ON-BOARD DIAGNOSTICS SCANNER

Supervisor: Dr. Samer Arandi

Prepared by: Ahmad Saqfalhait & Morad Abushamma
Outline

- Background
- High Level View
- Protocols
- User Interface
- Storage Unit
- Hardware Implementation
- Similar System
- Limitations
- Standers
- PLC Project
Nowadays cars are very essential element in people`s life, and it`s maintenance has a great consideration. In the early 1980s a new system that helps in car maintenance has been introduced which is on-board diagnostics system “OBD” a self-diagnostic and reporting system, It gives the repair technician and the car owner the ability to access the health information of a vehicle, sensors and many parameters, this system has been evolved and now it became a mandatory requirement in car industry.
Introduction

OBD scanner standalone device that will give the user the capability to request diagnostic data from the car through an interactive interface based on appropriate communication interface, storage unit and processing unit.

OBD scanner is OBD-II standard compliant so it supports almost all cars manufactured after 1996 and this device has three main functions: view current and Freeze frame data, Read and clear generic and manufacturer specific diagnostic trouble codes and Retrieve vehicle information.
High level view

- EEPROM
- PIC Microcontroller
- Group of push buttons (System Input)
- Graphical LCD
- Communication Interface
- Engine control unit
A group of protocols should be implemented in the OBD scanner device in order to support all OBD-II compliant cars. Each protocol has physical, data link and application layer specifications.

The protocols are:

- ISO15765-4 (CAN-BUS) (mandatory for 2008+ cars)
- ISO9141-2 (ISO) (the oldest protocol)
- SAE J1850 (VPW) (GM vehicles)
- SAE J1850 (PWM) (Ford vehicles)
OBD II Cable

1 (Signal Ground)  
2 (Chassis Ground)  
3 (CAN High (J-2284))  
4 (ISO 9141-2 K Line)  
5 (CAN Low J-2284)  
6 (J1850 Bus-)  
7 (J1850 Bus+)  
8 (ISO 9141-2 L Line)  
9 (Battery Power)
ISO 9141

Message Format

<table>
<thead>
<tr>
<th>Type</th>
<th>Target Address</th>
<th>Source Address</th>
<th>Byte #1</th>
<th>Byte #2</th>
<th>Byte #3</th>
<th>Byte #4</th>
<th>Byte #5</th>
<th>Byte #6</th>
<th>Byte #7</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>68</td>
<td>6A</td>
<td>F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7 Data bytes (max)</td>
</tr>
<tr>
<td>Response</td>
<td>48</td>
<td>6B</td>
<td>ECU Physical Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7 Data bytes (max)</td>
</tr>
</tbody>
</table>

Frame structure

Start Bit "0" | bit 0 | bit 1 | bit 2 | bit 3 | bit 4 | bit 5 | bit 6 | bit 7 | Stop Bit "1"
ISO 9141 5-baud Initialization
ISO 9141 Interface

ISO9141 is an asynchronous serial data protocol with 10.4 baud rate, similar to RS232 in timing and frame structure and differs in signal levels with 12 V as logic 1 and 0 V as logic 0, one bidirectional line used to transmit and receive data known as K-Line.
ISO 14230 (KWP2000)

Message Format

<table>
<thead>
<tr>
<th>Header bytes</th>
<th>Data bytes</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format Byte</td>
<td>Target Address</td>
<td>Source Address</td>
</tr>
<tr>
<td>Request</td>
<td>11LL LLLL</td>
<td>33</td>
</tr>
<tr>
<td>Response</td>
<td>10LL LLLL</td>
<td>F1</td>
</tr>
</tbody>
</table>

Frame structure

Start Bit "0" | bit 0 | bit 1 | bit 2 | bit 3 | bit 4 | bit 5 | bit 6 | bit 7 | Stop Bit "1"

KWP2000 5-baud initialization and hardware interface same ISO 9141
KWP2000 Fast Initialization

**Wake up pattern**

<table>
<thead>
<tr>
<th>Twup</th>
<th>Tinil</th>
<th>Twup</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

**Start communication request**

<table>
<thead>
<tr>
<th>Header bytes</th>
<th>Data bytes</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format Byte</td>
<td>Target Address</td>
<td>Source Address</td>
</tr>
<tr>
<td>C1</td>
<td>33</td>
<td>F1</td>
</tr>
</tbody>
</table>

