A Rapid Prototyping System, Intelligent Watchdog and Gateway Tool for Automotive Applications

Maid Dzambic*, Christoph Kreuzberger¹, Omar Veledar² and Georg Macher*

*Institute for Technical Informatics, Graz University of Technology, AUSTRIA
Email: {maid.dzambic, georg.macher}@tugraz.at

AVL List GmbH, Graz, AUSTRIA
Email: {maid.dzambic, christoph.kreuzberger, omar.veledar}@avl.com

Abstract—Hand in hand with the inevitable increase in vehicle connectivity solutions, the high security and safety demands for automotive embedded systems are emphasizing the importance of thorough testing of newly developed software components. The software quality is ensured through the usage of various tools at the development stage. This paper introduces an implementation of such a tool, which is based on a standard microcontroller platform and provides functionalities of a Rapid Prototyping System (RPS). The solution is based on the universal calibration protocol XCP and use of an XCP-Master controller. The tool also acts as an intelligent watchdog, as it enables close behavioural monitoring of novel functionalities within an Electronic Control Unit (ECU). Hence, the tool is specially apt for dependable AI testing in the scope of safety-critical applications. This is potentially a key building block of a safety net for testing of novel functionalities on a ‘grey-box-like’ ECU. The XCP-Master controller also provides the possibility to utilize the platform as a gateway for access to remote devices.

I. THE AUTOMOTIVE TOOLING CHALLENGE

The peculiarities of the automotive applications exert unique challenges to the associated embedded systems [1]. The rigorous safety and security, with a frequently added real-time aspect, demand thorough testing measures in a realistic environment. This is matched by a constant rise in the complexity of the vehicle controls, hence enforcing more complex communication in the evolving system of systems [2]. The added complexity is posed by usage of Electronic Control Units (ECU) from a wide range of providers. These devices execute the embedded software, which can contain up to 100 million lines of code [3]. Inevitable time-consuming and inefficient software testing, prior to integration into ECUs, are creating a conflict between the drive to improve existing functionalities and the need for their meticulous testing.

AVL regularly encounters this issue when performing verification, validation or calibration of the developed functions on in-situ ECUs. The challenge is tackled in collaboration with Graz University of Technology, which also brings additional competencies in the field of automotive safety.

A. Rapid Prototyping System

Rapid Prototyping Systems (RPS) offer a possible solution. Such systems are able to directly interface to existing integrated systems to conduct measurements and calibration. The added possibility to bypass ECU functions is exploited when isolating specific aspects or functions within the ECU. Such RPS, which are readily available on the market, enable rapid deployment and testing of new software components, without any hardware-specific considerations [4].

These high-end devices couple a multitude of functionalities with high processing power at a considerable financial cost. Hence, they are often shared between engineers and their use is limited. In contrast, the testing procedures frequently do not demand the full capabilities of these RPSs [5]. The tests often rely on available functionalities and could be performed at a fraction of the available processing power. Hence, we target the usage of a microcontroller-based platform, which provides the possibility to access an ECU and perform measurement and calibration. Aside from the conceptualisation, the presented work goes into the implementation of such a tool.

B. Intelligent Watchdog

Just as all computer systems, embedded systems are also prone to errors. These result from a range of factors, such as random bit flips when writing to the RAM, or environmental influence, such as radiation. Some faults cause permanent system failure, with severe consequences in the safety-critical scenarios. This can be prevented using watchdog devices. These system monitors form general fault detection schemes. They are much simpler systems than the ones they are monitoring and can be connected either as external devices, or implemented directly on the same board as the monitored system. Once it detects a fault, the watchdog’s task is to trigger a system reset and thereby restore the system to its former, fully functional state [6].

