EHOOKS makes Prototyping Rapid

Shruthi Ananthachar
ETAS Automotive India Private Limited

ABSTRACT

Nearly 90% of all new functionality of a vehicle is realized by electronics which is more flexible and cost effective than mechanics. Today's midsize cars in the mass market features about 10-15 ECUs inter-connected on the ECU onboard data bus. Of these onboard ECUs, the Powertrain Engine control units contain a substantial segment of features and functions with up to 20,000 function parameters required for adjusting the engine management function to a variant and vehicle with an increase Year-on-year as more number of variants are added to the product line. The volume of code is proportional to this large magnitude of parameter tuning. As a result the overall share of the software in the vehicle price is increasing.

One effective way to control the cost of the software is through rapid control prototyping. But more often than not, there are no easy approaches suitable for integration of the new systems being developed. The traditional approach has been to wait for the supplier to include this with the ECU development, but this means repeated consultation, repeated iterations and a long wait time.

The bypass based rapid control prototyping technique is the effective was to quickly model the new ideas and integrate them with the existing Control unit software for easy verification, validation and even calibration. This limits the development disruption and brings down the development cost significantly. ETAS' EHOOKS allows a controls engineer/function developer to quickly add the necessary hooks in the base ECU software without any modifications to the source code

INTRODUCTION

Powertrain controls development is complex and continuously evolving with new features. It is also a complex development procedure involving repeated consultation between the OEM and the Tier1 who is responsible for software and sometimes even function development.

A function developer (usually represented in OEM) identifies and specifies changes from different parts of the software to be implemented as an improvement over the existing function. The software pertaining to these functions are then developed by a Software developer (usually represented in Tier1) and included in the next software release step. The wait for this function to be available for testing/calibrating is anywhere between several weeks to a few months.

This incremental software development approach and the turnaround time required for each increment, limits the frequency at which new ideas/innovations can be introduced or tested. Also due to inherent approach of only requirement specification rather than function implementation needs the idea to be realized very clearly and sharing of IP with a Tier1. Also, it is not cost-effective, since the OEM has to pay for the software development, irrespective of whether they productionize the function or not.

