Features

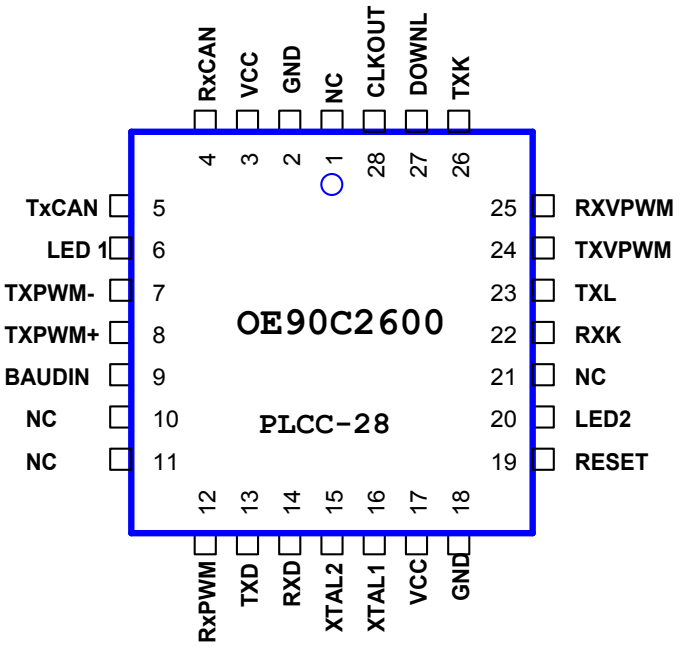
- Compatible with EOBD/OBDII standard
- Communication at 9600 Baud
- OBDII / EOBD Modes 1..9
- ISO9141
- KWP2000
- J1850 VPWM and PWM
- CAN Bus
- Multiple response from multiple ECU
- Physical and fonctional addressing

EOBD/OBDII to RS232 gateway

OE90C2600

Description

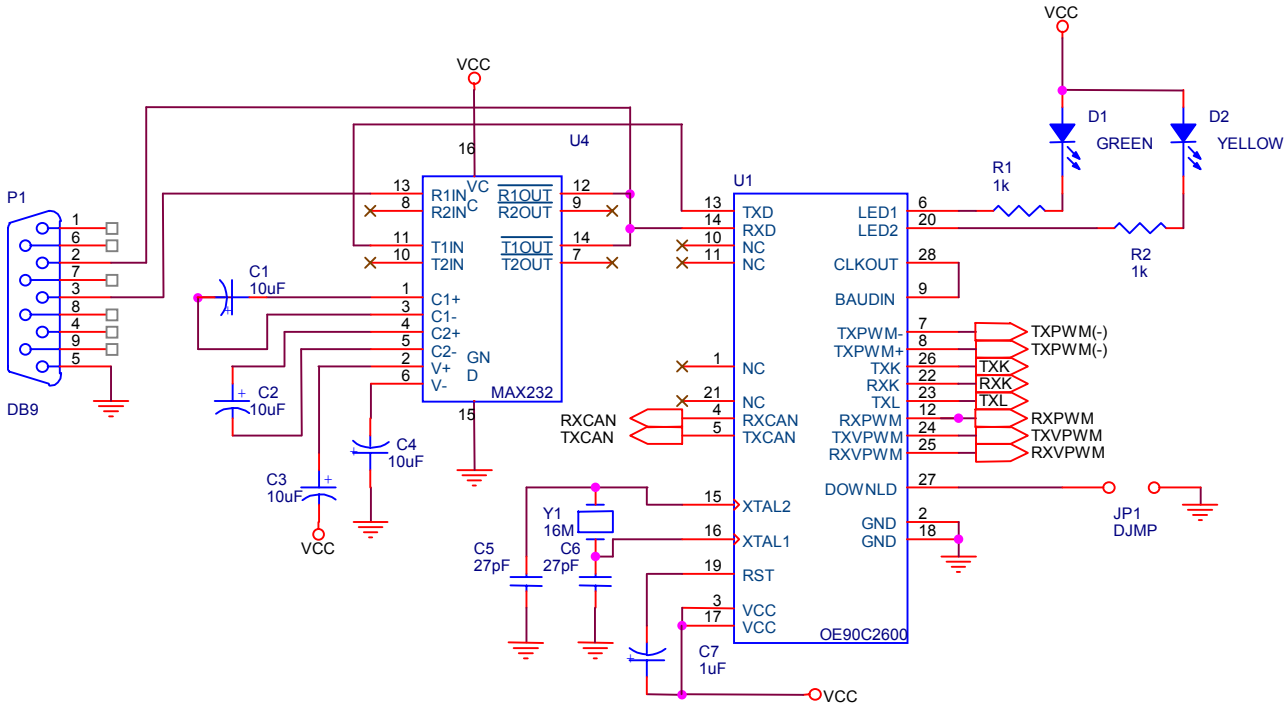
OBDII is an international standard for communication between automobiles and diagnostic testers.it specifies a serial data communication bus between ECUs and diagnostic test SAE OBDII Scan Tool (SAE j1978).
The OE90C2600 provides an inteligent interface between a Host and the vehicle's Electronic control units. Both LEDs display the current communication status and connected information.
OE90C2600 is developed to meet the ISO 15031-5 standard with ISO9141,KWP2000,SAE j1850 and CAN Bus protocols in Service 1..9



