



SURFACE VEHICLE RECOMMENDED PRACTICE	J2534™	JUL2019
	Issued 2002-02 Stabilized 2019-07	
	Superseding J2534 FEB2002	
Recommended Practice for Pass-Thru Vehicle Programming		

RATIONALE

This document is still required in some regulatory language, but no more changes are intended for this technology level.

STABILIZED NOTICE

This document (API Version "02.02") has been declared "Stabilized" by the SAE Vehicle E E System Diagnostic Standards Committee and will no longer be subjected to periodic reviews for currency. Users are responsible for verifying references and continued suitability of technical requirements. Newer technology may exist.

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 revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2019 SAE International

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

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

SAE WEB ADDRESS:

Full document details: [SAE J2534_201907](http://standards.sae.org/J2534_201907)

SAE values your input. To provide feedback on this Technical Report, please visit http://standards.sae.org/J2534_201907

Foreword—The use of reprogrammable memory technology in vehicle electronic control units (ECU's) 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 ECU's 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 ECU's 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 ECU's.

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 ECU's for all vehicles with a minimal investment in hardware needed to communicate with the vehicles. Both agencies have proposed 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.

TABLE OF CONTENTS

1.	Scope	4
2.	References	4
2.1	Applicable Publications	4
2.1.1	SAE Publications	4
2.1.2	ISO Documents	4
3.	Definitions	4
4.	Acronyms	5
5.	Pass-Thru Concept.....	5
6.	Pass-Thru System requirements	6
6.1	PC requirements.....	6
6.2	Software Requirements and Assumptions	6
6.3	Connection to PC.....	6
6.4	Connection to Vehicle	6
6.5	Communication Protocols.....	6
6.5.1	ISO 9141.....	7
6.5.2	ISO 14230-4 (KWP2000)	7
6.5.3	SAE J1850 41.6 kbps PWM (pulse width modulation).....	7
6.5.4	SAE J1850 10.4 kbps VPW (variable pulse width)	7
6.5.5	CAN	7
6.5.6	ISO 15765-4 (CAN)	7
6.5.7	SAE J2610 DaimlerChrysler SCI	7
6.6	Programmable power supply	8
6.7	Data Buffering	8
7.	Win32 Application Programming Interface	8
7.1	API Functions – Overview	8
7.2	API Functions - Detailed Information	8
7.2.1	PassThruConnect	8
7.2.2	PassThruDisconnect.....	10
7.2.3	PassThruReadMsgs	11
7.2.4	PassThruWriteMsgs	12
7.2.5	PassThruStartPeriodicMsg	13
7.2.6	PassThruStopPeriodicMsg	14
7.2.7	PassThruStartMsgFilter	14
7.2.8	PassThruStopMsgFilter	16
7.2.9	PassThruSetProgrammingVoltage	17
7.2.10	PassThruReadVersion	18
7.2.11	PassThruGetLastError	19
7.2.12	PassThruIoctl	19
7.3	IOCTL Section	21
7.3.1	GET_CONFIG	21
7.3.2	SET_CONFIG.....	22
7.3.3	READ_VBATT	25
7.3.4	READ_PROG_VOLTAGE	26
7.3.5	FIVE_BAUD_INIT	26
7.3.6	FAST_INIT	26
7.3.7	CLEAR_TX_BUFFER.....	27

