# An Open Device Driver Architecture for Direct Machining and Control

Wei Li[1], W. E. Red[2] and C. G. Jensen[3]

[1]GM China, wl32@et.byu.edu
[2]Brigham Young University, ered@et.byu.edu
[3]Brigham Young University, cjensen@byu.edu

**ABSTRACT**

This paper proposes a Reconfigurable Mechanism for Application Control (RMAC) framework, which allows for mapping a mechanism into a device driver for direct control by an application like CAD/CAM. The RMAC paradigm is one of a mechanism device driver assigned to each mechanism class or model, which can be used by a CAD/CAM process planner to control the mechanism. Under the RMAC architecture, the traditional M & G code language is no longer necessary since motion entities (NURBS, lines, arcs, etc) can be passed directly to the mechanism through the device driver interface. The design strategy of using dynamic-link libraries (DLL) to form a mechanism ***device driver*** permits a mechanism to assume different operating configurations, depending on the number of axes, machine resolution, and the relevant device driver functionalities. For example, the machine can perform as a milling machine in one instant, and then, by loading a new device driver, act as a Coordinate Measuring Machine (CMM). The architectural framework is explained in detail and the methodology for control software reconfiguration into a device driver is presented. Finally, a device driver to connect a three-axis tabletop mill directly to CATIA is implemented.

**Keywords:** Direct control, CAD/CAM, Device driver, Dynamic-link library.

## 1. INTRODUCTION

The integration of computing advancements into manufacturing operations lags behind the hardware and software innovations. For example, parametric surfaces are now popular and are widely used in product design and manufacturing, such as the complex NURBS mathematical form. Unfortunately, current machine tools still rely on closed architecture controllers that process the same M & G code designed over fifty years ago for punch-tape controllers. Because the standard M & G code specification is limited in the motion forms supported, and in manufacturing intent, each machine tool vendor extends the standard for proprietary forms and process intent. This means that their adapted controllers are closed and proprietary, thus, making it difficult to interchange machine codes between different machine tool controllers. More importantly, the obsolete M & G code does not allow machine tools to easily adapt to new manufacturing demands or technology innovations.

CAD/CAM systems and their part geometry and process instructions must currently be decomposed into the forms required for each machine's controller. This requires that CAD/CAM vendors develop specific post-processing interface software for each machine tool. Indeed, this decades-old design strategy limits the model/process associativity with the actual manufacturing process. Ultimately, the factory floor cannot maintain pace with the revolution in software technologies. Both design and manufacturing are limited in their evolution.

The RMAC architecture proposed in this paper overcomes these limitations by seamlessly integrating CAD/CAM applications to the mechanism control system. A device driver paradigm maps the mechanism configuration and capabilities to the manufacturing process intent of a CAD/CAM process plan. Machine tools are treated like ***part printing*** devices directly connected to CAD/CAM (or other) applications. The controller driver can be run-time loaded by CAD/CAM applications for directly controlling a particular machine.

## 2. BACKGROUND

### 2.1 Related Research

In the past decade there has been significant pressure from end users to open current closed and proprietary CNC architecture. This has resulted in a wave of so-called Open Architecture Control (OAC) system projects [5][7]. The goals are to develop standard components that can be integrated into different control systems to satisfy end users' specific needs.

More recently, with open-architecture control as a basis, some researchers proposed to develop reconfigurable machine tool controllers that will allow end users to have even more flexible control systems on their factory floors. In a European Union-sponsored report [1] in the early nineties, a strategy was outlined to ensure the long-term survival of the European machine tool industry. The report stresses the need for machine tools to be designed and built modularly, so that machine tool vendors can specialize in particular modules instead of complete systems. This strategy requires splitting a machine tool into a set of autonomous functional units that can be "plug-and-play" interfaced to form complete systems for particular customers' needs. The European MOSYN (Modular Synthesis of Advanced Machine Tools) project [4] and the Reconfigurable Machining Systems [10] of the special research program (SRP) 467, sponsored by the German Research Foundation, are all aimed at developing modular and reconfigurable controllers that can be utilized on Reconfigurable Manufacturing Systems (RMS).

Even though significant research has been done, these developed architectures are still insufficient for two reasons. First, these control architectures still rely on machine-dependent M & G codes. Therefore, these so-called open control and reconfigurable control systems are still not truly interchangeable, reconfigurable, or open to machine end users or any third party developers. Second, these control architectures do not maintain associativity between the CAD model, CAM system, and the CNC machine. This is a great deterrent to fully integrated CAD/CAM and sensor-based control.

### 2.2 Step-NC

A modernized machining code standard (ISO 14649), called STEP-NC, is being developed. STEP-NC extends the STEP geometric data exchange standard (ISO 10303), a neutral data exchange format, into the manufacturing domain by defining a two-way interface between CAM process planning systems and NC control systems. STEP-NC is a neutral data description language designed to be CAM independent and NC machine-tool independent; thus, the post-processing of process plans into M & G codes specific to each machine is no longer necessary.

