INITIALIZING COMMUNICATION TO VEHICLE OBDII SYSTEM

Peter Dzhelekarski, Dimiter Alexiev

Faculty of Electronic Engineering and Technologies, Technical University of Sofia,
8 Kliment Ohridski Str., 1000 Sofia, Bulgaria, phone: +359 2 965 2622, e-mail: pid@tu-sofia.bg

This paper presents initialization of data communication link to vehicle OBDII (On-Board Diagnostics II) system using a PC-based diagnostic tester. The current work represents the physical layer of a diagnostic tester project. A concise overview of OBDII system is shown at the beginning. Diagnostic interfaces for communication with external testers are investigated. The ISO 9141-2 / ISO 14230 interface has been chosen for implementation mainly because it can be directly interfaced from a PC using simple electrical converter to RS232 (interface adapter). The circuit of the interface adapter is included and the operation principle is explained. The software part is developed using C++ language. Prior to any diagnostic communication an initialization must be performed. After successful initialization data can be transferred. The initialization and data transfer mechanisms are described in detail. Practical results from verification of diagnostic tester in real conditions are included.

Keywords: OBD, diagnostics, initialization, ISO 9141-2 and ISO 14230.

1. INTRODUCTION

This paper represents the physical layer of a PC-based diagnostic tester project. The data link layer and the application layer of the project are given in [1].

1.1 OBDII SYSTEM OVERVIEW

The purpose of an OBD (On-Board Diagnostics) system, mandatory for new vehicles, is to ensure correct operation of the emissions control system of a vehicle during its lifetime by monitoring emissions related components for deterioration and malfunction. The OBDI system was first introduced in California from model year 1991. Now the USA are using systems meeting higher standard OBDII, which was introduced from 1996 model year. Europe has developed the system EOBD (European OBD), equivalent to OBDII, which has become mandatory since model year 2001 for gasoline vehicles and 2003 for diesel vehicles. In this article, from this point forward, the OBD abbreviation refers to OBDII/EOBD.

OBD monitors the following components/systems: catalytic converters; evaporative control system; emissions control system; oxygen sensors; emissions related sensors and actuators; engine misfire; exhaust gas recirculation (EGR); fuel system – closed loop system performance, etc.

The output from OBD system is a warning light with engine symbol, presented to the driver in the instrument cluster. This is known as the malfunction indicator lamp (MIL). When a fault has been detected, a diagnostic trouble code (DTC) is set and stored in ECU (Electronic Control Unit) memory. Each DTC indicates the fault component or its circuit. The information stored within OBD system can be obtained via 16 pin data link connector (DLC) located in the passenger compartment.

Diagnostic tester (scan tool) is required to obtain and display the diagnostic information stored via serial diagnostic interface. [8]

1.2 DIAGNOSTIC INTERFACES

The diagnostic interfaces provide a communication link between OBD system and external test equipment. There are three physical diagnostic interfaces:

1) SAE J1850 (ISO 11519-4) – Class B data communication interface: two alternative physical implementations: single-wired 10.4 kbit/s VPW (Variable Pulse Width) and two-wired differential 41.6 kbit/s PWM (Pulse Width Modulation). Application: mainly GM and Ford. [9]
2) ISO 9141-2 / ISO 14230 – K-line interface: 10.4 kbit/s, single-wired interface, compatible with UART/SCI byte/word interface. The optional L-line is used only during initialization. Application: most European and Asian manufacturers, also Chrysler and GM. [4, 5]
3) ISO 15765-4 (ISO 11898) – CAN interface: high speed two-wired differential interface, 500 kbit/s. Application: will be mandatory after model year 2008. [2]

The second option ISO 9141-2 / ISO 14230 has been chosen for the current project mainly because it can be directly connected to RS 232 interface via a simple electrical converter (interface adapter) and also due to the fact that it is the most common interface in Europe.

