International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 10, October- 2014

Multicore OS Design based on AUTOSAR for MPC5668G

Roshin Angel Cherian, Tressa Michael
Electronics and Communication Engineering
Rajagiri School of Engineering & Technology
Kochi, Kerala, India

Laya Raj
Embedded Product Design
Tata Elxsi Ltd
Trivandrum, Kerala, India

***Abstract* — This paper introduces the design of an operating system based on AUTOSAR [Automotive Open System Architecture] for MPC5668G. The designed operating system is for a dual core micro-controller. As the number of ECUs increases, it further increases the design complexity of automotive control systems. Multicore processors on a chip, have emerged to be the main computing controllers not only for embedded control systems but also for high end servers. Hence more centralized architecture designs can be implemented for automotive control systems with the help of multicore processors. There has been a steady rise in the amount of computational power required for an ECU. This requirement has been met with architectural changes in the hardware and by increasing the clock rate. Key to exploit multicore is the decomposition of the application tasks and interrupts handlers into smaller components that communicate with each other. The distribution of components is carried out over the available computing resources. AUTOSAR supports the automotive ECU software development, which is based on the idea of static (i.e. Compile time) configuration. Changing it's configuration of the system whilst it is running is too expensive and complex. So the issuing of workload to cores is static and this implies that the communication between the cores is also static. The MPC5668G microcontroller is designed to address the need for more integration of electronic features within the vehicle. In the core architecture, e200z650 acts as the CPU and e200z0 as the input output processor (IOP). The controller is developed by Freescale on 32 bit Power architecture for primary use in automotive and industrial control systems. This MCU family built for automotive gateway and high end body applications. Also, it is user compatible one.**

Keywords - AUTOSAR; Automotive domain; OSEK/VDX; ECUs; Power architecture; Multicore

I. INTRODUCTION

Embedded system applications are mainly used today in aerospace, automotive areas. The number of electronic control units (ECUs) in modern vehicles has continuously increased in last few decades. The design complexity of automotive control systems increases due to the introduction of advanced functions like anti-lock braking systems, adaptive cruise con- trol and climate control put higher computational demand on ECUs. Multicore processors on a chip have become to be the main computing controllers not only for high end servers but also for embedded control systems. More centralized architecture designs can be implemented for control systems with the help of multicore processors. With the help of in- vehicle network like CAN and FlexRay, it is expected that computational control tasks of different functions can share one ECU.

Recently multicores become economical for mass produced deeply embedded systems due to the ability to put more than one core on the same piece of silicon along with memory and peripherals. Key to exploit multicore is the decomposition of the application tasks and interrupts handlers into smaller com- ponents that communicate with each other. The distribution of components is carried out over the available computing resources. AUTOSAR supports the automotive ECU software development, based on the idea of static (i.e. compile time) configuration. Changing it's configuration of the system whilst it is running is too complex and expensive. So the issuing of workload to cores is static and this implies that the communication between the cores is also static.

This work is for the designing of a real time operating sys- tem based on AUTOSAR architecture working on a multicore although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

II. LITERATURE SURVEY

A. Existing Standards

Specialized real time operating systems called Vehicular Application Specific Embedded Operating Systems (VASOS) are developed, after the intervention of application specific operating systems. Some VASOS are VxWorks, QNX, and PSOS etc. They provide functions of vehicle domain such as device drivers, fundamental network functions. OSEK/VDX (Open systems and corresponding interfaces for automotive electronics/Vehicle Distributed executive) is an open vehicular industry standard. It provides environment for developing and redeveloping ECU software and improve compatibility of those applications. OSEK and its successor AUTOSAR plays an important role in software development of automotive domain.

1) VxWorks: VxWorks is a versatile RTOS which offers the developer a great deal of control over scheduling of tasks and synchronization. It is one of the strongest points of the operating system and for RTOS this is essential one, because it is a need to make sure things happen when they should and that they do not interfere with other, i.e. aerospace or healthcare with a safety critical system. One of the weakest points of VxWorks, is the allocation of memory with fragmentation. It is due to the first fit algorithm used in the allocation of memory space.

