Reconfigurable Mechanisms for Application Control (RMAC): Applications

Wei Li[1], Edward Red[2], Greg Jensen[3] and Mark Evans[4]

[1]General Motors, China, weili109@yahoo.com
[2]Brigham Young University, USA, ered@et.byu.edu
[3]Brigham Young University, USA, cjensen@byu.edu
[4]Direct Controls, Inc., USA, evans@directcontrols.net

ABSTRACT

Built upon the direct control architecture, the Reconfigurable Mechanisms for Application Control (RMAC) architectural framework is described in reference [1]. This companion paper describes several applications that are of historical significance to the RMAC architecture and also shows the flexibility and simplicity of direct control reconfiguration. We first present two direct control applications that have encouraged RMAC development. The first application describes the reconfiguration of a large 3-axis mill into a Coordinate Measuring Machine (CMM), where the popular PC-DMIS CMM planning software uses a device driver interface to plan and control inspection processes on the mill directly, without process plan conversion to typical machine control languages. Second, a robot is converted into a rapid prototyping commercial application, using direct control of the robot from popular CAD/CAM systems. RMAC flexibility is then demonstrated on a 3-axis prototype mill by reconfiguring the mill as either a 3-axis sculpting mill or as a 2-axis router. The RMAC software and interface components are considered in detail.

Keywords: CAD/CAM applications, direct control, reconfigurable machine control

1. APPLICATION METHODOLOGY

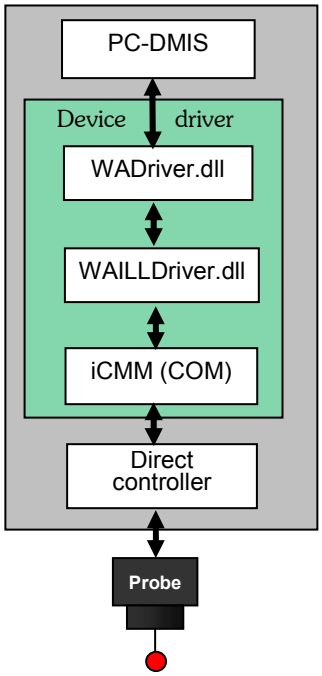
Prior to the RMAC (Reconfigurable Mechanisms for Application Control) architectural development, direct control had been demonstrated on both machine tools and robots, under the program acronym DMAC, and using applications in Unigraphics, CATIA, Alias Wavefront, GibbsCAM, and PC-DMIS [2-5] to directly control various mechanisms. Unlike conventional NC control methods that use APT, CL, and M&G files to bridge CAD/CAM applications to machine controllers, direct control connects modeling and process control applications directly to a machine. In these previous applications, communication occurred through a software layer that acts like a device driver, but configured specifically for each application machine, without architectural standardization or flexibility.

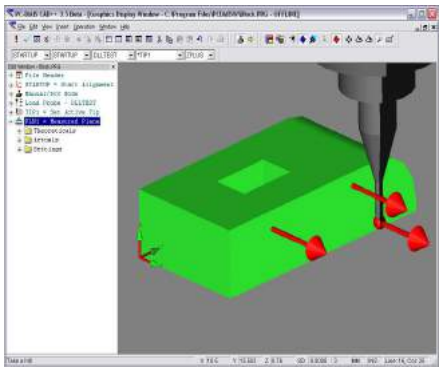
RMAC standardizes a direct machine interface where a single machine can be reconfigured according to application needs. First, we consider two applications that demonstrate how mechanisms can be reconfigured to act differently than their original application/design purpose. From these applications we generalize our direct control architectures to standardize the RMAC reconfigurable control concept. The final application serves to demonstrate some of the pertinent architectural details, by developing two different device drivers where a 3-axis mill is reconfigured for surface sculpting and then again reconfigured to act as a 2-axis router or planer, without loss of generality.

2. DIRECT CONTROL – COORDINATE MEASURING MACHINE

We first consider a direct control application that transforms the process control characteristics of a 3-axis Sugino machine tool into a Coordinate Measuring Machine (CMM). CMM's are widely used in ***off-line*** dimensional inspection of completed parts. In this application, PC-DMIS, a popular software application for planning CMM inspection processes, is actually used to control the mill as an ***in-process*** CMM. This is quite different from some modern machine tool controllers that switch control to a secondary PLC and changeable touch probe to perform measurement tasks.

