

**(R) Recommended Practice for Pass-Thru Vehicle Programming****Foreword**

The use of reprogrammable memory technology in vehicle electronic control units (ECUs) has increased in recent years, and is expected to continue in the future. Use of this technology has increased the flexibility of being able to use a single ECU hardware part to be used in many different vehicle configurations, with the only difference being the software and calibrations programmed into the unit. Reprogramming of those ECUs in the service environment also allows for ease of field modification of system operation and calibrations. Variations in reprogramming capability and the multiple tools necessary to reprogram vehicles are a burden on aftermarket repair facilities that service different makes of vehicles.

This document describes a standardized system for programming that includes a standard personal computer (PC), standard interface to a software device driver, and an interface that connects between the PC and a programmable ECU in a vehicle. The purpose of this system is to facilitate programming of ECUs for all vehicle manufacturers using a single set of programming hardware. Programming software from multiple vehicle manufacturers will be able to execute on this set of hardware to program their unique ECUs.

The U.S. Environmental Protection Agency (EPA) and the California Air Resources Board (ARB) have been working with vehicle manufacturers to provide the aftermarket with increased capability to service emission-related ECUs for all vehicles with a minimal investment in hardware needed to communicate with the vehicles. Both agencies have issued regulations that will require standardized programming tools to be used for all vehicle manufacturers. The Society of Automotive Engineers (SAE) developed this Recommended Practice to satisfy the intent of the U.S. EPA and the California ARB.

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 © 2004 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: 724-776-4970 (outside USA)  
Fax: 724-776-0790  
Email: [custsvc@sae.org](mailto:custsvc@sae.org)  
SAE WEB ADDRESS: <http://www.sae.org>

## TABLE OF CONTENTS

|         |                                                        |    |
|---------|--------------------------------------------------------|----|
| 1.      | Scope .....                                            | 5  |
| 2.      | References .....                                       | 5  |
| 2.1     | Applicable Documents .....                             | 5  |
| 2.1.1   | SAE Publications.....                                  | 5  |
| 2.1.2   | ISO Documents.....                                     | 6  |
| 3.      | Definitions.....                                       | 6  |
| 4.      | Acronyms .....                                         | 6  |
| 5.      | Pass-Thru Concept .....                                | 7  |
| 6.      | Pass-Thru System Requirements .....                    | 8  |
| 6.1     | PC Requirements .....                                  | 8  |
| 6.2     | Software Requirements and Assumptions.....             | 8  |
| 6.3     | Connection to PC .....                                 | 9  |
| 6.4     | Connection to Vehicle .....                            | 9  |
| 6.5     | Communication Protocols .....                          | 9  |
| 6.5.1   | ISO 9141 .....                                         | 9  |
| 6.5.2   | ISO 14230-4 (KWP2000) .....                            | 10 |
| 6.5.3   | SAE J1850 41.6 kbps PWM (Pulse Width Modulation) ..... | 10 |
| 6.5.4   | SAE J1850 10.4 kbps VPW (Variable Pulse Width) .....   | 10 |
| 6.5.5   | CAN.....                                               | 11 |
| 6.5.6   | ISO 15765-4 (CAN).....                                 | 11 |
| 6.5.7   | SAE J2610 DaimlerChrysler SCI.....                     | 11 |
| 6.6     | Simultaneous Communication on Multiple Protocols.....  | 11 |
| 6.7     | Programmable Power Supply.....                         | 12 |
| 6.8     | Pin Usage.....                                         | 13 |
| 6.9     | Data Buffering .....                                   | 14 |
| 6.10    | Error Recovery .....                                   | 14 |
| 6.10.1  | Device Not Connected .....                             | 14 |
| 6.10.2  | Bus Errors .....                                       | 14 |
| 7.      | Win32 Application Programming Interface .....          | 15 |
| 7.1     | API Functions – Overview.....                          | 15 |
| 7.2     | API Functions - Detailed Information .....             | 15 |
| 7.2.1   | PassThruOpen .....                                     | 15 |
| 7.2.1.1 | C/C++ Prototype .....                                  | 15 |
| 7.2.1.2 | Parameters.....                                        | 16 |
| 7.2.1.3 | Return Values .....                                    | 16 |
| 7.2.2   | PassThruClose.....                                     | 16 |
| 7.2.2.1 | C/C++ Prototype .....                                  | 16 |
| 7.2.2.2 | Parameters.....                                        | 16 |
| 7.2.2.3 | Return Values .....                                    | 17 |
| 7.2.3   | PassThruConnect .....                                  | 17 |
| 7.2.3.1 | C/C++ Prototype .....                                  | 17 |
| 7.2.3.2 | Parameters.....                                        | 17 |
| 7.2.3.3 | Flag Values .....                                      | 18 |
| 7.2.3.4 | Protocol ID Values .....                               | 19 |

# SAE J2534-1 Revised DEC2004

|          |                                     |    |
|----------|-------------------------------------|----|
| 7.2.3.5  | Return Values .....                 | 20 |
| 7.2.4    | PassThruDisconnect .....            | 20 |
| 7.2.4.1  | C/C++ Prototype .....               | 20 |
| 7.2.4.2  | Parameters .....                    | 21 |
| 7.2.4.3  | Return Values .....                 | 21 |
| 7.2.5    | PassThruReadMsgs .....              | 21 |
| 7.2.5.1  | C/C++ Prototype .....               | 22 |
| 7.2.5.2  | Parameters .....                    | 22 |
| 7.2.5.3  | Return Values .....                 | 23 |
| 7.2.6    | PassThruWriteMsgs .....             | 23 |
| 7.2.6.1  | C/C++ Prototype .....               | 24 |
| 7.2.6.2  | Parameters .....                    | 24 |
| 7.2.6.3  | Return Values .....                 | 25 |
| 7.2.7    | PassThruStartPeriodicMsg .....      | 26 |
| 7.2.7.1  | C/C++ Prototype .....               | 26 |
| 7.2.7.2  | Parameters .....                    | 26 |
| 7.2.7.3  | Return Values .....                 | 27 |
| 7.2.8    | PassThruStopPeriodicMsg .....       | 27 |
| 7.2.8.1  | C/C++ Prototype .....               | 28 |
| 7.2.8.2  | Parameters .....                    | 28 |
| 7.2.8.3  | Return Values .....                 | 28 |
| 7.2.9    | PassThruStartMsgFilter .....        | 28 |
| 7.2.9.1  | C/C++ Prototype .....               | 31 |
| 7.2.9.2  | Parameters .....                    | 31 |
| 7.2.9.3  | Filter Types .....                  | 32 |
| 7.2.9.4  | Return Values .....                 | 33 |
| 7.2.10   | PassThruStopMsgFilter .....         | 33 |
| 7.2.10.1 | C/C++ Prototype .....               | 33 |
| 7.2.10.2 | Parameters .....                    | 34 |
| 7.2.10.3 | Return Values .....                 | 34 |
| 7.2.11   | PassThruSetProgrammingVoltage ..... | 34 |
| 7.2.11.1 | C/C++ Prototype .....               | 34 |
| 7.2.11.2 | Parameters .....                    | 35 |
| 7.2.11.3 | Voltage Values .....                | 35 |
| 7.2.11.4 | Return Values .....                 | 35 |
| 7.2.12   | PassThruReadVersion .....           | 36 |
| 7.2.12.1 | C/C++ Prototype .....               | 36 |
| 7.2.12.2 | Parameters .....                    | 36 |
| 7.2.12.3 | Return Values .....                 | 37 |
| 7.2.13   | PassThruGetLastError .....          | 37 |
| 7.2.13.1 | C/C++ Prototype .....               | 37 |
| 7.2.13.2 | Parameters .....                    | 37 |
| 7.2.13.3 | Return Values .....                 | 37 |
| 7.2.14   | PassThruIoctl .....                 | 38 |
| 7.2.14.1 | C/C++ Prototype .....               | 38 |
| 7.2.14.2 | Parameters .....                    | 38 |
| 7.2.14.3 | Ioctl ID Values .....               | 39 |
| 7.2.14.4 | Return Values .....                 | 39 |
| 7.3      | IOCTL Section .....                 | 40 |
| 7.3.1    | GET_CONFIG .....                    | 41 |
| 7.3.2    | SET_CONFIG .....                    | 42 |

## SAE J2534-1 Revised DEC2004

|            |                                                               |    |
|------------|---------------------------------------------------------------|----|
| 7.3.3      | READ_VBATT .....                                              | 46 |
| 7.3.4      | READ_PROG_VOLTAGE .....                                       | 46 |
| 7.3.5      | FIVE_BAUD_INIT .....                                          | 47 |
| 7.3.6      | FAST_INIT .....                                               | 47 |
| 7.3.7      | CLEAR_TX_BUFFER .....                                         | 48 |
| 7.3.8      | CLEAR_RX_BUFFER .....                                         | 48 |
| 7.3.9      | CLEAR_PERIODIC_MSGS .....                                     | 49 |
| 7.3.10     | CLEAR_MSG_FILTERS .....                                       | 49 |
| 7.3.11     | CLEAR_FUNCT_MSG_LOOKUP_TABLE .....                            | 49 |
| 7.3.12     | ADD_TO_FUNCT_MSG_LOOKUP_TABLE .....                           | 50 |
| 7.3.13     | DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE .....                      | 50 |
| 8.         | Message Structure .....                                       | 51 |
| 8.1        | C/C++ Definition .....                                        | 51 |
| 8.2        | Elements .....                                                | 51 |
| 8.3        | Message Data Formats .....                                    | 52 |
| 8.4        | Format Checks for Messages Passed to the API .....            | 53 |
| 8.5        | Conventions for Returning Messages from the API .....         | 53 |
| 8.6        | Conventions for Returning Indications from the API .....      | 53 |
| 8.7        | Message Flag and Status Definitions .....                     | 54 |
| 8.7.1      | RxStatus .....                                                | 54 |
| 8.7.2      | RxStatus Bits for Messaging Status and Error Indication ..... | 55 |
| 8.7.3      | TxFlags .....                                                 | 56 |
| 9.         | DLL Installation and Registry .....                           | 57 |
| 9.1        | Naming of Files .....                                         | 57 |
| 9.2        | Win32 Registry .....                                          | 57 |
| 9.2.1      | User Application Interaction with the Registry .....          | 59 |
| 9.2.2      | Attaching to the DLL from an application .....                | 60 |
| 9.2.2.1    | Export Library Definition File .....                          | 61 |
| 10.        | Return Value Error Codes .....                                | 61 |
| 11.        | Notes .....                                                   | 63 |
| 11.1       | Marginal Indicia .....                                        | 63 |
| Appendix A | General ISO 15765-2 Flow Control Example .....                | 64 |
| A.1        | Flow Control Overview .....                                   | 64 |
| A.1.1      | Examples Overview .....                                       | 65 |
| A.2        | Transmitting a Segmented Message .....                        | 66 |
| A.2.1      | Conversation Setup .....                                      | 66 |
| A.2.2      | Data Transmission .....                                       | 67 |
| A.2.3      | Verification .....                                            | 68 |
| A.3        | Transmitting an Unsegmented Message .....                     | 69 |
| A.3.1      | Data Transmission .....                                       | 70 |
| A.3.2      | Verification .....                                            | 70 |
| A.4        | Receiving a Segmented Message .....                           | 70 |
| A.4.1      | Conversation Setup .....                                      | 70 |
| A.4.2      | Reception Notification .....                                  | 70 |
| A.4.3      | Data Reception .....                                          | 71 |
| A.5        | Receiving and Unsegmented Messages .....                      | 72 |

## **1. Scope**

This SAE Recommended Practice provides the framework to allow reprogramming software applications from all vehicle manufacturers the flexibility to work with multiple vehicle data link interface tools from multiple tool suppliers. This system enables each vehicle manufacturer to control the programming sequence for electronic control units (ECUs) in their vehicles, but allows a single set of programming hardware and vehicle interface to be used to program modules for all vehicle manufacturers.

This document does not limit the hardware possibilities for the connection between the PC used for the software application and the tool (e.g., RS-232, RS-485, USB, Ethernet...). Tool suppliers are free to choose the hardware interface appropriate for their tool. The goal of this document is to ensure that reprogramming software from any vehicle manufacturer is compatible with hardware supplied by any tool manufacturer.

U.S. Environmental Protection Agency (EPA) and the California Air Resources Board (ARB) "OBD service information" regulations include requirements for reprogramming emission-related control modules in vehicles for all manufacturers by the aftermarket repair industry. This document is intended to conform to those regulations for 2004 and later model year vehicles. For some vehicles, this interface can also be used to reprogram emission-related control modules in vehicles prior to the 2004 model year, and for non-emission related control modules. For other vehicles, this usage may require additional manufacturer specific capabilities to be added to a fully compliant interface. A second part to this document, SAE J2534-2, is planned to include expanded capabilities that tool suppliers can optionally include in an interface to allow programming of these additional non-mandated vehicle applications. In addition to reprogramming capability, this interface is planned for use in OBD compliance testing as defined in SAE J1699-3. SAE J2534-1 includes some capabilities that are not required for Pass-Thru Programming, but which enable use of this interface for those other purposes without placing a significant burden on the interface manufacturers.

Additional requirements for future model years may require revision of this document, most notably the inclusion of SAE J1939 for some heavy-duty vehicles. This document will be reviewed for possible revision after those regulations are finalized and requirements are better understood. Possible revisions include SAE J1939 specific software and an alternate vehicle connector, but the basic hardware of an SAE J2534 interface device is expected to remain unchanged.

## **2. References**

### **2.1 Applicable Publications**

The following publications form a part of this specification to the extent specified herein. Unless otherwise indicated, the latest version of SAE publications shall apply.

#### **2.1.1 SAE PUBLICATIONS**

Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE J1850—Class B Data Communications Network Interface

SAE J1939—Truck and Bus Control and Communications Network (Multiple Parts Apply)

SAE J1962—Diagnostic Connector

SAE J2610—DaimlerChrysler Information Report for Serial Data Communication Interface (SCI)

## 2.1.2 ISO DOCUMENTS

Available from ANSI, 25 west 43rd Street, New York, NY 10036-8002.

ISO 7637-1:1990—Road vehicles—Electrical disturbance by conduction and coupling—Part 1: Passenger cars and light commercial vehicles with nominal 12 V supply voltage

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

ISO 11898:1993—Road vehicles—Interchange of digital information—Controller area network (CAN) for high speed communication

ISO 14230-4:2000—Road vehicles—Diagnostic systems—Keyword protocol 2000—Part 4: Requirements for emission-related systems

ISO/FDIS 15765-2—Road vehicles—Diagnostics on controller area networks (CAN)—Network layer services

ISO/FDIS 15765-4—Road vehicles—Diagnostics on controller area networks (CAN)—Requirements for emission-related systems

## 3. Definitions

### 3.1 Registry

A mechanism within Win32 operating systems to handle hardware and software configuration information.

