Features

- Compatible with EOBD/OBDII standard
- 2.7 to 6V operating range
- Direct LED drive for status display
- ISO9141
- KWP2000
- J1850
- CAN Bus
- KW71,81,82,1281
- Multiple response from multiple ECU
- Physical and fonctional addressing
- Discret or transceiver chip select

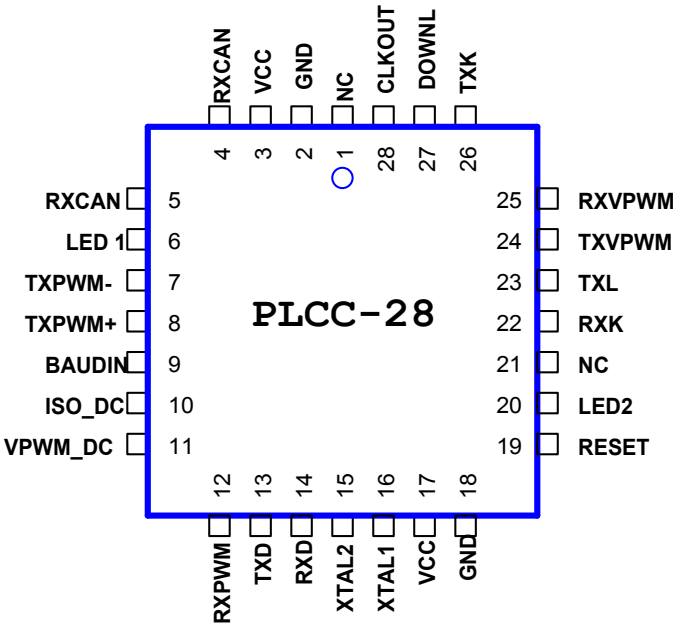
ÖZEN ELEKTRONIK

EOBD/OBDII to RS232 gateway

OE90C4000

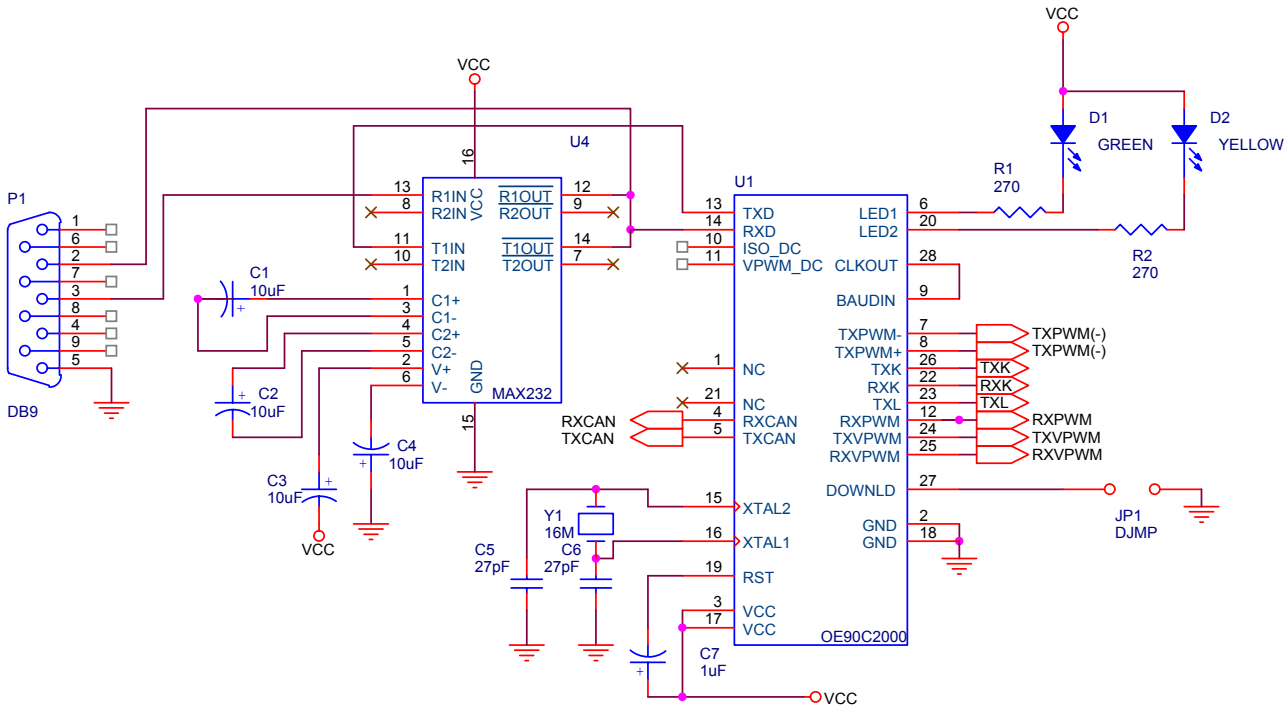
Description

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 OE90C4000 provides an inteligent interface between a Host and the vehicle's Electronic control units. Both LEDs display the current communication status and connected information.
OE90C4000 is developed to meet first the ISO 15031-5 standard with ISO9141,KWP2000,SAE j1850 and CAN Bus protocols but car specific protocols KW71 and KW1281 are implemented.

