Auterra OBD II Adapter Protocol Spec

October 21, 2012

David Lafreniere

Table of Contents

Tables

Introduction

The Auterra OBD II adapter communicates with a vehicle data bus using VPW, PWM, ISO, KWP, and CAN. The adapter communicates with a remote machine over RS-232. The default data rate is 19.2k, N, 8, 1. The DashDyno implementation data rate is 115k, N, 8, 1.

The USB adapter comes with a special Windows driver that creates a virtual COM port. For USB adapters, the host PC reads/writes to a COM port just as with the serial adapter.

The adapter is powered from a vehicle OBD II connector. The vehicle must either be running, or the ignition set to 'on' position.

Acronyms

FBGO – fast bus get off.

RS-232 Protocol

(Destination) (Command) (Byte Count) (Data 0)...(Data n) (Sum Check)

Destination: The destination address is always 0x2d.

Command: This byte is the command byte. The Commands that the Diagnostic Interface will respond to are described in the description of the Request Message and the Response Message.

Byte Count: This is the number of bytes that will be sent over the automobile data bus. This does not include the error correction byte sent on the automobile data bus.

Data: The data bytes of the message that will be sent on the automobile data bus.

Sum Check: The lower 8 bits of the sum of the Command byte through the last data byte. The Destination byte is not included in the Sum Check.

All host request messages are fixed length at 15 bytes. All response messages are fixed length at 14 bytes. Unused bytes with adapter response messages are not guaranteed to be set to 0.

Protocols Supported

The OBD II adapter supports 5 protocols: VPW, PWM, ISO/KWP, and CAN. ISO and KWP use the same physical hardware layer, the only difference being the upper layer software protocols.

Adapter Types

There are three Auterra adapter types. Each adapter has a different hardware configuration.

Non-CAN Adapter

The non-CAN adapter supports OBD II protocols VPW, PWM, ISO and KWP 2000 only.

CAN Adapter

The CAN adapter supports OBD II protocols CAN, VPW, PWM, ISO, KWP 2000.

Dual Driver CAN Adapter

The dual driver CAN adapter model has an extra CAN driver that allows selection between two CAN buses. One bus is considered the "high" bus at 250 or 500kbps. The "low" bus is at 125kbps. In actuality, only the adapter software is enforcing these speed limitations. The hardware could support any speed on any bus if desired.

The dual driver CAN adapter has additional commands not found on the CAN adapter, which are:

A7, A8, A9, AA AB, AC, AD, BA, BB, BC, BE, BF, C0, C1, C2 and C3

DashDyno Software

The DashDyno uses the CAN Adapter software with the following EEPROM commands:

B9 and BA

DashDyno also communicates at 115k baud to the DashDyno main ARM processor, whereas the other implementations use 19.2k.

Host Request Messages

Two types of message requests exist: vehicle data requests and local data requests. Vehicle data requests obtain data from the vehicle. Local data requests set/retrieve data local to the adapter and do not result in a message being sent the vehicle (e.g. set a new adapter timeout value). Generically both vehicle and local data requests are termed request messages.

Each host request message invokes one or more response messages from adapter. Before issuing another request, the host must wait for a response message or wait the maximum duration specified in section Response Timing, except on requests that do not return a response (see
No Response Commands).

Vehicle Request Message Format

Request messages are from host (i.e. Personal Computer or Palm) to the OBD II adapter.

Table 1: Vehicle Request Message Format

Byte	Description
1	Destination (always 0x2D)
2	Command (see Table 2)
3	Byte Count or 1st byte of message to be sent to automobile[1].
4	1st byte of the message to be sent to the automobile.
5	2nd byte of the message to be sent to the automobile.

[1] Byte 3 of the request message is defined as a byte count on VPW, PWM, ISO, and KWP. For CAN messages, it's the 1st byte of the message being sent to the automobile.

6	3rd byte of the message to be sent to the automobile.
7	4th byte of the message to be sent to the automobile.
8	5th byte of the message to be sent to the automobile.
9	6th byte of the message to be sent to the automobile.
10	7th byte of the message to be sent to the automobile.
11	8th byte of the message to be sent to the automobile.
12	9th byte of the message to be sent to the automobile.
13	10th byte of the message to be sent to the automobile.
14	11th byte of the message to be sent to the automobile.
15	Sum check

Request Message Commands

Table 2: Request Message Commands

Command	Implemented	Local Cmd[2]	Purpose
01	Yes		Send message on ISO-9141-2 with initialization
02	Yes		Send message on J1850 VPW
04	Yes		Send message on J1850 PWM with out IFR
08	Yes		Send message on CAN Standard Identifier 250kBit/s
10	Yes		Send message on ISO-9141-2 without initialization
20			VW Intelligent Pass through Mode with 5 baud init and baud rate detect
40			VW Intelligent Pass through Mode with out 5 baud init
80	Yes		Send message on J1850 PWM with IFR (recommended for Ford)
81	Yes		Send message on ISO-9141-2 receive only 1st response
82	Yes		Send message on J1850 VPW receive only 1st response
83			Analog to Digital conversion
84	Yes		Send message on J1850 PWM receive only 1st response
85	Yes		KWP with 5 baud initialization
86			KWP report KW1 and KW2. Must do command 85 first.
87	Yes		KWP send with fast initialization. Sets ISO baud rate to 10.4kbps just like the command AB.
88	Yes		KWP send without initialization
89	Yes		KWP send without initialization and receive only 1st response
90	Yes		Send message on CAN Standard Identifier 500kBit/s
91	Yes		Send message on CAN Extended Identifier 250kBit/s
92	Yes		Send message on CAN Extended Identifier 500kBit/s
93	Yes		Send message on CAN Standard Identifier 250kBit/s 1st response[3]

[2] Local commands are ones in where the adapter returns non-vehicle data (e.g. set a timeout or filter parameter). Otherwise the request is for vehicle data.

94	Yes		Send message on CAN Standard Identifier 500kBit/s 1st response
95	Yes		Send message on CAN Extended Identifier 250kBit/s 1st response
96	Yes		Send message on CAN Extended Identifier 500kBit/s 1st response
97	Yes	Yes	Change CAN standard identifier mask and filter.
98	Yes	Yes	Change CAN extended identifier mask and filter.
99	Yes	Yes	Change adapter $P2_{CAN}$ message wait time. Adapter default is 50mS.
9A	Yes		Send message on CAN Standard Identifier 250kBit/s FBGO[4]
9B	Yes		Send message on CAN Standard Identifier 500kBit/s FBGO
9C	Yes		Send message on CAN Extended Identifier 250kBit/s FBGO
9D	Yes		Send message on CAN Extended Identifier 500kBit/s FBGO
9E	Yes		Send message on CAN Standard Identifier 250kBit/s Listen[5]
9F	Yes		Send message on CAN Standard Identifier 500kBit/s Listen
A0	Yes		Send message on CAN Extended Identifier 250kBit/s Listen
A1	Yes		Send message on CAN Extended Identifier 500kBit/s Listen
A2	Yes		Send message on CAN Standard Identifier 250kBit/s No Response[6]
A3	Yes		Send message on CAN Standard Identifier 500kBit/s No Response
A4	Yes		Send message on CAN Extended Identifier 250kBit/s No Response
A5	Yes		Send message on CAN Extended Identifier 500kBit/s No Response
A6	Yes		Send message on ISO 9141-2 No Response
A7	Yes	Yes	Set CAN bus flow control processing enable
A8	Yes	Yes	Set flow control disable message receive timeout
A9	Yes	Yes	Set ISO L-line transmit enable

[3] Host must not send any CAN 1st response messages faster than 50mS. Per J1979 if external equipment doesn't know if all messages have arrived from ECU, then it shall wait $P2_{CAN}$ maximum before sending next message. 1st response messages don't wait for all ECUs to finish responding therefore its up to the host to throttle messages to 50mS max, unless total number of ECUs is known and all have responded.

