

***AutoEnginuity***  
***ProLine Hardware***  
***User Guide***

Copyright © 2009 AutoEnginuity, L.L.C. All rights reserved

The source code described in this document is furnished under license agreement. The source code may be used or copied only in accordance with the terms of the agreement. It is against the law to copy source code except as specifically allowed in the license agreement.

This material is copyright and any unauthorized reproduction or transmission in any form or by any means, electronic or mechanical, including photocopying and recording, in whole or in part, without written consent of AutoEnginuity, L.L.C., is expressly prohibited. Any unauthorized reproduction or transmission is illegal and a violation of Title §17 of the United States Code. Civil and criminal punishments can and will be applied to the maximum extent of the law.

### **Manufacturer Specific Data Stream**

This document does not contain any vehicle manufacturer specific data stream or proprietary instructions. Nor can AutoEnginuity provide this information or answer questions regarding its usage or implementation. Any and all information used to develop and design the AutoEnginuity ScanTool range of products is licensed to AutoEnginuity exclusively and is not available for sublicense; nor, is AutoEnginuity available to answer questions in regards to any data stream licensed by others. Do *NOT* contact AutoEnginuity support staff, or any of its affiliates, for manufacture specific data stream information. That information is to be provided by the user of this document and is made available from the respective vehicle manufacturers engineering departments.

### **Limitation of Liability**

The material and information contained within this User Guide is subject to change without notice. AutoEnginuity, L.L.C., makes no warranty, express or implied, and shall not be liable for any errors or for incidental or consequential damages in connection with the use of the instructions or examples herein.

### **Trademark References**

AutoEnginuity are trademarks of AutoEnginuity, L.L.C. All other trademarks and registered trademarks are proprietary to their respective manufacturers.

AutoEnginuity, L.L.C.  
Mesa, AZ  
1-480-827-TOOL  
[www.autoenginuity.com](http://www.autoenginuity.com)

# Table of Contents

<b>REVISION HISTORY</b> .....	<b>5</b>
<b>COMMAND PACKET INFORMATION</b> .....	<b>6</b>
<b>COMMAND LAYOUT</b> .....	<b>8</b>
*** GENERAL INTERFACE COMMANDS *** .....	8
Command \$01: Request Connector ID .....	8
Command \$02: Firmware Version .....	8
Command \$03: Request Serial Number .....	8
Command \$04: Supported Interfaces.....	8
Command \$05: Autodetect Interface .....	9
Command \$09: Connector Name.....	9
Command \$0A: Retrieve Pin16 Voltage .....	9
Command \$0B: Retrieve Buffered Result (NOTE: USB Devices only).....	10
Command \$0D: Retrieve Next Long Result (NOTE: USB Devices only).....	10
Command \$0F: Soft Reset Connector.....	10
*** J1850VPW COMMUNICATION INTERFACE COMMANDS *** .....	11
Command \$10: VPW Message Single Response.....	11
Command \$11: VPW Message Multiple Response .....	11
*** J1850PWM COMMUNICATION INTERFACE COMMANDS *** .....	12
Command \$20: PWM Message Single Response.....	12
Command \$21: PWM Message Multiple Response .....	12
*** ISO COMMUNICATION INTERFACE COMMANDS *** .....	13
Command \$30: ISO Message Single Response.....	13
Command \$31: ISO Message Multiple Response .....	13
Command \$32: ISO Send Init .....	13
Command \$33: Set ISO Baud Rate Speed and Response Time.....	14
Command \$34: Send ISO KWP1281 Single Response.....	15
Command \$35: Set ISO Pin output .....	15
Command \$36: Retrieve Current ISO Baud Rate .....	15
Command \$38: Send ISO Long Busy Response.....	16
*** CAN COMMUNICATION INTERFACE COMMANDS *** .....	17
Command \$40: Send 11-Bit SID CAN Message Single Response .....	17
Command \$41: Send 11-Bit SID CAN Message Multiple Response.....	17
Command \$42: Send 11-Bit SID CAN Message With Flow Control .....	17
Command \$43: Send 29-Bit CAN Message Single Response.....	18
Command \$44: Send 29-Bit CAN Message Multiple Response .....	18
Command \$45: Send 29-Bit SID CAN Message With Flow Control .....	19
Command \$46: Set CAN Bus Speed and Response Time.....	19
Command \$47: Send CAN Porsche Single Response .....	20
Command \$48: Send 11-Bit SID CAN No Message Response.....	20
Command \$49: Send BMW DCAN Message Multiple Response .....	20
Command \$4A: Enable CAN Device .....	21
Command \$4B: Set Transmission Data Padding.....	21
Command \$4C: Send CAN TP2.0 Enable Heart Beat .....	21
Command \$4D: Send CAN TP2.0 No Change.....	22
Command \$4E: Send CAN TP2.0 Disable Heart Beat .....	22
Command \$4F: Send CAN TP2.0 with Long Delay No Change.....	22
*** CHRYSLER SCI COMMUNICATION INTERFACE COMMANDS *** .....	24
Command \$50: SCI Engine Low Speed Message .....	24
Command \$51: SCI Engine High Speed Mode 1 Message .....	24
Command \$52: Send Engine Speed Init Command .....	24
Command \$53: Send Engine J2190 Command.....	25
*** SPECIAL COMMUNICATION INTERFACE COMMANDS *** .....	25

