

A Real-Time Control Operating System for Industrial Automation

MMS (Manufacturing Message Specification) is an international standard that concerns integration of shop-floor devices such as numerically controlled (NC) machine tools, robots, programmable logic controllers (PLCs), and control computer systems. MMS introduces the concept of a Virtual Manufacturing Device (VMD) providing a common view of all physical devices. Wide spread use of MMS in industrial plants depends heavily on real-time support. This paper presents an approach to the implementation of an MMS server for shop-floor devices that takes into account real-time requirements. Our MMS server has a dynamically reconfigurable software architecture of a Real-Time Control Operating System (RTCOS). It is based on Microware's OS-9 real-time multitasking operating system kernel. In our framework, multiple reconfigurable control sub-systems represent multiple Virtual Manufacturing Devices (VMD). The sub-systems can be developed independently of target driver devices. This paper focus on the software architecture of our Real-time Control Operating system MMS Server called RCOMS and discusses the advantages of our approach compared to other MMS implementations.

INTRODUCTION

MMS (Manufacturing Message Specification) is an OSI application layer protocol that enables remote applications (called clients) to control and supervise various heterogeneous industrial devices (called servers). MMS is a part of the Manufacturing Automation Protocol (MAP) project initiated by General Motors [1]. As physical devices may have different functionalities and capabilities, MMS provides a means for modeling them as abstract virtual objects in a way that hides their particular functionalities. The object model called a Virtual Manufacturing Device (VMD) lays between an application program and a device, and allows the functions and capabilities of a device to be mapped to objects defined in the MMS standard [2, 3].

Time is an important characteristics that should be taken into account in an industrial environment. The concern for real-time requirements is a key issue for wide adoption of MMS in industrial plants. Many authors have proposed real-time extensions to MMS [4, 5, 6, 7]. Most of these works propose a modification to the MMS protocol, the addition of new services and objects or an addition of new parameters to services. However, the MMS standard do not provide any hint of how to implement MMS services. This is left open to the designer of MMS servers. Implementation of MMS services is an interesting aspect but we do not address it in this paper. Instead, we focus on the integration of MMS servers in a real-time environment.

All existing implementations of MMS have been done on top of operating systems that are not dedicated to MMS applications. For example, there are MMS implementations on UNIX, Window NT, MS-DOS, or on real-time operating systems such as Vxworks or OS/2. However, the implementations do not support real-time execution of MMS services. In the case of operating systems such as Unix, they are not suitable for real-

time execution and they do not provide appropriate synchronization mechanisms. MMS implementations on real-time operating systems have been done in user space—in this case, MMS is viewed as an application program and does not benefit from real-time support.

There is a strong similarity between the MMS and an operating system. For example an in-depth comparison of classical semaphores and MMS semaphores is given in [8]. The question that we address in this paper is the following: if there is a strong similarity between MMS and an operating system, and if MMS is used for control shop-floor devices (such devices require an operating system anyway), then why should we implement MMS on top of an operating system. The goal of this paper is to show that implementing MMS on an operating system is not a good solution for real-time control industrial processes. We propose an approach to the implementation of MMS services that allows their real-time execution.

Many operating systems were designed to support the development of software for robotics and automation systems. Chimera Real-Time Operating System has been developed by the Advanced Manipulators Laboratory at the Carnegie Mellon University [9]. Harmony has been developed at the National Research Council of Canada (NRCC) Laboratories to control a flexible system for real-time control of robotics experiments and for the development of experimental robot controllers [10].

We present a specification of the Real-time Control Operating system MMS Server (RCOMS). RCOMS is an open operating system environment independent of a control system hardware. It is based on the Microware's OS-9 real-time operating system. The difference between our system and the others is that our system guarantee a real-time execution of MMS services and interoperability among heterogeneous

