UDSonCAN – a diagnostic protocol for various CAN applications

The Unified Diagnostic Services (UDS) are standardized in the ISO 14229 series. Part 3 specifies the CAN-mappable ISO-TP transport protocol standardized in ISO 15765-2. This approach has been adapted for J1939 and CANopen, too.

Sophisticated diagnostics is an important feature for any kind of vehicles. There is a growing need to incorporate diagnostic measures into CAN-connectable devices. Some CAN-based higher-layer protocols such as SAE J1939 specify a set of specific diagnostic messages, which cover different use cases, e.g. reporting of active Diagnostic Trouble Codes (DTCs) or control of communication behavior (such as for example 'stop broadcast messages'). In CANopen, Emergency (EMCY) messages can be used to publish diagnostic information and devices may keep error history objects, which can be read later. Finally, OBD-II is well known in the automotive industry for diagnostics purposes.

And, there are the Unified Diagnostic Services (UDS) defined in the ISO 14229 standard series. In the meantime, the UDS approach is used in different industries. Albeit, UDS can be used on different communication protocols such as LIN and Ethernet, this article focuses on UDSonCAN as specified in ISO 14229-3:2022. Classical CAN data frames have a limited data field of 8 byte and the CAN FD data frame is limited to 64 byte. But some UDS services require more payload. In this case, a transport layer protocol is needed, which segments the diagnostic message on the transmitter side and re-assembles the segments on the receiver side.

The ISO-TP transport layer protocol

ISO has standardized a CAN-based transport protocol in ISO 15765-2 (also known as ISO-TP), which is currently under revision. It supports classical CAN and CAN FD, accommodating payloads of up to 4 GiB. However, for the sake of simplicity, the article will focus on the explanation of the traditional ISO-TP protocol, which allows for payloads of up to 4095 byte on classical CAN.

There are four segment types specified:

- Single Frame (SF) to transfer data up to 7 bytes,
- First Frame (FF) to initiate a multiple-segment transmission,
- Flow Control (FC) frame to control multiple-segment transmissions,
- and Consecutive Frames (CF) that contain the payload of multiple-segment transmissions.

ISO-TP specifies four addressing formats

Normal Addressing arbitrary with 11-bit or 29-bit CAN-IDs can be used to designate the producer and receiver of the messages. Normal Fixed Addressing supports only 29-bit CAN-IDs, where some bits (28 to 26 and 23 to 16) are ▷

