

## Improved AUTOSAR Based on Multi-Core Architecture and its Application in the Body Control

Bin Sun and Yue-li Hu

School of Mechatronic Engineering and Automation, Shanghai University,  
Shanghai, 200072, China

**Abstract:** In this study, a improved software architecture named M\_AUTOSAR based on the standard AUTOSAR is introduced which can upgrade the software system performance in multi-core microprocessors system for automotive system design. At the beginning of this paper, based on the analysis of the existing development trend of embedded multi-core microprocessors, we refined and improved infrastructure based on the standard AUTOSAR, increasing the independence inter-core resource management module-multi-core module. Then we describe the detail of the multi-core module features. At the end of this paper, the current practice of project based on the realization of the function module which described in this paper is introduced. And we set out the characteristics and advantages of the new framework through the automotive body control application.

**Key words:** AUTOSAR, body control, multi-core microprocessors

### INTRODUCTION

AUTOSAR is short for “Automotive Open System Architecture”. It defines a set of supporting distributed, feature-driven software development methods and automotive electronic control unit software architecture standardization program for application to different vehicle and platform, to improve software reuse and reduce development cost, as says in Schreiner and Goschka (2007).

Currently, the architecture is becoming an important part of production design criteria for many vehicle manufacturer, especially in the automotive electronics industry. The structure is gradually replacing the previous software design approach which is decentralized, non-inheritance. It's becoming a new generation of software design strategies. On the other hand, the embedded system is gradually moving from single core to multi-core processing, but the AUTOSAR architecture is lack of coordination for multi-core processing, which is inappropriate according the study of Nolte *et al.* (2009). Based on this background, this article made some improvement for AUTOSAR architecture to adapt the multi-core processing architecture. And this article described a success application of this improved architecture in our project.

### AUTOSAR ARCHITECTURE

The standard AUTOSAR architecture use layer structural design. In general, each layer can only use the under layer interface. And the layer can only be used to

provide a layer interface for above layer. Shown in Fig. 1, the figure name from top to bottom is: Application layer, RTE layer, System services layer, ECU abstraction layer, Micro-controller abstraction layer and Complex drive module.

Application layer contains all the application software components, the software components in this layer, regardless of the internal communications or access to ECU resources, are both done by RTE. The software realizations in this application layer are independent to microprocessors, ECU and hardware. RTE's purpose is to make the application layer software has nothing to do with specific hardware, all communication services in application layer is provided by this layer.

Above RTE layer, the software architecture is component-based architecture, and the software architecture becomes hierarchical under this layer.

Services layer includes communication, services, operating systems and other modules. It provide basic services for the basis software modules and the application modules. According the AUTOSAR architecture (Schreiner *et al.*, 2009), it provides services include: operating system services, network communications, and vehicle management services, storage services, diagnostic services and ECU state management.

ECU abstraction layer provide a unified interface to access the peripheral. This layer shield the device eith the internal or external for the upper chip. It includes an external device drivers and so on.

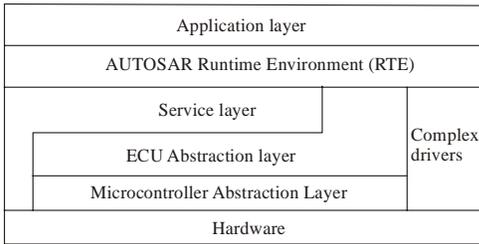


Fig. 1: standard AUTOSAR software architecture hierarchy

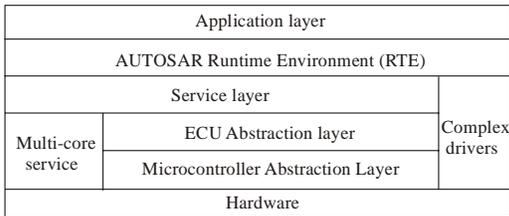


Fig. 2: M-AUTOSAR architecture

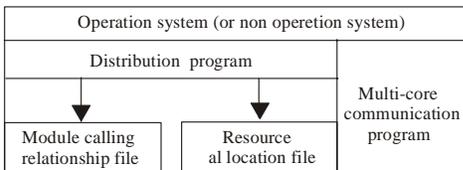


Fig. 3: Internal architecture of Multi-core Service module

Microcontroller abstraction layer is a direct contact layer with the actual hardware, it's the lowest layer in software infrastructure. Drivers are included in this layer, which is used to access the microcontroller peripherals or memory mapped to the microcontroller inside the device. According to the work of Pierre-Emmanuel *et al.* (2007), Complex driver layers are different from others, they cannot be classified as a layer, and therefore are listed separately, including the driver module dealing with complex sensors and actuators, which have special functions and time requirements.