[4] The FBGO (fast bus get off) commands do not rely on the CAN error count, but instead fail upon the first RX or TX error detected. Useful during auto-detection to prevent clobbering other vehicle ECU CAN messages when transmitting at the wrong speed.

[5] The listen commands do not send a message to the vehicle; therefore the request message payload is irrelevant.

[6] The adapter does not return a response message on the No Response series of commands.

AA	Yes	Yes	Set host UART speed
AB	Yes	Yes	Set the 9600 baud ISO/KWP enable
AC	Yes	Yes	Set the ISO/KWP checksum processing enable
AD	Yes	Yes	Set the ISO/KWP 5mS inter-byte send delay enable
AE	Yes		Send CAN message on CAN Standard Identifier 125kBit/s
AF	Yes		Send CAN message on CAN Extended Identifier 125kBit/s
B0	Yes[7]	Yes	Unlock Adapter
B1	Yes		Send CAN message on CAN Standard Identifier 125kBit/s Listen
B2	Yes		Send CAN message on CAN Extended Identifier 125kBit/s Listen
B3	Yes		Send CAN message on CAN Standard Identifier 125kBit/s No Response
B4	Yes		Send CAN message on CAN Extended Identifier 125kBit/s No Response
B9	Yes	Yes	Read Non-Volatile Storage
BA	Yes	Yes	Write Non-Volatile Storage
BB	Yes	Yes	CAN high/low speed switch
BC	Yes	Yes	Read CAN idle receive buffer
BE	Yes	Yes	Set the CAN bus burst mode enable
BF	Yes		Send CAN message on CAN Standard Identifier 500kBit/s High/Low[8]
C0	Yes		Send CAN message on CAN Standard Identifier 250kBit/s High/Low
C1	Yes		Send CAN message on CAN Extended Identifier 500kBit/s High/Low
C2	Yes		Send CAN message on CAN Extended Identifier 250kBit/s High/Low
C3	Yes	Yes	CAN set high/low repeat count
C4	Yes		KWP send with Toyota enhanced fast initialization (Toyota phase 3 protocol vehicles at 9.6kbps). Sets ISO baud rate to 9600, just like the AB command.
F0	Yes	Yes	Read vehicle battery voltage using A/D converter

Unimplemented and unused commands will result in an error message in the response byte. The software version code will be part of the error message.

Local Data Request Message Format

Local data request messages are handled by the OBD II adapter directly and not sent to the vehicle. Whereas the vehicle request messages follow the same basic structure, the local data requests do not. This section specifies each local data request and response format structure.

Table 3: Local Request Message Format

Byte	Description

[7] Version 3.02 CAN and 2.02 non-CAN adapters implement this command.
[8] The value of the data bytes returned on the high/low series of commands is not defined. See command C3 for more information.

1	Destination (always 0x2D)
2	Command (see Table 2)
3	1st byte
4	2nd byte
5	3rd byte
6	4th byte
7	5th byte
8	6th byte
9	7th byte
10	8th byte
11	9th byte
12	10th byte
13	11th byte
14	12th byte
15	Sum check

Command 97

The default standard identifier mask is 0x000007f8 and filter 0x000007e8. The adapter starts with these defaults.

To change, the host sends a request message command 97. The format for the command is:

Table 4: Request Message Command 97 Format

Byte	Description
2	97
3	Mask byte #1 (Most significant byte) Always 0x00.
4	Mask byte #2. Always 0x00.
5	Mask byte #3
6	Mask byte #4 (Least significant byte)
7	Filter byte #1 (Most significant byte) Always 0x00.
8	Filter byte #2. Always 0x00.
9	Filter byte #3
10	Filter byte #4 (Least significant byte)

See Table 67: CAN Identifier Packing Example (Standard 11 Bit) for an example of how bits are packed into the above bytes. See Message Filtering for more information regarding filtering.

Table 5: Request Message Command 97 Example

Dest	Cmd	Mask				Filter				Not used				CS
2D	97	00	00	07	F8	00	00	07	E8	XX	XX	XX	XX	XX

Table 6: Response Message Command 97 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	97	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command 98

The default extended identifier mask is 0x1fffff00 and filter is 0x18daf100. The adapter starts with these defaults.

To change, the host sends a request message command 98. The format for the command is:

Table 7: Request Message Command 98 Format

Byte	Description
2	98
3	Mask byte #1 (Most significant byte)
4	Mask byte #2
5	Mask byte #3
6	Mask byte #4 (Least significant byte)
7	Filter byte #1 (Most significant byte)
8	Filter byte #2
9	Filter byte #3
10	Filter byte #4 (Least significant byte)

Table 8: Request Message Command 98 Example

Dest	Cmd	Mask				Filter				Not used				CS
2D	98	1F	FF	FF	00	18	DA	F1	00	XX	XX	XX	XX	XX

Table 9: Response Message Command 98 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	98	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command 99

Set the $P2_{CAN}$ wait time per J1979 spec. The adapter default is 50mS. Allowable settings are between 0mS and 214748mS. This command is useful to make the adapter wait longer for ECUs to respond. Also useful for ECUs that respond with a negative response service identifier of 7F (table 9 SAE J1979).

Command 99 is only useful for non-1st response messages.

Example scenario:

1. Host sends mode 1 PID 0 request to adapter
2. Adapter sends mode 1 PID 0 to vehicle via CAN
3. Vehicle ECU responds with negative response 7F
4. Adapter sends negative response 7F to host
5. Host adjusts timing using adapter command 99 to 5 seconds
6. Host sends mode 1 PID 0 request again to adapter (not a 1st response command message)
7. Vehicle ECU responds with negative response 7F
8. Vehicle ECU responds with mode 1 PID 0 response within the allotted 5 seconds
9. Adapter sends two responses (i.e. 7F and mode 1 PID 0 response) to host

Table 10: Request Message Command 99 Format

Byte	Description

2	99
3	$P2_{CAN}$ timeout byte #1 (Most significant byte). Always 0x00.
4	$P2_{CAN}$ timeout byte #2
5	$P2_{CAN}$ timeout byte #3
6	$P2_{CAN}$ timeout byte #4 (Least significant byte)

Table 11: Request Message Command 99 Example

Dest	Cmd	$P2_{CAN}$ Timeout				Not used								CS
2D	99	00	00	13	88	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 12: Response Message Command 99 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	99	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command A7

This command enables/disables the CAN-bus flow control (PDU - Protocol Data Units), which there are four flow control messages: SingleFrame, FirstFrame, ConsecutiveFrame and FlowControl. With flow control processing disabled, the adapter behavior for CAN bus send commands changes in the following ways:

1. The adapter will wait for CAN messages for the duration specified by command A8 timeout.
2. After the first CAN message is received, the adapter terminates receiving CAN messages after either (1) the duration specified by the command 99 $P2_{can}$ timeout elapses without a message arriving or (2) the command A8 flow control timeout elapses, whichever occurs first. After the adapter stops receiving messages, all accumulated responses are sent to the host[9].
3. No FlowControl message is ever sent to the vehicle.

Disabling flow congtrol processing effects all CAN bus send message commands (e.g. 08, 90, 91, …). No other message types are affected. Disabling is useful if the CAN bus messages being received do not follow the CAN-bus flow control standards.

The adapter default is flow control processing enabled. This command was implemented on the rev 05 dual CAN drivers adapters.

