OSEC-II Project Technical Report

Development of OSEC (Open System Environment for Controller)

OSE Consortium

< TABLE OF CONTENTS >

Appendix: Machine Resource Object and Function Block API of OSEC-II

< TABLE OF FIGURES >

< TABLE LIST >

Foreword

On December 1994, three machine tool builders -- Toyoda Machine Works, Toshiba Machine, Yamazaki Mazak --, two information system companies – IBM Japan , SML -- , and a controller builder -- Mitsubishi Electric -- have established the project group OSEC (Open system Environment for Controllers) for an open architecture controller which will be used as a foundation to build up the next generation controller for the general factory automation equipment such as CNC. Since then, several companies and body who agree and collaborate to work for the open FA controller, have joined to OSEC and at present eighteen companies and one organization are promoting the project lively. As it is well known to, several open architectures have been proposed in USA and Europe which will meet flexibly to wide range of FA devices such as a NC machine tools and robots. OSEC is an abbreviation of an open controller which is proposed by OSE project group as a result of its works. Main objective of OSEC is to be “open”, and the architecture and the interface specifications discussed within it will be disclosed in public, and expected to contribute some day for the standardization of FA equipment for the CALS .

This document is a Version 2.0 of OSEC (OSEC-II) and it is a revised version of OSEC Version 1.0 (OSEC-I) published in September,1995. This new revision has also been released in public in accordance with the open objective of OSEC. We would like to disclose the result to as many vendors and customers as possible, and would like to receive wide range of opinions and suggestions to accomplish it.

We firmly believe that if the equipment based upon the OSEC specifications were prevailed among the world, multiple-vendor environment in its true sense would be realized and more user oriented and more user friendly open CNC could be freely chosen among the open market. Collaborating with relating bodies, OSE project will be going to up-to-date the developing system for open controllers , keeping up the transparency of its activities so that any bodies or companies could participate freely to it and inherit the concrete results of its activities.

It is our greatest pleasure if disclosure of this OSEC specifications could give any contribution to the progress to prepare and develop the environment for the next generation controller and to activate the relating industries.

List of OSE project group members;

Brother Industries, Fuji Electric, Fuji Machine Manufacturing, IBM Japan, JSPMI (Japan Society for the Promotion of Machine Industry) ,Karatsu Iron Works, Komatsu, Kubotech, Mitsubishi Electric, Niigata Enjineering, Nippei Toyama, SML, Sony Precision Technology, Toshiba Machine, Toyoda Machine Works, Tsugami, Yamazaki Mazak, Yasunaga, Yuasa

1 Summary of OSE Project Group Activities

Fundamental function of CNC is to be illustrated as a some kind of information converting device which convert the three dimensional configuration information into the control information which give motion to the control object (Figure 1-1).

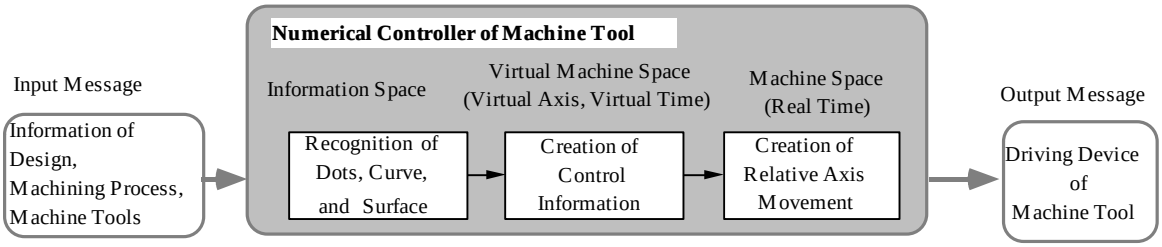

Figure 1-1CNC Fundamental Functions

Many functions to support manufacturing process are available at machine tools through CNC fundamental functions. So far, these CNC functions have been locked up in the black-box as shown in Figure 1-2

