SAE International™

SURFACE VEHICLE RECOMMENDED PRACTICE

SAE J2534-2

ISSUED MAR2006

Issued 2006-03

Optional Pass-Thru Features

TABLE OF CONTENTS

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
SAE WEB ADDRESS: http://www.sae.org

1. Scope

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

The SAE J2534-2 document meets these needs by detailing extensions to an SAE J2534-1 interface. Together, these extensions provide the framework for a common interface to protect the software investment of the Vehicle OEMs and Scan Tool manufacturers.

Only the optional features will be described by this document. Unless otherwise noted it is expected that these features are added to a fully compliant interface adhering to the December 2004 publication of SAE J2534-1.

1.1 Purpose

Each section included in this paper documents specific features that may be added to a fully compliant SAE J2534-1 interface. The specific feature operation will be described directly or reference another existing specification. In each case the required calling structure, via the SAE J2534-1 API, will be documented and coordinated by this document.

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

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

2. References

2.1 General References

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

SAE J2534-1—Recommended Practice for Pass-Thru Programming
SAE J1962—Diagnostic Connector

2.2 References for Single Wire CAN

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

GMW3089—GMLAN Single Wire CAN Physical and Data Link Layers Specification Definitions
GMW3173—Architecture and Bus Wiring Requirements
GMW3110—GMLAN Enhanced Diagnostics Test Mode Specifications

2.3 References for GM UART

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

SAE J2740—General Motors UART Serial Data Communications

3. Definitions

3.1 Definitions for Analog Inputs

Channel	An analog-to-digital channel
Subsystem	A collection of similar channels

4. Acronyms

4.1 Acronyms for Single Wire CAN

GMLAN	General Motors Local Area Network
SWCAN	Single Wire CAN.

4.2 Acronyms for GM UART

UART	Universal Asynchronous Receiver/Transmitter

4.3 Acronyms for Analog Inputs

A/D	Analog to Digital

5. SAE J1962 Pin Selection

5.1 Scope of the SAE J1962 Pin Selection Optional Feature

This section identifies the pin selection mechanism for ProtocolIDs that have the '_PS' suffix. The API extensions detailed here describe the method of specifying the SAE J1962 pin(s) to which the selected protocol should be connected. While the API allows for all combinations of protocol / pin assignments, the actual combinations implemented are vendor specific.

5.2 Pass-Thru System Requirements

5.2.1 PIN USAGE

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

5.3 Win32 Application Programming Interface

5.3.1 API FUNCTIONS – OVERVIEW

The new ProtocolIDs with '_PS' suffix indicates that the protocol physical layer is not connected to the SAE J1962 pins on PassThruConnect. A new Ioctl configuration parameter is also added, that allows connection of a physical layer to specific SAE J1962 pins.

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

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

FIGURE 1—PROTOCOL ID VALUES FOR SAE J2534-1 DEFINED PROTOCOLS

As an example, in order to utilize a CAN channel connected to Pins 3 and 11 (often used for Medium Speed CAN network), the new CAN_PS ProtocolID is used in PassThruOpen.

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

New SAE J2534-2 optional feature protocols use the ProtocolIDs defined in Figure 2:

Definition	Description
SW_ISO15765_PS	Single Wire CAN adhering to ISO15765-2 flow control
SW_CAN_PS	Raw Single Wire CAN
GM_UART_PS	GM UART

FIGURE 2— PROTOCOL ID VALUES FOR SAE J2534-2 DEFINED PROTOCOLS

The interface must manage resource locking of SAE J1962 pins that are in use. If an existing channel is using a pin that is requested by a PassThruIoctl call, the new request shall be rejected.

Figure 3 summarizes the changes to the SAE J2534-1 API Functions.

Function	Description of Change
PassThruConnect	For all protocols with the '_PS' suffix defined above, no connection to the SAE J1962 pins is made on the call to PassThruConnect.
PassThruIoctl	Add a new configuration parameter to allow selection of SAE J1962 pins.

FIGURE 3—SAE J2534 API FUNCTIONS

5.3.2 API FUNCTIONS – DETAILED INFORMATION

5.3.2.1 PassThruConnect

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

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

- The new ProtocolIDs (ending in '_PS') are not assigned pins upon connection. No transmission or reception on a channel is possible until the pins are assigned using the IOCTL parameter J1962_PINS.
- The '_PS' ProtocolIDs can be opened multiple times. A program could open CAN_PS and request pins 3 and 11, then open CAN_PS again and request pins 1 and 12.

Devices that do not support any transceivers of a particular type (e.g. SW_CAN_PS), must return ERR_NOT_SUPPORTED any time the channel is opened.

Devices with only one transceiver of a particular type (e.g., one physical CAN bus) shall disallow opening that '_PS' ProtocolID multiple times (since it could never satisfy the 2nd request). In that case, the device should return ERR_DEVICE_IN_USE.