From the above description, we know that the AUTOSAR basic software architecture is a hierarchical structure, in a single-core chip architecture can operate normally. If MCU is a dual-core or multi-core, how to better use the multi-core performance to make the system more stable, this may need some improvement to the structure to be able to better meet the new hardware environment, although we can use the operating system's approach to multi-core resource allocation.

### THE IMPROVED AUTOSAR ARCHITECTURE

**The improvement for AUTOSAR architecture:** AUTOSAR architecture does not involve a specific

single-core or multi-core. Their idea is that all the multi-core operation of the system is shielding in the RTE operating system. And the result is the average allocation of resources by the operating system often cannot effectively meet each processing core features request (mainly the case of heterogeneous processing core). This is one of the research results in Technical Overview 2.0.1, AUTOSAR GbR, (2006). And according to the work of Entities *et al.* (2009) and Pimentel (2007), the various communication links between the chips is also not closely, because it is a unified operating system itself, and not optimized for each core, they will be a waste of system resources as a result.

Since AUTOSAR defines the various components of the module design, we believe that core-communication module, core-distribution of resources module, core-call relationship module can be a separate stand-alone module in the software architecture. It will ensure that automotive software designers clearly know this part requires specific design. And it can also ensure the case will not occur such as uneven distribution of resources between cores in the software product. The improved AUTOSAR architecture as shown in Fig. 2 and we named this new AUTOSAR architecture as M-AUTOSAR (Modified AUTOSAR).

M-AUTOSAR architecture increases the Multi-core Service module, which provides the calling information related to multi-core, resource allocation information and inter-core communication information. These modules can be executed by the operation system. The internal components of this composition include the following components: module calling relations file, resource allocation file and inter-core communication program and a distribution program.

Working relationship between the various modules as shown in Fig. 3. Module calling relationship file is responsible for storing the distribution file, which describes the operation allocation for each program of each processor. The resource allocation file is responsible for the system resource using moment and the using processor. The allocation program calls these two files. The allocation program is used by the operating system when there is an operation system which is used in the project. If there is not an operation system existed in the project, the main program will call the allocation program in the initialization stage for resource allocation.

The multi-core communication program includes the interlock program and data exchange mode between cores, it is used by the operating system and called by the upper application.

**Module calling relationship file:** Module calling relationships. It describes the corresponding processor core id for each module. And the corresponding interrupt information for each module is described in this file also.

These information can be used to help the programmer direct optimized the program by the processor associated with the use of different functional modules

Here is an application of this project using C language to make the Module called relations file. This is a part of the text file:

```
//Model call File
Void Main();
Byte* RxCAN0() MCU2 INT 0X42;
Void TXCAN0() MCU2 INT 0X43;
Byte Timer() INT 0X16;
```

The default processor is MCU1, which is the main processor, the compiler is concerned, it will follow us to connect the processor set and generate the corresponding code for each processor.

**The design of resource allocation file:** Resource allocation file is a list file, including the resource name and which one uses these resources. Resource allocation file also includes a hardware interrupt handlers for interrupts. Of course, some interruption can be used by any one of multiple processors for processing. It will also be in the file specified.

Below is a part of a instances files for the Resource allocation file. This part include the allocation of pin, in addition, also includes register allocation and so on.

```
//Resource Allocation Document
//Pin resource distribution
PM0 RXCAN0();
PM1 TXCAN0();
TIOS0 Timer();
CFORC0 Timer();
```

All the resource was included in this file except the MCU. Of course, with the build system supporting, these two files, will automatically generated by the compiler.

**The design for the Program distribution module software architecture:** Program allocation module based on the above two files and this module is response for the resource allocation and resource management. The main module can be performed either in support of the early completion of the build system or by ain function. In our project, we chose the software module called by the main function because the build system still does not support the processing of these files.

**The software architecture design for multi-core communication program:** Multi-core communication consists by two parts: fist is multi-core tasks synchronization; the second is multi-core tasks communication.

