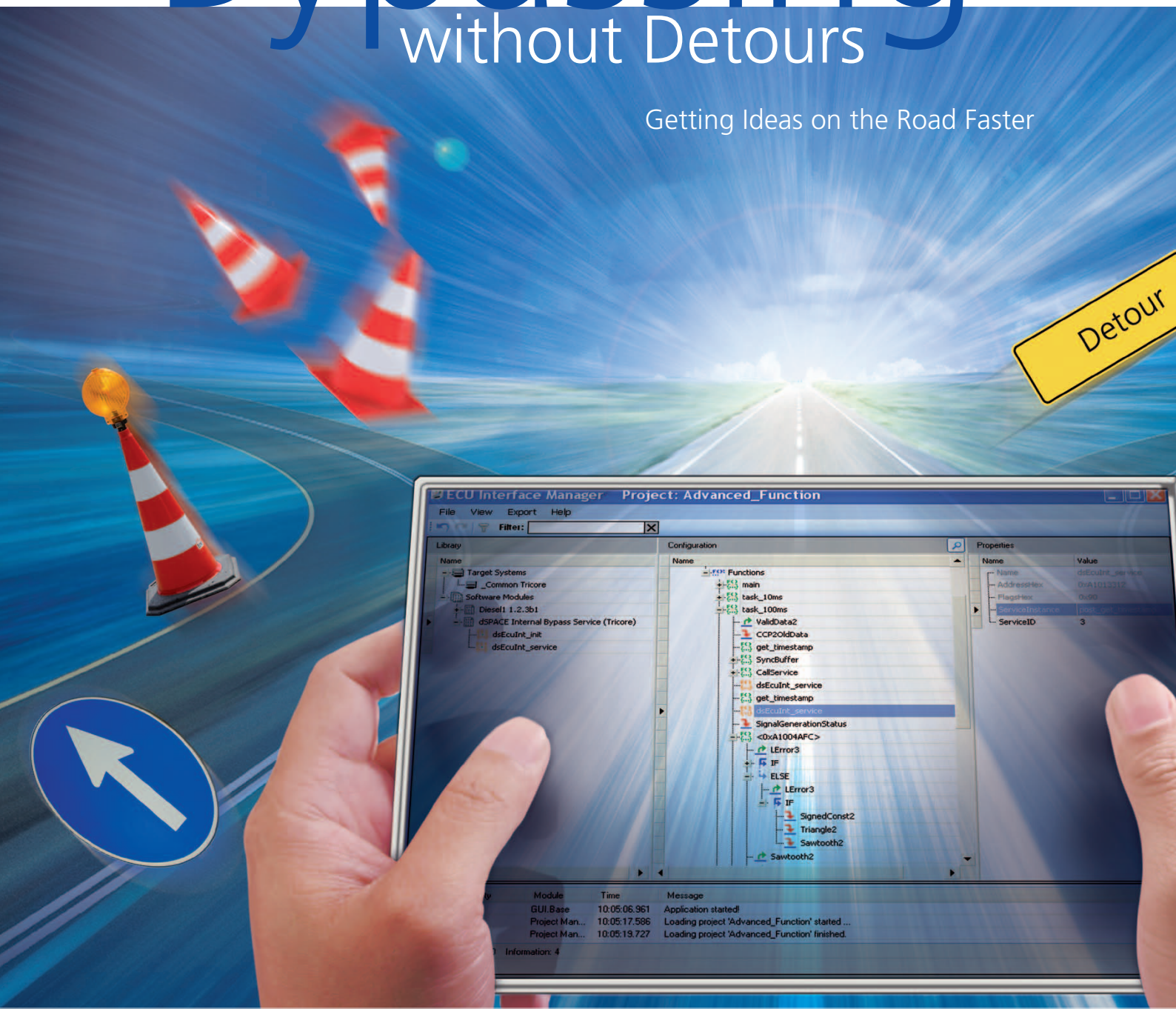
Bypassing
without Detours

Getting Ideas on the Road Faster



The ECU Interface Manager and the internal bypass option extend dSPACE's portfolio of tools for fast function development and service-based bypassing. Function calls can now be inserted directly into the compiled ECU software, and new ideas can be implemented directly on the target ECU. No ECU source code or build environment is necessary.

