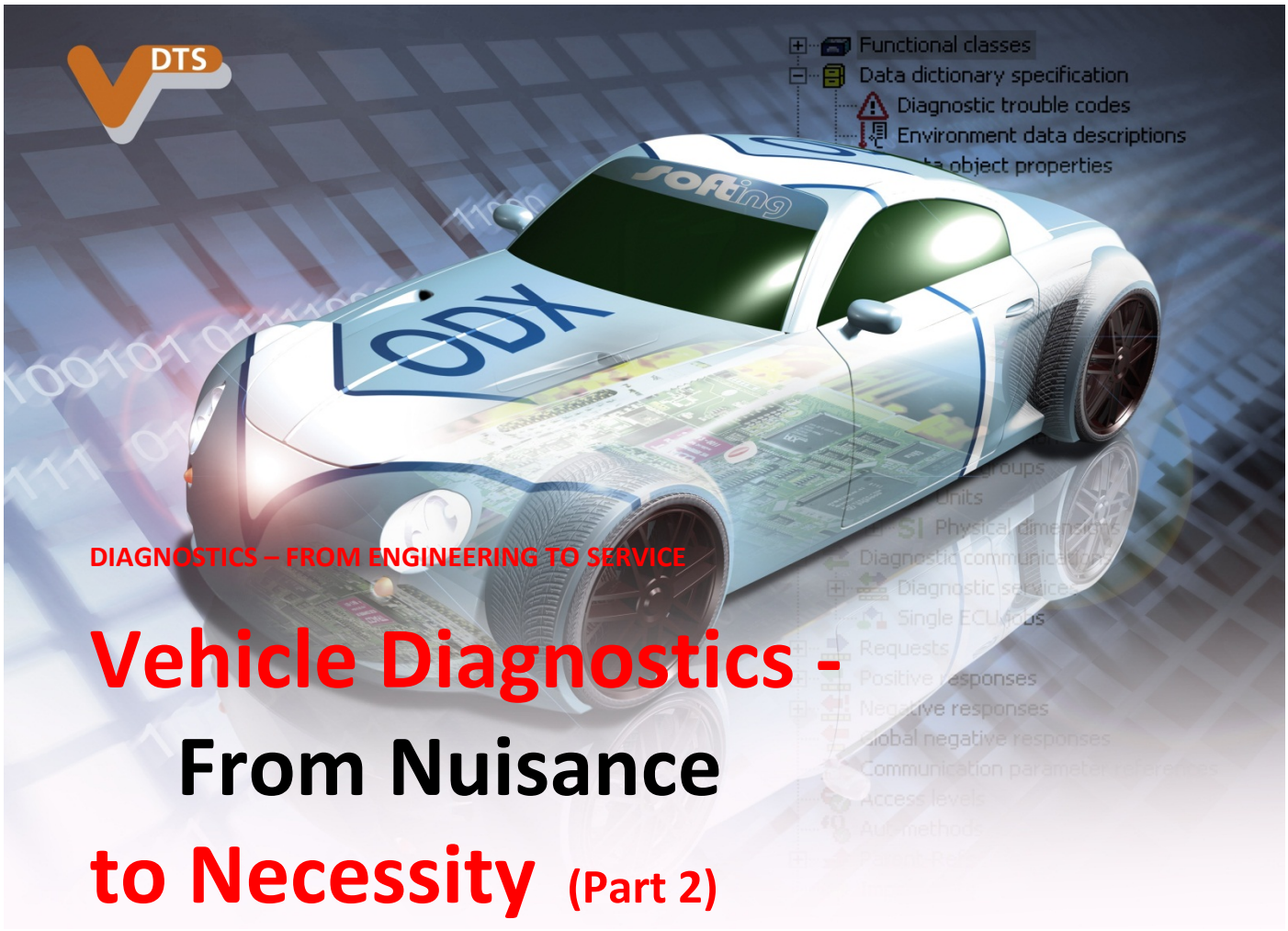

DIAGNOSTICS – FROM ENGINEERING TO SERVICE

Vehicle Diagnostics - From Nuisance to Necessity (Part 2)

For some years now, the vehicle diagnostics is undergoing a process of rapid change. Diagnostics thus constitutes an individual class of requirements for the vehicle per se; these must be implemented parallel to the intrinsic functions. In the second part of this article the different protocols and programming interfaces will be shown. Finally, at the example of Softing's DTS-Monaco an universal engineering tester will be introduced which is based on D-PDU API, ODX, MCD-3D and OTX.