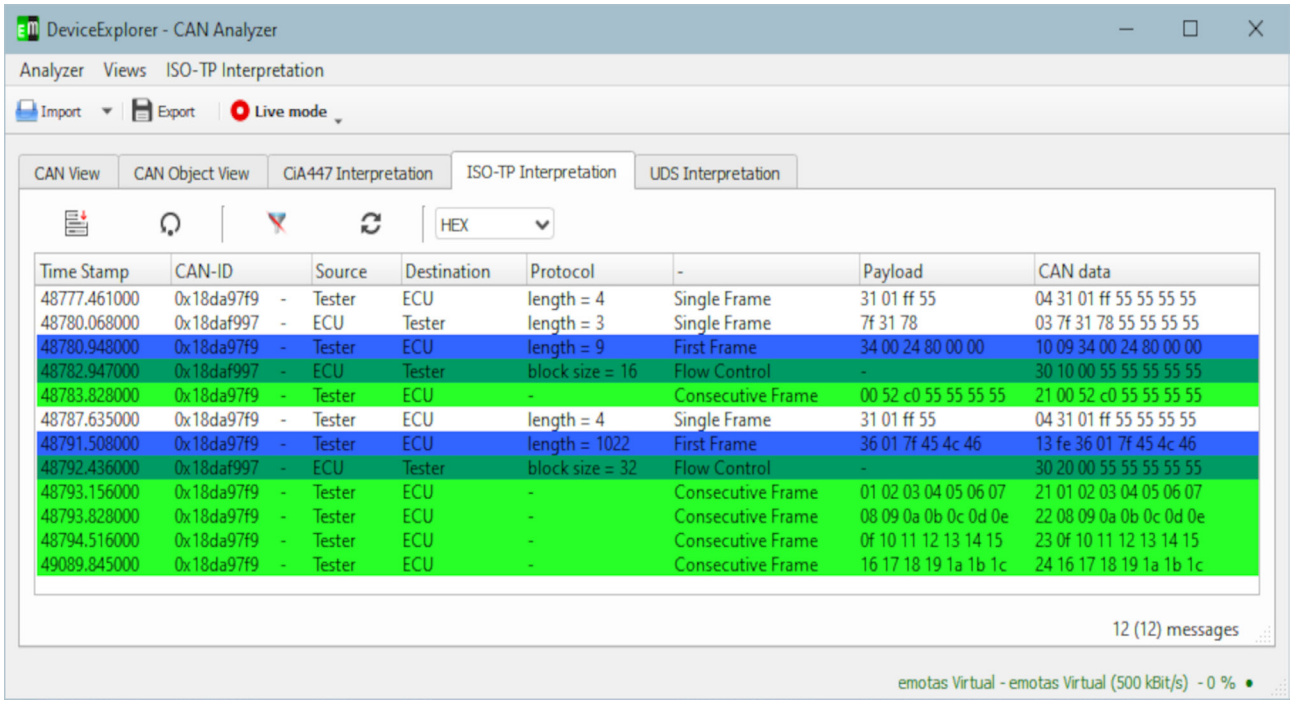

Figure 1: ISO-TP interpretation showing the four segment types using the DeviceExplorer tool (Source: Emotas)

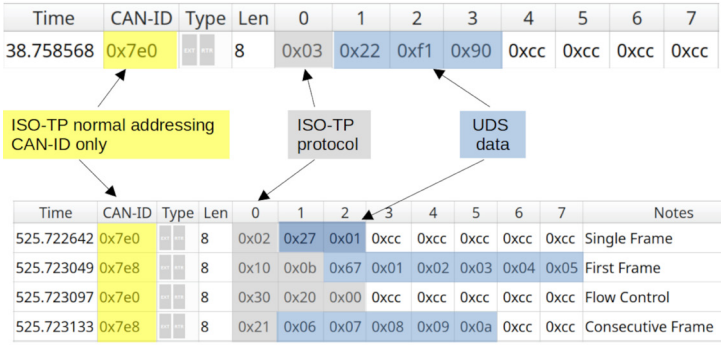

Figure 2: UDS data embedded in ISO-TP (Source: Emotas)

pre-defined and both the source address and target address are coded into the CAN-ID. Extended Addressing uses the first data byte of the CAN data frame to address devices, so that the possible payload is reduced but less CAN-IDs are required. Finally, Mixed Addressing is an additional address scheme for remote diagnostics.

ISO-TP is used by UDS to transfer the UDS data, but the usage of the different ISO-TP addressing schemes and the CAN-IDs is usually defined by the application layer protocol of the network. Using J1939 with UDS requires Normal Fixed Addressing with 29-bit CAN-IDs mapped to specific J1939 Parameter Groups (PG): DIAG1, DIAG2, DIAG3, and DIAG4. They are specified in detail in ISO 15765-2, the SAE J1939DA (digital annex) defines just the Parameter Group Numbers (PGN) and references to the ISO document.

UDS solutions for different platforms

Emotas offers UDS server and UDS client solutions. These software packages are available as extensions for the company's CANopen, J1939, and RawCAN protocol stacks. In other words, the company does not offer a separate UDS server stack. The UDS server is integrated into the company's protocol stacks. The same is possible with UDS client (tester) functionalities. ISO-TP with all mentioned addressing methods is also included. Both the UDS server and the UDS client extension support a set of UDS services. The API (application programming interface) covers both service-specific functions and higher functionalities that include multiple UDS services into a comprehensive procedure (e.g. to do a download of a complete firmware via UDS).

In addition to the stack extensions and UDS boot loaders, the provider offers a couple of UDS extensions for its tool chain that is available for Windows and Linux. Plug-ins for both the CANinterpreter and the CANopen DeviceExplorer tools allow to interpret CAN data frames according to the ISO-TP and UDS specification. Additionally, a 'UDS transmitter' provides functions to send single UDS commands or a sequence of them. The tools' scripting engine provides also a set of UDS-specific commands to implement customer-specific UDS sequences for testing purposes.