New Challenges

Nowadays, it is unusual to develop completely new ECU software for new vehicle generations: existing code is adapted instead. Service-based external bypassing, in which a rapid prototyping system running in parallel to the ECU computes new functions, is a proven method for this approach. The supplier has to integrate the necessary interfaces (called bypass hooks) in the ECU software as service calls in the source code. This process usually involves an iterative consultation process between the OEM and the supplier, and in some cases additional, previously unplanned software adaptations become necessary during a project. For the vehicle manufacturer, this can mean long project run times and high project costs.
AUTOSAR ECUs present additional challenges, because if software components are developed by the vehicle manufacturer or by project partners and then made available to the ECU supplier for integration into the overall software as object code, the ECU supplier is usually not able to change the components by inserting bypass hooks.
Even though current driving forces behind innovation, such as electromobility, CO_2 reduction and road safety, are pushing up the need for suitable prototyping development environments, budget restrictions are often an obstacle. Affordable starter systems are increasingly in demand. At the same time, developers soon come up against the limits of using the external bypass method to create fast control loops.

The Answer: Extended Tool Chain

dSPACE offers the answer to these challenges: an extended tool chain for function development with the bypass method. Besides their current solutions for external bypassing on high-performance rapid prototyping systems, dSPACE now also supports internal bypassing, also known as on-target prototyping. With an integrated development environment in MATLAB®/Simulink®, software functions can be developed directly on the existing ECU, using its free RAM and flash memory resources.
A new tool is also available: the ECU Interface Manager. This enables end users to integrate bypass hooks for internal and external bypassing into the compiled ECU software. Thus, special software versions from the ECU supplier are no longer needed.

Benefits of Service-Based Bypassing

dSPACE offers various services for internal and external bypassing, providing the following benefits:

- *Bypass hook configuration in the modeling environment*:
  If an ECU software version with service calls (bypass hooks) is available, the modeling environment lets developers define which bypass functions to start with

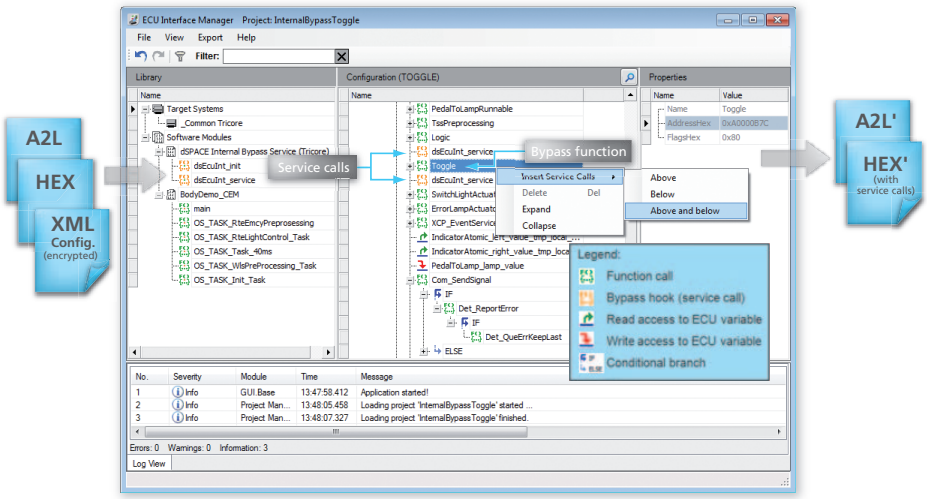


Figure 1: Fast integration of service calls via the ECU Interface Manager.

which bypass hook, and which ECU variables to read and write.

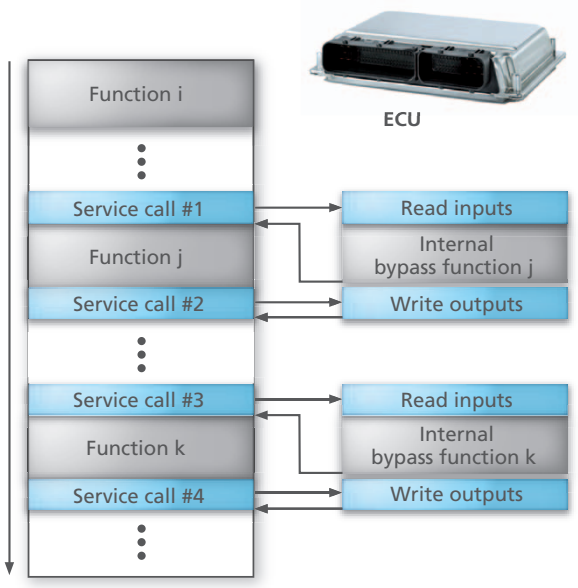
- *Data consistency:*
  When several ECU variables are read and written, double buffer mechanisms can ensure data consistency. This option is especially important if the execution of the bypass function and the reading of input values and writing of output values take place in different ECU tasks.
- *Safety:*
  There are various safety mechanisms, including error counters with automatic bypass hook switch-off and plausibility checks before writing variables.

Integrating Service Calls into the Compiled ECU Software

In internal and external bypassing, the ECU Interface Manager can integrate service calls into the compiled ECU software (hex file) without any changes to the ECU source code. Very little information is needed in addition to the hex file and the associated ECU variable description (A2L file): just information such as the details on free RAM and flash memory areas and entry points for the hex code parser, which the ECU supplier usually provides in an encrypted XML configuration file.