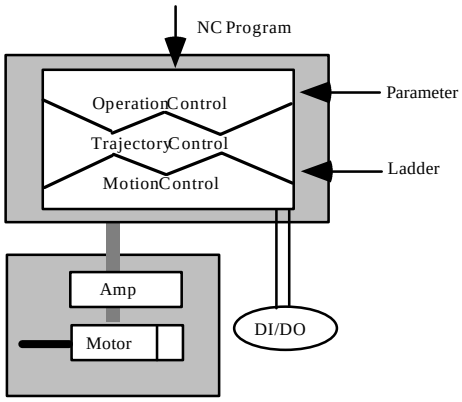

Figure 1-2 Conventional NC

A mission of OSE project is to study and develop the architecture and standard interface of manufacturing machines for the future open manufacturing system, based on the fundamental of control.
As a result, it is intended to achieve an open manufacturing environment in which machines can be freely constructed according to the manufacturing environment and purposes of the end user, easy to preserve and maintain the machines, and possible to minimize the life cycle cost.

1.1 Activities and an Outcome

Main activities of OSEC project are shown as follows;

■1994
Dec OSEC-I project started
■1995
25 Sept Release of OSEC-I& demonstration
(at Toyoda Machine Works @Kariya)

OSEC Architecture Version 1.0 (OSEC-I)
Two test stations conforming to OSEC-I& CAD/CAM station
Disclosure of OSEC-I specifications through internet home page
Home page URL http://www.sml.co.jp/OSEC/

Nov. OSEC-IIproject started
New member of OSE project was publicly recruited
(18 companies and one organization at April,1997)

■1996

26Aug. Release of OSEC-II (at JSPMI hall @Shiba Park)
OSEC Architecture Version 2.0 (OSEC-II)

27～28Aug. Demonstration of OSEC-II systems
(at JSPMI Technical Research Institute @ Higashi Kurume Tokyo)
Three test stations conforming to OSEC-II, OSEL system from CAD to real machining & remote monitoring through manufacturing network

Nov. Systems conforming to OSEC-II were exhibited at JIMTOF
(Japan International Machine Tool Fair)

1.2 Organization

Organization of OSEC at a point of April 1997 is shown in Figure 1-3. Steering committee is a body to decide fundamental policy and exterior activities of OSEC. Executive committee is an administrative body which works under steering committee to execute several working activities. There are four working groups in OSEC-II; Control WG, Human interface WG, Programming WG and Network WG ; these working group have conducted a study what the additional merits should be in open controllers from each points of view (i.e. from machine sides, operator sides, CAD sides and manufacturing network sides). Architecture working group review and confirm the OSEC architecture from a wide point of view, and it has conducted adjustment between each specifications proposed from each working groups.

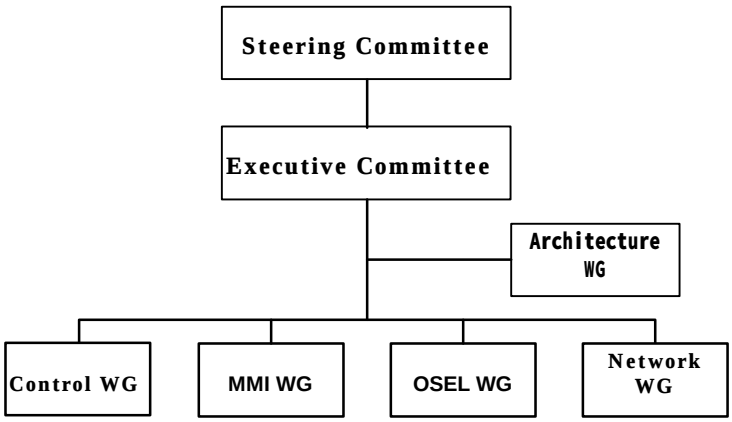

Figure 1-3 Organization of OSE Project Group

1.3 Outline of OSEC-I Project

At OSEC-I, the architecture for open controllers in its true sense was studied and constructed by defining a reference model shown in Figure 1-4. Within OSEC-I architecture, controller is defined as a part of manufacturing system, and information processing procedure from design data to machining process is clarified. More concretely

said, manufacturing system , including CNC which have been locked-out as black box, is divided into several functional layers. These seven layers include a layer to process input data , a layer to calculate and generate cutter locations, a layer to control machine and so forth, and the services and capabilities of each layer have been defined.