2) QNX: QNX Neutrino is widely used as the basis for automotive electromechanical components, industrial control systems, medical instruments, nuclear power plants, defense systems and other mission critical applications. It' s software architecture is the key to building the flexibility that automakers need. It helps the software strategy that maximizes their productivity.

3) PSOS: PSOSystem is a modular, high performance real time operating system designed specifically for embedded microprocessors. Based on open systems standards, it provides a complete multitasking environment. PSOS is well suited to situations in which one wants to support many applications with different or conflicting requirements. There is no much more difference between PSOS and Vxworks. Because PSOS is not allowed to develop more since WindRiver owned it.

4) OSEK/VDX: OSEK/VDX (Open systems and corresponding interfaces for automotive electronics / Vehicle Distributed executive) is an open vehicular industry standard. It provides environment for developing and redeveloping ECU software and improve compatibility of those applications. OSEK OS provides a sufficiently flexible scheduling policy to schedule AUTOSAR systems. It is a mature specification and implementations are used in millions of ECUs worldwide. OSEK time OS and the HIS Protected OSEK are immature specifications that contain concepts necessary for AUTOSAR and satisfy specific application domains. Advantages of OSEK/VDX are portability and reliability. OSEK/VDX is an operating system meant for distributed embedded control units.

III. MOTIVATION

The recent developments on AUTOmotive Open System ARchitecture (AUTOSAR) have established several standards for automotive software and hardware designs, which explains the directions for designing centralized architecture with multicore ECUs for automotive control systems. There has been a steady rise in the amount of computational power required for an ECU. This requirement has been met with architectural changes in the hardware and by increasing the clock rate. Both of these have resulted in ECU that provide more computational power but also have increased power dissipation. The heat thereby generated must be conducted away from the ECU. So increasing the clock speed does not result in more power for free. Instead it incurs a significant cost in terms of the thermal design and manufacture of the ECU. A simple architectural change to make to the microprocessor is to add more cores (or CPUs). A core takes a relatively small amount of the die area and overall power consumption. So adding more cores is a simple way to increase computing power for a specific clock frequency without significantly increasing the power dissipation. However, the software architectural implications of adding more cores must be taken into account.

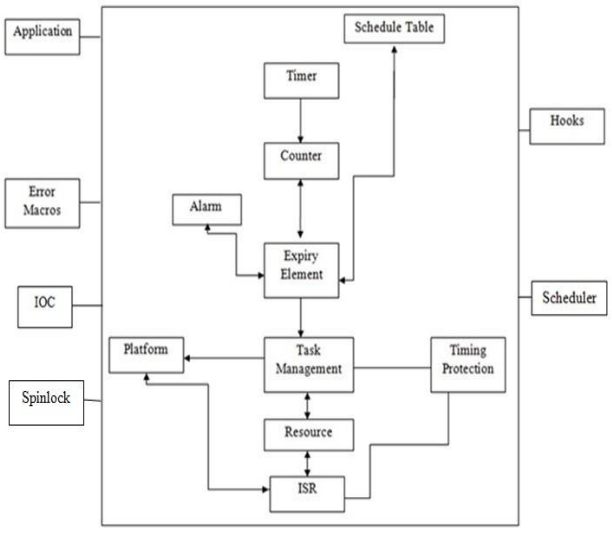
Recently multicores become economical for mass produced deeply embedded systems due to the ability to put more than one core on the same piece of silicon along with memory and peripherals. Key to exploit multicore is the decomposition of the application tasks and interrupts handlers into smaller components that communicate with each other. The distributions of components are carried out over the available computing resources. AUTOSAR supports the automotive ECU software development, based on the idea of static (i.e. compile time) configuration. Changing it's configuration of the system whilst it is running is too complex and expensive. So the issuing of workload to cores is static and this implies that the communication between the cores is also static.