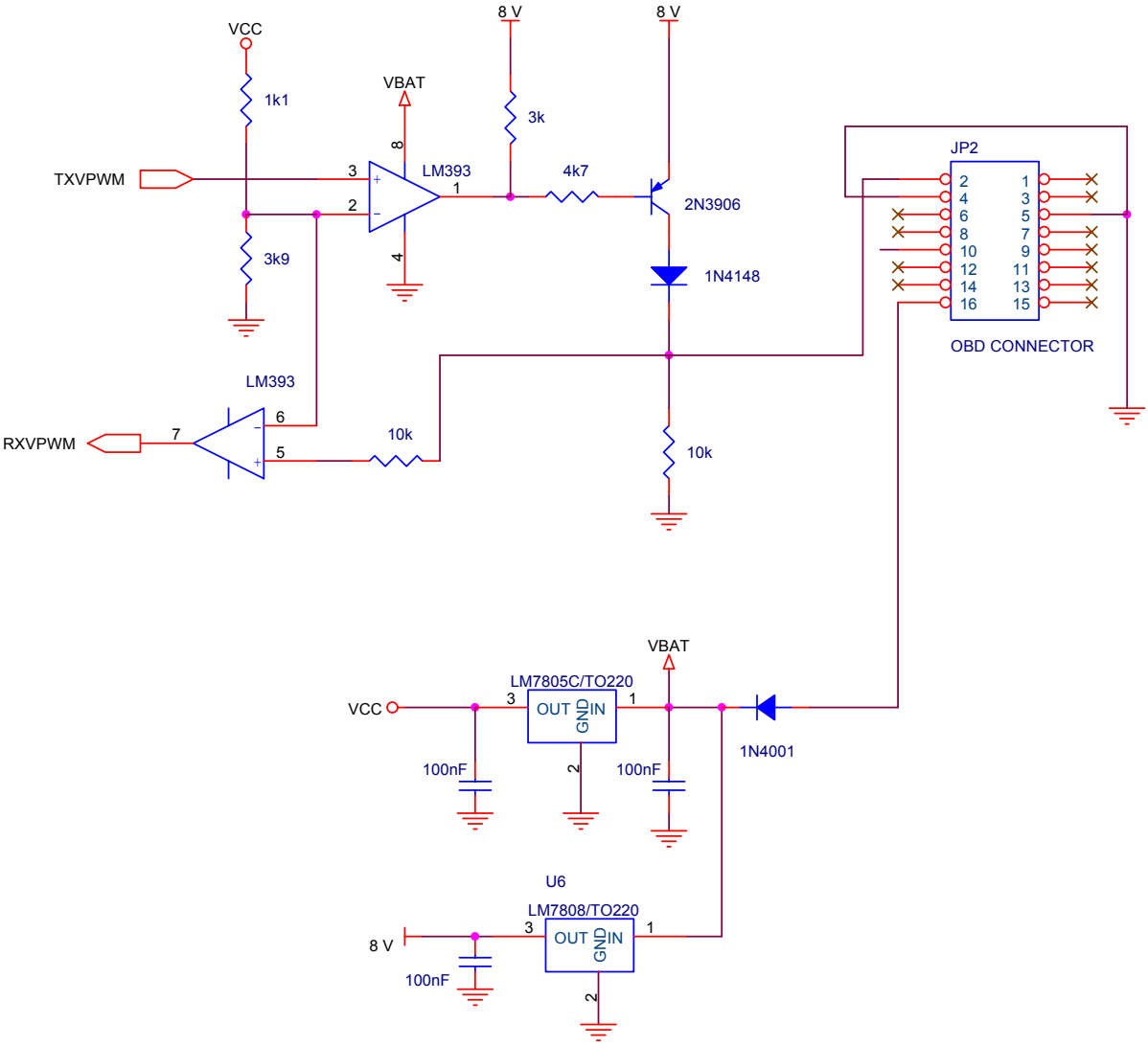
Pin description

Pin	Pin Name	Type	Description
1	NC		
2	GND		Ground
3	VCC		Supply voltage
4	RXCAN	I	CAN BUS input
5	TXCAN	O	CAN BUS output
6	LED1	O	OK LED max 5 mA for low current LED
7	TXPWM-	O	Transmit (-) output for PWM signal
8	TXPWM+	O	Transmit (+) output for PWM signal
9	BAUDIN		RS232 Baudrate input clock
10	NC		
11	NC		
12	RXPWM	I	PWM input
13	TXD	O	RS232 output
14	RXD	I	RS232 input
15	XTAL2	I	16 Mhz crystal input
16	XTAL1	I	16 MHz crystal input
17	VCC		Supply voltage
18	GND	I	Ground
19	RESET	I/O	A high level on this pin during 2 machine cycles while the oscillator is running resets the device.
20	LED2	O	LED output to indicate the frames exchange
21	NC		
22	RXK	I	ISO K-line input
23	TXL	O	ISO L-Line output
24	TXVPWM	O	VPWM transmit output
25	RXVPWM	I	VPWM receive input
26	TxK	O	ISO K-Line output
27	DOWNLD	I	A low on this pin puts the device in download mode
28	CLKOUT	O	Clock output for RS232 baud rate in