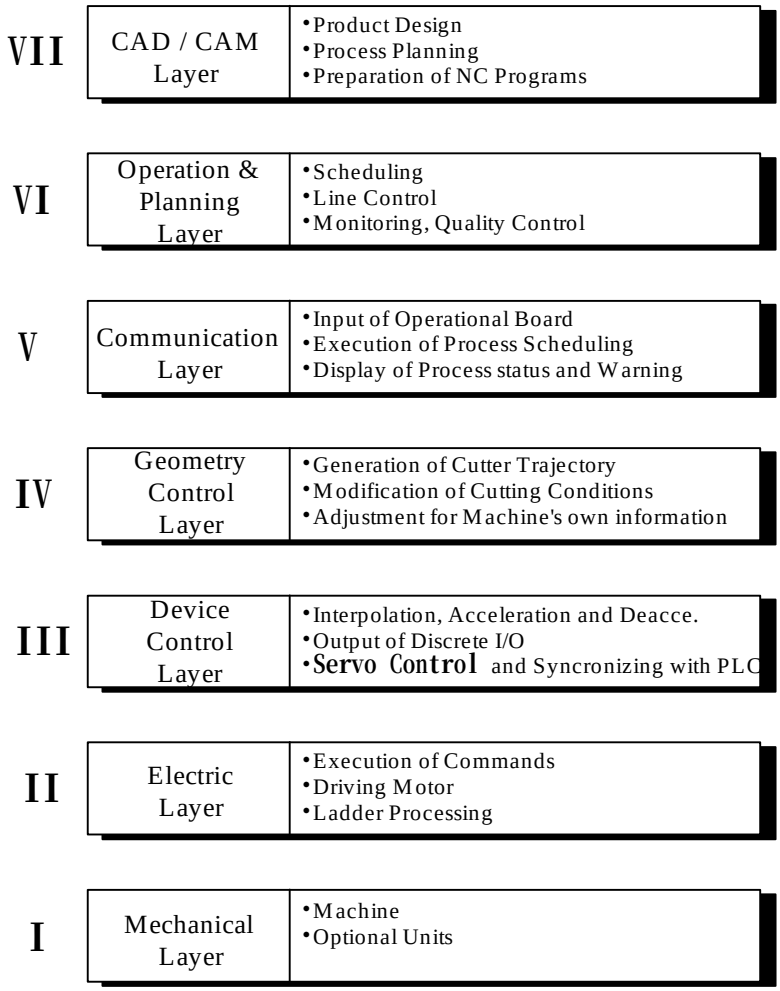

Figure 1-4 OSEC Reference Model

In addition, OSEC-I proposed a new FA equipment description language instead of EIA code (G code) programming, which have been used since 1960s but have been pointed out its defect in regards to flexibility, expandability and portability. FA equipment description language is constructed based on the script type programming language which has a control structures such as an arithmetic expression , repeat, conditional branch. And it is also intended to equip with functions to describe machine dependent information such as machine specifications, tool information, fixture information in abstract description format. It also intend to describe free curves/free surfaces directly through its arithmetic expression and several primitive axis motion commands have been installed. It was also intended to stimulate part program packaging by increasing libraries composed of macros and functions.

1.4 From OSEC-I to OSEC-II

Architecture Reference Model of OSEC- I was intended to clarify the information processing procedure from design process to machining process. Manufacturing process , including CNC which have been treated as black-box, was divided into several layers, and services and capabilities of each layer was defined.

However, this reference model is rather too abstract to install open controllers in practice. In OSEC-II, reference model have been seperated into following three categories – the abstract model (OSECI),the architecture model in which functions and interface is clearly defined, and the installation model on which hardware and software can be easily mapped in practice. Several software parts for control modules and machine resource object model for a human-machine interface have been produced from these architecture models.

FA equipment description language of OSEC-I have been mainly intended to control manufacturing equipment, and its main target was to increase flexibility , expandability of expression and portability of the programming language, which EIA code is lacked of. Part programming description language of OSEC- II i.e. OSEL, is further more upgraded in that machining methods have been packaged in a class library so that machining know-how necessary to make efficient use of machine tools can be circulated in an open market as a software modules. The methods to define machining procedure for several machining features such as holes or grooves have been discussed.
If we figure the part programming language to the computer programming languages, EIA code may be compared to a low level assembly language, FA equipment description language may be compared to a flexible and expandable C language, and OSEL may be compared to an object oriented C++ language which enables and promotes software packaging.