Direct control ***reconfigures*** the controller software to add the capabilities of modern CMM's and, through a device driver, uses PC-DMIS to inspect parts in-process. Reference [4] justifies use of a machine tool as a CMM because tool loads and machine compliance are non-issues when inspecting parts. Fig. 1 shows the implementation where Wilcox and Associates wrote a generic driver library (WADriver.dll, WAILLDriver.dll) that passed commands and received operational feedback directly to/from the 3-axis mill through the iCMM COM interface to the direct controller.





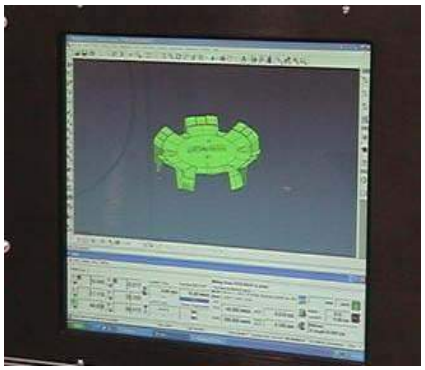


Fig. 1: CMM device driver. Fig. 2: PC-DMIS (top) and part inspection. Fig. 3: TruPath™ Panel & Denso Robot.

Results – Fig. 2 top shows a sample PC-DMIS interface screen, while the bottom shows actual direct control inspection of a part on the 3-axis Sugino mill. The inspection of circle diameter and part lengths were conducted on the mill and then on a Brown and Sharpe CMM. Although the Sugino mill was not calibrated to compensate for backlash and ball screw lead error, tests showed the part dimensions measured on the mill varied about 0.02 mm from those measured on a Brown and Sharpe CMM.

3. DIRECT CONTROL – ROBOTIC PROTOTYPING

We next consider an industrial robot that has been reconfigured for direct control, allowing the robot to be programmed using CAM-generated tool paths. Industrial robots are typically used for highly repetitive tasks that require end-point repeatability and fast cycle times where the actual robot path between end-points is not of primary concern. Reconfiguring the robot allows the robot to be used for many additional applications where accuracy of the robot path is of primary concern. It allows a robot to easily be programmed to follow complex paths defined as a function of part geometry as defined by a CAD system.

Feature	TruPath™ Controller	Standard Robot Controller
Servo rate (hz)	4,000+	80 – 100
Singularity avoidance	Passes through singularities without leaving the programmed path	Requires additional points causing the robot to leave the path
Look-ahead	Dynamic look-ahead dependent on path speed – stays on path	Limited to 5 or fewer points – may leave path due to motor limitations
Path following	NURBS, lines, arcs, points	Point-to-point
Platform	PC	Various – mostly proprietary

Tab. 1: TruPath™ / Standard Robot Controller Comparison.

Direct control **reconfigures** the controller to apply CNC-type motion planning and servo control to a robot. For the first time a robot is controlled with advanced look-ahead capabilities, high bandwidth servo control, and the ability to be driven by advanced geometric entities such as NURBS paths and surfaces. Tab. 1 shows a comparison between the DMAC robotic controller developed by Direct Controls, Inc. (trademarked as a TruPath™ controller) and standard industrial robot controllers.

Results – Fig. 3 top shows the TruPath™ control panel interfaced with a Unigraphics NX3 process plan showing tool paths. The bottom of Fig. 3 shows a **reconfigured** Denso robot fitted with a milling spindle head cutting a prototype of the CAD part in modeling clay. The surface quality of the finished part was equivalent to that obtained by a CNC milling machine, and the robot was controlled to follow the process plan directly from within NX3 without any file conversions or post processing required. Programming such a path using a standard robot controller would be prohibitive due to the large number of robot positions that would need to be taught.

4. RMAC CAD/CAM APPLICATIONS

We now describe the RMAC software system, which consists of a series of software components that allow application software to directly control various mechanisms through relevant device drivers. The initial prototype, designed to be quite comprehensive, realizes the "art-to-part" goal by making control reconfiguration a transparent process to the user. This section describes the software components introduced in [1], including the relevant interfaces that allow for bi-directional data flow between CAD/CAM applications and the machines they are directly controlling.