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

5.3.2.2 PassThruDisconnect

This API Function shall return the SAE J1962 Pins to the default state as specified by SAE J2534-1.

5.3.2.3 PassThruReadMsgs

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

5.3.2.4 PassThruWriteMsgs

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

5.3.2.5 PassThruStartPeriodicMsg

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

5.3.2.6 PassThruIoctl

Pins are assigned via SET_CONFIG with the J1962_PINS parameter. Refer to the SET_CONFIG section for details regarding this parameter. Other PassThruIoctl functions called prior to pin assignment will result in an error and the return value will be ERR_PIN_INVALID.

5.3.2.7 Return Values

Figure 4 defines added error value for the SAE J1962 Pin Selection Feature:

Definition	**Description**
ERR_PIN_INVALID	Invalid pin number, pin number already in use, or voltage already applied to a different pin.

FIGURE 4—ERROR VALUES

5.3.3 IOCTL SECTION

5.3.3.1 GET_CONFIG

There is a new configuration parameter called J1962_PINS. See Figure 5 for more details.

5.3.3.2 SET_CONFIG

A new configuration parameter, J1962_PINS, is added as defined in Figure 5. For the protocol channel referenced in the ChannelID parameter of the SET_CONFIG call, this parameter specifies which SAE J1962 pin or pins this protocol's physical layer is to be connected to. The act of setting this parameter causes the connection of the protocol physical layer to the specified pins.

Parameter	Valid values for Parameter	Default Value (Decimal)	Description
J1962_PINS	0xPPSS where: PP: 0x00 – 0x10 SS: 0x00 – 0x10 PP != SS, except when set to 0x0000 Exclude pins 4, 5, and 16	0	For a channel of any protocol type this selects the SAE J1962 pin, or pair of pins, onto which the physical layer is to be switched. NOTE: A value of 0 can never be set. Reading a value of 0 indicates that pin selection has not been performed. PP is the pin number for the primary signal e.g. ISO K-line, CAN-H, $+V_e$, SCI Tx, DIAG-H, SAE J1850+,…. SS is the pin number for the secondary signal, where a secondary signal is present e.g. ISO L-line, CAN-L, $-V_e$, SCI Rx, SAE J1850-, … SS shall equal 0x00 if no secondary pin is required or enabled.

FIGURE 5—IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

For existing SAE J2534-1 base protocols, the physical interface is connected to the SAE J1962 pins automatically when PassThruConnect is called, as defined by the Pin Usage section in SAE J2534-1. The J1962_PINS parameter does not need to be supported for SAE J2534-1 base protocols.

For protocols with the '_PS' suffix, the J1962_PINS parameter must be supported and the default value of the parameter shall be 0x0000. Certain combinations of SAE J1962 pins could result in vehicle or interface damage. ERR_PIN_INVALID is returned for combinations not supported or having the potential of damaging the interface device.

Only one SET_CONFIG, with the parameter J1962_PINS can be performed for a given Channel ID. If a second call for a given Channel ID is attempted, ERR_INVALID_IOCTL_VALUE will be returned. PassThruDisconnect is required before an alternate pin selection may be attempted.

For the J1962_PINS parameter, the following error handling shall be applied:

- **PassThruIoctl requests unsupported pin combination**

  PassThruIoctl, SET_CONFIG, is called, requesting a value of the J1962_PINS parameter that can't be supported by the hardware.

  Result: PassThruIoctl returns ERR_NOT_SUPPORTED

- **PassThruIoctl requests pin combination that would cause conflict**

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

  Result: PassThruIoctl returns ERR_PIN_INVALID.

6. Access to Additional Channels

This section defines an optional mechanism available to initiate and use multiple channels of the same protocol, if vendor hardware provides the support. The channels addressed this way may not be tied to pins on the SAE J1962 connector. What channel appears at what pins on the interface device will depend on the vendor configuration.

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

CAN_CH1
CAN_CH2
…………..
CAN_CH128

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

7. Accessing Multiple SAE J2534 Devices

SAE J2534-2 allows the use of multiple devices from the same PC. The connection is established by passing the device identification in the PassThruOpen pName parameter. This function will expect this parameter to be a NULL terminated string, typecast as a void pointer. The content of the string is currently vendor specific.

It is acceptable to pass a NULL parameter as defined in SAE J2534-1. In this case, the vendor determined default device will be selected.