<i>Command \$5A: PWM Signal Generator.....</i>	<i>25</i>
<i>Command \$5B: H-Diag Message Single Response .....</i>	<i>25</i>
<i>Command \$5C: Nissan Consult DDL Message Normal Response.....</i>	<i>25</i>
<i>Command \$5D: Nissan Consult DDL Message Long Response.....</i>	<i>26</i>
<i>Command \$5E: Nissan Consult DDL Set Response Timeout .....</i>	<i>26</i>

# Revision History

11/09/2009 Initial Release

# Command Packet Information

**Communication:** The connector is uses USB hardware to connect. The USB device has been implemented as a HID device with a block size of 64 bytes.

**HID Vendor ID:** 0x1AA0  
**HID Product ID:** 0x0003

**Connector ID:** For the sake of security all packets must be started with the connectors ID, or Vendor ID, to accept an incoming packet. If the incoming packet does not have the correct connector ID an Invalid Connector ID error will be returned. The connector ID does not count towards the over all packet size of the sent packet, it's an assumed BYTE. Below we state the smallest packet possible is 4 bytes, but this does not include this BYTE, so the overall smallest data sent would be 5 bytes. The useable ID is \$FF.

**NOTE:** For commercial purposes an ID can be generated for your use if the purchase quantity is large enough. Please contact AutoEnginuity Engineering for more details.

**Packet:** The AutoEnginuity connector is based on a variable sized packet. A single packet must contain the command byte (1 byte), size of packet (2 bytes), and the last byte in the packet is the checksum. Therefore the smallest packet possible is 4 bytes. The largest packet size is 256 bytes.

**Packet Size:** The two bytes used for the packet size also contains a flag that can be used to determine if more responses are available from the connector when performing a command that does multiple responses. The high bit on the two byte packet size is used to determine this.

**NOTE:** Remember to clear the flag before calculating the packet size otherwise you will have created an invalid packet size.

**Check Sum:** The check sum is created by adding up all of the bytes, not including the check sum it self.

**Wait Times:** The wait times listed for the commands are based on default times. Some wait times can be over written at which time the listed wait times don't have any meaning.

**Error Responses:** The connector will generate an error packet if an error has occurred.

**Response:** \$B0 \$00 \$05 XX CS

The XX is the error code; below is a table containing the error code descriptions:

Code	Description
\$01	Invalid CRC
\$02	Unknown command
\$03	Invalid Connector ID
\$10	CAN Bus error
\$11	CAN receive error count greater than 127
\$12	CAN transmit error count greater than 128
\$13	VPW Lost Arbitration
\$14	PWM Lost Arbitration
\$15	CCD Lost Arbitration

\$20	CAN no response
\$21	VPW no response
\$22	PWM no response
\$23	ISO no response
\$24	SCI no response
\$25	ALDL no Response
\$26	DDL no Response
\$27	HDIAG no Response
\$28	CCD no Response
\$30	Invalid ISO Check Sum
\$31	Invalid VPW Check Sum
\$32	Invalid PWM Check Sum

# Command Layout

## \*\*\* General Interface Commands \*\*\*

### Command \$01: Request Connector ID

This command is used to request the 32 bit connector id. This is defined as the vender ID for the connector and can be used by the software to determine if this is the correct connector to work with.

**Wait Timeout:** Wait 10ms for a response

**Request:** \$01 \$00 \$04 \$05

**Response:** \$A1 \$00 \$05 XX XX CS

#### Example for ID 1:

**Example Request:** \$01 \$00 \$04 \$05

**Example Response:** \$A1 \$00 \$05 \$00 \$01 \$A7

### Command \$02: Firmware Version

This command is used to request the current onboard firmware version.

**Wait Timeout:** Wait 10ms for a response

**Request:** \$02 \$00 \$04 \$06

**Response:** \$A1 \$00 \$04 XX CS

#### Example for version 19:

**Example Request:** \$02 \$00 \$04 \$06

**Example Response:** \$A1 \$00 \$04 \$13 \$B8

### Command \$03: Request Serial Number

This command is used to retrieve the connector's 8 byte serial number.

**Wait Timeout:** Wait 10ms for a response

**Request:** \$03 \$00 \$04 \$07

**Response:** \$A3 \$00 \$0C XX XX XX XX XX XX XX CS

#### Example of serial number 2012:

**Example Request:** \$03 \$00 \$04 \$07

**Example Response:** \$A3 \$00 \$0C \$00 \$00 \$00 \$00 \$00 \$00 \$07 \$DC \$92

### Command \$04: Supported Interfaces

This command is used to check to see what interfaces the connector supports.