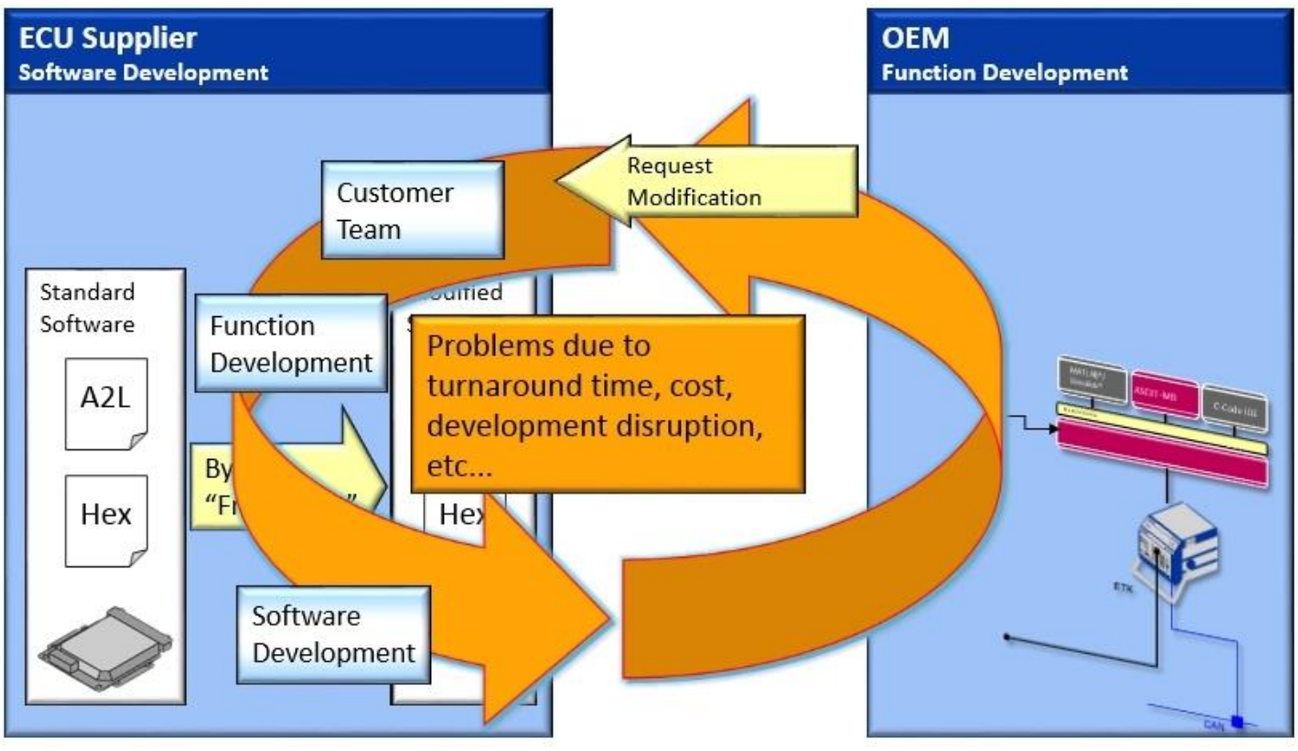


Figure 1: Development process problem statement

Rapid prototyping enables a function developer to implement the ideas independent of a Tier1 and close the feedback loop very quickly. This is also an early enabler for validation and stable calibration early in the life-cycle. As a result Rapid control prototyping has become an integral part of development cycle at most major OEMs in the leading markets.

SOFTWARE LIFECYCLE AND RAPID PROTOTYPING

A typical lifecycle model followed in the Automotive Industry is the V-Model. This model is used for phased development, deployment and maintenance of the software. Although the lifecycle phases(phase names) differ from organization to organization. For this document, we will classify the phases into four categories broadly –

- Function Development – covering Architecture definition, Simulation, Prototyping and finally leading to requirement specification
- Software Development – Software Modeling and Code creation, Integration with the Base Software and finally leading to a software release
- Test and Validation – MIL, SIL, HIL Validation and leading to quality assurance of a software release
- Calibration – Software parameter calibration & Measurement, leading to project specific customization and production readiness

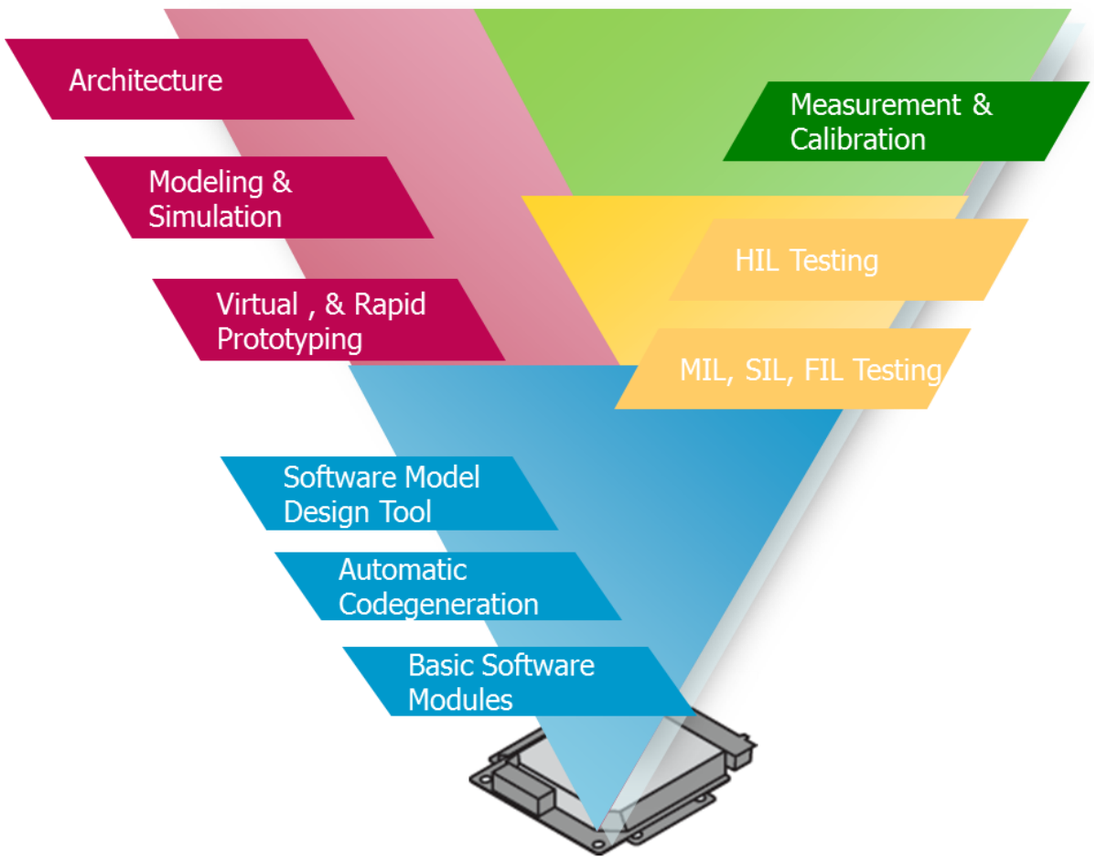


Figure 2: Development Process model

In a conventional approach from traditional automotive software development, the OEM would assume the role of system integrator and software inclusive of the functions would be developed only by the Tier1. But with the introduction of prototyping process and capability (both at OEM and Tier1), now the OEM can also act as an Add-On Function or even a software contributor.

