Recent Developments in Electronics and Communication Systems
KVS Ramachandra Murthy et al. (Eds.)
© 2023 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/ATDE221266

Design of UDS Protocol in an Automotive Electronic Control Unit

[1]Madhumati Shridhar Kuntoji, [2]Venkatanarasimharao Medam, [3]Veena Devi S.V
RV College of Engineering Bangalore, Karnataka madhumatk.lcs20@rvce.edu.in
Chief of *Advance* Engineering, Research and Development (R&D), Greaves Electric Mobility Bangalore, Karnataka, narasimha.mv@amperevehicles.com

Abstract. In automotive, diagnostics is manditorily required to be performed on every ECU(Electronic Control Unit) to ensure that there is no any problem with the electronically controlled components of the vehicle. Today ECU is used in almost every vehicle as an electronic device. Recent automobiles consist of more than 80 ECU for different tasks such that every ECU or group of ECU will perform their dedicated functions. This paper explains about configuration of clock modules for microcontroller based on motor controller applications and also deals with the implementation of UDS (Unified Diagnostic Service) protocol according to the standard ISO 14229-1 which is being used by diagnostic system to communicate or to transfer the data between client & server.

Keywords. Electronic Control Unit (ECU), Unified Diagnostic Service (UDS), Negative Response Code (NRC), Phase Locked Loop(PLL), Controller Area Network (CAN), Diagnostic Trouble Codes (DTCs), Open System Interconnection (OSI).

1. Introduction

Diagnostic technology is important part in vehicle industry. The increasing demand in application of embedded electronics, electrical and mechanical components in vehicles brings the need to use diagnostic systems for purpose of design, operations and control of several parameters as well as industrial developments. A diagnostic system must therefore contain a standard protocol for connecting different diagnostic services, various interfacing tools in which testers, repairers use them for checking ECUs to get complete diagnosis information. Diagnostic system verifies, determines and classifies symptoms in which they aim to get an overall picture in finding out the basic cause of a problem in any vehicle which are running today. The improvement, detection, verification and communication strategies applied to irregular operations of overall system is monitored by electrical and electronic devices. Ultimately the purpose of diagnostic protocol is to identify the basic cause of abnormal operation so that particular repair can be done. Recent vehicles are supplied with a diagnostic interface, which made it possible to connect directly to a diagnostic tool to communication system of an ECU present in any vehicle and able to diagnose immediately[1].

UDS is one of the diagnostic communication protocol that used in the automotive electronic control unit. This protocol combines various standards like 1SO 14230-3, ISO 15765 and defined in detail by International organization for standardization and describes few set of conversions used for diagnostic communication between a diagnostic device and an ECU in the vehicle. So, it is important to configure clock modules which are required for motor controller design and further UDS protocol implementation is analyzed. Generally microcontroller needs clock to manage the operations and trigger their command/timings which is set by the program/user. Single microcontroller requires clock source because memory bus, peripherals clocks, timers, counters which are present all over inside the microcontroller that regulate speed at which processor executes various instructions as per requirements in S32K144 Evaluation Board.

S32K144 EVB is a 32-bit general purpose MCU family and dedicated for automotive and wide range of industrial applications. It runs at 112 MHz clock frequency and supports four clock oscillators and one Phase locked loop (PLL). First the configuration of clock modules for the microcontroller, which is based on motor controller application requirements is done.

2. System Overview

Engine Control Unit is a system which controls various parameters of an internal combustion engine.

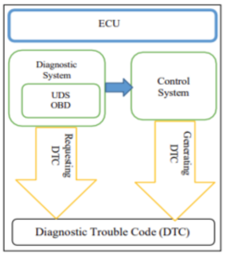

Figure 1: System Overview [2].

Application software inside an ECU is classified into two sections control system and diagnostic system as shown in Figure 1.Control system has capability to generate diagnostic trouble codes (DTCs) that provide all data of an Engine. Diagnostic system provides complete information of a DTC and control system is for further rectifying purposes. Each diagnostic system uses a different set of DTCs. Hence, UDS protocol is designed with ability to support numerous kinds of DTCs. As the engine runs various sensors check the engines parameters and when a fault is established then DTC is generated in the ECU, same problem can be resolved[2].