**Wait Timeout:** Wait 10ms for a response

**Request:** \$04 \$00 \$04 \$08

**Response:** \$A4 \$00 \$05 XX CS

XX is a bit mask

Bit 0 = VPW  
Bit 1 = PWM  
Bit 2 = 11bit CAN Dual Wire  
Bit 3 = ISO9141-2  
Bit 4 = KWP2000 (ISO14230-4)  
Bit 5 = Chrysler SCI  
Bit 6 = 29bit CAN Dual Wire

**Example of Supporting VPW, PWM, ISO9141-2 Chrysler SCI:**

**Example Request:** \$04 \$00 \$04 \$08

**Example Response:** \$A4 \$00 \$05 \$2B \$D4

## **Command \$05: Autodetect Interface**

This command will automatically check the interface on the connected vehicle and return the interface that is supported. This command will return the first OBDII PID request response by the given interface. The detection sequence is as follows: CAN 11-Bit, PWM, VPW, KWP2000, and then ISO9141-2.

**Wait Timeout:** Can wait up to 20s for a response.

**Request:** \$05 \$00 \$04 \$09

**Response:** \$A5 \$00 \$05 XX CS

**XX is a bit mask**

bit 0 = VPW  
bit 1 = PWM  
bit 2 = DW-CAN 11bit  
bit 3 = ISO9141-2  
bit 4 = KWP2000 (ISO14230-4)  
bit 5 = DW-CAN 29bit

**Example of a detecting KWP2000:**

**Example Request:** \$05 \$00 \$04 \$09

**Example Response:** \$A5 \$00 \$05 \$10 \$BA

## **Command \$09: Connector Name**

This command is used to return the name of the connector, AutoEnginuity.

**Wait Timeout:** Wait 20ms for a response

**Request:** \$09 \$00 \$04 \$0C

**Response:** \$A9 \$00 YY XX ... XX CS

YY = Packet Size

XX = Name

**Example:**

**Example Request:** \$09 \$00 \$04 \$0C

**Example Response:** \$A9 \$00 \$11 'A' 'U' 'T' 'O' 'E' 'N' 'G' 'I' 'N' 'U' 'I' 'T' 'Y' CS

## **Command \$0A: Retrieve Pin16 Voltage**

This command takes the onboard ADC and samples the voltage coming from PIN16. The conversion rate is 0.00732600.

**Wait Timeout:** Can wait up to 10ms for a response.

**Request:** \$0A \$00 \$04 \$14  
**Response:** \$A1 \$00 \$06 XX XX CS  
XX = ADC value

**Example of a detecting 12.1V:**

**Example Request:** \$0A \$00 \$04 \$14  
**Example Response:** \$A1 \$00 \$06 \$06 \$73 \$20

**Command \$0B: Retrieve Buffered Result (NOTE: USB Devices only)**

This command is called to retrieve stored packets in the buffer. This is used when a request caused multiple response packets. The first packet is returned, however all other packets are required to response from this command. This command is sent with a buffer index and the response is the actual packet response from the initial requested interface protocol request.

**Wait Timeout:** 20ms

**Request:** \$0B \$00 \$05 XX CS  
XX = The buffer index to return. (NOTE this is a zero based index)

**Response:** What ever data is stored in the buffer

**Command \$0D: Retrieve Next Long Result (NOTE: USB Devices only)**

This command is called to retrieve stored packets in the buffer that exceeded the actual USB buffer size, 64 bytes.

**Wait Timeout:** 20ms

**Request:** \$0D \$00 \$05 XX CS  
XX = The next 64 bytes to return. (NOTE this is a zero based index)

**Response:** What ever data is stored in the buffer

**Command \$0F: Soft Reset Connector**

This command causes the connector to perform a soft hardware reset which will cause all of the communication settings to resort back to their default settings. This command has no response message since the connector is performing a reset.

**Request:** \$0F \$00 \$04 \$12

### **\*\*\* J1850VPW Communication Interface Commands \*\*\***

#### **Command \$10: VPW Message Single Response**

This command sends a message over the VPW bus and will wait for a single response.

**Wait Timeout:** Wait 100ms for a response

**Request:** \$10 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the VPW bus

**Response:** \$B2 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the VPW bus.

#### **Example of a generic PID request:**

**Example Request:** \$10 \$00 \$09 \$68 \$6A \$F1 \$01 \$14 \$F1

**Example Response:** \$B2 \$00 \$0C \$48 \$6B \$10 \$41 \$14 \$5E \$80 \$D0 \$84

#### **Command \$11: VPW Message Multiple Response**

This command sends a message over the VPW bus and will return all responses.