**Start communication positive response**

<table>
<thead>
<tr>
<th>Header bytes</th>
<th>Data bytes</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format Byte</td>
<td>Target Address</td>
<td>Source Address</td>
</tr>
<tr>
<td>83</td>
<td>F1</td>
<td>10</td>
</tr>
</tbody>
</table>
ISO 15765 (CAN-BUS)

Versions

<table>
<thead>
<tr>
<th>Version NO.</th>
<th>Identifier length</th>
<th>Baud rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11 bit</td>
<td>500 Kbaud</td>
</tr>
<tr>
<td>2</td>
<td>29 bit</td>
<td>500 Kbaud</td>
</tr>
<tr>
<td>3</td>
<td>11 bit</td>
<td>250 Kbaud</td>
</tr>
<tr>
<td>4</td>
<td>29 bit</td>
<td>250 Kbaud</td>
</tr>
</tbody>
</table>

CAN Protocol is mandatory for 2008+ cars

CAN Initialization

1. Start
2. Input: Baud Rate
3. Functional 11 bit CAN identifier request message
4. TX Done
5. 11 bit CAN identifier response handling
6. TX Done
7. No response
8. TX ERROR
9. 11 bit CAN ID verified
10. ISO 15765-4 compliant

11. Functional 29 bit CAN identifier request message
12. TX Done
13. 29 bit CAN ID verified
14. ISO 15765-4 compliant
15. Failed
16. No response
17. TX ERROR
18. 29 bit CAN ID verified
19. ISO 15765-4 compliant
20. Failed
21. No response
**CAN REQUEST MESSAGE TRANSMISSION**

Transmit functional request message, service $01$(supported PID’s)

- CAN Error
  - yes
  - no
  - Tx Done
    - yes
    - no
    - yes
    - no

- Succeed
- Failed

**CAN RESPONSE HANDLING**

- Start P2CAN timer
- yes
- no

- P2Can Timeout
  - yes
  - no

- Response Started
  - yes
  - no

- Receive Response
  - yes
  - no

- P2CAN timeout and all started responses completely received
  - yes
  - no

- BusyRepeatRequest negative response
  - yes
  - no

- Any other negative response
  - yes
  - no
  - yes
  - no

- Number of retries elapsed
  - yes
  - no

- Resend Request
  - yes
  - no

- Failed
  - yes
  - no

- Resend Request
  - yes
  - no

- Failed
CAN Signaling specifications

- CAN Bus has two line CANH and CANL, logic is determined by both line level:
- Logic 1: Recessive or idle bus state: CANH and CANL signals are not driven
- Logic 0: Dominant or active bus state: CANH driven high while CANL driven low
- CANH signal voltage level: 3.5V
- CANL signal voltage level: 1.5V
Can Interface
Data inquiry

Message Format

<table>
<thead>
<tr>
<th>MODE</th>
<th>PID</th>
</tr>
</thead>
</table>

Mode List

<table>
<thead>
<tr>
<th>Mode (hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Show current data</td>
</tr>
<tr>
<td>2</td>
<td>Show freeze frame data</td>
</tr>
<tr>
<td>3</td>
<td>Show stored Diagnostic Trouble Codes</td>
</tr>
<tr>
<td>4</td>
<td>Clear Diagnostic Trouble Codes and stored values</td>
</tr>
<tr>
<td>5</td>
<td>Test results, oxygen sensor monitoring (non CAN only)</td>
</tr>
<tr>
<td>6</td>
<td>Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)</td>
</tr>
<tr>
<td>7</td>
<td>Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)</td>
</tr>
<tr>
<td>8</td>
<td>Control operation of on-board component/system</td>
</tr>
<tr>
<td>9</td>
<td>Request vehicle information</td>
</tr>
<tr>
<td>0A</td>
<td>Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)</td>
</tr>
</tbody>
</table>
## MODE 1 PID List