7.3.8	CLEAR_RX_BUFFER	27
7.3.9	CLEAR_PERIODIC_MSGS.....	27
7.3.10	CLEAR_MSG_FILTERS	27
7.3.11	CLEAR_FUNCT_MSG_LOOKUP_TABLE	28
7.3.12	ADD_TO_FUNCT_MSG_LOOKUP_TABLE.....	28
7.3.13	DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE.....	29
8.	Message Structure	29
8.1	C / C++ Definition	29
8.2	Elements.....	29
8.3	Message Data Formats	30
8.3.1	CAN Data Format	30
8.3.2	ISO 15765-4 Data Format	30
8.3.3	SAE J1850PWM Data Format	31
8.3.4	SAE J1850VPW Data Format.....	31
8.3.5	ISO 9141 Data Format	32
8.3.6	ISO 14230-4 Data Format	32
8.3.7	SCI Data Format	32
8.4	Message Flag and Status Definitions	33
8.4.1	RxStatus	33
8.4.2	TxFlags	33
9.	DLL Installation and Registration	34
9.1	Naming of Files.....	34
9.2	Win32 Registration	34
9.2.1	User Application Interaction with the Registry	36
9.2.2	Attaching to the DLL from an application	37
10.	Return Value Error Codes	37
Appendix A	General ISO 15765-2 Flow Control Example	39
A.1	Normal Addressing Used	39
A.2	General Request Message Flow Example	39
A.3	General Response Message Flow Example.....	40
Appendix B	Message Filter Usage Example	42
B.1	Filter Usage	42
B.2	Transmission of a Multi-Frame Request Message	42
B.3	Reception of a Multi-Frame Response Message.....	42
B.4	Filter Configuration	42
B.4.1	Request Message Transmission	44
B.4.2	Response Message Reception.....	45
B.5	ISO 15765-2 Extended Addressing Notes.....	46

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 (ECU's) 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.

The U.S. Environmental Protection Agency (EPA) and the California Air Resources Board (ARB) have proposed requirements for reprogramming vehicles for all manufacturers by the aftermarket repair industry. This document is intended to meet those proposed requirements for 2004 model year vehicles. Additional requirements for the 2005 model year 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/DIS 15765-2—Road vehicles—Diagnostics on controller area networks (CAN)—Network layer services
ISO/DIS 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 for Character 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.

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

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 manufacturer of an SAE J2534 pass-thru device must supply both the device driver software and the pass-thru device hardware that communicates directly with the vehicle. 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.

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.

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 SAE J1962 on vehicle end 5 meter max. length	Capabilities defined in J2534 Hardware supplied by tool supplier Could be implemented in scan tool	Tool supplier defines cable requirements, if any, between PC and interface	Tool supplier device driver based on hardware supported Examples are: RS-232, USB, PCMCIA, Ethernet, IEEE1394, Bluetooth	J2534	Vehicle manufacturer programming application controls user interface, vehicle software and calibration selection, programming sequence, and security access
Vehicle manufacturer	Tool supplier				J2534	Vehicle manufacturer

FIGURE 1—SAE J2534 OVERVIEW

6. Pass-Thru System Requirements

- 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 communicated on sequentially (serialized) from the single application thread. This will prevent the unnecessary complexity of determining what message responses belong to which application thread.
- 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. Vehicle manufacturers will need to supply information about necessary connections to any connector other than the SAE J1962 connector.
- 6.5 Communication Protocols**—A fully compliant pass-thru interface shall support all communication protocols as specified in this section. Additionally, the pass-thru device must support simultaneous communication of an ISO 9141 OR ISO 14230-4 protocol AND an SAE J1850 protocol AND a CAN or SCI based protocol during a single programming event. Note that only one type of SAE J1850 is required per programming event, as the two types of SAE J1850 are mutually exclusive on any given vehicle. As well, CAN and SCI are mutually exclusive on some vehicles as the same pins are used.

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 minimum 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 rates with $\pm 2\%$ tolerance: 9600, 9615, 10000, 10870, 11905, 12500, 13158, 13889, 14706, and 15625.
 - f. 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.
- 6.5.2 ISO 14230-4 (KWP2000)—The ISO 14230-4 protocol is the same as the ISO 9141 protocol with the following additions:
- a. The interface will handle the tester present message and 0x78 response code automatically (i.e., without intervention from the PC).
- 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 for 41.6 kbps PWM:
- a. Capable of 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 for 10.4 kbps VPW:
- a. High speed mode of 41.6 kbps
 - b. 4K block transfer
- 6.5.5 CAN—The following features of ISO 11898 must be supported by the pass-thru device:
- a. 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. Pass-thru message interface (i.e., raw CAN frames with no flow control in the pass-thru device)
- 6.5.6 ISO 15765-4 (CAN)—The following features of ISO 15765-4 must be supported by the pass-thru device:
- a. 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.
- 6.5.7 SAE J2610 DAIMLERCHRYSLER SCI—Reference the SAE J2610 Information Report for a description of the SCI protocol.

