BR16F84-1.07

OBD II Interface Chip Data Sheet

For PWM, VPW, and ISO 9141-2 Vehicles

VPW (general motors), PWM (Ford products), and ISO 9141-2 (Asian/European)

Final Release Date: Jul 8 2001 Firmware Version: 1.07 Chip ID# 107F

If you are using a completed unit, you don't need to read this data sheet. This data sheet is intended for those who are building their own hardware using the BR16F84-1.07 Chip, or writing custom software to employ this device in a particular application.

GENERAL DESCRIPTION:

This device is a CMOS microcontroller which is designed to interface a personal computer or laptop with a vehicle's On Board Diagnostic (OBD II) interface. It is intended to function with all three of the protocols used by vehicle manufacturers to implement the OBD II system as defined by SAE and ISO specifications. The OBD II system became mandatory for 1996 and up vehicles, but some vehicles were already fully or partially compatible with OBD II requirements prior to 1996. The chip is not suitable for the earlier vehicles such as OBD I.

The device is intended to function as a simple scan tool and is capable of sending and receiving any OBD II message defined in SAE J1979 for any of the three types of OBDII bus implementations (PWM, VPW, ISO 9141-2). It can also be used as an inexpensive interface for custom instrumentation monitoring various vehicle parameters such as speed, RPM, coolant or intake air temperature, engine load, intake air flow rate, etc.

FEATURES:

Operating Voltage 5.0 V
Operating Current 5 Ma. Typ
Clock: 20 MHz

Inputs: (0-5V levels)
- Serial Input (19200 Baud)
- PWM receive
- VPW receive
- ISO 9141-2

Outputs (0-5V levels)
- Serial output (19200 Baud)
- PWM transmit bus+ (inverted)
- PWM transmit bus- (inverted)
- VPW transmit
- ISO 9141-2 L line transmit (inverted)
- ISO 9141-2 K line transmit (inverted)

ELECTRICAL and PHYSICAL CHARACTERISTICS:

This device is a CMOS microcontroller produced by Microchip Technologies. You can obtain detailed specifications for the chip from Microchip data books, or on the Microchip website, so this section will only provide a brief summary. It is available in several package designs and temperature ratings. The baseline BR16F84-1.07 chip is a microchip 16F84A-20/P part, which is a commercial temperature range plastic 18 pin DIP rated at 20 MHz. The device can be special ordered if other package designs, temp ratings, etc are desirable.

TYPICAL APPLICATION CIRCUIT:

The typical application is a simple automotive scan tool using a PC or laptop as the host machine. The interface to the PC is a serial link running at 19200 baud. Only 3 wires are needed, as no handshaking signals are used. The serial output from the interface circuit is a 5 volt signal, which works well with most PC's and laptops. The serial input is clipped and current limited, so it is capable of accepting almost any signal voltage. The interface to the vehicle OBD bus is accomplished with a few external components and an LM339 comparator chip. The connection to the vehicle uses 6 wires, assuming all three OBD protocols are to be implemented. Operating voltage for the chip and associated circuits is small, and is derived from the vehicle's 12V system on pin 16 of the OBD connector.

Refer to the schematic of a typical application. This design uses readily available components.

APPLICATION HINTS

(1) The cable to vehicle can be simple unshielded wires. This cable should be kept as short as possible, and definitely no longer than 4 feet, especially for PWM operation. Somewhat longer cables could be used if the circuit's termination resistors are replaced with lower values.