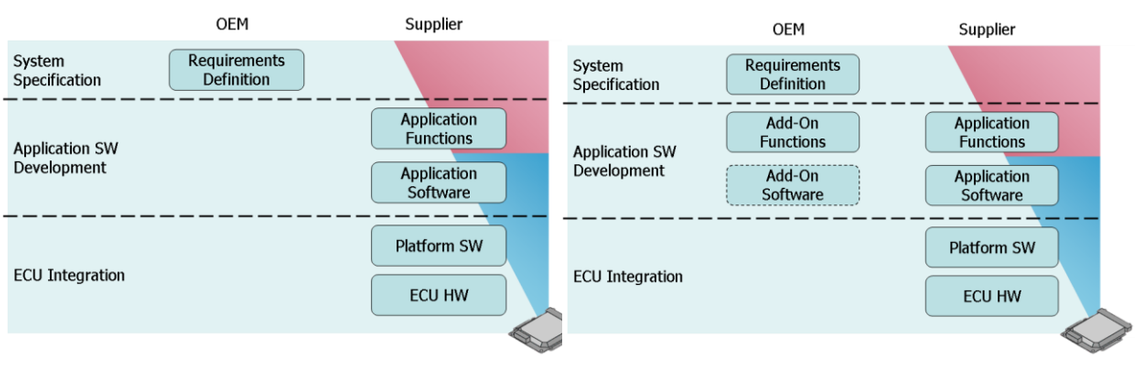


Figure 3: Co-operation model between OEM and Supplier

While, establishing the process of Rapid Prototyping is beneficial to both parties, the challenges associated with the creation, deployment and maintenance of infrastructure is also high. Some of the typical requirements for rapid prototyping include –

1. Access to software development/integration environment for integrating the new functionality

2. Access to ECU internal data read and write

3. Access to complete control software for integration testing

4. IP protection for the created functionality

An efficient solution to address all the above challenges is to implement a bypass based Rapid Prototyping solution. The bypass mechanism addresses all the requirements as stated above and help with building the right infrastructure.

INTRODUCTION TO BYPASS

Simply put, a bypass mechanism is to overwrite the existing ECU functionality with a new functionality. The new functionality then has to be integrated with the existing ECU functionality and be able to run on the actual target. An example of a service based bypass is shown below. New Functions (F4` and F7`) add-on and substitute the existing functions (F4 and F7) and the functions are integrated into the system. The inputs and outputs remain the same, while just the functions are replaced during execution.

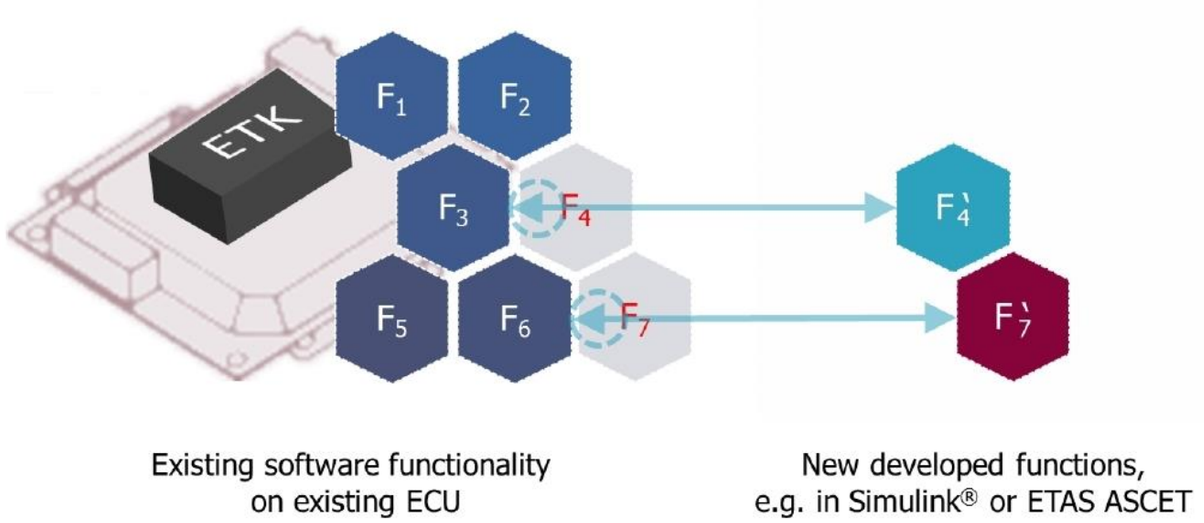


Figure 4: Bypass based system

So, the basic elements for a bypass infrastructure should include –
1. ECU infrastructure for hooking the variables/functions that needs to be bypassed