Among them, the multi-core tasks synchronization, including semaphores synchronization and message synchronization. The semaphores synchronization needs the interlock mechanism for RAM space access, which is consistent with the theory of classical programming semaphore interlocking relationship. The processing for message synchronization will need to be marked on the message processing step, including message processing steps finish situation and other information.

Multi-core tasks communication include message queue, non-queue message and send-receive notification. Among them, the processing for queue message or non-queue message all needs the adding and deleting the queue information. This part function is mainly finished by the application software, sending and receiving notifications is used for the sent processing and receive processing during the course of the program record. On the one hand, beginning and end of message sending and receiving need to be treatment, on the other hand is the confirmation between the process of sending and receiving.

As shown in Fig. 4 is a multi-core tasks communication process in our projects. Processor A and Processor B is two processors in a multi-core processor. The process begin with lock data in each processor, and end with unlock data after the data has been sent or received. Between lock and unlock data, the other processor should not be access the locked data to avoid data pollution.

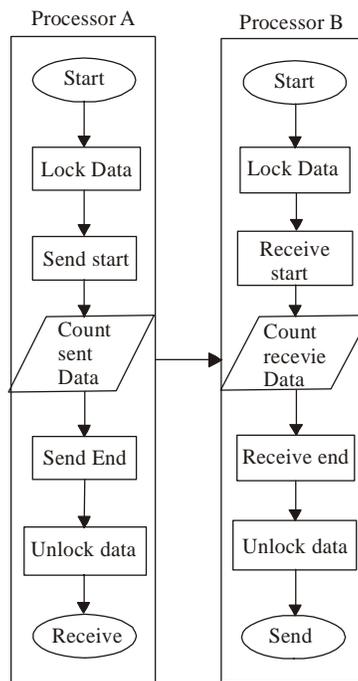


Fig. 4: Message queue synchronization

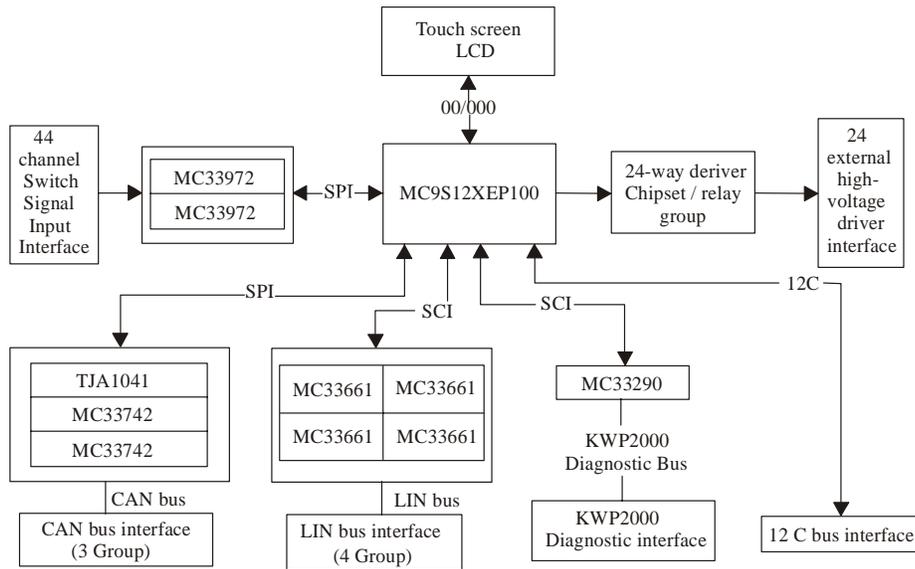


Fig. 5: BCM System Hardware Structure

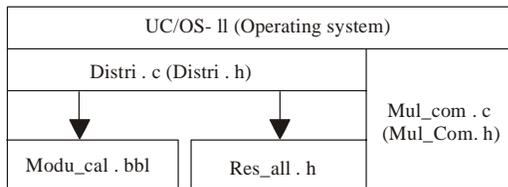


Fig. 6: Project application for Multic-core Service Module

**Improved benefits:** After the improvement, the throughput AUTOSAR architecture has the advantage, which is that the resources of system dual-core processing is under the effective control of the software designers and the system dual-core resources can be adequate use. Besides, the processing of dual-core can be easy to optimized, the system efficacy can be upgrade 30% and the system can be transplantation easily from single-core to multi-core, which need to do is just modify the Multi-core module to set the content. Of course, due to different modes of multi-core communication are not the same, for multi-core program communication will inevitably need to be optimized.

