TriCore[1]: a Hybrid DSP/Microcontroller Approach with Applications to the Automotive Industry

Fabrice Azevedo

Departamento de Informática, Universidade do Minho
4710 - 057 Braga, Portugal
PowerXplorer@Hotmail.com

Abstract. Emerging applications for embedded systems require a common feature set: integration (real-time control and signal processing), persistiveness (long-term architecture, flexible, re-usable and well supported) and competitiveness (performance and cost). This has driven the birth of hybrid architectures, merging RISC, DSP and μC architectures' best features, while overcoming their individual limitations. This communication lays out an analysis on the TriCore architecture, focusing on its core, innovative features and potential applications, namely those related to a specific application requirement from the automotive industry, the engine control.

1 Introduction

Today, embedded systems account for more than 75% of all microprocessor and microcontroller shipments. For the last decade, the increase in demand for embedded systems from three industry sectors (automotive, communications and general electronic devices) was the main driver for an architectural evolution, evolving from boards crowded of chips, programmed with specific hardware-dedicated software, to a 16-or 32-microcontroller tightly coupled with a DSP chip, programmed with suitable development tool sets [1].

The crave for new, smaller devices with more functionality (mobile phones, PDAs, digital cameras, wireless communication devices) or the requirements for efficiency, safety and environment for vehicles, together with the continuous strive for new products, has pushed the settled *status quo* to its ever re-defined limits, with more and more applications demanding higher system performance at a reasonable cost [1][2]. Furthermore, the complexity associated to the requirements demanded an unified architecture, in order to be able to respond to the different types of problems (DSP and/or μC) with minimum development time, and to justify the financial and human investments required by a specific architecture [3][4].

As so, the next evolution step was the convergence of the existing architectures:

- RISC, with its support for high-level programming languages, code optimizing compilers, relatively high performance with small die size but no real-time capabilities
- DSP, with its math intensive algorithmic capabilities, but poor interrupt response and inefficient compilers
- μC, with its multi-tasking features, fast interrupt response, code efficiency, but reduced DSP capabilities

resulting in a hybrid architecture characterised by a load/store Harvard architecture, SIMD capability, regular instruction set including DSP and μC features, fast context switching and direct control of peripherals [5].

[1] TriCore is a trademark of Infineon Technologies AG

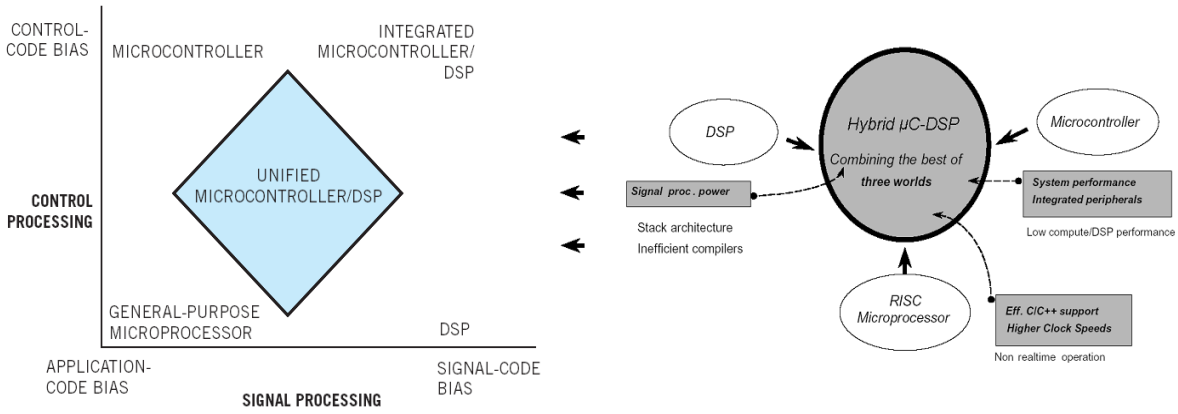


Fig. 1. Evolution trend of processor architecture for embedded systems (image courtesy of Cravotta and Infineon Technologies)

The next decade will also be for automotive electronic systems as demanding as the previous one, drived by enhanced safety features, entertainment systems, driver assistance, convenience functions and environmental constraints (namely on exhaust emissions).