IV. BLOCK DIAGRAM AND DESCRIPTION

A. OS Modules:

The operating system can be divided into several modules according to their functionality. These modules are essential for the smooth working of OS on the hardware. Some of the modules are hardware dependent and others are not. For regular functioning, each of the modules may depend one another. The modules of OS are Timer, Counter, Alarm, Schedule Table, Events, Tasks, ISR, Scheduler, Resources, Stack Monitoring, Applications, Protection facilities, Hook functions, Error macros, Platform, Spin lock mechanism, IOC, Locatable Entities and Multi core start up concept. OS as a whole does not support any post build variants. Only link time and pre - build variants are supported. The block diagram which describes the OS modules is given below.

Fig. 1 OS Modules



V. EXPERIMENTATIONS

It is possible to split the whole work into several pieces of small works and called them as tasks. That means, task is a simple program or part of an execution flow that competes with other concurrent tasks for processor execution time. It is an independent entity. Each task is assigned with a priority. With the correct ordering of these small programs make the whole work into a real one. Hence some measures should be taken, which keeps them in real time environment. Tasks are pre-assigned to both of the cores. According to the priorities, these tasks are getting executed.

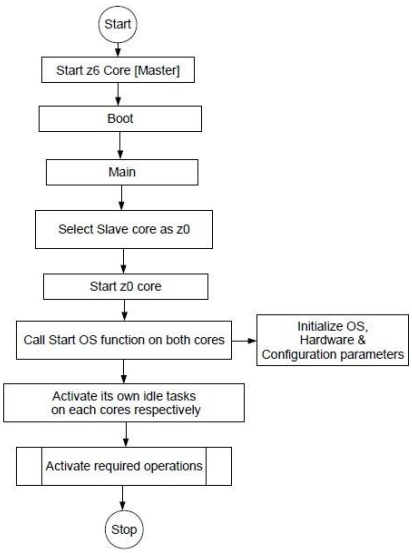
A. Using Alarms:

Tasks can be initiated with the help of alarms. Alarms can be made available for the execution of tasks according to the specifications given. Resources can also be used for the above mentioned purpose. It helps the low priority tasks to execute without keeping them in waiting state.

VI. FLOW CHART

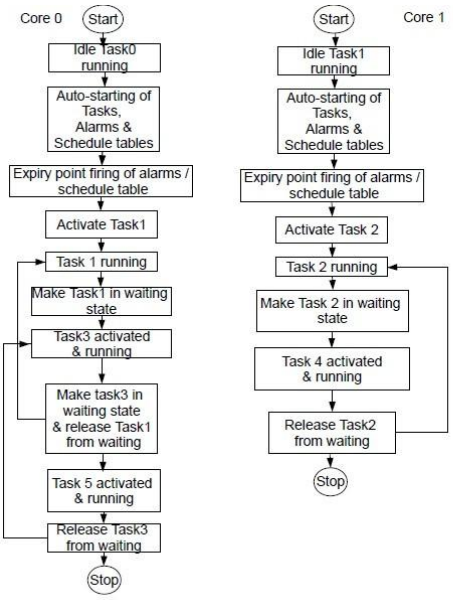
The functional flow of program execution on a real hardware is explained below with the help of flow charts. The Master core, z6 get activated manually. Then after, z0 called as Slave core activated by z6. Further flow of executions is shown here.

Fig. 2 Flow Chart 1



The second flow chart explains the working of a sample program. From the diagram, we can understand that the tasks are assigned to both cores and get executed in parallel manner. It is based on priority. After the activation of idle tasks on both cores, executions of required tasks are carried out. The flow chart is shown below, which explains it clearly. Core 0 and Core 1 indicate Master core and Slave core respectively.

Fig. 3 Flow Chart 2



VII. IMPLEMENTATIONS

Operating System design was implemented with the help of StarUML design tool. Freescale MPC5668G development board is used for hardware implementation and for compila- tion, WindRiver Diab Compiler is introduced. Then for flash- ing and debugging of the board, Trace32 – LAUTERBACH debugger tool used.

VIII. PARAMETERS

There are some parameters related to real time operating system. These parameters define the characteristics of the OS. Some of the parameters are given in the table.