3. DIAGNOSIS OF ELECTRONIC CONTROL UNIT

In Electronic control unit, all the inputs and outputs need self-diagnosis ability. Traditional engines which do not have self-diagnosis capability become outdated. Inputs and outputs of an ECU are individually being monitored by controller in the system. Automotive, systems are managed by electrical and electronics components with system software that are written as per the standards of automotive embedded system requirements. Figure 2 represents the block diagram of an engine electronic control unit which is used to manage and control the automotive system. It contains analog and digital input/output signals, PWM inputs conditioning and outputs, Injection Drivers which enables interaction between other parts of the system. ECUs are connected to various sensors to get required signals corresponding to the different desired parameters of engine. ECU has full control over the combustion of fuel, cooling system and emission systems. Development of an ECU involves both hardware and software required to perform dedicated functions.

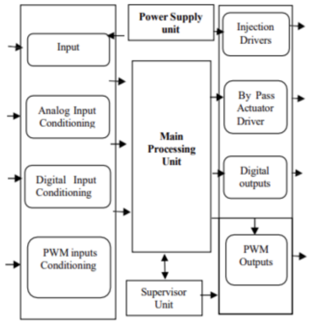

Figure 2. Block diagram of an Engine ECU

4. UDS DIAGNOSTIC SERVICES

Diagnostic Services are used to diagnose communication services such as data transmission, Input/Output operations and remote activation of routine upload/download the operations. Some certain frame format is to be followed to establish a proper communication between client and server. This protocol is used to diagnose errors and reprogram the electronic control unit. UDS works on several operating sessions, that are changed using diagnostic session control based on the active session. The commands in UDS are divided into six groups according to their functionalities and Service Identifier is represented as SID as shown in Table 1.

5. CLIENT AND SERVER BEHAVIOUR MODEL

Diagnostic Services can be diagnostic communication management requests, security/authentication access, data requests, fault code requests, IO reprogramming, etc. From Figure 3, Service Identifier is SID +data (Request) which is sent from client which acts as a tester to ECU which acts as a Server. SID +40 +Data (positive response). SID + 40 hex (Service Identifier is first byte of request). 7F SID NRC (negative response code) is negative response from server to client. Suppressed response is initiated by either client set bit suppress Indication Bit in sub-function byte or server handling a functionally addressed request. For all services using a sub-function byte, client sets a bit 7 of the sub function byte to signal that no response is necessary if the response is positive and bit is known as suppress PosRspMsg Indication Bit. When a request is sent to server it responds positively or negatively. If response is positive, tester suppresses response because as it measures inappropriate. This can be done by setting 1st bit to 1 in sub-function byte and negative responses cannot be suppressed. Left over 7 bits is used to describe up to 128 sub-function values. For example, when reading the DTC data through SID 0x19 (Read Diagnostic Information), sub-function is used to control.

Table 1. UDS Diagnostic services

Service Name	Request SID	Response SID	Functional group
Diagnostic Session Control	10	50	Diagnostic Communication Management
ECU reset	11	51	
Security access	27	67	
Communication Control	28	68	
Tester present	3E	7E	
Access timing parameters	83	C3	
Secured data transmission	84	C4	
Control DTC settings	85	C5	
Response on event	86	C6	
Link control	87	C7	
Read data by identifier	22	62	Data Transmission
Read Memory by address	23	63	
Read scaling data by Identifier	24	64	
Read data by identifier periodic	2A	6A	
Dynamically Define Data Identifer	2C	6C	
Write data by identifier	2E	6E	
Write memory by address	3D	7D	
Clear diagnostic information	14	54	Stored data transmission
Read DTC Information	19	59	
Input Output Control by identifier	2F	6F	Input/Output
Routine control	31	71	Remote activation of routine
Request Download	34	74	Upload/ Download
Request Upload	35	75	
Transfer data	36	76	
Request transfer exit	37	77	
Request file transfer	38	78	

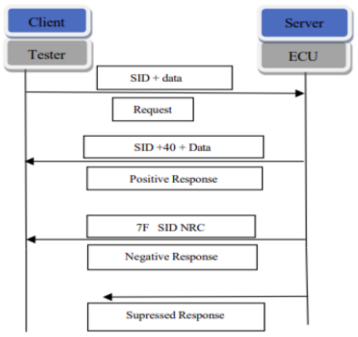

Figure 3: Request and Response behaviour between Client and Server.