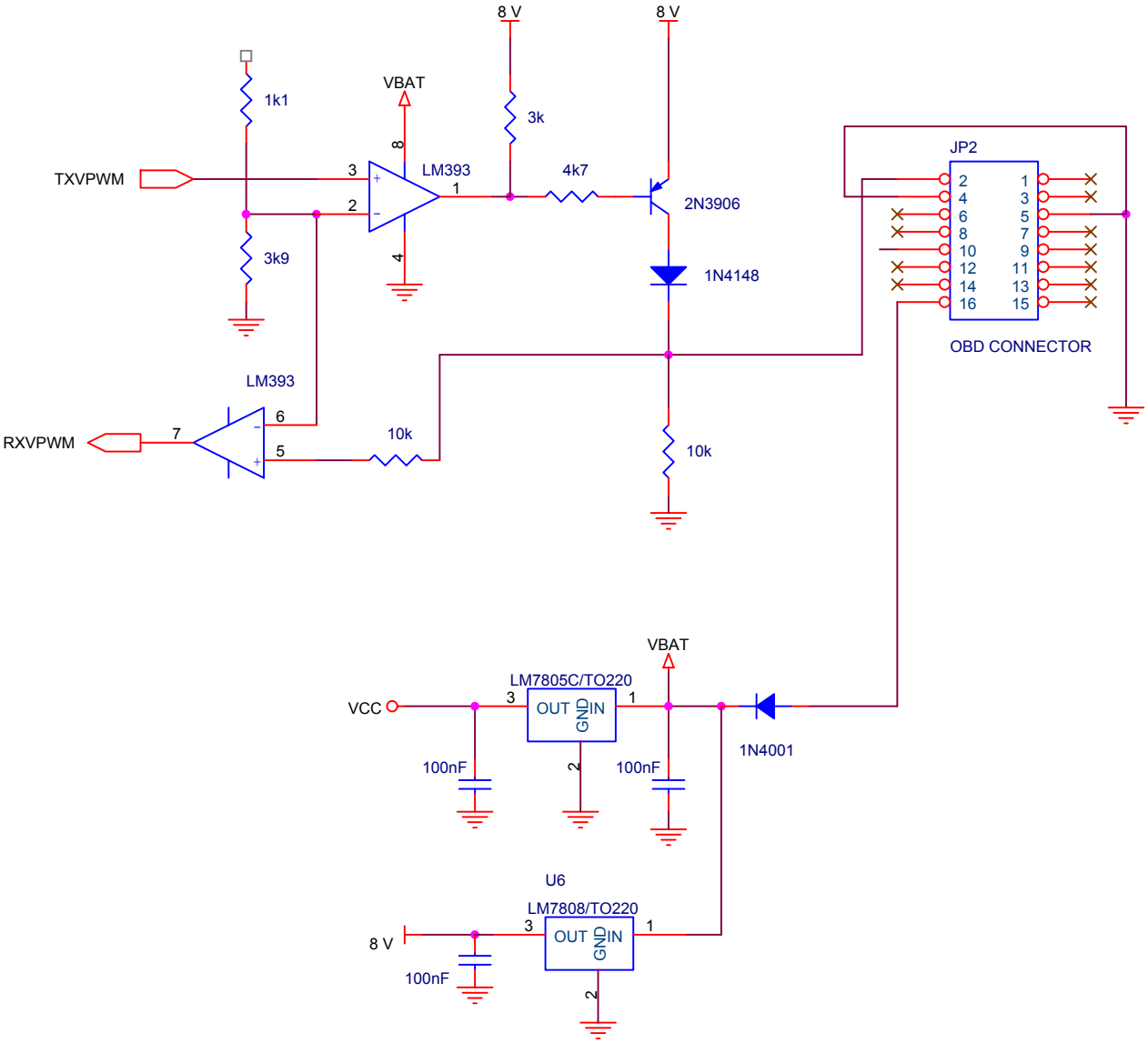


Pin description

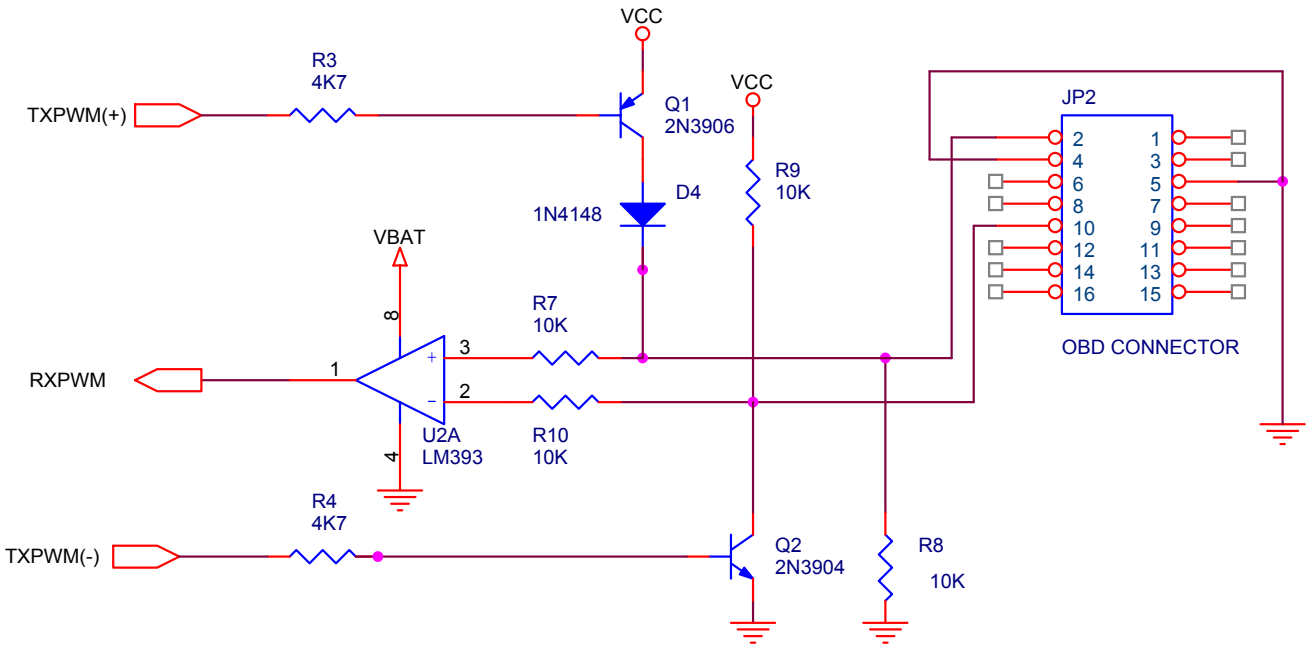
Pin	Pin Name	Type	Description
1	NC		
2	GND		Ground
3	VCC		Supply voltage
4	RXCAN	I	CAN BUS input
5	TXCAN	O	CAN BUS output
6	LED1	O	OK LED max 5 mA for low current LED
7	TXPWM-	O	Transmit (-) output for PWM signal
8	TXPWM+	O	Transmit (+) output for PWM signal
9	BAUDIN		16 x RS232 Baudrate input clock
10	ISO_DC	I	ISO discret / chip select input 1=discret
11	VPWM_DC	I	VPWM discret chip select input 1= discret
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