Intelligent watchdogs monitor system state by evaluating gathered vital system data, as well as data of interest (e.g. safety-critical functionalities). Any unexpected behaviour can trigger not only a reset, but also supplementary actions from the intelligent watchdog. Those actions are not limited to only setting an alert or triggering a system reset. In this way higher dynamics in error cases can be achieved as the smart watchdog can implement any algorithm to maintain safe functionality. Such intelligent watchdogs are based on more advanced algorithms for system evaluation and complex
decision making. Watchdogs are inevitable in safety-critical applications, such as automated driving (AD), which has zero tolerance for faults [7]. The trustworthiness of autonomous vehicles mandates safety and security. The trust is the key component for acceptance of AD by drivers [8] and other stakeholders. Hence, the vehicles must fully handle safety-critical situations [9], yielding that the safety-critical components must have redundant systems and system monitors for ensuring that the system does not fail under any circumstance. When needed, the built-in redundancy takes over the control of the system and a decision is reached in terms of operating mode [10]. The intelligent watchdog supports this need for monitoring of the safety-critical systems through utilization of the XCP protocol to directly access the memory and to gather the data of interest from the monitored system. These features of the intelligent watchdog are especially crucial for development of systems that adapt during operation time (such as adaptive systems or AI-based systems).

C. Ethernet-CAN Gateway

CAN remains the standard automotive communication protocol for message exchange between ECUs. However, many tools that access or test ECUs, rely on standard PC architecture, which does not contain a CAN module by default. The need for message exchange between Ethernet and CAN-based devices poses a challenge when using remote devices. Implementation of an Ethernet-CAN gateway would enable Internet access to CAN-based devices from remote locations.

II. The Method

The implementation relies on the Universal Calibration Protocol XCP for enabling RPS functionalities. The core of the offered solution is based on an integrated XCP-Master Controller, which manages all necessary processes. The platform connects to a target ECU via XCP on CAN and performs measurements, calibration, and function bypassing. The platform is also configured as an XCP-Slave and thus provides access for other XCP-Master tools such as CANape. This enables the run-time configuration of the RPS-parameters.

As the RPS-implementation runs on a basic microcontroller platform, it shows resource limitation issues early on. Such findings are absent when using powerful commercial RPSs. Despite being computationally inferior, this RPS-implementation offers an added flexibility and, in many cases, serves as a low-cost alternative to commercially available RPSs. The added benefit is the ability to check the behaviour of the tested functions before their real-hardware integration.

The XCP connection is used for monitoring the function behavior within an ECU. It is easily configurable to gather ECU data and evaluate it according to a predefined algorithm within the platform itself. Thus, the platform also acts as an intelligent watchdog, which could aid integration of AI-based or run-time adaptive systems in safety-related context. This implementation is extremely useful during hardware-in-the-loop tests, for capturing the behaviour of certain signals of the complete testing procedure. Furthermore, as it is possible to configure the intelligent watchdog to freely evaluate and act upon the data, it can also be configured to execute the same ECU function and thus act as a redundant system.

The Ethernet-CAN gateway relies on Ethernet and CAN transport layers. It enables message exchange between Ethernet and CAN-based devices. That establishes remote connection to external CAN-based devices by communication over the Internet and through the Ethernet-CAN gateway.

A. The RPS

The RPS (Figure 1) is realised on an Infineon's development platform, which is centred around TC277 microcontroller from Aurix™family. The algorithms use C programming language. Infineon’s MultiCAN library handles the CAN module on the development platform. The open-source lwIP stack controls the Ethernet module. Aside from being geared toward the usage of the XCP protocol and development of an XCP-Master Controller, this toolchain also integrates an XCP-Slave driver, which allows other calibration tools, such as CANape, to perform run-time configuration of the RPS.

1) The Implementation: The XCP-Master Controller enables the platform to access ECU’s memory and manipulate its data. The ensued core capabilities include measurement and calibration. A derivative of these two combined capabilities is ECU function bypass. To perform measurement, calibration or bypass, the RPS must be aware of all addresses that correspond to the variables of interest, their sizes in memory and their values. Thus, two buffers are implemented. One of those buffers stores the measurement-related data, while the other one stores calibration-related data, including calibration-enabling switch variables.

For easier configuration of the RPS parameters, a configuration tool is also provided [11]. Based on the .a2l description file of the target ECU, the tool provides an overview of all available functions and variables of the ECU. The user can choose the variables to be measured/calibrated by name, and the tool automatically creates configuration files that include all necessary information about the variables.
This specific implementation does not use all XCP-Master features. It focuses on resources that enable connection, measurement and calibration. The communication channel to an ECU is the CAN bus, which is still the standard vehicular communication interface and XCP on CAN is commonly present within ECUs. Therefore, a CAN transport layer is developed for the XCP-Master Controller.

