



| | | | | |
|--|---|--|---|------------------------|
|  | SURFACE VEHICLE RECOMMENDED PRACTICE | |  | J2534-2 OCT2010 |
| | | | Issued | 2006-03 |
| | | | Revised | 2010-10 |
| | | | Superseding | J2534-2 MAR2006 |
| (R) Optional Pass-Thru Features | | | | |

RATIONALE

The specification was expanded to provide the following enhancements:

- Added ability to access multiple devices simultaneously
- Added support for UART Echo Byte Protocol
- Added support for Honda Diag-H Protocol
- Added support for Repeat Messaging
- Added support for Extended Programming Voltage
- Added support for SAE J1939 Protocol
- Added support for SAE J1708 Protocol
- Added support for Device Parameter configuration
- Added support for TP2.0 Protocol
- Added support for Fault-Tolerant CAN
- Added Discovery Mechanism to discover the supported features
- Other editorial changes

TABLE OF CONTENTS

| | | |
|-------|---|---|
| 1. | SCOPE..... | 6 |
| 1.1 | Purpose..... | 6 |
| 1.2 | Documentation Convention..... | 6 |
| 2. | REFERENCES..... | 6 |
| 2.1 | General References..... | 6 |
| 2.2 | References for Single Wire CAN..... | 6 |
| 2.3 | References for GM UART..... | 7 |
| 2.4 | References for UART Echo Byte Protocol..... | 7 |
| 2.4.1 | SAE Publications..... | 7 |
| 2.4.2 | ISO Documents..... | 7 |
| 2.5 | References for Honda DIAG-H..... | 7 |
| 2.5.1 | SAE Publications..... | 7 |
| 2.5.2 | ISO Documents..... | 7 |
| 2.6 | References for J1939..... | 8 |
| 2.6.1 | SAE Publications..... | 8 |
| 2.7 | References for TP 2.0..... | 8 |
| 2.8 | References for Fault-Tolerant CAN..... | 8 |

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be reaffirmed, revised, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2010 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org

SAE WEB ADDRESS:

<http://www.sae.org>

**SAE values your input. To provide feedback
on this Technical Report, please visit
http://www.sae.org/technical/standards/J2534/2_201010**

| | | |
|--------|--|----|
| 3. | DEFINITIONS | 8 |
| 3.1 | Definitions for Analog Inputs | 8 |
| 3.2 | Definitions for Honda DIAG-H | 8 |
| 3.3 | Definitions for TP 2.0 | 8 |
| 4. | ACRONYMS | 8 |
| 4.1 | Acronyms for Single wire CAN | 8 |
| 4.2 | Acronyms for GM UART | 8 |
| 4.3 | Acronyms for Analog Inputs | 9 |
| 4.4 | Acronyms for UART Echo Byte Protocol | 9 |
| 4.5 | Acronyms for J1939 | 9 |
| 4.6 | Acronyms for TP 2.0 | 9 |
| 4.7 | Acronyms for Fault-Tolerant CAN | 9 |
| 5. | ENABLING SAE J2534-2 FEATURES AND ACCESSING MULTIPLE SAE J2534 DEVICES | 9 |
| 6. | PIN SELECTION | 9 |
| 6.1 | Scope of the Pin Selection optional feature | 9 |
| 6.2 | Pass-Thru system requirements | 9 |
| 6.2.1 | Pin Usage | 9 |
| 6.3 | Win32 Application Programming Interface | 10 |
| 6.3.1 | API Functions – Overview | 10 |
| 6.3.2 | API Functions - Detailed Information | 11 |
| 6.3.3 | IOCTL Section | 12 |
| 6.4 | Discovery Support | 14 |
| 7. | ACCESS TO ADDITIONAL CHANNELS | 14 |
| 7.1 | Discovery Support | 15 |
| 8. | MIXED FORMAT FRAMES ON A CAN NETWORK | 15 |
| 8.1 | Scope of the Mixed Format Frames on a CAN Network optional feature | 15 |
| 8.2 | Win32 Application Programming Interface | 15 |
| 8.2.1 | API Functions – Overview | 15 |
| 8.2.2 | API Functions - Detailed InformationPassThruReadMsgs | 18 |
| 8.2.3 | IOCTL Section | 19 |
| 8.3 | Message Structure | 20 |
| 8.3.1 | Elements | 20 |
| 8.4 | Discovery Support | 20 |
| 9. | SINGLE WIRE CAN | 20 |
| 9.1 | Scope of the Single Wire CAN optional feature | 20 |
| 9.2 | Pass-Thru System Requirements | 20 |
| 9.2.1 | Pin Usage | 20 |
| 9.3 | Win32 Application Programming Interface | 20 |
| 9.3.1 | API Functions – Overview | 20 |
| 9.3.2 | API Functions - Detailed Information | 22 |
| 9.3.3 | IOCTL Section | 22 |
| 9.4 | Message Structure | 25 |
| 9.4.1 | Elements | 25 |
| 9.5 | Discovery Support | 26 |
| 10. | ANALOG INPUTS | 26 |
| 10.1 | Scope of the Analog Inputs Optional Feature | 26 |
| 10.2 | Pass-Thru System Requirements | 26 |
| 10.2.1 | Analog Inputs | 26 |
| 10.2.2 | Simultaneous Communication on Multiple Protocols | 26 |
| 10.3 | Win32 Application Programming Interface | 27 |
| 10.3.1 | API Functions – Overview | 27 |
| 10.3.2 | API Functions - Detailed Information | 27 |

| | | |
|--------|--|----|
| 10.3.3 | IOCTL Section..... | 28 |
| 10.4 | Message Structure | 32 |
| 10.4.1 | Examples: | 32 |
| 10.4.2 | Message Flag and Status Definitions | 33 |
| 10.4.3 | DLL Installation and Registry | 34 |
| 10.5 | Discovery Support..... | 34 |
| 11. | GM UART (SAE J2740) | 34 |
| 11.1 | Scope of the GM UART optional feature | 34 |
| 11.2 | Pass-Thru System Requirements | 34 |
| 11.2.1 | Pin Usage..... | 34 |
| 11.3 | Win32 Application Programming Interface..... | 35 |
| 11.3.1 | API Functions – Overview..... | 35 |
| 11.3.2 | API Functions - Detailed Information | 36 |
| 11.3.3 | IOCTL Section..... | 36 |
| 11.4 | Message Structure | 39 |
| 11.4.1 | Elements | 39 |
| 11.4.2 | Message Data Formats..... | 39 |
| 11.5 | DLL Installation and Registration | 39 |
| 11.6 | Discovery Support..... | 39 |
| 12. | UART ECHO BYTE PROTOCOL | 39 |
| 12.1 | Scope of the UART Echo Byte protocol optional feature..... | 39 |
| 12.2 | Pass-Thru system requirements | 39 |
| 12.2.1 | Simultaneous communication on multiple protocols..... | 39 |
| 12.2.2 | Pin Usage..... | 40 |
| 12.3 | Win32 Application Programming Interface..... | 40 |
| 12.3.1 | API Functions – Overview..... | 40 |
| 12.3.2 | API Functions - Detailed Information | 41 |
| 12.3.3 | IOCTL Section..... | 43 |
| 12.4 | Message Structure | 45 |
| 12.4.1 | C / C++ Definition..... | 45 |
| 12.4.2 | Message Data Formats..... | 45 |
| 12.4.3 | Format checks for messages passed to the API | 45 |
| 12.4.4 | Message Flag and Status Definitions | 45 |
| 12.5 | Return Value Error Codes..... | 46 |
| 12.6 | Discovery Support..... | 46 |
| 13. | HONDA DIAG-H PROTOCOL | 46 |
| 13.1 | Scope of the Honda DIAG-H optional feature..... | 46 |
| 13.2 | Pass-Thru System Requirements | 46 |
| 13.2.1 | Simultaneous Communication on Multiple Protocols..... | 46 |
| 13.2.2 | Programmable power supply | 46 |
| 13.2.3 | Pin Usage..... | 46 |
| 13.2.4 | Honda Diagnostic Connectors | 46 |
| 13.2.5 | Serial Communication Interface / Protocol..... | 49 |
| 13.2.6 | Electrical Characteristics of Interface..... | 50 |
| 13.3 | Win32 Application Programming Interface..... | 52 |
| 13.3.1 | API Functions – Overview..... | 52 |
| 13.3.2 | API Functions - Detailed Information | 53 |
| 13.3.3 | IOCTL Section..... | 53 |
| 13.4 | Message Structure | 53 |
| 13.4.1 | C / C++ Definition..... | 53 |
| 13.4.2 | Elements | 54 |
| 13.4.3 | Message Data Formats..... | 54 |
| 13.4.4 | Format checks for messages passed to the API | 54 |
| 13.4.5 | Message Flag and Status Definitions | 54 |
| 13.5 | Discovery Support..... | 54 |

| | | |
|--------|---|----|
| 14. | REPEAT MESSAGING | 54 |
| 14.1 | Scope of the Repeat Messaging protocol optional feature | 54 |
| 14.2 | Win32 Application Programming Interface..... | 54 |
| 14.2.1 | API Functions – Overview..... | 54 |
| 14.2.2 | IOCTL Section..... | 55 |
| 14.3 | Discovery Support..... | 58 |
| 15. | EXTENDED PROGRAMMING VOLTAGE SUPPORT | 59 |
| 15.1 | Scope of the Extended Programming Voltage Feature | 59 |
| 15.2 | Pass-Thru System Requirements | 59 |
| 15.3 | Win32 Application Programming Interface..... | 59 |
| 15.3.1 | API Functions – Overview..... | 59 |
| 15.3.2 | API Functions - Detailed Information | 59 |
| 15.4 | Discovery Support..... | 59 |
| 16. | J1939 PROTOCOL | 60 |
| 16.1 | Scope of the J1939 protocol optional feature | 60 |
| 16.2 | Pass-Thru system requirements | 60 |
| 16.2.1 | Connection to Vehicle | 60 |
| 16.2.2 | Communication Protocol..... | 60 |
| 16.2.3 | Simultaneous communication on multiple protocols..... | 60 |
| 16.3 | Win32 Application Programming Interface..... | 61 |
| 16.3.1 | API Functions – Overview..... | 61 |
| 16.3.2 | API Functions - Detailed Information | 61 |
| 16.3.3 | IOCTL Section..... | 62 |
| 16.4 | Message Structure | 64 |
| 16.4.1 | C / C++ Definition..... | 64 |
| 16.4.2 | Elements | 64 |
| 16.4.3 | Message Data Formats..... | 65 |
| 16.4.4 | Format Checks for Messages Passed to the API | 65 |
| 16.4.5 | Conventions for Returning Messages from the API..... | 65 |
| 16.4.6 | Message Flag and Status Definitions | 65 |
| 16.5 | Return Value Error Codes..... | 66 |
| 16.6 | Discovery Support..... | 66 |
| 17. | J1708 PROTOCOL | 66 |
| 17.1 | Scope of the J1708 protocol optional feature | 66 |
| 17.2 | Pass-Thru System Requirements | 66 |
| 17.2.1 | Connection to Vehicle | 66 |
| 17.2.2 | Communication Protocol..... | 67 |
| 17.2.3 | Simultaneous Communication on Multiple Protocols..... | 67 |
| 17.3 | Win32 Application Programming Interface..... | 67 |
| 17.3.1 | API Functions – Overview..... | 67 |
| 17.3.2 | API Functions - Detailed Information | 67 |
| 17.4 | Message Structure | 68 |
| 17.4.1 | C / C++ Definition..... | 68 |
| 17.4.2 | Elements | 68 |
| 17.4.3 | Message Data Formats..... | 68 |
| 17.4.4 | Conventions for Returning Messages from the API..... | 69 |
| 17.4.5 | Message Flag and Status Definitions | 69 |
| 17.5 | Discovery Support..... | 69 |
| 18. | EXTENDED PASSTHRUIOCTL FOR DEVICE CONFIGURATION PARAMETERS..... | 69 |
| 18.1 | Scope of the Extended PassThruIoctl Optional Feature..... | 69 |
| 18.2 | Pass-Thru Concept | 69 |
| 18.3 | Win32 Application Programming Interface..... | 70 |
| 18.3.1 | API Functions – Overview..... | 70 |
| 18.4 | IOCTL Section..... | 70 |
| 18.4.1 | GET_DEVICE_CONFIG | 70 |
| 18.4.2 | SET_DEVICE_CONFIG..... | 71 |

| | | |
|--------|---|-----|
| 18.5 | Discovery Support..... | 73 |
| 19. | TP2.0 PROTOCOL | 73 |
| 19.1 | Scope of the TP2.0 protocol optional feature | 73 |
| 19.2 | Pass-Thru system requirements | 73 |
| 19.2.1 | Simultaneous communication on multiple protocols..... | 73 |
| 19.2.2 | Pin Usage..... | 73 |
| 19.3 | Win32 Application Programming Interface..... | 73 |
| 19.3.1 | API Functions – Overview..... | 73 |
| 19.3.2 | API Functions - Detailed Information | 75 |
| 19.3.3 | IOCTL Section..... | 76 |
| 19.4 | Message Structure | 79 |
| 19.4.1 | C / C++ Definition..... | 79 |
| 19.4.2 | Message Data Formats..... | 80 |
| 19.4.3 | Format Checks for Messages Passed to the API | 80 |
| 19.4.4 | Message Flag and Status Definitions | 80 |
| 19.5 | Return Values | 82 |
| 19.6 | Discovery Support..... | 82 |
| 20. | FAULT-TOLERANT CAN..... | 82 |
| 20.1 | Scope of the Fault-Tolerant CAN Optional Feature..... | 82 |
| 20.2 | Pass-Thru System Requirements | 82 |
| 20.2.1 | Pin Usage..... | 82 |
| 20.3 | Win32 Application Programming Interface..... | 82 |
| 20.3.1 | API Functions – Overview..... | 82 |
| 20.3.2 | API Functions - Detailed Information | 83 |
| 20.4 | Message Structure | 83 |
| 20.4.1 | Elements | 83 |
| 20.4.2 | Message Flag and Status Definitions | 83 |
| 20.5 | Discovery Support..... | 84 |
| 21. | DISCOVERY MECHANISM..... | 84 |
| 21.1 | Scope of the Discovery Mechanism Feature | 84 |
| 21.2 | Pass-Thru System Requirements | 84 |
| 21.3 | Win32 Application Programming Interface..... | 84 |
| 21.3.1 | API Functions – Overview..... | 84 |
| 21.3.2 | API Functions - Detailed Information | 85 |
| 22. | SAE J2534-2 RESOURCE | 104 |
| 22.1 | Connect Flag Values..... | 104 |
| 22.2 | ProtocolID Values | 105 |
| 22.3 | Filter Type Values | 107 |
| 22.4 | Ioctl ID Values | 107 |
| 22.5 | IOCTL GET / SET CONFIG Parameter Details | 108 |
| 22.6 | GET_DEVICE_CONFIG/ SET_DEVICE_CONFIG Parameter Details..... | 109 |
| 22.7 | GET_DEVICE_INFO Defines | 109 |
| 22.8 | GET_PROTOCOL_INFO Defines..... | 111 |
| 22.9 | Error Return Values | 111 |
| 22.10 | TxFlags Bits | 112 |
| 22.11 | RxStatus Bits..... | 112 |
| 23. | NOTES..... | 112 |
| 23.1 | Marginal Indicia | 112 |

1. SCOPE

SAE J2534-1 defines a standard vehicle network interface that can be used to reprogram emission-related control modules. However, there is a need to support vehicles prior to the 2004 model year as well as non-emission related control modules.

The SAE J2534-2 document meets these needs by detailing extensions to an SAE J2534-1 specification. It is not required for an interface to be fully compliant with SAE J2534-1 specification to implement some of the features specified in this document. Together, these extensions provide the framework for a common interface to protect the software investment of the Vehicle OEMs and Scan Tool manufacturers.

Only the optional features will be described by this document and are based on the December 2004 publication of SAE J2534-1.

1.1 Purpose

Each section included in this document specifies features that extend the SAE J2534-1 specification. The specific feature operation will be described directly or reference another existing specification. In each case the required calling structure, via the SAE J2534-1 API, will be documented and coordinated by this document.

Extending the protocols supported by SAE J2534-1 this document adds two new types of ProtocolIDs.

1. ProtocolIDs with the suffix '_PS' for connecting to a vehicle, via the SAE J1962 connector using the technique outlined in the section titled 'Pin Selection'.
2. Generic ProtocolIDs, with the suffixes '_CH1' through '_CH128' for protocols that terminate at a vendor specific connector on the device. See the section titled 'Access to Additional Channels'.

1.2 Documentation Convention

For each protocol defined in this document:

- Unless explicitly specified otherwise, all SAE J2534-1 PassThru function calls will be supported.
- SAE J2534-1 ConnectFlags, RxStatus, TxFlags and Indications are not supported unless explicitly specified.

2. REFERENCES

2.1 General References

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J2534-1 Recommended Practice for Pass-Thru Vehicle Programming

SAE J1962 Diagnostic Connector

2.2 References for Single Wire CAN

The current published manufacturer specific documents for Single Wire CAN may be acquired from the following URL: <http://global.ihs.com/>

GMW3089 GMLAN Single Wire CAN Physical and Data Link Layers Specification Definitions

GMW3173 Architecture & Bus Wiring Requirements

GMW3110 GMLAN Enhanced Diagnostics Test Mode Specifications

2.3 References for GM UART

The current published manufacturer specific definition of GM UART may be acquired from SAE as an SAE Information Report from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J2740 General Motors UART Serial Data Communications

2.4 References for UART Echo Byte Protocol

2.4.1 SAE Publications

Implementation of the optional feature requires reference to the following documents which are available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J2809 Honda Diagnostic Serial Data Link Protocol - ABS / VSA System

SAE J2818 Keyword Protocol 1281

2.4.2 ISO Documents

Available from American National Standards Institute, 25 West 43rd Street, New York, NY 10036-8002, Tel: 212-642-4900, www.ansi.org.

ISO 9141:1989 Road Vehicles - Diagnostic Systems - Requirements for interchange of digital information

ISO 9141-2:1994 Road Vehicles - Diagnostic Systems - CARB requirements for interchange of digital information

2.5 References for Honda DIAG-H

2.5.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J1962 Diagnostic Connector

2.5.2 ISO Documents

Available from American National Standards Institute, 25 West 43rd Street, New York, NY 10036-8002, Tel: 212-642-4900, www.ansi.org.

ISO 9141:1989 Road Vehicles - Diagnostic Systems - Requirements for interchange of digital information

ISO 9141-2:1994 Road Vehicles - Diagnostic Systems - CARB requirements for interchange of digital information

2.6 References for J1939

2.6.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J1939 Recommended Practice for a Serial Control and Communications Vehicle Network

SAE J1939-21 Data Link Layer

SAE J1939-81 Network Management

2.7 References for TP 2.0

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J2819 TP2.0 Vehicle DIAGNOSTIC Protocol

2.8 References for Fault-Tolerant CAN

Available from American National Standards Institute, 25 West 43rd Street, New York, NY 10036-8002, Tel: 212-642-4900, www.ansi.org.

ISO 11898-3:2006 Road Vehicles – Controller Area Network (CAN) – Part 3: Low-speed, Fault-tolerant, Medium-dependent Interface

3. DEFINITIONS

3.1 Definitions for Analog Inputs

CHANNEL: An analog-to-digital channel

SUBSYSTEM: A collection of similar channels

3.2 Definitions for Honda DIAG-H

DIAG-H: Honda proprietary single wire UART based diagnostic communication bus.

92Hm/2: Honda proprietary diagnostic communication protocol.

3.3 Definitions for TP 2.0

TELEGRAM: A CAN message used on an established Transport Protocol connection

4. ACRONYMS

4.1 Acronyms for Single Wire CAN

GMLAN: General Motors Local Area Network

SWCAN: Single Wire CAN.

4.2 Acronyms for GM UART

UART: Universal Asynchronous Receiver/Transmitter

4.3 Acronyms for Analog Inputs

A/D: Analog to Digital

4.4 Acronyms for UART Echo Byte Protocol

ABS: Anti-lock Braking System

VSA: Vehicle Stability Assist

4.5 Acronyms for J1939

BAM: Broadcast Announce Message.

4.6 Acronyms for TP 2.0

TP: Transport Protocol

4.7 Acronyms for Fault-Tolerant CAN

FTCAN: Fault-Tolerant CAN, ISO 11898-3

5. ENABLING SAE J2534-2 FEATURES AND ACCESSING MULTIPLE SAE J2534 DEVICES

To enable J2534-2 features, call PassThruOpen with pName pointing to a NULL terminated string that begins with the ASCII characters "J2534-2."

If pName is NULL only J2534-1 features shall be allowed.

If pName is not NULL and it does not begin with "J2534-2:" no assumptions can be made regarding the J2534-1 or J2534-2 features.

To use multiple devices from the same PC the application shall append the vendor specific device name to the string identified above.

Example

"J2534-2:" to open the J2534-2 default device as determined by the vendor

"J2534-2:VendorX Device1" to open a J2534-2 device identified as "VendorX Device1"

A device, to be compliant to the SAE J2534-2 specification, shall implement the Discovery Mechanism and shall support all the parameters specified for GET_DEVICE_INFO and GET_PROTOCOL_INFO ioctlIDs. Please see "Discovery Mechanism" section for details.

6. PIN SELECTION

6.1 Scope of the Pin Selection Optional Feature

This section identifies the pin selection mechanism for ProtocolIDs that have the '_PS' suffix. The API extensions detailed here describe the method of specifying the pin(s) to which the selected protocol should be connected. While the API allows for all combinations of protocol / pin assignments, the actual combinations implemented are vendor specific.

6.2 Pass-Thru System Requirements

6.2.1 Pin Usage

The set of pins that can be switched is dependant on the set of optional protocols supported by the interface. A new set of SET_CONFIG parameters are defined for the application to specify the pins to be switched on various standard cables.

6.3 Win32 Application Programming Interface

6.3.1 API Functions – Overview

The new ProtocolIDs with ‘_PS’ suffix indicates that the protocol physical layer is not connected to any pins on PassThruConnect. A new ioctl configuration parameter is also added, that allows connection of a physical layer to specific pins on the specified cable.

For support of SAE J2534-1 protocols on different pins than those defined in SAE J2534-1, new ProtocolIDs are assigned to enable the pin-switching feature as defined in Figure 1:

| Definition | Description |
|-------------|---|
| J1850VPW_PS | GM /Chrysler CLASS2 |
| J1850PWM_PS | Ford SCP |
| ISO9141_PS | ISO 9141 and ISO 9141-2 |
| ISO14230_PS | ISO 14230-4 (Keyword Protocol 2000) |
| CAN_PS | Raw CAN (flow control not handled automatically by interface) |
| ISO15765_PS | ISO 15765-2 flow control enabled |
| J2610_PS | SAE J2610 (Chrysler SCI) |

FIGURE 1 - PROTOCOL ID VALUES (J2534-1 DEFINED)

NOTE: This is not an exhaustive list of _PS protocols. Please refer to the “SAE J2534-2 Resources” Section for a complete list

As an example, in order to utilize a CAN channel connected to Pins 3 & 11 (often used for Medium Speed CAN network), the new CAN_PS ProtocolID is used in PassThruConnect. The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID. (The only exception to this is the Mixed Format Frames feature on a CAN Network.)

Note that the SAE J2610 (Chrysler SCI) protocols are consolidated into a single new ProtocolID, J2610_PS. However the SAE J2534-1 SCI protocols shall continue to be supported as defined in SAE J2534-1.

New SAE J2534-2 optional feature protocols use the ProtocolIDs defined in “SAE J2534-2 Resources” Section

The interface must manage resource locking of pins that are in use. If an existing channel is using a pin that is requested by a PassThruIoctl call, the new request shall be rejected. The interface must also prevent a channel from being assigned to different pins on different cables at the same time.

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|-----------------|--|
| PassThruConnect | For all protocols with the ‘_PS’ suffix defined above, no connection to any pins is made on the call to PassThruConnect. |
| PassThruIoctl | Add a new configuration parameter to allow selection of cable and pins. |

FIGURE 2 - SAE J2534 API FUNCTIONS

6.3.2 API Functions - Detailed Information

6.3.2.1 PassThruConnect

When PassThruConnect is called, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG, J1962_PINS, J1939_PINS or J1708_PINS is made.

There are two major differences from PassThruConnect usage in SAE J2534-1.

- The new ProtocolIDs (ending in '_PS') are not assigned pins upon connection. No transmission or reception on a channel is possible until the pins are assigned using the IOCTL parameter J1962_PINS, J1939_PINS or J1708_PINS.
- If the interface supports multiple instances of a given protocol device, the '_PS' ProtocolIDs can be opened multiple times. For example, if an interface device supports two independent CAN controllers, a program could open CAN_PS and request pins 3 & 11, then open CAN_PS again and request pins 1 & 12.

Devices that are not physically capable of supporting the requested protocol shall return ERR_NOT_SUPPORTED.

Example 1: Devices that do not support any transceivers of a particular type (e.g., SW_CAN_PS).

Example 2: Devices with only one controller of a particular type (e.g., one physical CAN controller) shall disallow opening that '_PS' ProtocolID multiple times (since it could never satisfy the 2nd request).

For further information regarding failure conditions refer to the PassThruIoctl section.

6.3.2.2 PassThruDisconnect

This API Function shall return the SAE J1962 Pins to the default state as specified by SAE J2534-1. The SAE J1939 and SAE J1708 pins shall be returned to a high impedance state.

6.3.2.3 PassThruReadMsgs

If pins have not been assigned, ERR_PIN_INVALID shall be returned if this function is called.

6.3.2.4 PassThruWriteMsgs

If pins have not been assigned, ERR_PIN_INVALID shall be returned if this function is called.

6.3.2.5 PassThruStartPeriodicMsg

If pins have not been assigned, ERR_PIN_INVALID shall be returned if this function is called.

6.3.2.6 PassThruStartMsgFilter

If pins have not been assigned, ERR_PIN_INVALID shall be returned if this function is called.

6.3.2.7 PassThruIoctl

Pins are assigned via SET_CONFIG with the J1962_PINS, J1939_PINS, or J1708_PINS parameters. Refer to the SET_CONFIG section for details regarding this parameter. All other PassThruIoctl functions for this Channel ID shall return ERR_PIN_INVALID until the pin assignment is successfully completed.

The application must call PassThruDisconnect before attempting to reassign pins.

6.3.3 IOCTL Section

6.3.3.1 GET_CONFIG

The following configuration parameters are supported:

J1962_PINS

J1939_PINS

J1708_PINS

See Figure 3 for more details.

6.3.3.2 SET_CONFIG

The configuration parameters, J1962_PINS, J1939_PINS and J1708_PINS are added as defined in Figure 3. For the protocol channel referenced in the ChannelID parameter of the SET_CONFIG call, these parameters specify which connector and pin or pins this protocol's physical layer is to be connected to. The act of setting this parameter causes the connection of the protocol physical layer to the specified pins. At the vendors discretion, detection of the correct cable can be use to reject an invalid request with a return code of ERR_INVALID_IOCTL_VALUE.

NOTE: Unpredictable results may occur if cable detection is not implemented and an incorrect cable is connected.

| Parameter | Valid values for Parameter | Default Value (Decimal) | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|---|-------------------------|--|--------------|----------|------------|---|--------|---|---|-----------|---|---|-------|---|---|-------|---|---|-----------|---|---|--------|---|---|--------|---|---|---|---|---|---|---|
| J1962_PINS | 0x0000PPSS where: PP: 0x00 – 0x10 SS: 0x00 – 0x10 PP != SS, except when set to 0x0000 Exclude pins 4, 5, & 16 | 0 | <p>For a channel of any protocol type this selects the SAE J1962 pin, or pair of pins, onto which the physical layer is to be switched.</p> <p>NOTE: A value of 0 can never be set. Reading a value of 0 indicates that pin selection has not been performed.</p> <p>PP is the pin number for the primary signal e.g. ISO K-line, CAN-H, +ve, SCI Tx, DIAG-H, SAE J1850+,....</p> <p>SS is the pin number for the secondary signal, where a secondary signal is present e.g. ISO L-line, CAN-L, -ve, SCI Rx, SAE J1850-, ... SS shall equal 0x00 if no secondary pin is required or enabled.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| J1939_PINS | 0x0000PPSS where: PP: 0x00 – 0x09 SS: 0x00 – 0x09 PP != SS, except when set to 0x0000 Exclude pins 1, 2 | | <p>For a channel of any protocol type this selects the SAE J1939-13 pin, or pair of pins, onto which the physical layer is to be switched.</p> <table><tr><th>J1939-13 PIN</th><th>Function</th><th>PIN Number</th></tr><tr><td>A</td><td>Ground</td><td>1</td></tr><tr><td>B</td><td>Battery +</td><td>2</td></tr><tr><td>C</td><td>CAN_H</td><td>3</td></tr><tr><td>D</td><td>CAN_L</td><td>4</td></tr><tr><td>E</td><td>Shield/NC</td><td>5</td></tr><tr><td>F</td><td>J1708+</td><td>6</td></tr><tr><td>G</td><td>J1708-</td><td>7</td></tr><tr><td>H</td><td>-</td><td>8</td></tr><tr><td>J</td><td>-</td><td>9</td></tr></table> | J1939-13 PIN | Function | PIN Number | A | Ground | 1 | B | Battery + | 2 | C | CAN_H | 3 | D | CAN_L | 4 | E | Shield/NC | 5 | F | J1708+ | 6 | G | J1708- | 7 | H | - | 8 | J | - | 9 |
| J1939-13 PIN | Function | PIN Number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A | Ground | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | Battery + | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C | CAN_H | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| D | CAN_L | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E | Shield/NC | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| F | J1708+ | 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| G | J1708- | 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| H | - | 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| J | - | 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | <p>NOTE: A value of 0 can never be set. Reading a value of 0 indicates that pin selection has not been performed.</p> <p>PP is the pin number for the primary signal e.g. ISO K-line, CAN-H, +ve, SCI Tx, DIAG-H, SAE J1850+,....</p> <p>SS is the pin number for the secondary signal, where a secondary signal is present e.g. ISO L-line, CAN-L, -ve, SCI Rx, SAE J1850-, ... SS shall equal 0x00 if no secondary pin is required or enabled.</p> | | | | | | | | | | | | | | | | | | | | | |
|------------|---|------------|---|-----------|----------|------------|---|--------|---|---|--------|---|---|----------|---|---|---|---|---|--------|---|---|---|---|
| J1708_PINS | 0x0000PPSS where: PP: 0x00 – 0x09 SS: 0x00 – 0x09 PP != SS, except when set to 0x0000 Exclude pins 3, 5 | | <p>For a channel of any protocol type this selects the pin or pair of pins on the SAE J1708 6 pin Deutsch connector, onto which the physical layer is to be switched.</p> <table><tr><th>J1708 PIN</th><th>Function</th><th>PIN Number</th></tr><tr><td>A</td><td>J1708+</td><td>1</td></tr><tr><td>B</td><td>J1708-</td><td>2</td></tr><tr><td>C</td><td>Battery+</td><td>3</td></tr><tr><td>D</td><td>-</td><td>4</td></tr><tr><td>E</td><td>Ground</td><td>5</td></tr><tr><td>F</td><td>-</td><td>6</td></tr></table> <p>NOTE: A value of 0 can never be set. Reading a value of 0 indicates that pin selection has not been performed.</p> <p>PP is the pin number for the primary signal e.g. ISO K-line, CAN-H, +ve, SCI Tx, DIAG-H, SAE J1850+,....</p> <p>SS is the pin number for the secondary signal, where a secondary signal is present e.g. ISO L-line, CAN-L, -ve, SCI Rx, SAE J1850-, ... SS shall equal 0x00 if no secondary pin is required or enabled.</p> | J1708 PIN | Function | PIN Number | A | J1708+ | 1 | B | J1708- | 2 | C | Battery+ | 3 | D | - | 4 | E | Ground | 5 | F | - | 6 |
| J1708 PIN | Function | PIN Number | | | | | | | | | | | | | | | | | | | | | | |
| A | J1708+ | 1 | | | | | | | | | | | | | | | | | | | | | | |
| B | J1708- | 2 | | | | | | | | | | | | | | | | | | | | | | |
| C | Battery+ | 3 | | | | | | | | | | | | | | | | | | | | | | |
| D | - | 4 | | | | | | | | | | | | | | | | | | | | | | |
| E | Ground | 5 | | | | | | | | | | | | | | | | | | | | | | |
| F | - | 6 | | | | | | | | | | | | | | | | | | | | | | |

FIGURE 3 - IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

For existing SAE J2534-1 base protocols, the physical interface is connected to the SAE J1962 pins automatically when PassThruConnect is called, as defined in SAE J2534-1. The J1962_PINS parameter does not need to be supported for SAE J2534-1 base protocols, but the use of these protocols will impact the pins that are available for the _PS protocols.

