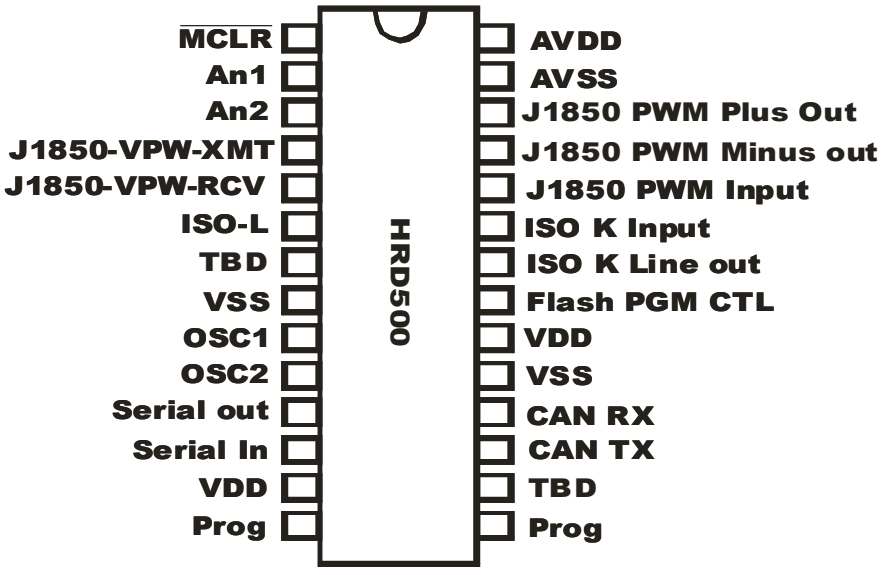
APEX V1.0 Vehicle Data Bus Interface

Introduction

Providing all OBD-II and EOBD protocols, including CAN, and a very software friendly interface, the APEX chip is ideal for scan tools, code readers, telematics and Car Computers. The APEX is based on a high speed Digital Signal Processor (DSP), which means it's fast enough to handle all vehicle messaging applications and it has large internal data buffers for storing multiple message responses from all protocols.
The APEX allows the ISO-9141 and ISO-14230 interfaces to be customized for different bit rates, message timing and 1,2 or 3 byte message headers. The serial data interface is set to 115,200 baud as default, but can go to 625,000 baud via user commands. There are two high speed, 10bit, analog inputs which can be used for battery voltage monitor, external Wide Band O2 sensor interface, Accelerometer input for a useful Dyno/Performance analyzer, etc. It's possible to get analog data rates of 1000 samples/sec at 115,200 baud, which makes a low speed oscilloscope possible too.
All OBD-II modes, 1 through 9, are handled by onboard firmware which automatically takes care of message formatting, CRC or checksum, bus initialization and error detection. This makes the software for creating a scan tool very simple and straight forward. All responses from the APEX are terminated with CR-LF codes so the users application software only has to wait for a "read line complete" event to get any response.

Specifications



MCLR (pin 1) - A logic low applied to this input will reset the IC. If unused, this pin should be connected to a logic high (VDD) level.	AVDD – Analog supply voltage, usually 5v supply connection or 2.5-5.0v reference.
AN1 - This analog input is used to measure a 0 to VCC signal that is applied to it. Care must be taken to prevent the voltage from going outside of the supply levels	AVSS – Analog ground reference
AN2 - This analog input is used to measure a 0 to VCC signal that is applied to it. Care must be taken to prevent the voltage from going outside of the supply levels	J1850 Plus Output – PWM
J1850-VPW-XMIT	J1850 Minus Output – PWM
J1850-VPW-RCV	J1850 PWM input
ISO-L – Output for ISO L line, usually not needed	ISO –K Line Input
CTS – Clear to Send for RS232	ISO-K Line Output
VSS - Ground	FLASH PGM Control – output line can be used to control the application of programming voltage for vehicle re-flash applications
OSC1 – 5.000 MHZ crystal, parallel cut	VDD
OSC2 -	VSS
Serial Out – 5v serial data output	CAN RX – Input
Serial In – 5v serial data input	CAN TX – Output
VDD	Future VPW 4X Enable – can be used to control the 4X pin of the MC33390 J1850 VPW driver
PROG – Factory use only	PROG – Factory use only

The APEX uses the standard RS232 type serial communication. The default data rate is 38400, 115,200 or 428000 baud with 8 data bits, no parity bit, and 1 stop bit. All responses from the APEX IC are terminated with a acknowledge as described in the detailed description below. The data rate can be changed on-the-fly by using the 'C1' command as defined below.