**Wait Timeout:** Wait 100ms for a response till no more responses are available

**Request:** \$11 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the VPW bus

**Response:** \$B2 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the VPW bus.

#### **Example of a generic PID request:**

**Example Request:** \$11 \$00 \$09 \$68 \$6A \$F1 \$01 \$14 \$F2

**Example Response:** \$B2 \$00 \$0C \$48 \$6B \$10 \$41 \$14 \$5E \$80 \$D0 \$84

### **\*\*\* J1850PWM Communication Interface Commands \*\*\***

#### **Command \$20: PWM Message Single Response**

This command sends a message over the PWM bus and will wait for a single response.

**Wait Timeout:** Wait 100ms for a response

**Request:** \$20 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the PWM bus

**Response:** \$B3 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the PWM bus.

#### **Example of a generic PID request:**

**Example Request:** \$20 \$00 \$09 \$61 \$6A \$F1 \$01 \$14 \$FA

**Example Response:** \$B3 \$00 \$0C \$41 \$6B \$10 \$41 \$14 \$5E \$80 \$D0 \$7E

#### **Command \$21: PWM Message Multiple Response**

This command sends a message over the PWM bus and will return all responses.

**Wait Timeout:** Wait 100ms for a response till no more responses are available

**Request:** \$21 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the PWM bus

**Response:** \$B3 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the PWM bus.

#### **Example of a generic PID request:**

**Example Request:** \$21 \$00 \$09 \$61 \$6A \$F1 \$01 \$14 \$FB

**Example Response:** \$B3 \$00 \$0C \$41 \$6B \$10 \$41 \$14 \$5E \$80 \$D0 \$7E

### **\*\*\* ISO Communication Interface Commands \*\*\***

#### **Command \$30: ISO Message Single Response**

This command sends a message over the ISO bus and will wait for a single response.

**Wait Timeout:** We wait for up to 4 seconds for an initial response

**Request:** \$30 YY YY XX ... XX C

YY is the size of the packet

XX .. XX is the data to be sent on the ISO bus

**Response:** \$B4 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the ISO bus.

##### **Example of a generic PID request:**

**Example Request:** \$30 \$00 \$09 \$68 \$6A \$F1 \$01 \$14 \$11

**Example Response:** \$B4 \$00 \$0C \$48 \$6B \$10 \$41 \$14 \$5E \$76 \$D0 \$7C

#### **Command \$31: ISO Message Multiple Response**

This command sends a message over the ISO bus and will return all responses.

**Wait Timeout:** We wait up to 4 seconds for an initial response then 50ms wait for other responses.

**Request:** \$31 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the ISO bus

**Response:** \$B4 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the ISO bus.

##### **Example of a generic PID request:**

**Example Request:** \$31 \$00 \$09 \$68 \$6A \$F1 \$01 \$14 \$12

**Example Response:** \$B4 \$00 \$0C \$48 \$6B \$10 \$41 \$14 \$5E \$76 \$D0 \$7C

#### **Command \$32: ISO Send Init**

This command sends an init command over the L-Line. You can send a fast or slow init and even specify the byte init to send.

**Slow Init** - After the init is sent there is a 4sec timeout for a response then there is a 50ms delay between each byte response.

**Fast Init** - There is a 400ms delay before we wait for a response and then there is 50ms delay between each byte response.

**Request:** \$32 YY YY XX ZZ...ZZ CS

YY = Packet size [Slow Init = \$05 / Fast Init = \$08]

XX = \$00 Slow initialization

XX = \$01 Fast initialization

XX = \$02 Honda H/99 initialization

XX = \$03 KWP1281 initialization

XX = \$04 MB Slow initialization

XX = \$05 Toyota Phase 3 initialization  
XX = \$06 Sprinter Fast initialization  
ZZ is the initialization command to send.  
Slow Init = \$33 for Emission controllers  
Fast Init = \$XX???

**Response:** \$C3 XX XX YY...YY CS

XX = Packet size [Slow Init = \$05 / Fast Init = \$10]

YY = \$00 Failed [Slow Init Response / Fast Init response if failure]

YY = \$01 Succeeded [Slow Init Response]

#### Example of performing a Slow Init

**Example Request:** \$32 \$00 \$06 \$00 \$33 \$70

**Example Response:** \$C3 \$00 \$05 \$01 \$C9

#### Example of performing a Fast Init

**Example Request:** \$32 \$00 \$09 \$01 \$C1 \$33 \$F1 \$81 \$A5

**Example Response:** \$C3 \$00 \$10 \$83 \$F1 \$11 \$C1 \$EF \$8F \$C3 \$55

**Example Failure:** \$C3 \$00 \$05 \$00 \$C8

## Command \$33: Set ISO Baud Rate Speed and Response Time

This command sets the baud rate speed at which the ISO module will operate at.