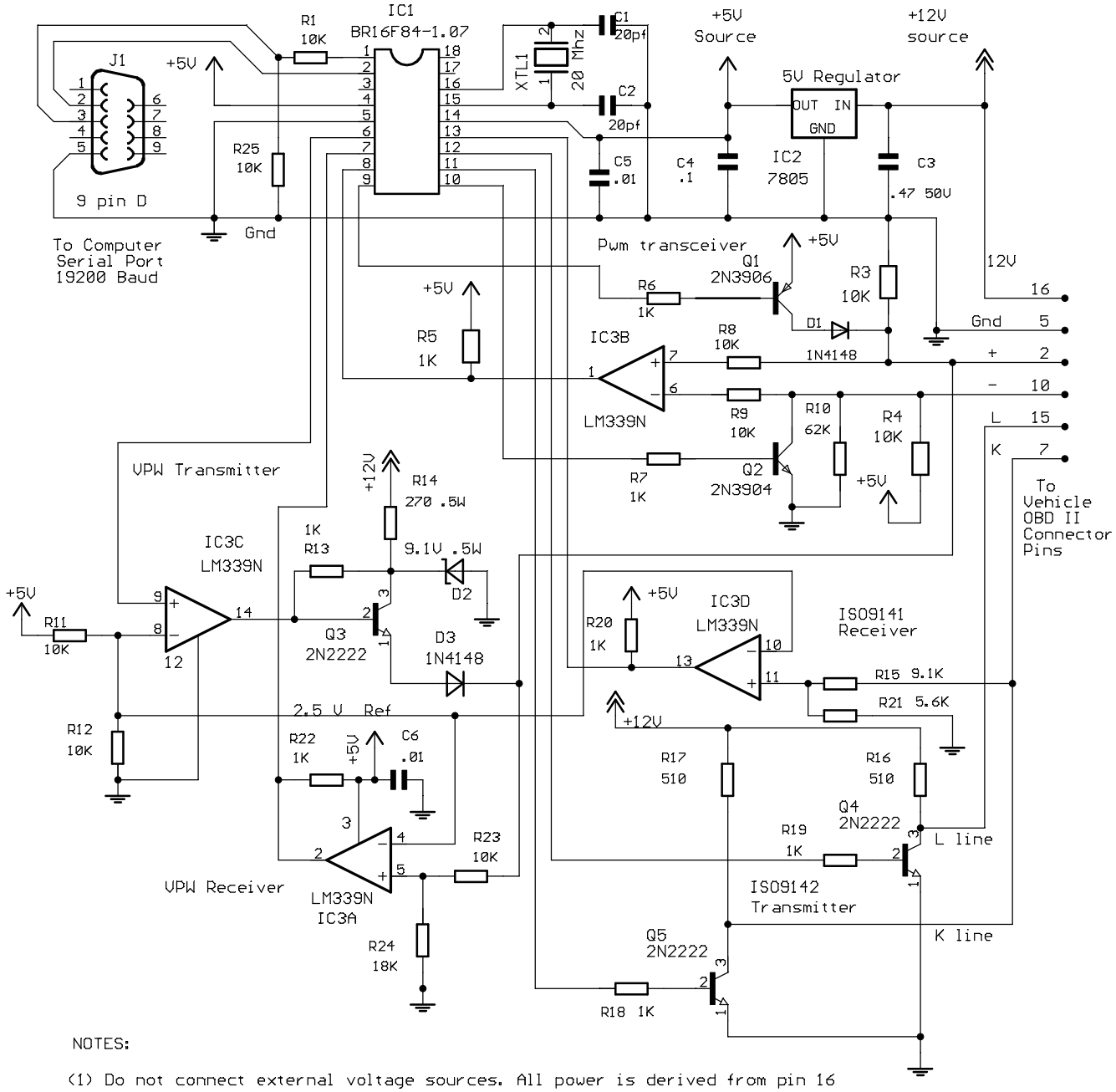
(2) The cable to Computer serial port can also be unshielded. The application circuit has been tested with unshielded cables up to 30 Ft., but specific implementations may vary, so you may want to experiment if you are attempting long runs. For every long runs, you might also wish to use an RS 232 transceiver chip in your design.

(3) Circuit layout is not critical. The circuit has been tested on both hard wired perfboard prototypes and plugboard breadboards. However, if your layout is very sloppy, you may want to add additional bypass capacitors.

(4) Free software to utilize the application circuit is available for download. The program will run under DOS, so it can be used on older computers as well as newer machines running windows. It is small enough that it will fit on a DOS boot floppy, so you can still run it if your main operating system is not DOS compatible. The computer need not even have a hard drive.

OBDII Interface

For VPW, PWM, and ISO 9141-2 vehicles



NOTES:

(1) Do not connect external voltage sources. All power is derived from pin 16 of the OBDII connector.

(2) Wire leads to OBD connector should be kept under a few feet. Serial link leads to the computer can be 20 or 30 feet or more. Use plain unsheilded 3 conductor cable.

(3) Be sure circuit is enclosed or otherwise protected from short circuits.

(4) IC1 (16F84-20) has Version 1.07 Firmware.

