A Robot Controller Development of a Large-scale System for Shipbuilding

Soo-ho Kim*, Gyehyung Kang*, Juyi Park*, Gilwhoan Chu*, Jinwook Kim*, Jiyun Kim*, and Sung-Kwun Kim**

*Robot R&D Institute, Daewoo Shipbuilding and Marine Engineering Co., Ltd.,
2121Jeongwand-dong, Shiheung-si, Kyounggi-do, Korea.
(Tel : +82-31-432-5208; E-mail: {shkim2, ghkang, juyipark, chgh, antioch2, jonathan}@dsme.co.kr)
**Department of Mechatronics, Korea Polytechnic University, 2121Jeongwand-dong, Shiheung-si, Kyounggi-do, Korea,
(Tel : +82-31-4968-140; E-mail: skim@kpu.ac.kr)

Abstract: This paper present a robot controller developed for shipbuilding yard. Since shipbuilding process handles large work pieces and has dusty and noisy environment, the developed controller has separated architecture into main control part and servo control part. Main control part is located in control room while servo control part is located near robot with work pieces. Commutation between two parts is done through SynqNet and RS485. Air purging system is adapted to servo control part for better reliability. We aimed open architecture in both hardware and software architecture. For open hardware architecture, we employed Compact PCI (cPCI) because it is widely used bus system and very reliable. Since lots of commercial boards are available with cPCI interface, upgrade and reconfiguration is easy. For open software architecture, Windows XP® Embedded is selected as operating system (OS), because it is very popular OS and most hardware vender supports device driver for the windows XP.

Keywords: Shipbuilding, Robot controller, SynqNet, Open architecture, Windows XP Embedded

1. INTRODUCTION

Difficulty in shipbuilding process is mainly due to the variety of tasks, the large size of work piece and dusty and noisy environment. The variety of tasks requires very flexible robot programming and the large size of work piece requires the movement of working robot. The dusty and noisy environment, of course, degrades the reliability of robot controllers. Such restrictions make difficult to utilize conventional robot controllers in shipbuilding processes. Although there are lots of proven robot controllers from many vendors [1, 2], it is hard to adopt them into shipbuilding process because the controllers were developed based on in-factory process such as automobile or PCB production lines. Moreover, the vendors do not allow modifying the controllers according to our special purpose.

Therefore, we developed a controller which is suitable for shipbuilding process and reconfigurable for various tasks and robots. In developing the controller, we aimed open architecture in both hardware and software architecture. For open hardware architecture, we employed Compact PCI (cPCI) because it is widely used bus system and very reliable. Since lots of commercial boards are available with cPCI interface, upgrade and reconfiguration is easy. For open software architecture, Windows XP Embedded is selected as operating system (OS), because it is very popular OS and most hardware vender supports device driver for the windows XP. Since Window XP does not support real-time operation, we implemented pseudo-real time operation using real-time operation of MEI's motion control board and internal buffer of the board.

We separated the controller into two parts: main control part and servo control part. Main control part is to remain control room, where is relatively clean, while servo control part is located near robot. Two parts communicate each other through SynqNet that supports communication up to 100 meters. Since the servo control part is still in dusty and noisy environment with robot, air purging system is adopted for better reliability.

The following sections describe hardware and software architectures of DSME controller. Application strategy and simple demonstrations are shown in section 4. Section 5 summaries our presentation and discuss the further works.

2. HARDWARE

Hardware of DSME controller is composed of two parts: main control part and servo control part (Fig.1). This separated structure is employed because of the special working environment of shipbuilding yard. Generally, the size of work pieces is about $20m \times 20m \times 20m$ and the working environment is very dusty and noisy. The big size of work piece make a restriction that controller cannot be located on control room because the servo motor vendors strongly recommend limiting the cable length to 20m from motor to servo controller. This restriction, however, conflicts to requirements from workers in the yard; controller should be located in control room, where is less dusty and less noisy, for user's convenience and controller's reliability. Using separated structure, main control part is to remain control room while servo control part is located near robot. Two parts communicate each other through SynqNet [3] that supports communication up to 100 meters.