Vehicle Communication Commands for OBD-II

Currently OBD-II defines 9 modes of operation, all 9 are supported by the APEX with specialized software to minimize communication overhead.

OBD-II Mode 1	Request data by PID
Send	01XX(CR) CR=Carriage Return, ASCII code 13 XX= PID byte
Receive	Returns single message string – example V486B10410536CA(CR

OBD-II Mode 2	Request Freeze Frame Data
Send	02-PID-Frame-CR Frame is always 00
Receive	single message string – example send 020000 – PID=00, show all supported Mode2 PIDs Rec. - V416B104200007F980000(CR)

OBD-II Mode 3	Request Trouble Codes
Send	03(CR)
Receive	Single or multiple messages, refer to SAE-J1979 for complete specifications V0A416B104304010000000099 – indicates trouble code P0401

OBD-II Mode 4	Clear Codes and reset Malfunction Lamp
Send	04(CR)
Receive	V486B10444436(CR) if successful, anything else means unsuccessful

OBD-II Mode 5	Legacy O2 Sensor Test Results
Send	05 TID O2sensor CR -050001(CR)
Receive	Returns single message string – example: Send 050001(CR) Rec-V0B486B0E4500010000000108 (CR) If TID or O2 not supported then response will be V(CR) Refer toSAE-J1979 for full information

OBD-II Mode 6	
Send	For non-CAN systems send:06 TID – CR, TID=Test ID, usually defined by the vehicle manufacturer. For CAN vehicles send 06-#of TID's-TID1-TID2….TID6 You should zero fill unused PIDs.

Receive	Returns single or multiple message string – example: Send-0600, Receive -V0B486B0E4600FF0001818109 (CR) – From ISO9141 Honda If TID not supported then response will be V(CR) For CAN, to get the results of TID 01, O2 Sensor Monitor Bank 1-Sensor 1, send 060101(CR), the vehicle replies: V101C4601010A0E68210000FFFF01800A2217401002FFFF 0123810E01CF00E60B24B8000000000000 Please see ISO-15031-5 for the description of the 28 byte reply.

OBD-II Mode 7	Pending Codes
Send	07(CR)
Receive	Single or multiple messages, refer to SAE-J1979 for complete specifications

OBD-II Mode 8	Control the operation of an on-board system, test or component.
Send	08 TID DataA-DataB-DataC-DataD-DataE
Receive	V486B1048-TID-D0-D1-D2-D3-D4(CR) Example: Send 08010000000000(CR) Rec. V0A416B104801000000000085 A response of 'VERR' means the data buffer to transmit did not contain sufficient characters to form a message.

OBD-II Mode 9	Read VIN, Calibration Numbers For NON-CAN OBD the VIN is 17 characters for vehicles that provide electronic access to the VIN. VIN characters shall be reported as ASCII values. The response consists of the following messages: Message #l shall contain three filling bytes of $00, followed by VIN character #l Message #2 shall contain VIN characters #2 to #5 inclusive Message #3 shall contain VIN characters #6 to #9 inclusive Message #4 shall contain VIN characters #lO to #13 inclusive Message #5 shall contain VIN characters #14 to #17 inclusive
Send	09-01-PID-(CR)

Receive	Send 090102(CR) to get VIN Multiple messages, CAN Bus example : - response to 090102 for CAN communications V10144902013146542150573134353235224642323532333 8 Legacy OBD Example Send 0902(CR), Receive: V0A416B1049020539000000A30A416B1049020434363938DE0A416B1049020333594E42AE0A416B1049020258313757F90A416B104902013146545278

In all cases, if the vehicle did not respond to an OBD request, only a 'V' followed by a (CR) will be sent. One more thing about the OBD-II communication. When a Mode 01, PID 00 command is issued, every OBD-II PCM in the vehicle will respond. Usually there is only one PCM, but some cars will have two, an engine controller and a transmission controller. This can lead to extra work in sorting out responses from multiple ECU's when acquiring data. The APEX default settings will only receive bus messages from the lowest number ECU (the Engine Controller) responding to a 00-01 command. This keeps the clutter out and simplifies user software.