## 4. Acronyms

|       |                                                    |
|-------|----------------------------------------------------|
| API   | Application Programming Interface                  |
| ASCII | American Standard Code for Information Interchange |
| CAN   | Controller Area Network                            |
| CRC   | Cyclic Redundancy Check                            |
| DLL   | Dynamic Link Library                               |
| ECU   | Electronic Control Unit                            |
| IFR   | In-Frame Response                                  |
| IOCTL | Input / Output Control                             |
| KWP   | Keyword Protocol                                   |
| OEM   | Original Equipment Manufacturer                    |
| PC    | Personal Computer                                  |
| PWM   | Pulse Width Modulation                             |
| SCI   | Serial Communications Interface                    |
| SCP   | Standard Corporate Protocol                        |
| USB   | Universal Serial Bus                               |
| VPW   | Variable Pulse Width                               |

## 5. *Pass-Thru Concept*

Programming application software supplied by the vehicle manufacturer will run on a commonly available generic PC. This application must have complete knowledge of the programming requirements for the control module to be programmed and will control the programming event. This includes the user interface, selection criteria for downloadable software and calibration files, the actual software and calibration data to be downloaded, the security mechanism to control access to the programming capability, and the actual programming steps and sequence required to program each individual control module in the vehicle. If additional procedures must be followed after the reprogramming event, such as clearing Diagnostic Trouble Codes (DTC), writing part numbers or variant coding information to the control module, or running additional setup procedures, the vehicle manufacturer must either include this in the PC application or include the necessary steps in the service information that references reprogramming.

This document defines the following two interfaces for the SAE J2534 pass-thru device:

- a. Application program interface (API) between the programming application running on a PC and a software device driver for the pass-thru device
- b. Hardware interface between the pass-thru device and the vehicle

The manufacturer of an SAE J2534 pass-thru device shall supply connections to both the PC and the vehicle. In addition to the hardware, the interface manufacturer shall supply device driver software, and a Windows installation and setup application that will install the manufacturer's SAE J2534 DLL and other required files, and also update the Windows Registry. The interface between the PC and the pass-thru device can be any technology chosen by the tool manufacturer, including RS-232, RS-485, USB, Ethernet, or any other current or future technology, including wireless technologies.

All programming applications shall utilize the common SAE J2534 API as the interface to the pass-thru device driver. The API contains a set of routines that may be used by the programming application to control the pass-thru device, and to control the communications between the pass-thru device and the vehicle. The pass-thru device will not interpret the message content, allowing any message strategy and message structure to be used that is understood by both the programming application and the ECU being programmed. Also, because the message will not be interpreted, the contents of the message cannot be used to control the operation of the interface. For example, if a message is sent to the ECU to go to high speed, a specific instruction must also be sent to the interface to go to high speed.

The OEM programming application does not need to know the hardware connected to the PC, which gives the tool manufacturers the flexibility to use any commonly available interface to the PC. The pass-thru device does not need any knowledge of the vehicle or control module being programmed. This will allow all programming applications to work with all pass-thru devices to enable programming of all control modules for all vehicle manufacturers.

## SAE J2534-1 Revised DEC2004

Figure 1 shows the relationship between the various components required for pass-thru programming and responsibilities for each component:

| Vehicle                                                                  | Cable                                                                                                            | Pass-thru interface                                                                                                  | Cable                                                                      | Programming PC                                                                                                                                                                       |           |                                                                                                                                                             |
|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                          |                                                                                                                  |                                                                                                                      |                                                                            | Interface driver                                                                                                                                                                     | API       | Application                                                                                                                                                 |
| Vehicle manufacturer determines programming sequence and security access | Tool supplier defines interface with pass-thru device<br><br>SAE J1962 on vehicle end<br><br>5 meter max. length | Capabilities defined in SAE J2534<br><br>Hardware supplied by tool supplier<br><br>Could be implemented in scan tool | Tool supplier defines cable requirements, if any, between PC and interface | Tool supplier device driver, installation, and setup procedure based on hardware supported<br><br>Examples are:<br>RS-232,<br>USB,<br>PCMCIA,<br>Ethernet,<br>IEEE1394,<br>Bluetooth | SAE J2534 | Vehicle manufacturer programming application controls user interface, vehicle software and calibration selection, programming sequence, and security access |
| Vehicle manufacturer                                                     | Tool supplier                                                                                                    |                                                                                                                      |                                                                            |                                                                                                                                                                                      | SAE J2534 | Vehicle manufacturer                                                                                                                                        |

FIGURE 1—SAE J2534 OVERVIEW

### 6. Pass-Thru System Requirements

A "fully" compliant SAE J2534 interface shall support all communication protocols and capabilities defined in this document. The interface's registry entry "Protocols Supported" (see section 9.2-Win32 Registry) specifies the protocols fully supported by the interface.

#### 6.1 PC Requirements

Generic PC running a Win32 Operating System (e.g., Windows 95/Windows 98/Windows NT/Windows Millennium Edition, Windows 2000, Windows XP, ...). The PC should be capable of connection to the Internet.

#### 6.2 Software Requirements and Assumptions

Reprogramming applications can assume that the PC will be connected to the Internet, although not all applications will require this. The OEM application is limited to a single thread for communication with the tool manufacturer DLL/API. Multiple protocols may be connected and used from the single application thread (see Section 6.6).

The interface will not handle the tester present messages automatically. The OEM application is responsible to handle tester present messages.

### 6.3 Connection to PC

The interface between the PC and the pass-thru device shall be determined by the manufacturer of the pass-thru device. This can be RS-232, USB, Ethernet, IEEE1394, Bluetooth or any other connection that allows the pass-thru device to meet all other requirements of this document, including timing requirements. The tool manufacturer is also required to include the device driver that supports this connection so that the actual interface used is transparent to both the PC programming application and the vehicle.

### 6.4 Connection to Vehicle

The interface between the pass-thru device and the vehicle shall be an SAE J1962 connector for serial data communications. The maximum cable length between the pass-thru device and the vehicle is five (5) meters. The interface shall include an insulated banana jack that accepts a standard 0.175" diameter banana plug as the auxiliary pin for connection of programming voltage to a vehicle specific connector on the vehicle.

If powered from the vehicle, the interface shall:

- a. operate normally within a vehicle battery voltage range of 8.0 to 18.0 volts D.C.,
- b. survive a vehicle battery voltage of up to 24.0 volts D.C. for at least 10 minutes,
- c. survive, without damage to the interface, a reverse vehicle battery voltage of up to 24.0 volts D.C. for at least 10 minutes.

### 6.5 Communication Protocols

The following communication protocols shall be supported:

#### 6.5.1 ISO 9141

The following specifications clarify and, if in conflict with ISO 9141, override any related specifications in ISO 9141:

- a. The maximum sink current to be supported by the interface is 100 mA.
- b. The range for all tests performed relative to ISO 7637-1 is -1.0 to +40.0 V.
- c. The default bus idle period before the interface shall transmit an address, shall be 300 ms.
- d. Support following baud rate with  $\pm 0.5\%$  tolerance: 10400.
- e. Support following baud rate with  $\pm 1\%$  tolerance: 10000.
- f. Support following baud rates with  $\pm 2\%$  tolerance: 4800, 9600, 9615, 9800, 10870, 11905, 12500, 13158, 13889, 14706, 15625, and 19200.
- g. Support other baud rates if the interface is capable of supporting the requested value within  $\pm 2\%$ .
- h. The baud rate shall be set by the application, not determined by the SAE J2534 interface. The interface is not required to support baud rate detection based on the synchronization byte.
- i. Support odd and even parity in addition to the default of no parity, with seven or eight data bits. Always one start bit and one stop bit.

- j. Support for timer values that are less than or greater than those specified in ISO 9141 (see Figure 30 in Section 7.3.2).
- k. Support ability to disable automatic ISO 9141-2 / ISO 14230 checksum verification by the interface to allow vehicle manufacturer specific error detection.
- l. If the ISO 9141 checksum is verified by the interface, and the checksum is incorrect, the message will be discarded.
- m. Support both ISO 9141 5-baud initialization and ISO 14230 fast initialization.
- n. Interface shall not adjust timer parameters based on keyword values.

#### 6.5.2 ISO 14230-4 (KWP2000)

The ISO 14230 protocol has the same specifications as the ISO 9141 protocol as outlined in the previous section. In addition, the following specifications clarify and, if in conflict with ISO 14230, override any related specifications in ISO 14230:

- a. The pass-thru interface will not automatically handle tester present messages. The application needs to handle tester present messages when required.
- b. The pass-thru interface will not perform any special handling for the \$78 response code. Any message received with a \$78 response code will be passed from the interface to the application. The application is required to handle any special timing requirements based on receipt of this response code, including stopping any periodic messages.

#### 6.5.3 SAE J1850 41.6 KBPS PWM (PULSE WIDTH MODULATION)

The following additional features of SAE J1850 must be supported by the pass-thru device:

- a. Capable of 41.6 kbps and high speed mode of 83.3 kbps.
- b. Recommend Ford approved SAE J1850PWM (SCP) physical layer

#### 6.5.4 SAE J1850 10.4 KBPS VPW (VARIABLE PULSE WIDTH)

The following additional features of SAE J1850 must be supported by the pass-thru device:

- a. Capable of 10.4 kbps and high speed mode of 41.6 kbps
- b. 4128 byte block transfer
- c. Return to normal speed after a break indication

#### 6.5.5 CAN

The following features of ISO 11898 (CAN) must be supported by the pass-thru device:

- a. 125, 250, and 500 kbps
- b. 11 and 29 bit identifiers
- c. Support for  $80\% \pm 2\%$  and  $68.5\% \pm 2\%$  bit sample point
- d. Allow raw CAN messages. This protocol can be used to handle any custom CAN messaging protocol, including custom flow control mechanisms.

#### 6.5.6 ISO 15765-4 (CAN)

The following features of ISO 15765-4 must be supported by the pass-thru device:

- a. 125, 250, and 500 kbps
- b. 11 and 29 bit identifiers
- c. Support for  $80\% \pm 2\%$  bit sample point
- d. To maintain acceptable programming times, the transport layer flow control function, as defined in ISO 15765-2, must be incorporated in the pass-thru device (see Appendix A). If the application does not use the ISO 15765-2 transport layer flow control functionality, the CAN protocol will allow for any custom transport layer.
- e. Receive a multi-frame message with an ISO15765\_BS of 0 and an ISO15765\_STMIN of 0, as defined in ISO 15765-2.
- f. No single frame or multi-frame messages can be received without matching a flow control filter. No multi-frame messages can be transmitted without matching a flow control filter.
- g. Periodic messages will not be suspended during transmission or reception of a multi-frame segmented message.

#### 6.5.7 SAE J2610 DAIMLERCHRYSLER SCI

Reference the SAE J2610 Information Report for a description of the SCI protocol.

When in the half-duplex mode (when SCI\_MODE of TxFlags is set to {1} Half-Duplex), every data byte sent is expected to be "echoed" by the controller. The next data byte shall not be sent until the echo byte has been received and verified. If the echoed byte received doesn't match the transmitted byte, or if after a period of T1 no response was received, the transmission will be terminated. Matching echoed bytes will not be placed in the receive message queue.

### 6.6 Simultaneous Communication On Multiple Protocols

The pass-thru device must be capable of supporting simultaneous communication on multiple protocols during a single programming event. Figure 2 indicates which combinations of protocols shall be supported. If SCI (SAE J2610) communication is not required during the programming event, the interface shall be capable of supporting one of the protocols from data link set 1, data link set 2, and data link set 3. If SCI (SAE J2610) communication is required during the programming event, the interface shall be capable of supporting one of the SCI protocols and one protocol from data link set 1.

|                    | DATA LINK SET<br>1             | DATA LINK SET<br>2    | DATA LINK SET<br>3 |
|--------------------|--------------------------------|-----------------------|--------------------|
| <b>Without SCI</b> | SAE J1850 VPW<br>SAE J1850 PWM | ISO 9141<br>ISO 14230 | CAN<br>ISO 15765   |
| <b>With SCI</b>    | SAE J1850 VPW<br>SAE J1850 PWM | SCI A<br>SCI B        | None               |

FIGURE 2—SIMULTANEOUS COMMUNICATION OPTIONS

### 6.7 Programmable Power Supply

The interface shall be capable of supplying between 5 and 20 volts to one of the following pins (6, 9, 11, 12, 13 or 14) on the SAE J1962 diagnostic connector, or to an auxiliary pin which would need to be connected to the vehicle via a cable that is unique to the vehicle. The auxiliary pin on the interface shall be a female banana jack (see Section 6.4- Connection to Vehicle). As well, short to ground capability on pin 15 is required. The following requirements shall be met by the power supply:

- a. Minimum 5 V DC
- b. Maximum 20 V DC
- c. Resolution 0.1V DC
- d. Accuracy  $\pm 2\%$  of requested voltage
- e. Maximum source current 150 mA
- f. Maximum sink current 300mA (only for SHORT\_TO\_GROUND on pin 15).
- g. Maximum 1 ms settling time (required for SCI protocol only, reference SAE J2610 Information Report)
- h. Pin assignment software selectable

## 6.8 Pin Usage

Figure 3 indicates the possible uses for each pin of the SAE J1962 connector and for the auxiliary pin. This figure also indicates the default condition for each pin, which is the required condition when the interface is connected to the vehicle, and the condition to return to when the pin is no longer used to supply programming voltage, short to ground, or serial data communication. For the following table, high impedance is defined as greater than 500 k $\Omega$  impedance relative to signal ground, and as greater than 500 k $\Omega$  impedance relative to chassis ground.

| PIN NO. | Possible Uses                                                                            | Default                    |
|---------|------------------------------------------------------------------------------------------|----------------------------|
| Aux     | Auxiliary programming voltage (not part of SAE J1962 connector)                          | High impedance             |
| 1       |                                                                                          | High impedance             |
| 2       | SAE J1850 (+)                                                                            | SAE J1850 (+)              |
| 3       |                                                                                          | High impedance             |
| 4       | Chassis Ground                                                                           | Chassis Ground             |
| 5       | Signal Ground                                                                            | Signal Ground              |
| 6       | ISO 15765-4/CAN High Programming Voltage<br>SCI A Engine (Rx)                            | High impedance             |
| 7       | ISO 9141/ ISO 14230 K-line<br>SCI A engine (Tx)<br>SCI A Trans (Tx)<br>SCI B Engine (Tx) | High impedance             |
| 8       |                                                                                          | High impedance             |
| 9       | SCI B Trans (Rx)<br>Programming Voltage                                                  | High impedance             |
| 10      | SAE J1850 (-)                                                                            | SAE J1850 (-)              |
| 11      | Programming Voltage                                                                      | High impedance             |
| 12      | SCI B engine (Rx)<br>Programming Voltage                                                 | High impedance             |
| 13      | Programming Voltage                                                                      | High impedance             |
| 14      | ISO 15765-4/ CAN Low Programming Voltage<br>SCI A Trans (Tx)                             | High impedance             |
| 15      | ISO 9141/ ISO 14230 L-line<br>Short to Ground<br>SCI B Trans (Tx)                        | High impedance             |
| 16      | Unswitched battery voltage                                                               | Unswitched battery voltage |

FIGURE 3—PIN USAGE

## 6.9 Data Buffering

The interface/API shall be capable of receiving 8 simultaneous messages. For ISO 15765 these can be multi-frame messages. The interface/API shall be capable of buffering a maximum length (4128 byte) transmit message and a maximum length (4128 byte) receive message.