Main control part has compact PCI (cPCI) bus system, which is reliable in vibrant, noisy, and dusty environment. Since the cPCI is widely used bus system and lots of boards are available from commercial sites for cPCI, it's easy to add or replace hardware components according to requirements.

Basic components of the main control part are CPU board, motion control board, serial board and system and user I/O boards. The CPU board is from Kontron Co. and has an Intel Pentium 4 processor at 2.4GHz. The motion control board, which is from Motion Engineering Inc.(MEI)[4], has a DSP processor. The motion control board controls eight motors with sampling frequency 10 kHz and can be upgraded to control 16 motors. Serial board is used for RS485 communications between main control part and servo driver. Main controller reads the status of servo driver including

absolute encoder value and set parameters in servo driver through this serial board. System I/O board is dedicated to manage signals related on system inputs and outputs such as switches and LEDs on user panel. User I/O board is offered for users to utilize them according to various applications like welding process, painting process, grinding process and blasting process.

Fig. 1 DSME controller and robot: The box in left side is main control part and the box near robot is servo control part.

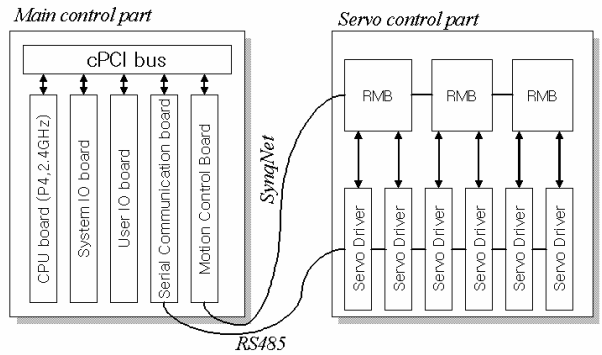

Fig 2 Main and servo control parts of DSME controller

Servo control part consists of remote motion block (RMB) for SynqNet communication and servo drivers for motor control. Main function of RMB is to transfers signal from main control part to servo drivers. Since the types and number of servo drivers vary depending on the robot type and the number of joints, the servo control part is constructed according to the robot to be controlled. On the other hand, main control part can be used without modification for various robots.

We equipped the servo control part is with a special cooling system because its components, servo drivers and power supplies, generate much heat. In the cooling system, compressed air is supplied through the inlet located bottom of the one side of controller (Fig. 3) and the air flows to the outlet in another upper side. Additionally, since the compressed air is filtered before supplying to the controller, it can isolate the controller from the dusty and noisy environment.

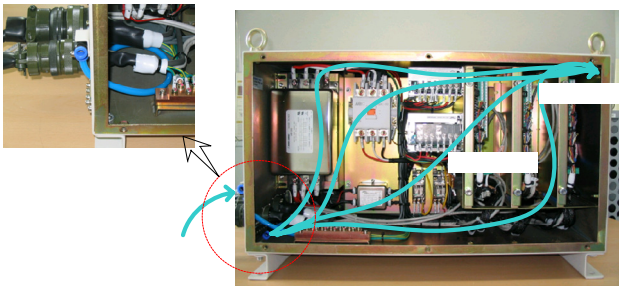

Fig. 3 The cooling system using compressed air

3. SOFTWARE

In software architecture, we aimed object oriented programming (OOP) as well as open architecture, so that one can easily add or modify functional modules according to user's requirements. Software has hierarchical architecture shown in Fig. 4, where each block represents a component with independent function. Since the components in device and control layers are supported by operating system and device drivers, we focused on development of the components in application layer.