2 The Basic Idea of OSEC Architecture

2.1 Targets of OSEC Architecture

The OSEC Architecture is defining the industrial machine controllers in the style of standard platform, and is open for public. The purpose of setting this open architecture is to provide end users, machine makers, controller vendors, software venders, system integrators, etc. the means with which they can add their own unique values to the industrial machines, and hence to promote the technical and commercial development of the industrial machines.

We identified the value-addition of the industrial machine controllers from several aspects of their environments; i.e. Machine itself, Operator, CAD System, and Network System (Figure 2-1). We then discussed about OSEC architecture so that the architecture can make it easier to created those added values. This approach naturally led us to the definition of the OSEC architecture of open controllers as the mechanism which unites and coordinates the machine, the operator, the CAD system and the network into an well organized system.

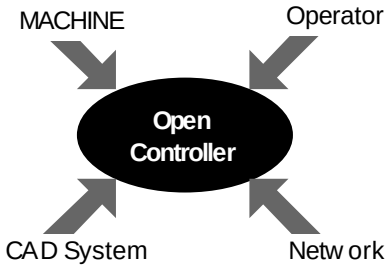

Figure 2-1 Four Environmental Aspects of Open Controller

(1) Machine Aspect of Open Controller

Establishing controller software components is an essential factor in making controller software and hardware partially interchangeable. OSEC API is defined in the form of message which is exchanged among controller software components representing the functionality and the real-time cycle. The controller software components are integrated into a consistent control system by means of this message exchange mechanism.

(2) Operator Aspect of Open Controller

In order to make the efficient development and improvement of operation handling software, the software development method of "application frame work" is introduced in OSEC architecture. This approach allows the selective implementation of man-machine interface system out of wide range of pre-defined components. The objective of operations, the industrial machines, is collectively defined as the standard Machine Resource Object. The OSEC operation handling software is developed as to control this Machine Resource Object, and thus is assured its portability to different computer system environment.

(3) CAD System Aspect of Open Controller

A new machining description language called OSEL (OSE Language) is proposed, which describes the machining process based on the pre-defined and accumulated software components. By this approach, the development of machining software which produces the parts defined by CAD System becomes highly efficient. The OSEL software components, which are created from the concrete knowledge of

manufacturing technologies, may also accelerate the shift to a new business paradigm where the software parts are developed, distributed and procured via various network systems.

(4) Network Aspect of Open Controller

Among the wide range of network application in manufacturing, we focused on the monitoring of production line working status. The interface protocol named OFMP (OSE Floor Management Protocol) is defined to transmit and collect the work status information of machining sites, and it is proposed to handle this via the Internet. This aspect of OSEC is still in working status, and accordingly the OFMP definition is not yet fully given.

2.2 Principal Policy of Setting OSEC Architecture

- OSEC Architecture should be defined as the evolutionary system, not revolutionary.
- The architecture should be software oriented, and be open based on components defined according to the identified functions, and to the objectives.
- The architecture should be able to cope with multi-platform enfironment.
- The architecture should cope with the controller component standards which are and will be developed by respective industries. It should also be well positioned in the job and information flow of manufacturing practice from the upstream of design to the downstream of shop floors.
- The architecture should be useful in the distributed manufacturing practice. It should also make the industrial machines useful as production management terminal in the integrated development and manufacturing environment, such as CALS.

2.3 Supporting Technologies

An open system, in general, will be developed in two levels; the whole system and its components. The whole system depends its technical level heavily on the supporting components, but at the same time, it will also possible to absorb the minor change of components and to keep the system consistency as a whole. This is one of the merits of open system concept. This feature is important for the digital system design based on software and hardware technologies, which are quite rapidly evolving at present. In this project, we considered the progress of the following list of supporting technologies essential for constructing the industrial machine controller system as an open system. Naturally, the list is contemporary, and should be revised as the time goes by.

■Software Development	Object Oriented Technology
	Application Framework
	Platform-Independent Processing Paradigm
■-Control System	Digital Control
	Distributed Control
■Networking	Internet / Intranet
■CAD/CAM Technology	Feature-Based Modeling

3 An Overview of OSEC Architecture

This chapter explains how OSEC defines the functional elements that support the four aspects of the open controller, which are described in the chapter 2.