8. Mixed Format Frames on a CAN Network

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

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

8.2 Win32 Application Programming Interface

8.2.1 API Functions – Overview

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

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

The function PassThruStartMsgFilter shall be used to identify ISO 15765 messages as well as unformatted CAN frames. A received message that matches a FLOW_CONTROL_FILTER shall be processed as an ISO 15765 message and shall have the appropriate *ProtocolID* in the PASSTHRU_MSG structure before it is added to the receive queue. Additionally, based upon the setting of CAN_MIXED_FORMAT, the CAN frames may also be subject to the PASS_FILTERs and BLOCK_FILTERs, where the *ProtocolID* in the PASSTHRU_MSG structure shall be set to reflect an unformatted CAN frame before it is added to the receive queue. The Figure 6 and Figure 7 outline how received messages are processed for the various settings of CAN_MIXED_FORMAT:

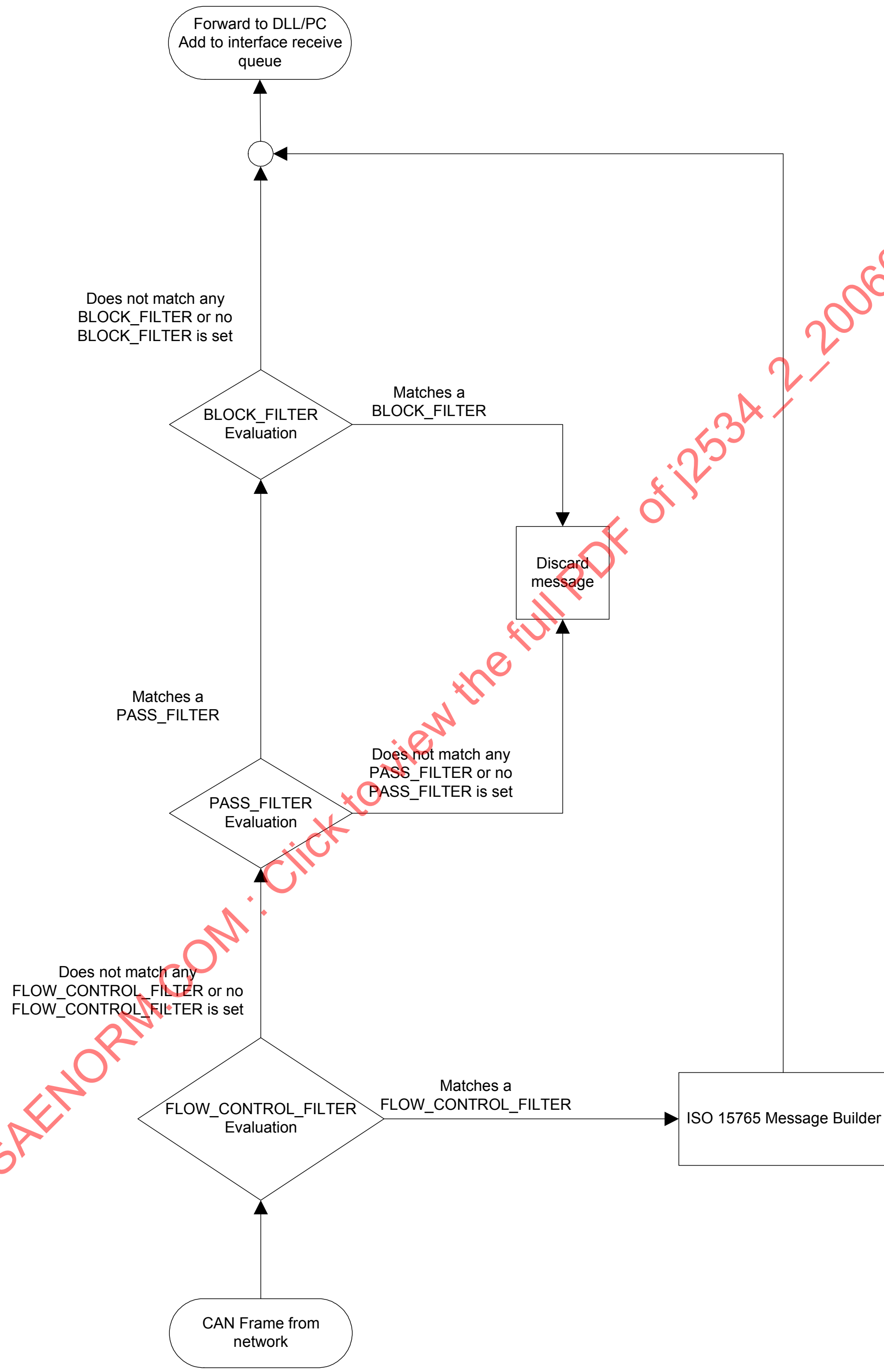

FIGURE 6—PROCESSING OF RECEIVED MESSAGES WHEN CAN_MIXED_FORMAT IS CAN_MIXED_FORMAT_ON

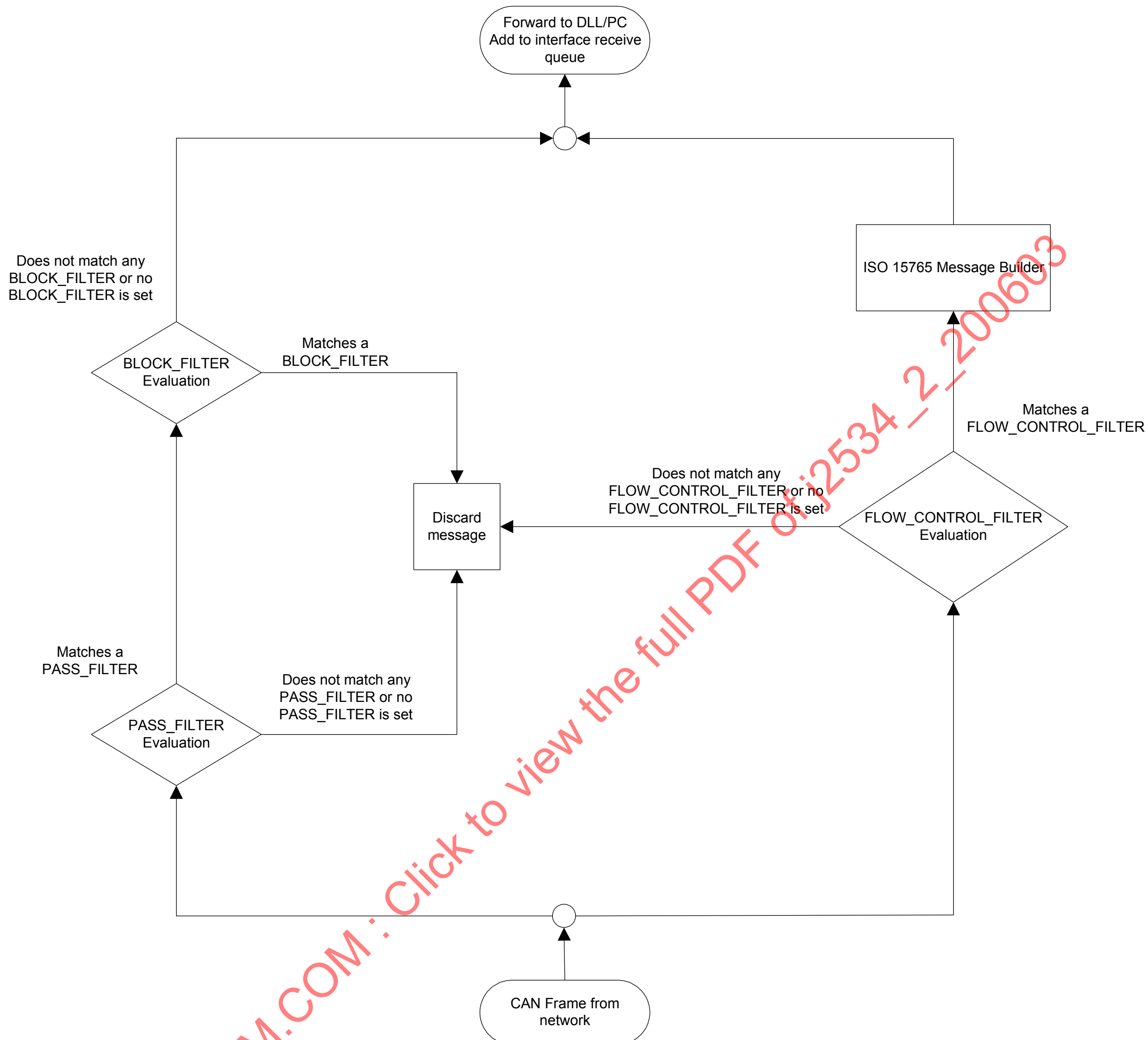

FIGURE 7—PROCESSING OF RECEIVED MESSAGES WHEN CAN_MIXED_FORMAT IS CAN_MIXED_FORMAT_ALL_FRAMES

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

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

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

FIGURE 8—SAE J2534 API FUNCTIONS

8.2.2 API Functions – Detailed Information

8.2.2.1 PassThruReadMsgs

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

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

8.2.2.2 PassThruWriteMsgs

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

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

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

8.2.2.3 PassThruStartPeriodicMsg

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

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

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

8.2.2.4 PassThruStartMsgFilter

Each ISO 15765 channel shall support a minimum of 64 FLOW_CONTROL_FILTERs as well as 10 PASS_FILTERs/BLOCK_FILTERs.

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

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

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

8.2.3 IOCTL SECTION

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

8.2.3.1 GET_CONFIG

The configuration parameter CAN_MIXED_FORMAT has been defined but it is only applicable to ISO 15765 channels. See Figure 9 for more details.

8.2.3.2 SET_CONFIG

The configuration parameter CAN_MIXED_FORMAT has been defined but it is only applicable to ISO 15765 channels. CAN_MIXED_FORMAT configuration parameter is defined in Figure 9.