PARTS LIST :

Value	Description	Qty	Designators
.01 UF	Capacitor, 25V	2	C5, C6
.1 UF	"	1	C4
.47 UF	Capacitor, 50V	1	C3
20 pf	"	2	C2,C1
10 KOhm	Resistor, 1/4 Watt	9	R11, R12, R23, R9, R1, R3, R4, R25, R8
18 KOhm	"	1	R24
1 KOhm	"	8	R19, R18, R7, R20, R22, R5, R6, R13
5.6 KOhm	"	1	R21
510 Ohm	"	2	R16, R17
62 KOhm	"	1	R10
9.1 KOhm	"	1	R15
270 Ohm	Resistor, 1/2 Watt	1	R14
-	Not used	1	R2
1N4739	Zener, 9.1V, 1W	1	D2
1N4148	Diode	2	D1, D3
PN2222	Transistor, NPN	3	Q3, Q4, Q5
PN3904	Transistor, NPN	1	Q2
PN3906	Transistor, PNP	1	Q1
7805	Regulator, 5V	1	IC2
LM339N	Quad Comparator	1	IC3
16F84	Microprocessor	1	IC1
20 MHz	Crystal, 20 Mhz	1	XTL 1

Miscellaneous:
Board, connectors, cable, wire, sockets, enclosure, etc,
Depending on your implementation.

NOTES:
1) All Resistors can be 5% Carbon Film types (Don't use wirewound types)
2) Capacitor voltage ratings can be higher than specified.
3) Capacitors can be ceramic or monolithic.
4) Microprocessor uses Ver 1.07 firmware
5) Transistor packages can be substituted. Example: 2N3904 for PN3904

CHIP COMMANDS AND RESPONSES:

This section details the information needed to send and receive data to the chip from a host computer. It will enable you to write your own software to utilize the chip with whatever host computer or other control device you may wish. If you are using the provided program, you need not read this section. Please note that in the following discussion, all numerical values are in hexadecimal base unless otherwise mentioned.

GENERAL DISCUSSION OF COMMUNICATIONS

This is an overview of communications between the chip and the host computer or controller. There are exceptions to these rules which are defined in the more detailed sections of this document. All communications is via a simple 3 wire serial link. No handshaking is used. The chip listens for a message, completes the task and then returns the results to the host. Then, it immediately waits for the next message. The typical exchange to or from the chip is a series of bytes. The first byte is called the control byte. The concept of the control byte is discussed here because it applies to many of the commands and responses (but not all; exceptions are noted in the specific commands defined below). Normally, the control byte is simply a number between 0 and 15 decimal (0-F hex) which indicates how many bytes follow this control byte. For example, if three bytes are to be sent, the control byte would be 3 decimal (or 03 hex), then the three bytes would follow; 03, byte1, byte2, byte3. This is the method used to ask the chip to retrieve a particular item of vehicle OBD data, and it is the method used by the chip to return the Vehicles response to the request message. Note that this only requires using the lower four bits of the control byte. The upper bits are reserved for certain special commands and responses. These bits are used for the host computer to make initial connection with the chip, to set the protocol to be used, and for the chip to return data indicating success or failure. In particular, note that the chip will set the most significant bit of the control byte if the action was not successful. If the operation was successful, the upper four bits of this control byte will be zero. These special uses of the control byte are discussed below, and exceptions where the control byte is not used are also indicated.

INITIALIZING THE CHIP AND THE VEHICLE

Before normal communications can start, the host must establish communication with the chip, then it must initialize the chip and the vehicle's data link.

(1) CHIP CONNECT

After the unit is connected to the host computer and the vehicle, an initialization sequence must be done. This sequence is used to prevent hangups due to noise on the serial lines in case they have not been connected before the chip is powered up. It also serves as a simple method to detect whether the interface is connected or not. The first operation is to send a single byte, 20H. This is a connect command to the chip, and it tells the chip that communication is to commence. The chip does not send a control byte in return, but responds with a single byte which is FF Hex, or decimal 255. At this point, the chip is waiting to receive data, and the host may then proceed with initialization. NOTE that this is one of the few cases where the chip does not send a control byte.

(2) INITIALIZATION

This initializes the protocol to be used, and also initializes the vehicle in the case of ISO. There are three protocols, VPW (general motors), PWM (Ford products), and ISO 9141-2 (Asian/European). There are many exceptions to these general protocols, for example, some Mazda's use Ford's PWM protocol. So, if you have problems with a particular protocol, you should try another. The protocol selection is done by sending a control byte which is 41H, followed by a protocol select byte. The latter byte is defined as follows; 0=VPW, 1=PWM, 2=ISO 9141. For example, 41H 02H is the sequence to initialize to ISO 9141 protocol.

The chip will respond with a control byte and a status byte. The control byte will have the MSB set if there was a problem, and in this case the next byte is a status byte which indicates internal conditions. If the initialization was a success, the control byte will be 01H, which indicates one verification status byte follows. This status byte is a verification byte and is defined as follows; For VPW or PWM, this byte is simply an echo of the protocol byte that you sent (0 or 1). For ISO 9141 it will be a "KEY number" that was returned from the vehicle itself and specifies one of two slightly different versions of ISO. It is for informational purposes only. Note that for VPW and PWM, the protocol select is rapid, because it is only needed to inform the chip which protocol is to be used. But, for ISO vehicles, it can take up to approximately 5 seconds, because there is a detailed initialization sequence that the chip must perform at 5 baud with the vehicle. Also NOTE that some legacy ISO 9141 vehicles will lose initialization if no data requests are made during a 5 second interval, so the host PC should automatically send a request every few seconds during idle periods.