A key characteristic of the proposed RPS is its ease to adopt functions-under-test into own structure and hence bypass the function of interest in a target ECU. This is possible because of the ability to measure and calibrate ECU variables. Thereby, the RPS measures the input values of the function of interest from the target ECU and feeds those values to the function to be tested, which is integrated on the RPS-platform. The RPS executes the function and sends output values for calibration to the target ECU, thereby bypassing the ECU function.

An integrated switch is ECUs software mechanism, which enables manipulation by calibration tools. The proposed platform uses this switch to bypass the ECU internal variables. This switch variable determines if the program flow within the ECU should use the internally calculated value or the value from an external device. Therefore, care must be taken when setting/resetting these switch-variables when performing calibration and bypass. This implementation sets the required calibration switch variables during the first calibration cycle.

2) The Test Environment: The test setup uses a production ECU. Selection of the test function is based on ease of demonstration. This function is interchangeable i.e. it is a representative example for the demonstration. In this instance, a function for regulating the duty cycle of a cooling pump is chosen for testing. The source code of the function is integrated into the RPS-Platform and configured for usage as a bypass function of its equivalent counterpart, which is integrated into the ECU. At the start, the bypass function and its ECU counterpart are identical. As no changes were made to the function on the RPS-platform, measurements show a comparison between the RPS-calculated and the ECU-calculated values. The input value of the function is the temperature, which is simulated by the RPS and calibrated into the ECU (Number 1 in Figure 2). The values are linearly increasing from 0°C to 100°C, with cyclic repetitions. This temperature value, which is calibrated into the ECU, is again “measured” by the RPS. This ensures that the input values for the function to be bypassed in the RPS are provided from the ECU (Number 2 in Figure 2). Upon obtaining the input signals, the RPS executes the function to be bypassed and generates the outputs (Number 3 in Figure 2). Finally, the output values, together with the calibration switch-variables (only during the first calibration cycle), are sent for calibration to the target ECU, thus bypassing the ECU-calculated output values (Number 4 and 5 in Figure 2).

A debugger, which is connected to the ECU, logs the calibrated and calculated duty cycle values and the calibrated temperature values. Two methods, DAQ and Polling, are employed for testing both XCP measurement methods. Details about the working principle of DAQ and Polling can be found in [11]. Furthermore, as in some cases RPSs are used to just scale a signal value, this use case is also integrated into the test by scaling the calibrated duty cycle with a factor of 1.1.

B. The Intelligent Watchdog

This implementation enables the desired watchdog approach. It gathers data from a monitored ECU via XCP on CAN and evaluates it based on the users’ needs. Furthermore, it can also impact the behaviour of the ECU via XCP calibration. The conceptualisation is shown in Figure 3.

The watchdog, which is connected to an ECU via XCP on CAN during hardware-in-the-loop tests, gathers data that are to be closely monitored. The watchdog can also be configured to act as a redundant system to an ECU by executing the
same function. By gathering the function output values from the ECU, it can ensure proper execution of the function. If it detects abnormal behaviour, it can overtake execution of that function, hence providing fail-operational performance. This feature can also be used for ensuring safe application limits of adaptive or AI-based systems. Especially in the context of these systems safety strategies can be based on the intelligent watchdog feature and its establishment of a safety frame.

1) The Implementation: To access the ECU to be monitored, the XCP-Master Controller is used in the same way as the RPS implementation. The initial step is to connect to the ECU and to set up the measurement configuration. For watchdog purposes, only DAQ is viable as the measurement method, since it guarantees that all measurements are from the same computation cycle and that they correlate to each other. After receiving the data to be monitored, it proceeds with evaluation based on the implemented algorithm. In this work, the watchdog is configured to execute the same function as the monitored ECU and compare the output values of both executions. The watchdog detects when the values start to considerably deviate from each other for too long.

2) The Test Environment: The RPS test setup is reused to evaluate the intelligent watchdog. The watchdog is configured to execute the same function as the monitored ECU, thus acting as a redundant system to that function. It connects to the ECU via XCP on CAN and gathers the data to be monitored, in this case, the ECU-calculated duty cycle. It also compares the duty cycle that is calculated within ECU with own calculated counterpart. At the time, new ECU measurements are disabled to enable fault simulation. As the deviations are formed between the watchdog-calculated values and the ECU-measured values, it is possible to observe the watchdog’s reactions.