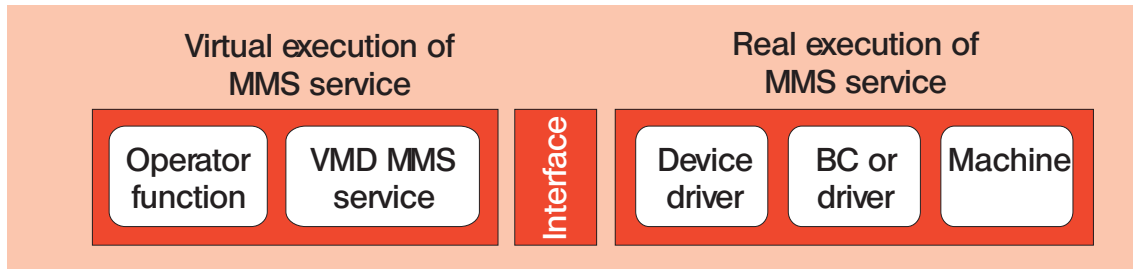


Figure 1. Real and Virtual Executions of MMS Services

industrial devices. In the rest of the paper we show how RCOMS fulfills these requirements.

The paper is structured as follows: Section 2 presents an overview of RCOMS. Section 3 gives reasons for the choice of OS-9 as a kernel for our target system. Section 4 provides a detailed description of the architecture and the implementation of VMD as an OS-9 module. Finally, Section 5 gives first conclusions and outlines the future work.

RCOMS: REAL-TIME CONTROL OPERATING SYSTEM MMS SERVER

Introduction

Recently, many users have become oriented towards an idea of creating their own system software or tailoring the available software to meet their perceived unique needs [11]. This orientation was induced by non-capability of existing operating systems to support the quality of services imposed by some specific applications.

The problem of time in industrial control system environments is crucial. Missing deadlines of critical servo-level periodic tasks results in losing data or missing control cycles. This can lead to some loss of efficiency in the best case, or can cause serious damage to equipment or human injury in the worst case. That is the reason for many works interested in developing an efficient operating system that can be used for control critical systems.

Our work belongs to this area of research. The development of a new operating system requires significant investment in time and resources. The total development time can be reduced by using an existing appropriate operating system kernel. Most programmers tend to think of a Real-Time Operating System, RTOS, instead of a real-time multi-task kernel. A RTOS is typically large, complex, slow, and expensive. One part of a set of generic services that can be used for developing an RTOS. In Section 3, we discuss in more details the choice of an operating system kernel for our needs.

2.2 Is MMS an Operating System?

An operating system separates users from hardware and software resources by means of virtual devices. Such a device makes writing and execution of an application easier.

MMS has a similar view of an operating system. However, we cannot consider MMS to be an operating systems even if it has the same functionalities as an operating system (see Table I), because in the case of MMS we distinguish two types of service execution: a virtual execution and a real execution (see Figure 1) The virtual execution depends dynamically on the real execution that must be synchronized with the virtual one. This feature does not exist in general purpose operating systems. The analogy presented in Table I will be used later for the specification of our Real-time Control Operating system MMS Server (RCOMS).

Ideally, the virtual execution of real-time control appli-

Operating System	MMS
Operating System Management Functions	VMD Management Services
Memory Management Functions	Domain Management Services
Access Management Functions	Variable Management Services
Process Management Functions	Program Invocation Management Services
Journal Management Functions	Journal Management Services
Event Management Functions	Event Management Services
Semaphore Management Functions	Semaphore Management Services
Input/Output Management Functions	Operator Station Management Services
File Management Functions	File Management Services