2. PROBLEM STATEMENT

The main task of the present work is to implement the physical layer of a PC-based OBD tester. The diagnostic tester must conform to the requirements for external test equipment specified in ISO 15031-4 [7] and should use ISO 9141-2 / ISO 14230 diagnostic interface.

Objectives

1) Interface adapter implementation (hardware implementation);
2) Software implementation (PC-program):
   a. Serial port access – sending and receiving bytes via RS 232 interface;
   b. Initialization – process, performed prior to any diagnostic communication;
   c. Data transfer – sending and receiving diagnostic messages.
3) Verification.

3. INTERFACE ADAPTER

The interface adapter serves as an electrical converter between ISO 9141-2 / ISO 14230-1 (vehicle side) and RS 232 (PC side). The diagnostic interface has two lines. K-line is a bidirectional line used for data transfer. L-line is a unidirectional line which can be utilized to convey address information during initialization (not all OBD systems have an L-line). The electrical levels on K&L- lines are given in Table 1. The slop times shall be less than 10% of bit time (9.6 μs for 10.4 kbit/s). [5]

Table 1. Electrical levels of ISO 9141-2 / ISO 14230-1

	logic "1"		**logic "0"**	
$\mathbf{U_{K/L}}$	min	max	min	max

Transmitter	0.8 U_B	U_B	0.2 U_B	U_B
Receiver	0.7 U_B	U_B	0.3 U_B	U_B

$U_{K/L}$ – K/L-line voltage; U_B – vehicle battery voltage.