Currently, under the IMS project [8][9] called STEP-NC in Europe and Asia, and Super Model in USA, industrialists and academics are collaborating to deliver a new data model as an ISO 14649 standard for CNC machines and to develop STEP-NC controllers. Even though STEP-NC provides a better link between CAM systems and CNC machine tools, it has not taken the integration process far enough. It still requires the translation of the CAD process planned model into a STEP-NC file that is disassociated with the CAD model before it is processed by the controller.

### 2.3 Direct Machining and Control (DMAC)

Beginning in 1998, the Direct Machining And Control (DMAC) research group at Brigham Young University has developed an open-architecture controller [2][3][6] that directly interfaces to application software like CAD/CAM. The DMAC architecture is configured on a dual-processor platform. Fig. 1 shows the current DMAC architecture. The top layer of the DMAC system is a non real-time Windows CAD/CAM application that runs on one processor. All real-time control applications, such as motion planning and servo-control loops, run on the second processor, with feedback between the motors and I/O occurring over a high speed digital network. The direct communication between CAD/CAM application and real-time application is through a Direct Machine Interface. The DMAC controller has been implemented to directly connect ParaSolids, Unigraphics, Alias, GibbsCAM, CATIA and PC-DMIS, a popular part dimensional inspection application.

The DMAC architecture is fully software-based and can be configured to communicate directly with any CAD/CAM system, given the right interface functionality. This direct control architecture forms the foundation for building device drivers to connect various mechanisms directly to a CAD/CAM application. This is the subject of this paper.

## 3. METHODOLOGY

### 3.1 Conventional controller vs. RMAC controller

In a traditional CNC paradigm, one machine tool controller is dedicated to a particular CNC machine tool. The functionality of that controller cannot be changed by end users for controlling different machines.
Fig. 2 shows the standard steps used to plan a process and conduct it on a machine tool:

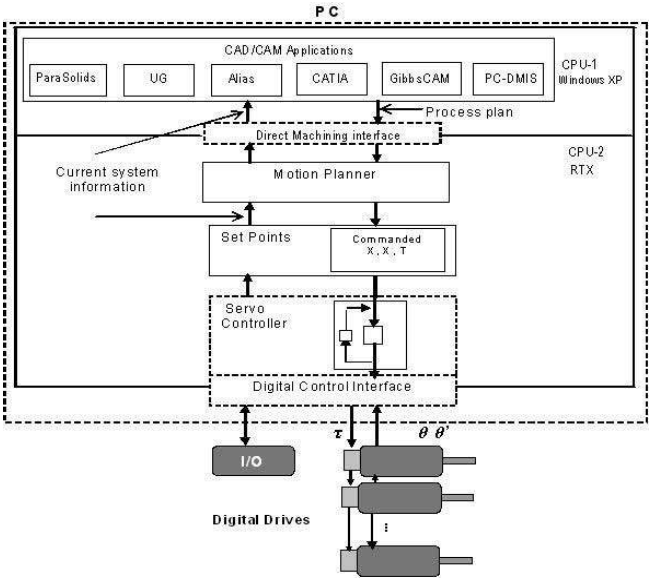


Fig. 1. DMAC architecture

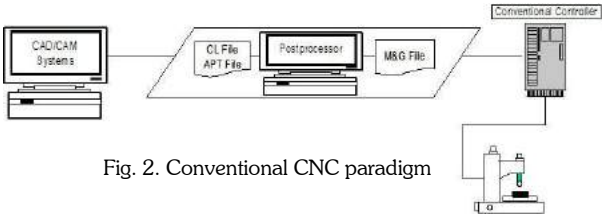


Fig. 2. Conventional CNC paradigm

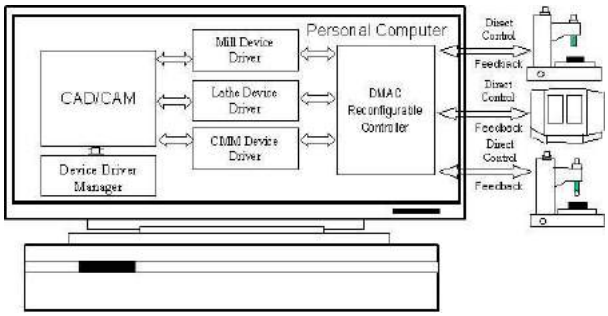


Fig. 3. RMAC paradigm

- Model a part using a CAD system.
- Create tool paths using a CAM system.
- Output a CL or APT file that contains tool path geometry data.
- Post-process the CL or APT file to obtain an M & G-code file, which then is delivered to the machine.
- Operate the machine until the part (or batch of parts) is made.

CL and APT files are independent of any machine tool controllers, but the M & G file is machine specific and is not always interchangeable between different machines.