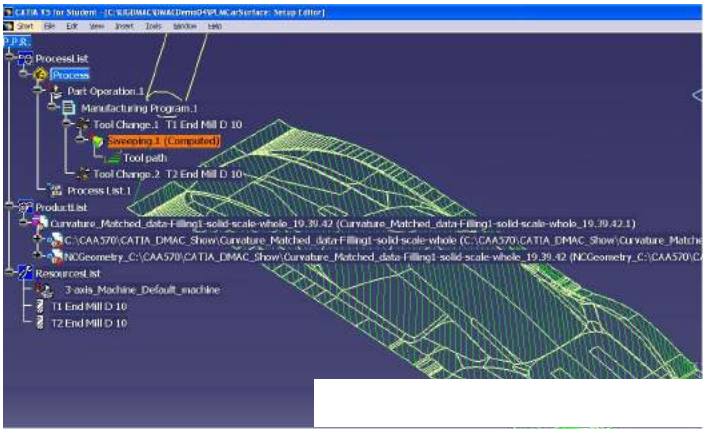


Fig. 4: CATIA process plan.

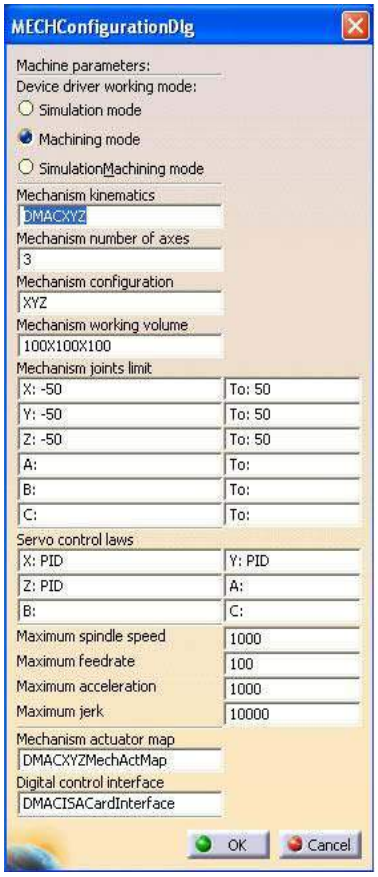


Fig. 5: Machine characteristics dialog.

4.1 CAD/CAM

Fig. 4 shows a Ford GT top surface model with CATIA tool paths and process plan information. What makes the RMAC architecture different is that an RMAC device driver interprets this information as a ***control process***, not as just a process plan. Under the RMAC architecture one or more device drivers are developed for each individual machine. Whenever a CAD/CAM user generates a process plan, they will first select a machine to perform the process. RMAC will load, with user input, a relevant device driver so the process plan, now a control process, can directly control the machine through the device driver. Post-processing of tool paths into M&G codes is completely eliminated. If the machine is capable of several configurations, through multiple device drivers, then a single machine might be used for different applications.

Because several machines may be capable of executing the process plan, a configuration dialog box (not shown) is designed as a plug-in user interface to CATIA to assist users in selecting the best machine tool. The *Device Driver Manager* described in [1] has a built-in search engine to assist users in their machine selection, based on various machine filter schemes. If the user needs more information, the user can click the machine characteristics button to open a new dialog box - see Fig. 5. This box contains more detailed information about the selected machine, such as machine configuration, working volume, machine limits, maximum feedrate, maximum spindle speed, etc. Applications can access the device driver manager to obtain the machine characteristics information.

4.2 Device Driver Manager

The *Device Driver Manager* is an independent dynamic linked library (DLL) that runs independent of the CAD/CAM application software and that:

- Maintains a device driver database relevant to a collection of machines and their driver DLL's.
- Provides a built-in database search engine to assist the user to narrow down their machine selection based on various search schemes.
- Provides interface API's to communicate with CAD/CAM applications.