Once connection and initialization is complete, messages would typically be request messages from the host computer, or response messages from the chip.

REQUEST AND RESPONSE MESSAGES

The operation of this chip is significantly different for the SAE (VPW or PWM) protocols, and the legacy ISO9141-2 protocols, so the information is presented separately for the two cases to avoid confusion.

SAE (VPW or PWM) COMMUNICATIONS

For these protocols, the chip can only buffer one frame of data, so you must specify which frame it should grab and return. The vehicle may send more than one frame, so if it' s a data item that can possibly have a multiple frame response, you would need to repeat the request until you have obtained all of the frames. Generally, this is only necessary for a few types of data.

The Request message always takes the following form:
[Control byte], [SAE Request Message], [Frame number]
As usual, the Control byte just indicates the total number of bytes to follow. The SAE request message is defined in SAE J1850 and J1979. It is composed of three header bytes, a series of message bytes, and a CRC byte. Note that the SAE Request Message conforms strictly to the relevant SAE specifications, while the control byte and the Frame number are required by the chip. The SAE messages are outside the scope of this data sheet but are available from any number of sources.

The Response message always takes the following form if the result was a success;
[Control byte], [SAE Response Message]. The control byte simply indicates how many bytes follow. These bytes are the SAE response message, and they contain the header bytes, frame message bytes for this response, and the CRC byte as per SAE specs referenced above.

The Response message always takes the following form if the result was not successful;
[Control byte], [Status byte]. The control byte will have the most significant bit set. The lower 4 bits will be 001, which simply indicates that one byte follows, and it is the status byte for internal conditions. This is a commonly occurring condition, because specifications allow a vehicle to return no data, or invalid data if the requested information is not supported by the vehicle, or if the data is not currently available in the vehicle's processor. If the chip received no response, or if it received an invalid response, the most significant bit of the control byte will be set, and one byte of internal condition status will follow.

ISO 9141-2 COMMUNICATIONS

The ISO 9141-2 system used by many Asian and European vehicles is quite different from SAE vehicles, so it is discussed separately. The request message from the host is very similar to SAE vehicles, but the chip does not require or allow a frame number. The request message is simply the control byte indicating the number of bytes following, then the request message frame, including the checksum. The response from the chip is simply a pure retransmission of data received from the vehide. There is no control byte, so the host should simply receive bytes until a time-out period of 55 Milliseconds occurs with no bytes received. This indicates the end of the vehicles response. The response is therefore one or more message frames as defined in SAE J1979. The chip performs no analysis of frames, does not reject non-diagnostic frames, etc. The host must process this data to extract particular frames by looking for the header bytes. For most data requests, there is only one response frame.

Notes and updated information:

The above information is basically unchanged for later versions. Omissions, changes and errata are listed below.

Bus contention error message: This applies to PWM and VPW protocols only. If a bus contention occurs, the interface will return a single byte (40 Hex). It is a control byte with the lower nibble set to zero, so no other bytes will follow. This indicates the contention. This condition can normally occur when the vehicle's bus is busy with higher prioirity messages and the diagnostic request looses arbitration.
PC Software should resend the original request message.

Interface Chip Changes for later versions:

This document specifies the changes in chip operation for later versions.
It defines how to use it for ISO 14230 for both fast and slow initialization.
There are also some changes that may affect useage for PWM and VPW.
The major changes to existing protocols are summarized below:

(1) For ISO 9141, The address byte is now also transmitted.
(2) For ISO 9141, Both keybytes are returned, rather than one.
      The extra byte is also returned for SAE modes, but unused.
(3) The ISO 14230 protocol is now supported.

Notes:
(1) All data bytes are in Hexadecimal.
(2) XX indicates a byte which is not defined, reserved, or unknown.