Fig. 3 presents the RMAC paradigm. Under this new paradigm, machine tools are controlled similar to the way printers are controlled by a personal computer. All machine tools are directly connected to CAD/CAM applications through different device drivers. CAD/CAM users can select different machines to execute the process plans based on manufacturing process requirements. By calling a specific device driver, the CAM process plans and tool paths can be sent directly through the driver interface. The software driver can then enable the same reconfigurable controller for controlling the machine that is connected through this driver.

### 3.2 RMAC control schemes

The RMAC architecture developed in this research is generic, and therefore applicable to various control applications, such as machining, welding, robotics, etc. For these different applications, the control software must be flexible enough to accommodate different control schemes.

*3.2.1 Position and velocity control*

For most modern machine tools or robots, position and velocity control is the most widely used method. Fig. 4 shows how RMAC controls the position and velocity of a mechanism's tool. The

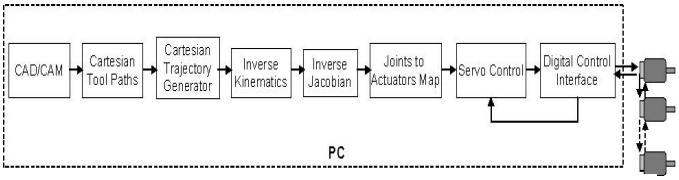


Fig. 4. RMAC controlling steps

first step is to generate tool paths inside a CAD/CAM package. The process plan may also specify path following speeds and a spindle rpm. A Cartesian trajectory generator is used to interpolate the tool paths to generate the tool position as well as the orientation that the tool can follow.

To follow the desired Cartesian tool path, position, velocity, and acceleration setpoints must be found for each individual joint. This requires a mapping between a mechanism's Cartesian space and its joint space. Inverse kinematics is used to map a mechanism's Cartesian state to its joint state. The inverse Jacobian is used to map a mechanism's Cartesian velocity to its joint velocity. The equation $\Theta' = J^{-1}(\Theta)v$ relates the joint speed vector to the corresponding tool speed vector, where $\Theta'$ denotes the joint speed vector and $v$, the tool speed vector. $J^{-1}(\Theta)$ denotes the inverse Jacobian matrix, reduced to a square independent form. The inverse kinematics and the inverse Jacobian are machine-dependent. Therefore, each mechanism requires a specific inverse kinematics and inverse Jacobian algorithm. The motion planner [6] is responsible for generating all these motion setpoints.

Once the actuator setpoints are determined, they are passed to the servo controller [2]. Based on certain servo control algorithms, the servo controller computes the torque value to command the motors to move to those desired setpoints. The servo control algorithms may be machine-specific as different mechanisms may require different servo control algorithms based on the machine tolerance or customer requirements.

Finally, a digital motor and I/O interface is necessary to handle the connections to the digital motor drives and external I/O sensors.

*3.2.2 Force or hybrid force/position control*

While position and velocity control are widely used in machine tools and robots, there are other occasions when position control alone may not suffice. For robotics assembling, or friction stir welding, the contact force or a combination of force and position may need to be controlled.

Fig. 5 shows a hybrid force/position control scheme applied to a three-axis kinematics structure. This mechanism has three prismatic joints directed along X, Y, and Z. The X and Y prismatic joints are free to move, while the Z axis is force constrained

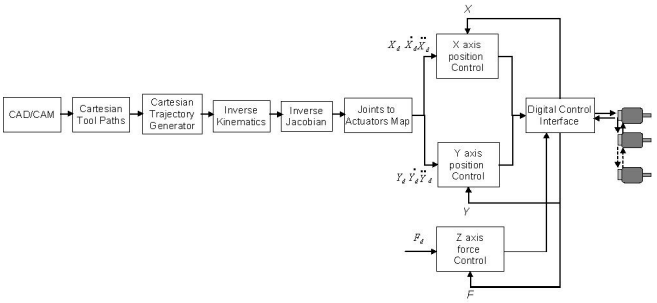


Fig. 5. Hybrid force/position control

The solution to this hybrid force/position control problem is to control joints X and Y with a position controller while simultaneously controlling the contact force along the Z axis with a force controller. Joints X and Y are processed within the motion planning loop as described in section 3.2.1. The Z axis is out of the motion planning loop. $F_d$ is the desired contact force that needs to be controlled. The actual force *(F)* is measured by a force sensor, which is attached to the Z axis. This value is fedback to the force controller for adjusting the necessary control effort for joint Z.

As illustrated, these control schemes are quite different in terms of their control characteristics and the control methods utilized. To take advantage of these control methods and to integrate them into RMAC, a flexible software architecture must be developed, allowing for easy reconfiguration of these different control methods.