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

FIGURE 9—IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

8.3 Message Structure

8.3.1 ELEMENTS

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

9. Single Wire CAN

9.1 Scope of the Single Wire CAN Optional Feature

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

9.2 Pass-Thru System Requirements

9.2.1 PIN USAGE

General Motors Corporation use of Single Wire CAN is connected to pin 1 of the SAE J1962 connector.

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

9.3 Win32 Application Programming Interface

9.3.1 API FUNCTIONS – OVERVIEW

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

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

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

At a high level a new ProtocolID has been defined to indicate the use of the Single Wire CAN physical layer. Additionally, Flags required to support high voltage wake up and high / normal speed are defined as required. The device is required to support high voltage wakeup and time critical data rate changes defined in SWCAN specification. This document details the API resources required to enable use of the SAE J2534 API as applied to SWCAN.

If this feature is not supported an error code, ERR_INVALID_PROTOCOL_ID, will be returned by the call PassThruConnect. The calling application will be required to notify the user that this optional feature may not be supported by the interface.

Example

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

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

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

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

Assume all settings are in the Default mode.

1. The application sets the configuration of the SW_CAN_RES_SWITCH parameter to 2 – AUTO_RESISTOR.
2. The application sets the configuration of SW_CAN_SPEEDCHANGE_ENABLE to 1 – ENABLE_SPDCHANGE.
3. The application sends the correct GMW 3110 HS programming message sequence ($A5 $02, $A5 $03).
4. The SAE J2534-2 hardware interface monitors the ($A5 $02, $A5 $03) sequence and automatically switches in the load resistor, changes the SWCAN transceiver mode, and reconfigures the CAN controller. Note that this must be accomplished within 30 msec of the $A5 $03 frame. A SW_CAN_HS_RX indication will be generated upon completion of this transition.
5. Upon completion of the flash programming event, the application transmits the return to normal mode command (mode $20). A SW_CAN_NS_RX indication will be generated upon completion of this transition.
6. The SAE J2534-2 hardware interface responds to the return to normal mode command, which reconfigures the CAN controller, changes the transceiver mode, and disconnects the load resistor. Note that this must be accomplished within 30 msec of the transmission of the return to normal mode frame.

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

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

Assume all settings are in the Default mode:

1. The application sets the configuration of SW_CAN_SPEEDCHANGE_ENABLE to 1 – ENABLE_SPDCHANGE.
2. The SAE J2534-2 hardware interface monitors the ($A5 $02, $A5 $03) sequence and automatically changes the SWCAN transceiver mode, and reconfigures the CAN controller. (It does not switch in the load resistor) Note that this must be accomplished within 30 msec of the $A5 $03 frame. A SW_CAN_HS_RX indication will be generated upon completion of this transition.
3. Upon completion of the flash programming event, the SAE J2534-2 hardware interface receives the return to normal mode command (mode $20). A SW_CAN_NS_RX indication will be generated upon completion of this transition.
4. The SAE J2534-2 hardware interface responds to the return to normal mode command, which reconfigures the CAN controller and changes the transceiver mode. Note that this must be accomplished within 30 msec of the transmission of the return to normal mode frame.

Although the option of manual load resistor activation (1 – CONNECT_RESISTOR) is not shown in any of the above examples, this capability was included to accommodate special test applications that might require this feature.

Figure 10 summarizes the changes to the SAE J2534-1 API functions.

Function	Description of Change
PassThruConnect	Added two new ProtocolID values.
PassThruIoctl	Added GET_CONFIG and SET_CONFIG parameters. Added new IOCTLs to support SWCAN capability

FIGURE 10—SAE J2534 API FUNCTIONS

9.3.2 API FUNCTIONS – DETAILED INFORMATION

9.3.2.1 PassThruConnect

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

9.3.2.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in Figure 11. The actual value is defined in the section titled ‘SAE J2534-2 Resources’.

Definition	Description
SW_CAN_PS	Raw Single Wire CAN messages
SW_ISO15765_PS	Single Wire CAN adhering to ISO15765-2 flow control

FIGURE 11—PROTOCOLID DESCRIPTIONS

9.3.2.3 PassThruReadMsgs

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

9.3.3 IOCTL SECTION

Figure 12 provides the details of the IOCTLs available through PassThruIoctl function:

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

FIGURE 12—IOCTL DETAILS

9.3.3.1 GET_CONFIG

See SET_CONFIG and Figure 13 for more details.

Support three new parameters added to SET_CONFIG.