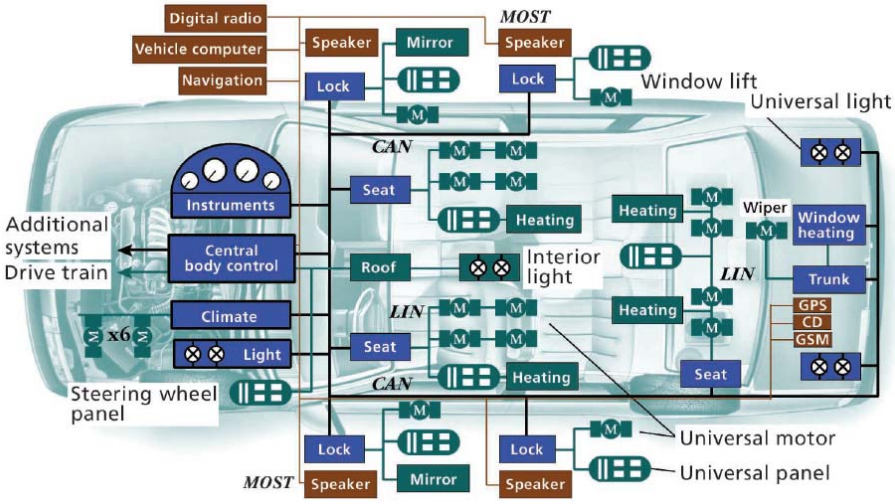


Fig. 2. Vehicle electronic systems overview (image courtesy of Leen and Heffernan)

The evolution of engine control was mainly led by the oil crisis in the late 70's and the growing environmental concerns since the 80's. With low emissions, high performance and low fuel consumption requirements, tighter control and precision are in order, unobtainable with mechanical devices.

With the development of more complex/accurate internal combustion engine models, supported by the novel sensors, new data signals were added to the basic throttle position data signal, leading to the replacement of the carburettor and distributor by microcontroller-based injection and ignition as actuators [6]. Control is now evolving towards more complex tasks, as pro-active determination of the deterioration of the control systems, increased complexity by adding new variables and granularity to algorithms.

This communication briefly analyses the internal combustion engine control paradigm, followed by an analysis of the TriCore architecture, its core and highlights. It wraps with a mapping between the paradigm requirements and the TriCore features, and concludes by extrapolating into the near future.

2 Engine Control

2.1 Overview

Engine control is concerned with the generation of correct and timely actions to run the engine at peak performance. Two basic forms are used: the static model (used for most applications) and the dynamic model. The static model is a set of relationships between inputs values and output values, defined by a table (lookup table). These values are fixed, based on the engine model, determined in laboratory conditions and for a specific set of operational modes (engine cranking, cold start-up, warm start-up, cold idle, cruising, acceleration, deceleration, diagnostics) [7]. The algorithm is easy to implement and fast to execute, but inaccurate under unexpected working conditions (variations on load, temperature,...). The dynamic model is the implementation of the internal combustion engine model, more accurate, with no or few assumptions, but slower and more complex.