## 6.10 Error Recovery

### 6.10.1 DEVICE NOT CONNECTED

If the DLL returns `ERR_DEVICE_NOT_CONNECTED` from any function, that error shall continue to be returned by all functions, even if the device is reconnected. An application can recover from this error condition by closing the device (with `PassThruClose`) and re-opening the device (with `PassThruOpen`, getting a new device ID).

### 6.10.2 BUS ERRORS

All devices shall handle bus errors in a consistent manner. There are two error strategies: Retry and Drop.

The Retry strategy will keep trying to send a packet until successful or stopped by the application. If loopback is on and the message is successfully sent after some number of retries, only one copy of the message shall be placed in the receive queue. Even if the hardware does not support retries, the firmware/software must retry the transmission. If the error condition persists, a blocking write will wait the specified timeout and return `ERR_TIMEOUT`. The DLL must return the number of successfully transmitted messages in `pNumMsgs`. The DLL shall not count the message being retried in `pNumMsgs`. After returning from the function, the device does not stop the retries. The only functions that will stop the retries are `PassThruDisconnect` (on that protocol), `PassThruClose`, or `PassThruIoctl` (with an `IoctlID` of `CLEAR_TX_BUFFER`).

Devices shall use the Retry strategy in the following scenarios:

- All CAN errors, such as bus off, lack of acknowledgement, loss of arbitration, and no connection (lack of terminating resistor)
- SAE J1850PWM or SAE J1850VPW bus fault (bus stuck passive) or loss of arbitration (bus stuck active)

The Drop strategy will delete a message from the queue. The message can be dropped immediately on noticing an error or at the end of the transmission. `PassThruWriteMsg` shall treat dropped messages the same as successfully transmitted messages. However, if loopback is on, the message shall not be placed in the receive queue.

Devices shall use the Drop strategy in the following scenarios:

- If characters are echoed improperly in SCI
- Corrupted ISO 9141 or ISO 14230 transmission
- SAE J1850PWM lack of acknowledgement (Exception: The device must try sending the message 3 times before dropping)

## 7. Win32 Application Programming Interface

### 7.1 API Functions – Overview

To conform to this document a vendor supplied API implementation (DLL) must support the functions included in Figure 4.

| Function                             | Description                                                                                                                                            |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PassThruOpen</b>                  | Establish a connection with a Pass-Thru device.                                                                                                        |
| <b>PassThruClose</b>                 | Terminate a connection with a Pass-Thru device.                                                                                                        |
| <b>PassThruConnect</b>               | Establish a connection with a protocol channel.                                                                                                        |
| <b>PassThruDisconnect</b>            | Terminate a connection with a protocol channel.                                                                                                        |
| <b>PassThruReadMsgs</b>              | Read message(s) from a protocol channel.                                                                                                               |
| <b>PassThruWriteMsgs</b>             | Write message(s) to a protocol channel.                                                                                                                |
| <b>PassThruStartPeriodicMsg</b>      | Start sending a message at a specified time interval on a protocol channel.                                                                            |
| <b>PassThruStopPeriodicMsg</b>       | Stop a periodic message.                                                                                                                               |
| <b>PassThruStartMsgFilter</b>        | Start filtering incoming messages on a protocol channel.                                                                                               |
| <b>PassThruStopMsgFilter</b>         | Stops filtering incoming messages on a protocol channel.                                                                                               |
| <b>PassThruSetProgrammingVoltage</b> | Set a programming voltage on a specific pin.                                                                                                           |
| <b>PassThruReadVersion</b>           | Reads the version information for the DLL and API.                                                                                                     |
| <b>PassThruGetLastError</b>          | Gets the text description of the last error.                                                                                                           |
| <b>PassThruIoctl</b>                 | General I/O control functions for reading and writing protocol configuration parameters (e.g. initialization, baud rates, programming voltages, etc.). |

FIGURE 4—SAE J2534 API FUNCTIONS

### 7.2 API Functions – Detailed Information

#### 7.2.1 PASSTHRUOPEN

This function is used to establish a connection and initialize the Pass-Thru Device. This function must be called one time before any other function with the exception of PassThruGetLastError. Any function called before a successful call to PassThruOpen must return ERR\_INVALID\_DEVICE\_ID. If the function is successful, a value of STATUS\_NOERROR is returned. The Device ID returned is used as a handle to the initialized SAE J2534 device.

##### 7.2.1.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruOpen
(
    void *pName
    unsigned long *pDeviceID
)
```

### 7.2.1.2 Parameters

pName            Must be NULL (reserved for future use with multiple devices).

pDeviceID        Pointer to location for the device ID that is assigned by the DLL.

### 7.2.1.3 Return Values – See Figure 5

| Definition               | Description                                                    |
|--------------------------|----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful                                       |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with the device                          |
| ERR_DEVICE_IN_USE        | Device is currently open                                       |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required        |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description |

FIGURE 5—RETURN VALUES

### 7.2.2 PASSTHRUCLOSE

This function is used to close the connection to a Pass-Thru Device. All periodic messages will be stopped, filters will be cleared, and all pins will return to their default state (see Section 6.8). This function must be called before an application exits. The DLL can use this function to de-allocate data structures and deactivate any device drivers. If the function is successful, a value of STATUS\_NOERROR is returned. After this call, all active protocols will be disconnected, Channel Ids will no longer be valid, and any function call other than PassThruOpen will result in the error ERR\_INVALID\_DEVICE\_ID being returned (with the exception of PassThruGetLastError).

#### 7.2.2.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruClose
(
    unsigned long DeviceID
)
```

#### 7.2.2.2 Parameters

DeviceID        DeviceID returned from PassThruOpen

## 7.2.2.3 Return Values – See Figure 6

| Definition               | Description                                                    |
|--------------------------|----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful                                       |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device.                             |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                              |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description |

FIGURE 6—RETURN VALUES

## 7.2.3 PASSTHRUCONNECT

This function is used to establish a logical connection with a protocol channel on the specified SAE J2534 device. After this function is called, the value pointed to by pChannelID is used as the logical identifier for the combination of Device ID and Protocol ID. If the function is successful, a value of STATUS\_NOERROR is returned and a valid channel ID will be placed in <pChannelID>. All future interactions with the protocol channel will be done using the pChannelID. Note that the interface will block all received messages on this channel until a filter is set.

## 7.2.3.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruConnect
(
    unsigned long DeviceID,
    unsigned long ProtocolID,
    unsigned long Flags,
    unsigned long BaudRate,
    unsigned long *pChannelID
)
```

## 7.2.3.2 Parameters

DeviceID      Device ID returned from PassThruOpen

ProtocolID    Protocol ID,

Flags        Connection flags,

BaudRate     Initial baud rate

pChannelID    Pointer to location for the channel ID that is assigned by the DLL.

# SAE J2534-1 Revised DEC2004

## 7.2.3.3 Connect Flag Values – See Figure 7

| Definition          | Flags Bit(s) | Description                                                  | Value                                                                                                                                                                                                                                                                                                |
|---------------------|--------------|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | 31-24        | Unused                                                       | Tool manufacturer specific- shall be set to 0                                                                                                                                                                                                                                                        |
|                     | 23-16        | Unused                                                       | Reserved for SAE J2534-2-shall be set to 0                                                                                                                                                                                                                                                           |
|                     | 15-13        | Unused                                                       | Reserved for SAE - shall be set to 0                                                                                                                                                                                                                                                                 |
| ISO9141_K_LINE_ONLY | 12           | L line usage for ISO9141 and ISO14230 Initialization address | 0 = use L-line and K-line for initialization address<br>1 = use K-line only line for initialization address                                                                                                                                                                                          |
| CAN_ID_BOTH         | 11           | CAN ID support type for CAN and ISO 15765 (also see bit 8)   | 0 = either standard or extended CAN ID types used – CAN ID type defined by bit 8<br>1 = both standard and extended CAN ID types used – if the CAN controller allows prioritizing either standard (11 bit) or extended (29 bit) CAN ID's then bit 8 will determine the higher priority ID type        |
|                     | 10           | Unused                                                       | Reserved for SAE - shall be set to 0                                                                                                                                                                                                                                                                 |
| ISO9141_NO_CHECKSUM | 9            | Checksum control for ISO9141 and ISO14230                    | 0 = The interface will generate and append the checksum as defined in ISO 9141-2 and ISO 14230-2 for transmitted messages, and verify the checksum for received messages.<br>1 = The interface will not generate and verify the checksum-the entire message will be treated as data by the interface |
| CAN_29BIT_ID        | 8            | CAN ID type for CAN and ISO 15765 (also see bit 11)          | 0 = Receive standard CAN ID (11 bit)<br>1 = Receive extended CAN ID (29 bit)                                                                                                                                                                                                                         |
|                     | 7            | Unused                                                       | Reserved for SAE- shall be set to 0                                                                                                                                                                                                                                                                  |

FIGURE 7—FLAG VALUES

## SAE J2534-1 Revised DEC2004

### 7.2.3.4 Protocol ID Values – See Figure 8

| Definition   | Description                                                                  | Value(s)             |
|--------------|------------------------------------------------------------------------------|----------------------|
| J1850VPW     | GM / DaimlerChrysler CLASS2                                                  | 0x01                 |
| J1850PWM     | Ford SCP                                                                     | 0x02                 |
| ISO9141      | ISO 9141 and ISO 9141-2                                                      | 0x03                 |
| ISO14230     | ISO 14230-4 (Keyword Protocol 2000)                                          | 0x04                 |
| CAN          | Raw CAN (flow control not handled automatically by interface)                | 0x05                 |
| ISO15765     | ISO 15765-2 flow control enabled (see Appendix A for high level description) | 0x06                 |
| SCI_A_ENGINE | SAE J2610 (DaimlerChrysler SCI) configuration A for engine                   | 0x07                 |
| SCI_A_TRANS  | SAE J2610 (DaimlerChrysler SCI) configuration A for transmission             | 0x08                 |
| SCI_B_ENGINE | SAE J2610 (DaimlerChrysler SCI) configuration B for engine                   | 0x09                 |
| SCI_B_TRANS  | SAE J2610 (DaimlerChrysler SCI) configuration B for transmission             | 0x0A                 |
| Reserved     | Reserved for SAE use                                                         | 0x0B – 0x7FFF        |
| Reserved     | Reserved for SAE J2534-2                                                     | 0x8000 - 0xFFFF      |
| Unused       | Tool manufacturer specific                                                   | 0x10000 – 0xFFFFFFFF |

FIGURE 8—PROTOCOL ID VALUES

## 7.2.3.5 Return Values – See Figure 9

| Definition               | Description                                                                                                                                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                                                                                                                                                                          |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                                                                                                                                                                                  |
| ERR_NOT_SUPPORTED        | Device cannot support a protocol, or a particular (requested) flag on a protocol mandated by this document. Device is not fully SAE J2534 compliant                                                                |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                                                                                                                                                  |
| ERR_INVALID_PROTOCOL_ID  | Invalid ProtocolID value, unsupported ProtocolID, or there is a resource conflict (i.e. trying to connect to multiple protocols that are mutually exclusive such as J1850PWM and J1850VPW or CAN and SCI_A, etc.). |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required                                                                                                                                                            |
| ERR_INVALID_FLAGS        | Invalid flag values.                                                                                                                                                                                               |
| ERR_INVALID_BAUDRATE     | The desired baud rate cannot be achieved within the tolerance specified in Section 6.5                                                                                                                             |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description                                                                                                                                                     |
| ERR_CHANNEL_IN_USE       | Channel number is currently connected.                                                                                                                                                                             |

FIGURE 9—RETURN VALUES

## 7.2.4 PASSTHRUDISCONNECT

This function is used to terminate a logical connection with a protocol channel. If the function is successful, a value of STATUS\_NOERROR is returned. After this call, all filters associated with the channel will be cleared, all periodic messages associated with the channel will be stopped, the associated pins will return to their default state (see Section 6.8) and the Channel ID will no longer be valid.

## 7.2.4.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruDisconnect
(
    unsigned long ChannelID
)
```

## 7.2.4.2 Parameters

ChannelID The channel ID assigned by the PassThruConnect function.

## 7.2.4.3 Return Values – See Figure 10

| Definition               | Description                                                     |
|--------------------------|-----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                       |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device.                              |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                               |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description. |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                        |

FIGURE 10—RETURN VALUES

## 7.2.5 PASSTHRUREADMSG

This function reads messages and indications from the receive buffer. All messages and indications shall be read in the order that they occurred on the bus. If a transmit message generated a loopback message and TxDone indication, the TxDone indication shall always be queued first. Except for loopback messages and indications, no messages shall be queued for reception without matching a PASS\_FILTER (for non-ISO 15765) or FLOW\_CONTROL filter (for ISO 15765). On ISO 15765, PCI bytes are transparently removed by the API. If the function is successful, a value of STATUS\_NOERROR is returned.