Device driver database - The *Device Driver Manager* maintains the *device driver database*. Tab. 2 displays sample database contents. This database allows CAD/CAM users to inquire about a mechanism and its device driver to assist their machine tool selection.

Mech. name	Device driver	Mech. type	No. joints	Working volume (mm^3)	Spindle hp	Max feed rate(mm/s)
DMACXYZ	DMACXYZ.dll	Mill	3	100X100X100	2.5	300
TarusXYZCA	TarusXYZCA.dll	Mill	5	800X800X800	10	500
SuginoXYZ	SuginoXYZ.dll	Mill	3	100X100X100	5	400
SuginoXYZAC	SuginoXYZAC.dll	Mill	5	100X100X100	5	400
CMM	CMM.dll	CMM	3	100X100X100	5	400

Tab. 2: Device driver database.

Structured query language (SQL) statements are used to query the database's machine characteristics upon a user's request. For instance, if a CAD/CAM user desires a 5-axis mill with a working volume greater than 150x150x150 mm, and a mechanism spindle greater than 5 hp, one SQL query in the database will narrow their selection to a single machine in the Tab. 1 example: TarusXYZCA. This search and selection process reduces the need for CAD/CAM users to review every available machine before finding one capable of performing the process plan.

Device driver object - To retrieve and store mechanism and device driver related information from the device driver database, a *device driver object* data structure is defined (partial structure for illustration):

```
typedef struct _DEVICE_DRIVER_OBJECT{
char        MechanismName[100];
char        DeviceDriver[100];
…
char        DeviceDriverVersion[100];
}DEVICE_DRIVER_OBJECT, *PDEVICE_DRIVER_OBJECT;
```

The *Device Driver Manager* uses the *device driver object* data structure to represent each device driver. The mechanism and device driver related information is retrieved from the device driver database and then stored in each field corresponding to one relevant column defined in the device driver database.

Interface API's - The *Device Driver Manager,* designed as a stand-alone DLL, must expose interface API's to enable communication with CAD/CAM applications. These interface APIs are separated into two groups: functions that allow CAD/CAM applications to access the device driver database, and functions that return the selected machine information back to the application.
CAD/CAM applications call the first group of interface API's to operate on a device driver database. The sequence for operating this database is: (1) open the database; (2) use SQL statements to query the database; (3) return the query results; and (4) close the database.
CAD/CAM applications call the second group of interface API's to obtain information related to a selected machine. This machine information will then be displayed to CAD/CAM users, as described in the CAD/CAM section, for proper machine evaluation and selection.

4.3 Device Driver

The device driver is a collection of methods and classes that can be called to perform various operations on the selected mechanism, and also feed back operational parameters, such as feedrate, spindle speed, joint/axis value, current/torque, etc. Each device driver determines a particular mechanism's behavior. Specifically, the device driver is designed to:

- Apply a self-contained device database to expose the details of a configured mechanism.
- Expose functions required by the CAD/CAM application.
- Communicate directly with the RMAC reconfigurable controller.

The device driver is designed as a DLL and thus can be run-time loaded by CAD/CAM applications. For each driver DLL, machine-specific software modules can be designed, linked, and debugged independently. These DLL's are completely independent executable files. As a result, if a mechanism's device driver functionality needs to be updated or enhanced, driver developers simply update the device driver DLL.

Mech name	Kinematics	No. joints	Max feed rate (mm/s)	Spindle max speed (rpm)	Joints	Joint min limit (mm)	Joint max limit (mm)	Joint max speed (mm/s)	Joint max accel (mm/s^2)	Servo control law	Kp	Ki	Kd
DMACXYZ	DMACXYZ	3	300	1000	X	-50	50	200	1000	PID	.95	0	.0067
					Y	-50	50	200	1000	PID	.95	0	.006
					Z	-50	50	200	1000	PID	1	0	.0075

Tab 3: Device database.