9.3.3.2 SET_CONFIG

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

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

Parameter	**Valid values for Parameter**	**Default Value (Decimal)**	**Description**
SW_CAN_HS_DATA_RATE	5 - 500000	83333	The data rate to be used in response to a call to SW_CAN_HS IOCTL
SW_CAN_SPEEDCHANGE_ENABLE	0 (DISABLE_SPDCHANGE) 1 (ENABLE_SPDCHANGE)	0	Control the behavior of the SAE J2534 device in response to speed change SWCAN messages 0 = Ignore all bus speed transition messages on the bus. 1 = Process transmitted and received bus speed transition messages as per GMW3110 Section 10.17.5.2.
SW_CAN_RES_SWITCH	0 (DISCONNECT_ RESISTOR) 1 (CONNECT_ RESISTOR) 2 (AUTO_ RESISTOR)	0	Control Load Resistor switching 0 = Default value. Disable automatic switching and disconnect load resistor. 1 = Disable automatic switching and connect load resistor. 2 = Automatically Switch in the load resistor when transitioning to high speed. (and switch off the load resistor while transitioning back to normal speed)

FIGURE 13—IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

9.3.3.3 SW_CAN_HS

The IoctlID value of SW_CAN_HS is used to initiate a transition of the SW CAN Bus to High Speed Mode. A successful transition will be noted by a SW_CAN_HS_RX indication. The speed transitioned to is the value of the SW_CAN_HS_DATA_RATE parameter. Parameter definition for SW_CAN_HS is defined in Figure 14.

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

FIGURE 14—SW_CAN_HS DETAIL

9.3.3.4 SW_CAN_NS

The IoctlID value of SW_CAN_NS is used to initiate a transition of the SWCAN Bus to Normal Speed Mode. A successful transition will be noted by a SW_CAN_NS_RX indication. The speed transitioned to is the value of the DATA_RATE parameter. Parameter definition for SW_CAN_NS is detailed in Figure 15.

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

FIGURE 15—SW_CAN_NS DETAILS

9.4 Message Structure

9.4.1 ELEMENTS

There is no change to any of the elements, but it should be noted that the *ProtocolID* element (either SW_ISO15765_PS or SW_CAN_PS) shall identify the message type. SAE J2534-1 defines requirements, restrictions, and error conditions for each protocol.

9.4.1.1 RxStatus

Definitions for RxStatus bits are defined in Figure 16:

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

FIGURE 16—RXSTATUS BIT DEFINITIONS

9.4.1.2 TxFlags

Figure 17 defines the TxFlags bit definition.

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

FIGURE 17—TXFLAGS BIT DEFINITIONS

10. Analog Inputs

10.1 Scope of the Analog Inputs Optional Feature

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

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

10.2 Pass-Thru System Requirements

10.2.1 ANALOG INPUTS

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

10.2.2 SIMULTANEOUS COMMUNICATION ON MULTIPLE PROTOCOLS

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

10.3 Win32 Application Programming Interface

10.3.1 API FUNCTIONS – OVERVIEW

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

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

Figure 18 summarizes the changes to the SAE J2534-1 API Functions.

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

FIGURE 18—SAE J2534 API FUNCTIONS

10.3.2 API Functions – Detailed Information

10.3.2.1 PassThruConnect

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

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

ProtocolIDs beyond those supported by the device shall return ERR_INVALID_PROTOCOL_ID.

10.3.2.2 ProtocolID Values

Only the definition and description of the ProtocolID value is defined in the following table. The actual value is defined in section 12 titled 'SAE J2534-2 Resources' Figure 42. Protocol values for the analog feature are identified in Figure 19.

Definition	Description
ANALOG_IN_1	Analog subsystem 1
ANALOG_IN_2	Analog subsystem 2
ANALOG_IN_3	Analog subsystem 3
…	…
ANALOG_IN_32	Analog subsystem 32

FIGURE 19—PROTOCOLID VALUES

10.3.2.3 PassThruReadMsgs

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

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

10.3.2.4 PassThruWriteMsgs

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

10.3.2.5 PassThruStartPeriodicMsg

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

10.3.2.6 PassThruStopPeriodicMsg

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

10.3.2.7 PassThruStartMsgFilter

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

10.3.2.8 PassThruStopMsgFilter

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

10.3.3 IOCTL SECTION

Each analog subsystem shall support 3 IOCTL functions: GET_CONFIG, SET_CONFIG and CLEAR_RX_BUFFER. The CLEAR_RX_BUFFER Ioctl shall remove any queued messages for the subsystem. All other Ioctl functions must return ERR_INVALID_CHANNEL_ID.

10.3.3.1 GET_CONFIG

There are several new parameters that are used to setup and control the A/D subsystem. See Figure 20 for more details.

10.3.3.2 SET_CONFIG

The following parameters control the analog subsystem. Note that there is no way to set parameters for each channel individually. All other parameters shall return ERR_INVALID_CHANNEL_ID. See Figure 20 for more details.