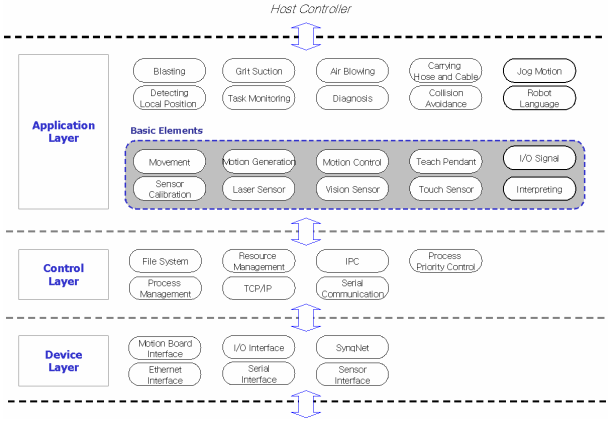

Fig. 4 Hierarchical software architecture of DSME robot controller

Selection of operating system (OS) was one of the most important issues. Examining many operating systems including real-time OS, we selected Windows XP Embedded (XPE) [5] because of its popularity. Because it is most widely used operating system, most hardware vendors support device drivers for it and most users are familiar with it. Plentiful device drivers give us many choices for auxiliary devices and hence it is easy to add or upgrade devices. Since most people are familiar with Windows XP, they may feel comfortable and learn quickly how to use the controller.

Since a shortcoming of using XPE is lack of real-time support, we utilized the real-time operation of the motion control board; the motion control board actually moves the robot in real-time with motion commands in motion buffer and a process on XPE updates the motion buffer with desired robot position. Although the process on XPE does not run in real-time, robot can be controlled reliably by motion control if the motion buffer is not empty.

In current implementation, the software of DSME controller consists of four main modules as shown in Fig. 5. Each

module communicates each other through Windows Messages and shared memory.

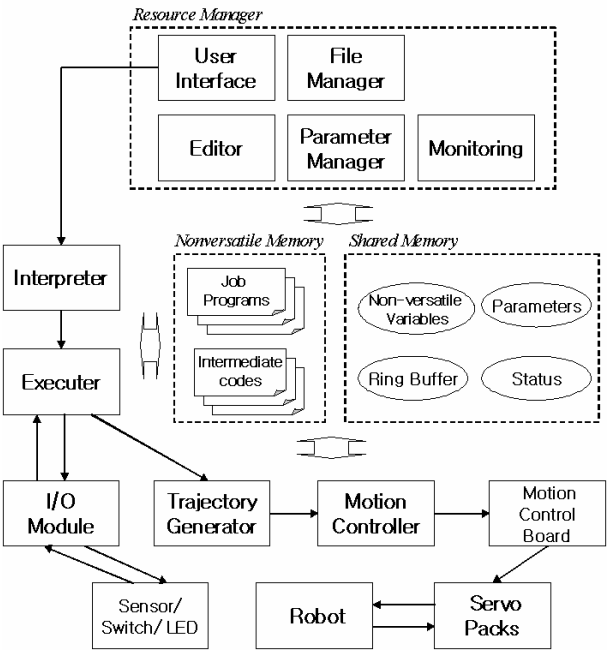

Fig. 5 Interfacing between software modules

Resource Manager has user interface, file manager, editor, parameter manager and monitor. User interface receives commands from user and distributes it to other modules. Fig. 6 shows current implementation of UI with editor, file manager and monitor. Editor is to edit job-program and file manager is to manage the job program and intermediate codes, which is generated by Interpreter. Monitor displays current robot status such as current joint angles, tool position and the status of I/O ports.

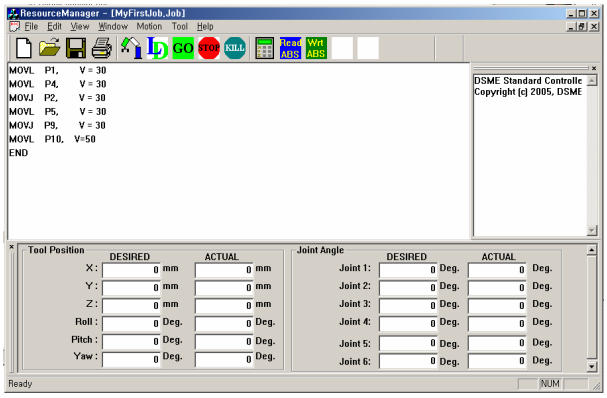

Fig. 6 User interface of DSME controller

Parameter Manager shown in Fig. 7 manages system parameters, so that users can easily configure DSME controller for given robots and machines. Parameters of DSME controller are arranged in a tree structure: a controller itself is the root of tree, robot in the next level, and joint and motors are added in following level. A controller can have up to 4 robots and each robot has up to 6 joints and each joint has maximum 4 motors. Total number of joint and motor cannot exceed the total number of axes supported by MEI motion control board, which is normally 8 and can be extended to 16.