### 3.3 Software architecture

Fig. 6 shows the RMAC architecture. The software system is composed of the following six different programs:

- **CAD/CAM system** creates 3D representations of physical models and generates the manufacturing process plans.
- **Device driver manager** maintains a device driver database relevant to a collection of different mechanism devices and their driver DLLs. Meanwhile it provides interface APIs for CAD/CAM users to query for a proper machine and then locates a driver DLL for that machine.
- **Device driver** connects a physical mechanism directly to a CAD/CAM application and processes the CAD/CAM function calls to enable easy reconfiguration of the DMAC controller necessary for direct control.

- **DMAC_Config interface** directs the configuration commands from a device driver to the DMAC controller to allow the reconfiguration of the motion planner, servo controller, and the underlying digital control interface.
- **DMAC_CAM interface** directs the motion and control commands to DMAC, receives the machining feedback information, and sends it to a device driver software.
- **DMAC** open-architecture reconfigurable controller interpolates motion and control commands to generate the necessary torque values to drive each individual motor.

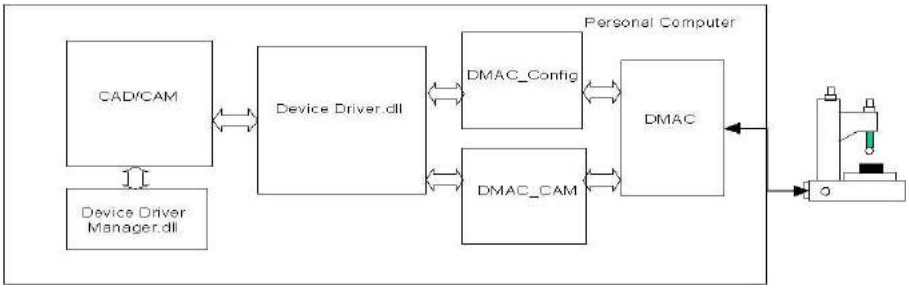


Fig. 6. RMAC overall architecture

The remaining of this section describes each part of the software system as displayed in Fig 6. In addition, the flow of information among these different programs and the interface APIs enabling this information flow is also discussed.

*3.3.1 CAD/CAM*

CAD/CAM systems are computer-aided engineering tools that are widely used to assist product design and manufacturing. Fig. 7 shows a Ford GT top surface being modeled and process planned in CATIA. Traditionally, these manufacturing process plans must be post-processed into the ASCII APT and M & G files to be executed on a machine. To overcome this limitation, a device driver is developed for each individual machine to be connected directly with CAD/CAM systems. Whenever CAD/CAM users generate the manufacturing process plans and are ready to execute them on a machine, they will first select a machine to perform the process. CAD/CAM software will automatically load a relevant device driver and then pass the process plans directly to that machine through the device driver.

Because many different machines exist that are feasible for executing a manufacturing process plan, a configuration dialog box (see Fig. 8a) is designed as a plug-in user interface to CATIA to assist users in selecting the best machine tool. This dialog box allows CAD/CAM users to see the different machines, and provide users with enough information to select the proper machine to perform the process. To better assist CAD/CAM users, the device driver manager has a built-in search engine, enabling them to narrow down their selection to a few machines, based on various machine filter schemes. Fig. 8b shows two selected machine tools classified as 5-axis mills, with working volume greater than 100x100x100 mm. If CAD/CAM users want more information about a particular machine to make a better decision, they can click the machine characteristics button to open a new dialog box, as shown in Fig. 8c.

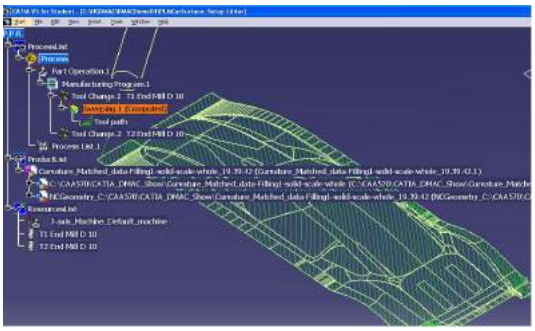

Fig. 7. CATIA process plan

*3.3.2 Device driver manager*

The device driver manager (see Fig. 9) is a DLL running independently from CAD/CAM systems. The functions of the device driver manager are as follows:

- Maintain a device driver database relevant to a collection of different machines and their driver DLLs.
- Provide built-in database search engine to assist CAD/CAM users to narrow down their selected machines based on various search schemes.
- Provide interface APIs to communicate with CAD/CAM applications.

*3.3.2.1 Device driver database*

The device driver manager maintains a database called the *device driver database*. Tab. 1 displays part of the database for demonstration purposes. This database contains the minimum amount of information relevant to a mechanism and its device driver. Its purpose is to allow CAD/CAM users to inquire about a mechanism and its device driver to assist their evaluation and selection of a machine tool.