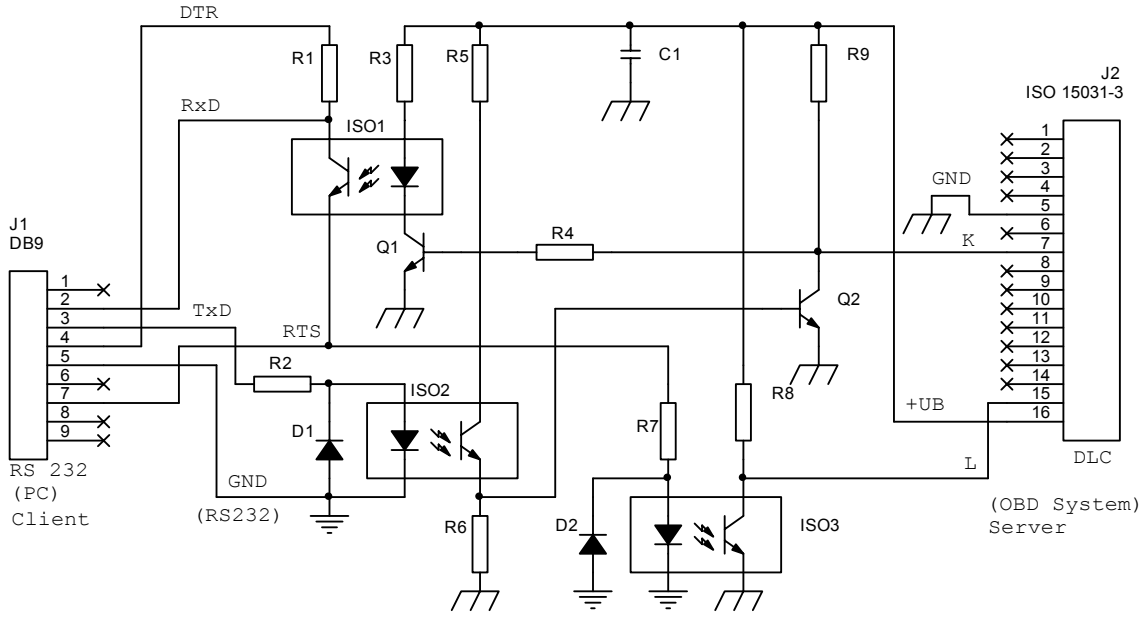


Fig. 1. Interface adapter circuit

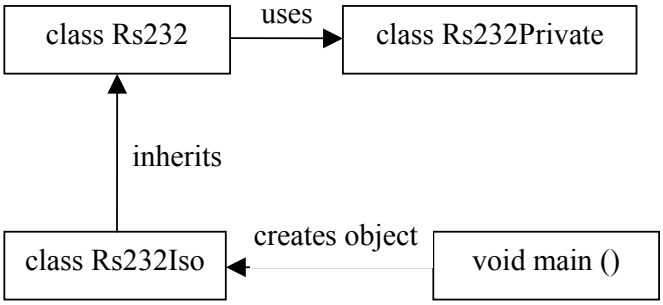
The circuit of the interface adapter (see Fig. 1) is based on Jeff Noxon's project [10]. The adapter is powered from the vehicle battery (+U_B of DLC) and from RS 232 signals. The three optrons are the heart of the adapter providing electrical level shifting with isolation. The optron *ISO2* and the transistor *Q2* are used for sending signals from *TxD* to K-line. The optron *ISO1* and the transistor *Q1* are used for receiving signals from K-line to *RxD*. The transistor of *ISO1* is powered from $\overline{DTR}$ which state must be kept set (+12 V). The optron *ISO3* is used for sending signals from $\overline{RTS}$ to L-line. In normal conditions $\overline{RTS}$ must be cleared (-12 V). Only when L-line is used, is the information submitted as an output. The diodes *D1* and *D2* protect the optron's LEDs from reverse voltages. The capacitor *C1* is used for filtering the vehicle's supply voltage. Two connectors are used, DB9 for RS232 and DLC for OBD system.

4. SOFTWARE IMPLEMENTATION

The software part of the current work is developed using C++ object-oriented language. The development tool utilized is the free command-line Borland C++ 5.5 Compiler© for Win32. The software implementation supports the following operating systems: Microsoft Windows© 98/ME/NT/2000/XP (98- and NT family). This is achieved using Win32 base API services for serial port communications (e.g. *CreateFile*, *ReadFile*, *WriteFile*, etc.). The port is opened using non-overlapped (synchronous) access. This is useful for creating portable and multithreaded applications. [3]

The program structure is shown on Fig. 2. It contains the following three classes:

- *class Rs232* provides serial port access methods for: opening/closing of port; port settings; writing/reading array of bytes with specified size; setting/clearing $\overline{DTR}$, $\overline{RTS}$ and *TxD*; setting communication timeouts.



Source files:
Rs232_Win32.cpp (1483 lines)
Rs232_Win32.h (208 lines)
main.cpp (113 lines)

Fig. 2. Program Structure

- *class Rs232Private* is hidden from users, declared in the Rs232_Win32.cpp file, used by *Rs232*, containing handle to port and other data members.
- *class Rs232Iso* inherits *Rs232* and provides methods for initialization and data transfer, namely the following:
  - *initFast ()* performs fast initialization;
  - *initSlow ()* performs 5-baud initialization;
  - *transfer ()* sends request message and receives response message(s)

5. INITIALIZATION

Prior to any diagnostic communication an initialization must be performed. This is the process of activating the OBD system for starting communication. The initialization can be started after an idle-time on the bus. Two types of initialization can be used: 5-baud and fast initialization. ISO 9141-2 supports only 5-baud initialization, whereas ISO 14230-2 supports both types. [4, 6]

5.1 5-BAUD (CARB) INITIALIZATION

This initialization starts with sending of 5-baud address byte from the tester (client) to the OBD system (server). If L-line exists, the address byte must be sent simultaneously on both K & L-lines, otherwise it should be sent only on K-line. The address byte contains the functional address of the OBD system. This is implemented in two ways in *initSlow()* method. In K&L initialization, both lines are driven by simultaneous setting/clearing of $\overline{RTS}$ and *TxD* according to the bit values of the address byte: this is computed 5-baud init. A system timer for 200 ms (5-baud bit time) is used. For K-only initialization the port is configured at 5 baud rate; the address byte is sent; 2000 ms pause follows; baud rate is restored to 10,400 baud.