**Wait Timeout:** 25ms

**Request:** \$33 \$00 \$0B XX YY WW ZZ AA AA BB CS

XX = \$00 9600 bps

XX = \$01 9615 bps

XX = \$03 10.4k bps **[Default Baud rate upon reset]**

XX = \$0A 15.625k bps

XX = \$0B 100k bps

XX = \$0C 4800 bps

XX = \$0D 10.416k bps

XX = \$0E 2400 bps

XX = \$0F 9800 bps

XX = \$FF No Speed Change

YY = RX Inter-byte Delay in ms [\$00 is default timeout, 50ms]

WW = TX Inter-byte Delay in ms [\$00 is default timeout, 50ms]

ZZ = Serial Settings

Bit 1 - Enable Parity

Bit 2 - Parity Type

True = Odd Parity / False = Even Parity

Bit 3 - Append CRC

AA = Response Timeout Delay in ms [\$00 is default timeout, 50ms]

BB = \$01 Use TX inter-byte delay in request

\$00 Do not use the TX inter-byte delay in the request

**Response:** \$C7 \$00 \$05 XX CS

YY = \$00 Failed to change speed

YY = \$01 Succeeded to change speed

#### Example for setting the speed to 9600 bps, 6 ms RX/TX inter-byte delay

**Example Request:** \$33 \$00 \$0B \$00 \$06 \$06 \$00 \$00 \$01 \$38

**Example Response:** \$C7 \$00 \$05 \$01 \$CC

## Command \$34: Send ISO KWP1281 Single Response

This command is for Honda ABS Type 2 systems. This command sends a message over the ISO bus and will wait for a single response.

**Wait Timeout:** We wait for 4 seconds for an initial response

**Request:** \$34 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the ISO bus

**Response:** \$B4 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the ISO bus.

### Example:

**Example Request:** \$34 \$00 \$07 \$03 \$08 \$07 \$03 \$52

**Example Response:** \$B4 \$00 \$07 \$06 \$09 \$FC \$FF \$FF \$88 \$03 \$4F

## Command \$35: Set ISO Pin output

This command sets at what OBDII pin we are to transmit on

**Wait Timeout:** 25ms

**Request:** \$35 00 05 XX CS

XX = 1 - OBDII Pin 1

XX = 3 - OBDII Pin 3

XX = 7 - OBDII Pin 7 [Default upon reset]

XX = 8 - OBDII Pin 8

XX = 9 - OBDII Pin 9

XX = 10 - OBDII Pin 10

XX = 11 - OBDII Pin 11

XX = 12 - OBDII Pin 12

XX = 13 - OBDII Pin 13

XX = 15 - OBDII Pin 15

**Response:** \$CC 00 05 XX CS

XX = 0 Failure, Invalid pin request

XX = 1 Success, Valid pin request

### Example for setting pin 10

**Example Request:** \$35 00 05 \$0A \$44

**Example Response:** \$CC \$00 \$05 \$01 \$D2

## Command \$36: Retrieve Current ISO Baud Rate

This retrieves the current baud rate.

**Wait Timeout:** 25ms

**Request:** \$36 00 04 3A

**Response:** \$CF 00 05 XX XX XX XX CS

XX = Actual Current Baud Rate

### Example for 10.416k baud rate

**Example Request:** \$36 \$00 \$04 \$3A

**Example Response:** \$CF \$00 \$95 \$00 \$00 \$28 \$B0 \$3C

## **Command \$38: Send ISO Long Busy Response**

This command sends a message over the ISO bus and will return all responses. This will wait an extended amount of time for responses on the bus.

**Wait Timeout:** 200ms

**Request:** \$38 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the ISO bus

**Response:** \$B4 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data received on the ISO bus.

### **Example of a generic PID request:**

**Example Request:** \$38 \$00 \$09 \$68 \$6A \$F1 \$01 \$14 \$19

**Example Response:** \$B4 \$00 \$0C \$48 \$6B \$10 \$41 \$14 \$5E \$76 \$D0 \$7C

### **\*\*\* CAN Communication Interface Commands \*\*\***

#### **Command \$40: Send 11-Bit SID CAN Message Single Response**

This command sends a message over the CAN bus using 11-bit SID address size and will wait for a single response. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms to make sure that all CAN controllers have responded

**Request:** \$40 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

#### **Example of clearing DTCs**

**Example Request:** \$40 \$00 \$0B \$07 \$E0 \$07 \$E8 \$01 \$04 \$26

**Example Response:** \$B1 \$00 \$0E \$07 \$E8 \$01 \$44 \$F3

#### **Command \$41: Send 11-Bit SID CAN Message Multiple Response**

This command sends a message over the CAN bus using 11-bit SID address size and will wait for multiple responses. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms after ever response to make sure that all CAN controllers have responded.