2. Interface tool to introduce and integrate the new functionality to the existing software

3. Provision to extend the computation to an external processor based on the need

4. Provision to extend the I/O capability depending on the need of the application

5. Trigger and synchronization mechanism to control the external processor in tandem with ECU processing

ETAS' SOLUTION

ETAS' EHOOKS is an easy to use software tool that enables rapid insertion of software hooks directly into the ECU software without access to the development environment. This enables rapid feed-back loop for function prototyping and reduced cost of development, validation and calibration.

With EHOOKS, a function developer places a hook directly in the ECU Hex and A2L file. With advanced hook insertion technology, EHOOKS provides a reliable and accurate way to modify the ECU software. EHOOKS is developed in close cooperation with ECU suppliers and hence the hook quality is reliable even for the complex addressed variables in the software. EHOOKS ability to introduce and integrate software changes immediately can reduce the downtime, cut the number of ECU software deliveries required and shorten the feedback loop to within the OEM development team without having to go into a repeated consultation mode with the ECU supplier.

EHOOKS TECHNOLOGY

EHOOKS is three-part tool used by different target groups at different stages of functionality. The first part is called the EHOOKS-PREP; this is essentially used by the Tier1 Supplier or the ECU Software provider and is used to prepare the EHOOKS ready software.

The second part of the tool is called the EHOOKS-DEV and is used by the Function developer at the OEM to integrate the new hooks/functions into the ECU software.

The third part of the tool is called the EHOOKS-CAL/EHOOKS-BYP and is essential for unlocking the HOOKS that are installed using the EHOOKS-DEV.