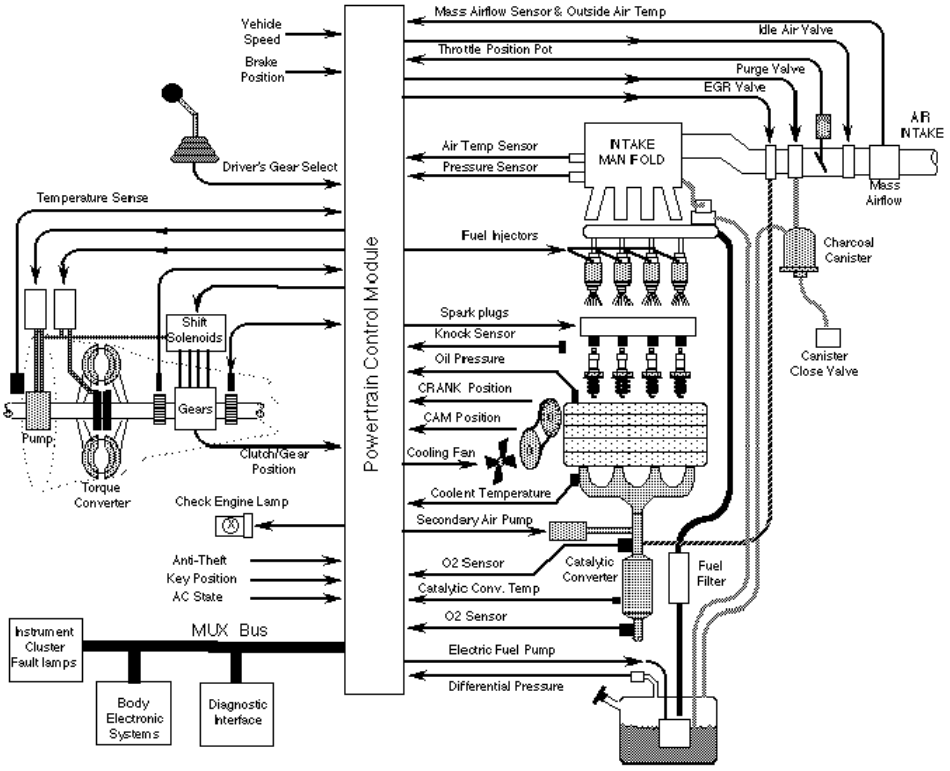


Fig. 3. Powertrain control overview (image courtesy of Motorola)

The engine controller is basically a state machine with a control algorithm, implemented with a microcontroller physically connected to the sensors and actuators. The control algorithm can be open-loop (variable control is static with no feedback) or closed-loop (effects of output data are taken in consideration with the input data for the next cycle). The data acquisition can be sampled (samples are taken at a constant rate by the processor for slow-changing inputs; example: water temperature) or event-driven (the sensor signal acts like an interrupt to the processor, which takes the signal as a sample, for fast-changing inputs; example: camshaft) [7]. Actuators are activated whenever necessary.

It is basically a continuous loop of:

- data acquisition through sensors, decoding included
- data processing (output values calculation, by lookup tables or model calculation)
- activation of actuators.

2.2 Sensors and Actuators

Sensors are input devices that provide the information required to the engine controller regarding a specific variable. They are generally classified by type according to the data to be acquired (rotational motion, pressure, angular and linear position, temperature, etc) and by technology/operating principle (variable reluctance, Hall effect, magneto resistance, piezoresistive, capacitive, and so on), which determines the type of data supplied to the controller and therefore its calculation (by table lookup or by value calculation, for example FIR filter) [6].

Actuators are output devices that enable the engine controller to perform the effective control, by changing the output variables of the system (like injection and spark timing).

Examples for sensors and actuators are illustrated in Fig. 3.

As previously referred, the output type of the sensors, and associated control, determines the type of calculation or sampling used by the controller for the data supplied. The data can be an analogic or digital value. For analogic values, the raw value (sample value) is used or it is subject of calculation, as mean value, variance, ..., according to the defined control model.

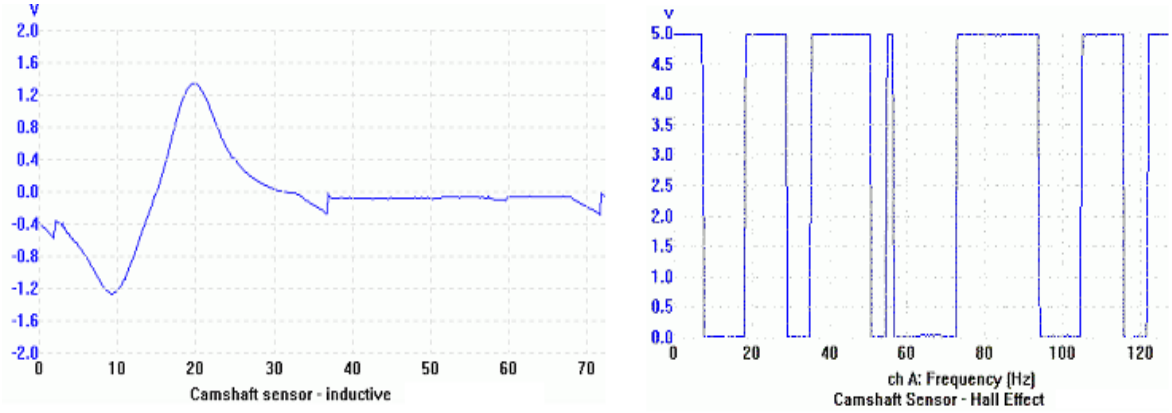


Fig. 4. Example of two data signals (different sensor technology) for camshaft rotational motion (image courtesy of Pico Technology).