<table>
<thead>
<tr>
<th>PID (hex)</th>
<th>bytes returned</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>PIDs supported [01 - 20]</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Freeze DTC</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Fuel system status</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Calculated engine load value</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Engine coolant temperature</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Short term fuel % trim—Bank 1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Long term fuel % trim—Bank 1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>Short term fuel % trim—Bank 2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>Long term fuel % trim—Bank 2</td>
</tr>
<tr>
<td>0A</td>
<td>1</td>
<td>Fuel pressure</td>
</tr>
<tr>
<td>0B</td>
<td>1</td>
<td>Intake manifold absolute pressure</td>
</tr>
<tr>
<td>0C</td>
<td>2</td>
<td>Engine RPM</td>
</tr>
<tr>
<td>0D</td>
<td>1</td>
<td>Vehicle speed</td>
</tr>
<tr>
<td>0E</td>
<td>1</td>
<td>Timing advance</td>
</tr>
<tr>
<td>0F</td>
<td>1</td>
<td>Intake air temperature</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>MAF air flow rate</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Throttle position</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>Commanded secondary air status</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>Oxygen sensors present</td>
</tr>
</tbody>
</table>
# OBD Error codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1102</td>
<td>O2S Heating Circuit Bank 1 Sensor 1 Voltage Too Low/Air Leak</td>
</tr>
<tr>
<td>P1105</td>
<td>O2S Heating Circuit Bank 1 Sensor 2 Short To Positive</td>
</tr>
<tr>
<td>P1107</td>
<td>O2S Heating Circuit Bank 2 Sensor 1 Short To Positive</td>
</tr>
<tr>
<td>P1110</td>
<td>O2S Heating Circuit Bank 2 Sensor 2 Short To Positive</td>
</tr>
<tr>
<td>P1113</td>
<td>O2S Sensor Heater Resistance Too High Bank 1 Sensor 1</td>
</tr>
<tr>
<td>P1115</td>
<td>O2S Sensor Heater Circuit Short To Ground Bank 1 Sensor 1</td>
</tr>
<tr>
<td>P1116</td>
<td>O2S Sensor Heater Circuit Open Bank 1 Sensor 1</td>
</tr>
<tr>
<td>P1117</td>
<td>O2S Sensor Heater Circuit Short To Ground Bank 1 Sensor 2</td>
</tr>
<tr>
<td>P1118</td>
<td>O2S Sensor Heater Circuit Open Bank 1 Sensor 2</td>
</tr>
<tr>
<td>P1127</td>
<td>Long Term Fuel Trim B1 System Too Rich</td>
</tr>
<tr>
<td>P1128</td>
<td>Long Term Fuel Trim B1 System Too Lean</td>
</tr>
<tr>
<td>P1129</td>
<td>Long Term Fuel Trim B2 System Too Rich</td>
</tr>
<tr>
<td>P1130</td>
<td>Long Term Fuel Trim B2 System Too Lean</td>
</tr>
<tr>
<td>P1136</td>
<td>Long Term Fuel Trim Add. Fuel B1 System Too Lean</td>
</tr>
<tr>
<td>P1137</td>
<td>Long Term Fuel Trim Add. Fuel B1 System Too Rich</td>
</tr>
<tr>
<td>P1138</td>
<td>Long Term Fuel Trim Add. Fuel B2 System Too Lean</td>
</tr>
</tbody>
</table>
User Interface

User interact with the device via:

- **Input:** Push Button
- **Output:** GLCD
Input: push buttons
Input: push buttons — Proteus & PCB Design
Output: Graphical LCD

We have used GLCD (with t6963c controller) to present the output for the user.

Why this GLCD?

- It supports text and graphical modes
- It supports custom character – Arabic feature.
- Available Recourse.
Output: Graphical LCD - Software

Paging and select functions.

Menu

1-Scan Device  4-History  
2-Protocols    5-Settings  
3-Manufacturers 6-About us
Output: Graphical LCD - Software

Welcome

Protocols
1-ISO
2-CAN
3-KWP 2000

1-Scan Device
2-Protocols
3-Manufacturers

4-History
5-Settings
6-About us

An-Najah National University
Made by: Ahmad Saqfalhait
Morad AbuShamma
Supervisor: Dr. Samer Arandi
Output: Graphical LCD – Custom Char

Process

- Generate array of hex values for every Arabic letter in all of its forms (connected and disconnected letter).
- Store characters’ hex values at the EEPROM.
- Character’s mapping address.

```
byte G_D[8] = {B00000, B01100, B01010, B00001, B11111, B00000, B00010, B00000};
byte G_C[8] = {B00000, B01110, B00010, B00100, B01000, B01010, B01000, B00110};
byte 8_D[8] = {B00000, B00101, B00000, B00111, B00101, B10111, B10001, B11111};
```
The whole system (input and output)
We used 24LC256 EEPROM.