Example scenario:

1. Host sends command A8 and sets the timeout to 2000mS
2. Host sends command 99 and sets the timeout to 100mS
3. Host sends command 08 and requests a CAN bus message from the vehicle
4. Adapter waits up to 2000mS waiting for a response
5. Vehicle sends a response at 1500mS, the adapter accepts the message
6. Vehicle sends another response at 1575mS, the adapter accepts the 2nd message
7. Vehicle sends another response at 1650mS, the adapter accepts the 3rd message
8. Adapter command 99 100mS timeouts at 1750mS and the adapter sends the three received CAN messages to the host

[9] If a "1st response" series CAN-bus command is used, only the first ECU to respond messages will be returned to the host. All other non-first ECU messages are discarded.

Table 13: Request Message Command A7 Format

Byte	Description
2	A7
3	00 means disable CAN bus flow control processing; any other value means enable

Table 14: Request Message Command A7 Example

Dest	Cmd	Enb	Not used											CS
2D	A7	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 15: Response Message Command A7 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	A7	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command A8

This command sets the disable flow control message timeout. The adapter default value is 50mS. Allowable settings are between 0mS and 214748mS. This command is useful to make the adapter wait longer for ECUs to respond.

This command can also be used to extend the wait time for the CAN bus "listen" series of request commands (e.g. command 9E).

This command delay is only used if command A7 is set to disable flow control processing.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 16: Request Message Command A8 Format

Byte	Description
2	A8
3	Flow control message receive timeout byte #1 (Most significant byte). Always 0x00.
4	Flow control message receive timeout byte #2
5	Flow control message receive timeout byte #3
6	Flow control message receive timeout byte #4 (Least significant byte)

Table 17: Request Message Command A8 Example

Dest	Cmd	Flow control timeout				Not used								CS
2D	A8	00	00	13	88	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 18: Response Message Command A8 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	A8	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command A9

This command enables/disables the ISO L-line 10.4k transmit. Disabled is the adapter default. When disabled, any send ISO or KWP command will cause the adapter to transmit on the ISO K-line only. When enabled, the adapter will transmit on both the K and L lines at 10.4k. The adapter first sends on the K-line and then on the L-line. The transmission on both K and L lines is not simultaneous.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 19: Request Message Command A9 Format

Byte	Description
2	A9
3	00 means disable L-line transmit; any other value means enable

Table 20: Request Message Command A9 Example

Dest	Cmd	Enb	Not used											CS
2D	A9	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 21: Response Message Command A9 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	A9	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command AA

This command adjusts the host baud rate of the adapter. The adapter default is 19.2k. The adapter will always respond to the command AA request at the old speed. The *next* adapter request message must be at the new speed, if the command AA response indicates success. If an error response message is generated, the adapter will remain at the old speed.

Example scenario:

1. Host sends command B0 to unlock the adapter at 19.2k
2. Host sends 115.2k baud command AA to the adapter at 19.2k
3. Adapter responds with command AA success at 19.2k
4. Adapter now set at 115.2k. All commands sent at the new speed.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 22: Request Message Command AA Format

Byte	Description
2	AA
3	00 means set the host speed to 19.2k. 01 means set to 256k baud. Any other value generates an error response message error code 1C.

Table 23: Request Message Command AA Example

Dest	Cmd	Baud	Not used											CS
2D	AA	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 24: Response Message Command AA Example

Dest	Cmd	ReqC	Not used										CS
2D	40	AA	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command AB

This command enables and disables the 9600-baud ISO transmit. Disabled is the adapter default. When disabled, any ISO or KWP request message command will cause the adapter to transmit/receive on the ISO K-line, and optionally the L-line (see command A9), at 10.4k baud. When enabled, the adapter shall transmit/receive at 9600 baud.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 25: Request Message Command AB Format

Byte	Description
2	AB
3	00 means transmit/receive at 10.4k baud; any other value means transmit/receive at 9600 baud.

Table 26: Request Message Command AB Example

Dest	Cmd	Enb	Not used											CS
2D	AB	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 27: Response Message Command AB Example

Dest	Cmd	ReqC	Not used										CS
2D	40	AB	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command AC

This command enables and disables the ISO/KWP checksum byte processing. Enabled is the adapter default. When enabled, the adapter automatically computes the checksum for the all bytes being sent to the vehicle and sends this as the last message byte. On received messages, the last byte sent from the vehicle is error checked as a checksum. If any message from the vehicle has a checksum error, an error response message is returned to the host. When disabled, no checksum is sent; only the vehicle bytes (up to 11) are sent without appending a checksum byte. Nor is any checksum byte checked on messages received from the vehicle.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 28: Request Message Command AC Format

Byte	Description
2	AC
3	00 disable sending the checksum byte; any other value means compute and send the checksum as the last data byte of the vehicle message.

Table 29: Request Message Command AC Example

Dest	Cmd	Enb	Not used											CS
2D	AC	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 30: Response Message Command AC Example

Dest	Cmd	ReqC	Not used										CS
2D	40	AC	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command AD

This command enables and disables adding a 5mS inter-byte delay on ISO/KWP messages to the vehicle. Enabled is the adapter default. When enabled, the adapter automatically pauses for 5mS between each data byte being sent to the vehicle. When disabled, the data bytes are sent with no delay between each data byte.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 31: Request Message Command AD Format

Byte	Description

2	AD
3	00 disable the 5mS ISO/KWP inter-byte delay; any other value means the 5mS delay is used between each byte sent to the vehicle.

Table 32: Request Message Command AD Example

Dest	Cmd	Enb	Not used											CS
2D	AD	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 33: Response Message Command AD Example

Dest	Cmd	ReqC	Not used										CS
2D	40	AD	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command B0

This command was implemented on the 02 and later versions of the CAN and non-CAN adapter software. The Dyno-Scan software shows these versions as 3.02 and 2.02 respectively.

The B0 command must be sent to the adapter before the adapter will respond to any other commands. Otherwise the error code 85 is returned on all commands. This B0 command was implemented as part of the new policy of using software product keys in selling software.

Previously, all Palm software was provided "free". Users just downloaded new versions, and no product key was required. To prevent all non-product key Palm software from working with the newer software versions, this new command is implemented and thereby making all older versions of software incompatible with the newer adapters. This prevents someone from posting the older free Palm software on the Internet circumventing software sales.

The new Dyno-Scan Palm and Windows product keyed version implements this unlock command. The new Dyno-Scan software still works with pre 02 adapters since if the adapter fails to respond to the B0 command the Dyno-Scan software allows it.

Dual CAN driver specific: If all 8-bytes within non-volatile sector 0 locations are FF (FF is the default non-volatile memory state from manufacturing), then sending a B0 command using any values for the 8-byte optional unlock argument will unlock the adapter. Otherwise the host must send a B0 command an optional 8-byte unlock argument that matches the 8-bytes stored within non-volatile sector 0 before the adapter will unlock.

Table 34: Request Message Command B0 Format

Byte	Description
2	B0
3	Byte 1 unlock argument (optional)
4	Byte 2 unlock argument (optional)
5	Byte 3 unlock argument (optional)
6	Byte 4 unlock argument (optional)
7	Byte 5 unlock argument (optional)
8	Byte 6 unlock argument (optional)
9	Byte 7 unlock argument (optional)
10	Byte 8 unlock argument (optional)

Table 35: Request Message Command B0 Example

Dest	Cmd	Not used												CS
2D	B0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 36: Response Message Command B0 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	B0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command B9

This command reads from the non-volatile storage within the adapter. There are 248 bytes available to the host for storage. Each 8-byte sector can be write-protected to prevent further writes to a sector. There are 31 total sectors. Once the host write-protects a sector, it cannot be unprotected.