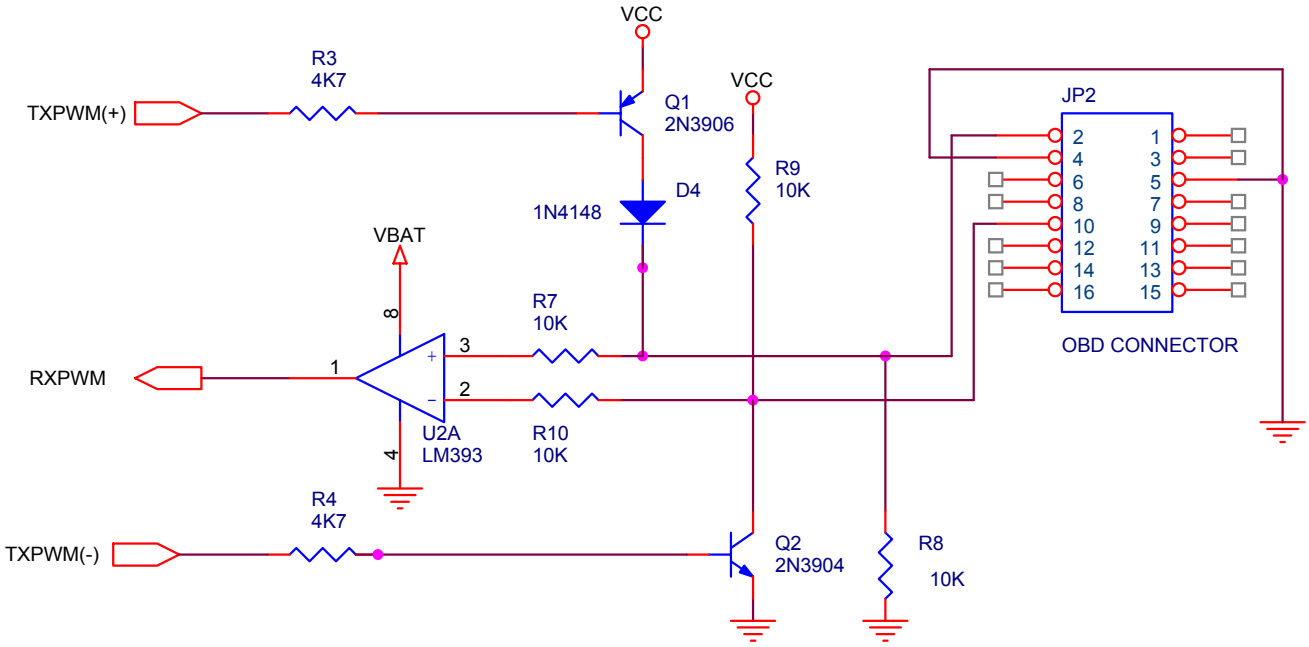
Application notes



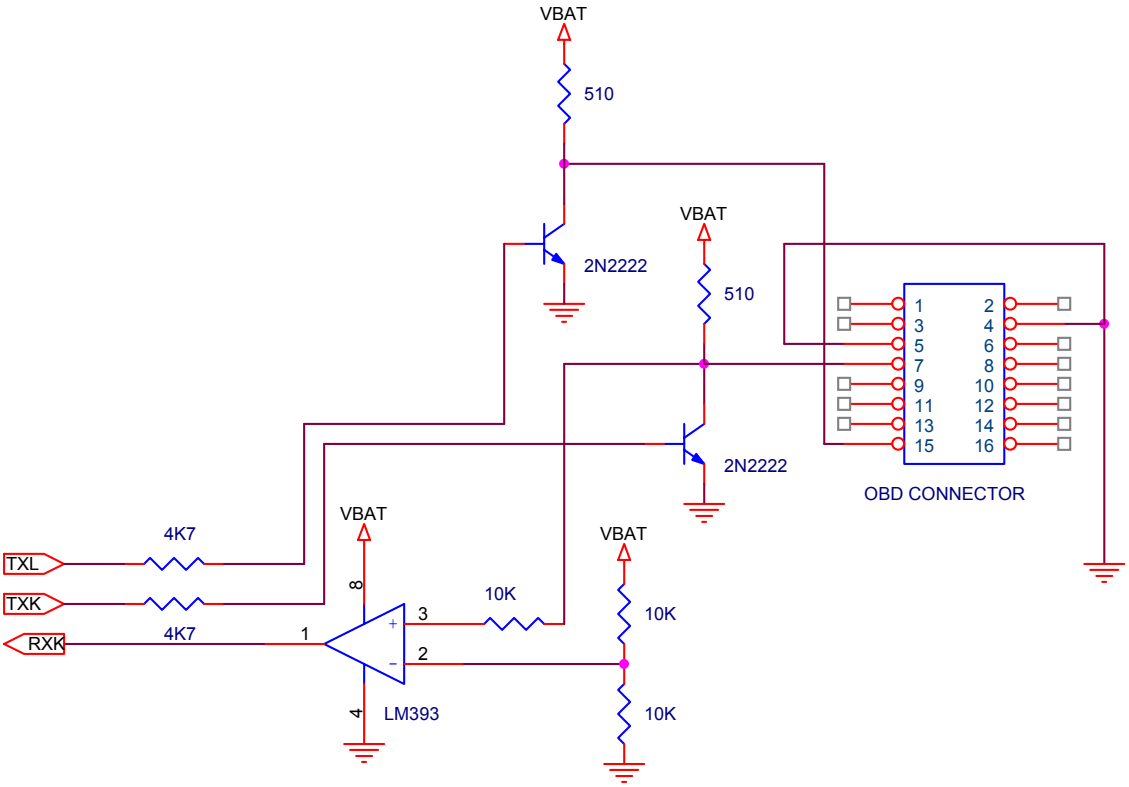
- it is recommended to use a brown out device (e.g TL7705 from TI).
- the both LEDs are low current If < 5 mA.
- close jumper to download a new release.
- Don’t change the value of crystal.
- LED yellow indicates everytime a frame exchange occurs. LED green is on if a valid protocol is found.