6.6 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. 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
- b. Maximum 20 V
- c. Accuracy ± 0.1 V
- d. Maximum source current 200 mA
- e. Maximum sink current 300mA (only for SHORT_TO_GROUND option).
- f. Maximum 1 ms settling time (required for SCI protocol, reference SAE J2610 Information Report)
- g. Pin assignment software selectable

6.7 Data Buffering—The interface shall be capable of buffering a 4K byte transmit message as well as a 4K byte receive message.

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 2.

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

FIGURE 2—SAE J2534 API FUNCTIONS

7.2 API Functions - Detailed Information

7.2.1 PASSTHRUCONNECT—This function is used to establish a logical connection with a protocol channel. After this function is called, the value pointed to by pChannelID is used as the logical identifier for the connection. The DLL can use this function to initialize data structures and device drivers. If the function operates successfully, 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 all filters for the given protocol will be cleared with this function.

7.2.1.1 C / C++ Prototype

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

7.2.1.2 Parameters

ProtocolID Protocol ID.
 Flags Connection flags, normally set to zero.
 pChannelID Pointer to location for the channel ID that is assigned by the DLL.

7.2.1.3 Flag Values—See Figure 3.

Flags Bit(s)	Description	Value
31-9	Unused	Tool manufacturer specific
8	CAN ID type	0 = 11-bit, 1 = 29-bit
7	ISO15765-2 Addressing Method	0 = no extended address, 1 = extended address is first byte after the ID bytes
6-0	Unused	Reserved for SAE - shall be set to 0

FIGURE 3—FLAG VALUES

7.2.1.4 ProtocolID Values—See Figure 4.

Definition	Description	Value(s)
J1850VPW	GM / DaimlerChrysler CLASS2	0x01
J1850PWM	Ford SCP	0x02
ISO9141	ISO9141 and ISO9141-2	0x03
ISO14230	ISO14230-4 (Keyword Protocol 2000)	0x04
CAN	Raw CAN (flow control not handled automatically by interface)	0x05
ISO15765	ISO15765-2 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
Unused	Reserved for SAE use	0x0B – 0xFFFF
Unused	Tool manufacturer specific	0x10000 – 0xFFFFFFFF

FIGURE 4—PROTOCOL ID VALUES

7.2.1.5 Return Values—See Figure 5.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_INVALID_PROTOCOL_ID	Invalid ProtocolID value 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_NULLPARAMETER	NULL pointer supplied where a valid pointer is required
ERR_INVALID_FLAGS	Invalid flag values.
ERR_FAILED	Undefined error, use PassThruGetLastError for text description
ERR_CHANNEL_IN_USE	Channel number is currently connected.

FIGURE 5—RETURN VALUES

7.2.2 **PASSTHRU_DISCONNECT**—This function is used to terminate a logical connection with a protocol channel. The DLL can use this function to de-allocate data structures and deactivate any device drivers. If the function operates successfully, a value of STATUS_NOERROR is returned. After this call the Channel ID will no longer be valid.

7.2.2.1 C / C++ Prototype

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

7.2.2.2 Parameters

ChannelID The channel ID assigned by the PassThruConnect function.

7.2.2.3 Return Values—See Figure 6.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_FAILED	Undefined error, use PassThruGetLastError for text description
ERR_INVALID_CHANNEL_ID	Invalid ChannelID value.

FIGURE 6—RETURN VALUES

7.2.3 PASSTHREADMSGs—This function reads messages from the receive buffer in the order they were received. If the function operates successfully, a value of STATUS_NOERROR is returned. Note that the ISO 15765-2 FirstFrame and TxDone indications will be returned as messages when calling this function. Also note that all messages and indications shall be read in the order that they occurred on the bus.