3.1 Controller Software as Software Component: OSEC API

The software of manufacturing equipment controller has been treated as a black box. In contrast, OSEC aims at establishing an open architecture of controller software. We first revealed the controller software in details by applying the latest software development method, then arranged and redefined it as software components. To be concrete, we broke the controller software down to the level at which they can be implemented, then categorized them concerning their responsibilities, processes, and real-time requirements. Thus we could reorganize these groups to software components named "functional blocks". By clarifying the architecture of the software system for the controller, OSEC aims at enabling the controller software to be plugged and played as each functional block. Furthermore, we defined "function groups" that put functional blocks together at the same level in accordance with the functional layers defined in the OSEC-Ⅰ specification (refer to table 3-1).

Table 3-1 Function Groups in OSEC Architecture

Function Group	Function Block	Response Time	Service Description
Resource Management	Resource Manager	32～100msec	Manage and control system behavior by events from both peripheral devices and software.
Motion Generation	OSEL Executor	8～100msec	Generate motion by interpreting OSEL-X that is the final output of OSEL program.
	EIA Code Decoder		Generate motion by interpreting EIA data.
	Manual Control		Generate motion to manual input like jog or dial.
Machine Control	Machine Control	8～16msec	Execute interpolation, acceleration and deceleration, etc.. Synchronize servo and I/O.
Device Control	Servo Control	1～2msec	Control servo.
	PLC DI/DO	20～50msec	Input/output from/to the ports of PLC.

Table 3-1 summarizes the function groups and their services. It also includes required real-time orders assumed to reorganize the functional blocks.

Conceptually, we set a policy to make each functional block as software component in shape of object, then defined messages to the object as interface protocol (OSEC API). By this policy, each functional block can be encapsulated as an object thus we do not necessarily deal with how a functional block processes messages to it at architecture level. This policy assures that developers can freely design and implement functional blocks. With this encapsulation, companies and individuals can add their own values independently. Since we do not define the process implementation, we do not have to define any cohesion of functional blocks either. OSEC architecture defines only the contents of services of logical functional blocks and messages to them. Therefore, in OSEC architecture, functional blocks do not form layered structure, but they are connected to other blocks in parallel through messages (OSEC API) (refer to Figure 3-1).

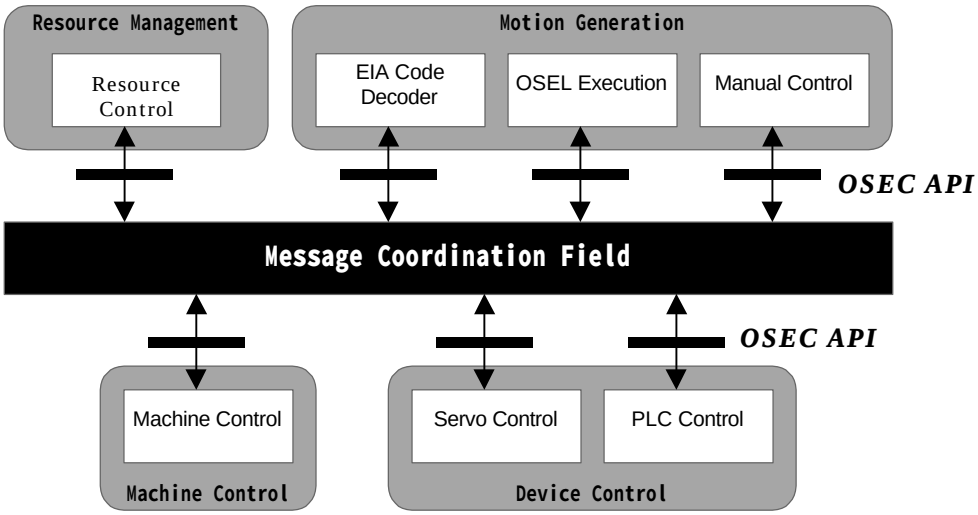

Figure 3-1 OSEC Architecture Model

In order to enable the plug and play of control component hardware such as motor and I/O, each hardware device is encapsulated in the device control function group. The only interface to those hardware devices is messages to corresponding functional block like servo control or PLC control.