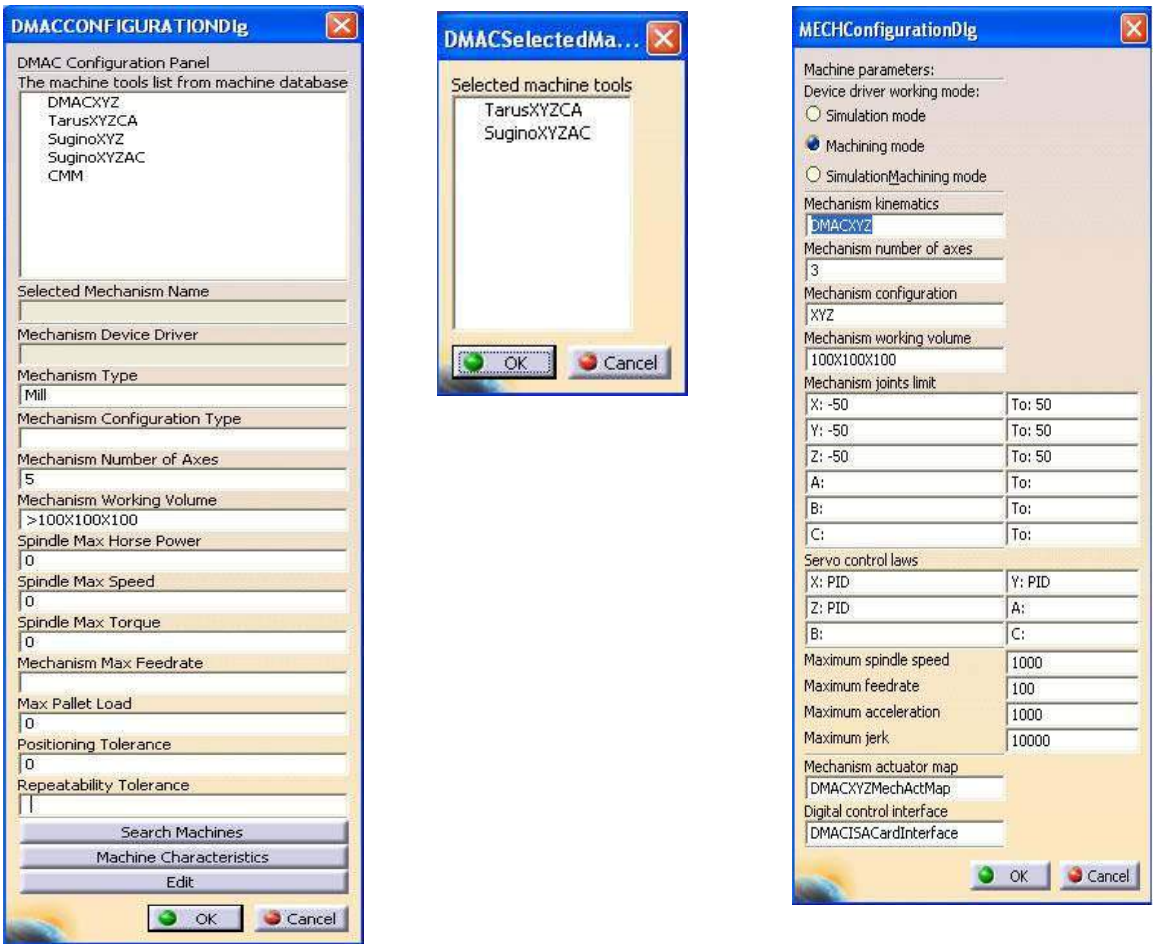


Fig. 8(a) Machine configuration user interface, (b) Selected machine tools, (c) Machine characteristics dialog box

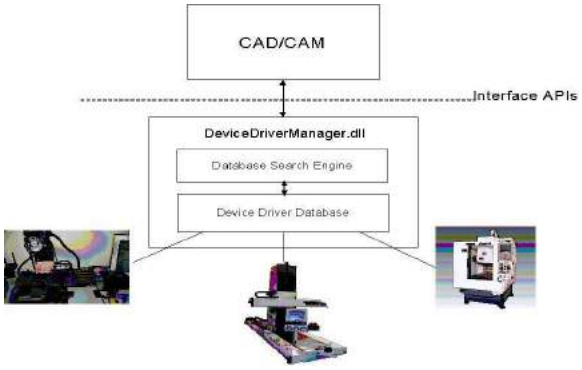


Fig. 9. Device driver manager

Tab. 1. Device driver database

| MechanismName | DeviceDriver | MechanismType | NumberofJoints | WorkingVolume | SpindleHorsePower | MaxFeedrate |
|---|---|---|---|---|---|---|
| DMACXYZ | DMACXYZ.dll | Mill | 3 | 100X100X100 | 2.5 | 2 |
| TarusXYZCA | TarusXYZCA.dll | Mill | 5 | 800X800X800 | 10 | 20 |
| SuginoXYZ | SuginoXYZ.dll | Mill | 3 | 100X100X100 | 5 | 5 |
| SuginoXYZAC | SuginoXYZAC.dll | Mill | 5 | 100X100X100 | 5 | 10 |
| CMM | CMM.dll | CMM | 3 | 100X100X100 | 5 | 2 |

The design form for the database is Microsoft Access, a low-end relational database widely used on small and medium sized databases. The device driver manager uses the *structured query language* (SQL) statements to query this database to search for proper machines upon users' request.