Table I. Analogy between MMS and an operating system

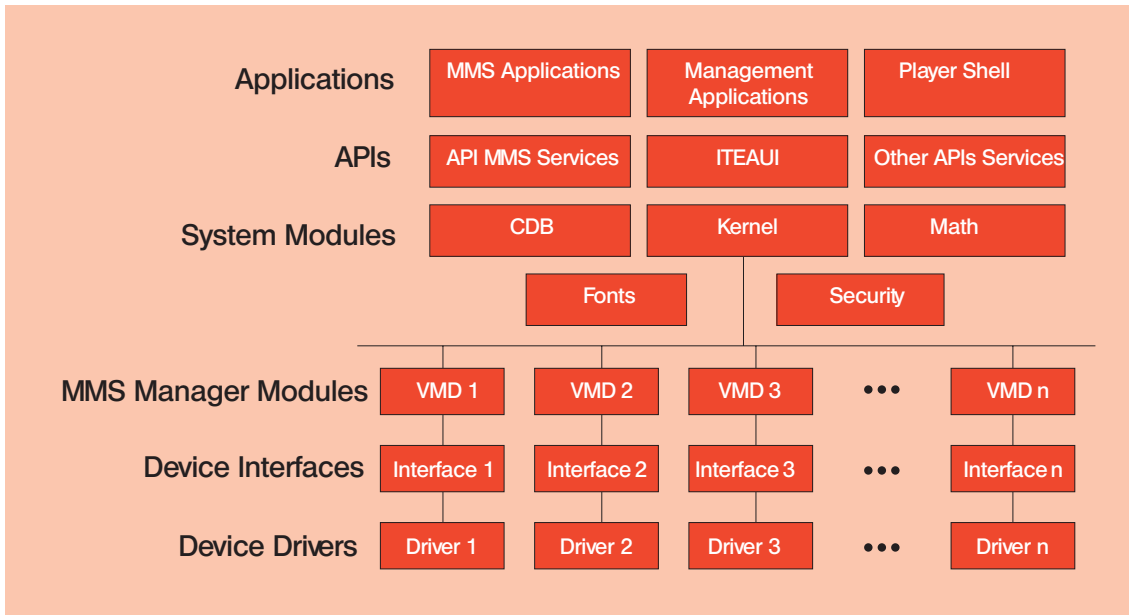


Figure 2. RCOMS System Software Architecture

cations must be synchronized the real execution. Our objective is to implement MMS such that the virtual and the real executions are well synchronized.

Many implementations of MMS were proposed using real-time operating systems or real-time programming languages. Also, some authors showed how to use some algorithms such as the Earliest Deadline First (EDF) [12] in the MMS context in order to obtain real-time execution of MMS services [13]. Following this direction, we propose a new method for the same goal which is the integration of MMS in a real-time environment. The idea is to implement MMS services as a sub-system inside of an operating system kernel.

Overview of RCOMS

There are two major problems that must be taken into account when implementing control systems with tight time constraints: the systems require time-critical communications and real-time supervisors. In this paper, we focus on the second problem.

Based on the observation that MMS and an operating system have many common functions (see Table I), we decided to implement an VMD (Virtual Manufacturing Device) as a sub-system server inside of the OS-9 kernel. It is a right choice in our opinion, because we can take advantage of the kernel support for real-time execution. Hence, we propose the following software architecture of the RCOMS system (see Figure 2).

1. **Applications:** The Player Shell is the first process executed on a RCOMS system. It is specific to each MMS server. It gives all the information related to the controlled machine, the number and the name of implemented MMS services.
2. **Application Programming Interfaces:** To improve portability of applications across a wide variety of system architectures, RCOMS provides a set of Application Program Interfaces (APIs). The MMS Service API provides all MMS services that are implemented by RCOMS. The ITEAUI (Integrated Telecommunications Environment Application User

Interface) is an open communication API for supporting network-specific protocols and initial applications.

3. **Systems Modules:** The OS-9 Core system block includes modules for Floating Point Mathematics support, System Security, Graphics Text Fonts, and a Configuration Description Block (CDB) used by application programs to determine the basic system configuration of MMS management modules.
4. **MMS Management Modules:** The modules form an important part of the RCOMS software. They implement the MMS services that are defined in each VMD. Section 4 provides more explanation about their implementation.
5. **Device Interfaces:** The interface (see Fig. 1) between a MMS server (VMD) and the device is left open by the MMS standard. SO, the designer of a MMS server must define this interface.
6. **Device Drivers:** A Device Driver is an interface of manufacturing device with the external world. It supports communication between the virtual manufacturing device (VMD) and the real manufacturing device.

CHOICE OF AN OPERATING SYSTEM KERNEL

Ideally, an RTOS (Real-Time Operating System) should provide a set of robust system services and a flexible task scheduling system without adversely affecting performance. RTOS should be efficient and easily configurable while offering a feature-rich development environment [14].

We have chosen OS-9 as a best suited operating system for implementation of RCOMS, because it has many characteristics of an ideal RTOS. In addition to that, OS-9 is widely used in industrial environments, so that our software system will be easily implemented in current environments without much additional work. In that way, integrating MMS with a real-time kernel

APPLICATION

results in a system software that have two interesting characteristics: real-time support provided by the OS-9 kernel and interoperability provided by MMS.