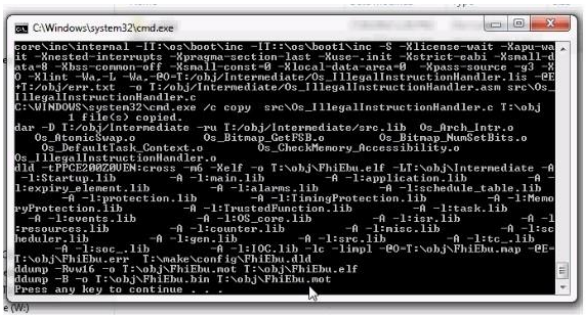
Table 1 – Parameters of OS

Features	Characteristics
Scheduling	Pre-emptive Priority based Algorithm
Memory footprint ROM RAM	 23.432KB 35.676KB
Resource Scheduling	Priority Ceiling Protocol

IX. SIMULATION

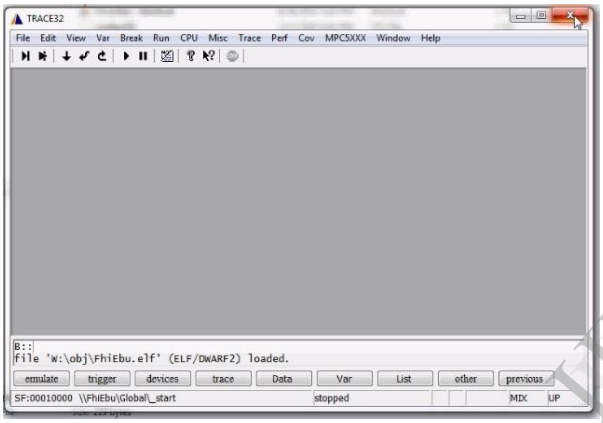
As per the part of the work, an operating system based on AUTOSAR for a multicore microcontroller was developed. That means, the two cores are ready for executing tasks simultaneously, which are defined offline. The first step of this work was the code generation. It includes coding of functions, include files, make files, configuration files etc. After this, compilation carried out with the help of WindRiver Diab Compiler. Thus, .elf file generated as the executable image file. The snapshot of elf file generation is given below.

Fig. 4 .elf file generation



Then, this file gets flashed on to the board with the help Trace32 - LAUTERBACH debugger. After flashing, the elf file loaded into the hardware and ready for execution. The snapshot is given below.

Fig. 5 Flashing



Then, for knowing the execution flow, breakpoints were set and carried out debugging process. During debugging, it can able to watch the program flow based on the tasks defined for execution. Also, the given below snapshots show context switching between tasks of lower and higher priorities. The following are the snapshots during the debugging process.