Since a manufacturing organization may have hundreds of different machines for CAD/CAM users to choose from, by using this search engine, CAD/CAM users can easily narrow down their selection to a few machines. For instance, if CAD/CAM users wish to find a 5 axis mill classified machine, 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: TarusXYZCA, (see Tab. 1). This search and selection process reduces the need for CAD/CAM users to review every available machine before finding one capable of performing the manufacturing process. Once the device driver manager obtains this machine and device driver information, it can pass this information back to the CAD/CAM user.

*3.3.2.2 Interface to CAD/CAM*

The device driver manager is designed as a DLL and is a stand-alone program; thus, it must expose some interface APIs to enable communication with CAD/CAM systems. The interface APIs are separated into two groups: functions that allow CAD/CAM systems to access the device driver database, and functions that return the selected machine information back to CAD/CAM.

CAD/CAM applications call the first group of interface APIs 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 APIs to obtain information related to a selected machine. This machine information will then be displayed to CAD/CAM users, as described in section 3.3.1, for proper machine evaluation and selection.

*3.3.3 Device driver*

A device driver (see Fig. 10) must be developed to connect a mechanism device directly to CAD/CAM. It may be useful to think of a complete mechanism device driver as a container for a collection of methods and classes. These methods and classes can be called by CAD/CAM systems to perform various operations on the connected mechanism device and to read back the mechanism operational parameters, such as current feedrate, spindle speed, joint value, current torque, etc. Each device driver must be able to entirely determine a particular mechanism's behavior and understand exactly how to make the mechanism work for the user. Specifically, the device driver should be designed with the following functions:

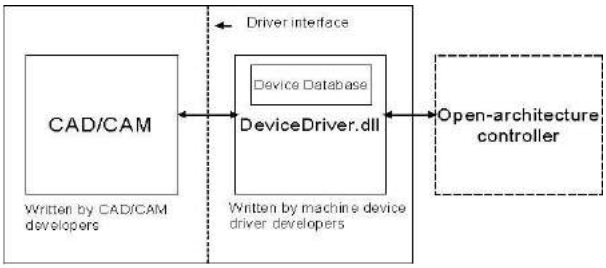


Fig. 10. Device driver

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

The device driver is designed as a DLL. Under the RMAC paradigm, each device driver is assigned to a mechanism class. The functionality of a DLL makes it the perfect form for a mechanism device driver. By using DLLs, the machine-specific software modules can be designed, linked, and debugged independently. These DLLs are separate executable files and are completely independent of all other software. If the functionality of a mechanism device driver needs to be updated or enhanced, the driver developers only need to update this device driver DLL.

*3.3.3.1 Device database*

Each device driver has a self-contained device database, as shown in Tab. 2. The primary purpose for developing this device database is to allow CAD/CAM users to easily access any machine information prior to selecting a particular machine to execute the manufacturing processes. The secondary purpose is to allow the device driver to correctly configure the DMAC controller based on information contained within the device database.

As shown in Tab. 2, the device database contains three categories of information that are relevant to a mechanism device: 1) primary machine characteristics, such as the number of axes, etc; 2) machine operational parameters, such as mechanism maximum feedrate, spindle maximum RPM, etc; 3) machine-specific motion planning and servo controlling capabilities, such as kinematics, servo control law and servo gains used on each axis, joint to actuator mapping, etc.

Tab. 2. Device database

| MechanismName | Kinematics | NumofJoints | MechActuatorMap | MaxFeedrate | SpindleMaxSpeed | Joints | JntType | JntMinLimit | JntMaxLimit | JntMaxSpeed | JntMaxAccel | ServoControlLaw | Kp | Ki | Kd |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMACXYZ | DMACXYZ | 3 | DMACXYZActMap | 2 | 1000 | X | Trans | -50 | 50 | 200 | 1000 | PID | .95 | 0 | .0067 |
| | | | | | | Y | Trans | -50 | 50 | 200 | 1000 | PID | .95 | 0 | .006 |
| | | | | | | Z | Trans | -50 | 50 | 200 | 1000 | PID | 1 | 0 | .0075 |

*3.3.3.2 Interface to CAD/CAM*

The communication between the device driver and CAD/CAM systems are through a set of device driver interface APIs. The interface APIs are divided into three groups: 1) functions that allow CAD/CAM systems to access the device database, 2) functions that return this specific machine information back to CAD/CAM, and 3) functions that instruct the DMAC open-architecture controller to set up correct operational parameters and configure the motion planning and servo controlling algorithms that are specific to this mechanism. The first two groups of APIs are similar to that of the device driver manager's interface APIs described in section 3.3.2. CAD/CAM applications use the third group of interface APIs to correctly configure DMAC prior to executing a manufacturing process plan.