APEX ConfigurationCommands

The APEX uses default settings for normal communication with OBD-II vehicles. For those cases when non-OBD communication is desired, we provide a means of configuring the APEX to send- receive raw messages on any of the supported protocols, the filters for selecting received messages can be changed, the ISO-9141 and KWP2000 baud rate can be changed

Command C1	Set Baud Rate - allows the user to change the baud rate of the RS232 communication. Allowable baud settings are: **64 – 19200** **31 – 38400** **10 – 115,200** **02 – 428,000**
Send	C1BB(CR) – BB is baud rate value
Receive	VB(CR)

Command 43	**Set keep alive message.** Used to send the ISO required message to keep the interface alive if no commands are sent for 5 seconds. Usually the default will never be changed
Send	42-68-6A-F1-01-00
Receive	VK(CR)

Command 92	Sets the active protocol to one of the 8 OBD-II types 0- No protocol 1- J1850 PWM 2- J1850 VPW 3- ISO9141 4- ISO14230 (KWP2000) 5- CAN 11 bit 500k 6- CAN 29bit 500k 7- CAN 11 bit 250k 8- CAN 29bit 250k
Send	9200XX(CR), XX= protocol type 920001(CR) sets interface to J1850 PWM
Receive	VP(CR)

Command 93	Find vehicle OBD-II protocol. Commands the APEX determine the OBD bus. The chip will scan through all interface types once to determine which interface is currently connected. The vehicle interface must respond according to the ISO-15031 specification to be detected. The response will be PX, where X is a number representing the protocol detected or 0 for no protocol detected. If ISO9141 or KWP2000 is detected, then the Keep Alive message will be sent every 3 seconds. The default value for the Keep Alive is a "Mode 1, PID 00" command.
Send	93(CR)
Receive	PX(CR), example, P5(CR) would indicate CAN 11 bit, 500k baud found for vehicle protocol.

Command 94	**Read Vehicle Protocol -** This command returns the selected or found protocol. A value of 00 means no protocol has been selected or detected yet.
Send	94(CR)
Receive	XX, for example 01(CR) – found PWM A value of 00 means no protocol has been selected or detected yet.

Command 95	**Set ISO Timeout -** Sets the timeout value. – determines how long to wait before determining the messages are complete. The resolution of the wait is about 3uS. A value of 0x4000 will set the wait to 50ms.
Send	95XXXX(CR) – example 958000(CR) set ISO time out to 100ms
Receive	VT(CR)

Command 96	**Set ISO bit timing -** default set to 96us for 10.4kbps, can be changed by the user if required. Add 2us to the value to get the actual bit time. For a value of 96us, use a value of 94. This is used to change the baud rate of the ISO interface for non-OBD-II applications.
Send	9600XX(CR) – XX is 8 bit value of microseconds, in hex, representing bit time. Add 2us to the value to get the actual bit time. For a value of 96us, use a value of 94. Allows bit rate to go as low as 3922b/sec.
Receive	VB(CR)

Command 98	**Set ISO gap time -** The ISO-9141-2 specification requires a 5ms gap between words transmitted by a tester (scan tool). For non-OBD applications, the 5ms gap slows down the communication by a factor of about 5. The Set ISO Gap time command allows you to set the gap to any number from 0-255ms for the gap.
Send	98XX(CR) - XX time in ms, nominal 5
Receive	VG(CR)

Command 99	**Set Filters for non-CAN received messages** – This set the match filter for J1850 and the ISO -9141 and ISO-14230 messages. When messages are received the header is checked and if the first two bytes of the received header match the H1-H2 values the message is stored and transmitted out the serial port. The standard headers specified in non-CAN communications have three bytes, the third byte is address of the transmitting ECU. Since the interface address is byte 2, we can receive all messages destined for the interface with only the first two header bytes. Example: send OBD-II command for RPM on J1850 PWM: Diagnostic request 61 6A F1 01 0C Diagnostic response: 41 6B F1 41 0C XX XX, XXXX-16bit RPM value If H1=41 and H2 = 6B then the message will be accepted. The Command 99 is need only be used for non-OBD-II applications, the interface will set the filters to the proper values if the vehicle interface is found using the Command 93.
Send	99H1H2(CR) – H1 and H2 are header bytes 1 and 2 of a 3 byte header. When messages are received the header is checked and if the first two bytes of the received header match the H1-H2 values the message is stored and transmitted out the serial port. Since the Tester address is the third header byte, we can receive all messages destined for the interface by filtering on only the first two header bytes.
Receive	VH(CR)

Command F0	**Passthrough mode for legacy OBD-II**. This will allow the user to send any message on any of the non-CAN protocols. The user need to be sure the proper protocol has been set, Command 92, and the filters are set for the expected message repaly, Command 99, the ISO time out if the protocol is ISO, Command 95. Data received as a result of the F0 command will be formatted and sent out the serial data port so long as bus messages meeting the Header criteria are found, until the interface times out wait for a message.
Send	F0-Byte Count-B1-B2-B3-B4…B11(CR) Example: send a message to J1850 PWM physical address 0x10 F006C410F122164800000000 The message length is 6 bytes and the remainder are zero filled. It's not required to send 11 bytes at all times.
Receive	VA(CR) followed by the received message or messages. The first byte returned will the the total byte count of all messages followed by the message(s). 0BC4F02853076200000000 Here the byte count is 0B, or 11 bytes.