7.2.3.1 C / C++ Prototype

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

7.2.3.2 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 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.

7.2.3.3 Return Values—See Figure 7.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_INVALID_CHANNEL_ID	Invalid ChannelID value.
ERR_NULLPARAMETER	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 should be returned.
ERR_BUFFER_EMPTY	No messages available to read.
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 7—RETURN VALUES

7.2.4 **PASSTHRUWRITEMSGs**—This function is used to send messages. The messages are placed in the buffer and sent in the order they were received. If the function operates successfully, a value of STATUS_NOERROR is returned. To perform blocking writes (i.e., the function does not return until message is successfully sent on the vehicle network or a timeout occurs), use the blocking flag in the TxFlags element of the message structure (Reference 8.4.2).

7.2.4.1 C / C++ Prototype

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

7.2.4.2 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 or placed in the transmit queue.
Timeout	Write timeout (in milliseconds). If a value of 0 is specified the function 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 written. If the number of messages requested have been written, the function shall not return ERR_TIMEOUT, even if the timeout value is zero.

7.2.4.3 Return Values—See Figure 8.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_INVALID_CHANNEL_ID	Invalid ChannelID value.
ERR_INVALID_MSG	Invalid message structure pointed to by pMsg (e.g. sending a 20 byte long J1850PWM message, sending a J1850PWM message where the third data byte is not the same as the node ID, etc.).
ERR_NULLPARAMETER	NULL pointer supplied where a valid pointer is required.
ERR_FAILED	Undefined error, use PassThruGetLastError for text description
ERR_TIMEOUT	Timeout.
ERR_MSG_PROTOCOL_ID	Protocol type in the message does not match protocol associated with the ChannelID
ERR_BUFFER_FULL	Protocol message buffer is full.

FIGURE 8—RETURN VALUES

7.2.5 **PASSTHRUSTARTPERIODICMSG**—This function starts sending a message at the specified interval. If the function operates successfully, a value of STATUS_NOERROR is returned. The maximum number of periodic messages is ten.

7.2.5.1 C / C++ Prototype

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

7.2.5.2 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.

7.2.5.3 Return Values—See Figure 9.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_INVALID_CHANNEL_ID	Invalid ChannelID value.
ERR_INVALID_MSG	Invalid message structure pointed to by pMsg.
ERR_NULLPARAMETER	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 protocol associated with the ChannelID
ERR_EXCEEDED_LIMIT	Exceeded the maximum number of periodic message IDs or the maximum allocate space.

FIGURE 9—RETURN VALUES

7.2.6 **PASSTHRUSTOPPERIODICMSG**—This function stops the process of sending a periodic message. If the function operates successfully, a value of STATUS_NOERROR is returned. After this call the MsgID will be invalid.

7.2.6.1 *C / C++ Prototype*

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

7.2.6.2 *Parameters*

ChannelID The channel ID assigned by the PassThruConnect function.
MsgID Message ID that is assigned by the PassThruStartPeriodicMsg function.

7.2.6.3 *Return Values*—See Figure 10.

Definition	Description
STATUS_NOERROR	Function call successful.
ERR_DEVICE_NOT_CONNECTED	Device not connected to PC.
ERR_INVALID_CHANNEL_ID	Invalid ChannelID value.
ERR_FAILED	Undefined error, use PassThruGetLastError for text description
ERR_INVALID_MSG_ID	Invalid MsgID value.

FIGURE 10—RETURN VALUES

7.2.7 **PASSTHRUSTARTMSGFILTER**—This function starts filtering incoming messages. If the function operates successfully, a value of STATUS_NOERROR is returned. The maximum number of message filters is ten. See Appendices A and B for a description of the use of these message filters for transmission and reception of multi-frame messages.

7.2.7.1 *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 *pMsgID
)
```