C. The Ethernet CAN interface

The implementation of Ethernet and CAN transport layers for utilization of the XCP-Master and XCP-Slave drivers laid the foundation for the Ethernet-CAN interface. The payload of incoming messages over one layer can be extracted and embedded into the payload of a message for another layer.

1) The Implementation: Incoming Ethernet packages trigger a function, which extracts the message payload. The payload is forwarded to a structure for defining CAN message payloads. To conclude, the function for triggering CAN message transmission is executed and the payload of the Ethernet message is forwarded to the CAN bus, as in Figure 4.

This process is reversible i.e. an XCP master can connect to the platform via XCP on Ethernet and obtain measurement results from a device which is connected via XCP on CAN to the platform.

2) The Test Environment: The interface is evaluated by busload measurements. CANape is used as the XCP-master and the same test ECU as in previous measurements as the XCP-slave. The Ethernet-CAN interface is used for directly transferring the incoming messages from one layer to another. The CAN baud rate is set to 1 Mb/s and the bus is loaded with an increasing number of signals, until the maximum busload is reached. Measurements are performed with DAQ and polling measurement mode, and with 4-byte and 1-byte signals.

III. MEASUREMENTS AND EVALUATION

This section considers measurement results obtained during the evaluation of all three use cases. Besides the visual representation, it also includes the discussion of results.

A. Validating RPS

The results associated with the DAQ measurements during bypass of the function and scaling of the duty cycle are shown in Figure 5. The switch process is observed through enabled calibration during the measurement taking. Before the activation of the switch, the function output value and the ECU calculated value are identical. As the switching takes place, there is an observable jump in the function output signal from the ECU-calculated value, to the scaled RPS-calibrated value. The associated messages which travel over the CAN bus during one calibration cycle are documented in [11]. Figure 5 depicts a linear temperature increase from 0°C to 100°C. The third plot in the Figure shows the ratio between the function output value and the ECU-calculated value of the duty cycle. Just as anticipated, the ratio jumps from 1.0 to 1.1 at the moment when calibration is enabled. Due to the computation
errors (i.e. floating point arithmetic and rounding of errors),
the ratio deviates between 1.098 and 1.108

As this test is performed in DAQ mode, all signal values
are sent to the RPS-platform cyclically by the ECU itself.
The CAN traffic demonstrates 12 signal values being obtained
within a time window of 0.9 ms. Furthermore, the calibration
of the two signals takes 0.8 ms.

Figure 6 shows the obtained signal values during the mea-
surement with Polling. The test setup remains the same as
when implementing DAQ measurements. The depicted signal
behaviour is identical to that with DAQ-based measurements.
Hence, we conclude the correct implementation of the func-
tion bypass. The CAN traffic during the testing process are
documented in [11]. In this case, data gathering of 12 signals
with polling takes 2.7 ms. Calibration of 2 signals takes the
same amount of time as with DAQ, 0.8 ms.

As expected, the main limitation of the RPS implementation
is based on a limited communication speed between the RPS
and the target ECU. A proper implementation must consider
the number of signals that should be measured, the time taken
to execute the bypassing function and the number of signals
that should be calibrated.

In line with these observations, DAQ measurement method
provides improved performance over the polling functionality
and is the recommended measurement method.

B. Function execution monitoring

The measurement results obtained during evaluation of the
watchdog are shown on Figure 7. The watchdog is executing
the same function as the ECU, thereby gathering the ECU-
calculated values and checking their correlation to its own
calculated values. The upper plot of Figure 7 shows the two
duty cycle values. When the new ECU-measurements are
disabled, the watchdog error counter begins to increase, as
shown in the plot below. When this counter reaches a critical
value (set to be 10 in this case), the watchdog fault detection
signal is triggered. Furthermore, to observe the watchdog
recovery stage, the fault is disabled at a point of time. Thereby,
it can be observed that the watchdog error counter begins to
decrease until it reaches the normal values again and the fault
detection signal gets deactivated.