Fig. 6 Execution of StartOS function

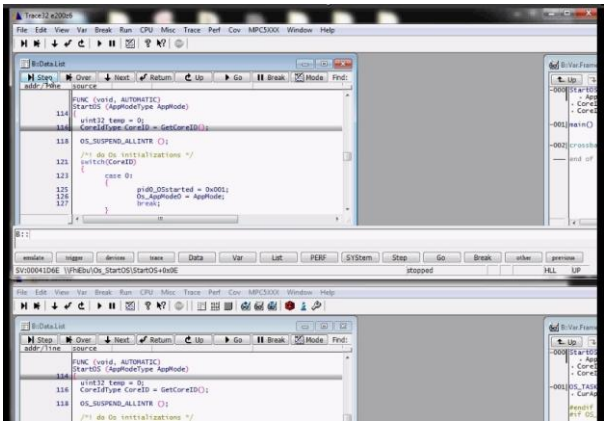


Fig. 7 Execution of Idle Task0

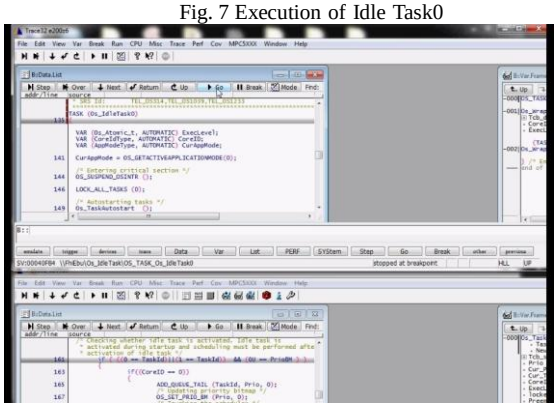


Fig. 8 Execution of Idle Task1

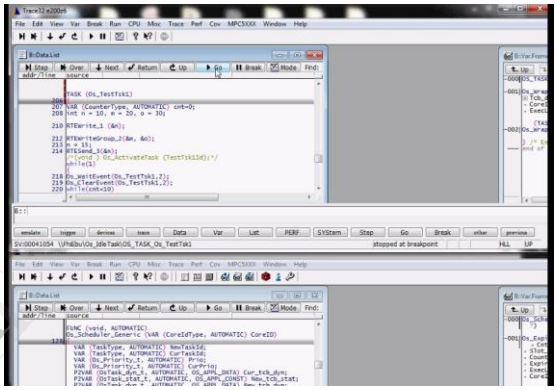


Fig. 9 Execution of Task1

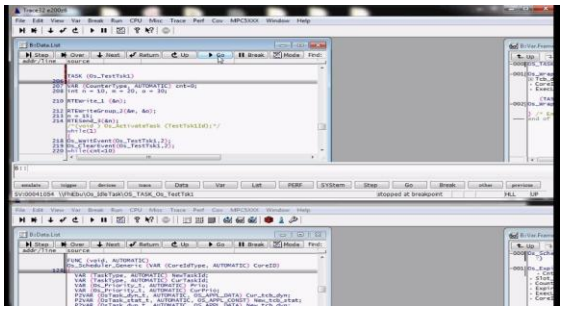
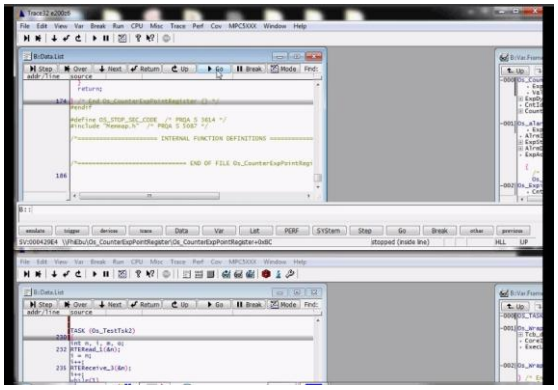


Fig. 10 Execution of Task2 - Core1



Then executions of Task 3 and task 5 are carried out by Core 0. After that Task 4 executed by Core 1. With the help of break-points, these executions can be examined.

X. FUTURE SCOPE

The design of Hybrid cars need to know the state of the vehicle to control in an optimal way. There is a need of central software. Communication between cars, between cars and road infrastructure will be able to implement in future. Standardization of vehicle platform with the help of AUTOSAR is needed at that time. Operating system plays an important role in the development process.

XI. CONCLUSION

As the increasing complexity of Electrical and electronics circuits, there is need of circuits which have more efficiency in reduced space. Thus ECUs are developed for the need of controlling systems. AUTOSAR plays an important role in software development of automotive domain. The concept of AUTOSAR is introduced to standardize the communication and non-functional software. The non-functional software includes the Run time environment and the Basic software. The functional software i.e., the application software is standardized by its interfaces. Without the problem of compatibility, it can be able to design soft and hardware components for an application. That means, reduced integration cost with standardization. As by the introduction of new concept, safety measures are also updated.

ACKNOWLEDGEMENT

The authors would like to thank Mr. Jaison Jacob, Mr. Anoop Thomas, RSET, Kochi, Kerala, India for their contributions to this work. Also, we express gratitude towards Mr. Anurag R, Competency Manager, Tata Elxsi Ltd., Trivandrum.