OS-9 is a real-time, multiuser, multitasking operating system developed by Microware Systems Corporation [15]. Since its introduction in 1983, OS-9's scalability and modular architecture has been adapted to a wide variety of applications ranging from medical instrumentation and avionics, to robotics, telecommunications and industrial process control. It allows communication between processes in the form of named and unnamed pipes, as well as shared memory in the form of data modules. Its architecture is modular allowing new devices to be added to the system simply by writing new device drivers. This fits perfectly well the architecture of a MMS server based on the flexibility of adding and modifying new virtual devices. OS-9 is available for embedded PowerPC systems and for any 680x0 based hardware platform.

A similar approach to the ours has been taken by Microware Systems Corporation, an ISO 9001-certified company. The company has developed DAVID (Digital Audio/Video Interactive Decoder), an open operating system environment for digital consumer devices such as interactive television decoders [16]. It can also be used in telephone, cable TV and wireless networks. DAVID is based on OS-9 kernel.

VIRTUAL MANUFACTURING DEVICE (VMD) AS OS9 MODULE

The OS-9 operating system is a collection of system modules that includes a high-speed kernel, robust I/O manager, dedicated file managers, and interrupt-driven device drivers. All modules are independent and dynamic allowing high flexibility and easy configuration. This architecture promotes structured programming techniques to make system integration quick and manageable. Its micro-kernel core and host of independent I/O file managers support a broad spectrum of industrial applications including process control, telecommunications, medical instrumentation and intelligent consumer electronics.

We will use the features of OS-9 to implement our MMS manager as set of OS-9 modules. Each MMS manager module corresponds to one virtual manufacturing device (VMD). Each VMD is an abstract object related to one physical device (see Figure 2). We present below an overview of the VMD architecture as an OS-9 module.

Virtual Manufacturing Device (VMD)

The Virtual Manufacturing Device (VMD) is an abstract object that represents the behavior of a physical device. A VMD is an abstract representation of common characteristics of all physical manufacturing devices and represents the externally visible behavior relevant to the MMS (see Figure 1). All standardized services refer to a virtual device that is mapped through implementation to real functions. The MMS describes only the effects of services on a VMD and does not dictate a specific implementation or mapping to real functions.

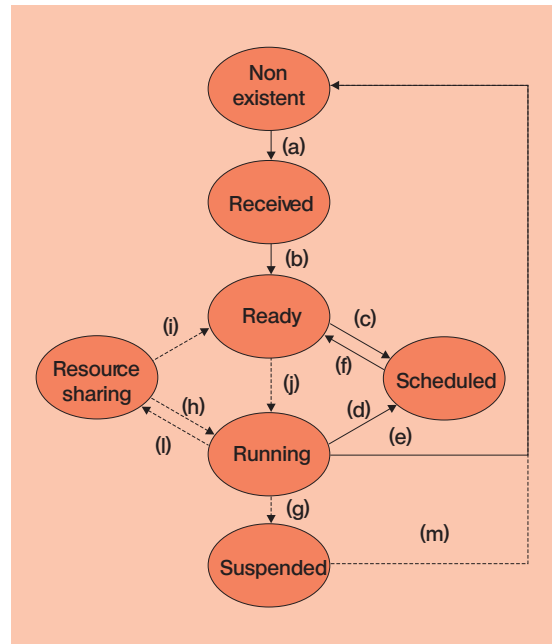


Figure 3. Execution model of MMS services

Most MMS services are confirmed services. When a VMD receives an Indication Primitive for one of confirmed services, a Transaction Object is created that governs processing of the service. We do not want to specify here all the services defined by MMS standard, but we present an execution model of the services and a general architecture of an VMD that are not specified by the MMS standard.

