7F	B4	xx	xx	xx	00	
	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 12 (Subfunction not supported)	- some of the data in the request was out of range or not supported	7F	B4	xx	xx	xx	12	
	General Response [7F] of: 21 (Busy)	- the ECU is Busy - try again	7F	B4	xx	xx	xx	21	
Execution Routine State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B4	xx	xx	xx	22	
Information Transfer State	General Response [7F] of: 11 (Mode not supported)	- the ECU does not support this Mode (message)	7F	B4	xx	xx	xx	11	
	General Response [7F] of: 22 (conditions not correct)	- cannot perform the requested action in this state	7F	B4	xx	xx	xx	22	

Appendix B – References

B.1 General

The following table lists and describes all of the defined references used in this document.

Reference	Document Title	Document, Section Number or location
[1]	–EESYS SDS	EY-0051, EY-0052, EY-0053, EY-0104, EY-0129
[2]	Electrical/Electronic Systems Diagnostic Terms, Definitions, Abbreviations, and Acronyms	SAE J1930
[3]	Diagnostic Connector	SAE J1962
[4]	Diagnostic Trouble Code Definitions	SAE J2012
[5]	Enhanced E/E Diagnostic Test Modes	SAE J2190
[6]	E/E Data Link Security	SAE J2186
[7]	Global Diagnostic Specification (Part 1)	v2003.0
[8]	Module Programming and Configuration Specification	v2003.0
[9]	Ford-9141 Protocol Definition and Interface Requirements	ES-F7LC-12K529-FD
[10]	Multiplex Standards Section Web-page	www_eese.ford.com/mux
[11]	Worldwide Diagnostics Group Web-page	www_eese.ford.com/diag
[12]	SCP Vehicle Network Implementation Requirements (SCP-003)	ES-F7LC-12K529-DE
[13]	Class B Data Communications Network Interface	SAE-J1850
[14]	SCP Vehicle Network Implementation Requirements	ES-F7LC-12K529-DE (SCP-003)
[15]	SCP Protocol Definition and Interface Requirements	ES-F7LC-15K529-BC (SCP-001)
[16]	Section 418-00 - Module Communications Network	http://tso.so.ford.com/service/outline/System4.htm
[17]	Section 418-01 - Module Configuration	NO TAG
[18]	Class B Data Communications Network Interface	SAE J2150
[19]	EE/E Diagnostic Test Modes	SAE J1979
[20]	Class B Data Communication Network Messages	SAE J2178
[21]	Emission related diagnostics - Communication between vehicle and test equipment	ISO 15031-5
[22]	OBD Scan Tool	SAE J1978
[23]	UBP Vehicle Network Implementation Requirements	ES-XS4T-12K529-HC
[24]	UBP Protocol Definition and Interface Requirements	ES-XS4T-12K529-F_

[25]	Road Vehicles - Communication Between Vehicle and Test Equipment	ISO 15031
[26]	Global Diagnostic Specification (Part One) Test Procedure Specification	v2003.0
[27]	Communication "Linked Based" Service Diagnostic Requirements	https://f1.ford.com/eRoom/EESE/NetCom

Ford requirements documents, such as this one may be accessed on the Ford Intranet at the R&VT/EESE on-line documentation site (http://www_eese.ford.com/diag).

Information for obtaining SAE documents may be found on the external Internet: http://www.sae.org.

Information for obtaining ISO documents may be found on the external Internet: http://www.iso.ch.

Appendix C – Abbreviations, Acronyms and Glossary

C.1 Purpose

This Appendix contains a listing of the Abbreviations, Acronyms, and a Glossary of the Terms used in this document.

C.2 Abbreviations

2H	Transfer Case Gear position
2WD	2-Wheel Drive
4H	Transfer Case Gear position
4L	Transfer Case Gear position
4WABS	4-Wheel Anti-Lock Braking System
4WAS	4-Wheel Air Suspension
4WD	4-Wheel Drive
4X4	4-Wheel Drive Vehicle
A4WD	Automatic 4-Wheel Drive
ABM	AirBag Module
ABS	Anti-Lock Braking System
ABS/TA	Anti-Lock Braking System/Traction Assist
ABS/TC	Anti-lock Braking System/Traction Control
A/C	Air Conditioning
ACC	Accessory
ACD	Automotive Components Division
ACM	Audio Control Module
ACP	Audio Control Protocol
A/D	Analog-to-Digital
Ah	Ampere-hour
AMB	Ambient temperature sensor
APSM	Alternator Power Supply Module
ARC	Automatic Ride Control
AWD	All Wheel Drive
BB	engineering development BreadBoard
BCD	Binary Coded Decimal
BESS	Body Electrical Subsystem Specification
BMP	Bit-Mapped
B & A	Body & Assembly
C	Celsius/centigrade