For protocols with the '_PS' suffix, at least one of the J1962_PINS, J1939_PINS or J1708_PINS parameters must be supported and the default value of the parameter shall be 0x0000. Certain combinations of SAE J1962, SAE J1939 or SAE J1708 pins could result in vehicle or interface damage. ERR_PIN_INVALID is returned for combinations that are outside of the defined range. The application programmer is responsible for determining if the potential exists for damage to the vehicle and taking the necessary steps to prevent it.

Only one SET_CONFIG, with the parameter J1962_PINS, J1939_PINS or J1708_PINS can be performed for a given Channel ID. After a successful call to SET_CONFIG if a second call for a given Channel ID is attempted, ERR_CHANNEL_IN_USE shall be returned. PassThruDisconnect is required before an alternate cable or pin selection may be attempted.

For the J1962_PINS, J1939_PINS and J1708_PINS parameters, the following error handling shall be applied:

- PassThruIoctl requests unsupported pin combination

PassThruIoctl, SET_CONFIG, is called, requesting a value of the J1962_PINS, J1939_PINS or J1708_PINS parameter that are valid but can't be supported by the hardware or having the potential of damaging the interface device.

Result: PassThruIoctl returns ERR_NOT_SUPPORTED

- PassThruIoctl requests pin combination that would cause conflict

PassThruIoctl, SET_CONFIG, is called, setting the J1962_PINS, J1939_PINS or J1708_PINS parameter to a value that causes a pin conflict with an existing channel.

Result: PassThruIoctl returns ERR_PIN_IN_USE.

6.4 Discovery Support

The device shall report support for Pin Switching and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

7. ACCESS TO ADDITIONAL CHANNELS

This section defines an optional mechanism available to initiate and use multiple channels of the same protocol, if vendor hardware provides the support. The channels addressed this way are not be tied to pins on the SAE J1962 connector. What channel appears at what pins on the interface device will depend on the vendor configuration. The channels are typically accessed using custom cable built to hardware vendor's specification of connection details.

For example a particular vendor's hardware may support four channels of dual-wire high-speed CAN. These additional channels will be available as separate ProtocolIDs defined by SAE J2534-2. This document will allocate 128 predefined ProtocolIDs for each protocol to target a possible maximum of 128 channels of each protocol. The ProtocolIDs will follow the following format:

CAN_CH1
CAN_CH2
.....
CAN_CH128

This scheme of providing additional ProtocolIDs will apply to both SAE J2534-1 and SAE J2534-2 defined protocols. A complete list of ProtocolIDs for multiple channel access can be found in the 'SAE J2534-2 Resource' section.

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID. (The only exception to this is the Mixed Format Frames feature on a CAN Network.)

Based on Vendor implementation protocols initiated using J2534-1 or _PS or _CHx ProtocolIDs may use the same hardware resources. When there is a resource conflict a call to PassThruConnect will return ERR_RESOURCE_IN_USE. In the case where the requested resource does not exist ERR_NOT_SUPPORTED is returned.

7.1 Discovery Support

The device shall report support for multiple channels through the discovery mechanism defined in “Discovery Mechanism” Section.

8. MIXED FORMAT FRAMES ON A CAN NETWORK

8.1 Scope of the Mixed Format Frames on a CAN Network Optional Feature

This section details the extensions to SAE J2534-1 that will allow the simultaneous reception and transmission of ISO 15765 messages and unformatted CAN frames on an ISO 15765 channel. The *ProtocolID* (in the PASSTHRU_MSG structure) will be used to identify the format of the associated message. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed.

8.2 Win32 Application Programming Interface

8.2.1 API Functions – Overview

Connecting to an ISO 15765 channel and then setting the IOCTL configuration parameter CAN_MIXED_FORMAT to CAN_MIXED_FORMAT_ON shall allow messages to be processed as either unformatted CAN frames or as ISO 15765 messages. Additionally, setting the IOCTL configuration parameter CAN_MIXED_FORMAT to CAN_MIXED_FORMAT_ALL_FRAMES shall allow the individual frames of an ISO 15765 message (including Flow Control) to also be processed in a parallel path as an unformatted CAN frame. Unformatted CAN frames shall have the *ProtocolID* in the PASSTHRU_MSG structure set to an appropriate CAN protocol ID and shall follow the requirements and restrictions for that protocol. ISO 15765 messages shall have the *ProtocolID* in the PASSTHRU_MSG structure set to an appropriate ISO 15765 protocol ID and shall follow the requirements and restrictions for that protocol. For example CAN is associated with ISO15765, SW_CAN_PS is associated with SW_ISO15765_PS, CAN_PS is associated with ISO15765_PS, etc.

When transmitting a message or starting a periodic message, the *ProtocolID* in the PASSTHRU_MSG structure shall be used to identify how the associated message shall be processed. If the configuration setting LOOPBACK is set to ON, then transmitted messages (including flow control) shall be processed in the same manner as received messages. As with SAE J2534-1, messages will be sent one at a time. This makes it possible for an ISO 15765 message to block other messages until transmission is complete, including unformatted CAN frames.

The function PassThruStartMsgFilter shall be used to identify ISO 15765 messages as well as unformatted CAN frames. A received message that matches a FLOW_CONTROL_FILTER shall be processed as an ISO 15765 message and shall have the appropriate *ProtocolID* in the PASSTHRU_MSG structure before it is added to the receive queue. Additionally, based upon the setting of CAN_MIXED_FORMAT, the CAN frames may also be subject to the PASS_FILTERs and BLOCK_FILTERs, where the *ProtocolID* in the PASSTHRU_MSG structure shall be set to reflect an unformatted CAN frame before it is added to the receive queue. The Figure 4 and Figure 5 outline how received messages are processed for the various settings of CAN_MIXED_FORMAT:

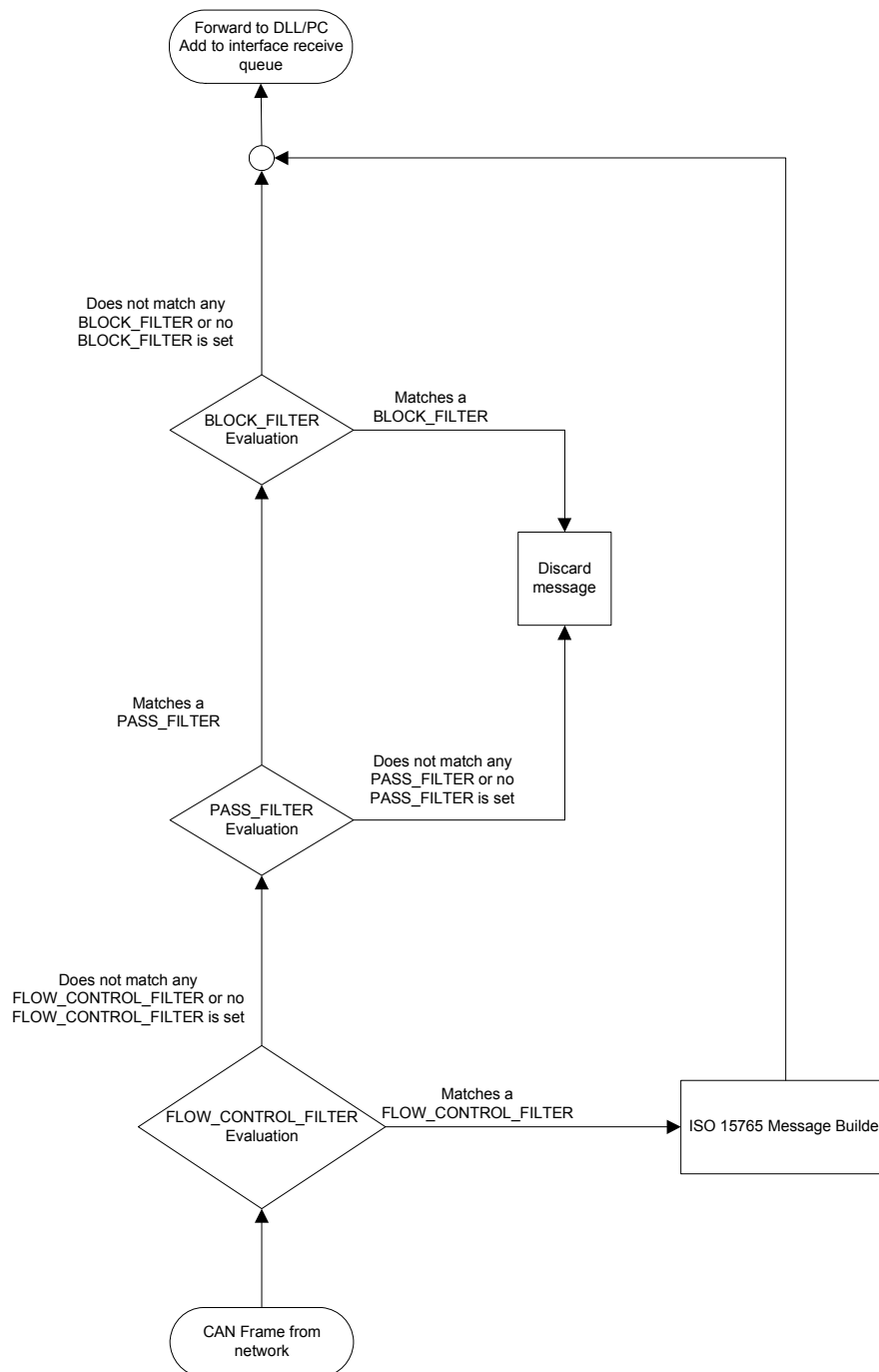


FIGURE 4 - PROCESSING OF RECEIVED MESSAGES WHEN CAN_MIXED_FORMAT IS CAN_MIXED_FORMAT_ON

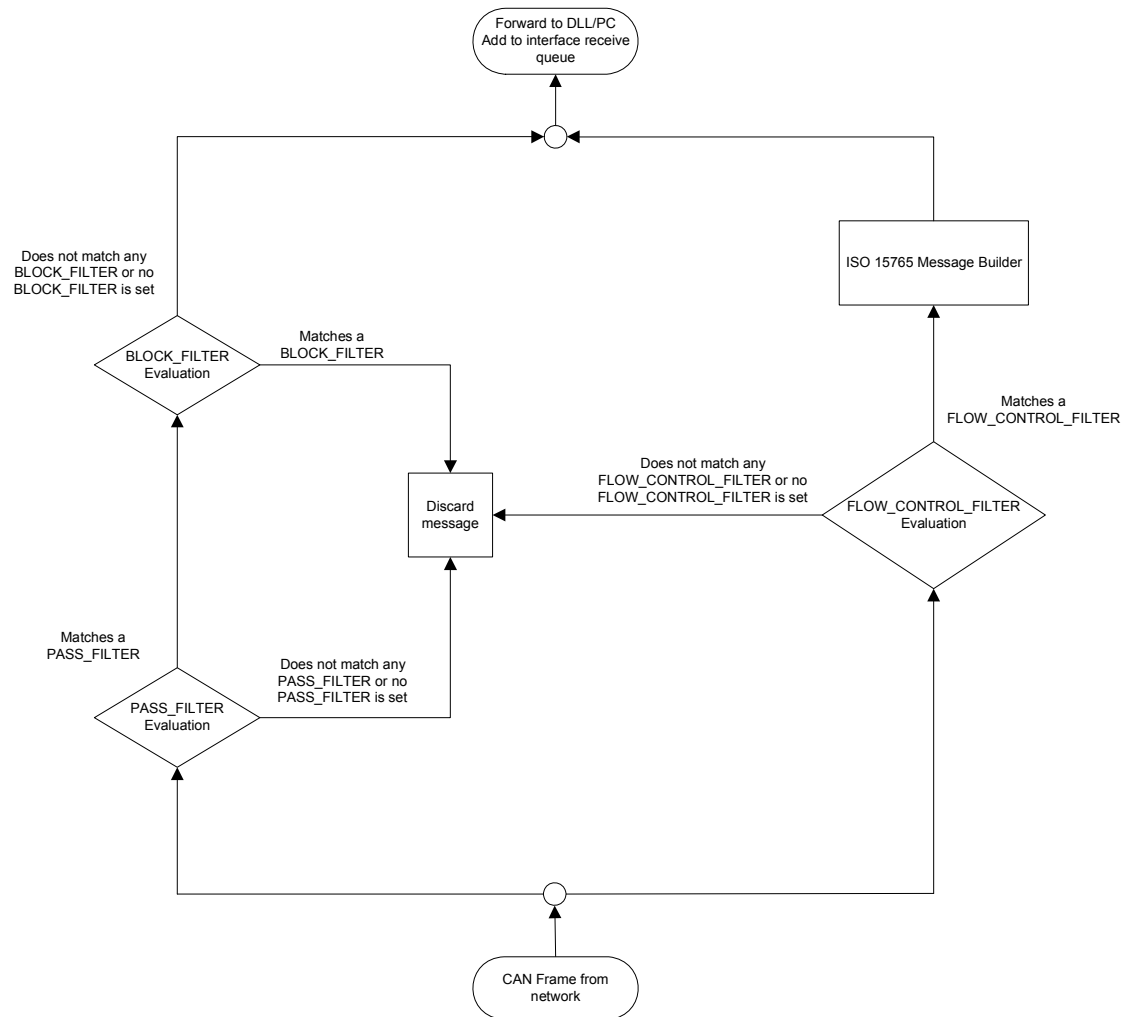


FIGURE 5 - PROCESSING OF RECEIVED MESSAGES WHEN CAN_MIXED_FORMAT IS CAN_MIXED_FORMAT_ALL_FRAMES

If the optional feature is not supported on the current ISO 15765 channel, the call to get or set the IOCTL configuration parameter CAN_MIXED_FORMAT shall return the value ERR_NOT_SUPPORTED.

Figure 6 summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|------------------------|--|
| PassThruStartMsgFilter | Increase the minimum number of FLOW_CONTROL_FILTERs to 64 and PASS_FILTERs/BLOCK_FILTERs to a total of 10. |
| PassThruIoctl | Add a new configuration parameter. |

FIGURE 6 - SAE J2534 API FUNCTIONS

8.2.2 API Functions - Detailed Information

8.2.2.1 PassThruReadMsgs

There is no change to this function. However, only ISO 15765 channels will allow the IOCTL configuration parameter `CAN_MIXED_FORMAT` to be either `CAN_MIXED_FORMAT_ON` or `CAN_MIXED_FORMAT_ALL_FRAMES`. In these cases, the *ProtocolID* in the `PASSTHRU_MSG` structure shall reflect either an unformatted CAN frame (e.g., `CAN`, `SW_CAN_PS`, etc.) or an ISO 15765 message (e.g., `ISO15765`, `SW_ISO15765_PS`, etc.). Consult the appropriate SAE J2534 document for the requirements, restrictions, and error conditions for the specific protocol.

Additionally, each time the IOCTL configuration parameter `CAN_MIXED_FORMAT` is set, the receive queue shall be cleared.

8.2.2.2 PassThruWriteMsgs

There is no change to this function. However, only ISO 15765 channels will allow the IOCTL configuration parameter `CAN_MIXED_FORMAT` to be either `CAN_MIXED_FORMAT_ON` or `CAN_MIXED_FORMAT_ALL_FRAMES`. In these cases, the *ProtocolID* in the `PASSTHRU_MSG` structure shall reflect either an unformatted CAN frame (e.g., `CAN`, `SW_CAN_PS`, etc.) or an ISO 15765 message (e.g., `ISO15765`, `SW_ISO15765_PS`, etc.). Consult the appropriate SAE J2534 document for the requirements, restrictions, and error conditions for the specific protocol. This function will return a value of `ERR_MSG_PROTOCOL_ID` if the IOCTL configuration parameter `CAN_MIXED_FORMAT` is set to `CAN_MIXED_FORMAT_OFF` and the *ProtocolID* reflects an unformatted CAN frame.

Additionally, each time the IOCTL configuration parameter `CAN_MIXED_FORMAT` is set the transmit queue shall be cleared.

The SAE J2534 device will not protect the user from writing an unformatted CAN frame that may be interpreted as a valid ISO 15765 frame. The consequences of this action are undefined and may disrupt ISO 15765 communications taking place on the network.

8.2.2.3 PassThruStartPeriodicMsg

There is no change to this function. However, only ISO 15765 channels will allow the IOCTL configuration parameter `CAN_MIXED_FORMAT` to be either `CAN_MIXED_FORMAT_ON` or `CAN_MIXED_FORMAT_ALL_FRAMES`. In these cases, the *ProtocolID* in the `PASSTHRU_MSG` structure shall reflect either an unformatted CAN frame (e.g., `CAN`, `SW_CAN_PS`, etc.) or an ISO 15765 message (e.g., `ISO15765`, `SW_ISO15765_PS`, etc.). Consult the appropriate SAE J2534 document for the requirements, restrictions, and error conditions for the specific protocol. This function will return a value of `ERR_MSG_PROTOCOL_ID` if the IOCTL configuration parameter `CAN_MIXED_FORMAT` is set to `CAN_MIXED_FORMAT_OFF` and the *ProtocolID* reflects an unformatted CAN frame.

Additionally, each time the IOCTL configuration parameter `CAN_MIXED_FORMAT` is set, periodic messages with *ProtocolID* of `CAN` shall be deleted.

The SAE J2534 device will not protect the user from writing a CAN message that may be interpreted as a valid ISO 15765 frame. The consequences of this action are undefined and may disrupt ISO 15765 communications taking place on the network.

8.2.2.4 PassThruStartMsgFilter

Each ISO 15765 channel shall support a minimum of 64 `FLOW_CONTROL_FILTERS` as well as 10 `PASS_FILTERS/BLOCK_FILTERS`.

This function shall be used to identify ISO 15765 messages as well as unformatted CAN frames. A received message that matches a `FLOW_CONTROL_FILTER` shall be processed as an ISO 15765 message and shall have the appropriate *ProtocolID* in the `PASSTHRU_MSG` structure before it is added to the receive queue. Additionally, based upon the setting of `CAN_MIXED_FORMAT`, the CAN frames may also be subject to the `PASS_FILTERS` and `BLOCK_FILTERS`, where the *ProtocolID* in the `PASSTHRU_MSG` structure shall be set to reflect an unformatted CAN frame before it is added to the receive queue.

Only ISO 15765 channels will allow the IOCTL configuration parameter CAN_MIXED_FORMAT to be set to either CAN_MIXED_FORMAT_ON or CAN_MIXED_FORMAT_ALL_FRAMES. In these cases, the *ProtocolID* in the PASSTHRU_MSG structure must reflect an unformatted CAN frame (e.g., CAN, SW_CAN_PS, etc.) for a PASS_FILTER or a BLOCK_FILTER and an ISO 15765 message (e.g., ISO15765, SW_ISO15765_PS, etc.) for a FLOW_CONTROL_FILTER. Consult the appropriate SAE J2534 document for the requirements, restrictions, and error conditions for the specific protocol and filter type. This function will return a value of ERR_MSG_PROTOCOL_ID if the *ProtocolID* is not appropriate for the type of filter being started.

Additionally, each time the IOCTL configuration parameter CAN_MIXED_FORMAT is set all PASS_FILTERs and BLOCK_FILTERs shall be deleted.

8.2.3 IOCTL Section

If this feature is not supported on the current ISO 15765 channel, the call to get or set the IOCTL configuration parameter CAN_MIXED_FORMAT shall return the value ERR_NOT_SUPPORTED.

8.2.3.1 GET_CONFIG

The configuration parameter CAN_MIXED_FORMAT has been defined but it is only applicable to ISO 15765 channels. See Figure 7 for more details.

8.2.3.2 SET_CONFIG

The configuration parameter CAN_MIXED_FORMAT has been defined but it is only applicable to ISO 15765 channels. CAN_MIXED_FORMAT configuration parameter is defined in Figure 7.

| Parameter | Valid values for Parameter | Default Value | Description |
|------------------|---|--------------------------|---|
| CAN_MIXED_FORMAT | 0 (CAN_MIXED_FORMAT_OFF) 1 (CAN_MIXED_FORMAT_ON) 2 (CAN_MIXED_FORMAT_ALL_FRAMES) | 0 (CAN_MIXED_FORMAT_OFF) | <p>For ISO 15765 channels only, this enables the transmission and reception of messages with the ProtocolID of ISO 15765 or unformatted CAN frames. FLOW_CONTROL_FILTERs will identify messages with the ProtocolID that reflects an ISO 15765 message, while PASS_FILTERs and BLOCK_FILTERs will identify messages with the ProtocolID that reflects an unformatted CAN frame.</p> <p>Any time this parameter is set, the transmit and receive queues shall be cleared, all PASS_FILTERs and BLOCK_FILTERs shall be deleted, and periodic messages whose ProtocolID reflects an unformatted CAN frame shall be deleted.</p> <p>0 = Messages will be treated as ISO 15765 ONLY. 1 = Messages will be treated as either ISO 15765 or an unformatted CAN frame. 2 = Messages will be treated as ISO 15765, an unformatted CAN frame, or both.</p> |

FIGURE 7 - IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

8.3 Message Structure

8.3.1 Elements

There is no change to any of the elements, but it should be noted that the *ProtocolID* in the *PASSTHRU_MSG* structure shall identify the message type (e.g., ISO15765, CAN, SW_ISO15765_PS, SW_CAN_PS, etc.). Consult the appropriate SAE J2534 document for the requirements, restrictions, and error conditions for the specific protocol.

8.4 Discovery Support

The device shall report support for mixed format frames on the CAN network through the discovery mechanism defined in "Discovery Mechanism" Section.

9. SINGLE WIRE CAN

9.1 Scope of the Single Wire CAN Optional Feature

Information contained in this section will define extensions to a compliant SAE J2534-1 interface to support Single Wire CAN.

9.2 Pass-Thru System Requirements

9.2.1 Pin Usage

Single Wire CAN is connected to pin 1 of the SAE J1962 connector.

Note that when *PassThruConnect* is called, the physical layer remains disconnected until a call to *PassThruIoctl*, *SET_CONFIG*, *J1962_PINS* is made.

9.3 Win32 Application Programming Interface

9.3.1 API Functions – Overview

Information contained in this section is intended to define the API resources required to incorporate an optional protocol channel. This channel, identified as Single Wire CAN (SWCAN), will require hardware and software API support to fully implement this feature.

The details on the physical implementation of SWCAN are defined in GMW3089, titled "GMLAN Single Wire CAN Physical and Data Link Layers Specification."

This section outlines the requirements for providing an interface channel supporting this protocol.

At a high level a new *ProtocolID* has been defined to indicate the use of the Single Wire CAN physical layer. Unless indicated otherwise, SAE J2534-1 definitions of requirements, restrictions, and error conditions for CAN and ISO15765 protocols will apply to Single Wire CAN channels. Additionally, Flags required to support high voltage wake up and high / normal speed are defined as required. The device is required to support high voltage wakeup and time critical data rate changes defined in SWCAN specification. This section details the API resources required to enable use of the SAE J2534 API as applied to SWCAN.

If this feature is not supported an error code, *ERR_NOT_SUPPORTED*, will be returned by the call *PassThruConnect*.

The IOCTL and Configuration settings related to SW_CAN speed change and load resistor control are designed for use when the SAE J2534-2 hardware interface is used in either of two modes:

1. As a hardware interface for a flash programming application (such as DPS).
2. As a hardware interface for any other application that could be used to monitor traffic during a flash programming event performed by another test tool on the bus.

GMW3110 requires a maximum transition time of 30 ms to switch between bus speeds. Depending on the individual SAE J2534-2 hardware interface and its associated PC communications interface and application processing speed, the 30 ms transition time may or may not be met using speed transition logic that is embedded in the PC application. The IOCTL and Configuration settings allow for this speed change logic to be controlled by either the application or the SAE J2534-2 hardware interface itself.

Since not all application / SAE J2534-2 interface combinations can control speed transition timing within the GMW3110 specification, a method is required to allow the SAE J2534-2 hardware interface to automatically switch speeds while performing a flash programming operation. The following example illustrates this sequence:

Assume all settings are in the Default mode.

1. The application sets the configuration of the SW_CAN_RES_SWITCH parameter – AUTO_RESISTOR.
2. The application sets the configuration of SW_CAN_SPEEDCHANGE_ENABLE - ENABLE_SPDCHANGE.
3. The application sends the correct GMW 3110 HS programming message sequence (0xA5 0x02, 0xA5 0x03)
4. The SAE J2534-2 hardware interface monitors the (0xA5 0x02, 0xA5 0x03) sequence and automatically switches in the load resistor, changes the SWCAN transceiver mode, and reconfigures the CAN controller. Note that this must be accomplished within 30 ms of the 0xA5 0x03 frame. A SW_CAN_HS_RX indication will be generated upon completion of this transition.
5. Upon completion of the flash programming event, the application transmits the return to normal mode command (mode \$20). A SW_CAN_NS_RX indication will be generated upon completion of this transition.
6. The SAE J2534-2 hardware interface responds to the return to normal mode command, which reconfigures the CAN controller, changes the transceiver mode, and disconnects the load resistor. Note that this must be accomplished within 30 ms of the transmission of the return to normal mode frame.

There are also times when the SAE J2534-2 hardware interface could be used with a separate application to monitor a flash-programming event being performed by another test tool. In this case, the SAE J2534-2 hardware interface would need to be able to perform speed transitions within the GMW3110 specified limit. An SAE J2534-2 hardware interface used in this manner should not connect its load resistor, as this functionality should already be contained in the test tool performing the flash-programming event.

The following example illustrates the setup for monitoring such an event:

Assume all settings are in the Default mode:

1. The application sets the configuration of SW_CAN_SPEEDCHANGE_ENABLE - ENABLE_SPDCHANGE.
2. The SAE J2534-2 hardware interface monitors the (0xA5 0x02, 0xA5 0x03) sequence and automatically changes the SWCAN transceiver mode, and reconfigures the CAN controller. (It does not switch in the load resistor) Note that this must be accomplished within 30 ms of the 0xA5 0x03 frame. A SW_CAN_HS_RX indication will be generated upon completion of this transition.

3. Upon completion of the flash programming event, the SAE J2534-2 hardware interface receives the return to normal mode command (mode \$20). A SW_CAN_NS_RX indication will be generated upon completion of this transition.
4. The SAE J2534-2 hardware interface responds to the return to normal mode command, which reconfigures the CAN controller and changes the transceiver mode. Note that this must be accomplished within 30 ms of the transmission of the return to normal mode frame.

Although the option of manual load resistor activation (CONNECT_LOAD_RESISTOR) is not shown in any of the above examples, this capability was included to accommodate special test applications that might require this feature.

Figure 8 summarizes the changes to the SAE J2534-1 API functions.

| Function | Description of Change |
|-----------------|--|
| PassThruConnect | Added new ProtocolID values. |
| PassThruIoctl | Added GET_CONFIG and SET_CONFIG parameters. Added new IOCTLS to support SWCAN capability |

FIGURE 8 - SAE J2534 API FUNCTIONS

9.3.2 API Functions - Detailed Information

9.3.2.1 PassThruConnect

When PassThruConnect is called with either the SW_CAN_PS or SW_ISO15765_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

9.3.2.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in Figure 9. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|-----------------|--|
| SW_CAN_PS | Raw Single Wire CAN messages with pin selection |
| SW_ISO15765_PS | Single Wire CAN adhering to ISO15765-2 flow control with pin selection |
| SW_CAN_CHx | Additional channels of Raw Single Wire CAN messages |
| SW_ISO15765_CHx | Additional channels of Single Wire CAN adhering to ISO15765-2 flow control |

FIGURE 9 - PROTOCOLID DESCRIPTIONS

9.3.2.3 PassThruReadMsgs

The RxStatus indications identified in section 9.4.1.1 will be received for both commanded and automatic speed changes.

9.3.3 IOCTL Section

The Protocol IDs SW_ISO15765_PS and SW_ISO15765_CHx support the same IOCTL IDs as defined in J2534-1 specification for ISO15765. SW_CAN_PS and SW_CAN_CHx support the same IOCTL IDs as defined in the J2534-1 specification for CAN. This section details additional IOCTLS defined in this document.

Figure 10 provides the details of the IOCTLs available through PassThruIoctl function:

| Value of IoctlID | InputPtr represents | OutputPtr represents | Purpose |
|---------------------|------------------------|-------------------------|--|
| SW_CAN_HS | NULL Pointer | NULL Pointer | Initiates the transition of the SW_CAN channel from SW_CAN_NS (Normal Speed) mode to SW_CAN_HS (High Speed) mode. This transition includes resetting the SW_CAN transceiver mode to the HS setting and changing the SW_CAN controller configuration to the SW_CAN_HS_DATA_RATE |
| SW_CAN_NS | NULL Pointer | NULL Pointer | Initiates the transition of the SW_CAN channel from SW_CAN_HS (High Speed) mode to SW_CAN_NS (Normal Speed) mode. This transition includes resetting the SW_CAN transceiver mode to the normal mode setting and changing the SW_CAN controller configuration to the DATA_RATE. |

FIGURE 10 - IOCTL DETAILS

9.3.3.1 GET_CONFIG

Support three new parameters added to GET_CONFIG. See SET_CONFIG and Figure 11 for more details.

