Bypassing with NBD Bus at DENSO

CUSTOMERS

- **Access to internal RAM without altering the ECU code**
- **Graphical programming in MATLAB/ Simulink**
- **Software support with RTI blockset**

Manji Suzuki is Senior Manager in the Corporate Research and Development Center at DENSO CORP.

The growing need for high-performance electronic control units (ECU) increases the demand for powerful 32-bit microcontrollers containing RISC architectures. This refers especially to ECUs for engine management where control functions are becoming more and more complex. Since an integral element of modern 32-bit microcontrollers is built-in debug hardware, such as the Non Break Debug bus (NBD bus), DENSO successfully incorporates these interfaces into their dSPACE development environment.

Historical Background

In 1994, Yokogawa Digital Computer, YDC, launched a built-in debug interface for microcontrollers. This technology has evolved over the years and is likely to become standard in the future. There are already a number of manufacturers that equip their controllers for automotive applications with NBD bus. Several tool suppliers jumped on board making debugging with NBD bus an established approach.

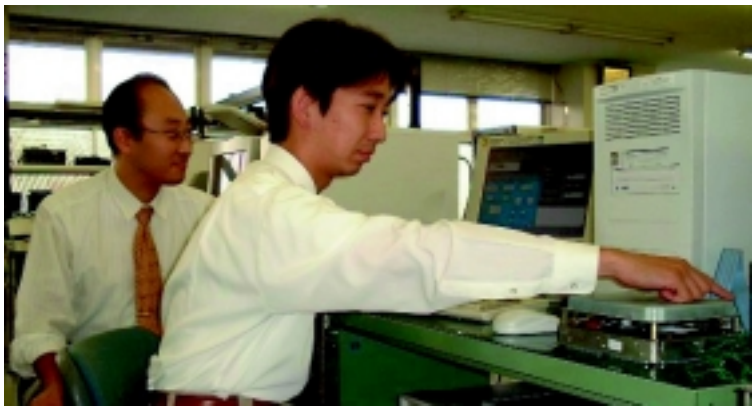
NBD - An Integral Component

Engine management control software implemented on modern ECUs is constantly evolving. This is due to, for example, the demand for more extensive control and diagnosis functions. This trend clearly points out the need for powerful 32-bit microcontrollers based on RISC architectures. For debug purposes, these controllers often accommodate corresponding debug units.

NBD basically allows the communication with an external tool, without any dependency on the current target microcontroller. Since NBD is onboard the controller, it is possible to monitor and change RAM data without stopping the CPU. For this reason, we evaluated the dSPACE customer solution: the NBD bus interface. This interface provides access to dSPACE Prototyper and links it to the controller. The question is: Is this approach applicable for performing function bypassing?

Test Arrangement

To realize bypassing, the computation of extracted functions, including the transfer time for I/O data, must be lower than the idle time of the ECU. Therefore, we set up a test arrangement to determine if the dSPACE system can meet this requirement. We used dSPACE Prototyper along with the NBD bus interface. The experiment software we used for testing was dSPACE's ControlDesk. The implementation of Simulink blocks was performed with Real-Time Interface (RTI). We incorporated several control functions on dSPACE Prototyper and encountered no problems whatsoever due to the lack of communication speed. The ability to access internal RAM without changing the ECU code is a great advantage that we highly appreciate. Thanks to the NBD approach, we can now use the same setup for any type of microcontroller with built-in NBD interface.



Benchmarking communication speed at DENSO: Takafumi Inaba and Yasuhiro Inaba.

A customer solution becomes a standard product

The dSPACE NBD bus interface has now become a standard solution, available off the shelf. Software support via Real-Time Interface (RTI) can be delivered for various types of microcontrollers.

Conclusion

To answer the question on applicability: Yes, we will certainly adopt the dSPACE approach, since it keeps us independent from the target platform. We do not need to change our development environment to perform bypassing since the NBD bus is integrated into new releases of 32-bit controllers. We intend to complete our design chain with TargetLink, which generates the production type C code for the ECU. We are confident that a development environment built around dSPACE tools is the solution to radically reduce our development time.

Manji Suzuki,
DENSO CORPORATION
Japan