The communication protocol UDS

Today, the most common protocols in Europe are ISO 14765 (KWP2000 on CAN) and, based on the former, ISO 14229 (UDS - Unified Diagnostic Services). They share a common transportation protocol and describe in substance the same classes of diagnostic services. However, correlative to its name, the services for diverse applications, and correspondent to the earlier protocols used by the different manufacturers, were so universalized for UDS that a migration is relatively easy. In substance, the following service categories are described:

- Data reading
- Flash programming
- Failure memory
- ECU routines
- Input/Output control
- Control functions

Additionally, to enable savings in non-competitive areas, a kind of diagnostics operating system was engineered conjointly by OEMs and toolmakers a number of years ago within the framework of ASAM e.V. (Association for Standardization of Automation and Measuring Systems). It is a data-driven system; the data describe the abilities of the system in interaction with the corresponding ECU or vehicle. The necessity of such an action is easy to understand when we call to mind that a door ECU implements completely other functionalities and units than, for instance, an engine ECU.

Within ASAM e.V., an interface (ASAM MCD-3D) was defined to enable symbolic access to ECU and vehicle information and the data description (ASAM MCD-2D ODX – Open Diagnostic data eXchange) defined as exchange format between those involved in the diagnostics process. Both were also taken over as ISO standards. Within the ISO, two further standards were completed: D-PDU API as low-level application programming interface for a simple exchange of diagnostics interfaces and OTX (Open Test sequence eXchange format). The latter facilitates the exchange of diagnostics sequences, for instance between engineering and production.

D-PDU API – the VCI integration layer

The standard describes an API on hexadecimal level. The transportation protocol is treated completely transparently, i.e., for the higher application it is unimportant which protocol was used in D-PDU API. The data in the form of a byte stream are overhanded to the interface together with the addressing information and parameterized protocol details (e.g. timings). The implementation correspondent to the set protocol

follows automatically. ECU replies will then be likewise reported back as byte stream with address information of the answering ECU. More far-reaching capabilities of the interface - such as inputs and outputs - that could be addressed can also be operated with D-PDU API. The description of interface capabilities, which deviates from product to product, is provided as an XML file. A change of the interface is thus very easy. For instance a test program in the engineering department can be run with a USB interface from Manufacturer A, and later in a test stand with a Wi-Fi interface from Manufacturer B. In practice, adaptations - albeit few - are necessary.

ODX – the exchange format for diagnostics data

The standard describes the data contents of diagnostics communications for ECUs built into vehicles in an XML file format. It thus places at disposal the documentation and the parameterization of the test system in a source. The file can thus be utilized equally from the specification phase to the use in production and after-sales services. Through the file, a conversion from hexadecimal to symbolic values takes place. For instance for an engine ECU there is the description of a diagnostics service Reading Revolutions. The corresponding hexadecimal values to send by way of D-PDU API can be determined from the ODX data file. From the answer, the hexadecimal value can be determined, which will be converted to "1900 rpm". Besides this communications information, in ODX the protocol-parameterization, irregularity list of a vehicle in a universalized form, flash programming and variant coding data are also described.

One of the main goals of standardization was the creation of a standard that is machine-readable and has long-term stability (use of XML), that supports the engineering process and is redundant-free as much as possible (derivation concept).

The derivation concept is based on the idea that a large amount of information for a designated ECU can be given that remains constant over the entire life cycle. This information is described in the so-called Base Variant. The individual variants of an ECU, described by the identification string or a software-version, for example, differ only slightly. The main part of the information can thus be inherited from the basis variant for a variant; in the ECU variant. Only the Delta then needs to be described. This can be added, or superfluous information can be omitted. Moreover, information defined differently for ECU variants than for Base Variant can be overwritten, similar to an object-oriented programming language. In addition to these two levels, Base Variant and ECU Variant, there are two further levels: the protocol describes services and parameterizations that are given through the diagnostics protocol and functions thus as a "blueprint" for the ECUs derived from it. The Functional Group level, finally, enables the description of the functional addressing through which several ECUs of a logical group can be addressed over a common address. The best-known example is OBD functionality, which summarizes all emissions-relevant ECUs and queries them together.