OSEC architecture is abstract as the result of generalization to cover wide range of control systems as much as possible. Although the structure of functional blocks can be defined uniquely by OSEC architecture from logical point of view, the system is neither determined nor limited at its implementation phase because there are so many options for implementations. These options may include system contrivances such as device driver, process, and inter process communication, installation mechanisms such as static library and DLL, hardware factors like selection of controller card, and implementations of software modules added for execution control and/or monitoring of various software. In another word, the implementation model to realize the architecture model is not limited to a particular model. In this way, it is assured to incorporate various ideas in the implementation model depending on the system size or its hardware implementation and/or utilization. Figure 4-1 is an implementation model that defines relationship among functional blocks explicitly.

3.2 Standard Machine Model for Operational Software: Machine Resource Objects

Human machine interface draws high attention from machine tool builders and end users concerning realization of operational environment customized for equipment. Since we are setting our goal to assure independence of software for the open controller, the machine tool builders and end users are expecting that they can construct and/or customize operational software as desired.

In order to materialize this, the following should be considered:

- Development tools for software components
- Reusability of components such as GUI components
- Interoperability with existing application software products
- Utilization of latest software technologies

■Portability of operational software

In order to satisfy the above requirements, OSEC architecture for operational software introduced "application framework" as development method extensively.

In general, as to a template of operational display software, necessary functions can be covered well by applying MVC model that divides the software into three domains namely Model, View and Controller. If the MVC model applies to the operational software for machine, Model represents a machine to be operated, and View represents a screen and its components like GUI components, and Controller represents a definition of relationship between Model and View. Control is a collection of transactions for instance a transaction to be taken when GUI push button is pressed or a transaction to update the screen upon a change of state of machine. OSEC defined the machine model as objects and named it "machine resource objects".

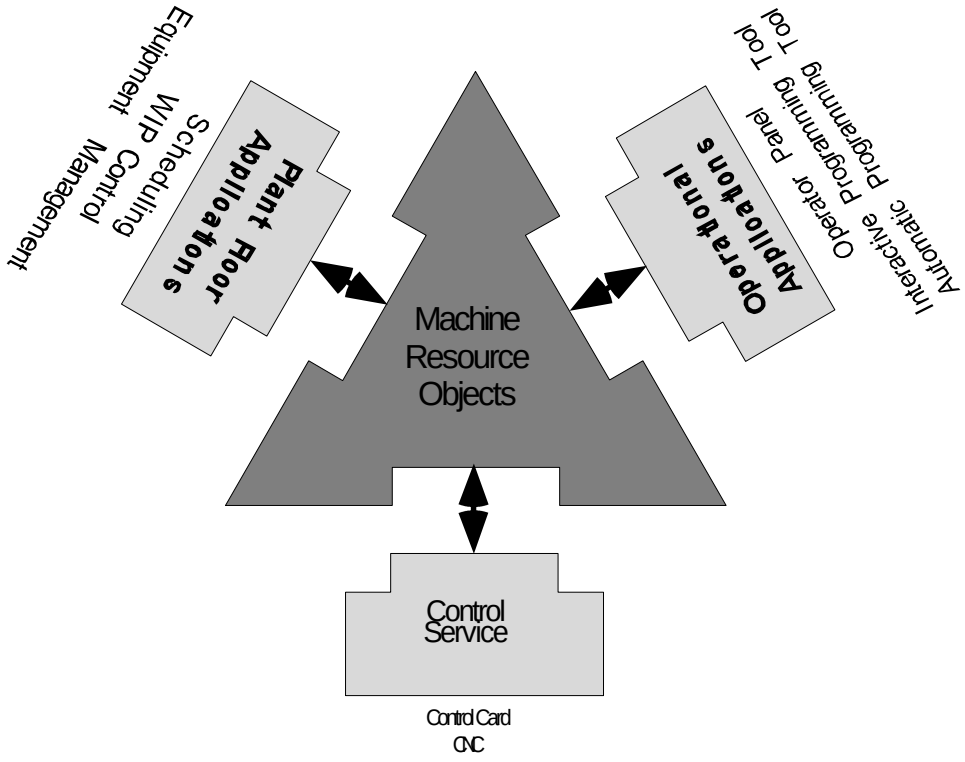

Figure 3-2 Conceptual View of Machine Resource Objects

The machine resource object aims at enhancement of portability, ease of migration, reusability, and expandability by standardizing interface to a machine by abstracting the view of machine from operational applications. The machine resource objects are encapsulated by the following three interfaces (refer to Figure 3-2):

■Application interface to display and operation
■Interface to plant floor applications such as manufacturing execution system
■Interface to NC control block (NC controller card, PLC, servo, etc.)