Section 8.3 shows the formatting of messages and indications in the PASSTHRU\_MSG structure. Section 8.7.2 shows the valid combinations of the RxStatus bits for the messages and indications to be returned to the application. For each protocol, this function receives the indications from the interface as shown in Figure 11.

| Message/<br>Indication           | Protocol ID |              |                     |                     |     |                    |                       |
|----------------------------------|-------------|--------------|---------------------|---------------------|-----|--------------------|-----------------------|
|                                  | ISO<br>9141 | ISO<br>14230 | SAE<br>J1850<br>PWM | SAE<br>J1850<br>VPW | CAN | ISO<br>15765-<br>4 | SAE<br>J2610<br>(SCI) |
| Normal Message                   | X           | X            | X                   | X                   | X   | X                  | X                     |
| RxStart Indication               | X           | X            |                     |                     |     | X                  |                       |
| RxBreak Indication               |             |              |                     | X                   |     |                    | X                     |
| TxDone Indication                |             |              |                     |                     |     | X                  |                       |
| Loopback Message<br>(if enabled) | X           | X            | X                   | X                   | X   | X                  | X                     |

FIGURE 11—INDICATIONS

### 7.2.5.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruReadMsgs
(
    unsigned long ChannelID,
    PASSTHRU_MSG *pMsg,
    unsigned long *pNumMsgs,
    unsigned long Timeout
)
```

### 7.2.5.2 Parameters

|           |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | The channel ID assigned by the PassThruConnect function.                                                                                                                                                                                                                                                                                                                                                          |
| pMsg      | Pointer to message structure(s).                                                                                                                                                                                                                                                                                                                                                                                  |
| pNumMsgs  | Pointer to location where number of messages to read is specified. On return from the function this location will contain the actual number of messages read.                                                                                                                                                                                                                                                     |
| Timeout   | Read timeout (in milliseconds). If a value of 0 is specified the function retrieves up to pNumMsgs messages and returns immediately. Otherwise, the API will not return until the Timeout has expired, an error has occurred, or the desired number of messages have been read. If the number of messages requested have been read, the function shall not return ERR_TIMEOUT, even if the timeout value is zero. |

## 7.2.5.3 Return Values – See Figure 12

| Definition               | Description                                                                                                                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                                                                                                                                                                               |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                                                                                                                                                                                       |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                                                                                                                                                       |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                                                                                                                                                                                |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required.                                                                                                                                                                |
| ERR_TIMEOUT              | Timeout. Device could not read the specified number of messages. The actual number of messages read is placed in <NumMsgs>. If a timeout occurs and there are no available messages, ERR_BUFFER_EMPTY must be returned. |
| ERR_BUFFER_EMPTY         | Protocol message buffer empty, no messages available to read.                                                                                                                                                           |
| ERR_NO_FLOW_CONTROL      | No flow control filter set or matched (for protocolID ISO15765 only).                                                                                                                                                   |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description                                                                                                                                                          |
| ERR_BUFFER_OVERFLOW      | Indicates a buffer overflow occurred and messages were lost. The actual number of messages read is placed in <NumMsgs>.                                                                                                 |

FIGURE 12—RETURN VALUES

## 7.2.6 PASSTHRUWRITE MSGS

This function is used to send messages. The messages are placed in the buffer and sent in the order they were received. Only one message per protocol can be in transmission at a time (with one exception- See PassThruStartPeriodicMsg). If the function is successful, a value of STATUS\_NOERROR is returned. Specifying a non-zero Timeout performs a blocking write. When using blocking writes, this function does not return until all messages are successfully sent on the vehicle network, or the timeout has expired or an error occurs.

Messages must follow the format specified in Section 8.3. The interface shall not modify structures pointed to by pMsg. Note that some protocols will generate indications when transmitting (See PassThruReadMsg).

When using the ISO 15765-4 protocol, only SingleFrame messages can be transmitted without a matching flow control filter. Also, PCI bytes are transparently added by the API. See PassThruStartMsgFilter and Appendix A for a discussion of flow control filters.

#### 7.2.6.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruWriteMsgs
(
    unsigned long ChannelID,
    PASSTHRU_MSG *pMsg,
    unsigned long *pNumMsgs,
    unsigned long Timeout
)
```

#### 7.2.6.2 Parameters

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | The channel ID assigned by the PassThruConnect function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| pMsg      | Pointer to message structure(s).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| pNumMsgs  | Pointer to the location where number of messages to write is specified. On return will contain the actual number of messages that were transmitted (when Timeout is non-zero) or placed in the transmit queue (when Timeout is zero).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Timeout   | Write timeout (in milliseconds). When a value of 0 is specified, the function queues as many of the specified messages as possible and returns immediately. When a value greater than 0 is specified, the function will block until the Timeout has expired, an error has occurred, or the desired number of messages have been transmitted on the vehicle network. Even if the device can buffer only one packet at a time, this function shall be able to send an arbitrary number of packets if a Timeout value is supplied. Since the function returns early if all the messages have been sent, there is normally no penalty for having a large timeout (several seconds). If the number of messages requested have been written, the function shall not return ERR_TIMEOUT, even if the timeout value is zero.<br><br>When an ERR_TIMEOUT is returned, only the number of messages that were sent on the vehicle network is known. The number of messages queued is unknown. Application writers should avoid this ambiguity by using a Timeout value large enough to work on slow devices and networks with arbitration delays. |

## 7.2.6.3 Return Values – See Figure 13

| Definition               | Description                                                                                                                                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                                                                                                                                                                          |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                                                                                                                                                                                  |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                                                                                                                                                  |
| ERR_NOT_SUPPORTED        | Device cannot support a particular (requested) flag on a protocol mandated by this document. Device is not fully SAE J2534 compliant.<br><br>Example: Requesting SCI TX VOLTAGE on a device without the capability |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                                                                                                                                                                           |
| ERR_INVALID_MSG          | Invalid message structure pointed to by pMsg (Reference Section 8 Message Structure).                                                                                                                              |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required.                                                                                                                                                           |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description                                                                                                                                                     |
| ERR_TIMEOUT              | Timeout. Device could not write the specified number of messages. The actual number of messages sent on the vehicle network is placed <NumMsgs>. Only applies when Timeout is non-zero.                            |
| ERR_MSG_PROTOCOL_ID      | Protocol type in the message does not match the protocol associated with the ChannelID                                                                                                                             |
| ERR_NO_FLOW_CONTROL      | Multi-segment transmission without matching flow control filter set (for protocolID ISO15765 only).                                                                                                                |
| ERR_BUFFER_FULL          | Protocol message buffer is full. Some messages could not be queued. Only applies when Timeout is zero.                                                                                                             |

FIGURE 13—RETURN VALUES

## 7.2.7 PASSTHRUSTARTPERIODICMSG

This function will immediately queue the specified message for transmission, and repeat at the specified interval. Periodic messages are limited in length to a single frame message of 12 bytes or less, including header or CAN ID. Periodic messages shall have priority over messages queued with PassThruWriteMsgs, but periodic messages must not violate bus idle timing parameters (e.g. P3\_MIN). Periodic messages shall generate TxDone indications (ISO 15765) and loopback messages (on any protocol, if enabled). On ISO 15765, periodic messages can be sent during a multi-frame transmission or reception. If the function is successful, a value of STATUS\_NOERROR is returned. The Pass-Thru device must support a minimum of ten periodic messages.

PassThruDisconnect shall delete all periodic messages on that channel. PassThruClose shall delete all periodic messages on all channels for the device. All periodic messages will be stopped on a PassThruDisconnect for the associated protocol or a PassThruClose for the device.

## 7.2.7.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruStartPeriodicMsg
(
    unsigned long ChannelID,
    PASSTHRU_MSG *pMsg,
    unsigned long *pMsgID,
    unsigned long TimeInterval
)
```

## 7.2.7.2 Parameters

|              |                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID    | The channel ID assigned by the PassThruConnect function.                                                                               |
| pMsg         | Pointer to message structure.                                                                                                          |
| pMsgID       | Pointer to location for the message ID that is assigned by the DLL.                                                                    |
| TimeInterval | Time interval between the start of successive transmissions of this message, in milliseconds. The valid range is 5-65535 milliseconds. |

## 7.2.7.3 Return Values – See Figure 14

| Definition                | Description                                                                                                                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR            | Function call successful.                                                                                                                                                                             |
| ERR_DEVICE_NOT_CONNECTED  | Unable to communicate with device                                                                                                                                                                     |
| ERR_INVALID_DEVICE_ID     | Device ID invalid                                                                                                                                                                                     |
| ERR_NOT_SUPPORTED         | Device cannot support requested functionality mandated by this document. Device is not fully SAE J2534 compliant.<br><br>Example: Requesting 20ms interval, but hardware only supports 50ms intervals |
| ERR_INVALID_CHANNEL_ID    | Invalid ChannelID value.                                                                                                                                                                              |
| ERR_INVALID_MSG           | Invalid message structure pointed to by pMsg. (Reference Section 8 Message Structure)                                                                                                                 |
| ERR_NULL_PARAMETER        | NULL pointer supplied where a valid pointer is required.                                                                                                                                              |
| ERR_INVALID_TIME_INTERVAL | Invalid TimeInterval value.                                                                                                                                                                           |
| ERR_FAILED                | Undefined error, use PassThruGetLastError for text description                                                                                                                                        |
| ERR_MSG_PROTOCOL_ID       | Protocol type in the message does not match the protocol associated with the ChannelID                                                                                                                |
| ERR_EXCEEDED_LIMIT        | Exceeded the maximum number of periodic message IDs or the maximum allocated space.                                                                                                                   |

FIGURE 14—RETURN VALUES

## 7.2.8 PASSTHRUSTOPPERIODICMSG

This function stops the specified periodic message. If the function is successful, a value of STATUS\_NOERROR is returned. After this call the MsgID will be invalid.

Note that periodic messages that have been queued for transmission or that are in the process of being transmitted might not be stopped by this function.

## 7.2.8.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruStopPeriodicMsg
(
    unsigned long ChannelID,
    unsigned long MsgID
)
```

## 7.2.8.2 Parameters

**ChannelID**      The channel ID assigned by the PassThruConnect function.

**MsgID**            Message ID that is assigned by the PassThruStartPeriodicMsg function.

## 7.2.8.3 Return Values – See Figure 15

| Definition               | Description                                                     |
|--------------------------|-----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                       |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device.                              |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                               |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                        |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description. |
| ERR_INVALID_MSG_ID       | Invalid MsgID value.                                            |

FIGURE 15—RETURN VALUES

## 7.2.9 PASSTHRUSTARTMSGFILTER

This function starts filtering of incoming messages. If the function is successful, a value of STATUS\_NOERROR is returned. A minimum of ten message filters shall be supported by the interface for each supported protocol. PassThruDisconnect shall delete all message filters on that channel. PassThruClose shall delete all filters on all channels for the device. Pattern and Mask messages shall follow the protocol formats specified in Section 8. However, only the first twelve (12) bytes, including header or CAN ID, are used by the filter. ERR\_INVALID\_MSG shall be returned if the filter length exceeds 12. Note that this function does not clear any messages that may have been received and queued before the filter was set. Users are cautioned to consider performing a CLEAR\_RX\_BUFFER after starting a message filter to be sure that unwanted frames are purged from any receive buffers.

For all protocols except ISO 15765:

- PASS\_FILTERs and BLOCK\_FILTERs will be applied to all received messages. They shall not be applied to indications or loopback messages
- FLOW\_CONTROL\_FILTERs must not be used and shall cause the interface to return ERR\_INVALID\_FILTER\_ID
- Both pMaskMsg and pPatternMsg must have the same DataSize and TxFlags. Otherwise, the interface shall return ERR\_INVALID\_MSG
- The default filter behavior after PassThruConnect is to block all messages, which means no messages will be placed in the receive queue until a PASS\_FILTER has been set. Messages that match a PASS\_FILTER can still be blocked by a BLOCK\_FILTER
- Figure 16 and Figure 17 show how the message filtering mechanism operates

| PASS Filter | BLOCK Filter | Message Matches PASS Filter | Message Matches BLOCK Filter | Action |
|-------------|--------------|-----------------------------|------------------------------|--------|
| No          | No           | N/A                         | N/A                          | Block  |
| Yes         | No           | No                          | N/A                          | Block  |
| Yes         | No           | Yes                         | N/A                          | Pass   |
| No          | Yes          | N/A                         | No                           | Block  |
| No          | Yes          | N/A                         | Yes                          | Block  |
| Yes         | Yes          | No                          | No                           | Block  |
| Yes         | Yes          | No                          | Yes                          | Block  |
| Yes         | Yes          | Yes                         | No                           | Pass   |
| Yes         | Yes          | Yes                         | Yes                          | Block  |

FIGURE 16—MESSAGE FILTER ACTIONS

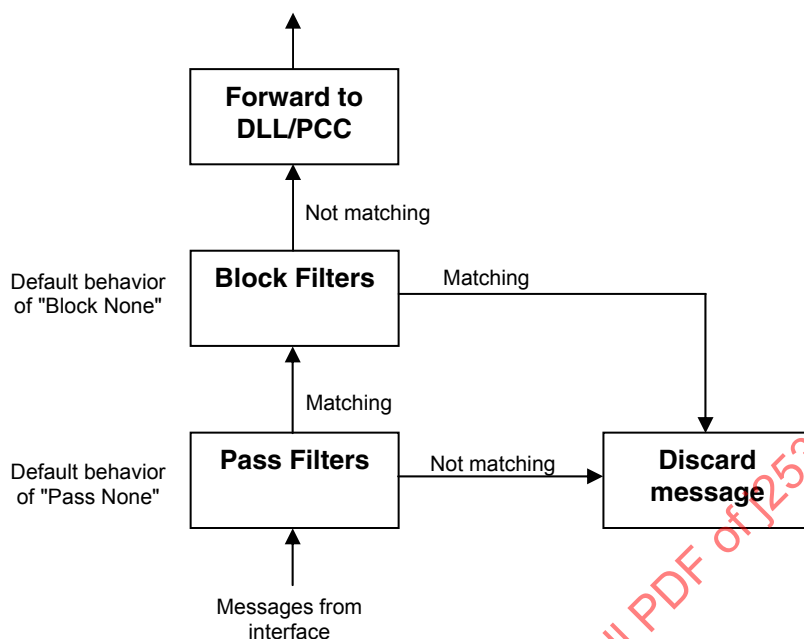


FIGURE 17—MESSAGE FILTER FLOW

For ISO 15765:

- PASS\_FILTERs and BLOCK\_FILTERs must not be used and shall cause the interface to return ERR\_INVALID\_FILTER\_ID
- Filters shall not be applied to indications or loopback messages. When loopback is on, the original message shall be copied to the receive queue upon the last segment being transmitted on the bus
- Non-segmented messages do not need to match a FLOW\_CONTROL\_FILTER
- No segmented messages can be transmitted without matching an appropriate FLOW\_CONTROL\_FILTER. An appropriate filter is one in which the pFlowControlMsg CAN ID matches the messages to be transmitted. Also, the ISO 15765\_ADDR\_TYPE (reference TxFlags in Section 8.7.3) bits must match. If that bit is set, the first byte after the CAN IDs (the extended address) must match too
- No message (segmented or unsegmented) shall be received without matching an appropriate FLOW\_CONTROL\_FILTER. An appropriate filter is one in which the pPatternMsg CAN ID matches the incoming message ID. If the ISO 15765\_ADDR\_TYPE (reference TxFlags in Section 8.7.3) bit is set in the filter, the first byte after the CAN IDs (the extended address) must match too
- All 3 message pointers must have the same DataSize and TxFlags. Otherwise, the interface shall return ERR\_INVALID\_MSG
- Both the pFlowControlMsg ID and the pPatternMsg ID must be unique (not match any IDs in any other filters). The only exception is that pPatternMsg can equal pFlowControlMsg to allow for receiving functionally addressed messages. In this case, only non-segmented messages can be received
- See Appendix A for a detailed description of flow control filter usage.