The host can request one sector at a time. The adapter response includes a byte indicating whether the sector is write-protected.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 37: Request Message Command B9 Format

Byte	Description
2	B9
3	Sector to read (0 to 30)

Table 38: Response Message Command B9 Format

Byte	Description
2	40
3	B9
4	Sector read
5	Write-protect status. 1 if write-protected, 0 otherwise.
6	Byte 1 read from sector
7	Byte 2 read from sector
8	Byte 3 read from sector
9	Byte 4 read from sector
10	Byte 5 read from sector
11	Byte 6 read from sector
12	Byte 7 read from sector
13	Byte 8 read from sector

Table 39: Request Message Command B9 Example

Dest	Cmd	Sect	Not used											CS
2D	B9	02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 40: Response Message Command B9 Example

Dest	Cmd	ReqC	Sect	Prot	Sector bytes read								CS
2D	40	B9	02	01	01	02	03	04	05	06	07	08	XX

Command BA

This command writes to the non-volatile storage within the adapter. There are 248 bytes available to the host for storage. Each 8-byte sector can be write-protected to prevent

further writes to a sector. There are 31 total sectors. Once the host write-protects a sector, it cannot be unprotected.

The host can write one sector at a time. The adapter request includes a byte indicating whether the sector should be write-protected after the host bytes are written. The write-protect is permanent and cannot be undone.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 41: Request Message Command BA Format

Byte	Description
2	BA
3	Sector to write
4	Write-protect sector. 0 does not write-protect the sector, any other value write-protects.
5	Byte 1 write to sector
6	Byte 2 write to sector
7	Byte 3 write to sector
8	Byte 4 write to sector
9	Byte 5 write to sector
10	Byte 6 write to sector
11	Byte 7 write to sector
12	Byte 8 write to sector

Table 42: Response Message Command BA Format

Byte	Description
2	40
3	BA
4	Sector wrote

Table 43: Request Message Command BA Example

Dest	Cmd	Sect	Prot	Sector bytes to write								Not used		CS
2D	BA	03	01	01	02	03	04	05	06	07	08	XX	XX	XX

Table 44: Response Message Command BA Example

Dest	Cmd	ReqC	Sect	Not used									CS
2D	40	BA	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command BB

This command switches the CAN controller driver between the high-speed bus (i.e. the normal OBD II adapter CAN pins) and the Volvo low speed CAN bus. The high-speed bus is the adapter default.

When the CAN driver bus is changed, it will also set the CAN controller to the destination bus speed, CAN identifier type (SI or EI), and apply the last known CAN message masks and filters stored in the adapter.

This command can only be used on the OBD II adapter with two CAN drivers connected to one CAN controller.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 45: Request Message Command BB Format

Byte	Description
2	BB
3	00 connects the CAN driver to the high-speed bus; any other value sets the CAN driver to the low-speed bus.
4	CAN bus speed and identifier type of the bus being switched to. 0 = SI 125kbps, 1 = EI 125kbps, 2 = SI 250kbps, 3 = SI 500 kbps, 4 = EI 250 kbps, 5 = EI 500 kbps.

Table 46: Request Message Command BB Example

Dest	Cmd	Driver	Speed	Not used										CS
2D	BB	01	03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 47: Response Message Command BA Example

Dest	Cmd	ReqC	Not used										CS
2D	40	BB	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command BC

This command reads the CAN bus receive idle buffer. When the adapter is idle, the adapter will store the last two CAN messages received, if any. If more than two messages arrive, the oldest buffer if overwritten with the newer data.

The adapter being "idle" means:

1) The adapter is not actively processing a command.
2) The adapter is processing a command, but the command being processed will not send and/or receive CAN bus messages.

For instance, if the adapter is sending an ISO message the CAN receive idle buffer can still receive CAN bus messages and store them into the receive buffer.

The host must set the CAN bus to the correct speed and CAN bus high/low switch using command BB before using command BC. As long as no other CAN bus adapter commands are used, the adapter will remain on the last known CAN bus setting and retrieve messages for storage into the idle receive buffers.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 48: Request Message Command BC Format

Byte	Description
2	BC
3	Buffer to read. 0 and 1 are the only accepted values.

Table 49: Response Message Command BC Format

Byte	Description
2	88
3	Byte 1 CAN identifier
4	Byte 2 CAN identifier
5	Byte 3 CAN identifier
6	Byte 1 of the CAN data field
7	Byte 2 of the CAN data field
8	Byte 3 of the CAN data field

9	Byte 4 of the CAN data field
10	Byte 5 of the CAN data field
11	Byte 6 of the CAN data field
12	Byte 7 of the CAN data field
13	Byte 8 of the CAN data field

Table 50: Request Message Command BC Example

Dest	Cmd	Buf	Not used											CS
2D	BC	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 51: Response Message Command BC Example

Dest	Cmd	CAN Identifier			CAN Data Field								CS
2D	88	00	07	E0	04	41	0C	00	00	00	00	00	xx

Command BE

This command enables or disables the CAN bus burst mode. Disabled is the adapter default.

The CAN bus burst mode is where each message sent from the host is repeated three times on the CAN bus. Each of the three messages is the same.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 52: Request Message Command BE Format

Byte	Description
2	BE
3	00 disables the CAN bus burst mode; any other value sets enables CAN bus burst mode.

Table 53: Response Message Command BE Format

Byte	Description
2	40
3	BE

Table 54: Request Message Command BE Example

Dest	Cmd	Enb	Not used											CS
2D	BE	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 55: Response Message Command BE Example

Dest	Cmd	ReqC	Not used										CS
2D	40	BE	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command C3

This command sets the repeat count used on the high/low CAN bus send command BF, C0, C1 and C2. Zero is the adapter default.

The adapter uses the high/low repeat count to limit the high/low messages sent to the vehicle. Each count will send one message. A high message is always sent first. For instance, if the count is set at 3 then two high messages and one low message will be send in this order: high – low – high.

When the BE burst mode command is enabled, each count will send three messages, such as: high high high – low low low – high high high.

If the high/low count is set to a large value, the adapter will not respond for a long time while it transmits all the requested messages.

This command was implemented on the rev 05 dual CAN drivers adapters.

Table 56: Request Message Command C3 Format

Byte	Description
2	C3
3	High/low message count #1 (Most significant byte)
4	High/low message count #2
5	High/low message count #3
6	High/low message count #4 (Least significant byte)

Table 57: Request Message Command C3 Example

Dest	Cmd	High/Low Count				Not used								CS
2D	C3	00	00	12	34	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 58: Response Message Command C3 Example

Dest	Cmd	ReqC	Not used										CS
2D	40	C3	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Command F0

The number returned is the raw A/D value for the battery voltage. Use the formula below to convert to battery voltage. The A/D converter is 10-bits.

DashDyno adapter: battery voltage = ((5.2V / 1024) x ADC count) x 4.743

Windows adapter: battery voltage = ((5.0V / 1024) x ADC count) x 4.743

Table 59: Response Message Command F0 Format

Byte	Description
2	40
3	F0
4	ADC count byte #1 (Most significant byte). Bits 0 and 1 are used from this byte. Bits 2 to 7 always 0.
5	ADC count byte #2 (Least significant byte)

Table 60: Request Message Command F0 Example

Dest	Cmd	Not used												CS
2D	F0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 61: Response Message Command F0 Example

Dest	Cmd	ReqC	ADC Cnt		Not used								CS
2D	40	F0	03	FF	XX	XX	XX	XX	XX	XX	XX	XX	XX

No Response Commands

The No Response series of commands sends to the vehicle the provided message, but the adapter does not wait for a vehicle response nor is any adapter host response returned.

Care must be taken to ensure the host does not overrun the OBD II adapter by sending messages too quickly. Since there is no "acknowledgement" message from the adapter on the No Response commands, the host is responsible for throttling the speed at which the messages are sent to the adapter. Furthermore, some interface types are faster (e.g. CAN bus) and some are slower (e.g. ISO) and therefore the maximum speed messages can be sent will vary depending on the vehicle protocol.