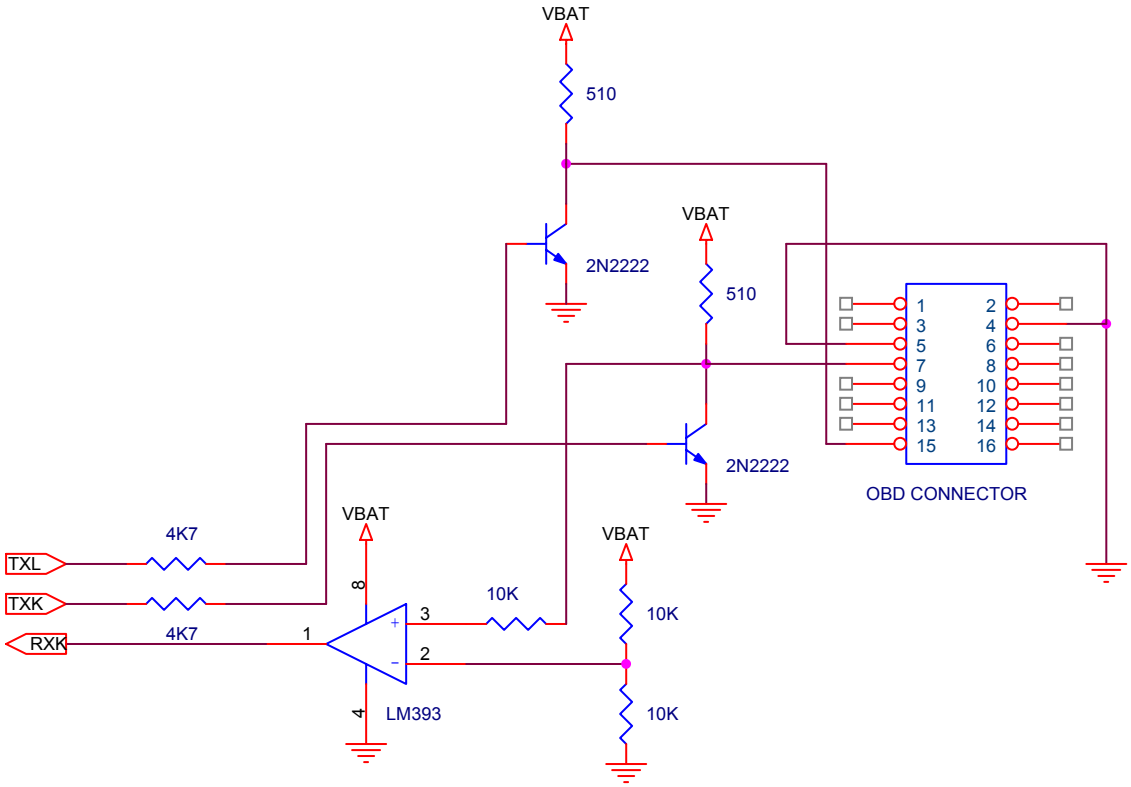
- it is recommended to use a brown out device (e.g TL7705 from TI).
- the both LEDs are low current If < 5 mA.
- close jumper to download a new release.
- Don’t change the value of crystal.
- LED yellow indicates everytime a frame exchange occurs. LED green is on if a valid protocol is found.