**Request:** \$41 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

#### **Example of a generic PID request**

**Example Request:** \$41 \$00 \$0B \$07 \$E0 \$07 \$E8 \$02 \$01 \$00 \$25

**Example Response:** \$B1 \$00 \$0E \$07 \$E8 \$06 \$41 \$00 \$BF \$BF \$B9 \$93 \$00 \$BF

#### **Command \$42: Send 11-Bit SID CAN Message With Flow Control**

This command sends a message over the CAN bus using 11-bit SID with Flow Control. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms after ever response to make sure that all CAN controllers have responded.

**Request:** \$42 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

#### Example of a generic PID request

**Example Request:** \$42 \$00 \$0B \$07 \$E0 \$07 \$E8 \$02 \$01 \$00 \$25

**Example Response:** \$B1 \$00 \$0E \$07 \$E8 \$06 \$41 \$00 \$BF \$BF \$B9 \$93 \$00 \$BF

### **Command \$43: Send 29-Bit CAN Message Single Response**

This command sends a message over the CAN bus using 29-bit SID address size and will wait for a single response. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms to make sure that all CAN controllers have responded

**Request:** \$43 SS SS XX XX XX XX YY YY YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 29-bit SID address to send too

YY is the 29-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

#### Example of a generic PID request

**Example Request:** \$43 \$00 \$0E \$18 \$DB \$33 \$F1 \$18 \$DA \$F1 \$00 \$01 \$01 \$4C

**Example Response:** \$B1 \$00 \$0B \$18 \$DA \$F1 \$10 \$01 \$01 \$AE \$5F

### **Command \$44: Send 29-Bit CAN Message Multiple Response**

This command sends a message over the CAN bus using 29-bit SID address size and will wait for multiple responses. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms to make sure that all CAN controllers have responded

**Request:** \$44 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 29-bit SID address to send too

YY is the 29-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

### Example of a generic PID request

**Example Request:** \$44 \$00 \$0E \$18 \$DB \$33 \$F1 \$18 \$DA \$F1 \$00 \$01 \$01 \$4D

**Example Response:** \$B1 \$00 \$0B \$18 \$DA \$F1 \$10 \$01 \$01 \$AE \$5F

## **Command \$45: Send 29-Bit SID CAN Message With Flow Control**

This command sends a message over the CAN bus using 29-bit SID and will wait for multiple responses. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms after every response to make sure that all CAN controllers have responded.

**Request:** \$45 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 29-bit SID address to send too

YY is the 29-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

### Example of a generic PID request

**Example Request:** \$45 \$00 \$0E \$18 \$DB \$33 \$F1 \$18 \$DA \$F1 \$00 \$01 \$01 \$4E

**Example Response:** \$B1 \$00 \$0B \$18 \$DA \$F1 \$10 \$01 \$01 \$AE \$5F

## **Command \$46: Set CAN Bus Speed and Response Time**

This command sets the Bus speed at which the CAN module will operate at and also the timeout value for responses. The default value is 100ms. The value sent is x10ms. So for example if you send it 10 then it's for 100ms, if what is sent is 30 then the timeout is for 300ms. The time out counter is used to extend the overall delay response. So if you say the counter is 3 then the total delay time will be the response timeout times 3. The handle long relay lets the connector know if you want to use the timeout counter for all responses or just for the first response.

**Wait Timeout:** 25ms

**Request:** \$46 \$00 \$08 YY XX ZZ WW CS

XX = \$00 500k bps [Default speed after reset]

XX = \$01 250k bps

XX = \$02 125k bps

XX = \$03 95.2k bps

XX = \$04 83.33k bps

XX = \$05 33.33k bps

YY = response timeout x10ms

ZZ = timeout counter

WW = \$01 handle long delay responses

WW = \$00 do not handle long delay responses

**Response:** \$C4 \$00 \$05 XX CS

YY = \$00 Failed to change speed

YY = \$01 Succeeded to change speed

### Example for setting the speed to 250k bps

**Example Request:** \$46 \$00 \$05 \$01 \$4A  
**Example Response:** \$C4 \$00 \$05 \$01 \$CA

## Command \$47: Send CAN Porsche Single Response

This command sends a message via the CAN bus in the CAN TP2.0 version which a Porsche Cayenne requires.

**Wait Timeout:** 100ms

**Request:** \$47 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

## Command \$48: Send 11-Bit SID CAN No Message Response

This command sends a message over the CAN bus using 11-bit SID address size and will not wait for a single response. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 25ms

**Request:** \$48 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

### Example of clearing DTCs

**Example Request:** \$48 \$00 \$0B \$07 \$E0 \$07 \$E8 \$01 \$04 \$95

## Command \$49: Send BMW DCAN Message Multiple Response

This command sends a message over the CAN bus using 11-bit SID address size and will not wait for a single response. The max CAN message is 8 bytes making the largest packet size for this command to be 13 bytes.