9.3.3.2 SET_CONFIG

The Protocol IDs SW_ISO15765_PS and SW_ISO15765_CHx support the same Get/Set parameter Ids as defined in J2534-1 specification for ISO15765. SW_CAN_PS and SW_CAN_CHx support the same Get/Set parameter Ids as defined in the J2534-1 specification for CAN. This section details additional IOCTLs defined in this document.

SW_CAN_HS mode is to be used exclusively for the reprogramming of devices. It requires the coordinated and selective configuration of three pieces of hardware – the load resistor, the SW_CAN transceiver and the SW_CAN controller. Specific information regarding each piece is as follows:

1. A load resistor is connected to the SW_CAN bus within the tool, which helps compensate for reduced bit times by decreasing the active to passive transition times. To prevent excessive electrical loading of the SW_CAN bus, this feature shall only be activated by the programming device. All other devices or tools used to monitor high speed communication shall remain in the normal impedance state.
2. The SW_CAN transceiver is placed into a mode which also compensates for the reduced bit times by disabling waveshaping and decreasing the passive to active transition times.
3. The CAN controller is configured to provide the appropriate high speed data rate.

| Parameter | Valid values for Parameter | Default Value (Decimal) | Description |
|---------------------------|--|-------------------------|--|
| SW_CAN_HS_DATA_RATE | 5 – 100000 (Some transceivers support 100k) | 83333 | The data rate to be used in response to a call to SW_CAN_HS_IOCTL |
| SW_CAN_SPEEDCHANGE_ENABLE | 0 (DISABLE_SPDCHANGE) 1 (ENABLE_SPDCHANGE) | 0 | Control the behavior of the SAE J2534 device in response to speed change SW_CAN messages 0 = Ignore all bus speed transition messages on the bus. 1 = Process transmitted and received bus speed transition messages as per GMW3110 Section 10.17.5.2. |
| SW_CAN_RES_SWITCH | 0 (DISCONNECT_RESISTOR) 1 (CONNECT_RESISTOR) 2 (AUTO_RESISTOR) | 0 | Control Load Resistor switching 0 = Default value. Disable automatic switching and disconnect load resistor. 1 = Disable automatic switching and connect load resistor. 2 = Automatically Switch in the load resistor when transitioning to high speed. (and switch off the load resistor while transitioning back to normal speed) |

FIGURE 11 - IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

9.3.3.3 SW_CAN_HS

The ioctlID value of SW_CAN_HS is used to initiate a transition of the Single Wire CAN Bus to High Speed Mode. A successful transition will be noted by a SW_CAN_HS_RX indication. The speed transitioned to is the value of the SW_CAN_HS_DATA_RATE parameter. Parameter definition for SW_CAN_HS is defined in Figure 12.

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during PassThruConnect. |
| ioctlID | Is set to SW_CAN_HS |
| InputPtr | Is a NULL pointer, as this parameter is not used. |
| OutputPtr | Is a NULL pointer, as this parameter is not used. |

FIGURE 12 - SW_CAN_HS DETAIL

9.3.3.4 SW_CAN_NS

The ioctlID value of SW_CAN_NS is used to initiate a transition of the Single Wire CAN Bus to Normal Speed Mode. A successful transition will be noted by a SW_CAN_NS_RX indication. The speed transitioned to is the value of the DATA_RATE parameter. Parameter definition for SW_CAN_NS is detailed in Figure 13.

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during PassThruConnect. |
| ioctlID | Is set to SW_CAN_NS |
| InputPtr | NULL Pointer |
| OutputPtr | NULL Pointer |

FIGURE 13 - SW_CAN_NS DETAILS

9.4 Message Structure

9.4.1 Elements

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID. (The only exception to this is the Mixed Format Frames feature on a CAN Network.)

9.4.1.1 RxStatus

This section defines RxStatus bits are in addition to the ones defined for CAN/ISO15765 sections of J2534-1 specification.

Definitions for RxStatus bits are defined in Figure 14:

| Definition | RxStatus Bit(s) | Description | Value |
|--------------|-----------------|---|--|
| SW_CAN_NS_RX | 18 | Indicates that the Single Wire CAN bus has transitioned to Normal Speed. All communication after this event will occur in normal-speed mode. The message data in this message is undefined. | 0 = No Event 1 = Transition to Normal Speed |
| SW_CAN_HS_RX | 17 | Indicates that the Single Wire CAN bus has transitioned to High Speed. All communication after this event will occur in high-speed mode. The message data in this message is undefined. | 0 = No Event 1 = Transition to High Speed |
| SW_CAN_HV_RX | 16 | Indicates that the Single Wire CAN message received was high-voltage message. | 0 = Normal Message 1 = High-Voltage Message |

FIGURE 14 - RXSTATUS BIT DEFINITIONS

9.4.1.2 TxFlags

This section defines TxStatus bits are in addition to the ones defined for CAN/ISO15765 sections of J2534-1 specification.

Figure 15 defines the TxFlags bit definition.

| Definition | TxFlags Bit(s) | Description | Value |
|--------------|----------------|--|--|
| SW_CAN_HV_TX | 10 | Indicates that the Single Wire CAN message should be transmitted as a high-voltage message. Simultaneously transmitting in high voltage and high speed mode will result in undefined behavior. | 0 = Normal Message 1 = High-Voltage Message |

FIGURE 15 - TXFLAGS BIT DEFINITIONS

9.5 Discovery Support

The device shall report support for Single Wire CAN Network and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

10. ANALOG INPUTS

10.1 Scope of the Analog Inputs Optional Feature

This section details the extensions to SAE J2534-1 that define the common method of supporting analog input channels. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed.

This standard does not specify the timing between the same subsystem or different subsystems. Depending on the device, all the active channel readings could be made simultaneously, or could be spaced out in time.

10.2 Pass-Thru System Requirements

10.2.1 Analog Inputs

Information contained in this section will define extensions to a compliant SAE J2534-1 interface. This section specifically defines the common method of supporting analog input channels.

10.2.2 Simultaneous Communication on Multiple Protocols

The operation of the A/D subsystem shall be independent of the operation of the communications protocols. The interface must support simultaneous collection of analog data and communication on multiple protocols as specified in SAE J2534-1 and SAE J2534-2.

10.3 Win32 Application Programming Interface

10.3.1 API Functions – Overview

Information contained in this section is intended to define the API resources required to incorporate analog input channels on a PassThru device. Analog inputs will require hardware and software API support to fully implement this feature. It allows compliant devices to acquire analog data in an efficient and deterministic manner. Physical connection of the SAE J2534-2 interface to the vehicle is defined by the interface manufacturer.

This new feature allows an application to open a connection to an analog subsystem via PassThruConnect. The subsystem parameters can be set via the GET_CONFIG/SET_CONFIG ioctls. The actual analog readings can be obtained with PassThruReadMsgs, using the ChannelID from PassThruConnect.

Figure 16 summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|------------------|---|
| PassThruConnect | Add new ProtocolID values. |
| PassThruReadMsgs | Format of returned Analog readings. |
| PassThruIoctl | Add a new configuration parameters to control A/D |

FIGURE 16 - SAE J2534 API FUNCTIONS

10.3.2 API Functions - Detailed Information

10.3.2.1 PassThruConnect

The new protocol identifiers ANALOG_IN_1 through ANALOG_IN_32 connect to analog subsystems. Each subsystem can have up to 32 discrete equivalent channels allowing for as many as 1024 analog inputs to be supported.

The various parameters (such as sample rate and averaging method) apply to all channels within a subsystem. The parameters can be different on different subsystems. A device with 8 A/D channels should have 8 subsystems only if each A/D channel can be controlled independently. The device should have 1 subsystem if all 8 channels must have the same sample rate.

ProtocolIDs beyond those supported by the device shall return ERR_NOT_SUPPORTED.

10.3.2.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in section titled 'SAE J2534-2 Resource'. Protocol values for the analog feature are identified in Figure 17.

| Definition | Description |
|--------------|---------------------|
| ANALOG_IN_1 | Analog subsystem 1 |
| ANALOG_IN_2 | Analog subsystem 2 |
| ANALOG_IN_3 | Analog subsystem 3 |
| ... | ... |
| ANALOG_IN_32 | Analog subsystem 32 |

FIGURE 17 - PROTOCOLID VALUES

10.3.2.3 PassThruReadMsgs

To the application, each analog subsystem appears like all the other vehicle protocols. An analog subsystem will periodically generate PASSTHRU_MSG structures which are placed in the queue where the application can read them. The normal PassThruReadMsg features, such as waiting (using Timeout) and gathering multiple messages (using *pNumMsgs) are supported.

See the Message Structure section for the formatting of the samples within a message.

10.3.2.4 PassThruWriteMsgs

This function will return ERR_NOT_SUPPORTED if passed a ChannelID opened for analog input.

10.3.2.5 PassThruStartPeriodicMsg

This function will return ERR_NOT_SUPPORTED if passed a ChannelID opened for analog input.

10.3.2.6 PassThruStartMsgFilter

This function will return ERR_NOT_SUPPORTED if passed a ChannelID opened for analog input.

10.3.3 IOCTL Section

Each analog subsystem shall support 3 IOCTL functions: GET_CONFIG, SET_CONFIG and CLEAR_RX_BUFFER. The CLEAR_RX_BUFFER ioctl shall remove any queued messages for the subsystem.

10.3.3.1 GET_CONFIG

There are several new parameters that are used to setup and control the A/D subsystem. See Figure 18 for more details.

10.3.3.2 SET_CONFIG

The following parameters control the analog subsystem. Note that there is no way to set parameters for each channel individually. See Figure 18 for more details.

| Parameter | Valid values for Parameter | Default (decimal) | Description |
|---------------------|------------------------------|--------------------|---|
| ACTIVE_CHANNELS | 0 – 0xFFFFFFFF | Hardware dependent | Bitmask of channels being sampled |
| SAMPLE_RATE | 0 – 0xFFFFFFFF | 0 | Samples/second or Seconds/sample |
| SAMPLES_PER_READING | 1 – 0xFFFFFFFF | 1 | Samples to average into a single reading |
| READINGS_PER_MSG | 1 – 0x00000408 (1 - 1032) | 1 | Number of readings for each active channel per PASSTHRU_MSG structure |
| AVERAGING_METHOD | 0 – 0xFFFFFFFF | 0 | The way in which the samples will be averaged. |
| SAMPLE_RESOLUTION | 0x1-0x20 (1 – 32) | Hardware dependent | The number of bits of resolution for each channel in the subsystem. Read only. |

| Parameter | Valid values for Parameter | Default (decimal) | Description |
|------------------|---|--------------------|---|
| INPUT_RANGE_LOW | 0x80000000 through 0x7FFFFFFF (-2147483648 through 2147483647) | Hardware dependent | Lower limit in millivolts of A/D input. (Example 0xFFFFB1E0 = -20.00V) Read only. |
| INPUT_RANGE_HIGH | 0x80000000 through 0x7FFFFFFF (-2147483648 through 2147483647) | Hardware dependent | Upper limit in millivolts of A/D input. (Example 20000 = +20.00V) Read only. |

FIGURE 18 - IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

10.3.3.2.1 ACTIVE_CHANNELS

The ACTIVE_CHANNELS parameter controls the number of channels that are actively read into the PASSTHRU_MSG structure. The ACTIVE_CHANNELS parameter is a 32 bit unsigned long bit mask. Each bit that is set indicates that the corresponding channel is active.

Changes to the ACTIVE_CHANNELS takes effect after the completion of the current message (i.e specified readings per message).

The interface must reject combinations of ACTIVE_CHANNELS and READINGS_PER_MSG that would result in a message that is larger than the size of a PASSTHRU_MSG structure (1032 data points). In this case, the error returned shall be ERR_INVALID_IOCTL_VALUE. The interface may not reject valid combinations of ACTIVE_CHANNELS and READINGS_PER_MSG.

The default value for ACTIVE_CHANNELS is for all available channels to be active. For example, a subsystem with 7 channels will set ACTIVE_CHANNELS to 0x7F (127) initially. Trying to set bits for channels that don't exist will return error ERR_INVALID_IOCTL_VALUE.

10.3.3.2.2 SAMPLE_RATE

The SAMPLE_RATE parameter sets the number of samples per second or the number of seconds per sample for each of the active channels. If the SAMPLE_RATE is less than 0x80000000, then the SAMPLE_RATE represents the number of samples per second. For example, 0x7D0 represents 2000 samples/second for each channel. On the other hand, values above 0x80000000 represent seconds per sample (minus the most significant bit). For example, 0x8000000A would be one sample on each channel every 10 seconds. Note that 0x80000001 is the same as 0x00000001. 0x80000000 should be treated the same as 0.

Setting this value to zero has the effect of disabling the associated A/D subsystem. No new messages will be queued, but the existing messages will not be cleared.

If the device does not support the requested sample rate, the device must return ERR_INVALID_IOCTL_VALUE. Changes to this value take effect at the end of the current cycle or immediately if the subsystem was disabled.

The default value for SAMPLE_RATE is zero (subsystem disabled).

NOTE: Setting SAMPLE_RATE to 0 value will stop the data streaming.

10.3.3.2.3 SAMPLES_PER_READING

The SAMPLES_PER_READING parameter sets the number of samples per reading. The parameter AVERAGING_METHOD determines how the reading will be derived from the collected samples.

As you increase the SAMPLES_PER_READING, you increase the number of samples required to fill a PASSTHRU_MSG structure. For example, setting SAMPLES_PER_READING to 3 (without changing other parameters) will make the messages come 3 times slower.

If the device does not support the requested value, the device must return ERR_INVALID_IOCTL_VALUE. The device must support the default value of 1, even if the device does not support averaging. A value of 1 means that averaging is off.

The default value for SAMPLES_PER_READING is one.

The return code ERR_NOT_SUPPORTED will be returned if the sample rate is not zero.

10.3.3.2.4 READINGS_PER_MSG

The READINGS_PER_MSG parameter sets the number of readings of each active channel that will be placed in a PASSTHRU_MSG structure.

The readings will be placed in the PASSTHRU_MSG message in channel order starting from lowest active channel to highest active channel. This format will repeat "READINGS_PER_MSG" times.

The interface must reject combinations of ACTIVE_CHANNELS and READINGS_PER_MSG that would result in a message that is larger than the size of a PASSTHRU_MSG structure (1032 data points). In this case, the error returned shall be ERR_INVALID_IOCTL_VALUE. The interface shall not reject valid combinations of ACTIVE_CHANNELS and READINGS_PER_MSG.

Setting this value to zero has the effect of disabling the associated A/D subsystem.

Changes to this value take affect at the end of the current cycle or immediately if the subsystem was disabled.

The default value for READINGS_PER_MSG is one.

The return code ERR_NOT_SUPPORTED will be returned if the sample rate is not zero.

10.3.3.2.5 AVERAGING_METHOD

When SAMPLES_PER_READING is above one, each reading will consist of several samples. The AVERAGING_METHOD specifies how each reading will be computed. If the device does not support a particular value, ERR_INVALID_IOCTL_VALUE shall be returned. The default value (SIMPLE_AVERAGE) must be supported, even if the device does not support averaging. See Figure 19 for more details.

| Method | Value | Description |
|------------------------|-------------------------|--------------------------|
| SIMPLE_AVERAGE | 0x00000000 | Simple arithmetic mean |
| MAX_LIMIT_AVERAGE | 0x00000001 | Choose the biggest value |
| MIN_LIMIT_AVERAGE | 0x00000002 | Choose the lowest value |
| MEDIAN_AVERAGE | 0x00000003 | Choose arithmetic median |
| (SAE J2534-2 reserved) | 0x00000004 – 0x7FFFFFFF | Reserved |
| (Vendor Reserved) | 0x80000000 – 0xFFFFFFFF | Specific to the vendor |

FIGURE 19 - VALUES FOR THE AVERAGING_METHOD PARAMETER

10.3.3.2.5.1 SIMPLE_AVERAGE

The SIMPLE_AVERAGE is the arithmetic average of SAMPLES_PER_READINGS samples. In other words:

$$\text{Reading} = (\text{Sample}_1 + \text{Sample}_2 + \dots + \text{Sample}_{\text{SAMPLES_PER_READING}}) / \text{SAMPLES_PER_READING}$$

10.3.3.2.5.2 MAX_LIMIT_AVERAGE

The MAX_LIMIT_AVERAGE simply chooses the maximum value.

$$\text{Reading} = \text{Max} (\text{Sample}_1 + \text{Sample}_2 + \dots + \text{Sample}_{\text{SAMPLES_PER_READING}})$$

10.3.3.2.5.3 MIN_LIMIT_AVERAGE

The MIN_LIMIT_AVERAGE simply chooses the minimum value.

$$\text{Reading} = \text{Min} (\text{Sample}_1 + \text{Sample}_2 + \dots + \text{Sample}_{\text{SAMPLES_PER_READING}})$$

10.3.3.2.5.4 MEDIAN_AVERAGE

The MEDIAN_AVERAGE chooses the median value. Sort the samples, then compute:

$$\text{Reading} = \text{Sample}_{(\text{SAMPLES_PER_READING}+1)/2}$$

Or if SAMPLES_PER_READING is even:

$$\text{Reading} = (\text{Sample}_{\text{SAMPLES_PER_READING}/2} + \text{Sample}_{(\text{SAMPLES_PER_READING}/2)+1}) / 2$$

10.3.3.2.5.5 Vendor-specific Averaging Methods

Vendors are free to add their own averaging methods. They should use the range provided so they do not conflict with extensions to this standard.

10.3.3.2.6 SAMPLE_RESOLUTION

This read-only parameter indicates the number of bits of resolution that the A/D channels have in this subsystem. For example, a 12-bit A/D would return 12. Attempting to set this parameter shall return the value ERR_INVALID_IOCTL_PARAM_ID.

10.3.3.2.7 INPUT_RANGE_LOW

This, signed, read-only parameter indicates the lower limit of the A/D subsystem. For example, an A/D subsystem that can measure voltages from -20.0V to +36 would return -20000 (0xFFFFB1E0). Attempting to set this parameter shall return the value ERR_INVALID_IOCTL_PARAM_ID.

10.3.3.2.8 INPUT_RANGE_HIGH

This, signed, read-only parameter indicates the upper limit of the A/D subsystem. For example, an A/D subsystem that can measure voltages from -20.0V to +36 would return 36000 (0x00008CA0). Attempting to set this parameter shall return the value ERR_INVALID_IOCTL_PARAM_ID.

10.3.3.3 CLEAR_RX_BUFFER

The device shall remove any queued messages for the subsystem.

10.4 Message Structure

When a set of readings is ready, the device will queue a PASSTHRU_MSG structure for the application to read. This section specifies how to fill in that structure:

- ProtocolID contains the analog protocol that was connected (i.e. ANALOG_IN_1)
- RxStatus contains the overflow flags (see RxStatus section below).
- TxFlags shall be zero and should be ignored by the application.
- Timestamp contains the time stamp of the first set of readings in the message. The application can calculate the timestamp of the remaining readings. The timestamp must correlate with the timestamp of normal message traffic.
- DataSize contains the number of bytes that the readings take up in the message. DataSize must be a multiple of 4 between 4 (a single reading) and 4128 (a full message), inclusive. The value of DataSize can be computed as READINGS_PER_MSG * (# bits set in ACTIVE_CHANNELS).
- Data[] contains the actual readings. The formatting of the data depends on the various parameters:
 - Each reading will take 4 bytes (32 bits), signed little endian format in millivolts.
 - All active channels in the subsystem (and only channels marked active) are represented. Each active channel is appended in order (starting from lowest active channel to highest active channel).
 - This format will repeat READINGS_PER_MSG times.

10.4.1 Examples

The SAE J2534 device assumed for this example has two analog input subsystems that it supports via protocols ANALOG_IN_1 and ANALOG_IN_2. The ANALOG_IN_1 subsystem provides four 16-bit A/D converters. The ANALOG_IN_2 subsystem provides two 24-bit A/D converters.

Figure 20 to Figure 25 provide examples of different parameters and the resulting structures:

| Parameter | Value for ANALOG_IN_1 | Value for ANALOG_IN_2 |
|---------------------|-----------------------|-----------------------|
| ACTIVE_CHANNELS | 0xF | 3 |
| SAMPLE_RATE | 2 | 0x80000005 |
| SAMPLES_PER_READING | 1 | 1 |
| READINGS_PER_MSG | 1 | 1 |
| AVERAGING_METHOD | 1 | 1 |
| SAMPLE_RESOLUTION | 16 | 24 |

FIGURE 20 - SAMPLE A/D PARAMETER CONFIGURATION

This example uses the default values, except that a SAMPLE_RATE has been set for both subsystems. The first subsystem generates the following data twice per second: (Only the PASSTHRU_MSG Data[] array is shown. Each box represents a 4-byte sample.)

| | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|
| Channel 1 Sample 1 | Channel 2 Sample 1 | Channel 3 Sample 1 | Channel 4 Sample 1 |
|-----------------------|-----------------------|-----------------------|-----------------------|

FIGURE 21 - DATA FROM ANALOG_IN_1 SUBSYSTEM WITH READINGS_PER_MSG = 1

The second subsystem generates the following data once every 5 seconds:

| | |
|-----------------------|-----------------------|
| Channel 1 Sample 1 | Channel 2 Sample 1 |
|-----------------------|-----------------------|

FIGURE 22 - DATA FROM ANALOG_IN_2 SUBSYSTEM WITH READINGS_PER_MSG = 1

Changing the READINGS_PER_MSG parameter on each channel from “1” to “2” changes the format and the rate of messages. The first subsystem now generates this message once per second:

| | | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Channel 1 Sample 1 | Channel 2 Sample 1 | Channel 3 Sample 1 | Channel 4 Sample 1 | Channel 1 Sample 2 | Channel 2 Sample 2 | Channel 3 Sample 2 | Channel 4 Sample 2 |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|

FIGURE 23 - DATA FROM ANALOG_IN_1 SUBSYSTEM WITH READINGS_PER_MSG = 2

The second subsystem now generates this message every 10 seconds:

| | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|
| Channel 1 Sample 1 | Channel 2 Sample 1 | Channel 1 Sample 2 | Channel 2 Sample 2 |
|-----------------------|-----------------------|-----------------------|-----------------------|

FIGURE 24 - DATA FROM ANALOG_IN_2 SUBSYSTEM WITH READINGS_PER_MSG = 2

Changing the ACTIVE_CHANNELS on subsystem 1 to 0xB (11 decimal, 1011 binary) disables Channel 3. In this case the structure would now be:

| | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Channel 1 Sample 1 | Channel 2 Sample 1 | Channel 4 Sample 1 | Channel 1 Sample 2 | Channel 2 Sample 2 | Channel 4 Sample 2 |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|

FIGURE 25 - DATA FROM ANALOG_IN_1 SUBSYSTEM WITH ACTIVE_CHANNELS = 0xB

10.4.2 Message Flag and Status Definitions

Definitions for RxStatus bits are shown in Figure 26.

| Definition | RxStatus Bit(s) | Description | Value |
|------------|-----------------|--|--|
| OVERFLOW | 16 | Indicates that the input range of the A/D has been exceeded on one or more samples in the received message | 0 = All samples good 1 = Some samples clipped |

FIGURE 26 - RXSTATUS BIT DEFINITIONS

10.4.3 DLL Installation and Registry

A registry key ANALOG_IN_x indicates the support for an analog sub-system where <x> indicates the sub-system number.

Example: ANALOG_IN_3 indicates support for Analog Sub-System #3, which can have up to 32 analog input channels.

In the case of devices which contain dynamic hardware architecture, an application will be able to determine the lack of runtime ANALOG_IN_x support when ERR_NOT_SUPPORTED is returned from a PassThruConnect().

The registry entries are retained for support of legacy applications. New applications shall use the discovery mechanism defined in this document.

10.5 Discovery Support

The device shall report support for Analog channels and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

11. GM UART (SAE J2740)

11.1 Scope of the GM UART Optional Feature

Information contained in this section will define extensions to a compliant SAE J2534-1 interface. This section specifically defines the common method of supporting GM's UART protocol as defined in SAE J2740, titled "General Motors UART Serial Data Communications".

11.2 Pass-Thru System Requirements

11.2.1 Pin Usage

All GM vehicles built since the 1996 model year, and a few built during the 1995 model year, have been equipped with an SAE J1962 connector. GM UART uses either Pin 9 or Pin 1 of this connector. Typically, SAE J1962 Pin 9 (primary) is used while SAE J1962 Pin 1 (secondary) is occasionally used. As with all SAE J2534-2 optional protocols, no default pin is identified, therefore, the application developer will be required to set the Pin to be used. See the SAE J1962 Pin Selection section for discussion of pin usage.

Most GM vehicles with serial data links built prior to the 1996 model year are equipped with a 12 pin connector as shown in Figure 27. The mating tool connector is shown in Figure 28. For programming these older vehicles using an SAE J2534 interface, a 12 pin connector must be available instead of an SAE J1962 connector to interface to the vehicle. The signal ground, pin 5 on an SAE J1962 connector, must be connected to pin A of the 12 pin connector. The serial data line, pin 9 on an SAE J1962 connector, must be connected to pin M of the 12 pin connector. The 12 pin connector does not contain battery power, so the SAE J2534 interface cannot be powered from the 12 pin connector.

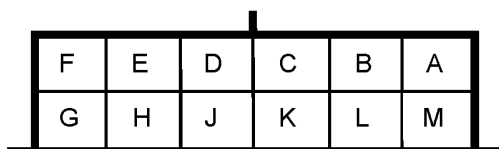


FIGURE 27 - 12 PIN VEHICLE CONNECTOR

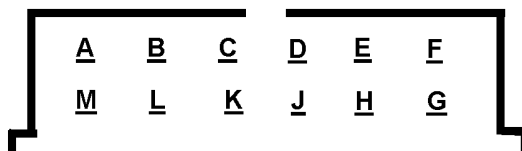


FIGURE 28 - 12 PIN TOOL CONNECTOR

11.3 Win32 Application Programming Interface

11.3.1 API Functions – Overview

Information contained in this section is intended to define the API resources required to incorporate an optional protocol channel. This protocol, identified as the GM UART protocol, will require software API support and hardware to fully implement this feature.

At a high level, the ProtocolID GM_UART_PS and GM_UART_CHx have been defined to indicate the physical layer. The protocol defines a master/slave relationship between the Tester and the Electronic Control Unit (ECU). The tester must request and be granted mastership over the vehicle bus before communications can begin. This section details the API resources required to enable use of the SAE J2534 API as applied to GM UART protocol. If the feature is not supported, an error code, ERR_NOT_SUPPORTED, will be returned by the call to PassThruConnect.

Generally, vehicle bus mastership is accomplished by calling the PassThruIoctl function BECOME_MASTER which monitors the vehicle bus for a poll message and when the poll message is received, a poll response message is returned. Upon receiving the poll response message, the current master will relinquish mastership to sender. However, in some vehicles there is no poll message, so the SAE J2534 Device will be instructed by the application to send the poll response message immediately.

Prior to calling the PassThruIoctl function BECOME_MASTER, the application will first listen to the communication link (using PassThruReadMsgs function) to determine if a tester polling message exists, the type of the polling message (3 byte or 4 byte), and the Device ID of the polling device (4 byte poll only). Using this data, the application will call the PassThruIoctl function SET_POLL_RESPONSE to define the poll response message. The application then calls the BECOME_MASTER function to direct the interface to either wait for a poll message with the same Message ID Byte (MIB) as specified in the poll_ID parameter passed to the BECOME_MASTER command, or to send the poll response message immediately (depending upon the poll ID specified).

Figure 29 summarizes the changes to the SAE J2534-1 API functions.

| Function | Description of Change |
|-----------------|--|
| PassThruConnect | Added new ProtocolID values. |
| PassThruIoctl | Added new PassThruIoctl Sub-functions – SET_POLL_RESPONSE and BECOME_MASTER. |

FIGURE 29 - SAE J2534 API FUNCTIONS

11.3.2 API Functions - Detailed Information

11.3.2.1 PassThruConnect

When PassThruConnect is called with the GM_UART_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

11.3.2.1.1 ProtocolID Values (GM UART)

One additional ProtocolID Value has been defined. Only the definition and description of the ProtocolID value is defined in Figure 30. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|-------------|---|
| GM_UART_PS | GM UART Protocol with pin selection |
| GM_UART_CHx | Additional channels of GM UART protocol |

FIGURE 30 - PROTOCOLID VALUES

11.3.3 IOCTL Section

Figure 31 provides the details on the IOCTLs available through the PassThruIoctl function:

| Value of ioctlID | InputPtr represents | OutputPtr represents | Purpose |
|-------------------|--|----------------------|--|
| SET_POLL_RESPONSE | Points to SBYTE_ARRAY which contains the Poll Response Message. | NULL Pointer | Defines poll response message. |
| BECOME_MASTER | Poll ID: 0 - don't wait for poll message or Non zero - wait for poll ID value | NULL Pointer | Waits for valid poll message and upon receipt transmits poll response message if POLL_ID does not equal zero. If Poll ID is zero, poll response message is sent immediately (no wait). |

FIGURE 31 - IOCTL DETAILS

11.3.3.1 SET_POLL_RESPONSE

Details of the SET_POLL_RESPONSE are shown in Figure 32. The ioctlID parameter of SET_POLL_RESPONSE is used to define the Poll Response Message. Typically, the Poll Response Message is a "Disable Normal Communications" (Mode 8) message, but this will be set by the application.

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during PassThruConnect. |
| ioctlID | Is set to SET_POLL_RESPONSE |
| InputPtr | Points to unsigned char PollResponseMsg[100] |
| OutputPtr | NULL Pointer |

FIGURE 32 - SET_POLL_RESPONSE DETAILS

Refer to Figure 33 for a typical 4 byte Poll Response Message example:

| Byte | Value | Description |
|------|-------|--|
| 1 | F4 | Message ID (MIB) |
| 2 | 56 | Message length (not including message ID OR CHECKSUM) + 0X55 |
| 3 | 08 | ALDL Mode ID |
| 4 | CS | CHECKSUM |

FIGURE 33 - POLL RESPONSE MESSAGE

The Poll Response Message shall be sent by the interface upon the receipt of a Tester Poll Message. There are two types of Tester Poll Messages based on length. The 4 byte type includes a Device ID while the 3 byte type does not. The two types are described are described in Figure 34 and Figure 35.

| Byte | Value | Description |
|------|-------|--|
| 1 | F0 | Message ID (MIB) |
| 2 | 55 | Message length (not including message ID or CHECKSUM) + 0X55 |
| 3 | CS | CHECKSUM |

FIGURE 34 - TYPICAL 3 BYTE TYPE POLL MESSAGE

| Byte | Value | Description |
|------|-------|--|
| 1 | F0 | Message ID (MIB) |
| 2 | 56 | Message length (not including message ID or CHECKSUM) + 0X55 |
| 3 | F4 | Device ID |
| 4 | CS | CHECKSUM |

FIGURE 35 - TYPICAL 4 BYTE TYPE POLL MESSAGE

11.3.3.2 BECOME_MASTER

The `IoctlID` value of `BECOME_MASTER` will transmit the Poll Response Message as identified in the `SET_POLL_RESPONSE` function depending upon which of the two modes it is functioning under. The two functional modes are:

1. Polling Mode where the Poll Response Message begins within 1.42 ms after receipt of a valid Poll Message. (Start of transmission is in 1.42 ms) Refer to SAE J2740 for more details.
2. Non-Polling Mode where the Poll Response Message is sent out immediately.