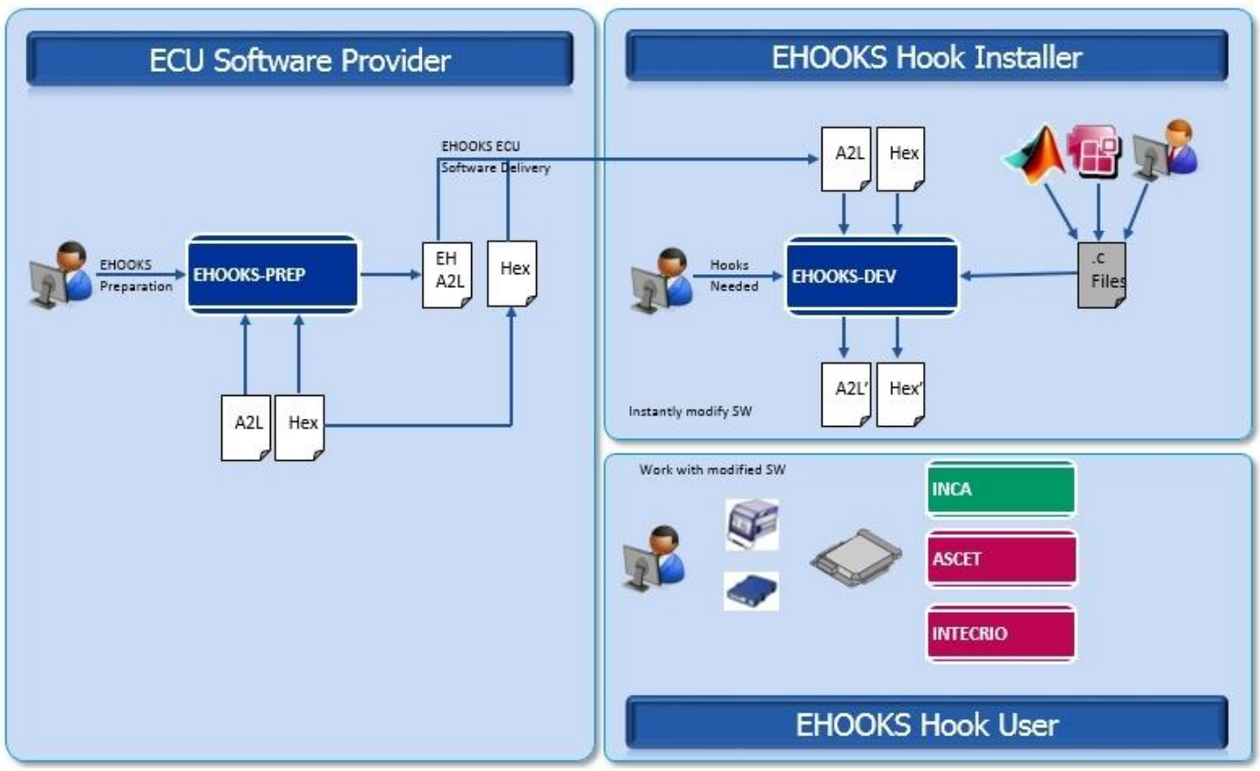


Figure 5: EHOOKS Overview

With the increased complexity of the control software, the effort to introduce hooks for each required ECU software variable will be huge. EHOOKS solves this problem with the component called EHOOKS-PREP. The PREP is part of the EHOOKS tool infrastructure at Tier1 or the ECU supplier. As a result, EHOOKS-PREP is a specific installation for each ECU family. Any new ECU family requires a 'port' creation for EHOOKS-PREP. The PREP allows the supplier to perform a one-time preparation and configuration of the ECU software.

All the information necessary for the hook and bypass preparation are captured during the PREP stage and this information is included in an encrypted block inside the A2L file. The block includes the following information –

- The list of variables and functions that may be hooked by a function developer
- Information about message copies for variables that can be hooked
- ECU software architecture information
- Memory section information – RAM and ROM for internal bypass function
- RTOS related information for Processes and rasters

The configuration information are generally captured in an XML format and the output of the analysis of this data is stored in an IF_DATA block of the ECUINTERNALS in the A2L file. In addition to the information in the A2L file, the PREP phase disables the ECU checksum and includes support functions to the base software.

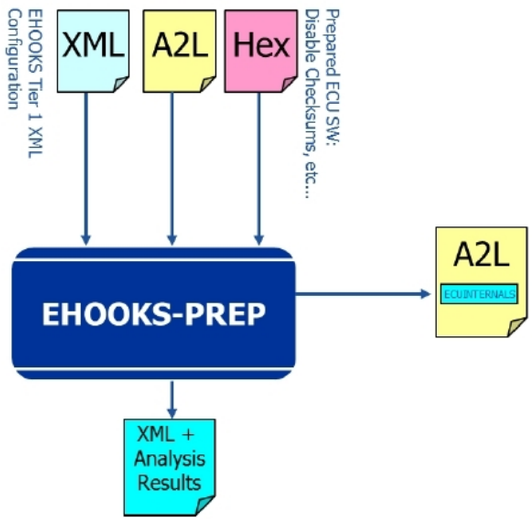


Figure 6: EHOOKS-PREP phase

The EHOOKS-DEV exploits the information placed in the A2L and locates the variables that are selected for hooking, it analyses the HEX image and locates the variables for hooking and the functions for bypass. It patches the ECU software image with the hook code and hook control characteristics. EHOOKS-DEV then compiles and builds the new ECU software and updates the A2L file with information on new hooks and hook control characteristics.