## 7.2.9.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruStartMsgFilter
(
    unsigned long ChannelID,
    unsigned long FilterType,
    PASSTHRU_MSG *pMaskMsg,
    PASSTHRU_MSG *pPatternMsg,
    PASSTHRU_MSG *pFlowControlMsg,
    unsigned long *pFilterID
)
```

## 7.2.9.2 Parameters

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID  | The channel ID assigned by the PassThruConnect function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| FilterType | Designates:<br><br>PASS_FILTER – allows matching messages into the receive queue. This filter type is only valid on non-ISO 15765 channels<br><br>BLOCK_FILTER – keeps matching messages out of the receive queue. This filter type is only valid on non-ISO 15765 channels<br><br>FLOW_CONTROL_FILTER – allows matching messages into the receive queue and defines an outgoing flow control message to support the ISO 15765-2 flow control mechanism. This filter type is only valid on ISO 15765 channels.                                                                                                                                                                                                                                                                                                                                                                                                             |
| pMaskMsg   | For a PASS_FILTER or BLOCK_FILTER:<br><br>This designates a pointer to the mask message that will be applied to each incoming message (i.e., the mask message that will be ANDed to each incoming message) to mask any unimportant bits.<br><br>When using the CAN protocol, setting the first 4 bytes of pMaskMsg to \$FF makes the filter specific to one CAN ID. Using other values allows for the reception or blocking of multiple CAN identifiers.<br><br>For a FLOW_CONTROL_FILTER:<br><br>The mask shall consist of 4 or 5 bytes of \$FF, with a corresponding DataSize. Five bytes are only allowed when the extended address bit in TxFlags is set. Flow control filters are point-to-point, and shall not be allowed to match multiple CAN identifiers. The only exception is to allow for masking the priority field in a 29-bit CAN ID as specified in ISO 15765-2 Annex A. In this case Data[0] can be \$E3. |

pPatternMsg For a PASS\_FILTER or BLOCK\_FILTER:

Designates a pointer to 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 and the FilterType is PASS\_FILTER, then the incoming message will be added to the receive queue (otherwise it will be discarded). If the result matches this pattern message and the FilterType is BLOCK\_FILTER, then the incoming message will be discarded (otherwise it will be added to the receive queue). Message bytes in the received message that are beyond the DataSize of the pattern message will be treated as “don’t care”.

For a FLOW\_CONTROL\_FILTER:

Designates a pointer to the CAN ID (with optional extended address) at the other end of an ISO 15765-2 conversation. Any messages on the bus not matching a pPatternMsg must be discarded.

pFlowControlMsg This pointer must be null when requesting a PASS\_FILTER or a BLOCK\_FILTER, otherwise ERR\_INVALID\_MSG shall be returned.

For a FLOW\_CONTROL\_FILTER:

Designates a pointer to the CAN ID used when sending CAN frames during an ISO 15765-2 segmented transmission or reception. This is the CAN ID to match against the CAN ID in a segmented PassThruWriteMsg. This message shall only contain the CAN ID (and extended address byte if the ISO15765\_EXT\_ADDR flag is set).

pFilterID Pointer to location for the filter ID that is assigned by the DLL.

### 7.2.9.3 Filter Type Values – See Figure 18

| Definition                 | Value                 |
|----------------------------|-----------------------|
| PASS_FILTER                | 0x00000001            |
| BLOCK_FILTER               | 0x00000002            |
| FLOW_CONTROL_FILTER        | 0x00000003            |
| Reserved                   | 0x00000004-0x00007FFF |
| Reserved for SAE J2534-2   | 0x00008000-0x0000FFFF |
| Tool manufacturer specific | 0x00010000-0xFFFFFFFF |

FIGURE 18—FILTER TYPE VALUES

## 7.2.9.4 Return Values – See Figure 19

| Definition               | Description                                                                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                                                                                                                |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device.                                                                                                                       |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                                                                                        |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                                                                                                                 |
| ERR_INVALID_MSG          | Invalid message structure pointed to by message pointers or messages do not share common TxFlags and DataSize. (Reference Section 8 - Message Structure) |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required.                                                                                                 |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description.                                                                                          |
| ERR_NOT_UNIQUE           | A CAN ID in pPatternMsg or pFlowControlMsg matches either ID in an existing FLOW_CONTROL_FILTER.                                                         |
| ERR_EXCEEDED_LIMIT       | Exceeded the maximum number of filter message IDs or the maximum allocated space.                                                                        |
| ERR_MSG_PROTOCOL_ID      | Protocol type in the message does not match the protocol associated with the ChannelID.                                                                  |

FIGURE 19—RETURN VALUES

## 7.2.10 PASSTHRUSTOPMSGFILTER

This function removes the specified filter. If the function is successful, a value of STATUS\_NOERROR is returned. After this call the FilterID will be invalid.

## 7.2.10.1 C/C++ Prototype

```
extern "C" long WINAPI PassThruStopMsgFilter
(
    unsigned long ChannelID,
    unsigned long FilterID
)
```

### 7.2.10.2 Parameters

**ChannelID**      The channel ID assigned by the PassThruConnect function.

**FilterID**        Filter ID that is assigned by the PassThruStartMsgFilter function.

### 7.2.10.3 Return Values – See Figure 20

| Definition               | Description                                                    |
|--------------------------|----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                      |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                              |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                              |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                       |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description |
| ERR_INVALID_FILTER_ID    | Invalid FilterID value.                                        |

FIGURE 20—RETURN VALUES

### 7.2.11 PASSTHRUSETPROGRAMMINGVOLTAGE

This function sets a single programming voltage on a single specific pin. The programming voltage pins are mutually exclusive, and at any given time programming voltage can only be applied to a single pin. This does not apply to pin 15 (short to ground), which can be shorted to ground simultaneously with programming voltage on another pin. The programming voltage must be turned off before the programming voltage can be applied to a different pin. The default state of the pins shall be VOLTAGE\_OFF.

If the function is successful, a value of STATUS\_NOERROR is returned. It is up to the application programmer to insure that voltages are not applied to any pins incorrectly. This function cannot protect from incorrect usage (e.g., applying a voltage to pin 6 when it is being used for the CAN protocol). Note that for SCI protocol, the application would set the PinNumber, set the Voltage to VOLTAGE\_OFF, and set SCI\_TX\_VOLTAGE in TxFlags of the message to pulse the programming voltage to 20 V DC.

This function can be called only after calling PassThruOpen.

#### 7.2.11.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruSetProgrammingVoltage
(
    unsigned long DeviceID,
    unsigned long PinNumber,
    unsigned long Voltage
)
```

## 7.2.11.2 Parameters

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeviceID  | Device ID returned from PassThruOpen                                                                                                                                                                                                                                                                                                                                                                                                                                |
| PinNumber | The pin on which the programming voltage will be set. Valid options are:<br>0 – Auxiliary output pin (for non-SAE J1962 connectors)<br>6 – Pin 6 on the SAE J1962 connector.<br>9 – Pin 9 on the SAE J1962 connector.<br>11 – Pin 11 on the SAE J1962 connector.<br>12 – Pin 12 on the SAE J1962 connector.<br>13 – Pin 13 on the SAE J1962 connector.<br>14 – Pin 14 on the SAE J1962 connector.<br>15 – Pin 15 on the SAE J1962 connector (short to ground only). |
| Voltage   | The voltage (in millivolts) to be set. Valid values are:<br>5000mV-20000mV (limited to 150mA with a resolution of 100 millivolts for pins 0, 6, 9, 11, 12, 13, and 14).<br>VOLTAGE_OFF – To turn output off (disconnect-high impedance $\geq 500K\Omega$ ).<br>SHORT_TO_GROUND – Short pin to ground (limited to 300mA on pin 15 only).                                                                                                                             |

## 7.2.11.3 Voltage Values – See Figure 21

| Definition          | Value                                         |
|---------------------|-----------------------------------------------|
| Programming Voltage | 0x00001388 (5000 mV) to 0x00004E20 (20000 mV) |
| SHORT_TO_GROUND     | 0xFFFFFFFF                                    |
| VOLTAGE_OFF         | 0xFFFFFFFF                                    |

FIGURE 21—VOLTAGE VALUES

## 7.2.11.4 Return Values – See Figure 22

| Definition               | Description                                                                                   |
|--------------------------|-----------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful.                                                                     |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device.                                                            |
| ERR_NOT_SUPPORTED        | Function not supported.                                                                       |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                             |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description.                               |
| ERR_PIN_INVALID          | Invalid pin number, pin number already in use, or voltage already applied to a different pin. |

FIGURE 22—RETURN VALUES

## 7.2.12 PASSTHRUREADVERSION

This function returns the version strings associated with the DLL. If the function is successful, a value of STATUS\_NOERROR is returned. A buffer of at least eighty (80) characters must be allocated for each pointer by the application.

This function can be called only after calling PassThruOpen.

## 7.2.12.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruReadVersion
(
    unsigned long DeviceID
    char*pFirmwareVersion,
    char*pDllVersion,
    char*pApiVersion
)
```

## 7.2.12.2 Parameters

|                  |                                                                                                                                                                                                                                                        |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeviceID         | Device ID returned from PassThruOpen                                                                                                                                                                                                                   |
| pFirmwareVersion | Pointer to Firmware version string. This string is determined by the interface vendor that supplies the device.                                                                                                                                        |
| pDllVersion      | Pointer to DLL version string. This string is determined by the interface vendor that supplies the DLL.                                                                                                                                                |
| pApiVersion      | Pointer to API version string in YY. MM format. This string corresponds to the date of the approved document (may not be equivalent to SAE publication date).<br><br>February 2002 Final = "02.02"<br><br>November 2004 Final (this version) = "04.04" |

## 7.2.12.3 Return Values – See Figure 23

| Definition               | Description                                                    |
|--------------------------|----------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful                                       |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                              |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                              |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required        |

FIGURE 23—RETURN VALUES

## 7.2.13 PASSTHRUGETLASTERROR

This function returns the text string description for an error detected during the last function call (except PassThruGetLastError). The error string must be retrieved before calling any other function. The buffer pointed to by pErrorDescription is allocated by the application and must be at least eighty (80) characters.

This function can be called without first calling PassThruConnect or PassThruOpen. The last error returned is not specific to any particular Channel ID or Device ID and is related to the last function call. It would be expected that the application would call this function immediately after a function fails. This function is mainly for application developers.

## 7.2.13.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruGetLastError
(
    char *pErrorDescription
)
```

## 7.2.13.2 Parameters

pErrorDescription Pointer to error description string.

## 7.2.13.3 Return Values – See Figure 24

| Definition         | Description                                             |
|--------------------|---------------------------------------------------------|
| STATUS_NOERROR     | Function call successful                                |
| ERR_NULL_PARAMETER | NULL pointer supplied where a valid pointer is required |

FIGURE 24—RETURN VALUES

## 7.2.14 PASSTHRUOCTL

This function is used to read and write all the protocol hardware and software configuration parameters. If the function is successful, a value of STATUS\_NOERROR is returned. The structures pointed to by pInput and pOutput are determined by the ioctlID. See section on IOCTL structures for details.

PassThruOpen must be called prior to any ioctl call. Some ioctl functions do not require a ChannelID and therefore do not require that PassThruConnect be called prior to ioctl. In this case, the DeviceID must be passed instead.

## 7.2.14.1 C / C++ Prototype

```
extern "C" long WINAPI PassThruIoctl
(
    unsigned long ChannelID,
    unsigned long ioctlID,
    void *pInput,
    void *pOutput
)
```

## 7.2.14.2 Parameters

|           |                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| ChannelID | The channel ID assigned by the PassThruConnect function, except in designated ioctls where the device ID is passed instead. |
| ioctlID   | ioctl ID (see the IOCTL Section).                                                                                           |
| pInput    | Pointer to input structure (see the IOCTL Section).                                                                         |
| pOutput   | Pointer to output structure (see the IOCTL Section).                                                                        |

7.2.14.3 *Ioctl ID Values – See Figure 25*

| Definition                         | Value              |
|------------------------------------|--------------------|
| GET_CONFIG                         | 0x01               |
| SET_CONFIG                         | 0x02               |
| READ_VBATT                         | 0x03               |
| FIVE_BAUD_INIT                     | 0x04               |
| FAST_INIT                          | 0x05               |
| CLEAR_TX_BUFFER                    | 0x07               |
| CLEAR_RX_BUFFER                    | 0x08               |
| CLEAR_PERIODIC_MSGS                | 0x09               |
| CLEAR_MSG_FILTERS                  | 0x0A               |
| CLEAR_FUNCT_MSG_LOOKUP_TABLE       | 0x0B               |
| ADD_TO_FUNCT_MSG_LOOKUP_TABLE      | 0x0C               |
| DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE | 0x0D               |
| READ_PROG_VOLTAGE                  | 0x0E               |
| Reserved for SAE                   | 0x0F – 0x7FFF      |
| Reserved for SAE J2534-2           | 0x8000 – 0xFFFF    |
| Tool manufacturer specific         | 0x10000 – 0FFFFFFF |

FIGURE 25—I\_IOCTL ID VALUES

7.2.14.4 *Return Values – See Figure 26*

| Definition               | Description                                                                                                          |
|--------------------------|----------------------------------------------------------------------------------------------------------------------|
| STATUS_NOERROR           | Function call successful                                                                                             |
| ERR_DEVICE_NOT_CONNECTED | Unable to communicate with device                                                                                    |
| ERR_INVALID_CHANNEL_ID   | Invalid ChannelID value.                                                                                             |
| ERR_INVALID_IOCTL_ID     | Invalid ioctlID value.                                                                                               |
| ERR_NULL_PARAMETER       | NULL pointer supplied where a valid pointer is required                                                              |
| ERR_NOT_SUPPORTED        | Invalid or unsupported parameter/value                                                                               |
| ERR_FAILED               | Undefined error, use PassThruGetLastError for text description                                                       |
| ERR_INVALID_MSG          | Invalid message structure pointed to by plnput when using FAST_INT_ioctl.<br>(Reference Section 8 Message Structure) |
| ERR_INVALID_DEVICE_ID    | Device ID invalid                                                                                                    |
| ERR_INVALID_IOCTL_VALUE  | Invalid value for ioctl parameter                                                                                    |

FIGURE 26—RETURN VALUES

### 7.3 IOCTL Section

Figure 27 provides the details on the IOCTLs available through PassThruIoctl function:

| Value of ioctlID                  | InputPtr represents             | OutputPtr represents            | Purpose                                                                                                                       |
|-----------------------------------|---------------------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| GET_CONFIG                        | Pointer to SCONFIG_LIST         | NULL pointer                    | To get the vehicle network configuration of the pass-thru device                                                              |
| SET_CONFIG                        | Pointer to SCONFIG_LIST         | NULL pointer                    | To set the vehicle network configuration of the pass-thru device                                                              |
| READ_VBATT                        | NULL pointer                    | Pointer to unsigned long        | To direct the pass-thru device to read the voltage on pin 16 of the J1962 connector                                           |
| FIVE_BAUD_INIT                    | Pointer to SBYTE_ARRAY          | Pointer to SBYTE_ARRAY          | To direct the pass-thru device to initiate a 5 baud initialization sequence                                                   |
| FAST_INIT                         | NULL or Pointer to PASSTHRU_MSG | NULL or Pointer to PASSTHRU_MSG | To direct the pass-thru device to initiate a fast initialization sequence                                                     |
| CLEAR_TX_BUFFER                   | NULL pointer                    | NULL pointer                    | To direct the pass-thru device to clear all messages in its transmit queue                                                    |
| CLEAR_RX_BUFFER                   | NULL pointer                    | NULL pointer                    | To direct the pass-thru device to clear all messages in its receive queue                                                     |
| CLEAR_PERIODIC_MSGS               | NULL pointer                    | NULL pointer                    | To direct the pass-thru device to clear all periodic messages on the channel, thus stopping all periodic message transmission |
| CLEAR_MSG_FILTERS                 | NULL pointer                    | NULL pointer                    | To direct the pass-thru device to clear all message filters on the channel                                                    |
| CLEAR_FUNC_MSG_LOOKUP_TABLE       | NULL pointer                    | NULL pointer                    | To direct the pass-thru device to clear the Functional Message Look-up Table                                                  |
| ADD_TO_FUNC_MSG_LOOKUP_TABLE      | Pointer to SBYTE_ARRAY          | NULL pointer                    | To direct the pass-thru device to add a functional address to the Functional Message Look-up Table                            |
| DELETE_FROM_FUNC_MSG_LOOKUP_TABLE | Pointer to SBYTE_ARRAY          | NULL pointer                    | To direct the pass-thru device to delete a functional address from the Functional Message Look-up Table                       |
| READ_PROG_VOLTAGE                 | NULL pointer                    | Pointer to unsigned long        | To direct the pass-thru device to read the feedback of the programmable voltage set by PassThruSetProgrammingVoltage          |