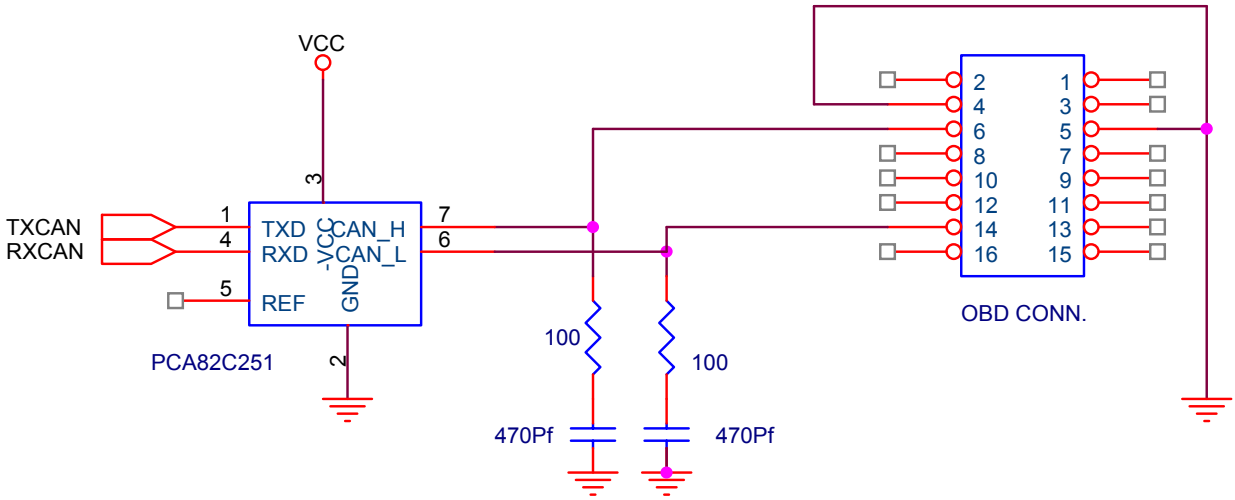
typical J1850-VPWM to RS232 interface



OBD to j1850 PWM interface



OBD to ISO interface



OBD to CAN interface

PC Serial Port Parameters

The Rs232 parameters are :

1200 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> <nack>

- 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.

Low Level Command Listing

Command : 01 Hardware Reset of Mobydic

Host sends : <01>
Mobydic sends : <01><ack>

Command : 02 Generic commands

Host sends : <stx>
Mobydic sends : <ack> or <nack>
Host sends : <service><PID>
Mobydic sends : <stx><length><user data>

Stx = 0x02
Ack = 0x06
Nack = 0x15
Length = all byte inc. CS

Service = 1 (request current powertrain diagnostic data)
2 (read freeze Frame data)
3 (read DTCs)
4 (clear DTCs)

PID = only used in service 1 and 9 . Set to 0 if not used

<User Data> = <priority/Type><Target add><source add><Data byte1..7 ><CS>

CS = summe of bytes mod 256

according to diagnostic message format of ISO 15031-5-5 page 22

Example : suppose that Mobydic found an J1850/VPWM protocol

Host sends → <Stx>
Mobidic sends → <ack>
Host sends → <01><05> read coolant temperatur (service 1 PID 5)

Mobydic sends → <stx><07><0x48><0x6b><ECUadd><0x41><0x05>
<0x64><CS>

coolant temp is 0x64 → 60 C (100 – 40)

Command : 03 Read serial number of mobydic

Host sends : <03>
Mobydic sends : <03><High Byte serial><Low Byte Serial>

Serial number = 0 for beta chip

Command : 04 Read version number of mobydic

Host sends : <04>
Mobydic sends : <04><High Byte version><Low Byte version>

Version 1.00 => 100 decimal

Command : 05 Read protocol Nr.

Host sends : <05>
Mobydic sends : <05><High Byte protocol><Low Byte protocol>

Protocol
0x0000 = no protocol
0x0020 = SAE j1850-VPWM

0xFFFF = protocol found by user

Command : 06 Tester present

Host sends : <06>
Mobydic sends : <06><ACK>

Command : 07 Read bytes from mOByDic user EEPROM

Host sends : <07>
Mobydic sends : <ACK>
Host sends : <Address EEPROM location>
Mobydic sends : <07><byte value of this location>

Address : 0..2000

Command : 08 Write byte into mOByDic user EEPROM

Host sends : <08>
Mobydic sends : <ACK>
Host sends : <Address EEPROM location> <byte to write>
Mobydic sends : <08><ack>

Command : 10 Init ISO 9141 channel

Host sends : <10>
Mobydic sends : <ack>
Host sends : <init pattern><baudrate><init_type><auto_keep_alive)

Init pattern = normally 0x33 but user can send another pattern. The user must set the even/odd parity bit .