Electronic control unit responds positively to an UDS request, then response frame is designed with related elements same as request frame. For example, positive response to a service 0x22 request contains response SID 0x62 (0x22 + 0x40) and 2-byte Data by Identifier, which is followed by the actual data payload for requested data identifier. It is a core component of building out ECU diagnostics and able to troubleshoot a malfunctioning ECU. Sometimes an ECU provides the negative response to an unified diagnostic service request only if the service is not supported to it.

6. WORK FLOW

Before UDS implementation, it is very necessary to configure clock modules for microcontroller based on motor controller applications. Clock configuration is the basic parameter which is must to configure in every electronic devices. In order to control the flow of data between the various building blocks of the microcontroller, one needs a clock. So, clock is needed to manage every operation of the microcontroller and clock circuit determines the speed in which microcontroller operates.

A. Configuration of clock modules and requirements

Clock is a periodic signal which oscillates between a high
and low state and manage to synchronize every action of all digital circuits and every clock signals are produced by clock generator. Clock source frequencies are summarized in Table 2. Dividers for the System Clock, Core Clock, Bus Clock and Flash Clock are first initialized before moving to another clock. System Clock Generator module controls the clock source which includes internal references, external crystals and external clocks.

Table 2: S32K144 Clock source and frequencies.

Sl. No	**Clock source**	**Frequency**
1	FIRC	48 MHz
2	SIRC	2-8 MHz
3	LPO	128 MHz
4	SPLL	90-160 MHz
5	SOSC	Between XTAL & XTAL
6	XTAL	4-8 MHz & 8-40 MHz
7	EXTAL	Upto 50 MHz

Two sample clocks are basically needed for the initialization of functions. System clock initialization is for configuring the oscillators and initial peripheral clock generator is for providing sources or dividers to the auxiliary clocks. First SOSC is initialized to 8MHz. SOSCDIV1 & SOSCDIV2 = 1 configuring XTAL oscillator for low power. Oscillator clock monitor disabled, system oscillator output clock disabled, system oscillator disabled in VLP modes whereas system oscillator disabled in stop modes.

Clock Source = 6 (SPLL_DIV2 CLK) (1)

Clock source is configured and is represented in equation 1. Configuring ports and initializing system oscillator for 8 MHz crystal. Registers control in which System clock generator is preferred then ported as clock out. System phase locked loop is clock source chosen from the source bitfield of the register therefore it is initialized to 8MHz. Ensuring SPLLCSR unlocked, SPLL(System phase locked loop) clock monitor if enabled and SPLL clock monitor disabled in stop modes and enabling SPLL = 1, whereas system clock source is enabled, whereas SPLLEN =1. Wait for SPLL valid and SIRDIV1_CLK and SIRDIV_2 = 8MHz. Changing to normal mode with 8MHz SOSC and SPLL of 80MHz, by selecting PLL as a clock source. DIVCORE =1 which is divided by 2 and Core clock=160/2 MHz = 80MHz. DIVBUS = 1, divided by 2, bus clock = 40MHz, DIVSLOW = 2, divided by 2, SCG(system clock generator) slow Flash clock = 26 2/3 MHz.

Implementation of UDS Protocol

This section deals with the implementation details of UDS protocol which is the only application layer in OSI (Open Systems Interconnection) model and takes input from CAN (Controller Area Network), a communication channel then connects CAN bus interface to OBD part of a vehicle and sends unified diagnostic service requests to it. It processes data that is stored in the message buffers and protocol will process only when it completely receives the message in a buffer to ensure correctness of the request.

Server reacts to the request and sends a respond message. UDS client is a tester that is connected to a vehicle diagnostic port and referred to as an electronic control unit and UDS services for boot flashing is shown in Figure 4.
Diagnostic session control programming session. (10 02) is programming session which is managed to upload the software. Extended Diagnostic Session is used to unlock further diagnostic functions. Safety system diagnostic session to test every safety-critical diagnostic tasks. All security-critical services are enabled by security access. In Routine control, a service is initiated in start-message. Running service is interrupted at any time with stop message. Downloading new software or additional data into control unit is done using Request Download and also involved in uploading and downloading of information. Using Transfer Exit service, a data transmission is completed. ECU reset is used to restart the control unit. Depending upon control unit hardware and interface, various kinds of reset is used such as Hard Reset, which simulates the shutdown of power supply. Key off on reset creates drain and turn on ignition with key and Soft Reset which permits the initialization of the few program units Negative Response Message is generally a three-byte message and has a negative response service identifier (0x7F) as a first byte, original SID as taken as a second byte and third byte is response code. UDS standard partially specifies response codes, if server realizes that it is not possible to perform requested service, UDS responds negatively thus gives a negative response in return. NRC contains rejection cause. Negative response message mainly consists of three sections negative response service identifier (size : 1 byte, value : 0x7F), SIDRQ (size : 1 byte) and negative response code (size : 1 byte) as shown in Figure 5.