With OEM, data for a production series are usually summarized in an ODX data file. The cumulative amount of data can be relatively large because the database includes following information:

- The protocols used in the vehicle
- The information on the ECUs accessible through functional addressing
- All ECUs used in the vehicle, thus the optional (gearbox control device for automatic transmission) as well as the alternative ECUs (engine control devices for 4 and 6 cylinder diesel and for gasoline engines)
- The variants throughout the vehicle's production period

Concrete ODX databases currently reach the 100 megabyte limit. To simplify data exchange, the PDX format (packed ODX) was standardized in addition to ODX. This involves an indexed ZIP data file.

ASAM MCD-3D – the diagnostics operating system

The standard describes an API for access to ECUs with help of ODX data. The relevant runtime system is usually called a D Server or a MCVI Server. In the diagnostics process, it can be used in many different applications - engineering, production, after-sales services - and thus guarantees a consistent usage of the ODX data. This is especially possible because reference implementations for use with various programming languages have been provided for COM/DCOM, JAVA and C++ and are available commercially.

Access occurs on a symbolic level, i.e., an ECU under its name "engine ECU" is selected and subsequently diagnostics services can be carried out for this ECU. An example is the above described service Reading Revolutions that is processed by the runtime engine in the D Server and is sent over D-PDU API to the ECU. The answer is also processed over the ODX data and the result allocated to the API as "1900 rpm".

To improve performance especially in production, in principle as many queries as wished can be processes parallel to one another. The limitations mainly result from the bus load of the (normally) utilized CAN bus and the capability of the gateway ECU to distribute queries sensibly among the different busses.

Currently, ODX data are processed in real D Servers with the help of a runtime format. This has, on the one hand, performance reasons - in a binary format, the data can be packed and optimized for access - and, on the other hand, security reasons. Especially in after-sales services, the data are basically unprotected on the service testers and can be changed and taken by those interested. Binary data are easy to encode and to secure against access so that increased security has clearly been achieved.

OTX – the interchange format for diagnostic sequences

The standard describes an interchange format for diagnostic test sequences. Here, too - similar to ODX - the machine-readability and long-term availability were the main requirements. The idea behind the standard is to create a possibility to give a formal description of basic sequences between test system and ECU as early as during the specification phase. These sequences can be further used and specialized during the process: it is no longer necessary to start from scratch and to copy a basic sequence from a written specification. Through machine-readability, a graphical representation of sequences will also be supported, for instance as flowchart, which offers engineers in engineering, verification, production and after-sales services a universal basis for their discussions.

The standard OTX is subdivided into a number of areas: The core comprises a programming language with the typical

elements: variables, solutions/ instructions and operations (grinding, bifurcation, benchmarking, …). It is not related to diagnostics and can basically be used for a variety of tasks. Standardized libraries are available for special tasks. In addition to diagnostics communication (connection to the D Server), especially string operation, size handling, presentation functions (HMI) and internationalization in order to be able to offer versions of sequences in different languages for the different markets. The standardized libraries use a general add-on mechanism. This can also be used for non-standardized add-ons, whereby any test system in the sequence is integrable. Typical examples are HiL systems, simulations and measuring.

OTX supports the engineering process through various mechanisms. First, it allows a mere specification view for early process phases. Here, the idea of the sequence is described, without furnishing a concrete implementation of the sequence. Thus, it can, for example, already sketch the communication with an ECU without real ODX-files in which real diagnostics services are described. A loadable OTX-sequence is not thereby created. This will be made up for later in the implementation of sequences. To support the variant diversity similar to ODX without having to release a sequence completely every time, special mechanisms were also realized.

In contrast to ODX, for OTX there is no standardized API for a state machine. As interchange format, the tool production can process the format in an interpreter, compile it in a machine format or to import it into an already existing test system and process it there further.

Engineering tester—an example

DTS-Monaco is an example for an engineering tester. It was based on the standards D-PDU API, ODX, MCD-3D and OTX and thus has a command of all relevant standards (figure 3). DTS-Monaco consists of only a few main components and can be expanded for specific uses: the framework, the HMI controls and the runtimes for the standards OTX and ODX. ODX is processed through the D Server DTS-COS. It places at disposal the standardized API complying with ASAM MCD-3D, offers in addition, though, several other extensions that simplify the use in test automations and enable functions in the engineering tester that go beyond the standard. For OTX a runtime interpreter was integrated. The framework provides the basis functionalities that are necessary for diagnostics. Besides the link to the runtime systems, this is especially the tool configuration, which can be saved and reloaded. The role administration is also integrated in the framework. Monaco supports the roles “administrator” and “user”. The administrator can author configurations, save and then provide them to the user. This pertains to the surface configurations as well as to OTX sequences. These can be provided by the administrator in the tool and adapted. The user can only load these pre-configurations in the tool and use them. Changes are only partially possible. This ensures that every user utilizes only those functions on the vehicle for which he/she has been trained.