<baudrate> → 0 = auto detect
1 = 9600 baud
2 = 10400 baud (default)

<init_type> → 0 = init according to ISO9141-2 (wait of inverted address from ECU) = default
1 = init according to ISO9141 (no wait of inverted address from ECU)

<auto_keep_alive> → 0 = no auto keep alive fonction
1 = auto keep alive performed by mOByDic. default

If succeful Mobydic sends:

<10><4><sync byte><kw1><kw2><add>

Sync byte = 0x55
Kw1 = keyword 1
Kw2 = keyword 2
Add = invers of ECU address that ECU sends

Else

<10><error code>

Error code → 0x00 = no response
0x81 = error in Sync byte
0x82 = error in kw1 or kw2

keep alive function is not performed automatically.

Example :

Host → 10
Mobydic → 06 (ack)
Host → <0x33><0x02><0x00>

Mobydic → <4><0x55><0x08><0x08><0xCC>

Command : 12 direct access to ECU via ISO 9141

Host sends : <12>
Mobydic sends : <ack>
Host sends : <length><ISO9141 string><timeout >

Mobydic sends : <12><length><ISO 9141 response string>, or
<12> <00> (for no response)

Timeout is a word variable in ms. Sent MSB first . default is 400 mS.

Functional addressing :

ISO 9141 string = <0x68><0x6a><0xf1><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CS is calculated by Mobydic.

ISO 9141 response string:

<0x48><0x6b><ECUadd><0x41><0x0d><speed><CS>

Pysical addressing :

ECU address = 0x11

ISO 9141 string = <0x68><0x11><0xf1><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CS is calculated by Mobydic

ISO 9141 response string:

<0x48><0x6b><0x11><0x41><0x0d><speed><CS>

All default timeout for communication is according to ISO 15031-5

Command : 15 K-Line Monitoring

Host sends : <15>
Mobydic sends : <15><length><Scan string>
Or <15> <00> for no response

The user can monitor the K-line . Mobydic scans the K-line continuosly at default or changed baudrate with a timeout of 6 sec.

Command : 16 Send ISO 9141 Keep alive message to ECU

Host sends : <16>
Mobydic sends : <16><ack>

Once the Mobydic is connected to ECU by a Init ISO command (10) there is a timeout of 5 sec. if during this timeout no activity is detected by ECU the communication is interrupted and a init sequence must begin by tester. Normally The user doesn't need to send this command for ISO 9141-2 because mOByDic connects automatically to ECU and a keep alive message is performed by mOByDic automatically every 3 sec..

Command : 20 Read j1850 PWM counters

Host sends : <20>
Mobydic sends : <20><length><counters>
Or <20><00> for no response

This fonction is used to see the event counters PWM pulse lengths . when a second tester is connected on PWM lines we can see the trace of PWM signal that it sends. It is used for Oscilloscope function.
Resolution is 375 nS.

Command : 21 Read j1850 VPWM counters

Host sends : <21>
Mobydic sends : <21><length><counters>
Or <21><00> for no response

This fonction is used to see the event counters VPWM ones and zeros . when a second tester is connected on VPWM line we can see the trace of VPWM signal that it sends. It is used for Osciloscope function.
Resolution is 375 nS.

Command : 22 Line monitoring j1850 PWM

Host sends : <22>
Mobydic sends : <22><length><counters>
Or <22><00> for no response

This command is the same as that of Command 20. But we can see the interpreted trace signal in j1850 format.

Command : 23 Line monitoring j1850 VPWM

Host sends : <23>
Mobydic sends : <23><length><counters>
Or <23><00> for no response

This command is the same as that of Command 21. But we can see the interpreted trace signal in J1850 format.

Command : 24 direct access to ECU via J1850 PWM

Host sends : <24>
Mobydic sends : <ack>
Host sends : <break><length><j1850 pwm string>

<break> → 0 = no break symbol send to ECU prior to transmitting
1 = a break symbol is send to ECO prior to transmitting

Mobydic sends : <24><length><j1850 pwm response string>, or
<24> <00> for no response, or
<24><0x81> format error checksum error

J1850 PWM string = <0x61><0x6a><0xf1><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic.