**Wait Timeout:** Wait for 125 ms after ever response to make sure that all CAN controllers have responded.

**Request:** \$48 SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is the 11-bit SID address to receive from

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

## Command \$4A: Enable CAN Device

This command enables the specific CAN device specified in the command. Multiple CAN buses are usually found on vehicles running at different baud rates and found on different DLC pins.

**Wait Timeout:** 25 ms to send message and enter into monitor mode

**Request:** \$4A SS SS KK CS

SS is the size of the packet

KK = \$00 HSCAN (Pins 6 & 14) [Default after reset]

KK = \$01 MSCAN (Pins 3 & 11)

KK = \$02 SWCAN (Pin 1)

**Response:** \$CA \$00 \$04 \$CE

### Example

**Example Request:** \$4A \$00 \$05 \$00 \$4F

**Example Response:** \$CA \$00 \$04 \$CE

## Command \$4B: Set Transmission Data Padding

This command sets up whether the transmission of CAN data will be padded to 8 bytes no matter what the transmission size is.

**Wait Timeout:** 25 ms to send message and enter into monitor mode

**Request:** \$4B \$00 \$05 XX CS

XX = \$00 Turn Padding Off

XX = \$01 Turn Padding On [default]

**Response:** \$CD \$00 \$04 \$D1

### Example turning padding Off

**Example Request:** \$4B \$00 \$05 \$00 \$50

**Example Response:** \$CD \$00 \$04 \$D1

## Command \$4C: Send CAN TP2.0 Enable Heart Beat

This functions sends data onto the can bus with the CAN TP2.0 protocol and will generate an automatic heart beat.

**Wait Timeout:** 50 ms to send message and enter into monitor mode

**Request:** \$4C SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is unused, fill space to conform to other CAN message commands

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

### Example

**Example Request:** \$4C \$00 \$0C \$07 \$51 \$03 \$00 \$00 \$00 \$02 \$10 \$89 \$4E

**Example Response:** \$B1 \$00 \$0A \$03 \$00 \$11 \$00 \$02 \$50 \$89 \$AA

## Command \$4D: Send CAN TP2.0 No Change

This functions sends data onto the can bus with the CAN TP2.0 protocol and will not change the heart beat status.

**Wait Timeout:** 50 ms to send message and enter into monitor mode

**Request:** \$4D SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is unused, fill space to conform to other CAN message commands

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

### Example

**Example Request:** \$4D \$00 \$0C \$07 \$51 \$03 \$00 \$00 \$00 \$02 \$10 \$89 \$4F

**Example Response:** \$B1 \$00 \$0A \$03 \$00 \$11 \$00 \$02 \$50 \$89 \$AA

## Command \$4E: Send CAN TP2.0 Disable Heart Beat

This functions sends data onto the can bus with the CAN TP2.0 protocol and will disable the automatic heart beat.

**Wait Timeout:** 50 ms to send message and enter into monitor mode

**Request:** \$4E SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is unused, fill space to conform to other CAN message commands

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

### Example

**Example Request:** \$4E \$00 \$0C \$07 \$51 \$03 \$00 \$00 \$00 \$02 \$10 \$89 \$50

**Example Response:** \$B1 \$00 \$0A \$03 \$00 \$11 \$00 \$02 \$50 \$89 \$AA

## Command \$4F: Send CAN TP2.0 with Long Delay No Change

This functions sends data onto the can bus with the CAN TP2.0 protocol and will not change the heart beat status.

**Wait Timeout:** 500 ms to send message and enter into monitor mode

**Request:** \$4F SS SS XX XX YY YY ZZ .. ZZ CS

SS is the size of the packet

XX is the 11-bit SID address to send too

YY is unused, fill space to conform to other CAN message commands

ZZ .. ZZ is the data to be sent on the CAN bus

**Response:** \$B1 \$00 \$0E YY YY XX ... XX CS

YY is the SID address the response came from

XX .. XX is the data received on the CAN bus. The CAN bus is 8 bytes so all 8 bytes are returned

**Example**

**Example Request:** \$4F \$00 \$0C \$07 \$51 \$03 \$00 \$00 \$00 \$02 \$10 \$89 \$51

**Example Response:** \$B1 \$00 \$0A \$03 \$00 \$11 \$00 \$02 \$50 \$89 \$AA

## **\*\*\* Chrysler SCI Communication Interface Commands \*\*\***

### **Command \$50: SCI Engine Low Speed Message**

This command sends a message over the engine SCI bus at low speeds (7812.5 bps)

**Wait Timeout:** 20 ms to send message and enter into monitor mode

**Request:** \$50 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the SCI bus

**Response:** \$B5 YY YY XX ... XX CS

XX is the data received on the SCI bus.