3 Architecture and Innovative Features

3.1 Overview

TriCore is a superscalar architecture, combining RISC-based load/store design with a DSP-like Harvard memory architecture, built around a 32-bit fixed-point data path (with 64-bit program and data buses), a load/store unit and a program control unit [5].

TriCore provides 4 GB virtual address space divided into 16 segments of 256Mb. Program and data memory use two scratch pad RAM with a maximum of 128k each. It defines 4 different memory access types: access to local SPR0 and to local SPR1, cached access, and non-cached access (direct access to off-core memory). It uses independent L1 4-way associative cache for data and program, dynamically managed (2 ways: by writing a core register or using cache management specific instructions), with a maximum of 128k.

TriCore can perform either one load or one store per instruction cycle. Although it can retrieve up to four contiguous 16-bit data words, they are routed to a single register, forcing the programmer to split its constituents, not taking full advantage of its wide data path.

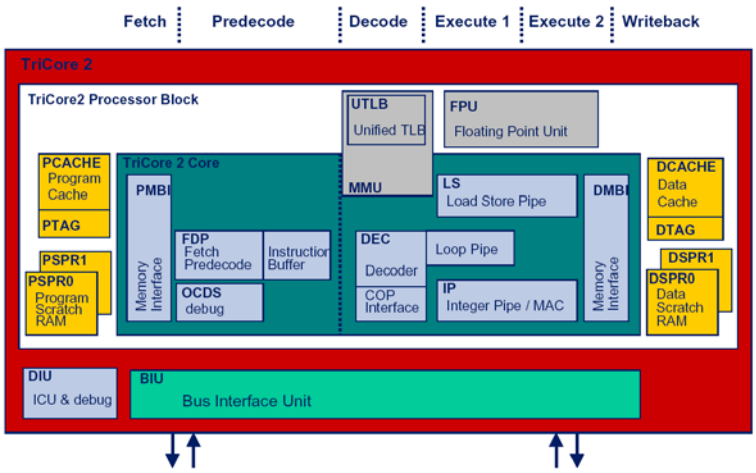


Fig. 5. TriCore 2 basic block diagram (image courtesy of Infineon Technologies)

TriCore uses an architecture with 3 pipelines, 2 major (integer and load/store) and 1 minor (loop). Pipeline depth (6), although small when compared with GPPs, is the now standard for high-end DSP processors.

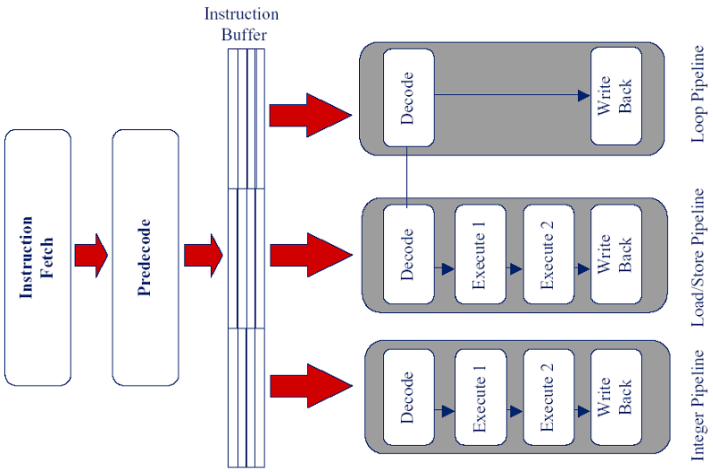


Fig. 6. 3 (6-stage) pipelines (image courtesy of Infineon Technologies)

The pipeline segregation (integer, load/store, loop) associated with the fact that most TriCore instructions complete in one cycle reduces the experience of pipeline stalls. Clearly, it also leads to a under usage of the processor if consecutive instructions fall under one pipeline [5]. Compilers will also experience some difficulty in presenting optimisation for this kind of architecture, and eventually some manually fine-tuning will be required from the programmer, in order to achieve peak instruction-execution rate.

3.2 Highlights

The highlights on TriCore are related to the fact that it is one of the few solutions available integrating RISC, DSP and μC. Beside this innovation, TriCore also brings some new features to the DSP world [5].

Mixed 16/32 Bit Instruction Set. By enabling the free intermix of 32 bit instructions for DSP and 16 bit instructions for control-oriented processing, TriCore enables reduced code-size, avoiding typical bit code waste associated to RISC's regular instruction sets.