J1850 PWM response string:

<0x41><0x6b><ECUadd><0x41><0x0d><speed><CRC>

Command : 25 direct access to ECU in J1850 VPWM

Host sends : <25>
Mobydic sends : <ack>
Host sends : <break><length><j1850 vpwm string>

<break> → 0 = no break symbol send to ECU prior to transmitting
1 = a break symbol is send to ECO prior to transmitting

Mobydic sends : <25><length><j1850 vpwm response string>, or
<25> <00> for no response, or
<25><0x81> format error checksum error

Functional addressing :

J1850 VPWM string = <0x68><0x6a><0xf1><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

J1850 VPWM response string =
<0x48><0x6b><ECUadd><0x41><0x0d><speed><CRC>

Command : 28 Set Header Bytes PWM

Host sends : <28>
Mobydic sends : <ack>
Host sends : <HB1><HB2>

HB1 , HB2 → Header Bytes default are (0x61 , 0x6A)

Command : 29 Set Header Bytes VPWM

Host sends : <29>
Mobydic sends : <ack>
Host sends : <HB1><HB2>

HB1 , HB2 → Header Bytes default are (0x68 , 0x6A)

Command : 30 Line Scan CAN Bus 11 bit ident.

Host sends : <30>
Mobydic sends : <ack>
Host sends : <baudrate>

<Baudrate> → 0 = 250 KB
1 = 500 KB

Mobydic sends : <30><ident 1><Ident 2><8 CAN Byte>
Or <30> <00> for no response
Or <30><0x81> format error checksum error

Length byte is not used . it is always fixed to 10. ident1 is high byte and ident2 is low byte of identifier.

Command : 31 Line Scan CAN Bus 29 bit ident.

Host sends : <31>
Mobydic sends : <ack>
Host sends : <baudrate>

<Baudrate> → 0 = 250 KB
1 = 500 KB

Mobydic sends : <31><ident 1><Ident 2><ident3><ident4><8 CAN Byte>
Or <31> <00> for no response
Or <31><0x81> format error checksum error

Length byte is not used . it is always fixed to 12. ident1..4 is 29 bit data of identifier.

Command : 33 direct access to ECU in CAN 11 / 250

Host sends : <33>
Mobydic sends : <ack>
Host sends : <ident1><ident2><8 CAN data byte>

Mobydic sends : <33><ident1><ident2><8 byte fixed CAN data>
Or <33> <00> for no response
Or <33><0x81> format error checksum error

Functional addressing :

<0x07><0xdf><0x02><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 11bit / 250 baud response string =

<0x07><0xe8><03><0x41><0x0d><speed><00><00><00><00>

<0x07><0xe8> = response from ECU 1
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Physical addressing :

ECU 2 identifier = <0x07><0xe1>

<0x07><0xe1><0x02><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 11/250 KB response string =
<0x07><0xe9><0x03><0x41><0x0d><speed><CRC>

<0x07><0xe8> = response from ECU 1
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

All timeout for communication is according to ISO 15031-5

Command : 34 direct access to ECU in CAN 11 / 500

Host sends : <34>
Mobydic sends : <ack>
Host sends : <ident1><ident2><8 CAN data byte>

Mobydic sends : <34><ident1><ident2><8 byte fixed CAN data>
Or <34> <00> for no response
Or <34><0x81> format error checksum error

Functional addressing :

<0x07><0xdf><0x02><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 11bit / 500 kbaud response string =

<0x07><0xe8><03><0x41><0x0d><speed><00><00><00><00>

<0x07><0xe8> = response from ECU 1
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Physical addressing :

ECU 2 identifier = <0x07><0xe1>

<0x07><0xe1><0x02><0x01><0x0d>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 11/250 response string =
<0x07><0xe9><0x03><0x41><0x0d><speed><CRC>

<0x07><0xe8> = response from ECU 1
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