We analyzed various resources of machine tools such as axis, spindle, tool, and PLC, then defined attributes and methods of each resource clearly. Operational software can give a

command to a machine or retrieve status of it by passing messages.

The machine resource objects represent a model of machine tool based on the concept of object. However, the machine resource objects should have variations reflecting the differences of product types or configurations. Therefore we provided a standard model of abstract machine and enabled to derive objects of particular machine to reflect the differences of product types or configurations. As a result, we can utilize the inheritance concept of the object-oriented method thus the efficiency of development of machine resource objects for particular machine will be improved.

3.3 Process Method Description Language: OSEL

EIA code, that is widely used for NC controller, was standardized by ISO et al in 1960's as standard data format for machine tools. Now the following problems are pointed out.

- It depends on each machine regarding to tooling and machine instructions.
- There is no compatibility among NC controller vendors.
- Data required for curved surface or line becomes huge.
- It is difficult for users to expand or improve it.

In addition to the resolution of the above problems, the purpose of OSEL is to propose a technique to package and reuse the inherent technologies for production and processing accumulated by end users and machine tool builders. This means we propose a significant paradigm shift for new business environment for the future industrial society focusing on networking in that the distribution and procurement of inherent manufacturing technologies captured in software are available.
Figure 3-3 explains the process of OSEL.

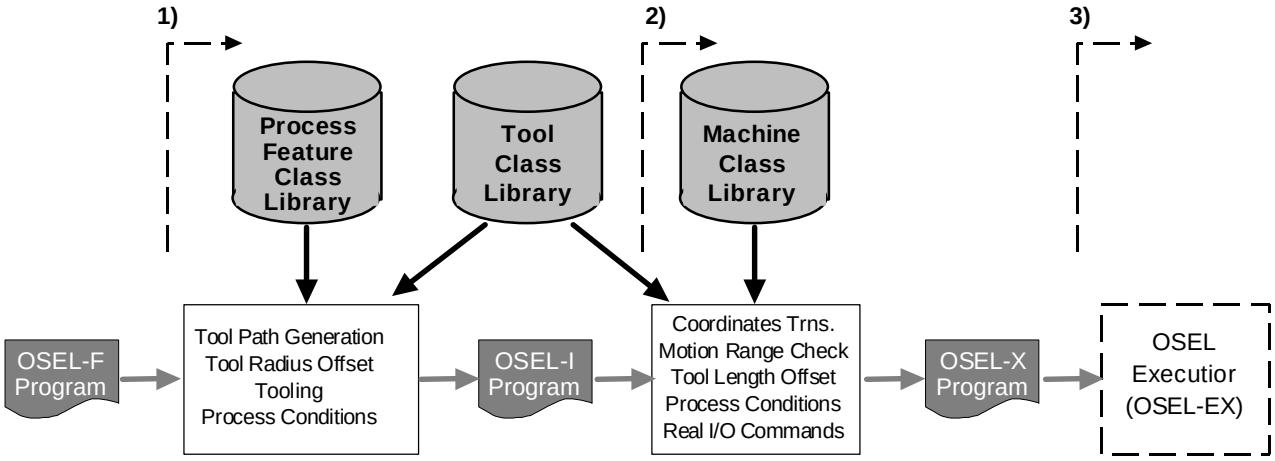

Figure 3-3 Process Flow and Level of Implementations of OSEL

In OSEL-F (Feature OSEL) that is highest level of description, a part program is given in the form of description of process feature like hole or groove. An example of process feature description is "create a perforation hole with drill". OSEL-F is translated into OSEL-I (Intermediate OSEL) by using a process feature class library and a tool class library. OSEL-I is an intermediate description and is independent from individual machine. OSEL-I

is translated into OSEL-X (eXecutable OSEL) using a machine class library. OSEL-X is dependent on individual machine. OSEL-X program interpretation (OSEL EXecutor: OSEL-EX) and following processes require real-time processing.