Execution Model of MMS Services

MMS associates a Transaction Object with each service request. The Transaction Object governs processing of MMS services [17]. When a service request is received by the server, the Transaction Object can be in one of the following states (see Figure 3):

- **Non-Existent:** The service request has not been received yet.
- **Received:** The service request has been received by the server but the corresponding transaction object has not been created.
- **Ready:** In this state, the Transaction Object is created and it can be processed.
- **Scheduled:** In this state, the Transaction Object of a service request is in competition with other Transaction Objects.
- **Running:** The service is being executed by the server.
- **Suspended:** The execution is interrupted due to internal or external reasons. An internal reason can be a disk access or overloaded processor. An external reason can be a device access.
- **Resource Sharing:** During this state, the Transaction Object reserves resources necessary to execute the request service.

In our model, we divide the real-time MMS services into two classes:

1. **Suspendable Services.** A service is suspendable if its execution may be postponed because some

conditions are not satisfied. The conditions depend on the response of a corresponding device or on resource sharing.

2. **Non-Suspendable Services.** The services are independent of the response of a device. Their execution is immediate because some resources have been reserved in advance.

For the first class, the execution cycle in the state model presented in in Figure 3 is: (a)(b)(c)(d)(g)(h)(i)(j)(e). The execution cycle for the second class is: (a)(b)(c)(d)(e).

VMD Implementation

Each VMD is implemented as an OS-9 module (has a modular architecture allowing new devices to be added to the system simply by writing new device drivers, or if a similar device already exists, by simply creating a new device descriptor). All I/O devices can be viewed as files, which unifies the I/O system. The modularity of OS-9 allows easy implementation of an VMD that presents a virtual abstraction of manufacturing devices. The features justify the choice of OS-9 operating system environment for development of our Real-time Control Operating system MMS Server (RCOMS).

In general, an operating system is viewed as a stack of superposed layers. RCOMS is composed of five layers (see Figure 4) presenting different levels of abstraction:

1. Control industrial device layer,
2. OS-9 kernel layer,
3. MMS services subsystem layer,
4. System interface layer,
5. MMS application layer.

The final architecture of RCOMS provides a base for development of real-time industrial applications. The development can be done by using OS-9 FasTrak. OS-9 FasTrak is a comprehensive software programming and management environment providing seamless interaction with OS-9 target systems such as RCOMS. FasTrak can be used as a highly integrated toolset that simplifies and automates conception, debugging, analyzing and managing RCOMS applications. FasTrak can be used by individual programmers as a stand-alone development platform or in a network-based distributed development environment as a group development platform. OS-9 FasTrak is available for popular UNIX workstations and PCs running Windows for development targeting RCOMS-based systems.

CONCLUSION AND FUTURE WORKS

Time is one of the most important characteristics that must be taken into account in an industrial environment. Interoperability among heterogeneous industrial devices with different functionalities and capabilities must be also considered. This paper has presented a new operating system environment for real-time control manufacturing devices. Our system supports real-time and interoperability. It uses the MMS (Manufacturing Message Specification) application layer protocol developed by ISO for interconnection and interworking of heterogeneous industrial devices. It is based on OS-9 operating system kernel that pro-

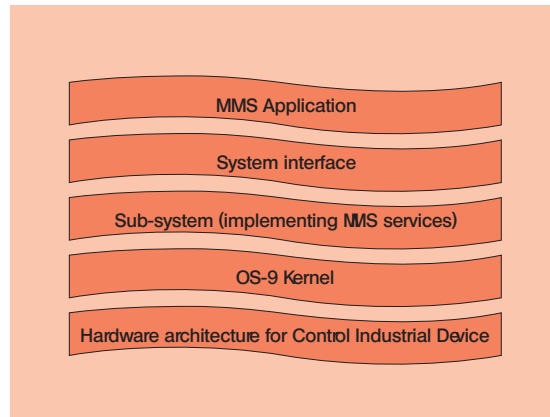


Figure 4. MMS Application development environment

vides a real-time execution of MMS services.

In this paper we have presented the general architecture of our system called a Real-time Control Operation system MMS Server (RCOMS). The architecture provides real-time scheduling of MMS services.