FIGURE 27—IOCTL DETAILS

## 7.3.1 GET\_CONFIG

The locID value of GET\_CONFIG is used to obtain the vehicle network configuration of the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 28. When the function is successfully completed, the corresponding parameter value(s) indicated in Figure 30 will be placed in each Value.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | Channel ID assigned by DLL during PassThruConnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| locID     | Is set to the define GET_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:<br/> NumOfParams is an INPUT, which contains the number of SCONFIG elements in the array pointed to by ConfigPtr.<br/> 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:<br/> Parameter is an INPUT that represents the parameter to be obtained (See Figure 30 for a list of valid parameters).<br/> Value is an OUTPUT that represents the value of that parameter (See Figure 30 for a list of valid values).</p> |
| OutputPtr | Is a NULL pointer, as this parameter is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

FIGURE 28—GET\_CONFIG DETAIL

## 7.3.2 SET\_CONFIG

The locID value of SET\_CONFIG is used to set the vehicle network configuration of the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 29. When the function is successfully completed the corresponding parameter(s) and value(s) indicated in Figure 30 will be in effect.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | Channel ID assigned by DLL during PassThruConnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| locID     | Is set to the define SET_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:<br/> NumOfParams is an INPUT, which contains the number of SCONFIG elements in the array pointed to by ConfigPtr.<br/> 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:<br/> Parameter is an INPUT that represents the parameter to be set (See Figure 30 for a list of valid parameters).<br/> Value is an INPUT that represents the value of that parameter (See Figure 30 for a list of valid values).</p> |
| OutputPtr | Is a NULL pointer, as this parameter is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

FIGURE 29—SET\_CONFIG DETAILS

SAE J2534-1 Revised DEC2004

| Parameter                         | ID Value | Valid values for Parameter                      | Default Value (decimal) | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------|----------|-------------------------------------------------|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DATA_RATE                         | 0x01     | 5-500000                                        | protocol specific       | Represents the desired baud rate.<br><br>An ERR_INVALID_IOCTL_VALUE will be returned if the desired baud rate cannot be achieved within the tolerance specified in Section 6.5. The interface will remain at the previous baud rate.                                                                                                                                                                                                                       |
| Unused                            | 0x02     |                                                 |                         | Reserved for SAE                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| LOOPBACK                          | 0x03     | 0 (OFF)<br>1 (ON)                               | 0                       | 0 = Don't echo transmitted messages in the receive queue.<br>1 = Echo transmitted messages, including periodic messages, in the receive queue. Loopback messages must only be sent after successful transmission of a message. Loopback frames are not subject to message filtering.                                                                                                                                                                       |
| NODE_ADDRESS                      | 0x04     | 0x00-0xFF                                       | N/A                     | For a protocol ID of J1850PWM, this sets the node address in the physical layer of the vehicle network.                                                                                                                                                                                                                                                                                                                                                    |
| NETWORK_LINE                      | 0x05     | 0 (BUS_NORMAL)<br>1 (BUS_PLUS)<br>2 (BUS_MINUS) | 0                       | For a protocol ID of J1850PWM, this sets the network line(s) that are active during communication (for cases where the physical layer allows this).                                                                                                                                                                                                                                                                                                        |
| P1_MIN<br>(not used by interface) | 0x06     | N/A                                             | N/A                     | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum inter-byte time for ECU responses. Application shall not get or set this value. The interface will not check for P1_MIN violations. Interface must be capable of handling P1_MIN = 0.                                                                                                                                                                                                      |
| P1_MAX                            | 0x07     | 0x1-0xFFFF<br>(.5 ms per bit)                   | 40<br>(20 ms)           | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum inter-byte time for ECU responses.                                                                                                                                                                                                                                                                                                                                                         |
| P2_MIN<br>(not used by interface) | 0x08     | N/A                                             | N/A                     | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum time between tester request and ECU responses or two ECU responses. Application shall not get or set this value. The interface will not check for P2_MIN violations. After the request, the interface shall be capable of handling an immediate response (P2_MIN = 0). For subsequent responses, a byte received after P1_MAX shall be considered as the start of the subsequent response. |
| P2_MAX<br>(not used by interface) | 0x09     | N/A                                             | N/A                     | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum time between tester request and ECU responses or two ECU responses. Application shall not get or set this value. The interface will not check for P2_MAX violations. The interface will accept all responses up to P3_MIN.                                                                                                                                                                 |

SAE J2534-1 Revised DEC2004

| Parameter                         | ID Value | Valid values for Parameter                         | Default Value (decimal) | Description                                                                                                                                                                                                                                      |
|-----------------------------------|----------|----------------------------------------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| P3_MIN                            | 0x0A     | 0x0-0xFFFF<br>(.5 ms per bit)                      | 110<br>(55 ms)          | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum time between end of ECU response and start of new tester request.                                                                                                                |
| P3_MAX<br>(not used by interface) | 0x0B     | N/A                                                | N/A                     | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum time between end of ECU response and start of new tester request. Application shall not get or set this value. Interface allows transmission of a request any time after P3_MIN. |
| P4_MIN                            | 0x0C     | 0x0-0xFFFF<br>(.5 ms per bit)                      | 10<br>(5 ms)            | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum inter-byte time for a tester request.                                                                                                                                            |
| P4_MAX<br>(not used by interface) | 0x0D     | N/A                                                | N/A                     | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum inter-byte time for a tester request. Application shall not get or set this value. The interface shall transmit at P4_MIN.                                                       |
| W0                                | 0x19     | 0x0-0xFFFF<br>(1 ms per bit)                       | 300                     | For protocol ID of ISO 9141, this sets the minimum bus idle time before the tester starts to transmit the address byte.                                                                                                                          |
| W1                                | 0x0E     | 0x0-0xFFFF<br>(1 ms per bit)                       | 300                     | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum time from the end of the address byte to the start of the synchronization pattern.                                                                                               |
| W2                                | 0x0F     | 0x0-0xFFFF<br>(1 ms per bit)                       | 20                      | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum time from the end of the synchronization pattern to the start of key byte 1.                                                                                                     |
| W3                                | 0x10     | 0x0-0xFFFF<br>(1 ms per bit)                       | 20                      | For protocol ID of ISO 9141 or ISO 14230, this sets the maximum time between key byte 1 and key byte 2.                                                                                                                                          |
| W4                                | 0x11     | 0x0-0xFFFF<br>(1 ms per bit)                       | 50                      | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum time between key byte 2 and its inversion from the tester.                                                                                                                       |
| W5                                | 0x12     | 0x0-0xFFFF<br>(1 ms per bit)                       | 300                     | For protocol ID of ISO 14230, this sets the minimum bus idle time before the tester starts to transmit the address byte.                                                                                                                         |
| TIDLE                             | 0x13     | 0x0-0xFFFF<br>(1 ms per bit)                       | 300                     | For protocol ID of ISO 9141 or ISO 14230, this sets the minimum amount of bus idle time that is needed before a fast initialization sequence will begin.                                                                                         |
| TINIL                             | 0x14     | 0x0-0xFFFF<br>(1 ms per bit)                       | 25                      | For protocol ID of ISO 9141 or ISO 14230, this sets the duration for the low pulse in fast initialization.                                                                                                                                       |
| TWUP                              | 0x15     | 0x0-0xFFFF<br>(1 ms per bit)                       | 50                      | For protocol ID of ISO 9141 or ISO 14230, this sets the duration of the wake-up pulse in fast initialization.                                                                                                                                    |
| PARITY                            | 0x16     | 0 (NO_PARITY)<br>1 (ODD_PARITY)<br>2 (EVEN_PARITY) | 0                       | For a protocol ID of ISO 9141 or ISO 14230 only.                                                                                                                                                                                                 |
| BIT_SAMPLE_POINT                  | 0x17     | 0-100<br>(1% per bit)                              | 80                      | For a protocol ID of CAN, this sets the desired bit sample point as a percentage of the bit time.                                                                                                                                                |
| SYNC_JUMP_WIDTH                   | 0x18     | 0-100<br>(1% per bit)                              | 15                      | For a protocol ID of CAN, this sets the desired synchronization jump width as a percentage of the bit time.                                                                                                                                      |

SAE J2534-1 Revised DEC2004

| Parameter      | ID Value | Valid values for Parameter                                                                                                                                                                                                                                  | Default Value (decimal) | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T1_MAX         | 0x1A     | 0x0-0xFFFF<br>(1 ms per bit)                                                                                                                                                                                                                                | 20                      | For protocol ID of SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE or SCI_B_TRANS, this sets the maximum inter-frame response delay.                                                                                                                                                                                                                                                                                                                                                                            |
| T2_MAX         | 0x1B     | 0x0-0xFFFF<br>(1 ms per bit)                                                                                                                                                                                                                                | 100                     | For protocol ID of SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE or SCI_B_TRANS, this sets the maximum inter-frame request delay.                                                                                                                                                                                                                                                                                                                                                                             |
| T3_MAX         | 0x24     | 0x0-0xFFFF<br>(1 ms per bit)                                                                                                                                                                                                                                | 50                      | For protocol ID of SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE or SCI_B_TRANS, this sets the maximum response delay from the ECU after processing a valid request message from the tester.                                                                                                                                                                                                                                                                                                                  |
| T4_MAX         | 0x1C     | 0x0-0xFFFF<br>(1 ms per bit)                                                                                                                                                                                                                                | 20                      | For protocol ID of SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE or SCI_B_TRANS, this sets the maximum inter-message response delay.                                                                                                                                                                                                                                                                                                                                                                          |
| T5_MAX         | 0x1D     | 0x0-0xFFFF<br>(1 ms per bit)                                                                                                                                                                                                                                | 100                     | For protocol ID of SCI_A_ENGINE, SCI_A_TRANS, SCI_B_ENGINE or SCI_B_TRANS, this sets the maximum inter-message request delay.                                                                                                                                                                                                                                                                                                                                                                           |
| ISO15765_BS    | 0x1E     | 0x0-0xFF<br>(see ISO 15765-2)                                                                                                                                                                                                                               | 0                       | For protocol ID of ISO 15765, this sets the block size the interface should report to the vehicle for receiving segmented transfers.                                                                                                                                                                                                                                                                                                                                                                    |
| ISO15765_STMIN | 0x1F     | 0x0-0xFF<br>(see ISO 15765-2)                                                                                                                                                                                                                               | 0                       | For protocol ID of ISO 15765, this sets the separation time the interface should report to the vehicle for receiving segmented transfers.                                                                                                                                                                                                                                                                                                                                                               |
| BS_TX          | 0x22     | 0x0-0xFF, 0xFFFF<br>(see ISO 15765-2)                                                                                                                                                                                                                       | 0xFFFF                  | For protocol ID of ISO 15765, this sets the block size the interface should use to transmit segmented messages to the vehicle. The flow control value reported by the vehicle should be ignored. If value is 0xFFFF, use the value reported by the vehicle.                                                                                                                                                                                                                                             |
| STMIN_TX       | 0x23     | 0x0-0xFF, 0xFFFF<br>(see ISO 15765-2)                                                                                                                                                                                                                       | 0xFFFF                  | For protocol ID of ISO 15765, this sets the separation time the interface should use to transmit segmented messages to the vehicle. The flow control value reported by the vehicle should be ignored. If value is 0xFFFF, use the value reported by the vehicle.                                                                                                                                                                                                                                        |
| DATA_BITS      | 0x20     | 0 (8 data bits)<br>1 (7 data bits)                                                                                                                                                                                                                          | 0                       | For protocol ID of ISO 9141 or ISO 14230 only.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| FIVE_BAUD_MOD  | 0x21     | 0-Initialization as defined in ISO 9141-2 and ISO 14230-4<br>1-ISO 9141 initialization followed by interface sending inverted Key Byte 2<br>2- ISO 9141 initialization followed by ECU sending inverted address<br>3- Initialization as defined in ISO 9141 | 0                       | For a protocol ID of ISO 9141 or ISO 14230 only.<br><br>Initialization for ISO 9141-2 and ISO 14230 include the initialization sequence as defined in ISO 9141 plus inverted key byte #2 sent from the tester to the ECU and inverted address sent from the ECU to the tester.<br><br>This parameter allows either ISO 9141 initialization sequence, ISO 9141-2 / ISO 14230 initialization sequence, or hybrid versions which include only one of the extra bytes defined for ISO 9141-2 and ISO 14230. |

## SAE J2534-1 Revised DEC2004

| Parameter                  | ID Value             | Valid values for Parameter | Default Value (decimal) | Description                                                                                                           |
|----------------------------|----------------------|----------------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------|
| ISO15765_WFT_MAX           | 0x25                 | 0x0-0xff                   | 0                       | For protocol ID ISO 15765, the number of WAIT flow control frames allowed (N_WFTmax) during a multi-segment transfer. |
| Reserved                   | 0x26- 0x7FFF         |                            |                         | Reserved for SAE                                                                                                      |
| Reserved                   | 0x8000 – 0xFFFF      |                            |                         | Reserved for SAE J2534-2                                                                                              |
| Tool manufacturer specific | 0x10000 – 0xFFFFFFFF | Manufacturer Specific      |                         | Manufacturer Specific                                                                                                 |

FIGURE 30—I\_IOCTL GET\_CONFIG / SET\_CONFIG PARAMETER DETAILS

### 7.3.3 READ\_VBATT

The ioctlID value of READ\_VBATT is used to obtain the voltage measured on pin 16 of the SAE J1962 connector from the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 31. When the function is successfully completed, battery voltage will be placed in the variable pointed to by OutputPtr. The units will be in milli-volts and will be rounded to the nearest tenth of a volt.

| Parameter | Description                                       |
|-----------|---------------------------------------------------|
| ChannelID | Device ID assigned by DLL during PassThruOpen     |
| ioctlID   | Is set to the define READ_VBATT.                  |
| InputPtr  | Is a NULL pointer, as this parameter is not used. |
| OutputPtr | Is a pointer to an unsigned long.                 |

FIGURE 31—READ\_VBATT DETAILS

### 7.3.4 READ\_PROG\_VOLTAGE

The ioctlID value of READ\_PROG\_VOLTAGE is used to obtain the programming voltage of the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 32. When the function is successfully completed, programming voltage will be placed in the variable pointed to by OutputPtr. The units will be in milli-volts and will be rounded to the nearest tenth of a volt.

| Parameter | Description                                       |
|-----------|---------------------------------------------------|
| ChannelID | Device ID assigned by DLL during PassThruOpen     |
| ioctlID   | Is set to the define READ_PROG_VOLTAGE.           |
| InputPtr  | Is a NULL pointer, as this parameter is not used. |
| OutputPtr | Is a pointer to an unsigned long.                 |

FIGURE 32—READ\_PROG\_VOLTAGE DETAILS

### 7.3.5 FIVE\_BAUD\_INIT

The `loctIID` value of `FIVE_BAUD_INIT` is used to initiate a five-baud initialization sequence from the pass-thru device. The ISO 9141 five baud initialization sequence includes the five baud address sent from the tester to the ECU, followed by the synchronization byte and two key bytes sent from the ECU to the tester. The ISO 9141-2 and ISO 14230 five baud initialization includes the ISO 9141 initialization sequence, but adds the inverted key byte 2 value sent from the tester to the ECU, and the inverted initialization address sent from the ECU to the tester. The `FIVE_BAUD_MOD` parameter in `SET_CONFIG` allows the application to selectively include or exclude the two additional bytes defined in the ISO 14230 initialization sequence.