Special Instructions. TriCore provides a rich instruction set for DSP (namely SIMD instructions) and control-oriented tasks. Instructions like multiply-add, absolute, min/max, rounding, counting leading, bit-field extract and comparison facilitate the programmers' work. SIMD instructions, native or packet data, generally for mathematical calculations, provide good execution on data parallelism algorithms, focused on DSP requirements.

Task Switching. TriCore claims low latency interrupt and fast context switch, by jumping execution to vectors in code memory instead of loading a new Program Status from a vector table and by achieving rapid data transfer between processor registers and on-chip memory.

Addressing Modes. Some interesting addressing modes are available: post-increment and pre-increment which may be used for forward or backward sequential access of arrays, indexed addressing which can aid on inner loops, circular addressing for accessing data values in circular buffers while performing filter calculation, and bit-reversed to access arrays used in Fast-Fourier Transforms algorithms.

4 Adequability Analysis

For analysis purposes on physical issues, a chip implementation of the TriCore architecture customized for engine control will be used, namely the T1765 based on the TriCore v1.3 specification, and evaluate against some particular requirements of engine control.

4.1 General

Unlike other popular embedded systems applications like mobile phones or PDAs, the environment for engine controllers does not have restrictions to size or power consumption.

Sustained clock performance is required through out the automotive temperature range (-40 Cº to +125 Cº).

4.2 Performance

Clock/Frequency. Engine control does not require frequencies as communications devices. For most variables required for engine control, 4 measurements per cycle are sufficient, with cycle duration in the order of milliseconds. The TC1765, running at 40Mhz, achieves a 25 nanoseconds instruction cycle time, with most instructions executing in one single cycle. Engine control algorithms, specially the ones based on table lookup, do not require such computational power.

Interface Capacity. In average, engine control uses around 10-15 sensors and around 5-8 actuators. TriCore provides a de-multiplexed bus with up to 32 address bits and 64 address bits with peak bandwidth of 800 Mbytes/s at 100 MHz for direct interconnection with the core, using modules with FPI bus interface that are able to connect with the environment specific sensors. The T1765 has for interconnection to the outside world 1 TwinCAN module, 2 A/D converters, 77 general-purpose I/O lines and synchronous/asynchronous serial channels.

Event Handling. Events and/or interrupts are caused by event-driven sensors (refer to section 2.1). TriCore supports interrupt (hardware) and call/return routines (software), using in average 2 instruction cycles for each operation (load or save).

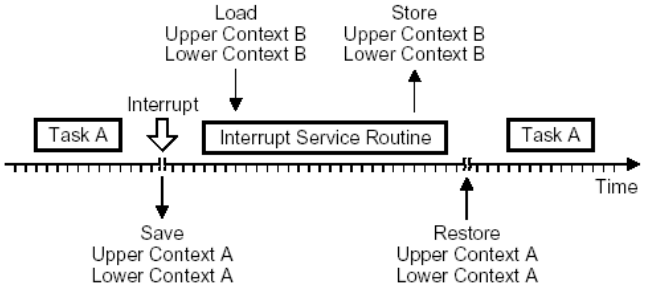


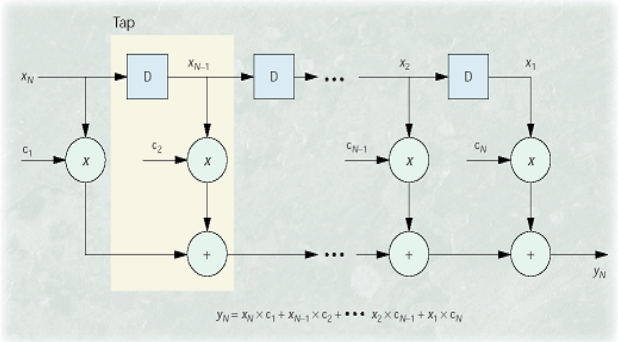
Fig. 7. Simple interrupt with context switch (image courtesy of Infineon Technologies)

4.3 Data processing

Engine control algorithms are tied to the type of sensors, and which calculations are required to be made. Standard algorithms or tasks already exist for each type of input value or output to be obtained. Examples are Angle-To-Time conversion, Bit Manipulation, CAN Remote Data Request, Fast Fourier Transform, Inverse Fast Fourier Transform, Finite Impulse Response Filter, Infinite Impulse Response Filter, Matrix Arithmetic, Pulse Width Modulation, Road Speed Calculation.