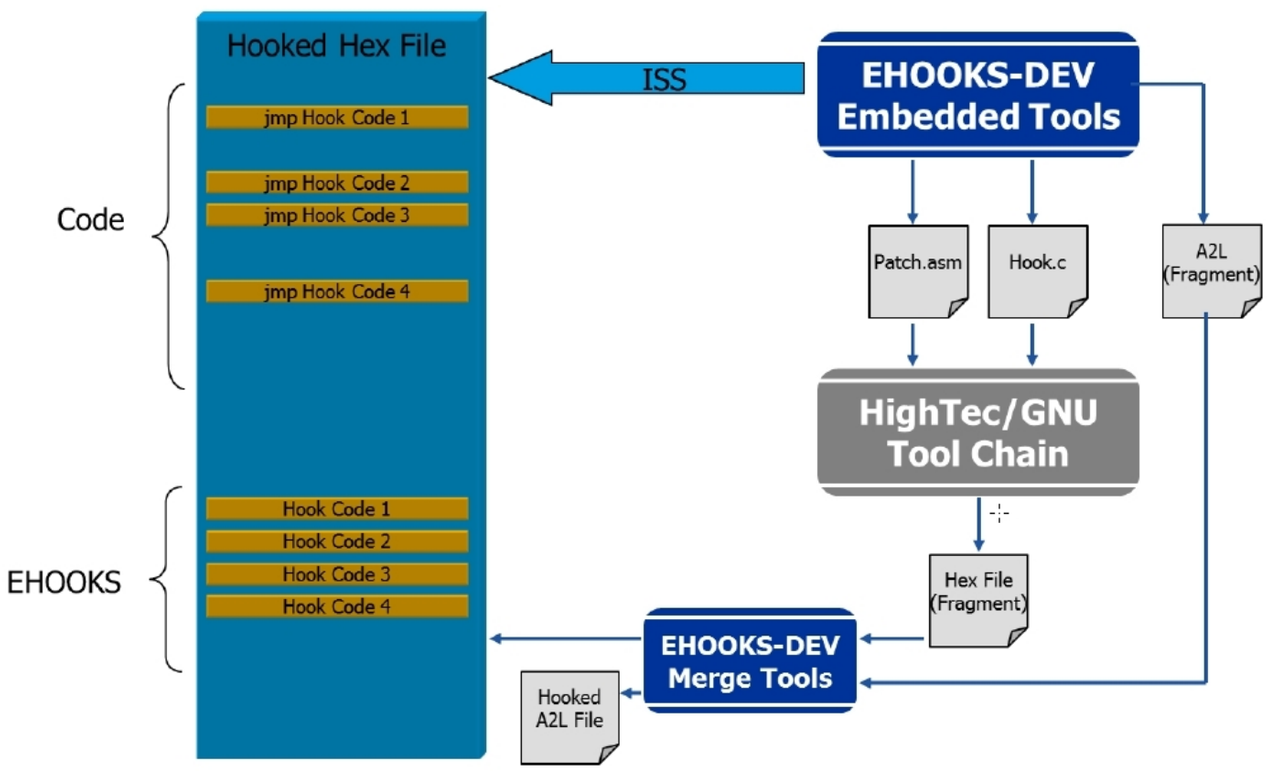


Figure 7: EHOOKS-DEV hook insertion phase

TYPES OF HOOKS

EHOOKS allows different types of hooks to be introduced into the original ECU software image, thus allowing many different types of use-cases for developers across the V-Cycle.

CONSTANT VALUE HOOKS

An ECU variable is hooked with a constant value. The value cannot be changed during ECU run-time but allows run-time control of whether the original ECU value is used or the bypass constant value. This is a good use-case for value stabilization to cancel noise during initial calibration for example.

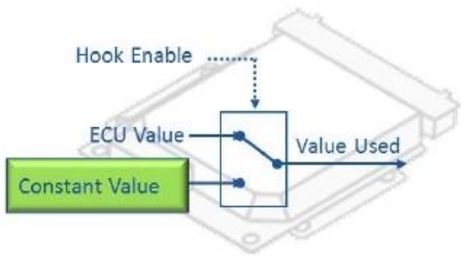


Figure 8: EHOOKS Hook types

CALIBRATION HOOKS

An ECU variable is bypassed with a characteristic value hook. This type of hook while allowing runtime control of whether the original ECU value is used or a characteristic bypass, also allows the change of the bypass value during runtime. In this case, INCA can be used to modify the value of the bypass variable during runtime. This kind of hooks are very useful to simulate software with known test values for easy fault simulation for example.

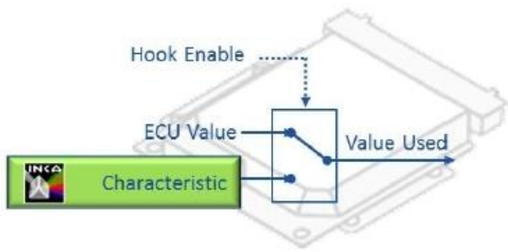


Figure 9: EHOOKS Hook types

INTERNAL BYPASS HOOKS

Internal bypass hooks overwrite an ECU variable with a new computation algorithm and the new code is running directly on the ECU. The spare ECU resources (RAM and ROM) are used to run the new algorithms. Again a runtime control of the existing calculated ECU value or the new value to be used is possible. This is the first step towards rapid control prototyping without having to worry about to actual software implementation.

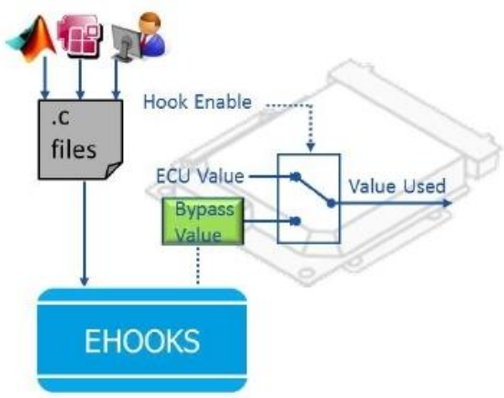


Figure 10: EHOOKS Hook types

EXTERNAL BYPASS HOOKS

Internal bypass hooks overwrite an ECU variable with a new computation algorithm and the new code is running on an external rapid prototyping system. Again a runtime control of the existing calculated ECU value or the new value to be used is possible. ETAS ASCET-RP or INTECRIO is used to configure the rapid prototyping software system that runs on dedicated rapid prototyping hardware like ES910.