*3.3.3.3 Interface to DMAC*

A device driver connects CAD/CAM applications to DMAC. It receives process instructions from CAD/CAM and then passes them to DMAC. The device driver software uses two COM interfaces, DMAC_Config and DMAC_CAM, to communicate with the DMAC controller. It uses the DMAC_Config interface APIs to instruct DMAC 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 DMAC, it then uses the DMAC_CAM interface APIs to pass process instructions to DMAC. The device driver software contains an instance of the DMAC_Config and DMAC_CAM objects. As a result, the interface APIs contained within these two COM interfaces are directly available to the device driver.

*3.3.4 DMAC_Config and DMAC_CAM interfaces*

To connect any control input to DMAC, multiple COM-based control plug-ins have been developed. These COM-based control plug-ins act as the interface between DMAC and an external control source. Fig. 11 shows a few existing COM interfaces that have been implemented. It also shows a newly developed COM interface that allows for reconfigurable control.

The DMAC_Config interface contains well-defined function calls that allow a device driver to instruct DMAC to reconfigure its motion planner, servo controller, and digital control interface, which, in turn, is necessary to control a particular mechanism.

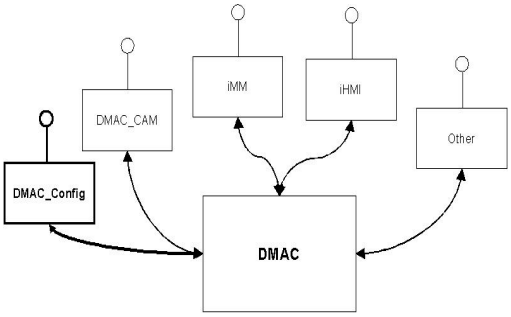


Fig. 11. COM-based plug-ins connected to DMAC

After a device driver instructs DMAC to reconfigure its control software components, the DMAC controller is ready to take motion and control commands for direct machining. Once control commands have been accepted by the device driver software, they are ready to be passed to DMAC. The device driver communicates with DMAC through a COM interface called DMAC_CAM. DMAC_CAM is directly interfaced with the motion planner, and allows the device driver to pass motion commands to DMAC as well as to send and receive other control information.

The interface APIs defined within the DMAC_CAM COM interface are divided into two groups. The first group is used by the device driver to instruct DMAC in executing process plans. The second group allows the device driver to obtain current operational parameters, such as the current spindle speed.

### *3.3.5 DMAC*

To allow reconfiguration of the DMAC controller for controlling different mechanisms, some additions and modifications must be made to DMAC's existing architecture, as shown in Fig. 12. As described in section 3.2 and 3.3.3, any machine-specific module is designed and linked separately into a DLL. This module must first be mapped into DMAC before CAD/CAM applications can send down process plans to DMAC for direct machining. The software module is mapped into DMAC through a configuration system, as displayed in Fig. 12.

The configuration system is directly interfaced with the DMAC_Config COM interface so that it can receive configuration commands from this interface. Based on these different configuration instructions, the configuration system will do one of two operations. It will either set up a correct machine operational parameter, such as machine joint limits; or load the corresponding DLLs and then map any mechanism-specific module, such as the machine kinematics object, into DMAC.

After the configuration system finishes all of these configuration processes, the DMAC controller is dynamically reconfigured for a particular mechanism. CAD/CAM applications can then pass the manufacturing process instructions to DMAC for direct machining.

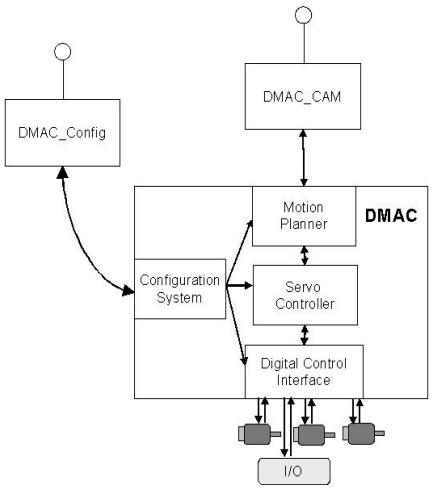


Fig. 12. Modified DMAC architecture

## 4. IMPLEMENTATION

A device driver was developed to connect a three-axis tabletop mill directly with CATIA. The controller runs 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) for Windows XP.

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 as digital information.