**BODY CONTROL APPLICATION**

We introduced the above M-AUTOSAR architecture to our body control design. We use the Freescale's dual-core processor chip MC9S12XEP100, there are two processors, including the main processor and a coprocessor S12 XGATE, the processing performance of two processors are not identical. Therefore, when use the embedded operating systems, such as uC/OS-II or Linux, the system will be balanced on the bias. In addition, when

we use the Multi-core Server to finish the distribution for interrupt handling, the performance is better than the performance of simply using operating system.

**Hardware design:** As shown in Fig. 5, the BCM system is including CAN\LIN bus, KWP2000 diagnostic interface and some other resource. These resources are directly link to MC9S12XEP100. And these resources are used by the MCU, which include S12 core and a XGATE core. Both cores can operation program and make some interrupt handling. The MC9S12XEP100 is a heterogeneous multi-core chip. S12 is main processor, and XGATE is a co-processor. In our project, S12 and XGATE are both used by uC/OS-II operation system, but effect is not same. We refined the uC/OS-II operation system according our improved AUTOSAR, so the operation system run base on the both core.

**Multi-core service module design:** We use the M-AUTOSAR architecture to this program design. The system software architecture we used is shown in Fig. 6. The uC/OS-II is modified from standard uC/OS-II operation system. And some function module is run in XGATE core. Below the Operation System is a istri.c program file as distribution module. As above description, the distribution module use the od\_cal.bbl as a module calling relationship file and es\_all.h as a resource allocation file to help the uC/OS-II operation system to allocation system resources. ul\_Com.c is a communication program file to be called by application program.

As an example for communication between multi-core, Figure 7 shows a CAN bus data process using both the main processor S12 core and the coprocessor XGATE.