The OBD II adapter does not buffer multiple commands – one command at a time is accepted and processed. Therefore, when the host sends a message, the adapter receives the message and sends it to the vehicle before another message will be accepted.

Empirical testing will determine the maximum speed at which messages can be reliably transmitted[10]. Start at a delay of 100mS between host requests and gradually reduce this interval until request messages are lost. This will determine your maximum throughput.

Vehicle communication bus traffic loading can effect the consistency of the adapter transmit time. Heavy bus traffic on a VPW or PWM vehicle can suspend when the adapter is allowed to transmit. CAN uses hardware transmit buffers, and ISO/KWP has no other inter-ECU traffic so these protocols should have very consistent transmit times. In practice, the transmit delay time variations due to bus loading may be so small as to be insignificant.

High/Low Commands

The high/low series of commands are used to repeatedly send a single CAN message to the vehicle as fast as possible. Some high/low commands are BF, C0, C1 and C2.

One CAN controller exists on the adapter. However, on some adapters there are two CAN hardware drivers that are selectable. Therefore one CAN controller can be used to transmit and receive on two different CAN buses running at different speeds. Only one bus at a time can be used for transmission and reception. Any data on the non-selected bus cannot be received.

The "high" CAN command is sent at either 250kbps or 500kbps. A "low" command is sent at 125kbps. The BF through C2 commands repeatedly send one high and one low message. The message sent on each CAN bus is the same and specified in the BF through C2 send commands.

The adapter uses the high/low repeat count set with command C3 to limit the high/low messages sent to the vehicle. Each count will send one message. A high message is always sent first. For instance, if the count is set at 3 then two high messages and one low message will be send in this order: high – low – high.

[10] Vehicle communication bus traffic loading can effect the consistency of the adapter transmit time.

When a message is sent, the adapter waits until the message is fully transmitted before proceeding to the next high/low message. CAN bus loading from other CAN messages can effect how fast the messages are transmitted.

The high/low commands does not return any CAN messages received during the high/low transmission sequence. Therefore, the data returned by the BF through C2 commands is undefined.

The BE burst mode enable command, when enabled, will cause all high/low commands to send each message three times. Each three-message burst is only counted a 1 on the C3 high/low message repeat count. For instance, if the C3 high/low message count is set to 3 the CAN send message pattern will be: high high high – low low low – high high high.

On non-dual CAN driver adapters, the BF to C2 series of commands will only transmit on the high CAN bus.

Adapter Response Messages

Response messages from the OBD II adapter to the host (i.e. Personal Computer or Palm). Response messages are categorized into vehicle response messages, local response messages, and error response messages.

Vehicle Response Message Format

Table 62: Vehicle Response Message Format

Byte	Description
1	Destination (always 0x2D on Auterra adapters and 0x40 on ME adapters)
2	Response Command (see Table 63)
3	1st byte of the response received from the automobile.
4	2nd byte of the response received from the automobile.
5	3rd byte of the response received from the automobile.
6	4th byte of the response received from the automobile.
7	5th byte of the response received from the automobile.
8	6th byte of the response received from the automobile.
9	7th byte of the response received from the automobile.
10	8th byte of the response received from the automobile.
11	9th byte of the response received from the automobile.
12	10th byte of the response received from the automobile.
13	11th byte of the response received from the automobile.
14	Sum check

If there were less than 11 bytes received over the interface, the extra bytes in the Vehicle Response Message are undefined.

Response Message Commands

Table 63: Response Message Commands

Byte	Description
80	Error occurred
81	Message from ISO-9141-2

82	Message from J1850 VPW
84	Message from J1850 PWM
88	Message from CAN
01	Message from KWP
40	Local data message

If the command type is "error" than the software version and error code will be included in the data. Format of the error data is shown below.

Local data message is a response to non-vehicle messages (e.g. set timeout value or set mask/filter).

Local Response Message Format

Table 64: Local Response Message Format

Byte	Description
1	Destination (always 0x2D)
2	Response Command (always 0x40)
3	Request Command (request command in Table 2 being responded to)
4	1st byte.
5	2nd byte.
6	3rd byte.
7	4th byte.
8	5th byte.
9	6th byte.
10	7th byte.
11	8th byte.
12	9th byte.
13	10th byte.
14	Sum check

If there were less than 10 bytes required for the local response, the extra bytes in the "Local Response Message" are undefined.

Error Response Message Format

Error message responses returned from the OBD II adapter to the host.

Table 65: Error Response Message Format

Byte	Description
1	Destination (always 0x2D)
2	Command (always 0x80)
3	Always 0xFF on Auterra adapter. Version code of software if ME adapter.
4	Error Code
5	Interface Capabilities (Auterra adapter only[11])
6	Version code if Auterra adapter. 0x00 if ME adapter.
7	Undefined or extra error data depending on the error code.

[11] Auterra adapter returns the interface capabilities. The ME adapter returns an undefined byte.

8	Undefined
9	Undefined
10	Undefined
11	Undefined
12	Undefined
13	Undefined
14	Sum check[12]

Error Codes

Not all error messages may be employed at this time.

Table 66: Error Codes

Code	Description	Comments
01	Request Message command not yet implemented.	
02	Request Message sum check error.	
03	Incorrect ISO sync byte[13].	Error response message byte 7 is the address byte. Byte 8 is the sync byte received.
04	Incorrect ISO inverted address received.	Error response message byte 7 is the address byte. Byte 8 is the sync byte received.
05	No ISO response to the request message.	
06	not used at this time	
07	No J-1850 response to the request message.	
08	ISO checksum error detected in response message.	
09	J1850 CRC error detected in response message.	
0A	unused	
0B	KWP baud rate too slow	
0C	No KWP response to the request message.	
0D	KWP incorrect inverted address returned by car	
0E	unused	
0F	unused	
10	ISO not enabled	
11	J1850 VPW not enabled	
12	J1850 PWM not enabled	
13	KWP not enabled	

[12] The version code 01 software within the Auterra OBD II adapter without CAN support incorrectly includes the destination byte in the checksum computation on error response messages. Normal response messages correctly omit the destination byte in the computation.

[13] On version 05 and earlier error 03 also meant "no ISO sync byte received". On version 06 the "no ISO sync" has its own error 20.

14	VW Pass through mode not enabled	
15	No CAN response to the request message (vehicle did not respond within 50mS)	Error response message byte 7 is the COMSTAT register.
16	CAN receive buffer overflow condition	
17	Not used	
18	CAN transmit buffer not available	
19	CAN receive message length not 8 bytes	
1A	No CAN response to the request message (CAN module error interrupt fired)	Error response message byte 7 is the COMSTAT register.
1B	CAN transmit or receive error.	Error response message byte 7 has TXB0CON, byte 8 has TXB1CON, and byte 9 has TXB2CON. This error can only occur with FBGO request message commands.
1C	Invalid host baud rate selected	
1D	Invalid buffer selected	
1E	Invalid configuration selected	
1F	No ISO inverted address returned by vehicle.	Error response message byte 7 is the address byte. Byte 8 is the KW2 received from vehicle.
20	No ISO sync byte returned by vehicle.	Error response message byte 7 is the address byte.
80	Wrong destination byte	
81	Byte count too large[14]	
82	J1850 framing error	
83	J1850 message overflow	
84	Error no ISO keybyte	
85	OBD II adapter locked	This error returned on all messages if the command B0 is not first issued.
86	Invalid non-volatile sector	
87	Non-volatile sector write-protected; write failed.	

The interface automatically handles the error correction bytes for the messages. If a request message is sent and no response is received from the car, an error response message will be sent by the interface over the RS-232.

Interface Capabilities

One bit set for each interface supported. For instance, a hexadecimal value of 0x03 means ISO and VPW are supported.

Bit	Meaning
0	ISO
1	VPW

[14] This error does not apply to CAN request messages or local request messages.