Time Stamp	CAN-ID	Source	Protocol	-	Payload	Interpretation
49401.245000	0x7e0	Tool	Request	DiagnosticSessionControl	10 02	programmingSession
49402.819000	0x7e8	Bootloader	Response	DiagnosticSessionControl	50 02 00 c8 01 f4	programmingSession 200 ms, 5 s

Figure 3: Request and response of the DiagnosticSessionControl service (Source: Emotas)

Time Stamp	CAN-ID	Source	Protocol	-	Payload	Interpretation
50503.084000	0x18da97f9	Tester	Request	ReadDataByIdentifier	22 f1 90	0xf190: VIN
50503.096000	0x18daf997	ECU_Device	Response	ReadDataByIdentifier	62 f1 90 65 6d 6f 74 61 73 20 42 6f 6f .. (+7)	

Figure 4: Request and response of the ReadDataByIdentifier service (Source: Emotas)

Time Stamp	CAN-ID	Source	Protocol	-	Payload	Interpretation
49689.810000	0x7e0	Tool	Request	SecurityAccess	27 01	requestSeed
49689.812000	0x7e8	Bootloader	Response	SecurityAccess	67 01 00 c8 01 f4	requestSeed
49689.814000	0x7e0	Tool	Request	SecurityAccess	27 02 af be 23 00	sendKey
49689.816000	0x7e8	Bootloader	Response	SecurityAccess	67 02	sendKey

Figure 5: Request seed and send key using the SecurityAccess service (Source: Emotas)

In the CiA 447 CANopen profile for special-purpose car add-on devices, Normal Addressing with 11-bit CAN-IDs is used. The specified CAN-IDs cannot be used for generic CANopen functions. Other higher-layer protocols or UDS in proprietary networks need to specify the desired and appropriate addressing.

The Unified Diagnostic Services

UDSonCAN provides a set of different services to read or to write data, to modify outputs, to manage access rights, to change communication behavior, and to download data to ECUs (electronic control units) in a CAN network.

Normally, UDS services are initiated as requests from a tester and they are usually responded by the ECU. In some cases, it is possible to suppress positive responses, which do not contain any additional data. Each UDS service has a 1-byte service-ID and the response replies with (service-ID + 40_h) in the first byte of the response. All additional bytes of both request and response depend on the actual service.

A detailed explanation of all UDS services would exceed the scope of the article. Instead, the focus is on a summary of four selected UDS services.

DiagnosticSessionControl (10_h) service

This service is used for transitions between multiple sessions. There are different sessions defined in UDS and a device always starts in the default session, in which only a subset of services is supported. Using the DiagnosticSessionControl service, the tester can command the ECU into other sessions such as Programming Session, Extended Diagnostic Sessions, or various implementation-specific sessions. Each session grants access to a specific set of supported services and actions tailored to its purpose. In the absence of UDS communication with this device, the ECU will automatically revert to the default session after a specific timeout period.

ReadDataByIdentifier (22_h) service

This service can be used to query the value of certain identifiers. One notable example is $F190_h$ for the Vehicle Identification Number (VIN). The identifier is a 16-bit value and approximately 200 of these identifiers are pre-defined by UDS and thus there is a large free range of manufacturer-specific identifiers. The service also allows reading multiple identifiers by one request.

SecurityAccess (27_h) service

This servcie provides the functionality to enter dedicated security levels and thus unlock restricted services. The process is based on a challenge-response authentication scheme:

1. The tester requests a seed from the ECU and the ECU generates and sends a seed – a random challenge value back to the tester.
2. The tester calculates a key value based on the provided seed. The algorithm is manufacturer-specific and there can be different algorithms for different security levels. The tester sends the key to the ECU.
3. The ECU verifies the key and if the key matches, the ECU grants access to the requested security level and unlocks the associated services.

TransferData (36_h) service

This service is used to transfer larger chunks of data from or to a device. It is started with the RequestDownload (34_h) service and is finished with the RequestTransferExit (37_h). One TransferData block can contain up to 4093 byte, but the length is defined by the ECU in the response to the RequestDownload request. Multiple TransferData blocks are used to transfer data that exceed the block size of one TransferData block. ◀

Author

Torsten Gedenk
Emotas Embedded Communication
ged@emotas.de
www.emotas.de