CAN Communication Commands

The Controller Area Network (CAN) is a newer and more sophisticated interface than the older OBD-II interfaces. CAN is very fast, running at 500k bits/sec on most vehicles and is used as a in-vehicle network for many engine, transmission, ABS,SRS and other systems as well. There are two types of CAN protocols in use on OBD-II and EOBD vehicles, the 11 bit address messages and 29bit address messages. The data rate can be either 500k or 250k bit/sec on either type. The APEX will initialize with the proper CAN ID (11 or 29bit) and message filters and masks for standard OBD-II or EOBD communication. When writing custom programs for non OBD-II communication, the user must set the proper can ID, Mask and Filter.

The message acceptance filters and masks are used to determine if a message should be loaded into either of the receive buffers. Once a valid message has been received the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifiers are examined with the filters. A truthtable is shown below that indicates how each bit in the identifier is compared to the masks and filters to determine if the message should be loaded into a receive buffer. The mask bit essentially

determines which bits to apply the filter to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit.

Mask Bit n-------	Filter Bit n------	Message Identifier bit--------	Accept or Reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: x = don't care

Command C3	**Set CAN Filter –** send a 32 bit value to the specified filter number. There are 5 different filters that can be used. When setting 29bit filters set bit 7 of the filter number. To set filter 3, 11 bit CAN, send C303000007E9(CR), to set filter 2, 29 bit CAN, send C3831C460A20. These are examples only, values must be set according to the device being communicated with.
Send	C3-NN-00-00-00-00-(CR)
Receive	CF(CR)

Command C4	**Set CAN Mask -** send a 32 bit value to the specified Mask number. There are 2 different filters that can be used. When setting 29bit filters set bit 7 of the filter number. To set filter 3, 11 bit CAN, send C303000007E9(CR), to set filter 2, 29 bit CAN, send C3831C460A20. These are examples only, values must be set according to the device being communicated with.
Send	C4-NN-00-00-00-00-(CR)
Receive	CM(CR)

Command C4	**set_CAN ID -** Allows the 29 or 11 bit CAN address of the APEX to be set.
Send	C8 –XX-XX-XX-XX(CR) Example C8000007E0, set to 7E0 hex.
Receive	VI(CR)

Command C9	**Passthrough CAN message -** The eight bytes following the C9 are transmitted on the CAN bus using the current CAN_ID. This command requires a fixed byte count of 9 bytes, the C9 command and 8 bytes following. All CAN messages are 8 bytes and any unused bytes are zero filled. Example: send packed CAN message -- temp = ("C9" & "07" & "01" & sPID1 & sPID2 & sPID3 & sPID4 & sPID5 & sPID6)
Send	C9B1B2B3B4B5B6B7B8(CR)
Receive	Data received as a result of the C9 command will be formatted and sent out the serial data port so long as CAN bus messages meeting the MASK/FILTER criteria are found. Messages are formatted with the message ID as the first 32bit long word, followed by the eight bytes of the CAN messageExample: 000007E8100D41040004000400007E82100040004000400 In the example the message ID is 000007E8 and there are two messages present and no spaces between messages.

APEX Utility Commands

Command 41	**Read Analog 1 –** Read the voltage on AN1. Returns a 10 bit value (3 digits) representing 0.0 to 5.0 volts. A value of 000=0v, value of 3FF=5.0v
Send	41(CR)
Receive	XXX(CR)

Command 42	**Read Analog 2 –** Read the voltage on AN1. Returns a 10 bit value (3 digits) representing 0.0 to 5.0 volts. A value of 000=0v, value of 3FF=5.0v
Send	42(CR)
Receive	XXX(CR)

Command 90	**Send F/W ID -** Read the current revision of firmware
Send	90(CR)
Receive	'OBDscan 1.00' or latest version

Command D0	**Reset to default** – Resets the HDR500 to all default values
Send	D0(CR)
Receive	'OBDscan 1.00' or latest version

Command 70	**Monitor Mode** - Starts requesting any message on bus and outputs to serial port. Monitor Mode is terminated when any data is received on the serial port input of the APEX. Prior to entering Monitor Mode, use the Set Filters to receive all messages, H1=00.
Send	70(CR)
Receive	All messages meeting the filter criteria will be transmitted until the HDR500 receives a character on the RS232 interface