Specifying a `Poll_ID` parameter value of zero puts `BECOME_MASTER` into Non-Polling mode, while a `Poll_ID` parameter of non-zero puts `BECOME_MASTER` into Polling Mode.

If in Polling Mode, `BECOME_MASTER` will scan incoming messages for one with a Message ID equal to the `Poll_ID` parameter value specified in the `BECOME_MASTER` command. Upon receipt of a valid Poll Message, `BECOME_MASTER` will respond by transmitting the Poll Response Message or if no Poll Message is received within 2 seconds, the function will return to the caller with an `ERR_FAILED` error. In the event that the Poll Response Message was successfully transmitted, the `BECOME_MASTER` function will return `STATUS_NOERROR`.

Refer to Figure 36 for BECOME_MASTER detail.

When a Poll Message is received, it may be either a 4 byte type (containing Device ID) or a 3 byte type (no Device ID).

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during PASSTHRUCONNECT. |
| loctlID | Is set to BECOME_MASTER |
| InputPtr | Points to unsigned char Poll_ID (Value to be compared to Poll Message MIB) |
| OutputPtr | NULL Pointer |

FIGURE 36 - BECOME_MASTER DETAIL

ERROR! REFERENCE SOURCE NOT FOUND. - DEFINES THE GENERALIZED PROCESS FLOW OF OBTAINING BUS MASTERSHIP AND IS INCLUDED FOR INFORMATION ONLY

Most of this logic will be included in the reprogramming application, and does not need to be incorporated into an SAE J2534-2 device and driver.

11.4 Message Structure

11.4.1 Elements

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

11.4.2 Message Data Formats

One additional Protocol has been added and is defined in Figure 37:

| Protocol ID | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|---------------------------|--------|--------|--------|--------|--|
| GM_UART_PS or GM_UART_CHx | 3 | 170 | 3 | 170 | 3 header bytes containing destination ID, source ID and length. Length will be the actual length plus 0x55. Messages can contain 0 data bytes. |

FIGURE 378 - ALLOWED MESSAGE SIZES PER PROTOCOL

11.5 DLL Installation and Registration

Upon device installation, a key will be set in the appropriate Registry folder to indicate the support for the GM UART protocol. In the case of devices which contain dynamic hardware architecture, an application will be able to determine the lack of runtime support when an error is returned from a PassThruConnect().

The registry entries are retained for support of legacy applications. New applications shall use the discovery mechanism defined in this document.

11.6 Discovery Support

The device shall report support for the GM UART protocol and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

12. UART ECHO BYTE PROTOCOL

12.1 Scope of the UART Echo Byte Protocol Optional Feature

This section details the extensions to a SAE J2534-1 interface that will allow support of the UART Echo Byte protocol. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed from SAE J2534-1

12.2 Pass-Thru System Requirements

12.2.1 Simultaneous Communication on Multiple Protocols

Support for simultaneous communication of UART Echo Byte protocol with other protocols shall be equivalent to the definition for ISO9141 in J2534-1.

12.2.2 Pin Usage

VW and Audi use of UART Echo Byte is connected to pin 7 of the SAE J1962 connector.

As with all SAE J2534-2 optional protocols, no default pin is identified. Therefore, the application developer will be required to set the Pin to be used.

See the J1962 Pin Selection section of J2534-2 for details of the method used to switch J1962 pins.

12.3 Win32 Application Programming Interface

12.3.1 API Functions – Overview

Feature Implementation Summary:

This feature support a UART based protocol that uses an ISO9141 K line physical layer interface.

Reference SAE J2809 Honda ABS/VSA or SAE J2818 KWP 1281 for the detailed protocol information.

ECU wakeup is done by a variation of the ISO9141-2 5 baud wakeup sequence, where the transmission of the inverse of the wakeup address by the ECU is replaced by the transmission of an ECU ID message.

A key feature of the protocol is the byte echo scheme employed for every tester and ECU transmission. Following the transmission of each byte from the source device, the destination device responds with the complement of the received byte. This interleaved sequence continues until the complete message has been transmitted by the source device.

The guidelines to implement this communication capability to SAE J2534-2 API are shown below:

- The interface shall receive and check the echo bytes returned from ECU during transmitting request message.
- Echoed bytes received from the ECU during a transmission by the interface shall not be returned to the application.
- Echo bytes generated by the interface when receiving a message from the ECU shall not be returned to the application.
- When any error is detected in transmitting a request message as defined in the Error handling during Normal Communication sections of SAE J2809 or SAE J2818, the interface shall retransmit the failed message. If two subsequent retransmissions fail, the error shall be indicated to the application (i.e. ERR_TIMEOUT is returned in case of a blocking write). To account for this, the application should set the PassThruWriteMsgs timeout to a value that accounts for the possibility of the three transmission attempts by the interface.
- The interface shall automatically transmit the echo bytes when receiving a response message.
- For response messages, if the interface detects ETX (the last byte) in response message lost, the interface shall wait for two times UEB_T7_MAX to receive possible repeat transmission of the response message from the ECU. Following two times the value of UEB_T7_MAX expiry, the response message with absent ETX shall be queued (in this case the application will detect loss of ETX and request the interface to transmit a No Acknowledge message).
- For a message to be valid, bytes at least upto ETX (excluding ETX) must be received. Otherwise the message will be discarded.
- The message counter (the second byte of messages) shall be generated and controlled by the application.
- The application is responsible for handling received No Acknowledge messages from the ECU and repeating transmission of the requested message.

- The interface shall discard any bytes received unexpectedly, i.e. outside the bounds of an in-progress receive message.

The interface is not required to transmit Acknowledge messages to the ECU, except in the case of the ECU ID receive mechanism described in "FIVE_BAUD_INIT" Section.

Messages passed to PassThruWriteMsgs by the application will include all message data bytes, including ETX. All received message bytes including ETX shall be queued by the interface for retrieval by calls to PassThruReadMsgs.

Support for the UART Echo Byte protocol is achieved by the addition of a new ProtocolID, modified behavior of FIVE_BAUD_INIT and a new low level implementation for transmission and reception of messages. This protocol identified as UART_ECHO_BYTE_PS or UART_ECHO_BYTE_CHx requires additional API and device software support, but no additional hardware compared with a standard J2534-1 device.

If UART Echo Byte protocol feature is not supported, ERR_NOT_SUPPORTED will be returned by the call PassThruConnect. The calling application will be required to notify the user that this optional feature may not be supported by the interface.

The J2534-2 Repeat Messaging feature shall not be supported on this protocol as it is not feasible for the interface to correctly maintain MessageCounter values.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|--------------------------|--|
| PassThruConnect | Added new ProtocolID values. |
| PassThruWriteMsgs | This function only accepts one message at a time. i.e. pNumMsgs must always be 1 |
| PassThruStartPeriodicMsg | This function is not supported |
| PassThruIoctl | FIVE_BAUD_INIT is enhanced. New configuration parameters are added. |

FIGURE 38 - SAE J2534 API FUNCTIONS

NOTE: There is no change to PassThruStartMsgFilter function. However, the interface shall participate in the echoed byte receive process for all received messages, regardless of any filter settings. Standard J2534-1 filtering shall be applied to all completed messages received by the interface.

12.3.2 API Functions - Detailed Information

12.3.2.1 PassThruConnect

When PassThruConnect is called with the UART_ECHO_BYTE_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

12.3.2.1.1 C / C++ Prototype

There are no changes defined for the function prototype.

12.3.2.1.2 Parameters

There are no changes to the Parameters.

12.3.2.1.3 Connect Flag Values

There are no changes to the Connect Flags

12.3.2.1.4 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|--------------------|---|
| UART_ECHO_BYTE_PS | Honda ABS/VSA described in SAE J2809 or KWP1281 described in SAE J2818 with pin selection |
| UART_ECHO_BYTE_CHx | Additional channels of Honda ABS/VSA described in SAE J2809 or KWP1281 described in SAE J2818 |

FIGURE 40 - PROTOCOL ID VALUES

12.3.2.2 PassThruWriteMsgs

12.3.2.2.1 C / C++ Prototype

There are no changes defined for the function prototype.

12.3.2.2.2 Parameters

Due to the MessageCounter being incremented alternately by tester and ECU, multiple messages cannot be transmitted by a single call to PassThruWriteMsgs. Therefore, the only acceptable value for *pNumMsgs is 1.

If value pointed to by pNumMsgs is greater than 1, ERR_EXCEEDED_LIMIT shall be returned.

12.3.2.3 PassThruStartPeriodicMsg

This function is not supported on this protocol as it is not feasible for the interface to correctly maintain MessageCounter values. If this function is called, ERR_NOT_SUPPORTED shall be returned.

12.3.3 IOCTL Section

12.3.3.1 SET_CONFIG and GET_CONFIG additional parameters.

Ten new configuration parameters will be added to support UART Echo Byte protocol.

| J2534-2 Parameter | VALID VALUES | DEFAULT VALUES | Description |
|----------------------|--------------------------------|-------------------|---|
| UEB_T0_MIN | 0X0 – 0XFFFF (1 ms per bit) | 10 | Minimum idle time before the start of transmission of the address byte. |
| UEB_T1_MAX | 0X0 – 0XFFFF (1 ms per bit) | 400 | Maximum time between correct stimulation and start of the synchronization byte |
| UEB_T2_MAX | 0X0 – 0XFFFF (1 ms per bit) | 200 | Maximum time between synchronization byte and key-byte 1 |
| UEB_T3_MAX | 0X0 – 0XFFFF (1 ms per bit) | 200 | Maximum time between key-byte 1 and key-byte 2 |
| UEB_T4_MIN | 0X0 – 0XFFFF (1 ms per bit) | 1 | Minimum time between key-byte 2 and complement key-byte 2 |
| UEB_T5_MAX | 0X0 – 0XFFFF (1 ms per bit) | 1000 | Maximum time between key-byte 2 and the renewed output of the synchronization byte (If complement key-byte 2 incorrectly received by control unit) |
| UEB_T6_MAX | 0X0 – 0XFFFF (1 ms per bit) | 200 | Maximum time between key-byte 2 and start of ECU identification |
| UEB_T7_MIN | 0X0 – 0XFFFF (1 ms per bit) | 1 | Minimum time between bytes within a message (i.e. between a byte from the ECU and its complement from the interface) |
| UEB_T7_MAX | 0X0 – 0XFFFF (1 ms per bit) | 40 | Maximum time between bytes within a message (i.e. between a byte from the interface and its complement from the ECU) |
| UEB_T9_MIN | 0X0 – 0XFFFF (1 ms per bit) | 1 | Minimum time between end of a message and start of the next message |

FIGURE 39 - SET_CONFIG AND GET_CONFIG ADDITIONAL PARAMETERS

The DATA_RATE parameter shall also be supported and the default value shall be 9600 bps.

No other J2534-1 defined configuration parameters are supported with the exception of J1962_PINS.

This table translates the UART Echo Byte protocol configuration parameters to the SAE J2809 and SAE J2818 parameters.

| J2534-2 Parameter | J2809 PARAMETER | J2818 PARAMETER | Description |
|------------------------------|----------------------------|----------------------------|---|
| UEB_T0_MIN | t_0 | T_R0 | Minimum idle time before the start of transmission of the address byte. |
| UEB_T1_MAX | t_1 | t_r1 | Maximum time between correct stimulation and start of the synchronization byte |
| UEB_T2_MAX | t_2 | t_r2 | Maximum time between synchronization byte and key-byte 1 |
| UEB_T3_MAX | t_3 | t_r3 | Maximum time between key-byte 1 and key-byte 2 |
| UEB_T4_MIN | t_4 | t_r4 | Minimum time between key-byte 2 and complement key-byte 2 |
| UEB_T5_MAX | t_5 | t_rkmax | Maximum time between key-byte 2 and the renewed output of the synchronization byte (If complement key-byte 2 incorrectly received by control unit) |
| UEB_T6_MAX | t_6 | t_r5 | Maximum time between key-byte 2 and start of ECU identification |
| UEB_T7_MIN | t_7 minimum | t_r6 | Minimum time between bytes within a message (i.e. between a byte from the ECU and its complement from the interface) |
| UEB_T7_MAX | t_7 maximum | t_r8 max | Maximum time between bytes within a message (i.e. between a byte from the interface and its complement from the ECU) |
| UEB_T9_MIN | t_9 | t_rb | Minimum time between end of a message and start of the next message |

FIGURE 40 - PARAMETER TRANSLATION TABLE

12.3.3.2 FIVE_BAUD_INIT

The interface to the IOCTL function is identical to the definition in J2534-1. However, as described in the Telegram-Specific Overview section of SAE J2809 or Stimulation and Initialization section of SAE J2818, the transmission of the Key Byte 2 is followed by the transmission of a two-part ECU ID message. The ECU ID message following 5 baud initialization shall be received and checked for correct timing by the interface, before indicating success of the initialization process. Then the ECU ID message shall be discarded by the interface.

Note that that interface shall not use the synchronization byte to set the baud rate. The baud rate shall be as specified by the DATA_RATE parameter. The interface shall not support the initialization retry strategy initiated by the control unit's retransmission of the synchronization byte, described in the Communication Set-Up section of SAE J2809 or SAE J2818.

The sequence of the ECU ID message output is shown below. Note that the echoed complement bytes are omitted in the example data streams below:

- First part of the identification message is transmitted by the ECU

Example: 0x0F 0x01 0xF6 0x30 0x32 0x36 0x35 0x39 0x35 0x30 0x30 0x31 0x34 0x20 0x20 0x03

- The interface sends an Acknowledge message.

0x03 0x02 0x09 0x03

- The ECU sends the second part of the identification message.

Example: 0x0B 0x03 0xF6 0x42 0x42 0x20 0x32 0x39 0x30 0x36 0x30 0x03

- The interface sends a second Acknowledge message.

0x03 0x04 0x09 0x03

- The ECU then sends an Acknowledge message in response.

0x03 0x05 0x09 0x03

- This signals the end of the sequence.

The interface is only required to generate Acknowledge messages in the specific situation described above. In all other situations, the application generates the Acknowledge messages. Note that the interface must generate the Message Counter field in both Acknowledge messages described above, and the values shall be 2 and 4.

12.4 Message Structure

12.4.1 C / C++ Definition

There is no change to the C/C++ Definition.

12.4.2 Message Data Formats

When using UART Echo Byte protocol, all message bytes including ETX are defined by the application.

All received messages that comply with the timing and message structure requirements shall be queued to the application.

UART Echo Byte Protocol IDs and message limitations are defined below:

| Protocol ID | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|--|--------|--------|--------|--------|--|
| UART_ECHO_BYTE_PS or UART_ECHO_BYTE_CHx | 4 | 256 | 3 | 256 | Data[0] = byte message length Data[1] = message counter Data[2] = message title Data[3 – (n-1)] = Optional message specific data Data[n] = ETX All application defined |

FIGURE 41 - ALLOWED MESSAGE SIZES PER PROTOCOL

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

12.4.3 Format Checks for Messages Passed to the API

The vendor DLL shall validate all PASSTHRU_MSG structures, and return an error, ERR_INVALID_MSG if DataSize violates Min Tx or Max Tx columns in Figure 41.

12.4.4 Message Flag and Status Definitions

No TxFlags, RxStatus, or Indications are supported for this protocol. These flags are undefined for this protocol.

12.5 Return Value Error Codes

ERR_EXCEEDED_LIMIT will be returned when more than one message is written at a time.

12.6 Discovery Support

The device shall report support for UART Echo Byte protocol and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

13. HONDA DIAG-H PROTOCOL

13.1 Scope of the Honda DIAG-H Optional Feature

This section details the extensions to SAE J2534-1 that will allow support of the Honda 92Hm/2 protocol and DIAG-H protocol interface. DIAG-H is a bi-directional serial communication line that provides diagnostic communication capability using the "92Hm/2" communication protocol which is Honda proprietary. DIAG-H line is a single wire communication bus based on a UART interface.

13.2 Pass-Thru System Requirements

13.2.1 Simultaneous Communication on Multiple Protocols

The interface shall be capable of supporting one of the protocols from data link set DATA LINK SET 1, one of the protocols from DATA LINK SET 2 and one of the protocols from DATA LINK SET 3.

An attempt to open an unsupported protocol channel shall result in the ERR_NOT_SUPPORTED return code from PassThruConnect.

| DATA LINK SET 1 | DATA LINK SET 2 | DATA LINK SET 3 |
|--------------------------------------|-----------------------|------------------|
| HONDA_DIAGH_PS or HONDA_DIAGH_CHx | ISO 9141 ISO 14230 | CAN ISO 15765 |

FIGURE 42 - SIMULTANEOUS COMMUNICATION OPTIONS

13.2.2 Programmable Power Supply

There are no changes to this section.

13.2.3 Pin Usage

When the vehicle is equipped with an SAE J1962 connector, DIAG-H may be absent, or may be present on SAE J1962 Pins 14 or Pin 1. On vehicles that have the CAN_L line present on Pin 14, normal mode communication between ECUs on the CAN bus may be corrupted by connection to a tester DIAG-H circuit. Therefore, the tester's DIAG-H circuit shall be disconnected (500kohm minimum impedance) from Pin 14 until commanded by the diagnostic application.

As with all SAE J2534-2 optional protocols, no default pin is identified. Therefore, the application developer will be required to set the Pin to be used. See the J1962 Pin Selection section of J2534-2 for details of the method used to switch J1962 pins.

13.2.4 Honda Diagnostic Connectors

Figures below show the diagnostic communication line configuration of OBD regulation compliant and non-OBD regulation compliant Honda vehicles.

The DIAG-H protocol shall be supported on SAE J1962 pin 1 and on SAE J1962 pin 14.

Most Honda non-OBD (1992 to 2000 Model Year) vehicles are equipped with 3 pin or 5 pin DLC. The "DIAG-H" bi-directional serial communication line is connected to the DLC and provides serial data stream to the Tester in 92Hm/2 protocol.

The 3 pin and 5 pin DLC vehicles are supported by existing SAE J1962 adaptor cables that route J1962 pin 14 to pin 1 of the 3 pin or 5 pin DLC. (The Honda diagnostic application always assigns DIAG-H to SAE J1962 pin 14 when communicating with these vehicles)

On the other hand, Honda OBD compliant vehicles are equipped with 16 pin DLC conformed to SAE J1962 / ISO 15031-3 for the off-board diagnostic communication. Mainly, 92Hm/2 protocol is used for the Chassis/Body System ECUs.

Note that N.C. in Figure 43 indicates that some 3 pin DLC vehicles do not include battery positive supply on the DLC. In this case, the J2534 interface device must be powered by a separate cable, from the cigar lighter, for example.

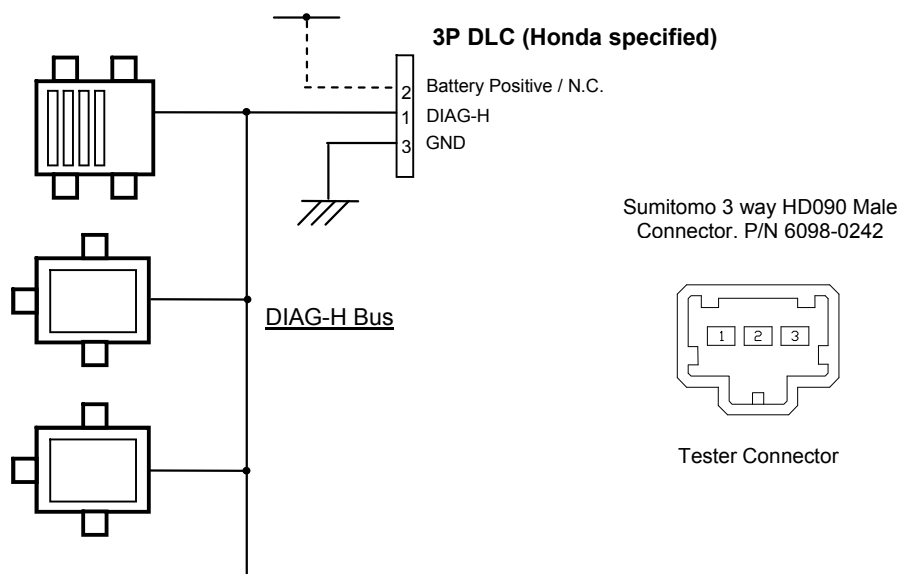


FIGURE 43 - DIAGNOSTIC COMMUNICATION LINE CONFIGURATION (3P DLC MODEL)

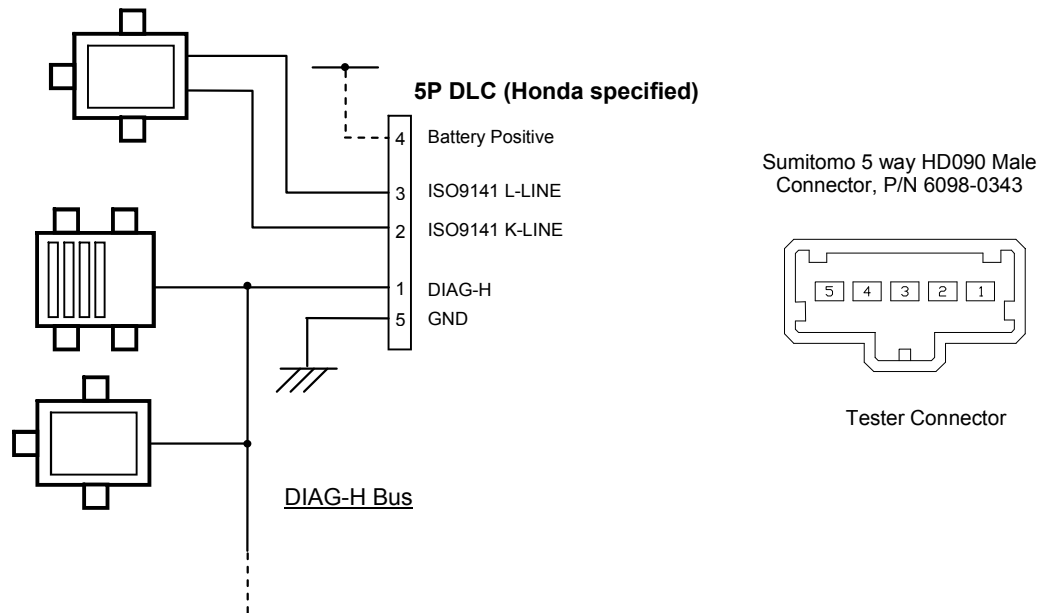


FIGURE 44 - DIAGNOSTIC COMMUNICATION LINE CONFIGURATION (5P DLC MODEL)

The maximum current on a ground pin of Honda 3 pin & 5 pin DLC is 1.5 A. It is equivalent to the specification for signal ground (pin 5) defined in SAE J1962.

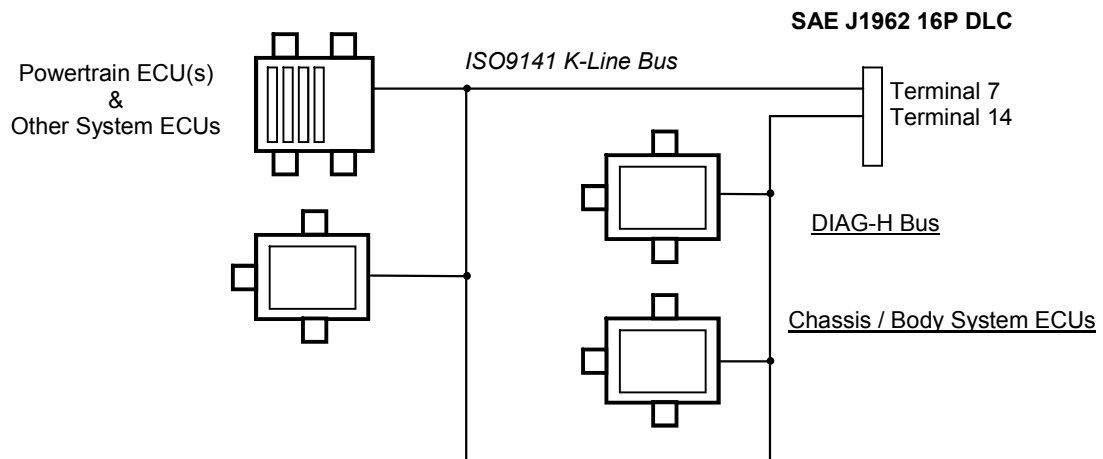


FIGURE 45 - DIAGNOSTIC COMMUNICATION LINE CONFIGURATION (16P DLC MODEL WITHOUT CAN)

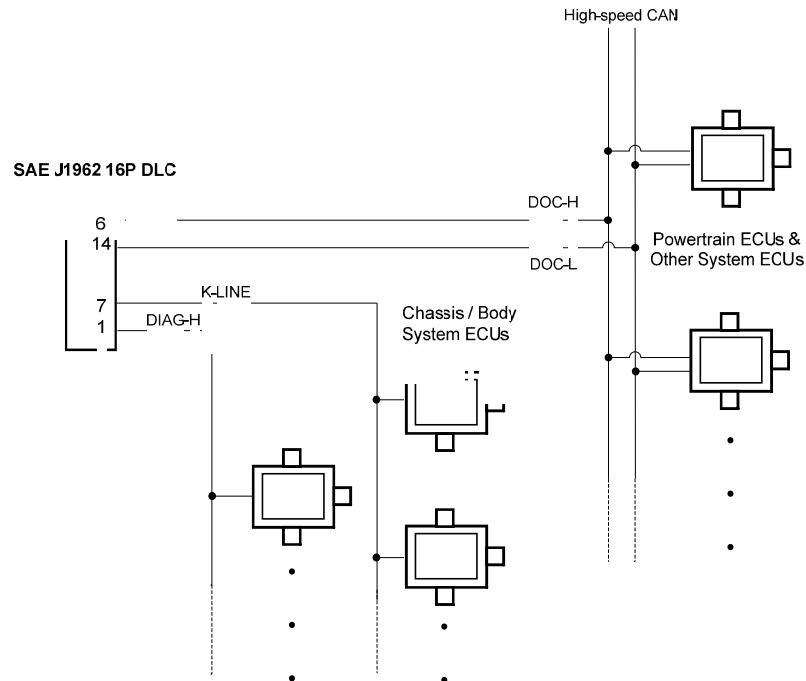


FIGURE 46 - DIAGNOSTIC COMMUNICATION LINE CONFIGURATION (16P DLC MODEL WITH CAN)

13.2.5 Serial Communication Interface / Protocol

Figure "Serial Interface Parameters" describes the serial interface of Honda DIAG-H communication.

| | |
|----------------------------|--|
| <i>Signal Line</i> | "DIAG-H"; Bidirectional, Half - Duplex, Voltage Drive (0 volt - 5 volt) |
| <i>Communication Speed</i> | 9600 bps |
| <i>Bit Encoding</i> | NRZ Asynchronous |
| <i>Byte Frame Format</i> | 1 start bit, 8 data bits, 1 stop bit. No parity bit |
| <i>Network Access</i> | Master / Slave Single Request / Single Response (Any initialization process such as ISO 9141/ -2 is NOT required.) |
| <i>Message Format</i> | Honda 92 Message Format ; Defined by HONDA |

FIGURE 47 - SERIAL INTERFACE PARAMETERS

13.2.6 Electrical Characteristics of Interface

This section states the requirements of the serial communication interface of the off-board diagnostic tester to communicate with the ECU on the Honda vehicles through DIAG-H bus line.

13.2.6.1 Electrical Characteristics

| Item | Requirements | Description |
|-------------------------------|------------------------|--|
| Power Supply Voltage (Vcc) | 5.0V \pm 5% | Voltage of power supply to DIAG-H line. |
| Capacitance (C _T) | Less than 2000pF | Total capacitance of tester and all its cabling (Figure "Recommended Interface Circuit on Tester") |
| Pull-up Resistance | 4.7k Ω \pm 5% | Resistance pull-up to Vcc. |

FIGURE 50 - ELECTRICAL CHARACTERISTICS

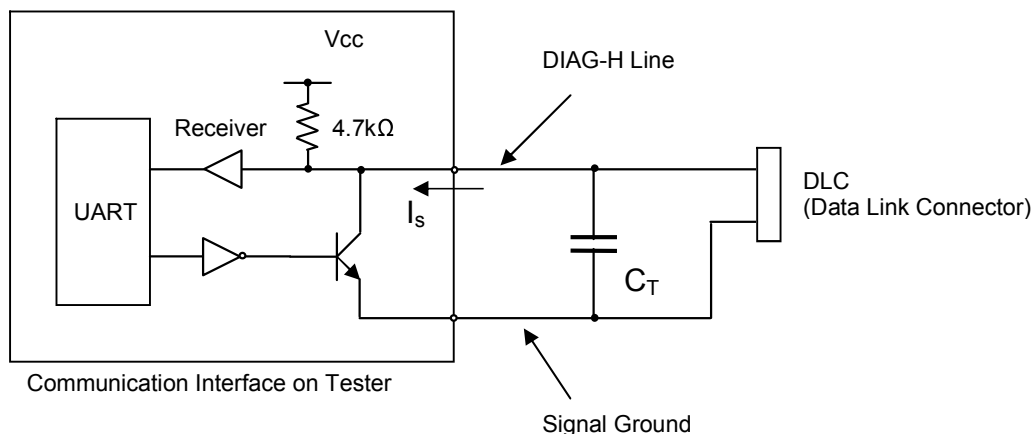


FIGURE 48 - RECOMMENDED INTERFACE CIRCUIT ON TESTER

13.2.6.2 Requirements of Transmitter

| Item | Requirements | Description |
|---------------------------------|----------------|---|
| Logic "0" Voltage | Less than 0.5V | Output voltage at logic "0" |
| Sink Current (I _S) | More than 50mA | Sink Current at logic "0" (Figure 51). |
| Logic "1" Voltage | More than 4.5V | Output voltage at logic "1" with the load shown in Figure "Load Circuit". |

FIGURE 49 - TRANSMITTER ELECTRICAL REQUIREMENTS

| Item | Requirements | Condition |
|------------------------------|----------------------|---------------------------|
| Transmission Speed | 9600 bps \pm 0. 2% | At DLC terminal |
| t _{DST} (Figure 55) | 5.0 s | Disconnected from vehicle |

FIGURE 50 - TRANSMITTER TIMING REQUIREMENTS

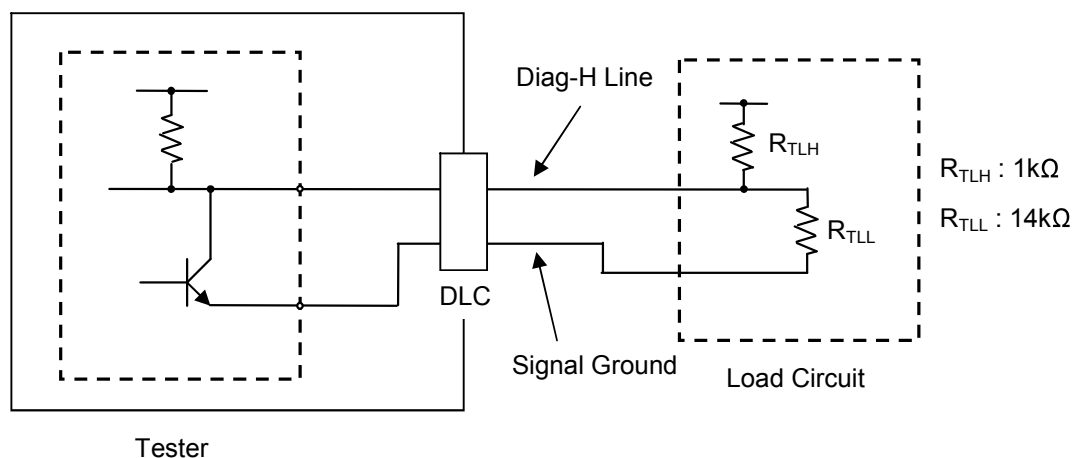
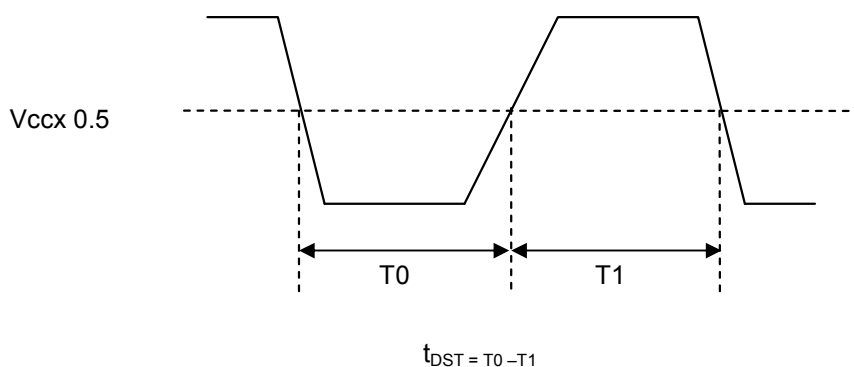


FIGURE 51 - LOAD CIRCUIT



" t_{DST} " is defined as time difference between duration of logic "0" and "1", when "0" and "1" bits are transmitted alternately.