Device database - Each device driver has an integrated database, shown in Tab. 3. This allows CAD/CAM users to easily access any machine information prior to selecting a particular machine for a manufacturing process. The database parameters also allow the device driver to correctly configure the RMAC controller.
The device database in Tab. 3 contains three categories of information relevant to a mechanism device: 1) primary machine configuration characteristics, such as the number of axes, etc; 2) machine operational parameters, such as maximum feed rate, spindle maximum RPM, etc; 3) motion planning and servo control capabilities, such as kinematics classification, servo control law, and servo gains used on each axis, etc.
Device object - A *device object* data structure assists the device driver software to better manage a selected mechanism. This data structure serves two purposes. First, it assists CAD/CAM applications to easily access machine information contained within the device database. Second, it contains the exact information necessary for the device driver to set up a mechanism's operational parameters and reconfigure the RMAC controller for the intended machine and application.

```
typedef struct _DEVICE_OBJECT{
        char    MechanismName[100];
        int     NumOfJoints;
        int     MechanismJointType[DMAC_MAX_JNT];
        …
        char    MechanismSpindleServoControlLaw[100];

}DEVICE_OBJECT, *PDEVICE_OBJECT;
```

Interface to CAD/CAM - The interface API's are divided into three groups: 1) functions that allow CAD/CAM applications to access the device database, 2) functions that return specific machine information back to CAD/CAM applications, and 3) functions that instruct the RMAC open-architecture controller to set up correct operational parameters and configure the motion planning and servo control algorithms that are specific to this mechanism. The first two groups of API's are similar to the device driver manager's interface API's described earlier.

Interface to RMAC reconfigurable controller - The device driver software uses two COM interfaces, *RMAC_Config* and *RMAC_CAM*, to communicate with the RMAC controller. It uses the *RMAC_Config* interface API's to instruct the controller to correctly set up machine parameters and to map mechanism-specific motion planning and servo controlling algorithms into its controller software. Once the device driver software finishes reconfiguring the RMAC controller, it then uses the *RMAC_CAM* interface API's to pass process instructions to the RMAC controller and return machining feedback information. The device driver software contains an instance of the *RMAC_Config* and *RMAC_CAM* objects.

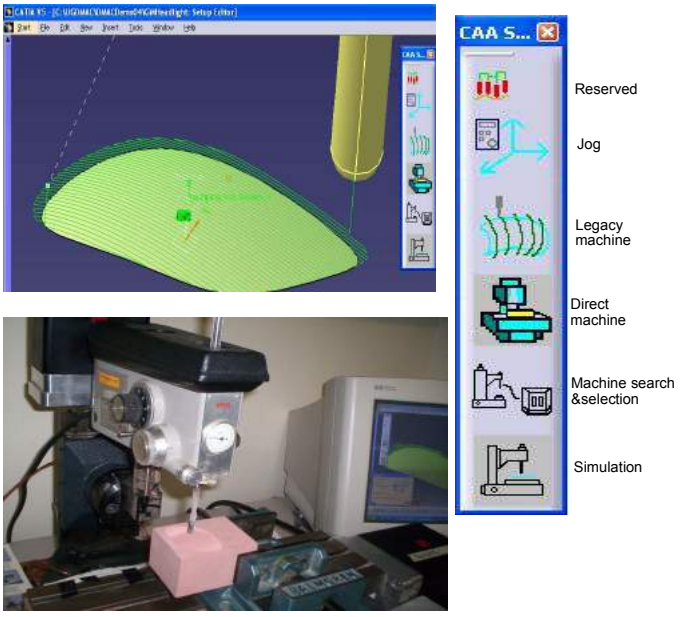


Fig. 6: Reconfigurable machining of headlight surface.

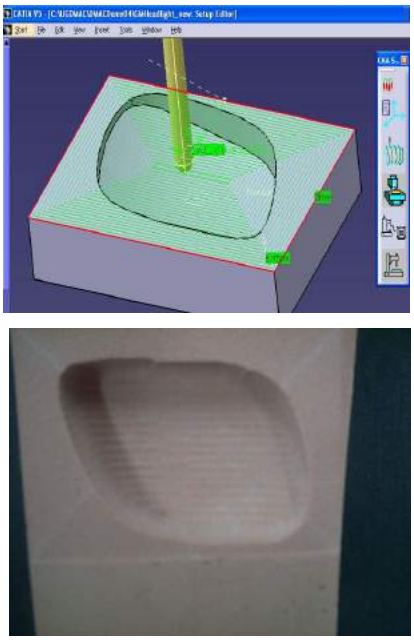


Fig. 7: Headlight XY process and machined part.