c/o	carry-over
CAN	Controller Area Network
CARB	California Air Resources Board
CCD	Climate Control Division
CD/DJ	Compact Disk/Disk Jockey
CE	Conducted Emission
CF PCI	Consecutive Frame PCI
CI	Conducted Imunity
CM	Control Module
CP	Confirmation Prototype
CPC	Confirmation Prototype Craftsmanship
CPF	Confirmation Prototype Functional
CPM	Cellular Phone Module
CPP	Contact Plate Power
CRC	Cyclical Redundancy Check
CRC	Cyclical Redundancy Code
CTM	Central Timing Module
CVSC	Continuously Variable Semi-active ride Control
DC	Direct Current
DCR	Design Change Request
DDM	Driver Door Module
DFMEA	Design Failure Modes & Effect Analysis
DLC	Data Link Connector
DMR	Direct Memory Reference
DPID	Data Packet IDentification number
DR	Driver
DRL	Daytime Running Light
DSM	Driver Seat Module
DSP	Diagnostic Service & Planning
DTC	Diagnostic Trouble Code.
DVP & R	Design Verification Procedure & Report
EACC	Electronic Automatic Climate Control
EATC	Electronic Air Temperature Control module
ECM	Engine Control Module
ECS	Electronic Crash Sensor
ECT	Engine Coolant Temperature
ECTC	Engine Coolant Temperature Control
ECU	Electronic Control Unit (module)
EDIS	Electronic Distributor-less Ignition System
E/E	Electrical/Electronic

EEC	Electronic Engine Controller (also see PCM)
EEPROM	Electrically Erasable Programmable Read Only Memory
EESE	Electrical/Electronic Systems Engineering
EIC	Electronic Instrument Cluster
ELD	Ford Electronics Division
EMC	ElectroMagnetic Compatibility
EMI	ElectroMagnetic Interference
EOL	End Of Line
EP	Evaluation Prototype
EROM	Erasable Read Only Memory
ES	Engineering Specification
ESD	ElectroStatic Discharge
ESOF	Electronic Shift-on-the-Fly
ESP	Electronic Stability Program
EVTM	Electrical & Vacuum Testing Manual
FC PCI	Flow Control PCI
FCSD	Ford Customer Service Division
FF PCI	First Frame PCI
FIFO	First-In First-Out
FIM	Fuel Indication Module
FMEA	Failure Modes & Effect Analysis
FMEA Plus	Ford FMEA software package
FPDS	Ford Product Development System
FPSD	Ford Parts and Service Division
FR	Function Read
FRU	Field Replaceable Unit
FWD	Front Wheel Drive
GCC	Gulf Coast Countries
GDS	Global Diagnostic Specification
GEM	Generic Electronic Module
HAT	Hot at All Times
HBM	Heated Backlight Module
HEC	Hybrid Electronic Cluster
HEX	Hexadecimal
HS	Headway Sensor
HS CAN	High Speed CAN
HSC	High Speed CAN
HSM	Heated Seat Module
H/W	Hardware
Hz	Hertz

IABM	Integrated AirBag Module
IBS	Inter-Byte Separation
IC	Instrument Cluster module
ICCM	Intelligent Cruise Control Module
ICP	Integrated Control Panel
ICT	In-Car Temperature
ID	IDentification
IIT	Input Integrity Test
I/O	Input or Output
IMS	Inter-Message Separation
I/P	Instrument Panel
IPM	Instrument Panel Module
ISO	International Standards Organization
ITS	Information Transfer State
IVD	Interactive Vehicle Dynamics
J1	Job Number One/ connector
JB	Junction Block
KAM	Keep Alive Memory
km/h	kilometers per hour
LCM	Lighting Control ECU
LED	Light Emitting Diode/Device
LF	Left Front
LLT	Low Level Tool
LOS	Limited Operating Strategy
LR	Launch Readiness
LR	Left Rear
LSb	Least Significant bit (bit 0)
LSB	Least Significant Byte
LSN	Least Significant Nibble (bits 0 - 3)
MC	Message Center module
mm	millimeters
mph	miles per hour
MRD	Material Required Date
MRDB	Master Reference DataBase
MRDB	Multiplex Reference DataBase
MROM	Masked Read Only Memory
ms	millisecond
MSb	Most Significant bit (bit 7)
MSB	Most Significant Byte
MSM	Memory Seat Module