Parameter	Valid values for Parameter	Default (decimal)	Description
ACTIVE_CHANNELS	0 – 0xFFFFFFFF	Hardware dependent	Bitmask of channels being sampled
SAMPLE_RATE	0 – 0xFFFFFFFF	0	Samples/second or Seconds/sample
SAMPLES_PER_ READING	1 – 0xFFFFFFFF	1	Samples to average into a single reading
READINGS_PER_ MSG	1 – 0x00000408 (1 – 1032)	1	Number of readings for each active channel per PASSTHRU_MSG structure
AVERAGING_ METHOD	0 – 0xFFFFFFFF	0	The way in which the samples will be averaged.
SAMPLE_ RESOLUTION	0x1-0x20 (1 – 32)	Hardware dependent	The number of bits of resolution for each channel in the subsystem. Read only.
INPUT_RANGE_LOW	0x80000000 through 0x7FFFFFFF (-2147483648 through 2147483647)	Hardware dependent	Lower limit in millivolts of A/D input. (Example 0xFFFFB1E0 = -20.00V) Read only.
INPUT_RANGE_HIGH	0x80000000 through 0x7FFFFFFF (-2147483648 through 2147483647)	Hardware dependent	Upper limit in millivolts of A/D input. (Example 0x00004E20 = +20.00V) Read only.

FIGURE 20—IOCTL GET_CONFIG / SET_CONFIG PARAMETER DETAILS

10.3.3.2.1 ACTIVE_CHANNELS

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

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

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

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

10.3.3.2.2 SAMPLE_RATE

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

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

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

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

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

10.3.3.2.3 SAMPLES_PER_READING

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

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

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

The default value for SAMPLES_PER_READING is one.

10.3.3.2.4 READINGS_PER_MSG

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

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

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

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

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

The default value for READINGS_PER_MSG is one.

10.3.3.2.5 AVERAGING_METHOD

When SAMPLES_PER_READING is above one, each reading will consist of several samples. The AVERAGING_METHOD specifies how each reading will be computed. If the device does not support a particular value, ERR_INVALID_IOCTL_VALUE shall be returned. The default value (SIMPLE_AVERAGE) must be supported, even if the device does not support averaging. See Figure 21 for more details.

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

FIGURE 21—VALUES FOR THE AVERAGING_METHOD PARAMETER

10.3.3.2.5.1 SIMPLE_AVERAGE

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

$$\text{Reading} = (\text{Sample}_1 + \text{Sample}_2 + \ldots + \text{Sample}_{\text{SAMPLES\_PER\_READING}}) / \text{SAMPLES\_PER\_READING} \quad \text{(Eq. 1)}$$

10.3.3.2.5.2 MAX_LIMIT_AVERAGE

The MAX_LIMIT_AVERAGE simply chooses the maximum value.

$$\text{Reading} = \text{Max}(\text{Sample}_1 + \text{Sample}_2 + \ldots + \text{Sample}_{\text{SAMPLES\_PER\_READING}}) \quad \text{(Eq. 2)}$$

10.3.3.2.5.3 MIN_LIMIT_AVERAGE

The MIN_LIMIT_AVERAGE simply chooses the minimum value.

$$\text{Reading} = \text{Min}(\text{Sample}_1 + \text{Sample}_2 + \ldots + \text{Sample}_{\text{SAMPLES\_PER\_READING}}) \quad \text{(Eq. 3)}$$

10.3.3.2.5.4 MEDIAN_AVERAGE

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

$$\text{Reading} = \text{Sample}_{(\text{SAMPLES\_PER\_READING}+1)/2} \quad \text{(Eq. 4)}$$

Or if SAMPLES_PER_READING is even:

$$\text{Reading} = (\text{Sample}_{\text{SAMPLES\_PER\_READING}/2} + \text{Sample}_{(\text{SAMPLES\_PER\_READING}/2)+1}) / 2 \quad \text{(Eq. 5)}$$

10.3.3.2.5.5 Vendor-Specific Averaging Methods

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

10.3.3.2.6 SAMPLE_RESOLUTION

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

10.3.3.2.7 INPUT_RANGE_LOW

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

10.3.3.2.8 INPUT_RANGE_HIGH

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

10.3.3.3 FIVE_BAUD_INIT

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

10.3.3.4 FAST_INIT

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

10.3.3.5 CLEAR_TX_BUFFER

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

10.3.3.6 CLEAR_RX_BUFFER

The device shall remove any queued messages for the subsystem.