Fig. 3 shows the format of the 5-baud initialization. The server returns baud rate synchronization pattern ($55) (standard baud rate 10,4 kbit/s, other rate is manufacturer specific). All subsequent communications are using this baud rate. After sending the pattern, the server sends two keywords (KW) which carry information about the form of the subsequent serial communications. From these keywords the tester identifies the diagnostic protocol to be used, ISO 14230-2 or ISO 9141-2. After

that the client confirms with sending logical inversion of keyword 2. Then the server confirms with logical inversion of the address byte. From this time on the communication link is established and the tester can start to request diagnostic data.

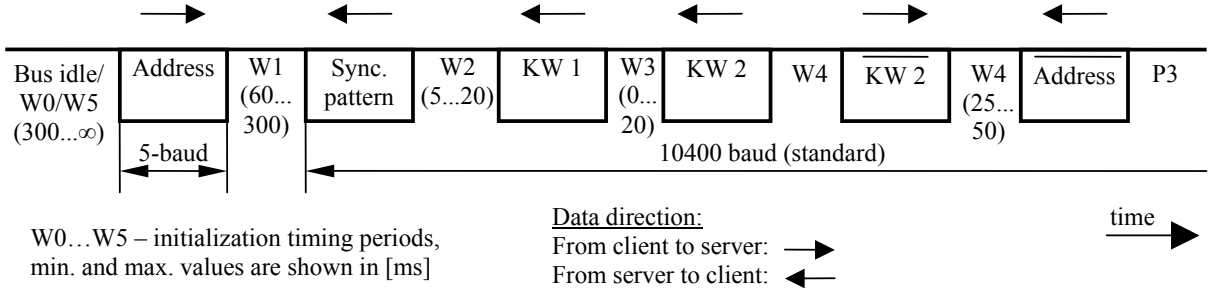


Fig. 3. 5-Baud Initialization

5.2 FAST INITIALIZATION

The fast initialization begins with wake-up pattern (WuP), transmitted by the tester simultaneously on K and L lines (if provided L line is used, otherwise transmitted only on K-line) – see Fig. 4. This is implemented in *initFast()* method by setting the *TxD* and $\overline{RTS}$, followed by waiting for T_{iniL} time interval and clearing of these signals. After the time T_{WuP} has elapsed the client sends *StartComminication* message request. This message uses functional addressing and contains the target address of the OBD system and the source address of the tester. The server responds with *StartCommunication* positive response(s), which contains the keywords.

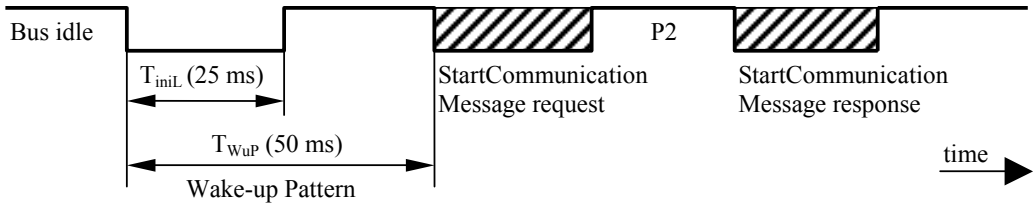


Fig. 4. Fast Initialization

6. DATA TRANSFER

After the communication link has been established, the tester can send request messages and receive message response(s) from the ECU(s) of the OBD system (see Fig. 5). This is referred to as data transfer and is performed by the *transfer()* method. The transfer timing is very important and is determined by the periods P1…P4 with specified maximal and minimal allowable values. The timing is implemented by using system timer for wait function and by using communication read timeouts. In this way it is possible to group received response bytes into messages, based on separation period P2. The period $P3_{max}$ has value of 5000 ms. After that time the communication link is automatically terminated by the server. Therefore, to keep connection alive the tester must send periodically request messages.