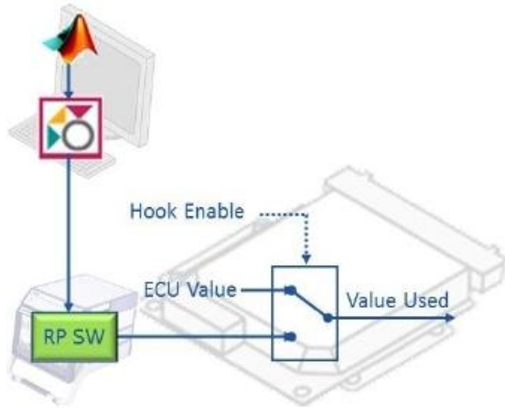
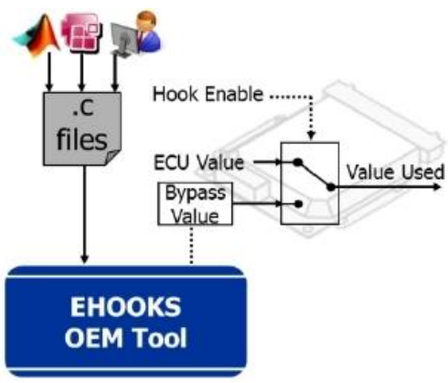


Figure 11: EHOOKS Hook types

Types of Bypass

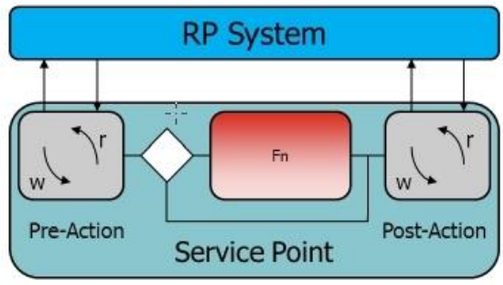
Hook Based Bypass (HBB)

EHOOKS Hook based bypass creates framework to enable a software bypass hook for ECU variables. The advanced hook insertion technology provides accurate and reliable results.



Service Based Bypass (SBB)

Existing ECU functions can be bypassed by service points using EHOOKS. SBB provides an option for run-time control of whether the ECU process or the service point is executed.



EHOOKS Usecases

- Internal Bypass Rapid Prototyping – Add the necessary Hooks to the base ECU software and merge your new algorithm (from ASCET, Simulink or Handcode) for quick evaluation. EHOOKS lets you take advantage of available resources (e.g. ROM, RAM) on your ECU to accelerate prototyping activities
- External Bypass Rapid Prototyping – With resource constraints in the ECU, prototyping can become a bottle neck for efficient ECU software development. EHOOKS enables you to insert necessary external bypass hooks and run your ASCET, Simulink or handcode models on the ETAS ES910 or ES1000 rapid prototyping hardware
- Workaround a Software Bug – Sometimes minor software bugs can prevent calibration work from progressing causing major delay and lost engineering/calibration time. EHOOKS provides a perfect solution for this by enabling you to bypass the ECU variable which potentially blocks your calibration with a constant or a calibration hook
- Improve calibration efficiency – An unstable software value can make a calibration job complicated and time consuming. Use EHOOKS to bypass the unstable value with a constant or calibration hook
- Testing Diagnostic Software – Testing the diagnostic software can sometimes be challenging to cause all the necessary diagnostic state transitions. Use EHOOKS to bypass with calibration hook to directly stimulate/trigger the required state transitions

Advantages of EHOOKS

- Reliable and safe bypass due to knowledge of a Port for each target, saturation of bypass values and prevention of out-of-range bypass values
- EHOOKS allows a ‘Forced write’ option which enables bypass value to be written into the ECU variable when ECU doesn’t normally write to the variable itself
- EHOOKS enables both On Target and External Bypass
- Efficient Prototyping with production ready hardware
- Only necessary hooks could be implemented for a particular experiment without much changes to the hardware

- Easy replacement of noisy signals with stable inputs and hence increased calibration efficiency
- Direct access to Diagnostics states