MSN	Most Significant Nibble (bits 4 - 7)
MSOF	Mechanical Shift-on-the-Fly
MTU	Measurement Time Unit
MY	Model Year
N/A	not applicable
NAV	Navigation module
Nibble	$-Byte = four bits; two Nibbles = Byte
NGS	New Generation Star tester
NGSC	Next Generation Speed Control
NGV	Natural Gas Vehicle
NRZ	Non-Return to Zero
NSCL	No Stored Codes Logging
NUM	Numeric
NVM	Non-Volatile Memory
NVRAM	Non-Volatile Random Access Memory
OASIS	On-line Automotive Service Information System
ODST	On-Demand Self-Test
OTD	One Touch Down
OTP	One Time Programmable
PA	Program Approval
PASS	Passenger
PAT	Program Activity Team
PATS	Passive Anti-Theft System
PC	Printed Circuit/Personal Computer
PCB	Printed Circuit Board
PCSD	Powertrain Control & Emissions Diagnostics
PCI	Protocol Control Information
PCM	Powertrain Control Module
PEO	Product Engineering Office
PDM	Passengers Door Module
PFMEA	Process Failure Modes & Effect Analysis
PI	Program Implementation
PIA	Part(s) In Assembly
PID	Parameter Identifier
PMT	Program Management Team
PRO	Product Request Order
PSD	Power Sliding Door module
PSDM	Power Sliding Door Module
PSDS	Power Sliding Door Subsystem
PSM	Passengers Seat Module

PST	Program Steering Team
PSW	Part Sample Warrant
PWM	Pulse Width Modulated
R & VT	Research & Vehicle Technolory
RAM	Random Access Memory
RAP	Remote Anti-theft Personality module
RAS	Rear Air Suspension
RASM	Rear Air Suspension Module
RCC	Remote Climate Control
RE	Radiated Emission
REM	Rear Electronic Module
RES	Remote Entry Sensor
RF	Radio Frequency
RF	Right Front
RI	Radiated Imunity
RIM	Rear Integrated Module
RKE	Remote Keyless Entry
rpm	Reveloutions Per Minute
RPN	Ranking/Probability Number
ROM	Read Only Memory
RPO	Regular Production Option
RR	Right Rear
R/R	Request/Response
RTS	Return To Subroutine
RWD	Rear Wheel Drive
SAE	Society of Automotive Engineering
SARC	Semi-Active Ride Control
SATC	Semi-Automatic Temperature Control
SBDS	Service Bay Diagnostic System
SCIL	Steering Column Integrated Light module
SCM	Speed Control Module
SCP	Standard Corporate Protocol
SCP-CARB	SCP for California
SCP-DIAG	SCP Diagnostics
SDS	System Design Specification
SED	State Encoded
SF PCI	Single Frame PCI
SLS	Sun Load Sensor
SO	engineering Sign-Off
SOF	Shift-On-the-Fly

SP	Structural Prototype
SPSS	Subsystem Program-Specific Specification
SRS	Supplemental Restraint System
SSDS	Subsystem Specific Diagnostic Specification
SSPR	SubSystem Program Requirements
STn	System Time period
S/W	Software
TA	Traction Assist
TBD	To be defined
TCS	Traction Control System
TIC	Transmitter Identification Code
TNI	Trouble Not Identified
TNODE	Transmitting Node
TSB	Technical Service Bulletin
UART	Universal Asynchronous Receiver Transmitter
UBP	UART-Based Protocol
UNM	Unsigned Numeric
VACM	Voice Activated Control Module
Vbatt	Vehicle battery
VDM	Vehicle Dynamics Module
VIC	Virtual Image Cluster
VID	Vehicle ID block
VIN	Vehicle Identification Number
VLCM	Variable Load Control Module
VP	Verification Prototype
VSCS	Vehicle Specific Configuration Specification
WCR	Worldwide Customer Requirements
WCT	Worldwide Customer Timing
WDR	Worldwide Design Requirements
WHII	WorkHorse II

C.3 Acronyms

CARB	California Air Resources Board
EESE	Electrical Electronic Systems Engineering
EXCEL	Microsoft software spreadsheet program
FAO	Ford Automotive Operations
FMEA	Failure Modes & Effects Analysis
GEM	Generic Electronic Module (ECU)
IAO	International Automotive Operations
ISO	International Standard Operations
KAM	Keep Alive Memory
NAAO	North American Automotive Operations
OBD II	On-board Diagnostics - Government emissions regulations
PCB	Printed Circuit Board
PID	Parameter Identification
PRNDL	Park, Reverse, Neutral, Drive, and Low settings in the gearshift structure
RAM	Random Access Memory
ROM	Read Only Memory
WWDO	WorldWide Diagnostic Organization

C.4 Terms

Backlight	(Backlight) Lighting on the rear of a display.
Bit	Smallest data unit of a computer command
Byte	Group of eight bits (two Nibbles) numbered 0 to 7
Interbyte	Time period(s) between two or more data Bytes.
ms	milliseconds
Nibble	Four bits - two nibbles make a byte
Petabyte	Thousand Trillion Bytes
PWM	Pulse Width Modulation
Sleep Mode	Module hibernates; enters a low power, dormant, quiescent state.
Wiggle	A physical test of the hardware and connections
$	Prefix defining a hexadecimal number