FIGURE 52 - DEFINITION OF T_{DST}

13.2.6.3 Requirements of Receiver

| Item | Requirements | Description |
|---------------------|---|--|
| Logic"0" Threshold | More than 1.9 V | Threshold voltage to determine logic "0". |
| Logic"1" Threshold | Less than 3.1 V | Threshold voltage to determine logic "1" |
| Internal Resistance | More than 100k \pm (to Signal Ground) More than 4.4k Ω (to Vcc) | Internal resistance in receiving state. |
| Hysteresis | More than 0.4V | Difference of threshold voltage between "0" and "1". |

FIGURE 53 - RECEIVER ELECTRICAL REQUIREMENTS

| Item | Requirements | Condition |
|--------------------------------|--------------------|-------------------------|
| Acceptable Communication Speed | 9600bps \pm 1.5% | With wiring and ECUs *1 |
| t _{DST} (Figure 55) | \pm 17.0 s | |

*1 - - - Total Capacitance of Vehicle: Less than 2700pF.
ECU pulled-up resistor to Vcc: 4.7k \pm 5% (Max. 7 units).

FIGURE 54 - RECEIVER TIMING REQUIREMENTS

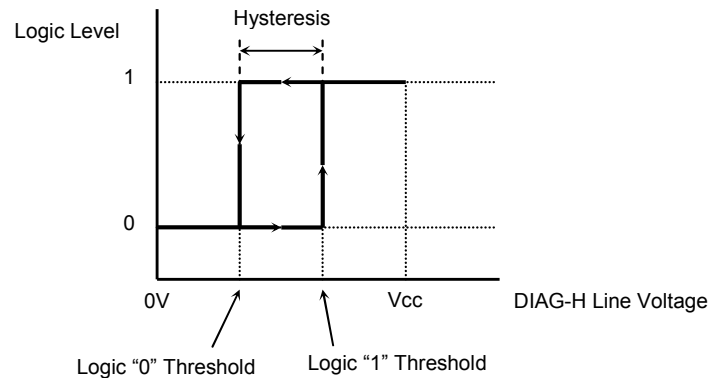


FIGURE 55 - THRESHOLD VOLTAGE

13.3 Win32 Application Programming Interface

13.3.1 API Functions – Overview

Feature Implementation Summary:

The DIAG-H physical interface supports the 92Hm/2 diagnostic communication protocol that is a UART based protocol operating at 9600 baud. The J2534-2 support of this protocol is based on the existing J2534-1 definitions for ISO9141. ISO9141 timing parameters are used. No support for 5 baud or fast initialization is required.

The interface is not required to perform any message transmission retries. Retries shall be handled by the application.

Support for the DIAG-H interface and 92Hm/2 protocol is achieved by the addition of a new physical layer interface in the interface hardware and a new ProtocolID.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|-----------------|------------------------------|
| PassThruConnect | Added new ProtocolID values. |

FIGURE 56 - SAE J2534 API FUNCTIONS

13.3.2 API Functions - Detailed Information

13.3.2.1 PassThruConnect

When PassThruConnect is called with the HONDA_DIAGH_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

13.3.2.1.1 C / C++ Prototype

There are no changes defined for the function prototype.

13.3.2.1.2 Parameters

There are no changes to the Parameters.

13.3.2.1.3 Connect Flag Values

There are no changes to the Connect Flags.

13.3.2.1.4 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|-----------------|--|
| HONDA_DIAGH_PS | Honda DIAG-H Protocol with pin selection |
| HONDA_DIAGH_CHx | Additional channels of Honda DIAG-H Protocol |

FIGURE 60 - PROTOCOL ID VALUES

13.3.2.1.5 Return Values

There are no changes defined for the Return Values.

13.3.3 IOCTL Section

13.3.3.1 SET_CONFIG and GET_CONFIG Supported Parameters

The LOOPBACK, P1_MAX, P3_MIN, P4_MIN, J1962_PINS configuration parameters shall be supported for the protocol HONDA DIAG-H protocol. Attempts to access any other parameters shall result in return of the ERR_INVALID_IOCTL_PARAM_ID error code.

P1_MAX expiry shall be used to detect the end of an ECU response message.

The interface shall not check for P2_MAX violations. All received messages shall be queued to the application regardless of the time since the end of the last transmission by the interface. Transmissions from the interface shall not be allowed within P3_MIN since the end of a received message.

13.4 Message Structure

13.4.1 C / C++ Definition

There is no change to the C/C++ Definition.

13.4.2 Elements

There is no change to any of the elements.

13.4.3 Message Data Formats

When using HONDA_DIAGH_PS, all message bytes are defined by the application.

Honda DIAG-H Protocol IDs and message limitations have been defined below:

| Protocol ID | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|-----------------------------------|--------|--------|--------|--------|--------------------------|
| HONDA_DIAGH_PS or HONDA_DIAGH_CHx | 3 | 255 | 1 | 255 | Data[0 – n] = Data bytes |

FIGURE 57 - ALLOWED MESSAGE SIZES PER PROTOCOL

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

13.4.4 Format Checks for Messages Passed to the API

The vendor DLL shall validate all PASSTHRU_MSG structures, and return an error, ERR_INVALID_MSG if DataSize violates Min Tx or Max Tx columns in "Message Data Formats" section.

13.4.5 Message Flag and Status Definitions

Only the TX_MSG_TYPE RxStatus bit shall be used by this protocol.

No TxFlags or Indications are supported for this protocol. These flags are undefined for this protocol.

13.5 Discovery Support

The device shall report support for HONDA DiagH Protocol and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

14. REPEAT MESSAGING

14.1 Scope of the Repeat Messaging Protocol Optional Feature

This section details the extensions to a SAE J2534-1 interface that will allow support of the Repeat Messaging feature. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed from SAE J2534-1.

14.2 Win32 Application Programming Interface

14.2.1 API Functions – Overview

Feature Implementation Summary:

Some operations on ECUs require a message to be transmitted repeatedly at a rapid and predictable rate until the operation is completed. Completion of the operation is indicated by a change in a byte value in the response message to the repeated request.

This optional feature defines an extension to the API that allows the interface to perform autonomous transmissions based on the content of messages received from a vehicle bus. The configuration of this behavior is performed by the application using new IOCTLs. In this way the feature provides a general purpose, protocol independent method of achieving high performance repeated transmissions while certain conditions persist on a vehicle.

Support of this feature is achieved by three additional PassThruIoctl ioctlID values. The first is used to set-up and start the repeat transmission, the second is used to query the status of a transmission and the third is used to stop the transmission and release resources. The device shall support a minimum of ten repeat messages per channel. Additionally, this feature shall be supported on all protocols unless explicitly excluded by the description of the protocol.

The operation of the feature is in three stages:

1. Specifying the message to transmit, transmission rate and stop criteria via PassThruIoctl START_REPEAT_MESSAGE.
2. Querying to see if the transmission has been stopped automatically via PassThruIoctl QUERY_REPEAT_MESSAGE.
3. Stopping an in-progress transmission and releasing resources via PassThruIoctl STOP_REPEAT_MESSAGE.

In general, repeat messages are almost identical to periodic messages. Repeat messages shall have the same DataSize and TxFlags limitations as periodic messages. Repeat messages shall generate loopback messages, if enabled on the channel. Additionally, repeat messages shall have the same priority as periodic messages. The queueing mechanism used by periodic messages shall be extended to include repeat messages. Finally, repeat messages shall not violate bus idle timing parameters (like P3_MIN) or other protocol specific requirements.

Responses to the repeat messages, including the response that caused termination of the repeat transmission, shall be treated in the same way as other received data and shall be queued to the application, subject to any filters configured on the channel. Incoming messages will be evaluated against the repeat mask/pattern before being processed by the standard Pass or Block filtering.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|---------------|---|
| PassThruIoctl | New IOCTL IDs are added: START_REPEAT_MESSAGE QUERY_REPEAT_MESSAGE STOP_REPEAT_MESSAGE |

FIGURE 58 - SAE J2534 API FUNCTIONS

14.2.2 IOCTL Section

| Value of ioctlID | InputPtr represents | OutputPtr represents | Purpose |
|----------------------|-----------------------------|--------------------------|--|
| START_REPEAT_MESSAGE | Pointer to REPEAT_MSG_SETUP | Pointer to unsigned long | Sets up and starts a repeat message transmission. |
| QUERY_REPEAT_MESSAGE | Pointer to unsigned long | Pointer to unsigned long | Reports the status of a repeat message transmission. |
| STOP_REPEAT_MESSAGE | Pointer to unsigned long | NULL pointer | Terminates an existing repeat message transmission. |

FIGURE 59 - IOCTL DETAILS

14.2.2.1 START_REPEAT_MESSAGE

This function is used to specify and initiate a repeat message. The parameters are specified in Figure 60.

| Parameter | Description |
|-----------|---|
| ChannelID | Channel ID assigned by DLL during PassThruConnect |
| IoctlID | Is set to the define START_REPEAT_MESSAGE |
| InputPtr | <p>Is a pointer to the structure REPEAT_MSG_SETUP which is defined as follows:</p> <pre>typedef struct { unsigned long TimeInterval; /* Repeat transmission rate */ unsigned long Condition; PASSTHRU_MSG RepeatMsgData[3]; } REPEAT_MSG_SETUP;</pre> <p>where:</p> <p>TimeInterval is the time interval between the end of a message transmission and the start of the next time this message is sent, in milliseconds. The valid range is 5-65535 milliseconds.</p> <p>Condition specifies whether the repeat transmission should: 0 = continue until a message is received that matches the pattern 1 = continue while every received message matches the pattern</p> <p>RepeatMsgData[0] is a PASSTHRU_MSG structure that specifies the message to be repeatedly transmitted. The Data, DataSize, and TxFlags elements of the PASSTHRU_MSG structure shall be populated and appropriate for the designated channel.</p> <p>RepeatMsgData[1] is a PASSTHRU_MSG structure that specifies the mask message that will be ANDed with each incoming message to mask any unimportant bits. The Data, DataSize, and TxFlags elements of the PASSTHRU_MSG structure shall be populated and appropriate for the designated channel.</p> <p>RepeatMsgData [2] is a PASSTHRU_MSG structure that specifies the pattern message that will be compared to the incoming message after the mask message has been applied. If the result matches this pattern message, the designated action shall be taken. The Data, DataSize, and TxFlags elements of the PASSTHRU_MSG structure shall be populated and appropriate for the designated channel. Additionally, the DataSize, and TxFlags elements shall match that of the mask message.</p> |
| OutputPtr | Is a pointer to an unsigned long, MsgId, the message ID that is assigned by the DLL. |

FIGURE 60 - START_REPEAT_MESSAGE DETAILS

If this function call is successful the return value shall be STATUS_NOERROR and the associated repeat message shall have been created (even if it duplicates an existing repeat message) and one copy shall be placed in the repeat message transmit queue.

If Condition = 0 (REPEAT_MESSAGE_UNTIL_MATCH), the repeat message transmissions shall continue until a message is received that, after ANDing with the mask message, matches the pattern message exactly. Additionally, repeat message transmissions shall continue if the TimeInterval expires without receiving any message.

If Condition = 1 (REPEAT_MESSAGE_WHILE_MATCH), the repeat message transmissions shall continue while **every** received message that, after ANDing with the mask message, matches the pattern message exactly. The repeat message transmissions shall cease when a message is received that, after ANDing the with the mask message, doesn't match the pattern exactly. Additionally, repeat message transmissions shall terminate if the TimeInterval expires without receiving any message.

ERR_EXCEEDED_LIMIT shall be returned if more than the maximum number of repeat messages supported by the interface is requested. ERR_INVALID_MSG shall be returned if the data size of an individual message exceeds that defined in SAE J2534-1 for periodic messages. ERR_INVALID_MSG shall also be returned if the TxFlags do not match those specified in SAE J2534-1 for periodic messages.

If this IOCTL is not supported by the interface a ERR_NOT_SUPPORTED shall be returned.

NOTE: Only the first RepeatMsgData[1].DataSize bytes of each incoming message shall be evaluated (i.e., additional bytes in a message longer than DataSize shall be ignored and treated as "don't care"). Incoming messages whose DataSize is less than that of the pattern message shall be considered not to match (even if the mask bytes indicated 'Don't Care').

Examples

Application sets up a repeat message:
0x64 0x28 0xF5 0x22 0x02 0x00

Normal ECU response is:
0x74 0xF5 0x28 0x62 0x02 0x00 0x00

The application requires the transmission to terminate when the following is received:
0x74 0xF5 0x28 0x62 0x02 0x00 0x01

Example #1:

The filtering arrays could be set up as:

| | |
|------------------|------------------------------------|
| Condition | 0 (REPEAT_MESSAGE_UNTIL_MATCH) |
| Mask | 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF |
| Pattern | 0x74 0xF5 0x28 0x62 0x02 0x00 0x01 |

This would require all bytes in the received message to match the Pattern specified above for transmission to stop.

If there were a sequence number that changed in the first byte every time, the following could be used to ignore the value of the changing sequence number byte:

| | |
|------------------|------------------------------------|
| Condition | 0 (REPEAT_MESSAGE_UNTIL_MATCH) |
| Mask | 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF |
| Pattern | 0x00 0xF5 0x28 0x62 0x02 0x00 0x01 |

Example #2:

If there are several responses that are required to cause the transmission to terminate, for example.:

0x74 0xF5 0x28 0x62 0x02 0x00 0x01

0x74 0xF5 0x28 0x62 0x02 0x00 0x52

0x74 0xF5 0x28 0x62 0x02 0x00 0x81

Then the following setup could be used to continue transmission while the Normal ECU response alone continues to be received:

| | |
|------------------|------------------------------------|
| Condition | 1 (REPEAT_MESSAGE_WHILE_MATCH) |
| Mask | 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF |
| Pattern | 0x74 0xF5 0x28 0x62 0x02 0x00 0x00 |

14.2.2.2 QUERY_REPEAT_MESSAGE

This function is used to check the status of the designated repeat message. The parameters are specified in Figure 61.

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during PassThruConnect |
| IoctlID | Is set to the define QUERY_REPEAT_MESSAGE |
| InputPtr | Is a pointer to an unsigned long, MsgId, the message ID of the repeat message transmission to be queried. |
| OutputPtr | Is a pointer to an unsigned long, Status, the current status of the specified repeat message transmission. |

FIGURE 61 - QUERY_REPEAT_MESSAGE DETAILS

If this function call is successful the return value shall be STATUS_NOERROR and the value of the Status parameter shall be set as follows:

| Condition | Status value |
|---|--------------|
| Repeat message transmission in progress, filter criteria not yet met. | TRUE (1) |
| Filter criteria met and repeat transmission ceased. | FALSE (0) |

FIGURE 62 - STATUS PARAMETER VALUES

If this function is called with an invalid MsgID, ERR_INVALID_MSG_ID shall be returned.

14.2.2.3 STOP_REPEAT_MESSAGE

This function is used to terminate a defined repeat transmission and release the specified MsgID. The parameters are specified in Figure 63.

| Parameter | Description |
|-----------|---|
| ChannelID | Channel ID assigned by DLL during PassThruConnect |
| IoctlID | Is set to the define STOP_REPEAT_MESSAGE |
| InputPtr | Is a pointer to an unsigned long, MsgId, the message ID of the repeat message transmission to be stopped. |
| OutputPtr | Is a NULL pointer, as this parameter is not used. |

FIGURE 63 - STOP_REPEAT_MESSAGE DETAILS

If this function call is successful the return value shall be STATUS_NOERROR, the MsgID shall be invalid, the interval timers shall be stopped, and the associated repeat message shall be removed from the transmit queue (if one is present). This function will not terminate the associated repeat message if it is currently being sent.

It should be noted that the MsgID shall not be released automatically on termination of the repeat transmission due to receipt of a message matching the filter conditions. Therefore, a call to STOP_REPEAT_MESSAGE shall always be required.

If this function is called with an invalid MsgID, ERR_INVALID_MSG_ID shall be returned.

14.3 Discovery Support

The device shall report support for Repeat Messaging and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

15. EXTENDED PROGRAMMING VOLTAGE SUPPORT

15.1 Scope of the Extended Programming Voltage Feature

This section details the additional programming voltage features supported by J2534-2

15.2 Pass-Thru System Requirements

15.3 Win32 Application Programming Interface

15.3.1 API Functions – Overview

The following table summarizes the changes to the SAE J2534-1 API function

| Function | Description |
|-------------------------------|--|
| PassThruSetProgrammingVoltage | Support Short-to-Ground feature on Pin 9 |

FIGURE 64 - SAE J2534 API FUNCTIONS

15.3.2 API Functions - Detailed Information

15.3.2.1 PassThruSetProgrammingVoltage

In addition to Pin 15 supported in SAE J2534-1, Short-to-Ground will also be allowed on Pin 9. Pin 9 Short-to-Ground feature is not mutually exclusive with the other programming pins, and can be used while supplying programming voltage through other supported pins.

Short-to-Ground feature shall be supported on only one pin at a time.

ERR_PIN_IN_USE shall be returned if:

- An attempt is made to set a programming voltage on pin 9 while it is at ground
- An attempt is made to ground pin 9 while it has a programming voltage applied

ERR_VOLTAGE_IN_USE shall be returned if:

- An attempt is made to ground pin 9 while pin 15 is grounded.
- An attempt is made to ground pin 15 while pin 9 is grounded.

ERR_NOT_SUPPORTED shall be returned if:

- The device does not support Pin 9 Short-to-Ground feature

15.4 Discovery Support

The device shall report Extended Programming Voltage Support and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

16. J1939 PROTOCOL

16.1 Scope of the J1939 Protocol Optional Feature

This section details the extensions to SAE J2534-1 that will allow support of the SAE J1939 protocol. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed.

This section specifically defines the common method of supporting the SAE J1939 Heavy-Duty vehicle protocol as defined in SAE J1939, titled "Recommended Practice for a Serial Control and Communications Vehicle Network".

The J2534-2 implementation will implement single frame messages and multiple frame messages using both the J1939 BAM process and the Connection Management message process. See SAE J1939-21 for details on multi frame messages. In addition, the J2534-2 device will support the Claim J1939 Address Process outlined in SAE J1939-81.

The BAM message process will be used when the J2534-2 device has a claimed address and a message to be transmitted or received has more than 8 data bytes and is being sent to or received from the Global Address (0xFF). To send a message using the BAM process fill in the Destination Address field with the Global Address.

The Connection Management message process will be used when the J2534-2 device has a claimed address and a message with more than 8 data bytes is being sent to a specific destination address or a message is being sent from the vehicle to the J2534-2 device's claimed address. To send a message using the Connection Management process fill in the Destination Address field with a specific address.

16.2 Pass-Thru System Requirements

16.2.1 Connection to Vehicle

Vendors will have to provide a custom cable to connect to a J1939 vehicle.

NOTE: Some vehicles that use J1939 operate with 24 Volt electrical systems. This J2534-2 optional feature does not specify changes to the J2534-1 device, thus it is up to the customer and J2534-2 device supplier to insure no damage is done to the J2534-2 device when used with 24 Volt systems.

16.2.2 Communication Protocol

The following features of SAE J1939 must be supported by the PassThru device:

- a. 250 kbps
- b. 29 bit identifiers
- c. BAM and Connection Management message transfers
- d. Protection of 10 source addresses and the NAME associated with each source address.

16.2.3 Simultaneous Communication on Multiple Protocols

The J1939 implementation will have the same simultaneous communication on multiple protocol requirements as the CAN protocol (J1850 and ISO9141 must be able to run simultaneously).

16.3 Win32 Application Programming Interface

16.3.1 API Functions – Overview

Feature Implementation Summary:

At a high level a new ProtocolID has been defined to indicate the use of the J1939 physical layer. The protocol defines a communication link between the Tester and the Electronic Control Unit (ECU). If the feature is not supported, an error code, ERR_NOT_SUPPORTED, will be returned by the call to PassThruConnect. The calling application will be required to notify the user that this optional feature may not be supported by interface.

When the J1939 protocol is opened, the application must use the CAN_29BIT_ID connect flag and set the baud rate to 250K.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|-------------------|---|
| PassThruConnect | Added new ProtocolID values. |
| PassThruWriteMsgs | Added a new return value. |
| PassThruIoctl | New IO control Sub-function and 5 configuration parameters. |

FIGURE 65 - SAE J2534 API FUNCTIONS

16.3.2 API Functions - Detailed Information

16.3.2.1 PassThruConnect

When PassThruConnect is called with the J1939_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

16.3.2.1.1 Connect Flag Values

There are no changes to the Connect Flag Values.

Only CAN_29BIT_ID shall be supported.

16.3.2.1.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|------------|---------------------------------------|
| J1939_PS | J1939 protocol with pin selection |
| J1939_CHx | Additional channels of J1939 protocol |

FIGURE 70 - PROTOCOL ID VALUES

16.3.2.2 PassThruWriteMsgs

Added a new return value.

16.3.2.2.1 Return Values

Figure below shows the new return value for PassThruWriteMsgs:

| Definition | Description |
|-------------------------|---|
| ERR_ADDRESS_NOT_CLAIMED | Returned when a write message has a data payload larger than 8 data bytes and the source address in the message does not match a current claimed address. |

FIGURE 66 - RETURN VALUES

16.3.2.3 PassThruIoctl

16.3.2.3.1 Ioctl ID Values

The new ioctl value PROTECT_J1939_ADDR is added to support the claiming of a J1939 CAN address for the tester (J2534 device).

16.3.3 IOCTL Section

The details on the new IOCTLs available through the PassThruIoctl function are as follows:

| Value of ioctlID | InputPtr represents | OutputPtr represents | Purpose |
|--------------------|-----------------------|----------------------|---|
| PROTECT_J1939_ADDR | Points to SBYTE_ARRAY | NULL Pointer | To direct the pass-thru device to claim and defend the specified address. |

FIGURE 67 - IOCTL DETAILS

16.3.3.1 Configuration Parameters

16.3.3.1.1 SET_CONFIG and GET_CONFIG Supported Parameters

The DATA_RATE, LOOPBACK, BIT_SAMPLE_POINT and SYNC_JUMP_WIDTH configuration parameters will be supported for the J1939 protocol. The default values for J1939 are LOOPBACK = 0, BIT_SAMPLE_POINT = 87.5% and SYNC_JUMP_WIDTH = 6.25% (1Tq).

16.3.3.1.2 Additional SET_CONFIG and GET_CONFIG Parameters

Figure below shows the five new configuration parameters will be added to support the J1939 protocol.

| Parameter | Valid values for Parameter | Default Value (decimal) | Description |
|------------------------|------------------------------------|-------------------------|--|
| J1939_T1 | 0X0 – 0X0000FFFF (1 MS PER BIT) | 750 | For Protocol J1939, this sets the maximum time the device waits for the next data frame in a multi frame transfer. |
| J1939_T2 | 0X0 – 0X0000FFFF (1 ms per bit) | 1250 | For Protocol J1939, this sets the maximum time a receiver will wait for the next data frame after sending a CTS in a multi frame transfer. |
| J1939_T3 | 0X0 – 0X0000FFFF (1 ms per bit) | 1250 | For Protocol J1939, this sets the maximum time the device waits for CTS in a multi frame transfer. |
| J1939_T4 | 0X0 – 0X0000FFFF (1 ms per bit) | 1050 | For Protocol J1939, this sets the maximum time the device waits for the next CTS after receiving a CTS with a request for zero frames in a multi frame transfer. |
| J1939_BRDCST_MIN_DELAY | 0X0 – 0X0000FFFF (1 ms per bit) | 50 | For Protocol J1939, this sets the minimum time between data frames in a multi frame broadcast transmission. |

FIGURE 68 - IOCTL GET_CONFIG / SET CONFIG PARAMTERS DETAILS

16.3.3.2 PROTECT_J1939_ADDR

The ioctlID parameter of PROTECT_J1939_ADDR is used to set the J1939 source address to defend and to assign the PassThru device's J1939 NAME as defined in J1939-81. The PassThru device must be able to protect 10 source addresses.

| Parameter | Description |
|-----------|---|
| ChannelID | CHANNEL ID ASSIGNED BY DLL DURING PASSTHRUCONNECT. |
| ioctlID | Is set to PROTECT_J1939_ADDR |
| InputPtr | Points to the structure SBYTE_ARRAY, which is defined as follows: Typedef struct { unsigned long NumOfBytes; /* number of bytes in array */ unsigned char *BytePtr; /* array of bytes */ } where: NumOfBytes is an input that indicates the number of bytes in the array BytePtr. It should always be 9 for PROTECT_J1939_ADDR. BytePtr[0] is the source address to claim (0 – 253), 254 is NOT allowed, 255 = is the default and means no address is claimed. BytePtr[1] – BytePtr[8] is the J1939 defined NAME. BytePtr[1] is the LSB as defined in J1939-81 (the LSB of the Identity Number). |
| OutputPtr | NULL Pointer |

FIGURE 69 - PROTECT_J1939_ADDR DETAILS

This call is non-blocking. When an address claim is successful a J1939_ADDRESS_CLAIMED indication will be received. A J1939_ADDRESS_LOST indication will be received if the address claim was unsuccessful.

If BytePtr[0] equals 254 or 255 the function will return ERR_INVALID_IOCTL_VAULE.

To cancel a protected address, BytePtr[0] equals the source address you want canceled and all the NAME values (BytePtr1 through BytePtr[8]) shall all be zero.

16.4 Message Structure

16.4.1 C / C++ Definition

There is no change to the C/C++ Definition.

16.4.2 Elements

There is no change to any of the elements.

16.4.3 Message Data Formats

When using the J1939 protocol, the first 5 bytes of Data contain the 29 bit CAN ID used in J1939 and the Destination Address. Data [0] contains CAN ID bits 28-24 (the three most significant bits will be zero), Data[1] contains bits 23-16, Data [2] contains bits 15-8, and Data [3] contains bits 7-0 and Data[4] contains the Destination Address. It is up to the application to pack this data correctly (as specified in J1939-21). Additional Protocol IDs and message limitations have been defined below:

| Protocol ID | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|-----------------------|--------|--------|--------|--------|--|
| J1939_PS or J1939_CHx | 5 | 1790 | 5 | 1790 | 4 bytes of CAN ID, 1 byte destination address followed by up to 1785 data bytes. |

FIGURE 70 - ALLOWED MESSAGE SIZES PER PROTOCOL

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

16.4.4 Format Checks for Messages Passed to the API

For messages with less than 9 data bytes, the destination Address (Data[4]) is a don't care and is not used in sending the message.

For messages with more than 8 data bytes, if the J2534-2 device has a claimed address and the destination address (Data[4]) is the Global address (0xFF), the BAM message process will be used to send the message.

For messages with more than 8 data bytes, if the J2534-2 device has a claimed address and the message is being sent to a specific destination address (Data[4]), the Connection Management message process will be used to send the message.

16.4.5 Conventions for Returning Messages from the API

Received messages with less than 9 data bytes, the destination address (Data[4]) will be filled in with the specific destination address of the message or the global Address.

Received messages greater than 8 bytes will be returned to the API when the J2534-2 device has a claimed address and a BAM message is received. Data[4] will contain the Global Address.

Received messages greater than 8 bytes will be returned to the API when the J2534-2 device has a claimed address and a Connection Managed message was sent to the J2534-2 device's claimed address. Data[4] will contain the claimed address that the message was received on.

16.4.6 Message Flag and Status Definitions

Supported Message Status (RxStatus) Definitions.

The TX_MSG_TYPE and CAN_29BIT_ID RxStatus bits will be supported. For J1939, loopback messages returned by the API will have TX_MSG_TYPE set in RxStatus. Messages with a 29 bit ID field will have CAN_29BIT_ID set.