The ECU Interface Manager analyzes the hex code to find function calls, accesses to ECU variables and conditional branches. The configuration file defines which information the end user will be able to see in the ECU Interface Manager. Filter and search mechanisms locate dedicated variable accesses and function calls in the program sequence. After specifying the service call positions, users can intuitively insert the service calls into the ECU code and generate modified A2L and hex files at the push of a button. Access to the ECU supplier's build environment is not necessary. With this method, it takes end users only a few minutes to independently integrate the bypass hooks into the ECU software. Thus, changes to the original software status can be limited to a specific bypassing task, thereby minimizing memory

Figure 2: Example implementation of internal, service-based bypassing using two bypassed ECU functions.



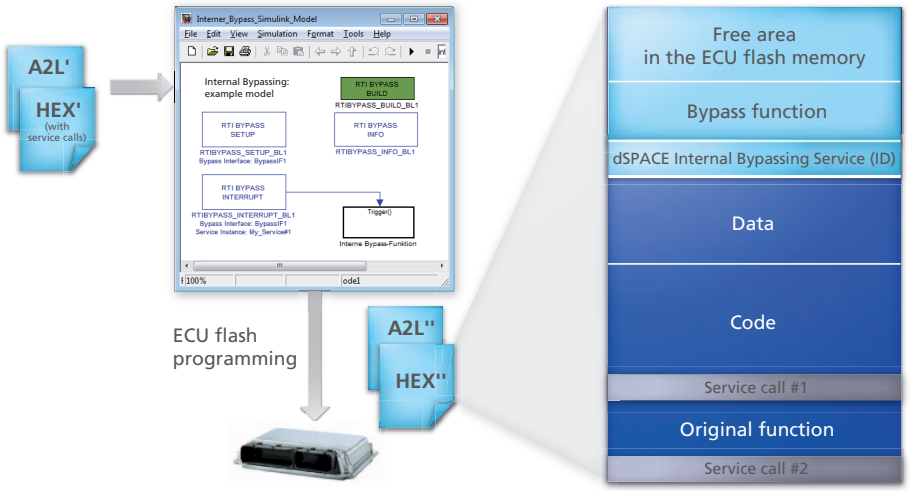


Figure 3: Developing a software function directly on the ECU with the new internal bypass option for the RTI Bypass Blockset.

requirements and effects on the ECU's run-time behavior. For example, in typical applications, bypass hooks for internal bypassing with Infineon TriCore™ microcontrollers require 32 bytes in ECU RAM and 2 kBytes in the ECU's flash memory.

On-Target Prototyping

The new internal bypass option for the RTI Bypass Blockset supports model-based function development directly on an ECU. It even allows users to switch between external and internal bypassing without changing the function model. For example, if the ECU's RAM and flash memories are too small during internal bypassing or additional sensor signals are needed, developers can quickly switch to an external prototyping system without modifying the models. The same applies in the opposite direction, for example, when a function that was developed in external bypassing has to be validated directly on ECUs in a fleet test. Moreover, external and internal bypass parts can be combined in one function model in any possible way. Because ECUs with production software versions usually have very little free RAM and flash memory, one possibility would be to compute function parts in very fast rasters on the ECU and execute calculations in slower control loops on an external prototyping system.

After reading in the A2L and hex files via the relevant RTI bypass blocks, in the modeling environment the user can select the service calls that were previously integrated by means of the ECU Interface Manager and use them to read and write any ECU variables or to call the bypass function. The code for the function model is then generated, linked to the free area of the ECU's flash memory, and merged with the original ECU software – all with push-button ease. The result is an A2L file with added internal bypass function variables, and new hex code that can be flashed to the ECU with proven tools. During the modeling process, an info block indicates how much free RAM and flash memory is available. When the Simulink® Coder™ is used to generate 32-bit floating-point code for a function model with around 400 blocks and 30 input and output variables, for example, about 30 kByte flash memory and less than 4 kByte RAM memory are required on Infineon TriCore™ microcontrollers.

The ECU Interface Manager and the internal bypass option currently support Infineon TriCore™ microcontrollers. There are no dependencies on ECU platforms of specific suppliers. Other microcontroller families, such as MPC5xxx from Freescale, will follow in the second half of 2012. ■

The Benefits at a Glance:

- End users can quickly integrate function bypass hooks into the compiled ECU software
- Support of on-target prototyping and external bypassing
- No need for access to ECU's source code and build environment
- Easier bypass hook integration thanks to graphical representation of function calls, conditional branches and variable accesses in the ECU software
- Switching between internal and external bypassing without modifying the model