C. Gateway performance measurements

Figure 8 shows the busload measurements obtained during
evaluation of the Ethernet-CAN interface with the DAQ and
Polling measurement methods. In the case of DAQ, it can
be observed that usage of 4-byte signals yield the maximum
busload at 140 signals, while it takes 560 1-byte signals to
saturate the bus. This is due to the fact that CANape configures
four 1-byte signals to be transferred with one XCP message,
while with 4-byte messages it is capable of transferring only
one signal per message.
In case of polling, reduced performance is achieved in comparison to DAQ measurement method. This is caused by the need to send a poll request from master to slave for every signal. Thereby, the average time delay between two successive poll requests is 0.5 ms, making it impossible to utilize the full potential of CAN bus bandwidth. Regardless of the signal size, the maximum number of signals which could be reliably transferred is 20.

The rapid prototyping system functionality enables calibration and bypass of functions within an ECU, but thereby the limiting factor is the CAN communication speed. The whole bypass process must be shorter than the computation cycle of the tested function. The intelligent watchdog functionality provides the possibility to closely monitor specific signals and implement specific fault detection and decision making algorithms. Its potential usage includes being an additional safety net during hardware-in-the-loop tests, enabling close monitoring of parameters of the ECU under test. This ability to closely monitor the behaviour of new ECU functions turns the watchdog into a powerful tool for dependable AI testing in the scope of safety-critical applications. With the Ethernet-CAN gateway, XCP messages are easily communicated between the CAN and Ethernet transport layers, hence achieving an easy access to devices on a CAN bus for remote monitoring and control over the Internet. As all three use cases are implemented on a standard microcontroller platform, a minor investment is required to return high value to automotive developers.

**ACKNOWLEDGMENTS**

This work was partially supported by PRYSTINE project, funded by the Austrian Research Promotion Agency (FFG) and Electronic Component Systems for European Leadership Joint Undertaking (ECSEL-JU) under GA n.783190 and by TEACHING project, funded by the EU Horizon 2020 framework programme under GA n.871385.

**REFERENCES**

[1] Kaiser M., Schaefer U., Haaf G. (2015) Electronic control unit. In: Reif K. (eds) Automotive Mechatronics. Bosch Professional Automotive Information. Springer Vieweg, Wiesbaden. https://doi.org/10.1007/978-3-658-03975-2_2

[2] Krah D., Mischo S. (2015) Architecture. In: Reif K. (eds) Automotive Mechatronics. Bosch Professional Automotive Information. Springer Vieweg, Wiesbaden. https://doi.org/10.1007/978-3-658-03975-2_2

[3] Dajsuren Y., den Brand M. (2019) Automotive Software Engineering: Past, Present, and Future. In: Dajsuren Y., van den Brand M. (eds) Automotive Systems and Software Engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-12157-0_1

[4] Reus H., Liao J., Gitt C. et al.: Efficient Automatic Application in Rapid Prototyping Software Development. ATZ Elektron Worldw 14, 54–59 (2019). https://doi.org/10.1007/s38314-019-00059-8

[5] Macher G., Bachinger M., Stolz M. (2017) Embedded Multi-Core System for Design of Next Generation Powertrain Control Units. In: Proceedings of 2017 13th European Dependable Computing Conference. IEEE Computer Society Conference Publishing Services (CPS). ISBN: 978-1-5386-0602-5/17. https://doi.org/10.1109/EDCC.2017.32.

[6] Beningo J.: A Review of Watchdog Architectures and their Application. Ann Arbor, University of Michigan (2010). [7] Veledar O., Damjanovic-Behrendt V., Macher G. (2019) Digital Twins for Dependability Improvement of Autonomous Driving. In: Walker A., O’Connor R., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2019. Communications in Computer and Information Science, vol 1060. Springer, Cham. https://doi.org/10.1007/978-3-030-28005-5_32

[8] Dimitrakopoulos, G., Uden, L., Varlamis, I,: The future of intelligent transport systems. Elsevier (2020)

[9] Macher G., Druml N., Veledar O., Reckenzaun J.: Safety and Security Aspects of Fuel-Operational Urban Surround perceptION (FUSION), 286-300 (2019). https://doi.org/10.1007/978-3-030-32872-6_19

[10] Druml N. et al., “Programmable Systems for Intelligence in Automobiles (PRYSTINE): Technical Progress after Year 2,” 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia, 2020, pp. 360-369, doi: 10.1109/DSD51259.2020.00065.

[11] Djarmbic M.: Development and Integration of a Rapid Prototyping System Utilising XCP Protocol. Master’s thesis. TU Graz (2020)