7. RESULTS

Before implementing the UDS protocol, it is necessary to configure the clock modules as it is basic requirement for motor controller on S32K144 Evaluation Board, that is executed on S32 design studio software. Therefore, first clock modules outputs are described then UDS implementation outputs are analysed. SPLL is being initialized and disabled and waited to be valid. First shall calculate the value of Voltage Controlled Oscillator (VCO_CLK) to get value of SPLL_CLK as shown in equation 2 and 3. Substituting the value of SPLL_SOURCE is taken as 8MHz and PREDIV lies between 000 to 111. 000 is considered as 0 and MULTI is taken as 24 which is Multiplying the system by PLL.

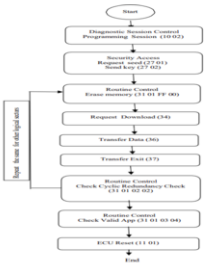
Start
Diagnostic Session Control Programming Session (10 02)
Security Access Request seed (27 01) Send key (27 02)
Routine Control Erase memory (31 01 FF 00)
Request Download (34)
Transfer Data (36)
Transfer Exit (37)
Routine Control Check Cyclic Redundancy Check (31 01 02 02)
Routine Control Check Valid App (31 01 03 04)
ECU Reset (11 01)
End
Repeat the same for other logical sectors

Figure 4: UDS services for Boot Flashing

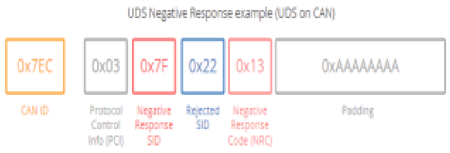
UDS Negative Response example (UDS on CAN)
0x7EC
0x03
0x7F
0x22
0x13
0xAAAAAAAA
CAN ID
Protocol Control Info (PCI)
Negative Response SID
Rejected SID
Negative Response Code (NRC)
Padding

Figure 5: UDS Negative Response example (UDS on CAN)

$$VCO\_CLK = [SPLL\_SRC / (PREDIV+1)] \times (MULTI+16) \quad (2)$$

$$VCO\_CLK = [8 / (0+1)] \times (24+16)$$

$$VCO\_CLK = [8] \times 40$$

$$VCO\_CLK = 320 MHz \quad (3)$$

Substituting the value of VCO_CLK from equation 3 into equation 4 to get the value of SPLL_CLK.

$$SPLL\_CLK = (VCO\_CLK / 2) \quad (4)$$

$$SPLL\_CLK = (320 / 2)$$

$$SPLL\_CLK = 160 MHz \quad (5)$$

From the Equation 4 and 5, it is observed that SPLL_CLK initialization is 160 MHz hence it is noted and the system clock output are shown in Table 3. Now, UDS protocol generated outputs are analyzed showing Positive Response and Negative Response. 18DA58F1 is a client ID (hex) and 18DAF158 is a server ID (hex) shown in detail in Table 4.

Table 3: System Clock Output

	Divisions	Clock	Frequency
DIVCORE=1	div by 2	Core Clock	80MHz
DIV BUS	div by 2	Bus Clock	40MHz
DIV SLOW	div by 2	Flash Clock	26 2/3 MHz
SPLLDIV 1	div by 1	SPLL Clock	160 MHz