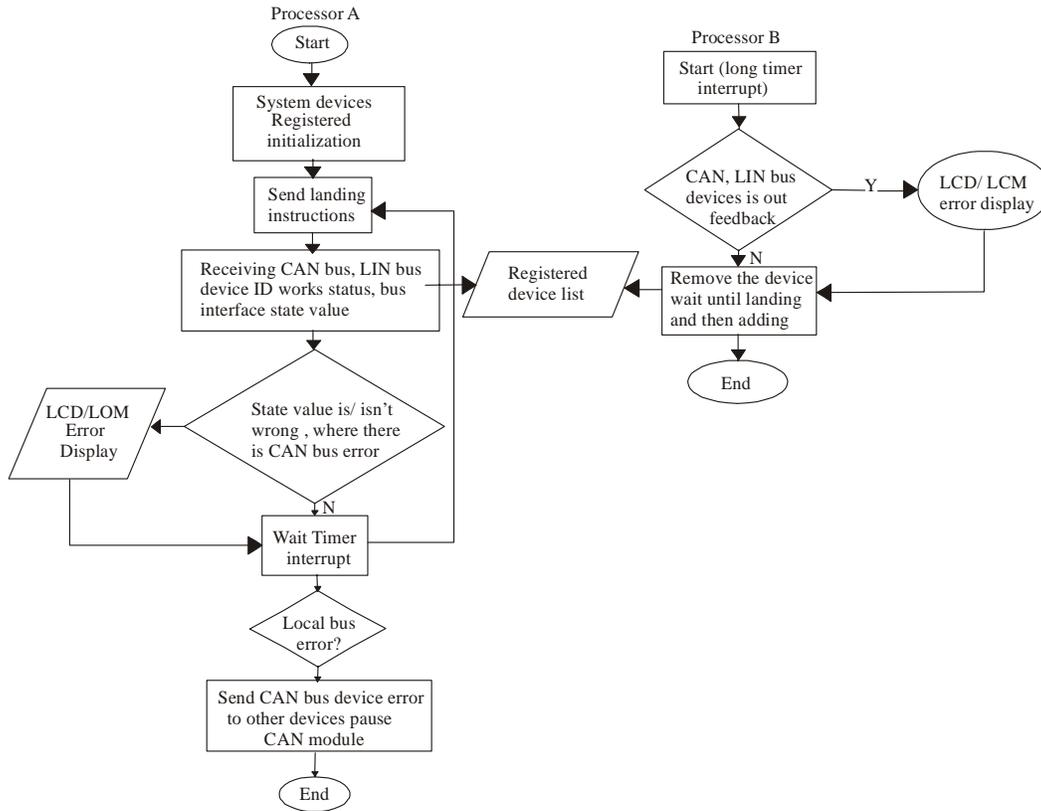


Fig. 7: CAN bus operation based on M-AUTOSAR structure

Table 1: Test result

Test Item	Tradition AUTOSAR	Modified AUTOSAR
CAN bus data operation	100 us	68 us
LIN bus data operation	1 ms	0.55 ms
SPI data operation	10 ns	9 ns

The main function is calling the Multi-core Service at first. And then all of the programs will process using Multi-core Service Module. In Fig. 7, the multi-core communication program is called to exchange information between processor A and processor B. In this operation process, all the processors are used in higher effectively.

**Application effective:** According the way of Moon *et al.* (2009), we test the tradition AUTOSAR and the M-AUTOSAR in this study. And the test result is shown Table 1.

All tests are using the uC/OS-II operation system, and the operation system is optimized by designer with multi-core.

As shown in Table 1, we found the M-AUTOSAR can upgrade the system feature, and the feature is affected by the hardware feature also. The upgrade for different communication protocol is not the same one.

## CONCLUSION

Now, microelectronics industry has moved from the pursuit of speed into the pursuit of a number of internal processing cores. But there isn't a complete system for multi-core processing software theory and programming methods. This study refined and improved the standard AUTOSAR architecture, based on the analysis of standard AUTOSAR software architecture. We add a single module for multi-core resource management. We describe the software module for detail function in this paper. At the end of this paper, based on the current practice of project realization, the function modules are described. We explained the characteristics and advantages of this structure in the body control. This study can be able to help for the software design and programming approach in the multi-core environment, especially for the automotive software design.

## **ACKNOWLEDGMENT**

This study was supported by IC Special Foundation of Shanghai Municipal Commission of Science and Technology (Grant NO. 09706201300), Shanghai Municipal Commission of Economic and information (Grant No.090344), Shanghai High-tech Industrialization of New Energy Vehicles (Grant No.09625029) and Graduate Innovation Fund of Shanghai University.

## **REFERENCES**

- Entities, R., R. Long, H. Li, W. Peng, Y. Zhang and M. Zhao, 2009. An approach to optimize Intra-Ecu communication based on mapping of autosar. Proceedings inter. Conf. Embedded Software Sys., ICCESS., pp: 138-143.
- Moon, H., G. Kim, Y. Kim, S. Shin and K. Kim, 2009. Automation Test Method for Automotive Embedded Software Based on AUTOSAR. 4th International Conference on Software Engineering Advances, ICSEA, Includes SEDES: Simposio para Estudiantes de Doutorameto em Engenharia deSoftware, pp: 158-162.
- Nolte, T., I. Shin, M. Behnam and M. Sjödin, 2009. A Synchronization Protocol for Temporal Isolation of Software Components in Vehicular Systems. IEEE Trans. Indus. Info., 5(4): 375-387.
- Pierre-Emmanuel, H., D. Anne-Marie, S. Faucou and Y. Trinquet, 2007. Adequacy between Autosar os specification and real-time scheduling theory symposium on industrial embedded. Systems Proceedings, SIES., pp: 225-233.
- Pimentel, J.R., 2007. An incremental approach to task and message scheduling for autosar based distributed automotive applications. Proceedings-ICSE Workshops: 4th International Workshop on Software Engineering for Automotive Systems. SEAS., 07: 1-7.
- Schreiner, D. and K.M.G. Oschka, 2007. A component model for the autosar virtual function bus. Proceedings International Computer Software and Applications Conference, 2: 635-641.
- Schreiner, D., M. Schordan and J. Knoop, 2009. Adding Timing-Awareness to Autosar Basic-Software-A Component Based Approac. Proceedings IEEE Inter. SymposiumObject/Component/Service-Oriented Real-Time Distributed Comp. ISORC., pp: 288-292.
- Technical Overview 2.0.1, AUTOSAR GbR, 2006. Retrieved from: <http://www.autosar.org/download/R2.0/AUTOSAR-TechnicalOverview.pdf>.