The calling application is responsible for allocating and initializing the associated parameters described in Figure 33. When the function is successfully completed, the key words will be placed in structure pointed to by `OutputPtr`. It should be noted that this only applies to Protocol ID of ISO9141 or ISO14230.

| Parameter              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ChannelID</code> | Channel ID assigned by DLL during <code>PassThruConnect</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>loctIID</code>   | Is set to the define <code>FIVE_BAUD_INIT</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>InputPtr</code>  | Points to the structure <code>SBYTE_ARRAY</code> , which is defined as follows:<br><pre> Typedef struct {     unsigned long NumOfBytes; /* number of bytes in the array */     unsigned char *BytePtr; /* array of bytes */ } SBYTE_ARRAY </pre> <p>where:<br/> <code>NumOfBytes</code> is an INPUT that must be set to "1" and indicates the number of bytes in the array <code>BytePtr</code>.<br/> <code>BytePtr[0]</code> is an INPUT that contains the target address.<br/> The remaining elements in <code>BytePtr</code> are not used.</p> |
| <code>OutputPtr</code> | Points to the structure <code>SBYTE_ARRAY</code> defined above<br><p>where:<br/> <code>NumOfBytes</code> is an INPUT which indicates the maximum size of the array <code>BytePtr</code> and an OUTPUT which indicates the number of bytes in the array <code>BytePtr</code>. Must be 2 or less.<br/> <code>BytePtr[0]</code> is an OUTPUT that contains key word 1 from the ECU.<br/> <code>BytePtr[1]</code> is an OUTPUT that contains key word 2 from the ECU.<br/> The remaining elements in <code>BytesPtr</code> are not used.</p>          |

FIGURE 33—FIVE\_BAUD\_INIT DETAILS

### 7.3.6 FAST\_INIT

The `loctIID` value of `FAST_INIT` is used to initiate a fast initialization sequence from the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 34. When the function is successfully completed, the response message will be placed in structure pointed to by `OutputPtr`. It should be noted that this only applies to Protocol ID of ISO9141 or ISO14230. In case of successful initialization the `OutputPtr` will contain valid data.

For the ISO9141 Protocol, no verification of message format will be applied.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                    |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | Channel ID assigned by DLL during PassThruConnect                                                                                                                                                                                                                                                                                                                              |
| loctlID   | Is set to the define FAST_INIT.                                                                                                                                                                                                                                                                                                                                                |
| InputPtr  | Points to the structure PASSTHRU_MSG (see the message definition section of this document) which the pass-thru device will send. This can be a Start Communication Request Message as defined by ISO 14230, or any other ISO 9141 or ISO 14230 message specified by the application. If NULL, no message is transmitted on the bus as part of the fast init sequence           |
| OutputPtr | Points to the structure PASSTHRU_MSG (see the message definition section of this document) which will hold the response to the Start Communication Request Message. If a response is not received in the time allowed by ISO14230 the Fast Initialization is deemed to have failed and the contents of this structure will be indeterminate. If NULL, no response is expected. |

FIGURE 34—FAST\_INIT DETAILS

### 7.3.7 CLEAR\_TX\_BUFFER

The loctlID value of CLEAR\_TX\_BUFFER is used to direct the pass-thru device to clear its transmit queue. The calling application is responsible for allocating and initializing the associated parameters described in Figure 35. When the function is successfully completed, the transmit queue will have been cleared.

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

FIGURE 35—CLEAR\_TX\_BUFFER DETAILS

### 7.3.8 CLEAR\_RX\_BUFFER

The loctlID value of CLEAR\_RX\_BUFFER is used to direct the pass-thru device to clear its receive queue. The calling application is responsible for allocating and initializing the associated parameters described in Figure 36. When the function is successfully completed, the receive queue will have been cleared.

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

FIGURE 36—CLEAR\_RX\_BUFFER DETAILS

## 7.3.9 CLEAR\_PERIODIC\_MSGS

The locID value of CLEAR\_PERIODIC\_MSGS is used to direct the pass-thru device to clear its periodic messages. The calling application is responsible for allocating and initializing the associated parameters described in Figure 37. When the function is successfully completed, the list will have been cleared and all periodic messages will have stopped transmitting.

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

FIGURE 37—CLEAR\_PERIODIC\_MSGS DETAILS

## 7.3.10 CLEAR\_MSG\_FILTERS

The locID value of CLEAR\_MSG\_FILTERS is used to direct the pass-thru device to clear its message filters on the specified channel. The calling application is responsible for allocating and initializing the associated parameters described in Figure 38. When the function is successfully completed, there will be no filters on the channel (the default state after a PassThruConnect). No more messages shall be queued until a PASS\_FILTER or FLOW\_CONTROL\_FILTER is added.

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

FIGURE 38—CLEAR\_MSG\_FILTERS DETAILS

## 7.3.11 CLEAR\_FUNCT\_MSG\_LOOKUP\_TABLE

The locID value of CLEAR\_FUNCT\_MSG\_LOOKUP\_TABLE is used to direct the pass-thru device to clear its functional message look-up table. The calling application is responsible for allocating and initializing the associated parameters described in Figure 39. When the function is successfully completed, the table will have been cleared. It should be noted that this only applies to Protocol ID of SAE J1850PWM.

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

FIGURE 39—CLEAR\_FUNCT\_MSG\_LOOKUP\_TABLE DETAILS

## 7.3.12 ADD\_TO\_FUNCT\_MSG\_LOOKUP\_TABLE

The `loctID` value of `ADD_TO_FUNCT_MSG_LOOKUP_TABLE` is used to add functional address(es) to the functional message look-up table in the physical layer of the vehicle network on the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 40. When the function is successfully completed, the look-up table will have been altered. It should be noted that this only applies to Protocol ID of SAE J1850PWM.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | Channel ID assigned by DLL during PassThruConnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| loctID    | Is set to the define <code>ADD_TO_FUNCT_MSG_LOOKUP_TABLE</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| InputPtr  | Points to the structure <code>SBYTE_ARRAY</code> , which is defined as follows:<br>Typedef struct<br>{<br>unsigned long NumOfBytes;   /* number of bytes in the array */<br>unsigned char *BytePtr;   /* array of bytes */<br>} <code>SBYTE_ARRAY</code><br><br>where:<br>NumOfBytes is an INPUT that indicates the number of bytes in the array <code>BytePtr</code> .<br><code>BytePtr[0]</code> is an INPUT that contains the first functional address to be added.<br>.<br>.<br>.<br><code>BytePtr[n]</code> is an INPUT that contains the nth functional address to be added. |
| OutputPtr | Is a NULL pointer, as this parameter is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

FIGURE 40—ADD\_TO\_FUNCT\_MSG\_LOOKUP\_TABLE DETAILS

## 7.3.13 DELETE\_FROM\_FUNCT\_MSG\_LOOKUP\_TABLE

The `loctID` value of `DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE` is used to delete functional address(es) from the functional message look-up table in the physical layer of the vehicle network on the pass-thru device. The calling application is responsible for allocating and initializing the associated parameters described in Figure 41. When the function is successfully completed, the look-up table will have been altered. It should be noted that this only applies to Protocol ID of J1850PWM.

| Parameter | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelID | Channel ID assigned by DLL during PassThruConnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| loctID    | Is set to the define <code>DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| InputPtr  | Points to the structure <code>SBYTE_ARRAY</code> , which is defined as follows:<br>Typedef struct<br>{<br>unsigned long NumOfBytes;   /* number of bytes in the array */<br>unsigned char *BytePtr;   /* array of bytes */<br>} <code>SBYTE_ARRAY</code><br><br>where:<br>NumOfBytes is an INPUT that indicates the number of bytes in the array <code>BytePtr</code> .<br><code>BytePtr[0]</code> is an INPUT that contains the first functional address to be deleted.<br>.<br>.<br>.<br><code>BytePtr[n]</code> is an INPUT that contains the nth functional address to be deleted. |
| OutputPtr | Is a NULL pointer, as this parameter is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

FIGURE 41—DELETE\_FROM\_FUNCT\_MSG\_LOOKUP\_TABLE DETAILS

## 8. Message Structure

The following message structure will be used for all messages (Transmit, Receive, Filters, and Periodics) and indications. The total message size (in bytes) is the DataSize, and includes header bytes, ID bytes, and data bytes. For consistency, all interfaces should detect only the errors listed for each protocol in the following sections when returning ERR\_INVALID\_MSG.

### 8.1 C / C++ Definition

```
typedef struct {
    unsigned long ProtocolID;
    unsigned long RxStatus;
    unsigned long TxFlags;
    unsigned long Timestamp;
    unsigned long DataSize;
    unsigned long ExtraDataIndex;
    unsigned char Data[4128];
} PASSTHRU_MSG;
```

### 8.2 Elements

|                |                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ProtocolID     | Protocol type                                                                                                                                                                                                                                                                                                                                                                                                       |
| RxStatus       | Receive message status – See RxStatus in “Message Flags and Status Definition” section                                                                                                                                                                                                                                                                                                                              |
| TxFlags        | Transmit message flags – See TxFlags in “Message Flags and Status Definition” section                                                                                                                                                                                                                                                                                                                               |
| Timestamp      | Received message timestamp (microseconds): For the START_OF_FRAME indication, the timestamp is for the start of the first bit of the message. For all other indications and transmit and receive messages, the timestamp is the end of the last bit of the message. For all other error indications, the timestamp is the time the error is detected.                                                               |
| DataSize       | Data size in bytes, including header bytes, ID bytes, message data bytes, and extra data, if any.                                                                                                                                                                                                                                                                                                                   |
| ExtraDataIndex | Start position of extra data in received message (for example, IFR). The extra data bytes follow the body bytes in the Data array. The index is zero-based. When no extra data bytes are present in the message, ExtraDataIndex shall be set equal to DataSize. Therefore, if DataSize equals ExtraDataIndex, there are no extra data bytes. If ExtraDataIndex=0, then all bytes in the data array are extra bytes. |
| Data           | Array of data bytes. Includes message headers, message body, and any extra data bytes.                                                                                                                                                                                                                                                                                                                              |

The application must fill all fields for each PASSTHRU\_MSG structure passed to the API, except for RxStatus, Timestamp, and ExtraDataIndex. These three fields are only valid when reading a message or indication with PassThruReadMsg.

### 8.3 Message Data Formats

This section describes the bytes in the Data section of the PASSTHRU\_MSG structure. Figure 42 shows the minimum and maximum transmit and receive message size for each protocol.

When using CAN or ISO 15765-4, the first 4 bytes of Data contain the CAN ID. 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. If ISO15765\_ADDR\_TYPE is set in TxFlags, then the next byte (Data[4]) will be the extended address.

| Protocol                    | Min Tx | Max Tx | Min Rx | Max Rx | Notes                                                                                                                                  |
|-----------------------------|--------|--------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
| CAN                         | 4      | 12     | 4      | 12     | 4 bytes of CANID, followed by up to 8 data bytes                                                                                       |
| ISO15765                    | 4      | 4099   | 4      | 4099   | 4 bytes of CAN ID, followed by up to 4095 data bytes                                                                                   |
| ISO15765 (Extended Address) | 5      | 4100   | 5      | 4100   | 4 bytes of CAN ID, 1 Extended Address byte, followed by up to 4095 data bytes                                                          |
| J1850PWM                    | 3      | 10     | 3      | 11     | 3 header bytes followed by up to 7 data bytes. On Rx, any number of IFR bytes as long as the total message size is less than 11 bytes. |
| J1850VPW                    | 1      | 4128   | 1      | 4128   |                                                                                                                                        |
| ISO9141                     | 1      | 4128   | 1      | 4128   |                                                                                                                                        |
| ISO14230                    | 1      | 259    | 1      | 259    | 1-4 header bytes followed by up to 255 data bytes.                                                                                     |
| ISO14230 (Manual Checksum*) | 1      | 260    | 1      | 260    | Format not defined. Allows for 4 header bytes, 255 data bytes and an application-defined checksum byte.                                |
| SCI                         | 1      | 4128   | 1      | 4128   |                                                                                                                                        |

FIGURE 42—ALLOWED MESSAGE SIZES PER PROTOCOL

NOTE—Manual checksum is when the ISO9141\_NO\_CHECKSUM bit in the Connect Flag is set to 1.

## 8.4 Format Checks for Messages Passed to the API

The vendor DLL shall validate all PASSTHRU\_MSG structures, and return an ERR\_INVALID\_MSG in the following cases:

- DataSize violates Min Tx or Max Tx columns in Figure 42
- Source address (Data[3]) is different from the Node ID (loctl SET\_CONFIG, Parameter NODE\_ADDRESS) on J1850PWM
- The header length field is incorrect for the number of bytes in the message on ISO14230
- The CAN\_29\_BIT flag of the message does not match the CAN\_29\_BIT flag passed to PassThruConnect, unless the CAN\_ID\_BOTH bit was set on connect

The vendor DLL shall return ERR\_MSG\_PROTOCOL\_ID when the ProtocolID field in the message does not match the Protocol ID specified when opening the channel.

## 8.5 Conventions for Returning Messages from the API

When returning a message in PassThruReadMsg:

- DataSize shall tell the application how many bytes in the Data array are valid. ExtraDataIndex will be the (non-zero) index of the last byte of the message. If ExtraDataIndex is not equal to DataSize there are extra data bytes after the message. If loopback is on, RxStatus must be consulted to tell if the message came via loopback.
- DataSize will be in the range shown in the Min Rx and Max Rx columns of Figure 42. If the device receives a message from the vehicle bus that is too long or too short, the message shall be discarded with no error.
- For received messages, ExtraDataIndex shall be equal to DataSize, except when the interface is returning SAE J1850 PWM IFR bytes. In no case shall ExtraDataIndex be larger than DataSize.
- When receiving a message on an SAE J1850 PWM channel, the message shall have any IFR bytes appended. In this case, ExtraDataIndex shall be the index of the first IFR byte, and DataSize shall be the total length of the original message plus all IFR bytes. For example, if there are two IFR bytes, DataSize will be incremented by two, and ExtraDataIndex will be DataSize - 2. When loopback is on, the loopback message shall contain any IFR bytes.