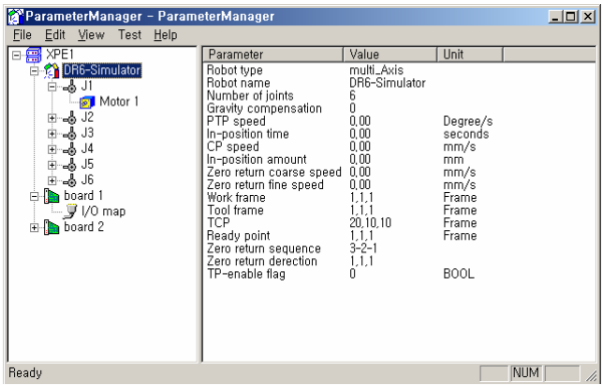

Fig. 7 Parameter Manager

Interpreter reads a job-program with ASCII text and translates it into intermediate code for robot executer. Robot executer then reads the code and actually performs the tasks described in the job-program. For job-programming, we developed our own language called RoboC based on ANSI C. RoboC is similar to Ansi C except on the following features, which are added for robot control.

- *Position variable type*: Position variable is to describe robot position in its joint angles and position of end effecter.
- *Non-volatile (NV) variable:* NV variables keep their value before programmer changes it even during power off. These variables, therefore, are stored in NV memory space like a hard disk drive. RoboC supports three types of NV variables: integer, float and position. Since it is convenient to fix the names of NV variables for maintenance and debugging purpose, their names are fixed to RI, RF and RP for the type integer, float and position, respectively.
- *Robot motion commands:* Robot motion commands are given as C function. For example MOVJ() is a function for point to point movement and MOVL() and MOVC() are for linear and circular movement, respectively.
- *Commands for I/O interface:* I/O commands are also given as C functions to interface with external device through I/O boards.

Since lots of robot users are familiar with BASIC type robot language, we support BASIC type language, DSRL. Translator converts DSRL to RoboC. Table 1 shows the expression in SDRL and RoboC for same tasks.

Table 1: Expressions with DSRL and RoboC for same task

DSRL	RoboC

10 INT I
20 MOVJ P1 V=50
30 MOVL P2 V=100
40 GOSUB FUNC1
50 MOVC P3 P4 V=I
60 END
70 SUB FUNC1
80 I= 30
90 ENDSUB
	int i;
	void main()
	{
	MOVJ(RP[1], 50);
	MOVL(RP[2], 100);
	func1();
	MOVC(RP[3],i);
	}
	void func1()
	{
	i=30;
	}

Trajectory Generator receives command from robot executer or user interface and generates trajectories, which are the list of joints angles for each sampling time. It reads robot parameters stored in shared memory for kinematics and inverse kinematics for trajectory generation. The generated joint angles are then sent to motion controller by writing them into a ring-buffer. Motion Controller read the ring-buffer every 30ms and interpolate joint angles to generate joint command angle for every 10ms, which are to be sent to motion buffer. These buffer structures are depicted in Fig. 8. Trajectory Generator pushes joint angles into ring buffer if is not full and Motion Controller then simply pop the ring buffer every 30ms. If the ring buffer is empty, Motion Controller asks motion control board to stop the robot.

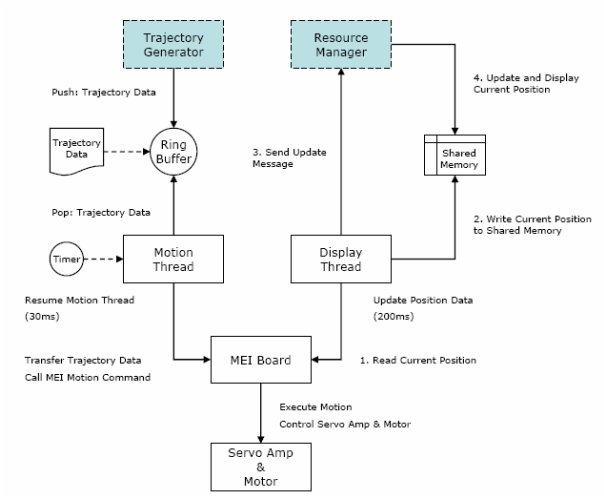

Fig.8 Buffer structure for the data transfer from Trajectory generator to motion control board