There are two additions to the Message Status (RxStatus) definitions to be used as indications.

| Definition | RxStatus Bit(s) | Description | Value |
|-----------------------|-----------------|---|---|
| J1939_ADDRESS_LOST | 17 | Address Lost status indication will be sent when a claimed Address and Name has been lost or when an attempt to claim an Address and Name fails. The source address lost will be in Data[0] of the message. ExtraDataIndex = 0, DataSize = 1. J1939 only. | 0 = no Address lost indication 1 = Address lost or claim attempt failed. |
| J1939_ADDRESS_CLAIMED | 16 | Address claimed status indication will be sent when an Address and Name is successfully claimed. The source address claimed will be in Data[0] of the message. ExtraDataIndex = 0, DataSize = 1. J1939 only | 0 = no Address claimed indication 1 = Address claimed was successful. |

FIGURE 71 - RXSTATUS BIT DEFINITIONS

The CAN_29BIT_ID TxFlags bit is the only TxFlags bit supported. When sending a message with a 29 bit ID field, set the TxFlags bit CAN_29BIT_ID.

16.5 Return Value Error Codes

ERR_ADDRESS_NOT_CLAIMED will be returned from PassThruWriteMsgs when a message has a data payload larger than 8 data bytes and the source address in the message does not match a current claimed address.

16.6 Discovery Support

The device shall report support for the J1939 protocol and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

17. J1708 PROTOCOL

17.1 Scope of the J1708 Protocol Optional Feature

This section details the extensions to SAE J2534-1 that will allow support of the J1708 protocol. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed.

17.2 Pass-Thru System Requirements

17.2.1 Connection to Vehicle

Vendors will have to provide a custom cable to connect to a J1708 vehicle.

NOTE: Some vehicles that use J1708 operate with 24 Volt electrical systems. This J2534-2 optional feature does not specify changes to the J2534-1 device, thus it is up to the customer and J2534-2 device supplier to insure no damage is done to the J2534-2 device when used with 24 Volt systems.

17.2.2 Communication Protocol

The following features of J1708 must be supported by the PassThru device:

- 9600 bps
- Priority message delay.

17.2.3 Simultaneous Communication on Multiple Protocols

The J1708 protocol implementation will have the same simultaneous communication on multiple protocol requirements as the ISO9141 protocol (J1850 and CAN must be able to run simultaneously with SAE J1708).

17.3 Win32 Application Programming Interface

17.3.1 API Functions – Overview

Feature Implementation Summary:

Support for the J1708 protocol is achieved by the addition of a new physical layer interface and a new ProtocolID. In addition, the J2534-2 device vendor may provide a Heavy Truck cable or adapter.

The protocol feature of calculating and appending the message checksum is required.

The protocol feature of Priority message delay for transmitted messages will be implemented by passing the Priority message delay to the PassThru device in the PassThru message TxFlags.

Reference SAE J1708 for details of communication interface byte encoding and the electrical interface design and characteristics.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|-----------------|------------------------------|
| PassThruConnect | Added new ProtocolID values. |

FIGURE 72 - SAE J2534 API FUNCTIONS

17.3.2 API Functions - Detailed Information

17.3.2.1 PassThruConnect

When PassThruConnect is called with the J1708_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

17.3.2.1.1 Connect Flag Values

The J1708 protocol uses the following J2534-1 Connect Flag.

| Definition | Description |
|-------------------|---|
| CHECKSUM_DISABLED | 0 = The interface will generate and append the checksum as defined in J1708 for transmitted messages. The interface will verify the checksum for received messages and place it in the data section pointed to by ExtraDataIndex. 1 = The interface will not generate and verify the checksum-the entire message will be treated as data by the interface. |

FIGURE 73 - CONNECT FLAG VALUES

17.3.2.1.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following figure. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|------------|---------------------------------------|
| J1708_PS | J1708 Protocol with pin selection |
| J1708_CHx | Additional channels of J1708 Protocol |

FIGURE 74 - PROTOCOL ID VALUES

17.3.2.2 Configuration Parameters

17.3.2.2.1 SET_CONFIG and GET_CONFIG Supported Parameters

The DATA_RATE and LOOPBACK configuration parameters will be supported for the J1708 protocol. The default value for LOOPBACK will be 0.

17.4 Message Structure

17.4.1 C / C++ Definition

There is no change to the C/C++ Definition.

17.4.2 Elements

There are no changes to any of the elements.

17.4.3 Message Data Formats

When transmitting messages, the Transmit Message Priority will be encoded in 4 TxFlags bits. The J2534-2 device will calculate the Check Sum and send it after the last data byte of the message unless the CHECKSUM_DISABLED Connect Flag was set on the PassThruConnect, in which case the application is responsible for calculating the Check Sum and placing it as the last data byte of the message. (See the SAE J1708 for instructions on how to calculate the Check Sum.)

For received messages, when the CHECKSUM_DISABLED flag is not set, the Check Sum will be calculated as per SAE J1708 and compared to the last byte in the data stream. If the Check Sum is correct, the message will be place in the receive queue with ExtraDataIndex pointing to the Check Sum. If the Check Sum does not verify, the message will be discarded. If CHECKSUM_DISABLED was set on the PassThruConnect, all the data received on the bus will be treated as message data.

| Protocol | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|-----------------------|--------|--------|--------|--------|-------|
| J1708_PS or J1708_CHx | 1 | 4095 | 1 | 4095 | |

FIGURE 80 - ALLOWED MESSAGE SIZES PER PROTOCOL

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

17.4.4 Conventions for Returning Messages from the API

For the J1708 protocol, loopback messages returned by the API will have TX_MSG_TYPE set in RxStatus.

17.4.5 Message Flag and Status Definitions

Changes to TxFlags bits:

| Definition | TxFlags Bit(s) | Description | Value |
|--------------------|----------------|------------------|---|
| MSG_PRIORITY_VALUE | 16 -19 | Message priority | Valid values 1 thru 8 Value of 0 or greater than 8 will be treated as priority 8 |

FIGURE 75 - TXFLAGS BIT DEFINITIONS

If the TX message is looped back to the application MSG_PRIORITY_VALUE must be preserved in the looped back message. If MSG_PRIORITY_VALUE is not valid the device will use the lowest priority (a value of 8) for that message

The TX_MSG_TYPE is the only RxStatus flag supported in J1708. It will be set on all loop back TX messages.

17.5 Discovery Support

The device shall report support for the J1708 protocol and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

18. EXTENDED PASSTHRUIOCTL FOR DEVICE CONFIGURATION PARAMETERS

18.1 Scope of the Extended PassThruIoctl Optional Feature

This section details the extensions to a SAE J2534-1 interface that will allow setting and retrieval of device-wide configuration parameters. Currently, this feature is limited to support of ten non-volatile and re-writable data storage parameters. However, the feature may be extended in future to support additional non-protocol-specific parameters. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed from SAE J2534-1

18.2 Pass-Thru Concept

This optional feature defines an extension to the Pass-Thru concept that allows device-wide parameters to be specified. More specifically, the current definition is for ten data items to be stored on the PassThru interface in a non-volatile storage area (i.e. an area in which data survives disconnection from a power source). Each data item will be 4 bytes long.

18.3 Win32 Application Programming Interface

18.3.1 API Functions – Overview

The PassThruIoctl called with GET_DEVICE_CONFIG and SET_DEVICE_CONFIG ioctlIDs are used to set device parameters and require the DeviceID to be passed as the first PassThruIoctl parameter.

The following table summarizes the changes to the SAE J2534-1 API function:

| Function | Description |
|---------------|---|
| PassThruIoctl | GET_DEVICE_CONFIG and SET_DEVICE_CONFIG are added. Parameters NON_VOLATILE_STORE_1 to NON_VOLATILE_STORE_10 are added. |

FIGURE 76 - SAE J2534 API FUNCTIONS

18.4 IOCTL Section

Figure 77 provides details of the additional ioctlIDs that support this feature:

| Value of ioctlID | InputPtr represents | OutputPtr represents | Purpose |
|-------------------|-------------------------|----------------------|--|
| GET_DEVICE_CONFIG | Pointer to SCONFIG_LIST | NULL pointer | To get the configuration or status of the pass-thru device |
| SET_DEVICE_CONFIG | Pointer to SCONFIG_LIST | NULL pointer | To set the configuration or status of the pass-thru device |

FIGURE 77 - IOCTL DETAILS

18.4.1 GET_DEVICE_CONFIG

The ioctlID value of GET_DEVICE_CONFIG is used to obtain the configuration or status of the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 78. When the function is successfully completed, the corresponding parameter value(s) indicated in Figure 80 will be placed in each Value.

| Parameter | Description |
|-----------|---|
| DeviceID | Device ID assigned by DLL during PassThruOpen |
| IoctlID | Is set to the define GET_DEVICE_CONFIG. |
| InputPtr | <p>Points to the structure SCONFIG_LIST, which is defined as follows:</p> <pre>typedef struct { unsigned long NumOfParams; /* number of SCONFIG elements */ SCONFIG *ConfigPtr; /* array of SCONFIG */ } SCONFIG_LIST</pre> <p>where: NumOfParams is an INPUT, which contains the number of SCONFIG elements in the array pointed to by ConfigPtr. ConfigPtr is a pointer to an array of SCONFIG structures.</p> <p>The structure SCONFIG is defined as follows:</p> <pre>typedef struct { unsigned long Parameter; /* name of parameter */ unsigned long Value; /* value of the parameter */ } SCONFIG</pre> <p>where: Parameter is an INPUT that represents the parameter to be obtained (See Figure 86 for a list of valid parameters). Value is an OUTPUT that represents the value of that parameter (See Figure 86 for a list of valid values).</p> |
| OutputPtr | Is a NULL pointer, as this parameter is not used. |

FIGURE 78 - GET_DEVICE_CONFIG DETAILS

Requesting other parameters shall result in the Return Value ERR_INVALID_IOCTL_PARAM_ID.

18.4.2 SET_DEVICE_CONFIG

The IoctlID value of SET_DEVICE_CONFIG is used to set the configuration or status of the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 79. When the function is successfully completed the corresponding parameter(s) and value(s) indicated in Figure 80 will be in effect.

| Parameter | Description |
|-----------|---|
| DeviceID | Device ID assigned by DLL during PassThruOpen |
| IoctlID | Is set to the define SET_DEVICE_CONFIG. |
| InputPtr | <p>Points to the structure SCONFIG_LIST, which is defined as follows:</p> <pre>typedef struct { unsigned long NumOfParams; /* number of SCONFIG elements */ SCONFIG *ConfigPtr; /* array of SCONFIG */ } SCONFIG_LIST</pre> <p>where: NumOfParams is an INPUT, which contains the number of SCONFIG elements in the array pointed to by ConfigPtr. ConfigPtr is a pointer to an array of SCONFIG structures.</p> <p>The structure SCONFIG is defined as follows:</p> <pre>typedef struct { unsigned long Parameter; /* name of parameter */ unsigned long Value; /* value of the parameter */ } SCONFIG</pre> <p>where: Parameter is an INPUT that represents the parameter to be set (See Figure 86 for a list of valid parameters). Value is an INPUT that represents the value of that parameter (See Figure 86 for a list of valid values).</p> |
| OutputPtr | Is a NULL pointer, as this parameter is not used. |

FIGURE 79 - SET_DEVICE_CONFIG DETAILS

Setting other parameters shall result in the Return Value ERR_INVALID_IOCTL_PARAM_ID.

| Parameter | VALID VALUES FOR PARAMETER | DEFAULT VALUE (DECIMAL) | Description |
|-----------------------|----------------------------|-------------------------|--|
| NON_VOLATILE_STORE_1 | 0X00000000 – 0XFFFFFFF | 0 | First 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_2 | 0X00000000 – 0XFFFFFFF | 0 | Second 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_3 | 0X00000000 – 0XFFFFFFF | 0 | Third 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_4 | 0X00000000 – 0XFFFFFFF | 0 | Fourth 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_5 | 0X00000000 – 0XFFFFFFF | 0 | Fifth 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_6 | 0X00000000 – 0XFFFFFFF | 0 | Sixth 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_7 | 0X00000000 – 0XFFFFFFF | 0 | Seventh 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_8 | 0X00000000 – 0XFFFFFFF | 0 | Eighth 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_9 | 0X00000000 – 0XFFFFFFF | 0 | Ninth 4-byte non-volatile data storage location. |
| NON_VOLATILE_STORE_10 | 0x00000000 – 0xFFFFFFFF | 0 | Tenth 4-byte non-volatile data storage location. |

FIGURE 80 - IOCTL GET_DEVICE_CONFIG / SET_DEVICE_CONFIG PARAMETER DETAILS

Notes for NON_VOLATILE_STORE parameters:

1. The SET_DEVICE_CONFIG operation shall complete and the DLL shall return from the PassThruIoctl call within ten seconds.
2. Parameter values shall survive disconnection from all power sources.
3. Parameter values shall not be affected by an interface device firmware programming operation.
4. Parameter values shall persist for a minimum of six months after each write operation.

Usage Guideline

A typical J2534 interface may have a limit on the number of writes to non-volatile memory. These parameters are intended to be SET infrequently. The number of write cycles can be minimized by writing as many parameters as possible in one SET_DEVICE_CONFIG call.

18.5 Discovery Support

The device shall report support for Extended PassThruIoctl for Device Configuration through the discovery mechanism defined in "Discovery Mechanism" Section.

19. TP2.0 PROTOCOL

19.1 Scope of the TP2.0 Protocol Optional Feature

This section details the extensions to a SAE J2534-1 interface that will allow support of the TP2.0 protocol. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed from SAE J2534-1

19.2 Pass-Thru System Requirements

19.2.1 Simultaneous Communication on Multiple Protocols

Support for simultaneous communication of TP2.0 with other protocols shall be equivalent to the definition for CAN in J2534-1.

19.2.2 Pin Usage

TP2.0 uses SAE J1962 Pins 6 and Pin 14. As with all SAE J2534-2 optional protocols, no default pin is identified. Therefore, the application developer will be required to set the Pin to be used.

See the J1962 Pin Selection section of J2534-2 for details of the method used to switch J1962 pins.

19.3 Win32 Application Programming Interface

19.3.1 API Functions – Overview

Feature Implementation Summary:

Some ECUs support the TP2.0 protocol that uses the CAN physical layer interface ISO11898.

Reference SAE J2819 for the detailed protocol information.

Key features of the TP2.0 protocol are the establishment and maintenance of a channel between 2 nodes. The maintenance of a connection on an established channel and the ability to send broadcast messages that repeat 5 times at a configurable time spacing.

The guidelines to implement this communication capability to SAE J2534-2 API are shown below:

- The interface shall be able to connect a 500K baud CAN physical layer interface on pins 6 and 14 of the J1962 connector.
- Send a broadcast message repeated 5 times with the last 2 bytes alternating between 0x55 and 0xAA with a configurable time spacing between the messages.
- Send a re-trigger broadcast message which starts out as stated above with additional messages sent at a slower configured rate.
- The ability to stop the re-triggered broadcast message
- The ability to establish and maintain 4 channels with connections simultaneously.
- Send indication messages to the application when a connection is established or disconnected.
- The interface must fragment and reassemble messages.
- The interface must be able to delay the transmission of the Transport packets as specified by T1 and T3 in the TP2.0 protocol. The interface must support a minimum time resolution of 500 μ s. Note: even though the TP2.0 specification can specify a time delay of 100 μ s, the PassThru device need only support a minimum delay of 500 μ s. The PassThru device should always extend a requested delay time to the nearest 500 μ s time interval.
- Buffer one outgoing and one incoming messages as they are being fragmented or reassembled on each channel.
- Handle the Acknowledge with retry and break events.
- The interface shall automatically transmit the Connection Test telegram to keep a connection active.
- The device shall maintain the sequence number as part of the Data telegram and Data Acknowledge telegram for each connection.
- The interface shall pass up to the application any messages received unexpectedly that passes any filter criteria, i.e. outside the bounds of an established connection.
- The interface shall support one connection as a passive device (a module on the network requests a connection with the interface). This passive connection is configurable meaning it can be disabled. A passive connection counts as one of the 4 connections supported by the interface.

Messages passed to PassThruWriteMsgs by the application will include all message address and data bytes, including the proper CAN address for the device or connection. All received messages shall be queued by the interface for retrieval by calls to PassThruReadMsgs. Data telegrams received on an established connection will be reassembled into a PASSTHRU_MSG that is stored in the receive message queue. The application will be able to retrieve this message with a call to PassThruReadMsgs.

Support for the TP2.0 protocol is achieved by the addition of a new ProtocolID, modified behavior of PassThruStartPeriodicMsg, new PassThruIoctl IoctlIDs and a new low level implementation for transmission and reception of sequenced messages. This protocol requires additional API and device software support, but no additional hardware compared with a standard J2534-1 device.

If the TP2.0 protocol is not supported, ERR_NOT_SUPPORTED will be returned by the call PassThruConnect. The calling application will be required to notify the user that this optional feature may not be supported by the interface.

API Change List:

The following figure summarizes the changes to the SAE J2534-1 API Functions.

| Function | Description of Change |
|--------------------------|---|
| PassThruConnect | Added new ProtocolID values. |
| PassThruWriteMsgs | Added new functionality to this function. |
| PassThruStartPeriodicMsg | Added new functionality to this function |
| PassThruIoctl | New configuration parameters are added. |

FIGURE 81 - SAE J2534 API FUNCTIONS

19.3.2 API Functions - Detailed Information

19.3.2.1 PassThruConnect

When PassThruConnect is called with the TP2_0_PS Protocol ID, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

19.3.2.1.1 C / C++ Prototype

There are no changes defined for the function prototype.

19.3.2.1.2 Parameters

There are no changes to the Parameters.

19.3.2.1.3 Connect Flag Values

There are no changes to the Connect Flags.

19.3.2.1.4 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|------------|---------------------------------------|
| TP2_0_PS | TP2.0 Protocol with pin selection |
| TP2_0_CHx | Additional channels of TP2.0 Protocol |

FIGURE 82 - PROTOCOL ID VALUES

19.3.2.2 PassThruWriteMsgs

When used with the TP2.0 protocol, the PassThruWriteMsgs function will perform the following additional functions.

- If the write message's address matches an established connection address, the message will be sent using the TP2.0 protocol.
- If the write message's address does not match an established connection address and the message size fits in a single CAN message, the message will be sent on the bus.

- If the write message's address does not match an established connection address and the message size is larger than a single CAN message, the error ERR_NO_CONNECTION_ESTABLISHED will be returned.
- If the TxFlag "TP2_0_BROADCAST_MSG" is set in the message and the message contains a valid broadcast address in Data[0] (Data[0] value between 0xF0 and 0xFF), the message will be sent 5 times using TP2_0_T_BR_INT as the rate with the last 2 bytes of the message alternating between 0xAA and 0x55. If Data[0] does not contain a valid broadcast address, the function will return ERR_INVALID_MSG.

19.3.2.3 PassThruStartPeriodicMsg

This function is used to send the re-triggered Broadcast messages as well as normal periodic messages. To send the TP2.0 re-triggered Broadcast message, Populate the message data with the Broadcast message, set the TxFlag "TP2_0_BROADCAST_MSG" bit and set the periodic rate to the re-trigger rate desired. This function will immediately send out the message 5 times using TP2_0_T_BR_INT as the rate and then send the message at the periodic rate (alternating the last two bytes of the message between 0xAA and 0x55).

19.3.3 IOCTL Section

Figure 83 provides the details on the new IOCTLIDs available through the PassThruIoctl function:

| Value of ioctlID | InputPtr Represents | OutputPtr | Purpose |
|---------------------|-----------------------|--------------|--|
| REQUEST_CONNECTION | Points to SBYTE_ARRAY | NULL pointer | To establish a Channel and connection. |
| TEARDOWN_CONNECTION | Points to SBYTE_ARRAY | NULL pointer | To tear down an established connection |

FIGURE 83 - IOCTL DETAILS

19.3.3.1 SET_CONFIG and GET_CONFIG Additional Parameters

The DATA_RATE, LOOPBACK, BIT_SAMPLE_POINT and SYNC_JUMP_WIDTH configuration parameters will be supported for the TP2.0 protocol. The default values for TP2.0 are LOOPBACK = 0, BIT_SAMPLE_POINT = 80% and SYNC_JUMP_WIDTH = 15%.

Ten new configuration parameters will be added to support TP2.0 protocol.

| Parameter | VALID VALUES | DEFAULT VALUES | Description |
|-------------------|-----------------------------------|----------------|---|
| TP2_0_T_BR_INT | 0X0 – 0XFFFF (1 ms per bit) | 20 | Broadcast Interval: Time interval between the 5 messages of a single broadcast message. |
| TP2_0_T_E | 0X0 – 0XFFFF (1 ms per bit) | 100 | Maximum time out waiting for a Channel Acknowledge or Connection Acknowledge telegram |
| TP2_0_MNTC | 0X0 – 0XFFFF (1 count per bit) | 10 | Retry count for Connection management messages. |
| TP2_0_T_CTA | 0X0 – 0XFFFF (1 ms per bit) | 1000 | Connection Test time out for the Active device |
| TP2_0_MNCT | 0X0 – 0XFFFF (1 count per bit) | 5 | Retry count for sending Connection Test messages |
| TP2_0_MNTB | 0X0 – 0XFFFF (1 count per bit) | 5 | Maximum number of Not Ready Acknowledge received in on block |
| TP2_0_MNT | 0X0 – 0XFFFF (1 count per bit) | 2 | Maximum repeats of acknowledge requests |
| TP2_0_T_WAIT | 0X0 – 0XFFFF (1 ms per bit) | 100 | Time to wait before the next transmission when a Not Ready acknowledge is received |
| TP2_0_T1 | 0X0 – 0XFFFF (1 MS PER BIT) | 100 | Time out used when waiting for a response. |
| TP2_0_T3 | 0X0 – 0XFFFF (1 MS PER BIT) | 0 | Minimum time the J2534 device needs between telegrams sent to it. |
| TP2_0_IDENTIFER | 0X0 OR 0X200 – 0X2EF | 0 | The passive identifier CAN ID is used by the interface to receive requests for a connection. The value of zero disables this feature. |
| TP2_0_RXIDPASSIVE | 0X0 OR 0X300 – 0X7FF | 0 | The passive RX CAN ID used in a Passive connection. The value of zero disables this feature. |

FIGURE 90 - TP2.0 CONFIGURATION PARAMETERS

No other J2534-1 defined configuration parameters are supported with the exception of J1962_PINS.

The TP2_0_IDENTIFIER is used by application to give the interface a CAN ID which will be used to compare with the Destination field in the TP2.0 messages. An implicit PASS filter will be started when this value is set between the range of 0x200 and 0x2EF. The implicit PASS filter will be cleared when the address is changed to zero. If a value that is not zero or in the range of 0x200 through 0x2EF is requested, the PassThruIoctl will return ERR_INVALID_IOCTL_VALUE.

If the TP2_0_IDENTIFIER has been set to a valid value and the request for a connection is received from the network, it will be rejected with a 0xD8 “Temporarily no resources are free” response if all 4 connections are in use or the TP2_0_RXIDPASSIVE is zero. If the request for connection is received and a connection is available and TP2_0_RXIDPASSIVE is set to a valid value, the connection will be established and the CONNECTION_ESTABLISHED Indication message will be sent to the application. The TP2_0_RXIDPASSIVE will be the first 4 data bytes Data[0] through Data[3] of the indication message and the TX-ID-P address from the network device will be the next 4 data bytes Data[4] through Data[7], Data [4] is MSB.

19.3.3.2 REQUEST_CONNECTION

The `loctlID` parameter of `REQUEST_CONNECTION` is used to request the establishment of a channel and connection between the J2534 device and an ECU.

| Parameter | Description | | | | | | | | | | | | | | |
|-------------------------------|--|------------|---|-------------------------------|--|------------|--------------------|------------|----------------------|-------------------------------|----------------------------|-------------------------------|----------------------------|-------------|-------------------------|
| ChannelID | Channel ID assigned by DLL during <code>PASSTHRUCONNECT</code> . | | | | | | | | | | | | | | |
| loctlID | Is set to <code>REQUEST_CONNECTION</code> | | | | | | | | | | | | | | |
| InputPtr | Points to the structure <code>SBYTE_ARRAY</code> , which is defined as follows: <pre> Typedef struct { unsigned long NumOfBytes; /* number of bytes in array */ unsigned char *BytePtr; /* array of bytes */ } </pre> <p>where:</p> <table> <tr> <td>NumOfBytes</td><td>is an input that indicates the number of bytes in the array <code>BytePtr</code>. It should always be 11 for <code>REQUEST_CONNECTION</code>.</td></tr> <tr> <td>BytePtr[0] through BytePtr[3]</td><td>is the CAN address (Identifier) BytePtr[0] is MSB.</td></tr> <tr> <td>BytePtr[4]</td><td>is the Destination</td></tr> <tr> <td>BytePtr[5]</td><td>is the Opcode (0xC0)</td></tr> <tr> <td>BytePtr[6] through BytePtr[7]</td><td>is the TX-ID-A information</td></tr> <tr> <td>BytePtr[8] through BytePtr[9]</td><td>is the RX-ID-A information</td></tr> <tr> <td>BytePtr[10]</td><td>is the Application Type</td></tr> </table> | NumOfBytes | is an input that indicates the number of bytes in the array <code>BytePtr</code> . It should always be 11 for <code>REQUEST_CONNECTION</code> . | BytePtr[0] through BytePtr[3] | is the CAN address (Identifier) BytePtr[0] is MSB. | BytePtr[4] | is the Destination | BytePtr[5] | is the Opcode (0xC0) | BytePtr[6] through BytePtr[7] | is the TX-ID-A information | BytePtr[8] through BytePtr[9] | is the RX-ID-A information | BytePtr[10] | is the Application Type |
| NumOfBytes | is an input that indicates the number of bytes in the array <code>BytePtr</code> . It should always be 11 for <code>REQUEST_CONNECTION</code> . | | | | | | | | | | | | | | |
| BytePtr[0] through BytePtr[3] | is the CAN address (Identifier) BytePtr[0] is MSB. | | | | | | | | | | | | | | |
| BytePtr[4] | is the Destination | | | | | | | | | | | | | | |
| BytePtr[5] | is the Opcode (0xC0) | | | | | | | | | | | | | | |
| BytePtr[6] through BytePtr[7] | is the TX-ID-A information | | | | | | | | | | | | | | |
| BytePtr[8] through BytePtr[9] | is the RX-ID-A information | | | | | | | | | | | | | | |
| BytePtr[10] | is the Application Type | | | | | | | | | | | | | | |
| OutputPtr | NULL Pointer | | | | | | | | | | | | | | |

FIGURE 84 - REQUEST_CONNECTION DETAILS

The data contained in the `BytePtr` array is the equivalent of the CAN message sent to Request the Channel. The `BytePtr` data will be sent on the network to try and start a connection. If the RX-ID-A is already in use by another established channel, the error `ERR_NOT_UNIQUE` will be returned. When a connection is successfully established, an implicit PASS filter will be created such that all messages received on the channel from CAN ID RX-ID-A will be available to the application. The implicit PASS filter will consume one of the minimum of 10 filters required by SAE J2534-1. NOTE: with 4 open channels a total of 4 filters will be consumed as implicit PASS filters.

The `REQUEST_CONNECTION` `loctl` call is non-blocking which means the API will not wait for the connection to be made. A `CONNECTION_ESTABLISHED` indication will be placed in the RX queue indicating a successful Connection Request. A `CONNECTION_LOST` indication will be placed in the RX queue to indicate a failed Connection Request. The RX-ID-A specified in the `REQUEST_CONNECTION` will be the CAN address of the indication message.

If the Data in the `BytePtr` array is ill formatted or the `NumOfBytes` is not 11 the function will return with the error `ERR_INVALID_IOCTL_VALUE`.

19.3.3.3 TEARDOWN_CONNECTION

The `loctlID` parameter of `TEARDOWN_CONNECTION` is used to tear down an established channel and connection between the J2534 device and an ECU.

| Parameter | Description |
|-----------|--|
| ChannelID | Channel ID assigned by DLL during <code>PassThruConnect</code> . |
| loctlID | Is set to <code>TEARDOWN_CONNECTION</code> |
| InputPtr | Points to the structure <code>SBYTE_ARRAY</code> , which is defined as follows: Typedef struct { unsigned long NumOfBytes; /* number of bytes in array */ unsigned char *BytePtr; /* array of bytes */ } where: NumOfBytes is an input that indicates the number of bytes in the array <code>BytePtr</code> . It must always be 4 for <code>TEARDOWN_CONNECTION</code> . <code>BytePtr[0]</code> through <code>BytePtr[3]</code> is the receiving CAN address, <code>BytePtr[0]</code> is MSB. |
| OutputPtr | NULL Pointer |

FIGURE 85 - TEARDOWN_CONNECTION DETAILS

The data contained in the `BytePtr` array is the CAN address that the J2534-2 device is receiving CAN messages for this connection. This is the same address as the `RX-ID-A` in the `REQUEST_CONNECTION` call. The implicit pass filter for this connection will be removed once the connection has been torn down. NOTE: all filters set up with `PassThruStartMsgFilter` will remain even if they match the ID used in the connection.

The `TEARDOWN_CONNECTION` `loctl` call is non-blocking which means the API will not wait for the connection to be torn down. A `CONNECTION_LOST` indication will be placed in the RX queue indicating the success of the Connection tear down and the implicit PASS filter will be removed.

If the data in the `BytePtr` array does not match an established connection or the `NumOfBytes` is not 4, then the function will return with the error `ERR_INVALID_IOCTL_VALUE`.

19.4 Message Structure

19.4.1 C / C++ Definition

There is no change to the C/C++ Definition.

19.4.2 Message Data Formats

All received messages that comply with the timing and message structure requirements shall be queued to the application.

Additional Protocol IDs and message limitations have been defined below:

| Protocol ID | Min Tx | Max Tx | Min Rx | Max Rx | Notes |
|--------------------------|--------|--------|--------|--------|--|
| TP2_0_PS or TP2_0_CHx | 4 | 4096 | 4 | 4096 | 4 bytes of CAN ID followed by 4092 bytes of data |

FIGURE 86 - ALLOWED MESSAGE SIZES PER PROTOCOL

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID.

19.4.3 Format Checks for Messages Passed to the API

If DataSize violates Min Tx or Max Tx columns the vendor DLL shall return, ERR_INVALID_MSG.

19.4.4 Message Flag and Status Definitions

Supported Message Status (RxStatus) Definitions.

The TX_MSG_TYPE and CAN_29BIT_ID RxStatus bits will be supported. For the TP2.0 protocol, loopback messages returned by the API will have TX_MSG_TYPE set in RxStatus. Messages with a 29 bit ID field will have CAN_29BIT_ID set.