4.4 RMAC Reconfigurable Controller

The RMAC controller uses a configuration system to dynamically map motion planning and servo controlling algorithms from separate DLL modules based on any specific machine configuration and application demands. As shown in [1], all machine-specific hardware controlling, motion planning, and servo controlling algorithms are developed as separate DLL modules. The motion planner and servo controller only contain generic control codes at machine startup. After CAD/CAM users select a particular machine, the configuration system will start importing all necessary motion planning and servo controlling codes into the RMAC controller.

The configuration system receives the configuration instructions directly from the *RMAC_Config* interface. Based on these different configuration instructions, the configuration system will conduct one of two operations. It will either set up a correct machine operational parameter, such as machine joint limits, or load the corresponding DLL's, then map mechanism-specific modules, such as the machine kinematics object, into the RMAC controller to replace the generic control codes that are set up at machine startup.

After the configuration system finishes these configuration processes, the RMAC controller is dynamically reconfigured for a particular mechanism. CAD/CAM applications can then pass the manufacturing process instructions, through the *RMAC_CAM* interface, to the RMAC reconfigured controller for direct machining.

RMAC's architectural simplicity does not fully explain the implementation details needed to make direct control work in the areas of servo-control, Cartesian and joint space motion planning, kinematics, and trajectory generation. References [6] – [10] are just a few of the prior publications that describe this development.

5. VERIFICATION TESTS

To demonstrate the RMAC architecture, two device drivers were independently developed to connect a 3-axis prototype mill in Fig. 6 directly with CATIA. Both the CAD/CAM application and the RMAC controller run on a Dual-Pentium 1GHz computer. One processor runs Windows XP, under which CATIA operates; the second processor uses VentureCom's Real-time Extensions (RTX), version 5.1.1 for Windows XP, to provide the real-time control environment.

Each mill axis is controlled by a digital torque drive developed by Semifusion, Inc. These digital torque drives send and receive digital data via two fiber-optic cables that connect them to the computer. The controller software uses these digital torque drives as torque slaves, sending commanded torque to the motors and receiving actual position, speed, torque, current, and errors as feedback, all in digital formats.

The first experiment considered a scaled down 3-D CAD model of a car headlight, using a model from one of the large automobile companies. The RMAC controller was loaded with the DMACXYZ.dll device driver to configure the mechanism as a 3-axis sculpting mill. Catia's CAM process was then used to directly control the material removal process as shown in Fig. 6. A customized direct machining tool bar was embedded into CATIA as shown in Fig.6.

The headlight surface was machined directly out of CATIA. For operational comparison, results were compared to the actual machining operations used to machine the headlight model using a Tarus prototype mill at the automobile company. An operations comparison between the RMAC process and the traditional M&G method is shown in Tab. 4. It takes four steps for RMAC to create a physical part from a 3-D CAD model. However, it takes eight steps and seventeen additional minutes to create a part from the same CAD model through the conventional M&G code method, assuming no organizational delays that will normally occur when multiple departments and personnel are involved.

The resulting decrease in operational time does not result from a reduction in actual machining time, but from a decrease in the time required for tool path post-processing and file handling. RMAC eliminates all unnecessary intermediate files, normally generated for conventional controllers, and other unnecessary process steps used in the conventional machining method. Direct control greatly simplifies the traditional design-to-manufacturing process.

The second test was conducted on the same mechanism, but the device driver DMACXY.dll was loaded so the mechanism could act as a router or a planer. A second process plan moved the tool through a sequence of XY tool paths as shown in Fig. 7. Thus, after the headlight surface was sculpted by the three-axis mill version, DMACXYZ.dll was unloaded from memory, then DMACXY.dll loaded. This driver commanded the prototype mill as a 2-axis face milling machine. The face milling process in Fig. 7 was then passed through the direct controller, resulting in the machined part of Fig. 7.