4. APPLICATION STRATEGY

Fig. 9 shows application strategy of the DSME controller for the blasting tasks [6, 7]. Since the blasting task blows steel grit in high density with highly compressed air, the environment is very dusty and noisy. Such environment degrades the reliability of most robot controllers that consists of highly sensitive electric devices. The figure shows that four servo control parts are located top of the blasting system and main control parts are located in control room that is far from the work pieces.

In order to perform tests for the blasting system, we first implemented a miniature of the blasting system with size of $4m \times 4m \times 4m$. The proposed DSME controller is then connected to the miniature with a 6 DOF robot and asked to move the miniature and robot based on a job-program that consists of point-to-point and continuous path motion commands with a sensor input command. This simple demonstration verified that the proposed architecture works fine, at least, for smaller systems.

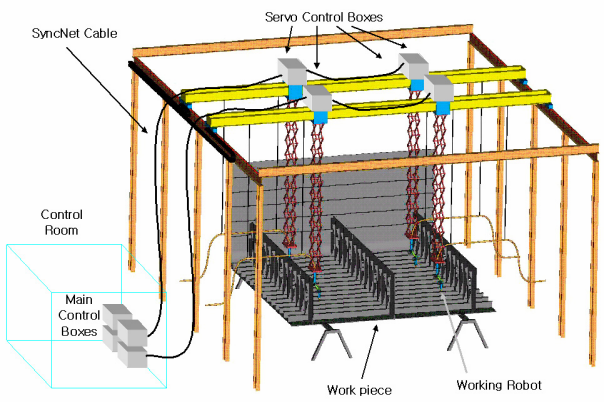

Fig. 9 Application strategy of DSME controller For blasting task

5. SUMMARY AND FURTHER WORKS

This paper presented a hardware and software architecture of a robot controller for large-scale system in shipbuilding yard. To isolate the robot controller from the noisy and dusty workplace, we proposed architecture that separate main controller from work cell. For flexible usage of controller for various applications, we aimed open architecture in hardware and software. Application strategy was explained to show how the controller can be used in shipbuilding yard and simple test verified that the proposed architecture works fine for a 6 DOF robot.

In the future, additional module in the following will be added to make the controller more useful in shipbuilding yard: Offline programming (OLP) module will imports CAD data and generate job-programs, so that teaching is omitted. Graphical simulation module will display robot movement on screen that user can validate job-programs without moving robot. Diagnostic module checks robot status and inform user through screen or speaker. It may automatically stop the robot for emergency situation. Database module manages various data such as parameters for various robots and job-programs for various tasks. It will decrease the user's effort of programming by increasing the reusability of robot configuration and job-program. In our opinion, the DSME controller with such modules will help to increase the productivity and reduce the accident in shipbuilding yard.

REFERENCES

[1] *UMAC specification,* Delta Tau Data Systems, Inc., http://www.deltatau.com/Common/products/

[2] *Adept Motion Controls*, Adept Technology, Inc., http://www.adept.com/main/products/controls

[3] *SynqNet User group*, http://www.synqnet.org/

[4] *XMP-SynqNet-CPCI Hardware Specification*, Motion Engineering Inc., http://www.motioneng.com/pdf/xmp_cpci.pdf

[5] Ridnour, K. Moozov, *Building a Reliable Windows XP Embedded Platform*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxpembed/html/xpplatform.asp

[6] *Abrasive Blasting*, Emission Factor Documentation For AP-42, Fifth Edition, Midwest Research Institute, Cary, NC, Section 13.2.6, January 1995.

[7] A. W. Mallory, "Guidelines For Centrifugal Blast Cleaning", *J. Protective Coatings And Linings,* Vol. 1, No. 1, June 1984.