#### **Example**

**Example Request:** \$50 \$00 \$06 \$28 \$00 \$62 \$E0

**Example Response:** \$B5 \$00 \$07 \$28 \$00 \$62 \$31 \$77

### **Command \$51: SCI Engine High Speed Mode 1 Message**

This command sends a message over the engine SCI bus at high speed mode 1 (62.5K bps)

**Wait Timeout:** 20 ms to send message and enter into monitor mode

**Request:** \$51 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the SCI bus

**Response:** \$B5 YY YY XX ... XX CS

XX is the data received on the SCI bus.

#### **Example**

**Example Request:** \$51 \$00 \$05 \$17 \$6D

**Example Response:** \$B5 \$00 \$05 \$62 \$1C

### **Command \$52: Send Engine Speed Init Command**

This command sends the proper speed init command to the ECM.

**Wait Timeout:** 20 ms to send message and enter into monitor mode

**Request:** \$52 \$00 \$06 XX YY CS

XX = \$00 low speed init (7812.5 bps)

XX = \$01 high speed mode 1 init (62.5K bps)

YY is the high speed byte init used when going from low to high speed

**Response:** \$C2 \$00 \$05 XX CS

XX = \$00 Failed

XX = \$01 Succeeded

#### **Example switching to F-Table \$F3**

**Example Request:** \$52 \$00 \$06 \$01 \$F3 \$4C

**Example Response:** \$C2 \$00 \$05 \$01 \$C8

## Command \$53: Send Engine J2190 Command

This command sends a message over the engine SCI bus using the J2190 protocol.

**Wait Timeout:** 20 ms to send message and enter into monitor mode

**Request:** \$53 YY YY XX ... XX CS

YY is the size of the packet

XX .. XX is the data to be sent on the SCI bus

**Response:** \$B5 YY YY XX ... XX CS

XX is the data received on the SCI bus.

### Example

**Example Request:** \$53 \$00 \$06 \$22 \$01 \$04 \$80

**Example Response:** \$B5 \$00 \$08 \$62 \$01 \$04 \$32 \$56

## \*\*\* Special Communication Interface Commands \*\*\*

### Command \$5A: PWM Signal Generator

This command creates a signal on pin 9 at the specified duty cycle. This command only sets up the pin to output the duty cycle. If a new interface is called after this is turned on the switch settings could have disabled the output of this.

**Wait Timeout:** 10 ms

**Request:** \$5C \$00 \$06 XX YY CS

XX = \$00 Signal Off

XX = \$01 Signal On

YY = Signal Duty Cycle in MS

### Example for setting a PWM cycle of 50ms

**Example Request:** \$5C \$00 \$06 \$01 \$32 \$95

### Command \$5B: H-Diag Message Single Response

This command sends a message onto the H-Diag Bus

**Wait Timeout:** 20 ms to send message and enter into monitor mode

**Request:** \$5B XX XX YY ... YY CS

XX is the size of the packet

XX ... XX is the packet to be sent over the HDiag bus.

**Response:** \$B8 XX XX YY ... YY CS

XX is the size of the packet

YY ... YY is the response from the request

### Example

**Example Request:** \$5B \$00 \$09 \$A0 \$05 \$70 \$01 \$EA \$64

**Example Response:** \$B8 \$00 \$08 \$00 \$04 \$31 \$CB \$C0

### Command \$5C: Nissan Consult DDL Message Normal Response

This command sends the specific data over the DDL bus while receiving the echo response.

**Wait Timeout:** 50 ms based on a single byte command

**Request:** \$5C XX XX YY ... YY CS

XX is the size of the packet

XX ... XX is the packet to be sent over the Nissan DDL bus.

**Response:** \$B7 XX XX YY ... YY CS

XX is the size of the packet

YY ... YY is the response from the request

**Example Request:** \$FF \$FF TRG CMD [INFOBYTE] [VALUEBYTE]

**Example Response:** XX XX NOT(TRG) NOT(CMD)

## Command \$5D: Nissan Consult DDL Message Long Response

This command sends the specific data over the DDL bus and waits for the response over the DDL bus. This is usually used with the DDL Execute command.

**Wait Timeout:** 50 ms

**Request:** \$5D XX XX YY ... YY CS

XX is the size of the packet

XX ... XX is the packet to be sent over the Nissan DDL bus.

**Response:** \$B7 XX XX YY ... YY CS

XX is the size of the packet

YY ... YY is the response from the request

**Example Request:** \$FF \$FF TRG CMD [INFOBYTE] [VALUEBYTE]

**Example Response:** XX XX NOT(TRG) NOT(CMD)

## Command \$5E: Nissan Consult DDL Set Response Timeout

This sets the DDL response timeout. The default timeout upon reset is 20ms.

**Wait Timeout:** 20 ms based on a single byte command

**Request:** \$5E \$00 \$04 XX XX \$62

XX ... XX is the time out in ms

XX ... XX = \$00 00 the default timeout is used. [50ms]

**Response:** \$B7 00 \$05 XX CS

XX = \$00 Setting the timeout failed

XX = \$01 Setting the timeout was successful