REFERENCES

1. AUTOSAR 4.0.3 - Specification of Operating System, http://www.autosar.org/
2. OSEK/VDX - Operating System Specification 2.2.3, http://www.osek- vdx.org/
3. Data sheet of Freescale MPC5668G/E
4. StarUML User Manual
5. WindRiver Compiler User Manual
6. Trace32 - LAUTERBACH debugger Manual
7. VxWorks - http://www.windriver.com/products/vxworks/
8. QNX - http://www.qnx.com/products/neutrino-rtos/index.html
9. pSOS - http://en.wikipedia.org/wiki/PSOS
10. JaeYoung Kim, JungWook Lee, Jeongho Son, Kee-Koo Kwon and Gwangsu Kim, Lightweight AUTOSAR Software Platform for Automo- tive, IEEE International Conference on Consumer Electronics (ICCE), January 2012.
11. Simon P. Brewerton, Natalia Willey, Swapnil Gandhi, Thorsten Rosen- thal, Claus Stellwag and Matthieu Lemerre, Demonstration of Automotive Steering Column Lock using Multicore AutoSAR Operating System, SAE International, April 2012.
12. Florian Kluge, Mike Gerdes and Theo Ungerer, AUTOSAR OS on a Message-Passing Multicore Processor, 7th IEEE International Symposium on Industrial Embedded Systems (SIES12), June 2012.
13. Sylvain Cotard, Sebastien Faucou and Jean Luc Bechennec, A Data-flow Monitoring Service Based on Run-time Verification for AUTOSAR OS: Implementation and Performances, High Performance Computing and Communication and 2012 IEEE 9th International Conference on Embed- ded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on, June 2012.
14. Kim Gruttner, Philipp A. Hartmann, Philipp Reinkemeier, Frank Oppenheimer and Wolfgang Nebel, Challenges of Multi- and Many-Core Architectures for Electronic System Level Design, Embedded Computer Systems (SAMOS), 2011 International Conference on, July 2011.
15. Senthilkumar.K and Ramesh Ramadoss, Designing Multicore ECU architecture in vehicle networks using AUTOSAR, Advanced Computing (ICoAC), 2011 Third International Conference on, December 2011.
16. Tong Li, Paul Brett, Rob Knauerhase, David Koufaty, Dheeraj Reddy and Scott Hahn, Operating System Support for Overlapping - ISA Heterogeneous Multi-core Architectures, High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on, January 2010.
17. Ke PEI, Gang ZHANG and Qing CHANG, OS-Level IPC Implemen- tation in Complementary Multiprocessor Systems, Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on, April 2010.
18. Huang Bo, Dong Hui, Wang Dafang and Zhao Guifan, Basic Concepts on AUTOSAR Development, International Conference on Intelligent Computation Technology and Automation, May 2010.
19. Jing Chen, Chung-Ping Young, Da-Wei Chang, Guan-Ying Huang, Chung-Yuan Ke, Shih-Tun Yen and Tsang-Shuo Kuo, Multi-Kernel Embedded System on PAC Multi-Core Platform, Quality Software (QSIC), 2010 10th International Conference on, July 2010.
20. Nicolas Navet, Aurlien Monot, Bernard Bavoux and Franoise Simonot Lion, Multi-source and multicore automotive ECUs OS protection mechanisms and scheduling, Industrial Electronics (ISIE), 2010 IEEE International Symposium on, July 2010.
21. Jorn Schneider and Christian Eltges, Towards an Evaluation Infrastructure for Automotive Multicore Real-Time Operating Systems, 4th International Conference on Leveraging Applications of Formal Methods, Verification and Validation, Heraclion, Crete, Special Track on Resource and Timing Analysis, 2010.
22. Aamir Shafi and Jawad Manzoor, Towards Efficient Shared Memory Communications in MPJ Express, Parallel and Distributed Processing, IPDPS 2009, IEEE International Symposium on, May 2009.
23. Ronald Goodman and Scott Black, Design Challenges for Realization of the Advantages of Embedded Multi-Core Processors, AUTOTESTCON 2008 IEEE, September 2008.
24. Lui Sha, Ragunathan Rajkumar and John P Lehoczky, Priority Inheritance Protocols: An Approach to Real Time Synchronization, IEEE Transactions on Computers, Volume 39, No. 9, September 1990.