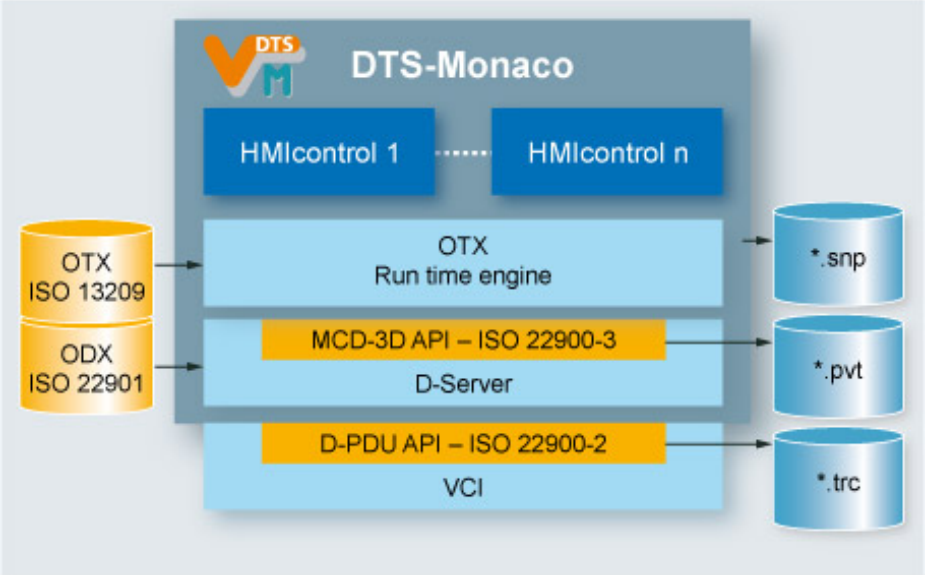

Figure 3: Softing’s DTS-Monaco is an example of an engineering tester. Any protocol can be used, if there are a D-PDU API interface and suitable ODX data.

HMI controls run within the framework as components. HMI controls translate the communications analysis, provided through the D Server and the OTX runtime, into the user view, expected by the user for his/her specific tasks. This can be a fault memory list or a measuring instrument. Every HMI control brings its own configuration. There are both common and specific properties. With the failure memory, this would be whether status information should be shown, for example, and with the measuring instrument if it should be shown as pointer instrument or as bar graph.

Implementation as components is a simple possibility to discuss additional uses with a client, to implement and to install later or in a specific setup. The advantage is that this component can be independently tested and the system integrity is not influenced at all. These add-ons are especially important when a link to a logistics system is required. This is, of course, user-specific, thus can be easily implemented in a product but should not be visible to every user and thus must be installed independently.

Data supply

ODX and OTX runtime environments are the basis for DTS-Monaco. The most important data in the system are thus OTX and ODX data. Access to ODX data takes place - whether through HMI controls or OTX - via names (Reading Revolutions). These ODX names must be administered with the configurations and contiguously with the OTX sequences. This occurs through projects.

The OTX data are processed directly in XML format. This simplifies handling because here data either have been taken from a specification phase and adapted or have been directly authored for the applications of the engineering tester. Here, data conversion would only be cumbersome in operations. For ODX data, this does not follow. These are authored in a small area of applications; in the majority of applications, though, they are subject to version administration and should only be altered under regulated guidelines. Here, therefore, data protection is advisable. In DTS-Monaco, a conversion takes place in a format that is encoded as well as password-protected. It cannot be opened even with a database editor if the password is not known. The data format supports various administrator processes: not only can data be summarized in a data file in an ODX data-

base - typically binary data for a production series - but also each ECU with its variants can be saved in its own data file. Both procedures have advantages and disadvantages. Apart from these data, further data types can be administered in a project, for example simulations datasets, the CAN matrix or filter datasets.