TriCore provides the specialized instructions (and introduces still some new ones) for optimising code execution (see section 3.2).

Let us take two representative examples: FIR filter and table lookup. For a FIR filter, on can associate TriCore's pipeline to its dual-MAC operations to execute 2 or even 3 operations in parallel and sustain, for 16 bit operand, four operations (two multiplies, two adds) per cycle rate.



Clock	Integer Unit	Load/Store Unit	Loop Unit
clock 1	-	Load C_0, C_1, C_2, C_3	-
clock 2	-	Load X_0, X_1, X_2, X_3	-
clock 3	MAC C_0X_0, MAC C_1X_1	Load C_4, C_5, C_6, C_7	Loop Start
clock 4	MAC C_2X_2, MAC C_3X_3	Load X_4, X_5, X_6, X_7	-
clock 5	MAC C_4X_4, MAC C_5X_5	Load C_8, C_9, C_{10}, C_{11}	-
clock 6	MAC C_6X_6, MAC C_7X_7	Load X_8, X_9, X_{10}, X_{11}	Loop
...	...	...	-
clock 130	MAC $C_{254}X_{254}$, MAC $C_{255}X_{255}$	-	-
clock 131	-	Store Result	-

Fig. 8. FIR filter example for sensor data calculation (image courtesy of Eyre and Bier and Infineon Technologies)

As referred in section 2.1, lookup tables are used to translate sensor information or for the calculation of formulas. The calculation by a lookup table is more efficient than the calculation by mathematical operations. TriCore supports addressing modes like base+offset, post-increment, pre-increment, and index addressing that enable code optimisation for table lookups.

5 Conclusions

TriCore is well equipped to face the challenges that lie ahead regarding engine control. Its performance for related GPPs tasks is not comparable to its DSP and μC capabilities. It is a specialized architecture and will always require some code tuning for upmost performance.

TriCore is a hybrid architecture, the result of the evolution trend for embedded systems processors, integrating GPPs, DSPs and μC features. The approximation to GPPs is progressing due to the increasing performance of the devices and the need for a better return on software development by focusing on the application and not on the hardware.

Embedded systems processors are very far from the complexity of the new GPPs, making them a poor relative, but they are characterised by their low cost, small die area and low power consumption. And for specialized tasks (namely DSP), they present equivalent performances and higher MIPS/MHz ratio. Furthermore, the internal complexity of GPPs restrains them of achieving real-time behaviour. Evolutionary speaking, hybrid architectures can be regarded as the first step towards total merging/integration.

Embedded systems have a promising future within the automotive universe. The future holds smarter cars with in-vehicle navigation systems and voice activated controls, safer cars with adaptive cruise control, electronic brakes and steering, and connected cars with GPS, instant local weather and transit information, Internet. Regarding engine control, electronics will always be present whenever a high degree of control is required. While the role of engine controllers, for the well-known internal combustion engine, depends on the development of new models and adequate sensors to evolve the control algorithm, developments in environmental-friendly alternatives as hybrid, electric systems or the exploitation of new solutions show themselves as a more enriching/challenging experience. As for the near future, engine controllers will be the primary tool for reducing engine exhaust emissions and fuel consumptions.

References

[1] Siemens: TriCore Architecture. White Paper (1998)

[2] Frantz, G.: Digital Signal Processor Trends. IEEE Micro, Vol. 20, No. 6 (2000)

[3] Cravotta, R.: Control and Signal Processing: Can one processor do it all?. EDN (March 2002)

[4] Eyre, J., Bier, J.: DSP Processors Hit The Mainstream. Computer (August 1998)

[5] Eyre, J., Bier, J.: Infineon's TriCore tackles DSP. BDTI, Inc., Vol. 13, No. 5 (1999).

[6] Flemming, W.: Overview of Automotive Sensors. IEEE Sensors Journal, Vol. 1, No. 4 (2001)

[7] Yeralan, S.: Engine Control and Event Scheduling. White Paper (2000)

[8] Leen, G., Heffernan, D.: Expanding Automotive Electronic Systems. IEEE Computer, Vol. 35, No. 1 (2002)

[9] Infineon Technologies: TriCore 2 Overview Handbook, v2.1. Infineon Technologies (2002)

[10] Infineon Technologies: TriCore 2 Architecture Manual, v1.0. Infineon Technologies (2002)