2	PWM
3	KWP 2000
4	CAN
5	Dual CAN Drivers
6	DashDyno Version
7	Reserved

Version

The version is stored in BCD (binary coded decimal), regardless if the version is located at byte 3 or 6 of the error response message. For instance, hexadecimal 25 is version 25.

Multiple Response Messages (VPW, PWM, ISO and KWP)

A car may respond to a Request Message with more than one response message. If the car responds with more than one response to a single Request Message, these multiple responses will be sent over the RS-232 bus as separate RS-232 Response Messages. The maximum time between the RS-232 response messages will be 5 msec. The adapter will handle up to seven responses on the non-CAN OBD II adapters, and up to 32 responses on the CAN OBD II adapters. If there are more than responses to a single request message than can be held by the OBD II adapter, they will be ignored.

Example Request and Response Messages (VPW)

This example Request Messages uses J-1850 VPW and a RS-232 destination of 80 hexadecimal. Your interface may have a different destination address.

Request Messages bytes on the RS-232 bus (request RPM):

Dest	Cmd	Len	Header Bytes			Data Bytes								CS
2D	02	05	68	6A	F1	01	0C	00	00	00	00	00	00	XX

The message transmitted (sent by the J-1850 interface) on the J-1850 bus will be:

Header Bytes			Data		ERR
68	6A	F1	01	0C	8B

The message received (sent by the car, engine off) on the J-1850 bus will be:

Header Bytes			Data Bytes				ERR
48	6B	10	41	0C	00	00	XX

The RS-232 Response Message to the above request message will be:

Dest	Cmd	Header Bytes			Data Bytes				Undefined					CS
2D	82	48	6B	10	41	01	00	00	00	00	00	00	00	XX

Response Timing

The adapter will respond to all host request commands within 350mS, except for the ISO INIT (01) and KWP 5 BAUD INIT (85) request commands, which will respond within 5500mS.

ISO 9141-2 Response Timing

An important issue is the amount of time needed to get responses over the ISO-9141-2 bus. There can be a long delay from the time a request message is sent, and the response. The long delay occurs due to ISO-9141-2 bus activity. The ISO-9141-2 bus does an initialization at 5 bits per second. It is totally unnecessary in some cars, but the specification requires it for the first message in any string of messages. To handle this there are different types of request messages commands bytes that can be used. One request message command byte does the initialization, and then sends the request. The other request message command byte causes just the request to be sent. For faster response, send the first request with initialization, and the rest without initialization. Per the ISO-9141-2 specification if a delay of five (5) seconds or greater occurs without a request being sent then the initialization must be repeated.

1st Response Commands

After the autodetection phase, the 1st response series of commands can be used to increase the data sampling speed (see Table 2: Request Message Commands). A 1st response command returns the first encountered vehicle ECU message. All subsequent remaining vehicle responses, if any, are discarded.

VPW, PWM, ISO and KWP Protocols

VPW, PWM, ISO and KWP request message commands have this message format:

<Destination> <Request Command> <Length> <Header> <Data Bytes> <Sum Check>

VPW, PWM, ISO and KWP response message commands have this message format:

<Destination> <Response Command> <Header> <Data Bytes> <Sum Check>

On VPW, PWM, ISO and KWP the adapter does not enforce or distinguish between the Header and Data Bytes. The adapter just sends up to 11 bytes of the data between byte 4 and 14.

Example Request and Response Message (VPW)

Host request messages bytes on the RS-232 link (request engine RPM). In the following example, 'x' means not defined or computed, and '?' means sensor data returned from the vehicle.

Dest	Cmd	Len	Header Bytes			Data Bytes								CS
2D	02	05	68	6A	F1	01	0C	00	00	00	00	00	00	xx

The adapter response message to the above host request message will be:

Dest	Cmd	Header Bytes			Data Bytes				Undefined				CS
2D	82	48	6B	10	41	0C	??	??	00	00	00	00	xx

CAN

CAN request message commands have this message format:

<Destination> <Request Command> <Identifier> <Data Field> <Sum Check>

CAN response message commands have this message format:

<Destination> <Response Command> <Identifier> <Data Field> <Sum Check>

On the CAN protocol, the adapter uses the Identifier bytes as the CAN standard or extended identifier, and the Data Field bytes and the CAN 8-byte message payload.

Adding the CAN protocol to the adapter should be backwards compatible with all existing non-CAN adapters already in the field. Therefore a design goal is to keep the request and response messages the same fixed length and message structure as is designed today.

Unlike the other vehicle protocols, the data field portion of the CAN message is always 8 bytes. This 8 bytes is hard coded into the CAN DLC (Data Length Code) portion of the CAN frame. In addition, byte #1 of the data field is encoded with the length of the OBD II portion of the message, which can be between 1 and 7 on a single frame message. Therefore for these reasons, on CAN messages the byte count field within the adapter request message (see Table 1: Vehicle Request Message Format) is used for additional CAN data and not a byte count.

Bytes 3 through 6 (4 bytes total) of a vehicle request message (see Table 2) are used for the CAN standard and extended identifier. This is a binary 32-bit number with the first bit transmitted MSB. The standard identifier is only 11 bits and the extended identifier is 29 bits. All unused bits shall be set to 0. The CAN identifier bytes are binary zero padded on the left.

Table 67: CAN Identifier Packing Example (Standard 11 Bit)

Byte 3		Byte 4		Byte 5		Byte 6	
Most Significant Nibble							Least Significant Nibble
0	0	0	0	0	7	D	F
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 1	1 1 0 1	1 1 1 1

The CAN 8 data bytes are stored within bytes 7 through 14 of the vehicle request message.

Response messages are placed in data bytes 3 through 13 (11 bytes) (see Table 62: Vehicle Response Message Format). Since an extended identifier response requires 12 bytes, the left most byte (i.e. the first 5 bits of the 29 bit identifier) is not sent back to the host. Per the OBD II standards, the first 5 bits will always be binary 0 0 0 0 1, therefore the adapter protocol will lose no critical information (e.g. a valid 29-bit identifier is 0x18DAF116).

Having only 11 bytes of data within the response message makes the message length backwards compatible with existing adapter in the field.

The CRC field, end of frame, and control field within a fully formed vehicle CAN message is handled by the adapter and is not sent within the request message. The only values

sent/received from the adapter in CAN mode is the identifier and the 8 data field bytes. The adapter handles all other CAN frame fields.

Note the first 1 to 3 bytes within the CAN data field have special meaning. On all frames the first nibble is encoded with the frame type. Then depending on the frame type, the number of bytes, etc… is encoded in the remaining bit fields. See the ISO 15765-4 for more information.

Segmented Vehicle Request Messages

The J1979 OBD II standard does not require supporting sending segmented CAN messages. The adapter doesn't support this feature per-se, but the host PC/Palm could implement segmented sending, however the host is responsible for handling the necessary flow control handshake. The adapter does not assist in any way.

Segmented Vehicle Response Messages

The J1979 OBD II standard requires support of receiving segmented messages. The adapter will assist in this process by automatically sending the FlowControl message to the vehicle when a FirstFrame is received. SingleFrame messages do not require flow control.

The adapter shall buffer up each received CAN message within a segmented message series before sending all messages to the host PC/Palm.

The host is responsible for concatenating all segmented messages together to reassemble the original message.

If a 1st response CAN message is sent and the vehicle responds with a segmented message, the adapter shall send back to the host the FirstFrame and all ConsecutiveFrame messages, if any, associated with the response. If any other ECUs respond between the FirstFrame and ConsectutiveFrames from the 1st ECU, the OBD II adapter discards these messages. Or in other words, only the 1st ECU to respond messages will be sent to the host even if other non-first ECU communication intersperses the first ECU.