All timeout for communication is according to ISO 15031-5

Command : 35 direct access to ECU in CAN 29 / 250

Host sends : <35>
Mobydic sends : <ack>

Host sends : <ident1><ident2><ident3><ident4><8 CAN data byte>

Mobydic sends : <35><ident1><ident2><ident3><ident4><8 byte fixed CAN data>, or
<35> <00> for no response, or
<35><0x81> format error checksum error

Functional addressing :

<0x18><0xdb><0x33><0xf1><0x03><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 29 bit / 250 baud response string =
<0x18><0xda><0xf1><ecuadd><03><0x41><0x0d><speed><00><00><00><00>

<0x18><0xda><0xf1><ecuadd> = 29 bit ident
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Physical addressing :

<0x18><0xda><ECUadd><0xf1><0x03><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 29 bit / 250 Kbaud response string =
<0x18><0xda><0xf1><ecuadd><03><0x41><0x0d><speed><00><00><00><00>

<0x18><0xda><0xf1><ecuadd> = 29 bit ident
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Command : 36 direct access to ECU in CAN 29 / 500

Host sends : <36>
Mobydic sends : <ack>
Host sends : <ident1><ident2><ident3><ident4><8 CAN data byte>

Mobydic sends : <36><ident1><ident2><ident3><ident4><8 byte fixed CAN data>, or
<36> <00> for no response, or
<36><0x81> format error checksum error

Functional addressing :

<0x18><0xdb><0x33><0xf1><0x03><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 29 bit / 500 Kbaud response string =
<0x18><0xda><0xf1><ecuadd><03><0x41><0x0d><speed><00><00><00><00>

<0x18><0xda><0xf1><ecuadd> = 29 bit ident
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Physical addressing :

<0x18><0xda><ECUadd><0xf1><0x03><0x01><0x0d><00><00><00><00><00>

Service 1 PID 0d (vehicle speed request)

CRC is calculated by Mobydic

CAN 29 bit / 500 Kbaud response string =
<0x18><0xda><0xf1><ecuadd><03><0x41><0x0d><speed><00><00><00><00>

<0x18><0xda><0xf1><ecuadd> = 29 bit ident
<0x03> there are 3 used Byte in 8 Byte Data
<0x41> response to service 01
<0x0d> requested PID nr
<speed> vehicle speed data

Command : 40 Init KW71 channel

Host sends : <40>
Mobydic sends : <ack>
Host sends : <init pattern>

Init pattern = ECU address to send at 5 baud

If succeful Mobydic sends:

<40><2><kw1><kw2>

Kw1 = keyword 1
Kw2 = keyword 2

Else

<40><error code>

Error code → 0x00 = no response
0x81 = error in Sync byte
0x82 = error in kw1 or kw2

Example :

Host → 40
Mobydic → 06 (ack)
Host → <0x01>
Mobydic → <2><0xc7><0x80>

Command : 41 block send and receive in KW71

Host sends : <41>
Mobydic sends : <ack>
Host sends : <length><code><datas>

Code→ fonction code
Datas→ max 13 bytes data

Mobydic sends : <41><length><KW71 response string>, or
<41> <00> (for no response)

KW71 response string → <code><datas>

Exemple → read ADC canal

Host sends → <0x01><0x08>

MOByDic sends → <41><length><0xF8><datas>

Command : 50 Init KW1281 channel

Host sends : <50>
Mobydic sends : <ack>
Host sends : <init pattern>

Init pattern = ECU address to send at 5 baud

If succeful Mobydic sends:

<50><2><kw1><kw2>

Kw1 = keyword 1
Kw2 = keyword 2

Else

<50><error code>

Error code → 0x00 = no response
0x81 = error in Sync byte
0x82 = error in kw1 or kw2

Example :

Host → 50
Mobydic → 06 (ack)
Host → <0x01>
Mobydic → <2><0xc7><0x80>

Command : 51 block send and receive in KW1281

Host sends : <51>
Mobydic sends : <ack>
Host sends : <length><code><datas>

Code→ fonction code
Datas→

Mobydic sends : <51><length><KW1281 response string>, or
<51> <00> (for no response)

KW1281 response string → <code><datas>