Using the APEX

Communication with an OBD-II vehicle should be started using the "93" command for 'Find Vehicle'. The APEX will cycle through all 5 interfaces one time in an attempt to

communicate with a compatible vehicle. If an OBD-II vehicle is found the APEX will respond with PX, where X is the interface type, 1=PWM, 2=VPW, 3=ISO9141, 4=KWP2000, 5=CAN 11bit, 6=CAN 29bit. If 11bit CAN is found the response would be P5. If the response is P0 then no vehicle was found in that attempt.
Once communication is started with a vehicle, the user can send any of the OBD-II Mode commands to the interface. For example, sending a 0100, mode 1, PID-0, the response back from the APEX should be something like V486B100100BFB39993, as an example. Even though the CAN protocols do not use the older 3 byte headers, the old headers are appended so the response from a CAN vehicle will look the same as the older OBD protocols. This makes the OBD-II software interface to the APEX the same for all OBD-II vehicles. It should be noted here that when using the Pass-Through mode the CAN interface is much different.

Monitor Mode – Command "70" – Starts requesting any message on bus and outputs to serial port. Monitor Mode is terminated when any data is received on the serial port input of the APEX. Prior to entering Monitor Mode, use the Set Filters to receive all messages, H1=00.

Application Software for OBD-II Communication

By using the new RS232 data functions in Visual Studio .NET 2005, interfacing to the APEX is very easy. Since all data replies from the APEX are terminated with a CRLF, use the 'Line Input' function and there is no need to know how many bytes are expected, when the function returns you will have the complete message from the chip. In VB.net, define the COM port as " `Public COMX As IO.Ports.SerialPort`" in a common code module. The ReadLine function has a Timeout value which is set using the `COMX.ReadTimeout`. For example, to set a 500ms timeout on a read, use `COMX.ReadTimeout = 500`.

Read_Serial_Data

```
    Try
        Incoming = COMX.ReadLine()
     Catch ex As Exception
        Incoming = ""
    End Try
    returnStr &= Incoming
```

Writing data to the APEX is equally easy. Use COMX.Write(string) to send data. To send a Find Vehicle command, COMX.Write("93" & vbCr). All communication to the APEX chip must be terminated with a CR as that is what the APEX uses to determine End Of Line. The vbCr function in VB.NET will send a Carrige Return character.

To assign COMX to a specific serial port, use :

```
COMX = My.Computer.Ports.OpenSerialPort("COM1", 115200)
```

All future calls to COMX will reference COM1 on the computer. Our default baud rate is 115,200, you could use any of the supported baud rates after sending a Change Baud Rate command to the APEX at the default rate.

Here is an example of reading the VIN using Mode 9, PID 2, OBD_II protocol is CAN
Send 0902 and Cr
Read the serial port until a line feed is received, data will look like:
V10144902013146542150573134353235224642323532333 8
This message is broken down as follows
V10 is appended header, 14 is the total byte count in Hex, 49 is response to Mode 9 with thr 40h bit set, 02 is the PID, 01 indicates first message, 314654 are first 3 bytes of the VIN (1FT), 21 indicates second message, 573134353235, next 7 bytes of VIN (PV14525), the 22 indicates third message and 46423235323338 are next 7 bytes of VIN (FB25238). When VIN is assembled: 1FTPV14525FB25238.

USB Applications

We recommend the new FTDI FT232R USB-RS232 interface. By using this chip in the Serial Port Emulation mode, you can use the same software as the RS232 program described above and lose no throughput when compared to running the FT232R in native USB mode. This greatly simplifies the USB interface work.

Appendix A

Table of standard OBD-II and EOBD messaging

Header bytes			Data bytes								
Priority/Type	Target Address	Source Address	# 1	# 2	# 3	# 4	# 5	# 6	# 7	ERR	RESP
Diagnostic request at 10.4 kbits IS0 11519-4 and IS0 9141-2											
68	6A	F1	Max. 7 Data Bytes							Yes	No
Diagnostic response at 10.4 kbit/s: IS0 11519-4 and IS0 9141-2											
48	6B	ECU Addr	Max. 7 Data Bytes							Yes	No
Diagnostic request at 10.4 kbit/s (IS0 14230-4)											
11 LL LLLLb	33	Fx	Max. 7 Data Bytes							Yes	No
Diagnostic response at 10.4 kbit/s (IS01 4230-4)											
1 OLL LLLLb	Fx	Addr	Max. 7 Data Bytes							Yes	No
Diagnostic request at 41.6 kbits (SAE-J1850 PWM)											
61	6A	Fx								Yes	Yes

Diagnostic response at 41.6 kbit/s (SAE-J1850 PWM)					
41	6B	Addr		Yes	Yes