Example Request and Response Message (CAN)

Request Message standard identifier on the RS-232 bus (request RPM). In the following example, 'x' means not defined or computed, and '?' means sensor data returned from the vehicle.

Dest	Cmd	CAN Identifier				CAN Data Field								CS
2D	08	00	00	07	DF	02	01	0C	00	00	00	00	00	xx

The RS-232 Response Message to the above request message will be:

Dest	Cmd	CAN Identifier			CAN Data Field								CS
2D	88	00	07	E0	04	41	0C	??	??	00	00	00	xx

Flow Control

The OBD II adapter will automatically respond with this FlowControl message without host PC/Palm intervention when a FirstFrame CAN message is received from the vehicle. This

message tells the sender to continue sending the remaining ConsecutiveFrame messages.

The host identifier used in the FlowControl message is the same as the last host request command identifier.

Per table 7 in 15765-4 on page 14, the BS and STmin will always be 0.

Table 68: FlowControl Message

Byte 1		Byte 2		Byte 3	
	FS	BS		STmin	
3	0	0	0	0	0
0 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Message Filtering

The default CAN identifier message filters and masks are:

Table 69: Default Mask and Filter Values

29-bit Extended Identifier Filter	18 DA F1 00
29-bit Extended Identifier Mask	1F FF FF 00
11-bit Standard Identifier Filter	7 E8
11-bit Standard Identifier Mask	7 F8

The masks and filters can be changed by the host using adapter request commands 97 and 98.

Auto-Detection

When auto-detecting the vehicle CAN speed (i.e. 250 or 500kb/sec) and identifier (i.e. standard or extended) care must be taken not to clobber existing vehicle communications for too long when transmitting at a speed different than the vehicle. For instance, if the adapter transmits at 250kb/sec on a vehicle equipped with 500kb/sec it should get off the bus quickly to avoid disturbing other CAN module communications. We're calling this feature "fast bus get off", or FBGO for short.

Normally the CAN module within the adapter has an error counter. This error counter keeps track of the number of transmission and reception retires. However, during auto-detection waiting for the counter limit to expire will clobber the other CAN module communication for too long causing U trouble codes to appear on some vehicles (e.g. U0100, U0101, U0121).

To prevent mismatched speed communications from disturbing other vehicle ECU CAN communications, the host should use the FBGO (fast bus get off) series of CAN request commands during auto-detection.

The listen series of CAN request command messages could also be used to auto-detect if the vehicle already has communication traffic. Note, the CAN filter and mask will need to be set to accept all vehicle traffic, not just scan tool destined traffic.

Vehicle Autodetection

If the protocol used by the vehicle is unknown, the host must autodetect the vehicle communication type by trying each interface protocol type. During autodetection, the following request command sequence should be employed:

Table 70: Autodetection Sequence

Request Command (hex)	Command Name
80	PWM
02	VPW
01	ISO INIT
10	ISO
88	KWP
85	KWP 5 BAUD INIT
87	KWP FAST INIT
9a	CAN SI 250KB AUTODETECT
9c	CAN EI 250KB AUTODETECT
9b	CAN SI 500KB AUTODETECT
9d	CAN EI 500KB AUTODETECT
08	CAN SI 250KB
91	CAN EI 250KB
90	CAN SI 500KB
92	CAN EI 500KB

If the vehicle doesn't support the interface command selected, the adapter will respond with an error response as shown in Table 65. If supported, a vehicle response message is returned as shown in Table 62.

ISO/KW5P Autodetection

Some ISO/KWP vehicles require an initialization command be sent before the vehicle responds to requests. There are three initialization commands: ISO INIT (01), KWP 5 BAUD INIT (85) and KWP FAST INIT (87).

Once a successful initialization command is sent and accepted by the vehicle, the host must switch to the non-initialization series of commands. For instance, if ISO INIT (01) returned a successful response message, then the host switches to ISO (10) commands for continued communication with the vehicle. Similarly, if KWP FAST INIT (87) or KWP 5 BAUD INIT (85) connected, then KWP (88) is used in subsequent communications.

The initialization command must be sent if:

1. it's the first communication with the vehicle.
2. a delay of 5 seconds or greater occurs without a host request being sent.
3. an error response message is returned from a vehicle that previously passed the autodetection phase.

During autodetection, the host must remember the initialization command that succeeded. If the host needs to re-initialize the vehicle, the same initialization command used during autodetection must be employed.

The ISO INIT (01) and KWP 5 BAUD INIT (85) commands can take over 5 seconds for the initial response. See section Response Timing for more information.

VPW, PWM and CAN vehicles do not use initialization commands.

CAN Autodetection

To communicate on CAN during the autodetection sequence, the "autodetect" series of CAN commands must be employed (9a, 9b, 9c, and 9d). After establishing communication, the corresponding non-autodetect series of commands should be used. For instance, during autodetection if CAN SI 500KB AUTODETECT (9a) established communication, subsequence requests should use CAN SI 500KB (90).

Bluetooth Adapter

The RN42 can auto-pair with a Windows PC. This is used for manufacturing to eliminate the manual steps to pair the adapter with the PC using a fixed Bluetooth address hard coded into the software. This address can be changed with a recompile if the Windows PC test fixture is ever changed. Once the adapter test fixture software passes, it writes to an EEPROM location to prevent the auto-pairing on the customer.

Example Host Request Messages

The following host request messages are completely formed, including sum check. Each message queries the vehicle for mode 1 PID 0. All byte data values are in hexadecimal.

PWM
2d 80 05 61 6a f1 01 00 00 00 00 00 00 00 42

VPW
2d 02 05 68 6a f1 01 00 00 00 00 00 00 00 cb

ISO INIT
2d 01 05 68 6a f1 01 00 00 00 00 00 00 00 ca

ISO
2d 10 05 68 6a f1 01 00 00 00 00 00 00 00 d9

KWP
2d 88 05 c2 33 f1 01 00 00 00 00 00 00 00 74

KWP 5 BAUD INIT
2D 85 05 C2 33 F1 01 00 00 00 00 00 00 00 71

KWP FAST INIT
2d 87 04 c1 33 f1 81 00 00 00 00 00 00 00 f1

CAN SI 250KB AUTO DETECT
2d 9a 00 00 07 df 02 01 00 00 00 00 00 00 83

CAN SI 500KB AUTO DETECT
2d 9b 00 00 07 df 02 01 00 00 00 00 00 00 84

CAN EI 250KB AUTO DETECT

2d 9c 18 db 33 f1 02 01 00 00 00 00 00 00 b6

CAN EI 500KB AUTO DETECT
2d 9d 18 db 33 f1 02 01 00 00 00 00 00 00 b7

VW Intelligent Pass Through Mode

This is not yet implemented on the Auterra ODB II interface adapter.

The interface has two special commands for doing the VW intelligent pass through mode. The 20 command allows initialization at 5 baud with the K and L lines and do what ever you want in pass through mode just like the simple pass through interfaces often used with VW diagnostic programs. The T16-003 interface is recommended for the VW Intelligent Pass Through Mode.

To use the VW intelligent pass through mode a normal 15-byte long request message is sent to the interface. The interface sends whatever 5-baud byte you specify in the request message, and then it looks for the 55 hexadecimal sync byte to determine the baud rate used by the ECU. After that the diagnostic interface is in pass through mode. You can receive the key bytes from the ECU in pass through mode. Because the interface is now sending whatever the current level of the RS-232 input you can send what ever inverted key byte you want. You can use whatever baud rate you want in pass through mode. You can send whatever length string you want.

The command for the VW pass through mode is 20 hexadecimal. Here is how it works;

1. You send a RS232 request message of 15 bytes to the interface. A part of this message (first data byte) is the controller address that you want to use during the 5-baud init.