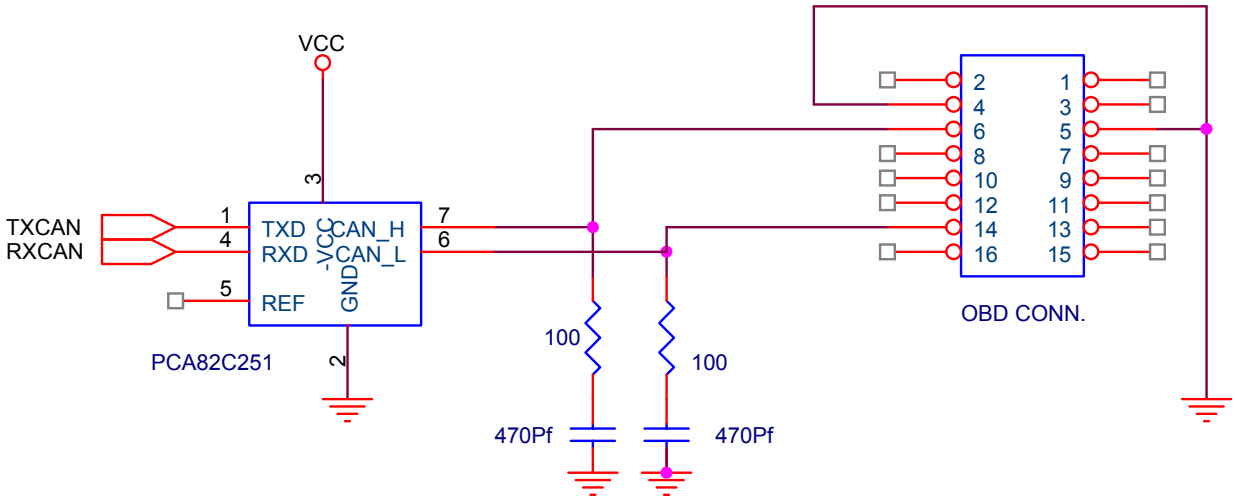
typical J1850-VPWM to RS232 interface



OBD to j1850 PWM interface



OBD to ISO interface



OBD to CAN interface

PC Serial Port Parameters

The Rs232 parameters are :

9600 Baud , 8 bits , no parity.

The PC is Master and the Adapter is slave.

Software

There are 2 types of commands for communication with the interface adapter.

High Level Commands

When sending to the adapter the service Nr. and the PID a response of 7 data bytes as specified in ISO 15031-5 is received.
For exemple:

- <STX> <03> service 3 and no pid to get the stored error codes. <STX> <02><frame> gets the freeze frame stored in ECU.
- <STX> <04> clears the DTC stored.
- <STX> <01><00> gets the PID supported and
- <STX><01><04> gets the coolant temperatur in 1.Byte in response Bytes.

Low Level Commands

In this mode the adapter can speak to ECU directly. We must first request the found protocol. For J1850-VPWM to speak to ECU for exemple we send

<length> <0x68><0x6a><0xf1><service Nr><PID>

to mobydic . thE CRC is calculated by adapter.

As a response we obtain

<length> <0x48><0X6B><ECU_adr><0x40+Service><PID> (for service 1)

Software driver

The Host software must wait at power on until a valid protocol is found. It is normally max 6 sec. The protocol can be requested with Command. 0x05. Is this number zero than is no protocol found. The host can search a valid protocol itself too. The Host must set a time out of 6 sec. for a response. **The Fonction numbers are all decimal**. But 0x30 means hexadecimal. The Mobydic has a timeout of 1000mS for a incoming Message.The unused commands return <Command-Nr> <00>

- first let the mOByDic search a valid protocol with default parameters
- use set header command to set the different ECU address
- generic command is only used when the mOByDic finds a valid protocol at power on.

Commande	00
Fonction	Reserved for system
Request	-
Pos. Response	N/A
Neg. Response	N/A

Commande	02
Fonction	OBDII/EOBD generic commands
Request	02
Pos. Response	<06>
	THEN SEND
MODE 1	<01> <PID>
MODE 2	<02> <PID>
MODE 3	<03>
MODE 4	<04>
MODE 5	<05> <TID> <O2SNO>
MODE 6	<06> <TID>
MODE 7	<07>
MODE 8	<08> <TID> <DataA><DataB><DataC><DataD><DataE>
MODE 9	<09> <INFO_Type>
Neg. Response	N/A