## 8.6 Conventions for Returning Indications from the API

When returning an indication in PassThruReadMsg:

- ExtraDataIndex must be zero
- DataSize shall tell the application how many bytes in the Data array are valid
- RxStatus must be consulted to determine the indication type (See Section 8.4).
- A TxDone indication (ISO 15765 only) is generated by the DLL after a SingleFrame message is sent, or the last frame of a multi-segment transmission is sent. DataSize shall be 4 (or 5 when the message was using Extended Addressing). Data shall contain the CAN ID (and possible Extended Address) of the message just sent. If loopback is on, the TxDone indication shall precede the loopback message in the receive queue.

## SAE J2534-1 Revised DEC2004

- An RxBreak indication (SAE J2610/SCI and SAE J1850VPW only) is generated by the DLL if a break is received.
- An RxStart indication is generated by the DLL when starting to receive a message on ISO9141 or ISO14230, or when receiving the FirstFrame signal of a multi-segment ISO 15765 message.

### 8.7 Message Flag and Status Definitions

#### 8.7.1 RxSTATUS

Definitions for RxStatus bits are shown in Figure 43. The application shall ignore any flags that do not apply to the current channel.

| Definition             | RxStatu<br>Bit(s) | Description                                                                                                                   | Value                                                                                                                                                                 |
|------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        | 31-24             | Tool manufacturer specific                                                                                                    | Shall be set to 0                                                                                                                                                     |
| Reserved               | 23-16             | Reserved for SAE J2534-2                                                                                                      | Shall be set to 0                                                                                                                                                     |
| Reserved               | 15-9              | Reserved for SAE                                                                                                              | Shall be set to 0                                                                                                                                                     |
| CAN_29BIT_ID           | 8                 | CAN ID Type for CAN and ISO 15765                                                                                             | 0 = 11-bit Identifier<br>1 = 29-bit Identifier                                                                                                                        |
| ISO15765_ADDR_TYPE     | 7                 | ISO 15765-2 Addressing Method                                                                                                 | 0= no extended address,<br>1= extended address is first byte after the CAN ID                                                                                         |
|                        | 6-5               | Reserved for SAE                                                                                                              | Shall be set to 0                                                                                                                                                     |
| ISO15765_PADDING_ERROR | 4                 | For ProtocolID ISO 15765 a CAN frame was received with less than 8 data bytes                                                 | 0 = No Error<br>1 = Padding Error                                                                                                                                     |
| TX_INDICATION          | 3                 | ISO 15765 TxDone indication- CANID and extended address, if present, shall be included in the message structure               | 0= No TxDone<br>1= TxDone                                                                                                                                             |
| RX_BREAK               | 2                 | Break indication received – SAE J2610 and SAE J1850 VPW only                                                                  | 0 = No break received<br>1 = Break received                                                                                                                           |
| START_OF_MESSAGE       | 1                 | Indicates the reception of the first byte of an ISO9141 or ISO14230 message or first frame of an ISO15765 multi-frame message | 0 = Not a start of message indication<br>1 = First byte or frame received                                                                                             |
| TX_MSG_TYPE            | 0                 | Receive Indication/Transmit Loopback                                                                                          | 0 = received i.e. this message was transmitted on the bus by another node,<br>1 = transmitted i.e. this is the echo of the message transmitted by the PassThru device |

FIGURE 43—RXSTATUS BIT DEFINITIONS

## 8.7.2 RxSTATUS BITS FOR MESSAGE STATUS AND ERROR INDICATIONS

Valid combinations of RxStatus bits for messages and indications are shown in Figure 44: Valid RxStatus Bit Combinations. When the DLL is returning both a TxDone indication and a Loopback message, the TxDone indication shall be queued first. The RxStatus bits CAN\_29BIT\_ID and ISO15765\_ADDR\_TYPE are not shown because they do not affect the message/indication type. See PassThruReadMsg for a listing of which indications are valid for each protocol.

| Status           | Description                                     | RxStatus Bit(s) / Definition        |                  |                   |                               |                          |
|------------------|-------------------------------------------------|-------------------------------------|------------------|-------------------|-------------------------------|--------------------------|
|                  |                                                 | 4<br>ISO15765_<br>PADDING_<br>ERROR | 3<br>TX_<br>DONE | 2<br>RX_<br>BREAK | 1<br>START_<br>OF_<br>MESSAGE | 0<br>TX_<br>MSG_<br>TYPE |
| Normal Message   | Response was received successfully              | 0                                   | 0                | 0                 | 0                             | 0                        |
| RxStart          | First byte/frame of a message received          | 0                                   | 0                | 0                 | 1                             | 0                        |
| RxBreak          | Break indication received                       | 0                                   | 0                | 1                 | 0                             | 0                        |
| RxPadError       | A CAN frame was received with less than 8 bytes | 1                                   | 0                | 0                 | 0                             | 0                        |
| TxDone           | Request was transmitted successfully            | 0                                   | 1                | 0                 | 0                             | 1                        |
| Loopback Message | Loopback of message transmitted                 | 0                                   | 0                | 0                 | 0                             | 1                        |

FIGURE 44—VALID RX STATUS BIT COMBINATIONS

## 8.7.3 TxFLAGS

Definitions for TxFlags bits are shown in Figure 45. The interface shall ignore any flags that do not apply to the current channel.

| Definition         | TxFlags Bit(s) | Description                                                                                                                     | Value                                                                                                                                                                                                                    |
|--------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | 31-24          | Unused                                                                                                                          | Tool manufacturer specific                                                                                                                                                                                               |
| SCI_TX_VOLTAGE     | 23             | SCI programming voltage                                                                                                         | 0 = no voltage after message transmit, 1 = apply 20V after message transmit                                                                                                                                              |
| SCI_MODE           | 22             | SCI transmit mode                                                                                                               | 0 = Transmit using SCI Full duplex mode.<br>1 = Transmit using SCI Half duplex mode.                                                                                                                                     |
|                    | 21 – 16        | Unused                                                                                                                          | Reserved for SAE J2534-2 – shall be set to 0                                                                                                                                                                             |
|                    | 15 - 10        | Unused                                                                                                                          | Reserved for SAE – shall be set to 0                                                                                                                                                                                     |
| WAIT_P3_MIN_ONLY   | 9              | Modified message timing for ISO 14230-used to decrease programming time if application knows only one response will be received | 0 = Interface message timing as specified in ISO 14230.<br>1 = After a response is received for a physical request, the wait time shall be reduced to P3_MIN. Does not affect timing on Responses to functional requests |
| CAN_29BIT_ID       | 8              | CAN ID type for CAN And ISO 15765                                                                                               | 0 = 11-bit, 1 = 29-bit                                                                                                                                                                                                   |
| ISO15765_ADDR_TYPE | 7              | ISO 15765-2 Addressing Method                                                                                                   | 0 = no extended address, 1 = extended address is first byte after the CAN ID                                                                                                                                             |
| ISO15765_FRAME_PAD | 6              | ISO 15765-2 Frame Padding                                                                                                       | 0 = no padding, 1 = pad all flow controlled messages to a full CAN frame using zeroes                                                                                                                                    |
|                    | 5-0            | Unused                                                                                                                          | Reserved for SAE – shall be set to 0                                                                                                                                                                                     |

FIGURE 45—TXFLAGS BIT DEFINITIONS

## **9. DLL Installation and Registry**

### **9.1 Naming of Files**

Each vendor will provide a different name implementation of the API DLL and a number of these implementations could simultaneously reside on the same PC. No vendor shall name its implementation "J2534.DLL". All implementations shall have the string "32" suffixed to end of the name of the API DLL to indicate 32-bit. For example, if the company name is "Vendor X" the name could be VENDRX32.DLL. For simplicity, an API DLL shall be named in accordance with the file allocation table (FAT) file system naming convention (which allows up to eight characters for the file name and three characters for the extension with no spaces anywhere). Note that, given this criteria, the major name of an API DLL can be no greater than six characters. The OEM application can determine the name of the appropriate vendor's DLL using the Win32 Registry mechanism described in this section.

### **9.2 Win32 Registry**

This section describes the use of the Windows Registry for storing information about the various vendors supplying the device drivers conforming to this recommended practice, the various devices supported by each vendor, information about each device, etc. The Win32 registration is shown in Figure 46.

The installation program provided with the interface shall create the appropriate registry entries listed below (including the PassThruSupport.04.04 key if not yet present). Only one key shall be created per DLL installed. The uninstall program shall remove its device-specific registry information, but shall not affect the remaining registry entries. The programming application shall use the Windows function RegEnumKeyEx (or similar) to enumerate all devices.

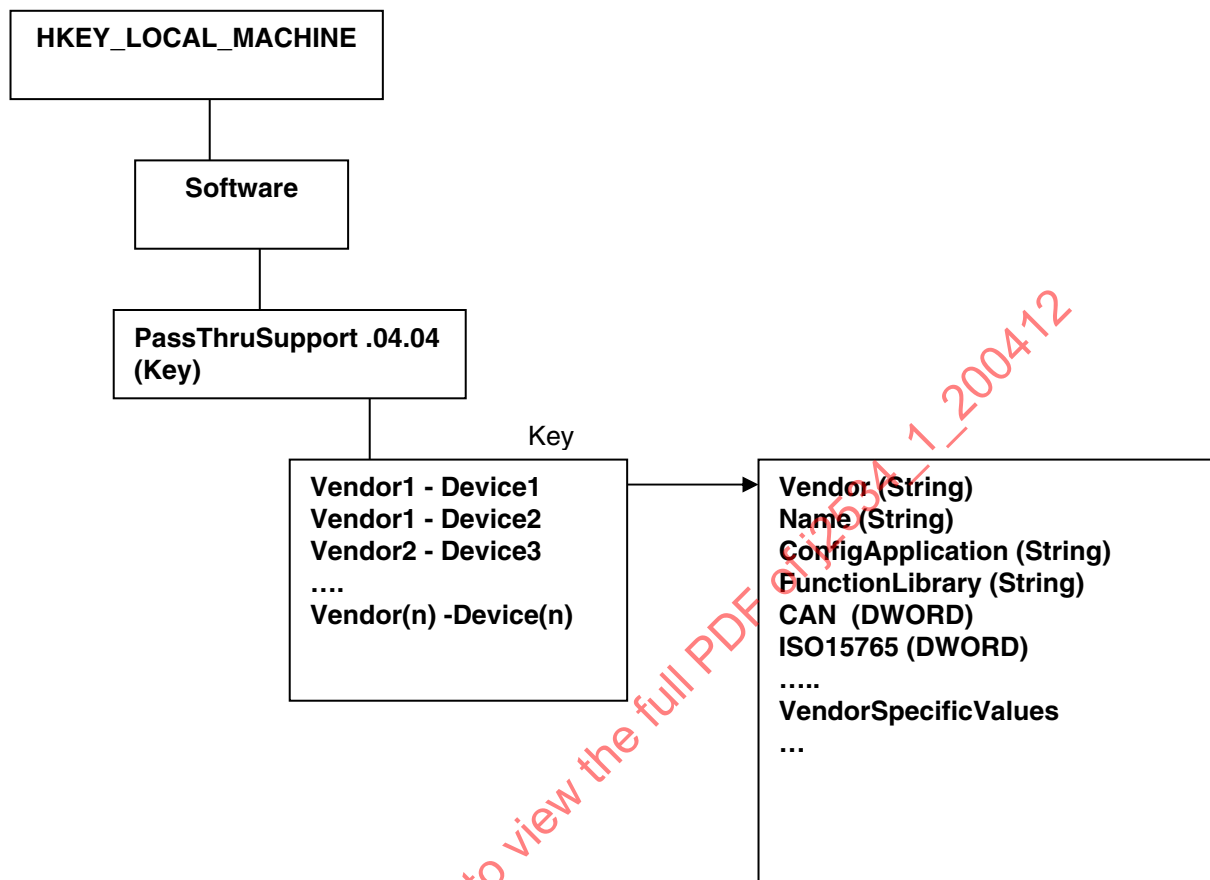


FIGURE 46—WIN32 REGISTRY

The registry will contain both:

- General information used by the user applications for selection of hardware, user information, etc.
- Vendor/Device specific information that the vendor uses in the implementation of the API. Considering that the object of this recommended practice is the need for interchangeability of hardware from various vendors, the user application using the this API will be required to use the registry to present to the users all the hardware devices that have been installed and display their capabilities. The user should be allowed to select any hardware having the required capabilities, in terms of protocols supported etc., for a particular reprogramming session.

## SAE J2534-1 Revised DEC2004

Under the PassThruSupport.04.04 key will be a list of devices. Each device key will consist of the name of the vendor, a hyphen (with spaces), and the name of the device. For example, "Acme Corp.-Turtle Programmer". Under each device key the following entries (values) included in Figure 47 shall be present:

|                                                                                                                              |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Vendor                                                                                                                       | String | The full name of the vendor. Ex: "ACME Corporation"                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Name                                                                                                                         | String | The name of the device. Ex: " ACME CAN Device over Ethernet"                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| CAN<br>ISO15765<br>J1850PWM<br>J1850VPW<br>ISO9141<br>ISO14230<br>SCI_A_ENGINE<br>SCI_A_TRANS<br>SCI_B_ENGINE<br>SCI_B_TRANS | DWORD  | For each of the supported protocols, the vendor can indicate how many simultaneous channels the hardware supports. The listing of a protocol here is only for the purpose of information and will not guarantee that the actual hardware will support the protocol, as it is possible that the hardware configuration may have changed.<br>Ex: CAN has a value of 2, J1850PWM has a value of 1, and ISO 9141 has a value of 0.<br>If protocol does not exist, its value is assumed to be zero. |
| ConfigApplication                                                                                                            | String | The complete path of the configuration application for this device. Every device vendor is required to provide a configuration application where the user can set the different parameters required for successfully using the device, like COM port, Ethernet address etc.<br>Ex: "c:\ACME\ACMESERCFG.exe"<br>The user applications using the API will automatically launch this application when the user needs to configure the selected device.                                            |
| FunctionLibrary                                                                                                              | String | The complete path of the DLL supplied by the vendor to communicate with this device. The user applications using this device should automatically load the DLL specified here and map into the J2534 API functions.<br>Ex: "C:\ACME\ACMESE32.dll"                                                                                                                                                                                                                                              |
| <Vendor Specific Values>                                                                                                     | -      | The vendor will store all the vendor specific information here.                                                                                                                                                                                                                                                                                                                                                                                                                                |

FIGURE 47—WIN32 REGISTRY VALUES

### 9.2.1 USER APPLICATION INTERACTION WITH THE REGISTRY

The user application should use the registry to present to the user the list of devices available for use from the application. Once the device has been selected by the user the Registry should be used to retrieve all the information regarding the device so that the appropriate DLL can be loaded for use etc. Figure 48 is a flow chart that shows a typical usage.