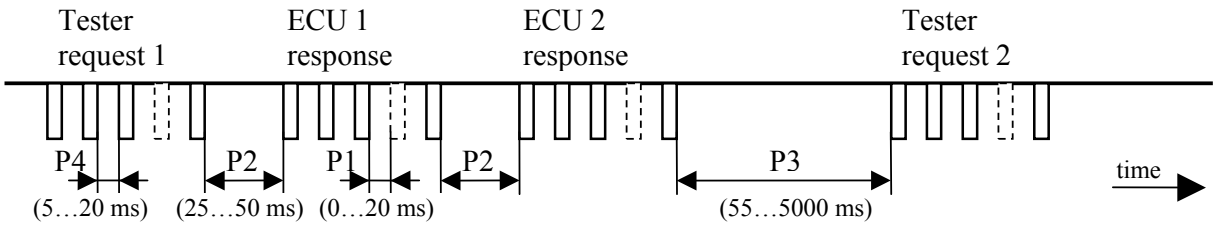


Fig. 5. Data Transfer

7. RESULTS

Bit Slop Times of Interface Adapter

The bit slope times have been measured by sending $55 pattern on K-line: rising slope time ≈ 1 μs, falling slope time ≈ 0.5 μs. The results comply with the standard.

Initializing communication

The 5-baud initialization has been successfully verified utilizing an ECU from Rover 25-LHD, model 2003. This vehicle has no L-line. The following data items has been transferred according to Fig. 3: address = $33, sync. pattern = $55, KW1 = $08, KW2 = $08, logical inversion of KW2 = $F7, logical inversion of address = $CC. The keywords inform the tester that the communication protocol is ISO 9141-2. After that, diagnostic data from the vehicle has been read and interpreted which is described in [1].

8. CONCLUSION

This paper describes the interface adapter and the initialization of communication to OBD system using a PC-based diagnostic tester. The tester utilizes the ISO 9141-2 / ISO 14230 (K-Line) interface, which is the most common diagnostic interface in Europe. The diagnostic software is written in C++. Two variants of initialization are supported: 5-baud and fast initialization. The tester has been successfully verified in practice and the results from 5-baud initialization are provided.

9. REFERENCES

[1] Dzhelekarski, P. and D. Alexiev. *Reading and interpreting diagnostic data from vehicle OBDII system*. Submitted for publication at Fourteenth Int. Conference ELECTRONICS'05, 2005.

[2] Dzhelekarski, P., V. Zerbe and D. Alexiev. *FPGA implementation of bit timing logic of CAN controller*. IEEE Proceedings, 27th Int'l Spring Seminar on Electronics Technology, 2004.

[3] *MSDN Library*. Microsoft Corporation, April 2003.

[4] *ISO 9141-2. Road vehicles – Diagnostic systems – Part 2: CARB requirements for interchange of digital information*. ISO, 1994.

[5] *ISO 14230-1. Road vehicles – Diagnostic systems – Keyword Protocol 2000 – Part 1: Physical Layer*. ISO, 1999.

[6] *ISO 14230-2. Road vehicles – Diagnostic systems – Keyword Protocol 2000 – Part 2: Data Link Layer*. ISO, 1999.

[7] *ISO/DIS 15031-4. Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 4: External Test Equipment*. ISO, 2004.

[8] Norris, J. and A. Reading. *Phase 2a report – Evaluation of the significance of OBD/OBM*. Oxfordshire, EMStec/02/026, 2002.

[9] Oliver, J. *Implementing the J1850 protocol*. Intel Corporation, 1997.

Internet

[10] http://www.planetfall.com/~jeff/obdii/ Jeff Noxon's Project: Interface RS232 (a laptop computer) to the ISO 9141-2 / SAE J1962 (OBD II) diagnostic connector. Jeff Noxon, 2003.