We stored 3 types of data:

- Errors’ codes.
- Manufacturers.
- History (last Scan result).

Data stored at the Microcontroller’s memory.

Main Sections Table

<table>
<thead>
<tr>
<th>Sections</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>OXxxxx–OXxxxx</td>
</tr>
<tr>
<td>Manufacturers and Codes - English</td>
<td>OXxxxx–OXxxxx</td>
</tr>
<tr>
<td>Manufacturers and Codes - Arabic</td>
<td>OXxxxx–OXxxxx</td>
</tr>
<tr>
<td>Description - English</td>
<td>OXxxxx–OXxxxx</td>
</tr>
<tr>
<td>Description - Arabic</td>
<td>OXxxxx–OXxxxx</td>
</tr>
<tr>
<td>History</td>
<td>OXxxxx–OXxxxx</td>
</tr>
</tbody>
</table>
### Manufacturers Table:

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acura</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>Audi</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>BMW</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>Dodge / Chrysler / Jeep</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>Ford</td>
<td>0Xxxxx</td>
</tr>
<tr>
<td>etc</td>
<td></td>
</tr>
</tbody>
</table>

### Segments Table:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Code</td>
<td>Static, because all codes has the same length</td>
<td>5 bytes</td>
</tr>
<tr>
<td>Description Address</td>
<td>Static</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Description Length</td>
<td>Static</td>
<td>3 bytes</td>
</tr>
<tr>
<td>Description – One Arabic char.</td>
<td>Static</td>
<td>2 byte</td>
</tr>
<tr>
<td>Description</td>
<td>Variable, assigned with its code</td>
<td>-</td>
</tr>
</tbody>
</table>
Storage unit EEPROM – Proteus & PCB Design
Hardware implementation – Eagle schematic

Group of push buttons (System Input)

Power interface

PIC Microcontroller

Communication Interface

Graphical LCD Connectors A

EEPROM

Graphical LCD Connectors B
Hardware implementation – Eagle
Printed circuit design
Hardware implementation – Screenshoots
Similar systems

**D900**

Standalone OBD scanner is connected to the OBD interface on car directly and supports the five main communication protocols.
Limitations

- **Standards availability**
  - On-board diagnostics system has five common communication protocols, this protocols specifics the communication process between the OBD scanner and the car ECU. These protocols are ISO standards, not available for free and the average price for each protocol 100-200$. This problem has been solved after a long search for free documents that describe the required details about each protocol.

- **Testing environment**
  - Our testing environment is cars, and for each protocol we need different type of cars, we could not find a car ECU emulator which will ease testing process, also we faced problems in finding people who will let us make our testing on their cars.

- **Time**
  - The available period for graduation project implementation which is four months is very adequate, but unfortunately we spent two and a half months on another project (implementing home automation system through home electrical network) but we have faced many problems that forced us to change the idea so we worked on OBD Scanner project for the half of the period.
Standards/Codes

- **ISO 15765-4:2011**
  - Road vehicles -- Diagnostic communication over Controller Area Network (DoCAN) -- Part 4: Requirements for emissions-related systems

- **ISO 9141:1989**
  - Road vehicles -- Diagnostic systems -- Requirements for interchange of digital information

- **ISO 14230-4:2000**
  - Road vehicles -- Diagnostic systems -- Keyword Protocol 2000 -- Part 4: Requirements for emission-related systems

- **ISO 15031-5:2011**
  - Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 5: Emissions-related diagnostic services
OBD simulator
PLC Project

About

- A remotely controlled electrical plug system which applies functionalities like (on & off) operations, timed operations and power readings. The brilliant idea here is the communication medium, its neither WIFI nor Bluetooth which always raise the costs; the electrical signals of electricity home network are used to communicate with plugs.

Technology

- Power line carrier communication
PLC Project - power line carrier modem
PLC Project - Problems

- Printed circuit design of NCN 49597
- Special circuit components availability
- High speed printing requirements
PLC Project-PLC evaluation board

- C2000 Power Line Modem Developer's Kit ($599.00)
Thanks