CHIP CONNECT SEQUENCE: (This sequence is unchanged)

	Send	20
	Receive	FF

PROTOCOL SELECTION SEQUENCE:

VPW:	Send	41, 00
	Receive	02, 00, XX
PWM:	Send	41, 01
	Receive	02, 01, XX
ISO 9141:	Send	42, 02, adr Where adr is the address, (normally 33 hex).
	Receive	02, K1, K2 Where K1, K2 are the ISO Keybytes.
	Or	82, XX, XX Indicates failure to initialize ISO 9141

ISO 14230 (Fast Init):

	Send	46, 03, R1,R2, R3, R4, R5
		Where R1 to R5 is ISO14230 Start Communication Request Message;
		Normally R1-R5= C1, 33, F1, 81, 66
	Receive	S1, S2, ...
		Where S1, S2..... is ISO14230 Start Communications Response message.
		Note: More than one ECU may respond in sequence. The response may also be a negative response code.
		A typical positive response message would look like this;
		S1, S2,... = 83, F1, 10, C1, E9, 8F, BD

ISO 14230 (Slow Init): Same as ISO 9141.

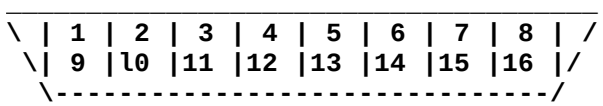
CIRCUIT NOTES FOR THOSE BUILDING THEIR OWN INTERFACE :

The schematic is shown with simple pads for signals to the vehicle
and the personal computer. Below is information to wire the connector
to the vehicle and also the connector to the PC serial port.

---------- VEHICLE 16 PIN CONNECTOR ----------

Diagramatically, the diagnostic connector looks like this:
(Use a fixed width font to view this diagram)

REAR view of male plug from interface device (where pins are inserted)
(This is also the Front view of Female Connector in the vehicle)



Here is how to hook them up to the interface circuit.
Note that only a few of the pins are used and listed here.
If you are wiring a male connector, remember:
The above diagram referrs to the REAR of the Male connector,
and be sure to wire the male connector properly:

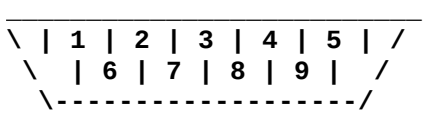
INTERFACE SIGNAL	CONNECTOR PIN #
12 Volts (vehicle battery)	16
Signal ground	5
Bus + (for VPW & PWM)	2
Bus - (for Pwm only)	10
L line (for ISO 9141)	15
K line (for ISO 9141)	7

All other pins are not used.
This cable should be no longer than 3 to 4 ft. A sheild is not needed,
so if present, leave it unconnected to reduce capacitance.

---------- SERIAL CONNECTOR ----------

The only other connector to wire is the standard 9 Pin Female D connector
which plugs into your PC. Only 3 pins need be wired.

REAR view of Female 9 pin D connector.
(where the pins are inserted or the wires are soldered).



Here is how to hook them up to the interface circuit.
Note that only a few of the pins are used and listed here.
Remember that the diagram refers to the REAR of the female D connector,
and be sure to wire the connector properly:

INTERFACE SIGNAL	PC SIGNAL Name	D CONNECTOR PIN #
Out	Rx Data	2
In	Tx Data	3
Gnd	Ground	5

All other pins are not used. Normally, you do not need a shield,
so if present, the shield need not be connected.
It is recomended to keep the cable length 30 Ft or less,
but you may be able to make it longer, depending on the characteristics
of the cable and the serial port itself.

---------- NOTES ON COMPONENTS ----------

Most of the resistors can be standard 5% carbon film.
There is one resistor (270 Ohms) which should be 1/2 watt.

If you only want to use this circuit with one or two types of protocols,
you can look at the schematic and omit any parts that are not needed.
For example, if you only wanted to use with VPW (Domestic General Motors),
you would only need 3 wires (pins 16,5 and 2) connected at the vehicle.
In this case, you could also omit several components from your circuit.

Here are some examples of parts that can be omitted.

If you don't use PWM, you can omit:
R4, R6, R7, R8, R9, R10, Q1, Q2, D1

If you don't use ISO, you can omit:
R15, R16, R17, R18, R19, R21, Q4, Q5

If you don't use VPW, you can omit:
R13, R14, R23, R24, D2, D3, Q3

The components used in the schematic are not critical.
Except for the pre programmed microcontroller and the crystal,
you can probably get all the parts from Radio Shack,
(They are usually available for less cost at electronics stores
or by mail order.)

You can use plain 5% tolerance carbon film resistors.

If you build this circuit on a piece of perfboard or a pc board
without an enclosure:
Take care that you don't lay it on a conducting surface,
causing it to short out. Your vehicle is pretty well protected,
but you could damage the interface circuit or keep it from operating.

Note that there is no reset button for the interface processor.
To reset just unplug from vehicle, then replug it to the vehicle.
(That will automatically reset the interface processor).
Then restart the PC's software program and reinitialize the interface.