10.3.3.7 CLEAR_PERIODIC_MSGS

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

10.3.3.8 CLEAR_MSG_FILTERS

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

10.3.3.9 CLEAR_FUNCT_MSG_LOOKUP_TABLE

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

10.3.3.10 ADD_TO_FUNCT_MSG_LOOKUP_TABLE

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

10.3.3.11 DELETE_FROM_FUNCT_MSG_LOOKUP_TABLE

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

10.4 Message Structure

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

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

10.4.1 EXAMPLES:

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

Figures 22 through 27 provide examples of different parameters and the resulting structures:

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

FIGURE 22—SAMPLE A/D PARAMETER CONFIGURATION

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

Channel 1 Sample 1	Channel 2 Sample 1	Channel 3 Sample 1	Channel 4 Sample 1

FIGURE 23—DATA FROM ANALOG_IN_1 SUBSYSTEM WITH READINGS_PER_MSG = 1

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

Channel 1 Sample 1	Channel 2 Sample 1

FIGURE 24—DATA FROM ANALOG_IN_2 SUBSYSTEM WITH READINGS_PER_MSG = 1

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

Channel 1 Sample 1	Channel 2 Sample 1	Channel 3 Sample 1	Channel 4 Sample 1	Channel 1 Sample 2	Channel 2 Sample 2	Channel 3 Sample 2	Channel 4 Sample 2

FIGURE 25—DATA FROM ANALOG_IN_1 SUBSYSTEM WITH READINGS_PER_MSG = 2

The second subsystem now generates this message every 10 seconds:

Channel 1 Sample 1	Channel 2 Sample 1	Channel 1 Sample 2	Channel 2 Sample 2

FIGURE 26—DATA FROM ANALOG_IN_2 SUBSYSTEM WITH READINGS_PER_MSG = 2

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

Channel 1 Sample 1	Channel 2 Sample 1	Channel 4 Sample 1	Channel 1 Sample 2	Channel 2 Sample 2	Channel 4 Sample 2

FIGURE 27—DATA FROM ANALOG_IN_1 SUBSYSTEM WITH ACTIVE_CHANNELS = 0XB

10.4.2 Message Flag and Status Definitions

Definitions for RxStatus bits are shown in Figure 28.

Definition	RxStatus Bit(s)	Description	Value
OVERFLOW	16	Indicates that the input range of the A/D has been exceeded	0 = All samples good 1 = Some samples clipped

FIGURE 28—RXSTATUS BIT DEFINITIONS

11. GM UART (SAE J2740)

11.1 Scope of the GM UART Optional Feature

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

11.2 Pass-Thru System Requirements

11.2.1 Pin Usage

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

Most GM vehicles with serial data links built prior to the 1996 model year are equipped with a 12 pin connector as shown in Figure 29. The mating tool connector is shown in Figure 30. For programming these older vehicles using an SAE J2534 interface, a 12 pin connector must be available instead of an SAE J1962 connector to interface to the vehicle. The signal ground, pin 5 on an SAE J1962 connector, must be connected to pin A of the 12 pin connector. The serial data line, pin 9 on an SAE J1962 connector, must be connected to pin M of the 12 pin connector. The 12 pin connector does not contain battery power, so the SAE J2534 interface cannot be powered from the 12 pin connector.

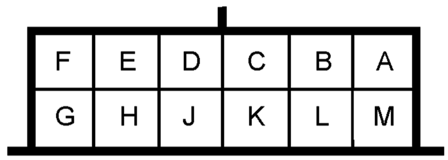

FIGURE 29—12 PIN VEHICLE CONNECTOR

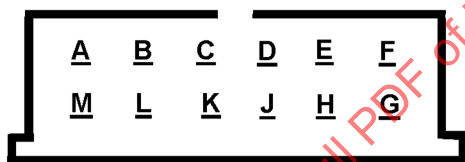

FIGURE 30—12 PIN TOOL CONNECTOR

11.3 Win32 Application Programming Interface

11.3.1 API Functions – Overview

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

This document outlines the requirements for providing a single interface channel supporting this protocol.

At a high level a new ProtocolID has been defined to indicate the use of the GM_UART_PS physical layer. The protocol defines a master/slave relationship between the Tester and the Electronic Control Unit (ECU). The tester must request and be granted mastership over the vehicle bus before communications can begin. This document details the API resources required to enable use of the SAE J2534 API as applied to GM_UART_PS protocol. If the feature is not supported, an error code, ERR_INVALID_PROTOCOL_ID, will be returned by the call to PassThruConnect. The calling application will be required to notify the user that this optional feature may not be supported by interface.

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

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

Figure 31 summarizes the changes to the SAE J2534-1 API functions.

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

FIGURE 31—SAE J2534 API FUNCTIONS

11.3.2 API Functions – Detailed Information

11.3.2.1 PassThruConnect

Added GM_UART_PS Protocol ID.

11.3.2.1.1 ProtocolID Values (GM UART)

One additional ProtocolID Value has been defined. Only the definition and description of the ProtocolID value is defined in Figure 32. The actual value is defined in the section titled ‘SAE J2534-2 Resources’.

Definition	Description
GM_UART_PS	GM UART Protocol

FIGURE 32—PROTOCOLID VALUES