Summary of Features and Benefits

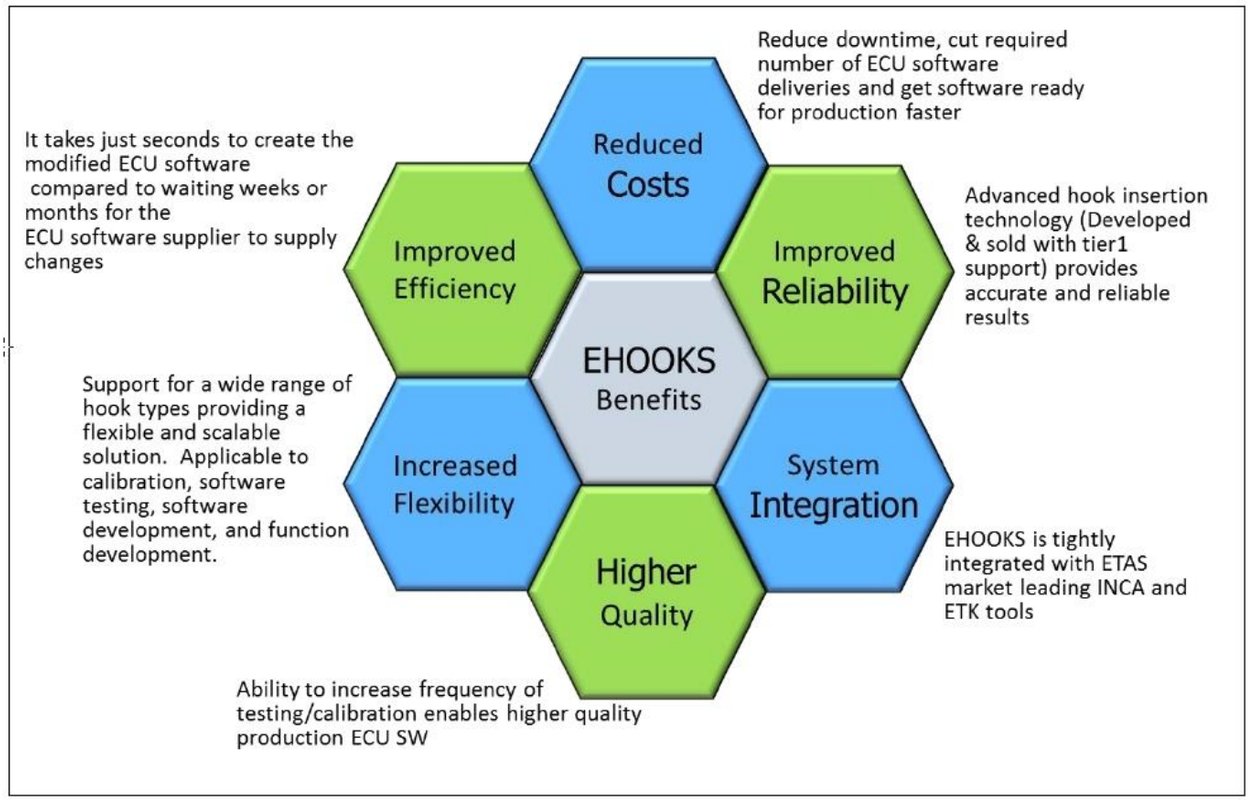


Figure 12: EHOOKS benefits

CONCLUSION

ETAS' EHOOKS is a versatile tool with many options for easy introduction of rapid prototyping into the automotive embedded software development process. The bypass technology allows easy reuse of existing software platforms and allows rapid prototyping to be performed on production ready hardware and systems for better confidence on the prototyping quality.

EHOOKS allows a Tier1 to securely create a safe prototyping infrastructure for an OEM with minimal effort. It allows both the OEM and Tier1 to protect their Intellectual property during the critical phase of ECU function development. The technology allows for easy extension of computation and IO capabilities of an original ECU which is essential during a prototype phase allowing a function developer to concentrate on algorithm development without bothering about software implementation and optimization for critical computer resources.

REFERENCES

1. EHOOKS – Prototyping is Rapid Again - By V. Jaikamal and N. Tracey
2. Function and Software Development for ECUs - 0811_ATZ_elektronik_Funktions_und_Softwareentwicklung_EN
3. *Hybrid Drives with Autosar-Compliant Control Units"* By N. Lestrée, A. Pouthier, PSA; G. Nice and Dr. N. Tracey, ETAS
4. Software Prototyping and Requirements Engineering – By Joseph E. Urban, Arizona State University

PREVIOUS PUBLICATION

Conference on Powertrain Technology Nov 2015

CONTACT

Shruthi Ananthachar
Field Application Expert
ETAS Automotive India Private Limited
Shruthi.Ananthachar@etas.com

DEFINITIONS, ACRONYMS, ABBREVIATIONS

OEM – Original Equipment Manufacturer
INCA – Integrated Calibration and Acquisition
ECU – Electronic Control Unit
ROM – Read Only Memory
RAM – Random Access Memory
MIL – Model In Loop
SIL – Software In Loop
HIL – Hardware In Loop