We believe that this new view of MMS servers is very important in the development of MMS real-time control applications. We have described the architecture of a MMS server and we have just began its implementation to validate our approach and evaluate performance gains in real operational conditions ■

Dr. Brahim Maaref received the degree diplome in Physics, Microelectronics from the University of Monastir (Tunisia), in 1993, the master's degree in Microelectronics from the University of Joseph Fourier, Grenoble (France), in 1995, and the Ph.D. degree in Microelectronics from the National Institute polytechnic of Grenoble (France) in 1999. His research interests include real-time communication systems for industrial automation. Dr. Mohamed Abid was born in Sfax (Tunisia). He received the "Ingenieur Principal" degree in Electrical Engineering from the National School of Engineering of Sfax (Tunisia) in 1986, and the Ph.D. degree in the area of Computer Science from the National Institute of Applied Sciences, Toulouse (France) in 1989. Dr. Abid is working as Associated Professor in the department of Electrical Engineering at National School of Engineering of Sfax (Tunisia). He is head of Electronic Systems Synthesis Group at Electronic and Micro-Electronic Laboratory of Sciences Faculty in Monastir (Tunisia). His current research interests include : hardware-software co-design and high-level synthesis. He has also been investigating for the design and implementation issues of embedded systems. He is co-author of about 50 publications in these areas.

REFERENCES

- [1] General Motors, Manufacturing Automation Protocol, Version 3.0, August 1988.
- [2] ISO/IEC. 9506-1, Industrial Automation Systems - Manufacturing Message Specification, Part 1: Service definition. International Standards

- Organization, 1990.
- [3] ISO/IEC. 9506-2, Industrial Automation Systems - Manufacturing Message Specification, Part 2: Protocol Specification. International Standards Organization, 1990.
 - [4] M.G. Rodd, G.F. Zhao, and I. Izikowitz, "RTMMS - An OSI-based Real-Time Messaging System," *Journal of Real-Time Systems*, November 1990, pp. 213-234.
 - [5] J. Akazan and Z. Mammeri, "Real-Time extensions to MMS," In J.-D. Decotignie, editor, *Proceedings of IEEE International Workshop on Factory Communication Systems*, Leysin, Switzerland, October 1995, pp. 193-200.
 - [6] J.-P. Elloy, P. Molinaro, and R. Ricordel, "A Temporal Extension to the MMS Protocol with Ker-Next Tool," In J.-D. Decotignie, editor, *Proceedings of IEEE International Workshop on Factory Communication Systems*, Leysin, Switzerland, October 1995, pp. 193-200.
 - [7] Z. Mammeri and J. Akazan, "An approach to make MMS real-time," In *Proceeding of the IEEE International Symposium on Industrial Electronics ISIE'95*, V. 2, Athens, Greece, pp. 586-591.
 - [8] P. Castori, "Semaphores revisited with MMS," *Operating System Review*, vol. 29, no. 3, July 1995, pp. 49-63.
 - [9] B. D. Stewart, D. E. Schmitz, and P. K. Khosla, "The Chimera II Real-Time Operating System for Advanced Sensor-Based Robotics Applications," *IEEE Transaction on systems, Man, and Cybernetics*, vol. 22, no. 6, November/December 1992, pp. 1282-1295.
 - [10] W. M. Gentleman, S. A. Mackay, D. A. Stewart, M. Wein, "An Introduction to the Harmony Real-Time Operating System," *Tech. report NRC No. 29288*, Summer 1988.
 - [11] S. H. Kaisler, "The Design of Operating Systems for Small Computer System," A Wiley- Interscience Publication, September 1982.
 - [12] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment," *Journal of the ACM*, vol. 20, no. 1, 1973, pp. 46-61.
 - [13] P. Castori, "Scheduling MMS service requests," (in French) *Proceedings of CFI'96, Ing[en]ierie des protocoles*, Rabat, Maroc, October 1996.
 - [14] J. Ready and D. Barnett, "Tradeoffs Driver Embedded OS Choice In Communications Designs", *Electronic Design Magazine*, May 1999.
 - [15] Microware Systems Corporation, *The OS-9 Catalog*.
 - [16] Eric Miller, "DAVID System Software for Interactive Digital Television Networks Version 2.0," White paper, Microware.
 - [17] M. Brill and U. Gramm, "MMS: MAP application services for the manufacturing industry," *Computer Network and ISDN Systems*, vol. 21, North-Holland 1991, pp. 357-380.