Commande	03
Fonction	Read serial number of mOByDic
Request	03
Pos. Response	03 HB LB HB = high byte of serial number LB = low byte of serial number
Neg. Response	N/A

Commande	04
Fonction	Read version of mOByDic
Request	04
Pos. Response	04 HB LB HB = high byte version LB = low byte of version
Neg. Response	N/A

Commande	05
Fonction	Read found protocol
Request	05
Pos. Response	05 HB LB HB = high byte of protocol LB = low byte of protocol 0x0000 : no protocol 0x0001 : ISO9141-2 keywords 08 08 0x0002 : ISO9141-2 keywords 94 94 0x0004 : KWP2000 slow init 0x0008 : KWP2000 fast init 0x0010 : J1850 PWM 0x0020 : J1850 VPWM 0x0040 : CAN 11 ident 250 KB 0x0080 : CAN 11 ident 500 KB 0x0100 : CAN 29 ident 250 KB 0x0200 : CAN 29 ident 500 KB 0x0400 : Reserved SAE J1939 0x0800 : Reserved KW1281 / KW71 0x1000 : Reserved KW82
Neg. Response	N/A

Commande	06
Fonction	Read chip ident of mOByDic
Request	06
Pos. Response	06 HB LB HB = high byte of chip ident LB = low byte of chip ident 2600 = OE90C2600
Neg. Response	N/A

Commande	07
Fonction	Connect to ECU
Request	07
Pos. Response	07 HB LB HB = high byte of found protocol LB = low byte of found protocol
Neg. Response	N/A

Commande	08
Fonction	Disconnect
Request	08
Pos. Response	08 HB LB HB = high byte of the disconnected protocol LB = low byte of the disconnected protocol ISO and KWP2000 needs yet 5 sec to disconnect
Neg. Response	N/A

Commande	09
Fonction	Get found keywords
Request	09
Pos. Response	09 KW1 KW2 this command returns the ISO/KWP read keywords
Neg. Response	N/A

Commande	10
Fonction	K-Line monitoring
Request	10
Pos. Response	mOByDic send continous data This command is inactive once after Reset
Neg. Response	N/A

Commande	12
Fonction	Send direct data to ECU ISO9141-2
Request	12
Pos. Response	<ACK> send <length><ISO message without checksum> receive <12><length><response data>
Neg. Response	N/A

Commande	13
Fonction	Send direct data to ECU KWP2000
Request	13
Pos. Response	<ACK> send <length><ISO message without checksum> receive <13><length><response data>
Neg. Response	N/A

Commande	18
Fonction	KWP2000 fast user init
Request	18
Pos. Response	<ACK> send <length><KWP message without checksum> receive <18><length><response data>
Neg. Response	N/A

Commande	19
Fonction	Stop auto keep alive
Request	19
Pos. Response	<ACK> send <KeepISO> , <KeepKWP> keepISO = 0 stop iso keep alive keepISO = 1 start ISO keep alive keepKWP = 0 stop KWP keep alive keepKWP = 1 start KWP keep alive receive <19><ACK>
Neg. Response	N/A

Commande	24
Fonction	Send direct data to ECU J1850 PWM
Request	24
Pos. Response	<ACK> send <length><PWM message without crc> receive <24><length><response data>
Neg. Response	N/A

Commande	25
Fonction	Send direct data to ECU J1850 VPWM
Request	25
Pos. Response	<ACK> send <length><VPWM message without crc> receive <25><length><response data>
Neg. Response	N/A

Commande	33
Fonction	Send direct data to ECU CAN 11/250
Request	33
Pos. Response	<ACK> send <10><CAN message without crc> receive <33><10><response data>
Neg. Response	N/A

Commande	34
Fonction	Send direct data to ECU CAN 11/500
Request	34
Pos. Response	<ACK> send <10><CAN message without crc> receive <34><10><response data>
Neg. Response	N/A

Commande	35
Fonction	Send direct data to ECU CAN 29/250
Request	35
Pos. Response	<ACK> send <12><CAN message without crc> receive <35><12><response data>
Neg. Response	N/A

Commande	36
Fonction	Send direct data to ECU CAN 29/500
Request	36
Pos. Response	<ACK> send <12><CAN message without crc> receive <36><12><response data>
Neg. Response	N/A