2. The interface sends the controller address that you want at exactly 5 baud on both the K and L lines.

3. The interface observes the sync byte (55) from the ECU and determines the period of baud rate used by the ECU.

4. The interface sends to you a RS232 message with a baud rate counter value that you can use to know the car's ISO baud rate. Each count of the baud rate counter equals 2.5 microseconds of baud rate period.

5. You then receive KW1 and KW2, send the KW2 inverted to the ECU. All future messages are in pass through mode unless 5 seconds passes without any ISO bus activity. After 5 seconds the interface returns to normal mode.

Command 40 allows a direct transition to pass through mode. The 5-baud init and the baud rate detection of the sync byte are skipped.

Checksum

Below is the host request/response checksum algorithm in C. Note, the first and last byte of the adapter message is not including in the checksum computation.

```
unsigned char checksum(char* msg, int len)
{
```

```
    unsigned char checksum = 0;
    for (int i=0; i<len; i++)
        checksum += msg[i];

    return checksum;
}
```

OBD II Adapter

One the serial OBD II adapter, the female DB9 connector connects to the vehicle using an OBD II cable. The male DB9 connector connects to the PC/Palm. The PC-to-adapter RS-232 cable has a null modem pinout.

One the USB OBD II adapter, the female DB9 connector connects to the vehicle using an OBD II cable. The USB connector connects to a Windows PC using a standard USB cable. The USB OBD II adapter can *only* connect to a PC, not a Palm or PocketPC device.

OBD II Cable

Figure 1 shows the standard Auterra OBD II cable pinout.

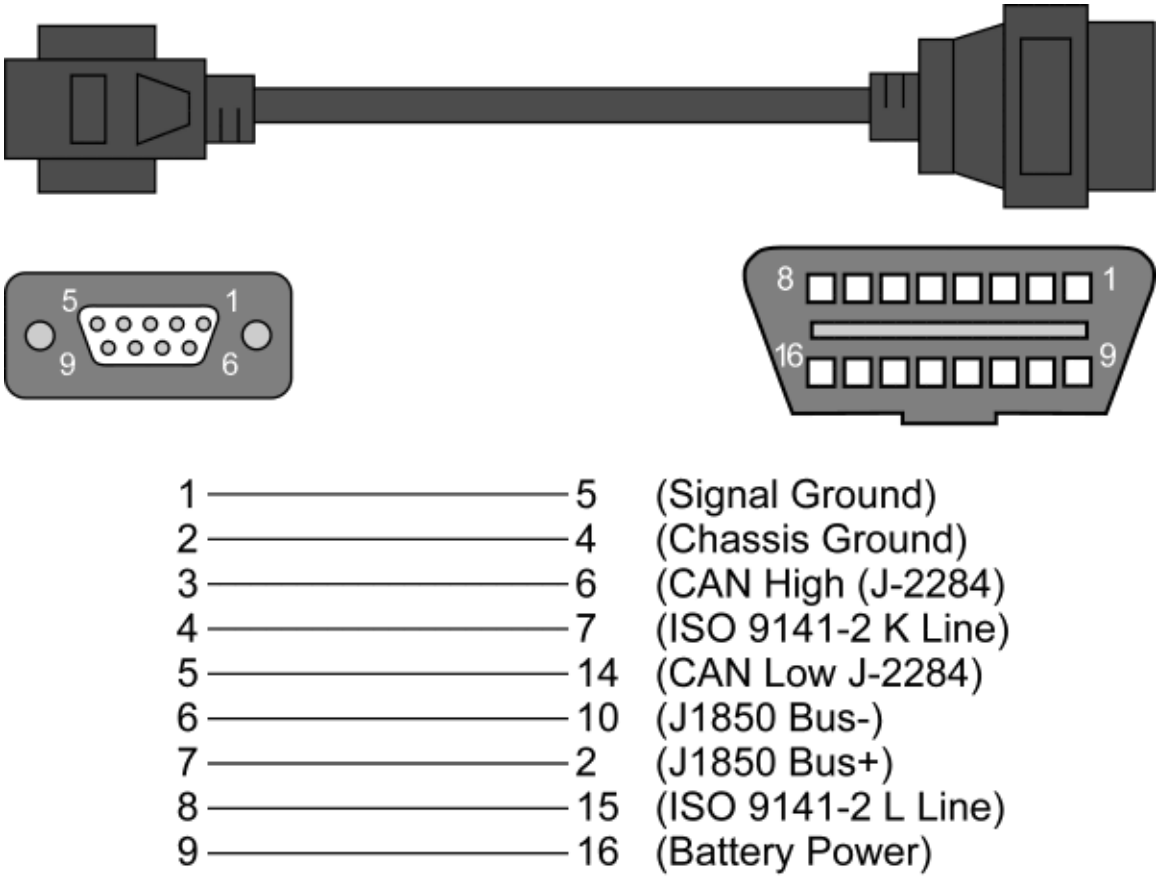

Figure 1: Auterra OBD II Cable Pinout

Adapter Revisions

Changes to the OBD II adapter are captured here. Version numbers are in hex and extractable from the adapter on every adapter error response message.

Non-CAN Adapter

01 – Initial release.

02 – Added command B0 to adapter. Fixed minor PWM bug.

03 – Fixed adapter rejecting host messages with byte count higher than 7; correct value was 10.

Allow more VPW/PWM messages to extend receive beyond the hard coded 300mS. J1979 APR2002 page 10 table 4 calls out 100mS max interval between received messages. Adapter now handles this.

CAN Adapter

01 – Initial release.

02 – Added command B0 to the adapter. Fixed minor PWM bug.

03 – Fixed adapter rejecting host messages with byte count higher than 7; correct value was 10.

Allow more VPW/PWM messages to extend receive beyond the hard coded 300mS. J1979 APR2002 page 10 table 4 calls out 100mS max interval between received messages. Adapter now handles this.

Increased maximum stored vehicle messages from 20 to 32.

04 – Fixed adapter lockup issue on Toyota Tacoma/Tundra 2006.

Eliminated the receive message filter for ISO vehicles.

Added new "no response" series of commands A2, A3, A4, A5 and A6.

05 – Added EERPOM commands B9 and BA on DashDyno build only (regular adapter did not support EEPROM in this version).

DashDyno build communicates with the host ARM at a default 115k, whereas the regular adapter uses 19.2k to communicate with the host PC.

06 – Added commands A9, AB, AC, and AD.

Commands 03, 04, 1F and 20 now return extra data in error response bytes 7 and 8 (see command table).

New #fuses in hardware.h that might be necessary to enable EEPROM read/write (meaning earlier versions may support the host EEPROM commands but the adapter may no be able to actually read/write to that area).

EERPOM host commands B9 and BA available on regular adapter build. Previously only DashDyno build implemented the EEPROM commands, now all build versions support these host commands.

07 – Changed to support the new 18F2480 part and the old 18F248. The ADC registers changed and the new code handles the ADC differently based upon the processor version. No new features added.

08 – Updated to support Bluetooth auto-pairing. Default baud 115k.

Dual Driver CAN Adapter

This CAN adapter is a special version with twice the flash and RAM. It also has two CAN drivers and commands to allow switching between two separate CAN buses. A DB15 connector is used for the OBD II cable in lieu of a DB9 because two extra CAN lines are required. The "high" speed is the OBD II cable bus (250kbps and 500kbps) and the "low" speed is the Volvo-specific bus on OBD II connector pins 3 and 11.

10 – Initial release. Added commands A7, A8, A9, AA AB, AC, AD, BA, BB, BC, BE, BF, C0, C1, C2 and C3 to the CAN adapter code base.

11 – Changed to support the new 18F2580 part and the old 18F258. The ADC registers changed and the new code handles the ADC differently based upon the processor version. No new features added. **(The code for this change has not been implemented yet!)**