Table 4: UDS Generated Output

Time offset (msec)	ID (hex)	Data bytes in hex	Service name
1786.6	18DA58F1	02 10 02 55 55 55 55 55	Diagnostic Session Control
1787.0	18DAF158	03 7F 10 78 00 00 00 00	
1806.3	18DAF158	06 50 02 00 32 01 F4 00	
1811.2	18DA58F1	02 27 01 55 55 55 55 55	Security Access
1811.5	18DAF158	06 67 01 12 34 56 78 00	
1815.1	18DA58F1	06 27 02 14 32 50 7E 55	
1815.5	18DAF158	02 67 02 00 00 00 00 00	
1819.9	18DA58F1	10 08 31 01 FF 00 00 00	Routine Control
1820.2	18DAF158	30 00 01 00 00 00 00 00	
1822.6	18DA58F1	21 10 00 55 55 55 55 55	
1830.5	18DAF158	04 71 01 FF 00 00 00 00	
1838.2	18DA58F1	10 09 34 00 24 00 00 10	Request Download
1838.5	18DAF158	30 00 01 00 00 00 00 00	
1840.9	18DA58F1	21 00 10 00 55 55 55 55	
1841.2	18DAF158	04 74 20 08 02 00 00 00	
1845.3	18DA58F1	18 02 36 01 00 70 00 20	Transfer data
2676.8	18DAF158	02 76 01 00 00 00 00 00	
2686.5	18DA58F1	18 02 36 02 FF FF FF FF	
3510.8	18DAF158	02 76 02 00 00 00 00 00	
3515.6	18DA58F1	01 37 55 55 55 55 55 55	Request Transfer Exit
3515.9	18DAF158	03 7F 37 78 00 00 00 00	
3516.2	18DAF158	01 77 00 00 00 00 00 00	
3550.7	18DA58F1	22 4B 03 B1 18 47 70 47	Read data by Identifier
13713.0	18DA58F1	02 11 03 55 55 55 55 55	ECU Reset
13713.4	18DAF158	02 51 03 00 00 00 00 00	

This Time offset is shown in the left side in m sec and data length is 8 bits. Data bytes are represented in hex values. From example 1, Client ID 18DA58F1, whose data bytes is 02 10 01 55 55 55 55 55, server displayed 7F 10 78 00 00 00 00 is received-response pending. NRC(Negative Response Codes) implies that the request message has been received accurately and every parameters in request message are valid but the action performance is not completed and the server is not prepared to receive another. As soon as requested service completes, the server sends positive response or negative response message with a response code distinct from this , 40 is added to 10 and displays 50 (0x10 +0x40) i.e 50 02 00 32 01 in 18DA58F1 and confirms the positive response. From example 2, 18DA58F1 displaying 02 27 01 55 55 55 55 55 and 40 is added to server ID 06 67 01 12 34 56 78 00 i.e 27 becomes 60 since positive response to service 0x27 request consists of response SID 0x67(0x27 + 0x40), hence shows the corresponding positive response as shown in Table 4.

8. CONCLUSION

This paper discusses & proposes the method of implementation of UDS protocol according to the standard ISO 14229-1 and describes how it can be developed in various embedded software applications. This method would improve the development efficiency of UDS protocol and reduces the workload of developers. The implementation of embedded electronic circuits, use of communication protocols and standards which has become important to ensure the proper functioning, monitoring and also working of automotive systems in daily life.

REFERENCES

[1] S. Dekanic, R. Grbic, T. Maruna and I. Kolak, "Integration of CAN Bus Drivers and UDS on Aurix Platform," 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), 2018, pp. 39-42.

[2] Panuwat Assawinijaipetch, Michael Heeg, Daniel Gross, "Unified Diagnostic Services Protocol Implementation in an Engine Control Unit, 2013.

[3] Muneeswaran. A "Automotive Diagnostics Communication Protocols Analysis- KWP2000, CAN and UDS", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), pp 20- 31.

[4] M. Wajape and N. B. Elamana, "Study of ISO 14229-1 and ISO 15765-3 and implementation in EMS ECU for EEPROM for UDS application," 2014 IEEE International Conference on Vehicular Electronics and Safety, 2014, pp. 168-173.

[5] Peti, Philipp, et al. "A quantitative study on automatic validation of the diagnostic services of Electronic Control Units". Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference.

[6] M. Matsubayashi et al., "Attacks Against UDS on DoIP by Exploiting Diagnostic Communications and Their Countermeasures," 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring).

[7] S. Bidkar, S. L. Patil and P. Shinde, "Virtual ECU Development for Vehicle Diagnostics Software Testing using UDS Protocol," 2021 Asian Conference on Innovation in Technology (ASIANCON), 2021, pp. 1-6.

[8] S. R. Nayak and A. Bagubali, "Study on Diagnosis of Automotive Network," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 2019, pp. 1-6.

[9] F. Luo and Q. Wen, "Implementation of bootloader based on DoIP," 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET), 2019, pp. 239-244.

[10] S. Nautch, "A serial Bootloader with IDE extension tools design and implementation technique based on rapid embedded firmware development for developers," *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2017, pp. 1865-1869.