Applications with HMI controls

One of the central HMI controls is the Diagnostic Services Control. It covers a large extent of the applications: data verification and analysis. This occurs through a three-part window: in the first part, the diagnostics services of the database can be "browsed" in a tree, sorted according to ECUs and to functional classes. The selected diagnostics service can be parameterized in the second part. This can be done symbolically by setting the request parameter "variable" to "revolutions" or directly by changing the hexadecimal values in the byte stream. This way, "not OK" values, which are not attainable symbolically, can be set for test purposes. In the third part of the window, results are presented, whereby it is configurable whether the results only are presented or also the structure of the results. For quick access, buttons can additional be configured so that access to the diagnostics services does not need a lengthy search with the browser. Two different methods are available to represent measured values: blocks of measurements can be textually represented in a list, which allows a very compact overview of the values. The repetition rate is adjustable, and access only with manual demand is also possible in order to keep the bus load low. Instruments can also be configured; the variability of these representation possibilities is considerable. Incidentally, such instruments can also be used with knobs or switches to alter values in the ECU. Representing a number of measurement values over time is additionally possible.

For flash programming, an individual HMI control is provided. This covers two applications: it can be used for engineering the function flash programming as well as for simple executions, for instance to update program versions or to swap datasets. In the former case, it is possible to put individual partial functions in the ECU into operation using a five-step process: initialization, compatibility check, validation and finalization. Thereby the single test runs are distinctly faster. In the latter case, swapping datasets, the data package to be programmed is only chosen and then the ECU is programmed at the touch of a button. In both cases, the programming progress is shown with a progress indicator.

Handling failure memory and identification is summarized in an HMI control called Quicktest. The Quicktest is frequently used especially in the prototype workshop. The ECU identifications are read and then the failure memories of all recognized ECUs. Users themselves can configure whether the surrounding conditions are also retrieved. The challenge is that the pilot plant frequently programs unreleased software versions in the ECUs so that variants cannot be recognized. Just as frequently, the valid ODX data are not yet available; in this case, too, no variants can be recognized and the diagnostics must take place using the basic variants. To be able to execute the common application Quick test with high performance, all ECUs possible should be addressed in parallel. This enables performance gains of factor 4 in practice. However, all ECUs cannot be addressed in parallel. First, the alternative ECUs must be recognized since only the ECU for the 4-cylinder diesel motor or for the 6-cylinder gasoline engine, for example, could be present. It is just as important to bear in mind, however, that - corresponding to the bus architecture - ECUs behind gateways often can be addressed only if the gateway itself has been addressed already. ECUs which need to be addressed sequentially have to be configured by a specialist.

Furthermore, numerous other views of diagnostics information are available that will not be discussed here.

Test Documentation

With DTS-Monaco, there are various available possibilities to document the tests performed. The symbolic trace records all send and receive information that is exchanged between tester and ECU(s). Thereby, values interpreted via ODX data are saved on the one hand; on the other, though, also the data sent and received on the hexadecimal level. Thus, a complete test is also subsequently representable and can be documented. The file is delivered as an XML data set so that conversions into other formats or filtering to necessary representations for particular customer processes are unproblematic.

Parallel to this, on the bus-communication level, a trace can be written down. This trace, usually a CAN trace, also allows abnormalities in the protocol to be recognized and be documented.

Conclusion

Standards have not only gained acceptance in the area of diagnostics; with increasing depths of application, there are even broader areas that are being standardized. Diagnostics protocols were the beginning from which producers soon no longer wanted to pay for implementing their special solutions with the increasing number of ECUs. Data formats and APIs to integrate diagnostic solutions in the most different test systems were a logical consequence. Independent of the fact that single-source is basically suitable to reduce costs; quality is also increased through the wide use of relevant standards. That is to say, as soon as diagnostics is easily integrated, nothing is to be said against also introducing its powerful methods in many areas and thus to reach a much larger test width and depth.

Quality, however, is the central argument for customer retention. Seeing how nervous producers react to recalls, it is clear that the message has been understood. Seeing how meticulously customers study emergency road service (breakdown) statistics and enlist them in their purchase decisions, it is clear why. When a car has a defect in early stages despite all efforts, though, a customer must be able to drive away from the garage after a stay as short as possible, satisfied. Diagnostics is the key to this.

The savings is very apparent if the costs for special testing mechanisms in engineering and production would be calculated. Naturally, special sensors and actuators for testing purposes can be built to carry out tests. Naturally, additional tools can be held ready for these tests. Faster and cheaper - and frequently sufficiently good - it is possible with the diagnostics testers available anyway over the installed mechanisms in the vehicle. Consequently, it is done exactly that way.

Markus Steffelbauer is head of product management at Softing Automotive Electronics. He takes responsibility for development and marketing of all hardware and software products and is member of several international committees.