There are two additions to the Message Status (RxStatus) definitions to be used as indications.

| Definition | RxStatus Bit(s) | DESCRIPTION | VALUE |
|------------------------|-----------------|---|---|
| CONNECTION_LOST | 17 | Connection Lost status indication will be sent when a established connection has been lost or when an attempt to establish a connection fails. The receiving CAN address RX-ID-A or RX-ID-P will be in Data[0] through Data[3] of the message, Data[0] is MSB. Data[4] will indicated why the connection failed: 0 = Teardown, 1= timeout, 0xD6 = Not supported, 0xD7 = Temporarily not supported, 0xD8 No resources free. ExtraDataIndex = 0, DataSize = 5. TP2.0 only. The implicit PASS filter is removed. | 0 = no Connection lost indication 1 = Connection lost or establish attempt failed. |
| CONNECTION_ESTABLISHED | 16 | Connection established status indication will be sent when an Connection is successfully established. The receiving CAN address RX-ID-A or RX-ID-P will be in Data[0] through Data[3] of the message Data[0] is MSB. The sending CAN address TX-ID-A or TX-ID-P will be in Data[4] through Data[7], Data[4] is MSB. ExtraDataIndex = 0, DataSize = 8. TP2.0 only | 0 = no connection indication 1 = Connection was successful. |

FIGURE 87 - RXSTATUS BIT DEFINITIONS

Supported TxFlags definitions:

The TxFlag CAN_29BIT_ID will be supported if the TP2.0 connection was established with CAN_ID_BOTH and a 29 bit CAN message needs to be sent.

The one additional TxFlags Definition.

| Definition | TxFlags Bit(s) | DESCRIPTION | VALUE |
|---------------------|----------------|--|--|
| TP2_0_BROADCAST_MSG | 16 | Send this message as a broadcast which means it will be sent 5 times with an interval of TP2_0_T_BR_INT. The last 2 data bytes will alternate between 0x55 and 0xAA. | 0 = normal message 1 = TP2.0 Broadcast Message. |

FIGURE 88 - TXFLAG BIT DEFINITION

19.5 Return Values

The following new Return Value has been defined.

| Definition | DESCRIPTION |
|-------------------------------|---|
| ERR_NO_CONNECTION_ESTABLISHED | TP2.0 connection has been lost or was never established |

FIGURE 89 - ADDITIONAL RETURN VALUES FOR THE TP2.0 PROTOCOL

19.6 Discovery Support

The device shall report support for the TP2.0 protocol and associated parameters through the discovery mechanism defined in “Discovery Mechanism” Section.

20. FAULT-TOLERANT CAN

20.1 Scope of the Fault-Tolerant CAN Optional Feature

Information contained in this section will define extensions to a compliant SAE J2534-1 interface to support Fault-Tolerant CAN.

20.2 Pass-Thru System Requirements

20.2.1 Pin Usage

Fault-Tolerant CAN (FTCAN) may be connected to either of these sets of pins on the J1962 connector:

- Pin 1 – CAN-high, pin 9 – CAN-low
- Pin 3 – CAN-high, pin 11 – CAN-low

As with all SAE J2534-2 optional protocols, no default pin is identified, therefore, the application developer will be required to set the pin to be used. See the SAE J1962 pin Selection section for discussion of pin usage.

20.3 Win32 Application Programming Interface

20.3.1 API Functions – Overview

Information contained in this section is intended to define the API resources required to incorporate an optional protocol channel. This channel, identified as Fault-Tolerant CAN (FTCAN), will require hardware and software API support to fully implement this feature.

From the J2534 API perspective, FT_CAN_PS and FT_CAN_CHx are equivalent to CAN_PS and CAN_CHx except as specifically mentioned in this section. Likewise, FT_ISO15765_PS and FT_ISO15765_CHx are equivalent to ISO15765_PS and ISO15765_CHx except as specifically mentioned in this section.

The details on the physical implementation of FTCAN are defined in ISO 11898-3.

If this feature is not supported an error code, ERR_NOT_SUPPORTED, will be returned by the call PassThruConnect.

Figure 90 summarizes the changes to the SAE J2534-1 API functions.

| Function | Description of Change |
|-----------------|------------------------------|
| PassThruConnect | Added new ProtocolID values. |

FIGURE 90 - SAE J2534 API FUNCTIONS

20.3.2 API Functions - Detailed Information

20.3.2.1 PassThruConnect

When PassThruConnect is called with either the FT_CAN_PS or FT_ISO15765_PS Protocol IDs, the physical layer remains disconnected until a call to PassThruIoctl, SET_CONFIG is made to select the desired cable and pins.

20.3.2.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in Figure 91. The actual value is defined in the section titled 'SAE J2534-2 Resource'.

| Definition | Description |
|-----------------|---|
| FT_CAN_PS | Raw Fault-Tolerant CAN messages with pin selection |
| FT_ISO15765_PS | Fault-Tolerant CAN adhering to ISO15765-2 flow control with pin selection |
| FT_CAN_CHx | Additional channels of Raw Fault-Tolerant CAN messages |
| FT_ISO15765_CHx | Additional channels of Fault-Tolerant CAN adhering to ISO15765-2 flow control |

FIGURE 91 - PROTOCOLID DESCRIPTIONS

20.4 Message Structure

20.4.1 Elements

The ProtocolID field of the Pass-Thru Message Structure shall always contain the ProtocolID that was passed into PassThruConnect function. All other ProtocolIDs will result in ERR_MSG_PROTOCOL_ID. (The only exception to this is the Mixed Format Frames feature on a CAN Network.)

SAE J2534-1 definitions of requirements, restrictions, and error conditions for CAN and ISO15765 protocols will apply to Single Wire CAN channels discussed above.

20.4.2 Message Flag and Status Definitions

There is one addition to the Message Status (RxStatus) definitions.

| Definition | RxStatus Bit(s) | DESCRIPTION | VALUE |
|------------|-----------------|--|------------------------------------|
| LINK_FAULT | 17 | Status bit in a received message that is set when the transceiver has detected a network fault but the device was still able to correctly receive the message. | 0 = No fault 1 = Fault detected |

FIGURE 92 - NEW RXSTATUS DEFINITIONS

20.5 Discovery Support

The device shall report support for Fault Tolerant CAN and associated parameters through the discovery mechanism defined in "Discovery Mechanism" Section.

21. DISCOVERY MECHANISM

21.1 Scope of the Discovery Mechanism Feature

This section details the extensions to SAE J2534-1 to support a mechanism to programmatically determine/confirm the capabilities of a specific J2534 device. This section details only the changes from SAE J2534-1. Items not specifically detailed in this section are assumed not to have changed.

21.2 Pass-Thru System Requirements

The Discovery Mechanism imposes no additional hardware requirements.

21.3 Win32 Application Programming Interface

21.3.1 API Functions – Overview

The purpose of this feature is to provide a mechanism to programmatically determine/confirm the capabilities of a specific J2534 device. These capabilities are static and shall not change based on the current state of the device. (For example, the identification of which pins ISO15765 can be switched to shall not be altered if some of those pins are currently in use by another protocol.)

To access the capabilities, an application must successfully call **PassThruOpen**. Then, the application must call **PassThruIoctl** with a list of parameters it is interested in. Upon return, the device will indicate if the parameter is supported and its associate value. Two new ioctlIDs have been defined to support this feature, they are:

GET_DEVICE_INFO – Used to acquire the general capabilities of the device.

GET_PROTOCOL_INFO – Used to acquire the protocol specific capabilities of the device.

To obtain the general capabilities of the device, the application shall call **PassThruIoctl** with the *ChannelID* set to the DeviceID returned from **PassThruOpen**, *ioctlID* set to GET_DEVICE_INFO, and *OutputPtr* pointing to a list of parameter/value pairs of interest to the user on the specified device.

To obtain the protocol specific capabilities of the device, the application shall call **PassThruIoctl** with the *ChannelID* set to the DeviceID returned from **PassThruOpen**, *ioctlID* set to GET_PROTOCOL_INFO, the *InputPtr* pointing to a Protocol ID, and *OutputPtr* pointing to a list of parameter/value pairs of interest to the user for the associated protocol on the specified device. It is expected that the application would repeat this step for each protocol it is interested in.

Figure summarizes the changes to the SAE J2534-1 API functions.

| Function | Description of Change |
|---------------|-------------------------|
| PassThruIoctl | Add new ioctlID values. |

FIGURE 100 - SAE J2534 API FUNCTIONS

21.3.2 API Functions - Detailed Information

21.3.2.1 IOCTL Section

There are two additional ioctlIDs, as follows:

| Value of ioctlID | InputPtr represents | OutputPtr represents | Purpose |
|-------------------|------------------------|------------------------|--|
| GET_DEVICE_INFO | NULL pointer | Pointer to SPARAM_LIST | To acquire the general capabilities of the device. |
| GET_PROTOCOL_INFO | Pointer to Protocol ID | Pointer to SPARAM_LIST | To acquire the protocol specific capabilities of the device. |

FIGURE 93 - IOCTL DETAILS

21.3.2.2 GET_DEVICE_INFO

The ioctlID value of GET_DEVICE_INFO shall be used to request the general capabilities of a J2534 device. The calling application is responsible for allocating and initializing the inputs described in Figure 94. Upon successful completion (a return code of STATUS_NOERROR), the corresponding *Supported* and/or *Value* elements shall have been updated. Requests for parameters that are not supported by the device shall cause the corresponding *Supported* element to be set to NOT_SUPPORTED and the corresponding *Value* to remain unaltered. In this case, the interface shall continue processing the rest of the parameters in the list and the return code shall be STATUS_NOERROR (as long as no other errors conditions exist). Valid device parameters and their associated descriptions are detailed in Figure 95.

| Parameter | Description |
|-----------|--|
| ChannelID | Device ID assigned by DLL during PassThruOpen |
| ioctlID | Is set to the define GET_DEVICE_INFO. |
| InputPtr | Is set to NULL, as this parameter is not used. |
| OutputPtr | <p>Points to the structure SPARAM_LIST, which is defined as follows:</p> <pre>typedef struct { unsigned long NumOfParams; /* number of SPARAM elements */ SPARAM *ParamPtr; /* array of SPARAM */ } SPARAM_LIST</pre> <p>where:</p> <p>NumOfParams is an INPUT, set by the application, which contains the number of SPARAM elements in the array pointed to by ParamPtr.</p> <p>ParamPtr is an INPUT, set by the application, which points to an array of SPARAM structures allocated by the application.</p> <p>The structure SPARAM is defined as follows:</p> <pre>typedef struct { unsigned long Parameter; /* name of parameter */ unsigned long Value; /* value of the parameter */ unsigned long Supported; /* support for parameter */ } SPARAM</pre> <p>where:</p> <p>Parameter is an INPUT, set by the application, which represents the ID of the parameter requested (see Figure 95 for a list of valid parameters).</p> <p>Value is typically an OUTPUT, set by the interface, which represents the parameter's value. However, for certain parameters Value may also be an INPUT set by the application (see Figure 95 for more details).</p> <p>Supported is an OUTPUT, set by the interface to 0 (indicating the associated parameter is NOT_SUPPORTED) or 1 (indicating the associated parameter is SUPPORTED). The contents of Value shall remain unchanged if the parameter is NOT_SUPPORTED.</p> |

FIGURE 94 - GET_DEVICE_INFO DETAILS

| Parameter | Description | Associated Value |
|--------------------|---|--|
| SERIAL_NUMBER | Represents the device serial number. | Value is an OUTPUT that, upon return, contains an unsigned long that is the device serial number. |
| PART_NUMBER | Represents the part number of the device. | Value is an OUTPUT that, upon return, contains an unsigned long that is the device part number. |
| J1850PWM_SUPPORTED | Represents support for J1850PWM related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1850PWM_CHx Protocol IDs that are available. RR is the number of channels for the J1850PWM_PS Protocol ID that are available. SS is 1 if the J1850PWM Protocol ID is supported; otherwise it must be 0. |
| J1850VPW_SUPPORTED | Represents support for J1850VPW related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1850VPW_CHx Protocol IDs that are available. RR is the number of channels for the J1850VPW_PS Protocol ID that are available. SS is 1 if the J1850VPW Protocol ID is supported; otherwise it must be 0. |
| ISO9141_SUPPORTED | Represents support for ISO9141 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO9141_CHx Protocol IDs that are available. RR is the number of channels for the ISO9141_PS Protocol ID that are available. SS is 1 if the ISO9141 Protocol ID is supported; otherwise it must be 0. |
| ISO14230_SUPPORTED | Represents support for ISO14230 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO14230_CHx Protocol IDs that are available. RR is the number of channels for the ISO14230_PS Protocol ID that are available. SS is 1 if the ISO14230 Protocol ID is supported; otherwise it must be 0. |
| CAN_SUPPORTED | Represents support for CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of CAN_CHx Protocol IDs that are available. RR is the number of channels for the CAN_PS Protocol ID that are available. SS is 1 if the CAN Protocol ID is supported; otherwise it must be 0. |
| ISO15765_SUPPORTED | Represents support for ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO15765_CHx Protocol IDs that are available. RR is the number of channels for the ISO15765_PS Protocol ID that are available. SS is 1 if ISO15765 is supported; otherwise it must be 0. |

| Parameter | Description | Associated Value |
|------------------------|--|--|
| FT_CAN_SUPPORTED | Represents support for FT_CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of FT_CAN_CHx Protocol IDs that are available. RR is the number of channels for the FT_CAN_PS Protocol ID that are available. SS must be 0 |
| FT_ISO15765_SUPPORTED | Represents support for FT_ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of FT_ISO15765_CHx Protocol IDs that are available. RR is the number of channels for the FT_ISO15765_PS Protocol ID that are available. SS must be 0 |
| SCI_A_ENGINE_SUPPORTED | Represents support for the SCI_A_ENGINE Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_A_ENGINE Protocol ID is supported; otherwise it must be 0. |
| SCI_A_TRANS_SUPPORTED | Represents support for the SCI_A_TRANS Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_A_TRANS Protocol ID is supported; otherwise it must be 0. |
| SCI_B_ENGINE_SUPPORTED | Represents support for the SCI_B_ENGINE Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_B_ENGINE Protocol ID is supported; otherwise it must be 0. |
| SCI_B_TRANS_SUPPORTED | Represents support for the SCI_B_TRANS Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_B_TRANS Protocol ID is supported; otherwise it must be 0. |
| SW_ISO15765_SUPPORTED | Represents support for SW_ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of SW_ISO15765_CHx Protocol IDs that are available. RR is the number of channels for the SW_ISO15765_PS Protocol ID that are available. SS must be 0. |
| SW_CAN_SUPPORTED | Represents support for SW_CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of SW_CAN_CHx Protocol IDs that are available. RR is the number of channels for the SW_CAN_PS Protocol ID that are available. SS must be 0. |

| Parameter | Description | Associated Value |
|--------------------------|---|---|
| GM_UART_SUPPORTED | Represents support for GM_UART related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of GM_UART_CHx Protocol IDs that are available. RR is the number of channels for the GM_UART_PS Protocol ID that are available. SS must be 0. |
| UART_ECHO_BYTE_SUPPORTED | Represents support for UART_ECHO_BYTE related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of UART_ECHO_BYTE_CHx Protocol IDs that are available. RR is the number of channels for the UART_ECHO_BYTE_PS Protocol ID that are available. SS must be 0. |
| HONDA_DIAGH_SUPPORTED | Represents support for HONDA_DIAGH related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of HONDA_DIAGH_CHx Protocol IDs that are available. RR is the number of channels for the HONDA_DIAGH_PS Protocol ID that are available. SS is 1 if HONDA_DIAGH is supported; otherwise it must be 0. |
| J1939_SUPPORTED | Represents support for J1939 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1939_CHx Protocol IDs that are available. RR is the number of channels for the J1939_PS Protocol ID that are available. SS must be 0. |
| J1708_SUPPORTED | Represents support for J1708 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1708_CHx Protocol IDs that are available. RR is the number of channels for the J1708M_PS Protocol ID that are available. SS must be 0. |
| TP2_0_SUPPORTED | Represents support for TP2_0 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of TP2_0_CHx Protocol IDs that are available. RR is the number of channels for the TP2_0_PS Protocol ID that are available. SS must be 0. |
| J2610_SUPPORTED | Represents support for J2610 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J2610_CHx Protocol IDs that are available. RR is the number of channels for the J2610_PS Protocol ID that are available. SS must be 0. |

| Parameter | Description | Associated Value |
|-----------------------|--|--|
| ANALOG_IN_SUPPORTED | Represents support for ANALOG_IN_XXX related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ANALOG_IN_x Protocol IDs that are available. RR must be 0. SS must be 0. |
| J1850PWM_SIMULTANEOUS | Represents support for J1850PWM related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1850PWM_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the J1850PWM_PS Protocol ID that can operate simultaneously. SS is 1 if the J1850PWM Protocol ID is supported; otherwise it must be 0. |
| J1850VPW_SIMULTANEOUS | Represents support for J1850VPW related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1850VPW_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the J1850VPW_PS Protocol ID that can operate simultaneously. SS is 1 if the J1850VPW Protocol ID is supported; otherwise it must be 0. |
| ISO9141_SIMULTANEOUS | Represents support for ISO9141 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO9141_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the ISO9141_PS Protocol ID that can operate simultaneously. SS is 1 if the ISO9141 Protocol ID is supported; otherwise it must be 0. |
| ISO14230_SIMULTANEOUS | Represents support for ISO14230 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO14230_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the ISO14230_PS Protocol ID that can operate simultaneously. SS is 1 if the ISO14230 Protocol ID is supported; otherwise it must be 0. |
| CAN_SIMULTANEOUS | Represents support for CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of CAN_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the CAN_PS Protocol ID that can operate simultaneously. SS is 1 if the CAN Protocol ID is supported; otherwise it must be 0. |
| ISO15765_SIMULTANEOUS | Represents support for ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ISO15765_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the ISO15765_PS Protocol ID that can operate simultaneously. SS is 1 if ISO15765 is supported; otherwise it must be 0. |

| Parameter | Description | Associated Value |
|---------------------------|--|--|
| FT_CAN_SIMULTANEOUS | Represents support for FT_CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of FT_CAN_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the FT_CAN_PS Protocol ID that can operate simultaneously. SS must be 0 |
| FT_ISO15765_SIMULTANEOUS | Represents support for FT_ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of FT_ISO15765_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the FT_ISO15765_PS Protocol ID that can operate simultaneously. SS must be 0 |
| SCI_A_ENGINE_SIMULTANEOUS | Represents support for the SCI_A_ENGINE Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_A_ENGINE Protocol ID is supported; otherwise it must be 0. |
| SCI_A_TRANS_SIMULTANEOUS | Represents support for the SCI_A_TRANS Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_A_TRANS Protocol ID is supported; otherwise it must be 0. |
| SCI_B_ENGINE_SIMULTANEOUS | Represents support for the SCI_B_ENGINE Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_B_ENGINE Protocol ID is supported; otherwise it must be 0. |
| SCI_B_TRANS_SIMULTANEOUS | Represents support for the SCI_B_TRANS Protocol ID. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ must be 0. RR must be 0. SS is 1 if the SCI_B_TRANS Protocol ID is supported; otherwise it must be 0. |
| SW_ISO15765_SIMULTANEOUS | Represents support for SW_ISO15765 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of SW_ISO15765_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the SW_ISO15765_PS Protocol ID that can operate simultaneously. SS must be 0. |
| SW_CAN_SIMULTANEOUS | Represents support for SW_CAN related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of SW_CAN_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the SW_CAN_PS Protocol ID that can operate simultaneously. SS must be 0. |

| Parameter | Description | Associated Value |
|-----------------------------|---|---|
| GM_UART_SIMULTANEOUS | Represents support for GM_UART related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of GM_UART_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the GM_UART_PS Protocol ID that can operate simultaneously. SS must be 0. |
| UART_ECHO_BYTE_SIMULTANEOUS | Represents support for UART_ECHO_BYTE related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of UART_ECHO_BYTE_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the UART_ECHO_BYTE_PS Protocol ID that can operate simultaneously. SS must be 0. |
| HONDA_DIAGH_SIMULTANEOUS | Represents support for HONDA_DIAGH related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of HONDA_DIAGH_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the HONDA_DIAGH_PS Protocol ID that can operate simultaneously. SS is 1 if HONDA_DIAGH is supported; otherwise it must be 0. |
| J1939_SIMULTANEOUS | Represents support for J1939 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1939_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the J1939_PS Protocol ID that can operate simultaneously. SS must be 0. |
| J1708_SIMULTANEOUS | Represents support for J1708 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J1708_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the J1708M_PS Protocol ID that can operate simultaneously. SS must be 0. |
| TP2_0_SIMULTANEOUS | Represents support for TP2_0 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of TP2_0_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the TP2_0_PS Protocol ID that can operate simultaneously. SS must be 0. |
| J2610_SIMULTANEOUS | Represents support for J2610 related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of J2610_CHx Protocol IDs that can operate simultaneously. RR is the number of channels for the J2610_PS Protocol ID that can operate simultaneously. SS must be 0. |

| Parameter | Description | Associated Value |
|--------------------------|--|---|
| ANALOG_IN_SIMULTANEOUS | Represents support for ANALOG_IN_XXX related Protocol IDs. | Value is an OUTPUT that, upon return, contains an unsigned long (with the format 0xPPQRRSS) where each byte represents the following: PP must be 0. QQ is the number of ANALOG_IN_x Protocol IDs that can operate simultaneously. RR must be 0. SS must be 0. |
| MAX_NON_VOLATILE_STORAGE | Represents the total number of Non-Volatile Memory Storage Locations supported. | Represents the total number of Non-Volatile Memory Storage Locations supported. |
| SHORT_TO_GND_J1962 | Represents a query from the application to see if the pin identified in Value is valid for short to ground line on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH must be 0. LLLL is the short to ground line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| PGM_VOLTAGE_J1962 | Represents a query from the application to see if the pin identified in Value is valid for programming voltage on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is the programming voltage line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |
| J1850PWM_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the J1850PWM_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is BUS - line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| J1850VPW_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the J1850VPW_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |

| Parameter | Description | Associated Value |
|--------------------------|---|---|
| ISO9141_PS_K_LINE_J1962 | Represents a query from the application to see if the K line pin identified in Value is valid for the ISO9141_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is K line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL indicates where the L line would be if it is supported. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: L line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for L line in this pair of pins shall be conveyed via ISO9141_PS_L_LINE_J1962. |
| ISO9141_PS_L_LINE_J1962 | Represents a query from the application to see if the L line pin identified in Value is valid for the ISO9141_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH indicates where the K line would be if it is supported. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: K line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for K line in this pair of pins shall be conveyed via ISO9141_PS_K_LINE_J1962. |
| ISO14230_PS_K_LINE_J1962 | Represents a query from the application to see if the K line pin identified in Value is valid for the ISO14230_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is K line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL indicates where the L line would be if it is supported. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: L line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for L line in this pair of pins shall be conveyed via ISO14230_PS_L_LINE_J1962. |
| ISO14230_PS_L_LINE_J1962 | Represents a query from the application to see if the L line pin identified in Value is valid for the ISO14230_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH indicates where the K line would be if it is supported. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: K line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for K line in this pair of pins shall be conveyed via ISO14230_PS_K_LINE_J1962. |

| Parameter | Description | Associated Value |
|----------------------|---|---|
| CAN_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the CAN_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| ISO15765_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the ISO15765_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| FT_CAN_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the FT_CAN_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| FT_ISO15765_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the FT_ISO15765_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |

| Parameter | Description | Associated Value |
|-------------------------|--|--|
| SW_CAN_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the SW_CAN_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |
| SW_ISO15765_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the SW_ISO15765_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |
| GM_UART_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the GM_UART_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |
| UART_ECHO_BYTE_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the UART_EHO_BYTE_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |

| Parameter | Description | Associated Value |
|----------------------|--|---|
| HONDA_DIAGH_PS_J1962 | Represents a query from the application to see if the pin identified in Value is valid for the HONDA_DIAGH_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pin on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL must be 0. |
| J1939_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the J1939_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| J1708_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the J1708_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is BUS - line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| TP2_0_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the TP2_0_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |

| Parameter | Description | Associated Value |
|-------------------------|--|--|
| J2610_PS_J1962 | Represents a query from the application to see if the pins identified in Value are valid for the J2610_PS Protocol ID on the J1962 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is the Tx line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is the Rx line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| J1939_PS_J1939 | Represents a query from the application to see if the pins identified in Value are valid for the J1939_PS Protocol ID on the J1939-13 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is CAN H line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is CAN L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| J1708_PS_J1939 | Represents a query from the application to see if the pins identified in Value are valid for the J1708_PS Protocol ID on the J1939-13 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is BUS - line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |
| ISO9141_PS_K_LINE_J1939 | Represents a query from the application to see if the K line pin identified in Value is valid for the ISO9141_PS Protocol ID on the J1939 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is K line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL indicates where the L line would be if it is supported. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: L line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for L line in this pair of pins shall be conveyed via ISO9141_PS_L_LINE_J1939. |

| Parameter | Description | Associated Value |
|--------------------------|---|---|
| ISO9141_PS_L_LINE_J1939 | Represents a query from the application to see if the L line pin identified in Value is valid for the ISO9141_PS Protocol ID on the J1939 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH indicates where the K line would be if it is supported. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: K line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for K line in this pair of pins shall be conveyed via ISO9141_PS_K_LINE_J1939. |
| ISO14230_PS_K_LINE_J1939 | Represents a query from the application to see if the K line pin identified in Value is valid for the ISO14230_PS Protocol ID on the J1939 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is K line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL indicates where the L line would be if it is supported. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: L line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for L line in this pair of pins shall be conveyed via ISO14230_PS_L_LINE_J1939. |
| ISO14230_PS_L_LINE_J1939 | Represents a query from the application to see if the L line pin identified in Value is valid for the ISO14230_PS Protocol ID on the J1939 connector. Upon return, Supported indicates if the desired pin is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH indicates where the K line would be if it is supported. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is L line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. NOTE: K line is included for reference ONLY and shall not be considered when indicating SUPPORTED or NOT_SUPPORTED. Support for K line in this pair of pins shall be conveyed via ISO14230_PS_K_LINE_J1939. |
| J1708_PS_J1708 | Represents a query from the application to see if the pins identified in Value are valid for the J1708_PS Protocol ID on the J1708 connector. Upon return, Supported indicates if the desired pins are SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the state of the associated pins on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains a bit mapped unsigned long (with the format 0xHHHHLLLL) where the bit that corresponds to the desired pin location is set to 1. All other bits shall be set to 0. HHHH is BUS + line. Bit 16 corresponds to pin 1, bit 17 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. LLLL is BUS - line. Bit 0 corresponds to pin 1, bit 1 corresponds to pin 2, etc. For this field, it is invalid to set more than one bit at a time. |

FIGURE 95 - GET_DEVICE_INFO PARAMETER DETAILS

21.3.2.3 GET_PROTOCOL_INFO

The `IoctlID` value of `GET_PROTOCOL_INFO` is used to request the protocol specific capabilities of a J2534 device. It is expected that the application will call this function for each protocol it is interested in. The calling application is responsible for allocating and initializing the inputs described in Figure 96. Upon successful completion (a return code of `STATUS_NOERROR`), the corresponding *Supported* and/or *Value* elements shall have been updated. Requests for parameters that are not supported by the device shall cause the corresponding *Supported* element to be set to `NOT_SUPPORTED` and the corresponding *Value* to remain unaltered. In this case, the interface shall continue processing the rest of the parameters in the list and the return code shall be `STATUS_NOERROR` (as long as no other errors conditions exist). If the device does not support the requested Protocol ID, the return value shall be `ERR_INVALID_PROTOCOL_ID`.

| Parameter | Description |
|-----------|---|
| ChannelID | Device ID assigned by DLL during <code>PassThruOpen</code> |
| IoctlID | Is set to the define <code>GET_PROTOCOL_INFO</code> . |
| InputPtr | Points to an unsigned long that contains the Protocol ID. |
| OutputPtr | <p>Points to the structure <code>SPARAM_LIST</code>, which is defined as follows:</p> <pre>typedef struct { unsigned long NumOfParams; /* number of SPARAM elements */ SPARAM *ParamPtr; /* array of SPARAM */ } SPARAM_LIST</pre> <p>where:</p> <p><code>NumOfParams</code> is an INPUT, set by the application, which contains the number of SPARAM elements in the array pointed to by <code>ParamPtr</code>.</p> <p><code>ParamPtr</code> is an INPUT, set by the application, which points to an array of SPARAM structures allocated by the application.</p> <p>The structure <code>SPARAM</code> is defined as follows:</p> <pre>typedef struct { unsigned long Parameter; /* name of parameter */ unsigned long Value; /* value of the parameter */ unsigned long Supported; /* support for parameter */ } SPARAM</pre> <p>where:</p> <p><code>Parameter</code> is an INPUT, set by the application, which represents the ID of the parameter requested (see Figure 98 for a list of valid parameters).</p> <p><code>Value</code> is typically an OUTPUT, set by the interface, which represents the parameter's value. However, for certain parameters <code>Value</code> may also be an INPUT set by the application (see Figure 98 for more details).</p> <p><code>Supported</code> is an OUTPUT, set by the interface to 0 (indicating the associated parameter is <code>NOT_SUPPORTED</code>) or 1 (indicating the associated parameter is <code>SUPPORTED</code>). The contents of <code>Value</code> shall remain unchanged if the parameter is <code>NOT_SUPPORTED</code>.</p> |

FIGURE 96 - GET_PROTOCOL_INFO DETAILS

Figure 97 groups the Protocol IDs so that the parameter/description in Figure 98 can also indicate the applicable Protocol IDs.

| Group Number | List of Protocol IDs in the Group |
|--------------|---|
| 1 | J1850PWM, J1850PWM_PS, and J1850PWM_CHx |
| 2 | J1850VPW, J1850VPW_PS, and J1850VPW_CHx |
| 3 | ISO9141 and ISO9141_CHx |
| 4 | ISO9141_PS |
| 5 | ISO14230 and ISO14230_CHx |
| 6 | ISO14230_PS |
| 7 | CAN, CAN_PS, CAN_CHx, FT_CAN_PS, and FT_CAN_CHx |
| 8 | ISO15765, ISO15765_PS, ISO15765_CHx, FT_ISO15765_PS, FT_ISO15765_CHx |
| 9 | SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE, SCI_B_TRANS, J2610_PS, and J2610_CHx |
| 10 | SW_CAN_PS and SW_CAN_CHx |
| 11 | SW_ISO15765_PS and SW_ISO15765_CHx |
| 12 | GM_UART_PS and GM_UART_CHx |
| 13 | UART_ECHO_BYTE_PS and UART_ECHO_BYTE_CHx |
| 14 | HONDA_DIAGH_PS and HONDA_DIAGH_CHx |
| 15 | J1939_PS and J1939_CHx |
| 16 | J1708_PS and J1708_CHx |
| 17 | TP2_0_PS and TP2_0_CHx |
| 18 | ANALOG_IN_x |