## 5. EXPERIMENTAL RESULTS

An experiment was arranged using a scaled 3D CAD model of a car headlight, similar to data that would typically be used in production at GM. Tool paths for the surface were generated in CATIA and sent to a three-axis tabletop mill that was directly connected to CATIA through a three-axis mill device driver, called DMACXYZ.dll. A customized direct machining tool bar was embedded into CATIA (see Fig. 13). The computer runs dual Pentium III processors at 1 GHz. The non real-time CATIA application runs on one processor while the real-time applications run on the second processor.

The headlight surface was machined directly out of CATIA, as shown in Fig. 14. The same process was also completed by using a conventional Tarus three-axis mill utilized at GM. The processing time comparison between the direct reconfigurable machining process and the traditional M & G method is shown in Tab. 3. As this table shows, it took four steps for the direct reconfigurable machining process to create a physical part from a 3D CAD model. However, it took eight steps and seventeen more minutes to create a part from the same CAD model through the conventional M & G code method.

The resulting decrease in processing time does not come from a reduction in actual machining time, but from a decrease in the time required for tool path post-processing and file handling. The direct reconfigurable machining method eliminates unnecessary intermediate files, generated for the conventional controllers, and unnecessary process steps used on the conventional machining method. Therefore, it greatly simplifies the traditional design-to-manufacturing processes.

Tab. 3. Direct reconfigurable machining process vs. conventional process

| Method | Direct reconfigurable machining | | Conventional Machining | |
|---|---|---|---|---|
| | Description | Time | Description | Time |
| Steps | Creates Part | equivalent | CAD designer Creates Part | Equivalent |
| | | | Sends to tool path planner | 5 min. |
| | Searches and selects a proper machine | equivalent | Tool path planner Searches and selects a proper machine | equivalent |
| | Generates Tool Paths | equivalent | Generates Tool Paths | equivalent |
| | | | Post-processes to M&G codes | 2 min. |
| | | | Sends M&G files to machine operator | 5 min. |
| | | | Machine operator loads M&G file on CNC machine | 5 min. |
| | Runs program | equivalent | Runs program | equivalent |
| Totals | 4 steps | 0 min. | 8 steps | 17 min. |

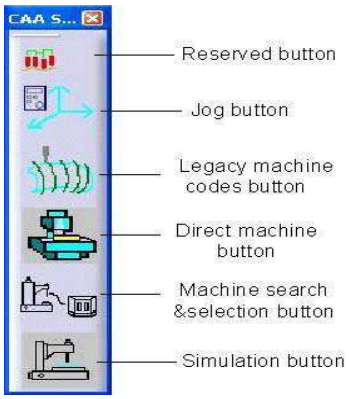


Fig. 13. Direct machining tool bar

Fig. 14. Direct machining a car headlight

## 6. CONCLUSIONS

This paper describes a new control architecture that allows mapping of each mechanism into a device driver and uses this device driver to reconfigure a DMAC open-architecture controller. This provides CAD/CAM users with greater flexibilities to fulfill different manufacturing operations. Under this RMAC control paradigm, 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. Experiment shows that a machine device driver can be run-time loaded by a CAD/CAM application for controlling a selected machine tool. It also demonstrates that a great amount of manufacturing process time can be saved by eliminating unnecessary process steps used on the traditional machining methods.

## 7. REFERENCES

[1] Atkins W. S., 1990 Strategic Study on the EU Machine Tool Sector, *Management Consultants*.

[2] Evans MS, Red WE, Jensen CG, Open Architecture for Servo Control using a Digital Control Interface, *Proceedings of the IASTED International Conference on Control and Applications,* 2000, pp. 339-344.

[3] Li W., Davis T., Jensen C.G., and Red W.E., Rapid and Flexible Prototyping through Direct Machining, *Computer-Aided Design and Applications*, Vol. 1, Nos. 1-4, CAD'04, 2004, pp. 91-100.

[4] M. Zatarain, E. Lejardi, and F. Egana, Modular Synthesis of Machine Tools, *Annals of the CIRP*, Vol. 37/1, 1998, pp. 333-336.

[5] P. Lutz and W. Sperling, OSACA – The Vendor Neutral Control Architecture, Proceedings of the European Conference on Integration in Manufacturing, 1997, 10pp.

[6] Red E, Evans M, Jensen G, Bosley J, Luo Y., Architecture for Motion Planning and Trajectory Control of a Direct Machining Application, *Proceedings of the IASTED International Conference on Control and Applications*, 2000, pp. 484-489.

[7] S. Fujita, T. Yoshida, OSE: Open System Environment for Controller, 7th International Machine Tool Engineers Conference, 1996, pp. 234-243.

[8] Suh S, Chung D, Lee B, Cho J, Cheon S, Hong H, Lee H., Developing an Integrated STEP-Compliant CNC Prototype. *Journal of Manufacturing Systems*, 2002, pp. 350-362.

[9] STEP-NC consortium, ESPRIT Project EP 29708, STEP-NC Final Report, version 1, Nov. 2001.

[10] http://www.5fb467.uni-stuttgart.de/objectives/index.html