RMAC Machining		Conventional Machining	
Description	Time	Description	Time
Creates parts	equivalent	Creates parts	equivalent
		Sends to tool path planner	5 min
Search/select proper machine	equivalent	Tool path planner searches/selects proper machine	equivalent
Generates tool path	equivalent	Generates tool path	equivalent
		Post-processes to M&G codes	2 min
		Sends M&G files to machine operator	5 min
		Machine operator loads M&G codes on CNC machine	5 min
Runs programs	equivalent	Runs programs	equivalent
4 steps		8 steps	17 more min

Tab. 4: Direct reconfigurable machining vs. conventional machining process.

6. SUMMARY

RMAC, a software alternative to hardware based proprietary controllers, offers many advantages, among them flexibility, simplicity, machine reconfiguration, easy operation, and reduced costs. Previous applications proved that software

reconfiguration could be used to make mechanisms perform in non-standard ways, leading to the RMAC architecture. This paper showed that a machine can be reconfigured into multiple mechanisms for different applications.

By developing modular software with device drivers, controller software components, and a configuration system, the RMAC controller can replace a *dedicated process machine* with a *general purpose, reconfigurable mechanism* for any specific application. CAD/CAM users can search for an optimal machine tool based on the needs of the current manufacturing process. A mechanism device driver will then be automatically loaded to connect the selected machine tool directly to a CAD/CAM application. Rather than serving only to plan the manufacturing process, the CAD/CAM application becomes a process controller by configuring a device driver that connects the intended application directly to the machine.

This paper demonstrates that a 3-axis mill can be reconfigured into either a 2-axis face mill (or router) or a 3-axis sculpting mill, depending on application purpose. These experiments also show how the RMAC control methods radically improve manufacturing efficiency by eliminating the traditional post-processing and file conversion/handling processes.

RMAC can eliminate or reduce a number of manufacturing inefficiencies configured around antiquated technologies and production processes. RMAC's modular, reconfigurable software structure will ultimately allow end users to quickly integrate new technological advancements into their production environment, stimulating more competitive manufacturing organizations in an increasingly globalized market. RMAC seamlessly transforms CAD/CAM applications into machine control systems, making the organizational boundaries between design, process planning, and production less defined, with less process conversion and data management than currently required in production environments. Unfortunately, these simplifying methods also prove to be disruptive to existing organizational structures.

7. REFERENCES

[1] Li, W.; Red, E.; Jensen, G.; Evans, M.: Reconfigurable Mechanisms for Application Control (RMAC): Architecture, Computer-Aided Design and Applications, 4(1-4), 2007, 539-547.

[2] Li, W.; Davis, T.; Jensen, G.; Red, E.: Rapid and Flexible Prototyping through Direct Machining, Computer-Aided Design and Applications, 1(1-4), 2004, 91-100.

[3] Cheng, Z.; Wei Li; Cheatham, R.; Wang, J.; Jensen, G.; Red, E.: A Direct Machining System for Commercial CAD/CAM Packages, ASME International Mechanical Engineering Congress Conference, IMECE-59992, November 13-19, 2004, Anaheim, California.

[4] Davis, T.; Carlson, S.; Red, E.; Jensen, G.; Sipfle, K.: Flexible in-process inspection through direct control, Measurement, 39, 2006, 57-72.

[5] Carlson, S.; Red, E.; Jensen, G.: Development of a Variational Part Model Using In-Process Dimensional Measurement Error, Computer-Aided Design and Applications, 3(1-4), 2006, 405-414.

[6] Cheng, Z.; Jensen G.; Red E.: Direct Generative Vehicle Machining, Computer-Aided Design and Applications, 2(1-4), 2005, 547-555.

[7] Red E.: A dynamic optimal trajectory generator for Cartesian path following, Robotica, 18, 2000, 451-458.

[8] Cheatham, R.; Red E.; Jensen, G.: Direct Process Control Using n-Dimensional NURBS Curves, Computer-Aided Design and Applications, 2(5), 2005, 825-834.

[9] Zhaoxue, Y.; Red, E.: On-line Cartesian Trajectory Control of Mechanisms along Complex Curves, Robotica, 15, 1997, 263-274.

[10] Li, W.; Red, E.; Jensen, G.: An Open Device Driver Architecture for Direct Machining and Control, Computer-Aided Design and Applications, 2(1-4), 2005, 557-566.