FIGURE 97 - PROTOCOL GROUPS

| Parameter | Description | Associated Value | Applicable Protocol Groups |
|-----------------------------|---|---|---|
| RESOURCE_GROUP | Represents the resource group that this protocol belongs to. Protocols that have the same group number share a common resource so there operation is mutually exclusive (that is, only one member of the group may operate at a time). | Value is an OUTPUT that, upon return, contains an unsigned long that represent the associated group number. The value zero (0) indicates no resource multiplexing. (Other than 0, the values are arbitrary and may not be consistent across manufacturers.) | All |
| TIMESTAMP_RESOLUTION | Represents the resolution of the message timestamp. | Value is an OUTPUT that, upon return, contains the resolution of the Timestamp element in the PASSTHRU_MSG structure (in micro-seconds). | All |
| MAX_RX_BUFFER_SIZE | Represents the maximum size of the Receive Buffer Pool for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long that is the maximum size in bytes. | All |
| MAX_PASS_FILTER | Represents the maximum number of PASS_FILTERs that can be defined for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17 |
| MAX_BLOCK_FILTER | Represents the maximum number of BLOCK_FILTERs that can be defined for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17 |
| MAX_FILTER_MSG_LENGTH | Represents the maximum filter message size for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long that is the maximum size in bytes. | 1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17 |
| MAX_PERIODIC_MSGS | Represents the maximum number of periodic messages that can be defined for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 |
| MAX_PERIODIC_MSG_LENGTH | Represents the maximum periodic message size for a message for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long that is the maximum size in bytes. | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 |
| DESIRED_DATA_RATE | Represents a query from the application to see if the DATA_RATE identified in Value is supported (within +- 2%) for the given protocol. Upon return, Supported indicates if the desired DATA_RATE is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. This query shall not change the current DATA_RATE on the device. NOTE: The application may include this parameter one or more of times in the list. | Value is an INPUT that contains the desired DATA_RATE. The application is allowed to request any value. The interface shall never return ERR_INVALID_IOCTL_VALUE for this parameter (even if the requested value is invalid for the given protocol ID). | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 |
| MAX_REPEAT_MESSAGING | Represents the maximum number of repeat messages that can be defined for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. (A value of 0 indicates that Repeat Messaging is not supported for this protocol.) | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17 |
| MAX_REPEAT_MESSAGING_LENGTH | Represents the maximum repeat message size for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long that is the maximum size in bytes. (A value of 0 indicates that Repeat Messaging is not supported for this protocol.) | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17 |

| Parameter | Description | Associated Value | Applicable Protocol Groups |
|----------------------------|---|--|----------------------------|
| NETWORK_LINE_SUPPORTED | Represents the possible NETWORK_LINE selections for the given protocol. | Value is an OUTPUT that, upon return, contains a bit mapped unsigned long where 1 indicates supported and 0 indicates not supported. Bit 0 corresponds to BUS +. Bit 1 corresponds to BUS -. All other bits shall be 0. | 1 |
| MAX_FUNCT_MSG_LOOKUP | Represents the maximum number of entries in the Functional Message Look Up Table for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 1 |
| PARITY_SUPPORTED | Represents the possible PARITY selections for the given protocol. | Value is an OUTPUT that, upon return, contains a bit mapped unsigned long where 1 indicates supported and 0 indicates not supported. Bit 0 corresponds to no parity. Bit 1 corresponds to odd parity. Bit 2 corresponds to even parity. All other bits shall be 0. | 3, 4, 5, 6, 12 |
| DATA_BITS_SUPPORTED | Represents the possible DATA_BITS selections for the given protocol. | Value is an OUTPUT that, upon return, contains a bit mapped unsigned long where 1 indicates supported and 0 indicates not supported. Bit 0 corresponds to 8 data bits. Bit 1 corresponds to 7 data bits. All other bits shall be 0. | 3, 4, 5, 6, 12 |
| FIVE_BAUD_MOD_SUPPORTED | Represents the possible FIVE_BAUD_MOD selections for the given protocol. | Value is an OUTPUT that, upon return, contains a bit mapped unsigned long where 1 indicates supported and 0 indicates not supported. Bit 0 corresponds to FIVE_BAUD_MOD = 0. Bit 1 corresponds to FIVE_BAUD_MOD = 1. Bit 2 corresponds to FIVE_BAUD_MOD = 2. Bit 3 corresponds to FIVE_BAUD_MOD = 3. All other bits shall be 0. | 3, 4, 5, 6 |
| L_LINE_SUPPORTED | Represents support for the ISO9141/ISO14230 L line for the given protocol. Upon return, Supported indicates if L line is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. NOTE: L Line support for the ISO9141_PS and ISO14230_PS protocol IDs is obtained using GET_DEVICE_INFO | Value shall be not be used. | 3, 5 |
| CAN_11_29_IDS_SUPPORTED | Represent support of simultaneous 11 and 29 bit CAN IDs for the given protocol. Upon return, Supported indicates if simultaneous 11 and 29 bit CAN IDs is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. | Value shall be not be used. | 7, 8, 10, 11 |
| CAN_MIXED_FORMAT_SUPPORTED | Represents support of CAN_MIXED_FORMAT for the given protocol. Upon return, Supported indicates if CAN_MIXED_FORMAT is SUPPORTED or NOT_SUPPORTED; the contents of Value shall remain un-altered. | Value shall be not be used. | 8, 11 |

| Parameter | Description | Associated Value | Applicable Protocol Groups |
|----------------------------|--|---|----------------------------|
| MAX_FLOW_CONTROL_FILTER | Represents the maximum number of FLOW_CONTROL_FILTERs that can be defined for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 8, 11 |
| MAX_ISO15765_WFT_MAX | Represents the maximum value of ISO15765_WFT_MAX for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 8, 11 |
| MAX_AD_ACTIVE_CHANNELS | Represents which channels are capable of being sampled, matching the format of ACTIVE_CHANNELS for the given protocol. | Value is an OUTPUT that, upon return, contains a bit mapped unsigned long (matching the format of ACTIVE_CHANNELS) where 1 indicates supported and 0 indicates not supported. | 18 |
| MAX_AD_SAMPLE_RATE | Represents the maximum value of SAMPLE_RATE for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 18 |
| MAX_AD_SAMPLES_PER_READING | Represents the maximum value of SAMPLES_PER_READING for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 18 |
| AD_SAMPLE_RESOLUTION | Represents the value of SAMPLE_RESOLUTION for the given protocol. | Value is an OUTPUT that, upon return, contains an unsigned long. | 18 |
| AD_INPUT_RANGE_LOW | Represents the value of INPUT_RANGE_LOW for the given protocol. | Value is an OUTPUT that, upon return, contains a signed long. | 18 |
| AD_INPUT_RANGE_HIGH | Represents the value of INPUT_RANGE_HIGH for the given protocol. | Value is an OUTPUT that, upon return, contains a signed long. | 18 |

FIGURE 98 - GET_PROTOCOL_INFO PARAMETER DETAILS

Examples:

1. A device that supports the J2534-1 protocol ID of ISO9141. Then re-uses that hardware resource for the J2534-2 protocol ID of ISO9141_PS (supporting K line on pin 7 and L line on pin 15 OR K line on pin 1 and L line on pin 9; only one channel at a time, no simultaneous operation) and provides two additional independent channels that can operate simultaneously using the protocol IDs ISO9141_CH1 and ISO9141_CH2. In this case, the device shall return the value of 0x00020101 for ISO9141_SIMULTANEOUS and the value of 0x00020201 for ISO9141_SUPPORTED. Additionally, ISO9141 and ISO9141_PS would have the same non-zero value for RESOURCE_GROUP, but ISO9141_CH1 and ISO9141_CH2 would both have the value 0x0 for RESOURCE_GROUP.
2. A device that supports the J2534-1 protocol ID of ISO15765, does not support ISO15765_PS, and provides four additional independent channels that can operate simultaneously using the protocol IDs ISO15765_CH1, ISO15765_CH2, ISO15765_CH3, and ISO15765_CH4 shall return the value of 0x00040001 for ISO15765_SIMULTANEOUS and the value of 0x00040001 for ISO15765_SUPPORTED. Additionally, each protocol ID (ISO15765, ISO15765_CH1, ISO15765_CH2, ISO15765_CH3, and ISO15765_CH4) would have the value 0x0 for RESOURCE_GROUP.
3. A device that supports the J2534-1 protocol ID of ISO14230, does not support ISO14230_PS, and provides two additional channels using the Protocol IDs ISO14230_CH1 and ISO14230_CH2. ISO14230 and ISO14230_CH1 share the same hardware resource and cannot operate simultaneously, but ISO14230_CH2 is independent. In this case, the device shall return the value of 0x00020001 for ISO14230_SIMULTANEOUS and the value of 0x00020001 for ISO14230_SUPPORTED. Additionally, ISO14230 and ISO14230_CH1 would have the same non-zero value for RESOURCE_GROUP and ISO14230_CH2 would have the value 0x0 for RESOURCE_GROUP.

4. A device that supports the J2534-1 protocol ID of J1850PWM and J1850VPW (which share the same pins on the J1962 connector), does not support J1850PWM_PS or J1850VPW_PS, and provides two additional independent J1850PWM channels that can operate simultaneously using the protocol IDs J1850PWM_CH1, and J1850PWM_CH2. In this case, the device shall return the value of 0x00020001 for J1850PWM_SIMULTANEOUS, the value of 0x00020001 for J1850PWM_SUPPORTED, the value of 0x00000001 for J1850VPW_SIMULTANEOUS, and the value of 0x00000001 for J1850VPW_SUPPORTED. Additionally, J1850PWM and J1850VPW would have the same non-zero value for RESOURCE_GROUP while J1850PWM_CH1, and J1850PWM_CH2 would both have the value 0x0 for RESOURCE_GROUP.

22. SAE J2534-2 RESOURCE

This section defines and manages the common resources reserved for SAE J2534-2. The RxStatus and TxFlags are specified in each optional feature section, as the values may be reused.

22.1 Connect Flag Values

Figure 99 identifies the Connect Flag values (for the function PassThruConnect) assigned by SAE J2534-2.

| Definition | Flag Bit(s) | Description | Value |
|------------|-------------|----------------------------|-------------------------------|
| Reserved | 16 - 23 | Reserved for SAE J2534-2 | Unused bits shall be set to 0 |
| Reserved | 24 - 31 | Tool Manufacturer Specific | Unused bits shall be set to 0 |

FIGURE 99 - CONNECT FLAG VALUES

In addition to the above this document uses the following Connect Flag that will be defined in the next revision of the SAE J2534-1 specification.

| Name | Bit(s) | Description |
|-------------------|--------|--|
| CHECKSUM_DISABLED | 9 | 0 = For transmission, the interface shall generate and appends the checksum. For reception, the interface shall verify and remove the checksum. Messages with invalid checksums will be discarded. 1 = The interface shall not generate or verify the checksum; the entire message shall be treated as data by the interface. |

FIGURE 100 - NEW J2534-1 CONNECT FLAGS

22.2 ProtocolID Values

SAE J2534-1 reserved ProtocolID values for SAE. Figure 101 defines the value assignment for each ProtocolID definition identified in this document.

| Definition | Value(s) |
|--------------------------|-------------------------|
| J1850VPW_PS | 0x00008000 |
| J1850PWM_PS | 0x00008001 |
| ISO9141_PS | 0x00008002 |
| ISO14230_PS | 0x00008003 |
| CAN_PS | 0x00008004 |
| ISO15765_PS | 0x00008005 |
| J2610_PS | 0x00008006 |
| SW_ISO15765_PS | 0x00008007 |
| SW_CAN_PS | 0x00008008 |
| GM_UART_PS | 0x00008009 |
| UART_ECHO_BYTE_PS | 0x0000800A |
| HONDA_DIAGH_PS | 0x0000800B |
| J1939_PS | 0x0000800C |
| J1708_PS | 0x0000800D |
| TP2_0_PS | 0x0000800E |
| FT_CAN_PS | 0x0000800F |
| FT_ISO15765_PS | 0x00008010 |
| Reserved for SAE J2534-2 | 0x00008011 – 0x00008FFF |
| CAN_CH1 | 0x00009000 |
| CAN_CH2 | CAN_CH1 + 1 |
| | |
| CAN_CH128 | CAN_CH1 + 127 |
| J1850VPW_CH1 | 0x00009080 |
| J1850VPW_CH2 | J1850VPW_CH1 + 1 |
| | |
| J1850VPW_CH128 | J1850VPW_CH1 + 127 |
| J1850PWM_CH1 | 0x00009100 |
| J1850PWM_CH2 | J1850PWM_CH1 + 1 |
| | |
| J1850PWM_CH128 | J1850PWM_CH1 + 127 |
| ISO9141_CH1 | 0x00009180 |
| ISO9141_CH2 | ISO9141_CH1 + 1 |
| | |
| ISO9141_CH128 | ISO9141_CH1 + 127 |
| ISO14230_CH1 | 0x00009200 |
| ISO14230_CH2 | ISO14230_CH1 + 1 |
| | |
| ISO14230_CH128 | ISO14230_CH1 + 127 |
| ISO15765_CH1 | 0x00009280 |
| ISO15765_CH2 | ISO15765_CH1 + 1 |
| | |
| ISO15765_CH128 | ISO15765_CH1 + 127 |

| | |
|--------------------------|---------------------------|
| SW_CAN_CAN_CH1 | 0x00009300 |
| SW_CAN_CAN_CH2 | SW_CAN_CAN_CH1 + 1 |
| | |
| SW_CAN_CAN_CH128 | SW_CAN_CAN_CH1 + 127 |
| | |
| SW_CAN_ISO15765_CH1 | 0x00009380 |
| SW_CAN_ISO15765_CH2 | SW_CAN_ISO15765_CH1 + 1 |
| | |
| SW_CAN_ISO15765_CH128 | SW_CAN_ISO15765_CH1 + 127 |
| | |
| J2610_CH1 | 0x00009400 |
| J2610_CH2 | J2610_CH1 + 1 |
| | |
| J2610_CH128 | J2610_CH1 + 127 |
| | |
| FT_CAN_CH1 | 0x00009480 |
| FT_CAN_CH2 | FT_CAN_CH1 + 1 |
| | |
| FT_CAN_CH128 | FT_CAN_CH1 + 127 |
| | |
| FT_ISO15765_CH1 | 0x00009500 |
| FT_ISO15765_CH2 | FT_ISO15765_CH1 + 1 |
| | |
| FT_ISO15765_CH128 | FT_ISO15765_CH1 + 127 |
| | |
| GM_UART_CH1 | 0x00009580 |
| GM_UART_CH2 | GM_UART_CH1 + 1 |
| ... | |
| GM_UART_CH128 | GM_UART_CH1 +127 |
| | |
| ECHO_BYTE_CH1 | 0x00009600 |
| ECHO_BYTE_CH2 | ECHO_BYTE_CH1 + 1 |
| ... | |
| ECHO_BYTE_CH128 | ECHO_BYTE_CH1 +127 |
| | |
| HONDA_DIAGH_CH1 | 0x00009680 |
| HONDA_DIAGH_CH2 | HONDA_DIAGH_CH1 + 1 |
| ... | |
| HONDA_DIAGH_CH128 | HONDA_DIAGH_CH1 +127 |
| | |
| J1939_CH1 | 0x00009700 |
| J1939_CH2 | J1939_CH1 + 1 |
| ... | |
| J1939_CH128 | J1939_CH1 +127 |
| | |
| J1708_CH1 | 0x00009780 |
| J1708_CH2 | J1708_CH1 + 1 |
| ... | |
| J1708_CH128 | J1708_CH1 +127 |
| | |
| TP2_0_CH1 | 0x00009800 |
| TP2_0_CH2 | TP2_0_CH1 + 1 |
| ... | |
| TP2_0_CH128 | TP2_0_CH1 +127 |
| | |
| Reserved for SAE J2534-2 | 0x00009880 – 0x0000BFFF |
| | |

| | |
|----------------------------|-------------------------|
| ANALOG_IN_1 | 0x0000C000 |
| ANALOG_IN_2 | ANALOG_IN_1 + 1 |
| ... | |
| ANALOG_IN_32 | ANALOG_IN_1 + 31 |
| Reserved for SAE J2534-2 | 0x0000C020 – 0x0000FFFF |
| Tool Manufacturer Specific | 0x00010000 – 0xFFFFFFFF |

FIGURE 101 - PROTOCOLID VALUES

22.3 Filter Type Values

| Definition | Value(s) |
|----------------------------|-------------------------|
| Reserved for SAE J2534-2 | 0x00008000 – 0x0000FFFF |
| Tool Manufacturer Specific | 0x00010000 – 0xFFFFFFFF |

FIGURE 102 - FILTER TYPE VALUES

22.4 ioctl ID Values

| Definition | Value(s) |
|----------------------------|-------------------------|
| SW_CAN_HS | 0x00008000 |
| SW_CAN_NS | 0x00008001 |
| SET_POLL_RESPONSE | 0x00008002 |
| BECOME_MASTER | 0x00008003 |
| START_REPEAT_MESSAGE | 0x00008004 |
| QUERY_REPEAT_MESSAGE | 0x00008005 |
| STOP_REPEAT_MESSAGE | 0x00008006 |
| GET_DEVICE_CONFIG | 0x00008007 |
| SET_DEVICE_CONFIG | 0x00008008 |
| PROTECT_J1939_ADDR | 0x00008009 |
| REQUEST_CONNECTION | 0x0000800A |
| TEARDOWN_CONNECTION | 0x0000800B |
| GET_DEVICE_INFO | 0x0000800C |
| GET_PROTOCOL_INFO | 0x0000800D |
| Reserved for SAE J2534-2 | 0x0000800E – 0x0000FFFF |
| Tool Manufacturer Specific | 0x00010000 – 0xFFFFFFFF |

FIGURE 103 - IOCTL ID VALUES

22.5 IOCTL GET / SET CONFIG Parameter Details

SAE J2534-1 reserved ID values for SAE J2534-2. Figure 104 defines the value assignment for each parameter identified in this document.

| Parameter | ID Value |
|---|-------------------------|
| CAN_MIXED_FORMAT | 0x00008000 |
| J1962_PINS | 0x00008001 |
| Reserved for SAE J2534-2 | 0x00008002 - 0x0000800F |
| SW_CAN_HS_DATA_RATE | 0x00008010 |
| SW_CAN_SPEEDCHANGE_ENABLE | 0x00008011 |
| SW_CAN_RES_SWITCH | 0x00008012 |
| Reserved for SAE J2534-2 | 0x00008013 - 0x0000801F |
| ACTIVE_CHANNELS | 0x00008020 |
| SAMPLE_RATE | 0x00008021 |
| SAMPLES_PER_READING | 0x00008022 |
| READINGS_PER_MSG | 0x00008023 |
| AVERAGING_METHOD | 0x00008024 |
| SAMPLE_RESOLUTION | 0x00008025 |
| INPUT_RANGE_LOW | 0x00008026 |
| INPUT_RANGE_HIGH | 0x00008027 |
| UEB_T0_MIN | 0x00008028 |
| UEB_T1_MAX | 0x00008029 |
| UEB_T2_MAX | 0x0000802A |
| UEB_T3_MAX | 0x0000802B |
| UEB_T4_MIN | 0x0000802C |
| UEB_T5_MAX | 0x0000802D |
| UEB_T6_MAX | 0x0000802E |
| UEB_T7_MIN | 0x0000802F |
| UEB_T7_MAX | 0x00008030 |
| UEB_T9_MIN | 0x00008031 |
| Reserved for SAE J2534-2 | 0x00008032 - 0x0000803C |
| J1939_PINS | 0x0000803D |
| J1708_PINS | 0x0000803E |
| J1939_T1 | 0x0000803F |
| J1939_T2 | 0x00008040 |
| J1939_T3 | 0x00008041 |
| J1939_T4 | 0x00008042 |
| J1939_BRDCST_MIN_DELAY | 0x00008043 |
| TP2_0_T_BR_INT | 0x00008044 |
| TP2_0_T_E | 0x00008045 |
| TP2_0_MNTC | 0x00008046 |
| TP2_0_T_CTA | 0x00008047 |
| TP2_0_MNCT | 0x00008048 |
| TP2_0_MNTB | 0x00008049 |
| TP2_0_MNT | 0x0000804A |
| TP2_0_T_Wait | 0x0000804B |
| TP2_0_T1 | 0x0000804C |
| TP2_0_T3 | 0x0000804D |
| TP2_0_IDENTIFER | 0x0000804E |
| TP2_0_RXIDPASSIVE | 0x0000804F |
| Reserved for SAE J2534-2 | 0x00008050 - 0x0000C000 |
| Reserved for SAE J2534-2 (Device Configurations) | 0x0000C001 - 0x0000FFFF |
| Tool Manufacturer Specific | 0x00010000 - 0xFFFFFFFF |

FIGURE 104 - IOCTL GET_CONFIG / SET_CONFIG ID VALUES

22.6 GET_DEVICE_CONFIG/ SET_DEVICE_CONFIG Parameter Details

| Definition | Value(s) |
|--------------------------|-------------------------|
| NON_VOLATILE_STORE_1 | 0x0000C001 |
| NON_VOLATILE_STORE_2 | 0x0000C002 |
| NON_VOLATILE_STORE_3 | 0x0000C003 |
| NON_VOLATILE_STORE_4 | 0x0000C004 |
| NON_VOLATILE_STORE_5 | 0x0000C005 |
| NON_VOLATILE_STORE_6 | 0x0000C006 |
| NON_VOLATILE_STORE_7 | 0x0000C007 |
| NON_VOLATILE_STORE_8 | 0x0000C008 |
| NON_VOLATILE_STORE_9 | 0x0000C009 |
| NON_VOLATILE_STORE_10 | 0x0000C00A |
| Reserved for SAE J2534-2 | 0x0000C00B – 0x0000FFFF |

FIGURE 105 - IOCTL GET_CONFIG / SET_CONFIG DEVICE DETAILS

22.7 GET_DEVICE_INFO Defines

| Definition | Value(s) |
|--------------------------|------------|
| SERIAL_NUMBER | 0x00000001 |
| J1850PWM_SUPPORTED | 0x00000002 |
| J1850VPW_SUPPORTED | 0x00000003 |
| ISO9141_SUPPORTED | 0x00000004 |
| ISO14230_SUPPORTED | 0x00000005 |
| CAN_SUPPORTED | 0x00000006 |
| ISO15765_SUPPORTED | 0x00000007 |
| SCI_A_ENGINE_SUPPORTED | 0x00000008 |
| SCI_A_TRANS_SUPPORTED | 0x00000009 |
| SCI_B_ENGINE_SUPPORTED | 0x0000000A |
| SCI_B_TRANS_SUPPORTED | 0x0000000B |
| SW_ISO15765_SUPPORTED | 0x0000000C |
| SW_CAN_SUPPORTED | 0x0000000D |
| GM_UART_SUPPORTED | 0x0000000E |
| UART_ECHO_BYTE_SUPPORTED | 0x0000000F |
| HONDA_DIAGH_SUPPORTED | 0x00000010 |
| J1939_SUPPORTED | 0x00000011 |
| J1708_SUPPORTED | 0x00000012 |
| TP2_0_SUPPORTED | 0x00000013 |
| J2610_SUPPORTED | 0x00000014 |
| ANALOG_IN_SUPPORTED | 0x00000015 |
| MAX_NON_VOLATILE_STORAGE | 0x00000016 |
| SHORT_TO_GND_J1962 | 0x00000017 |
| PGM_VOLTAGE_J1962 | 0x00000018 |
| J1850PWM_PS_J1962 | 0x00000019 |
| J1850VPW_PS_J1962 | 0x0000001A |
| ISO9141_PS_K_LINE_J1962 | 0x0000001B |
| ISO9141_PS_L_LINE_J1962 | 0x0000001C |
| ISO14230_PS_K_LINE_J1962 | 0x0000001D |
| ISO14230_PS_L_LINE_J1962 | 0x0000001E |
| CAN_PS_J1962 | 0x0000001F |
| ISO15765_PS_J1962 | 0x00000020 |
| SW_CAN_PS_J1962 | 0x00000021 |

| | |
|-----------------------------|-------------------------|
| SW_ISO15765_PS_J1962 | 0x00000022 |
| GM_UART_PS_J1962 | 0x00000023 |
| UART_ECHO_BYTE_PS_J1962 | 0x00000024 |
| HONDA_DIAGH_PS_J1962 | 0x00000025 |
| J1939_PS_J1962 | 0x00000026 |
| J1708_PS_J1962 | 0x00000027 |
| TP2_0_PS_J1962 | 0x00000028 |
| J2610_PS_J1962 | 0x00000029 |
| J1939_PS_J1939 | 0x0000002A |
| J1708_PS_J1939 | 0x0000002B |
| ISO9141_PS_K_LINE_J1939 | 0x0000002C |
| ISO9141_PS_L_LINE_J1939 | 0x0000002D |
| ISO14230_PS_K_LINE_J1939 | 0x0000002E |
| ISO14230_PS_L_LINE_J1939 | 0x0000002F |
| J1708_PS_J1708 | 0x00000030 |
| FT_CAN_SUPPORTED | 0x00000031 |
| FT_ISO15765_SUPPORTED | 0x00000032 |
| FT_CAN_PS_J1962 | 0x00000033 |
| FT_ISO15765_PS_J1962 | 0x00000034 |
| J1850PWM_SIMULTANEOUS | 0x00000035 |
| J1850VPW_SIMULTANEOUS | 0x00000036 |
| ISO9141_SIMULTANEOUS | 0x00000037 |
| ISO14230_SIMULTANEOUS | 0x00000038 |
| CAN_SIMULTANEOUS | 0x00000039 |
| ISO15765_SIMULTANEOUS | 0x0000003A |
| SCI_A_ENGINE_SIMULTANEOUS | 0x0000003B |
| SCI_A_TRANS_SIMULTANEOUS | 0x0000003C |
| SCI_B_ENGINE_SIMULTANEOUS | 0x0000003D |
| SCI_B_TRANS_SIMULTANEOUS | 0x0000003E |
| SW_ISO15765_SIMULTANEOUS | 0x0000003F |
| SW_CAN_SIMULTANEOUS | 0x00000040 |
| GM_UART_SIMULTANEOUS | 0x00000041 |
| UART_ECHO_BYTE_SIMULTANEOUS | 0x00000042 |
| HONDA_DIAGH_SIMULTANEOUS | 0x00000043 |
| J1939_SIMULTANEOUS | 0x00000044 |
| J1708_SIMULTANEOUS | 0x00000045 |
| TP2_0_SIMULTANEOUS | 0x00000046 |
| J2610_SIMULTANEOUS | 0x00000047 |
| ANALOG_IN_SIMULTANEOUS | 0x00000048 |
| PART_NUMBER | 0x00000049 |
| FT_CAN_SIMULTANEOUS | 0x0000004A |
| FT_ISO15765_SIMULTANEOUS | 0x0000004B |
| Reserved for SAE J2534-2 | 0x0000004C – 0xFFFFFFFF |

FIGURE 106 - GET_DEVICE_INFO DEFINE VALUES

22.8 GET_PROTOCOL_INFO Defines

| Definition | Value(s) |
|-----------------------------|-------------------------|
| MAX_RX_BUFFER_SIZE | 0x00000001 |
| MAX_PASS_FILTER | 0x00000002 |
| MAX_BLOCK_FILTER | 0x00000003 |
| MAX_FILTER_MSG_LENGTH | 0x00000004 |
| MAX_PERIODIC_MSGS | 0x00000005 |
| MAX_PERIODIC_MSG_LENGTH | 0x00000006 |
| DESIRED_DATA_RATE | 0x00000007 |
| MAX_REPEAT_MESSAGING | 0x00000008 |
| MAX_REPEAT_MESSAGING_LENGTH | 0x00000009 |
| NETWORK_LINE_SUPPORTED | 0x0000000A |
| MAX_FUNCT_MSG_LOOKUP | 0x0000000B |
| PARITY_SUPPORTED | 0x0000000C |
| DATA_BITS_SUPPORTED | 0x0000000D |
| FIVE_BAUD_MOD_SUPPORTED | 0x0000000E |
| L_LINE_SUPPORTED | 0x0000000F |
| CAN_11_29_IDS_SUPPORTED | 0x00000010 |
| CAN_MIXED_FORMAT_SUPPORTED | 0x00000011 |
| MAX_FLOW_CONTROL_FILTER | 0x00000012 |
| MAX_ISO15765_WFT_MAX | 0x00000013 |
| MAX_AD_ACTIVE_CHANNELS | 0x00000014 |
| MAX_AD_SAMPLE_RATE | 0x00000015 |
| MAX_AD_SAMPLES_PER_READING | 0x00000016 |
| AD_SAMPLE_RESOLUTION | 0x00000017 |
| AD_INPUT_RANGE_LOW | 0x00000018 |
| AD_INPUT_RANGE_HIGH | 0x00000019 |
| RESOURCE_GROUP | 0x0000001A |
| TIMESTAMP_RESOLUTION | 0x0000001B |
| Reserved for SAE J2534-2 | 0x0000001C – 0xFFFFFFFF |

FIGURE 107 - GET_PROTOCOL_INFO DEFINE VALUES

22.9 Error Return Values

| Definition | Value |
|-------------------------------|-----------------------|
| ERR_ADDRESS_NOT_CLAIMED | 0x00010000 |
| ERR_NO_CONNECTION_ESTABLISHED | 0x00010001 |
| ERR_RESOURCE_IN_USE | 0x00010002 |
| Reserved for SAE J2534-2 | 0x00010003-0xFFFFFFFF |

FIGURE 108 - ERROR RETURN VALUES

In addition to the above this document uses the Return Values that are in the process of being defined in the next revision of the SAE J2534-1 specification.

| Name | Value | Description |
|----------------------------|-------|--|
| ERR_INVALID_IOCTL_PARAM_ID | 0x1E | Invalid IOCTL Parameter ID (either in the reserved range or not appropriate for the current channel) |
| ERR_VOLTAGE_IN_USE | 0x1F | Programming voltage is currently being applied to another pin |
| ERR_PIN_IN_USE | 0x20 | Pin number specified is currently in use (another protocol is using the pin) |

FIGURE 109 - NEW J2534-1 RETURN VALUES

22.10 TxFlags Bits

Identifies the TxFlags bits (defined by J2534-2) that can be used in the PASSTHRU_MSG structure.

| Name | Bit(s) | Description |
|----------------------------|--------|-------------------------------|
| Reserved for SAE J2534-2 | 23-16 | Unused bits shall be set to 0 |
| Tool Manufacturer Specific | 31-24 | Unused bits shall be set to 0 |

FIGURE 110 - TXFLAGS VALUES

22.11 RxStatus Bits

Identifies the RxStatus bits (defined by J2534-2) that can be used in the PASSTHRU_MSG structure.

| Name | Bit(s) | Description |
|----------------------------|--------|-------------------------------|
| Reserved for SAE J2534-2 | 21-16 | Unused bits shall be set to 0 |
| Tool Manufacturer Specific | 31-24 | Unused bits shall be set to 0 |

FIGURE 111 - RXSTATUS BITS

23. NOTES

23.1 Marginal Indicia

A change bar (l) located in the left margin is for the convenience of the user in locating areas where technical revisions, not editorial changes, have been made to the previous issue of this document. An (R) symbol to the left of the document title indicates a complete revision of the document, including technical revisions. Change bars and (R) are not used in original publications, nor in documents that contain editorial changes only.

PREPARED BY THE VEHICLE E/E SYSTEMS DIAGNOSTIC STANDARDS COMMITTEE