Process method that is independent from individual machine is defined as methods of classes that are based on process feature like hole, groove, or side face. Machine tool builders and end-users can add their own processing know-how by adding classes derived from base classes. In the machine class library, machine dependent process expansion is described as method. An example of such expansion is a translation from physical machine coordination into virtualized program coordination or a treatment of real I/O instruction.

As to the implementation of OSEL system to the open controller, the developer can select from three implementation models indicated in figure 3-4 based on the system configuration of the controller and/or user requirements. In the implementation model 1), the controller receives OSEL-F program that is highest abstraction level, and is responsible for program translation to motion control. An advanced controller with CAD/CAM may take this configuration. In the implementation model 3), the controller receives OSEL-X as executable program. This configuration can be implemented in low power system.

We evaluated OSEL by implementing it in JAVA.

3.4 Interface of Machine Tool for Manufacturing Network

As to the connection between the existing CNC and manufacturing execution system, it is difficult for us to connect a machine tool to a manufacturing network application since neither CNC's nor manufacturing execution systems have standardized interface. Therefore, developers have had to implement the interface between a machine tool and a networking application by utilizing additional sensors and/or POP terminals.

Emergence of PC based open controllers has made construction and operation of FA system more efficient since machine tools will be integrated into a networking application and will exchange information with remote machine tools.

What is important to construct a manufacturing network including machine tools are to realize multi-vendors environment to procure network hardware and software. It is also important that the machine tools provide a common interface for network.

In order to realize multi-vendors environment, we selected TCP/IP as a network protocol and WEB browser for GUI applications on the network. We also defined a network object for machine tools in order to provide a common interface for network. A plant floor management application can communicate with machine tools in common interface. The network object exchanges messages with the machine resource objects described in section 3.2.

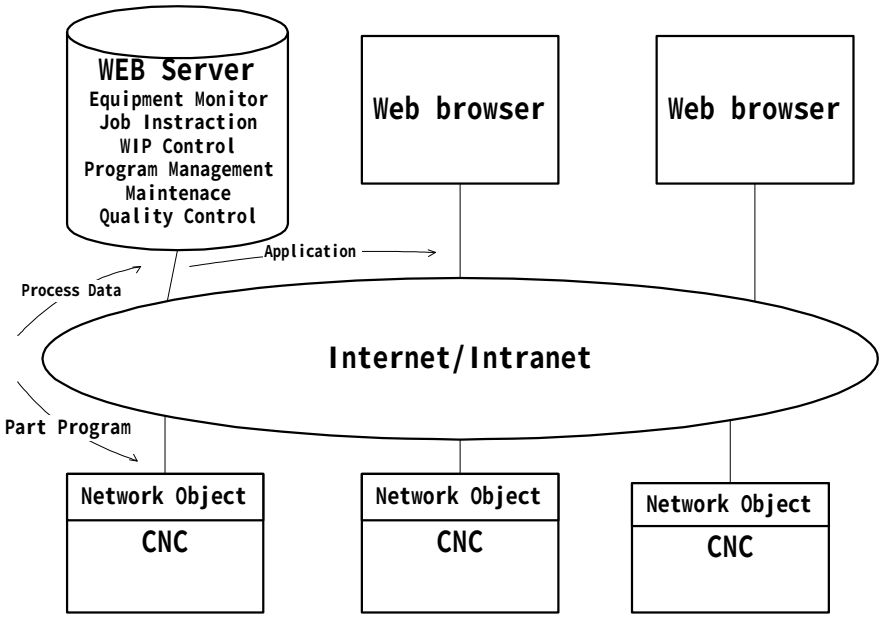

Figure 3-4 Networking CNCs with Manufacturing System

4 Implementation Model with Function Groups and OSEC-API

In chapter 4, the process of designing an architecture of implementation model based on function groups which are defined for some specific purposes, finally implementing it on a micro-computer based controller with hardware and software, are described.

4.1 Implementation Model

In OSEC- II, the function groups composing the open CNC are defined, and then the architecture is defined upon these function groups.
An implementation model is defined upon the following 5 function groups:

- HMI (Human Machine Interface) function groups
- Resources management function groups
- Motion generation function groups
- Machine control function groups
- Device control (Servo control, PLC-DIDO) function groups

Figure 4-1 shows the implementation model with protocol flow of function groups.

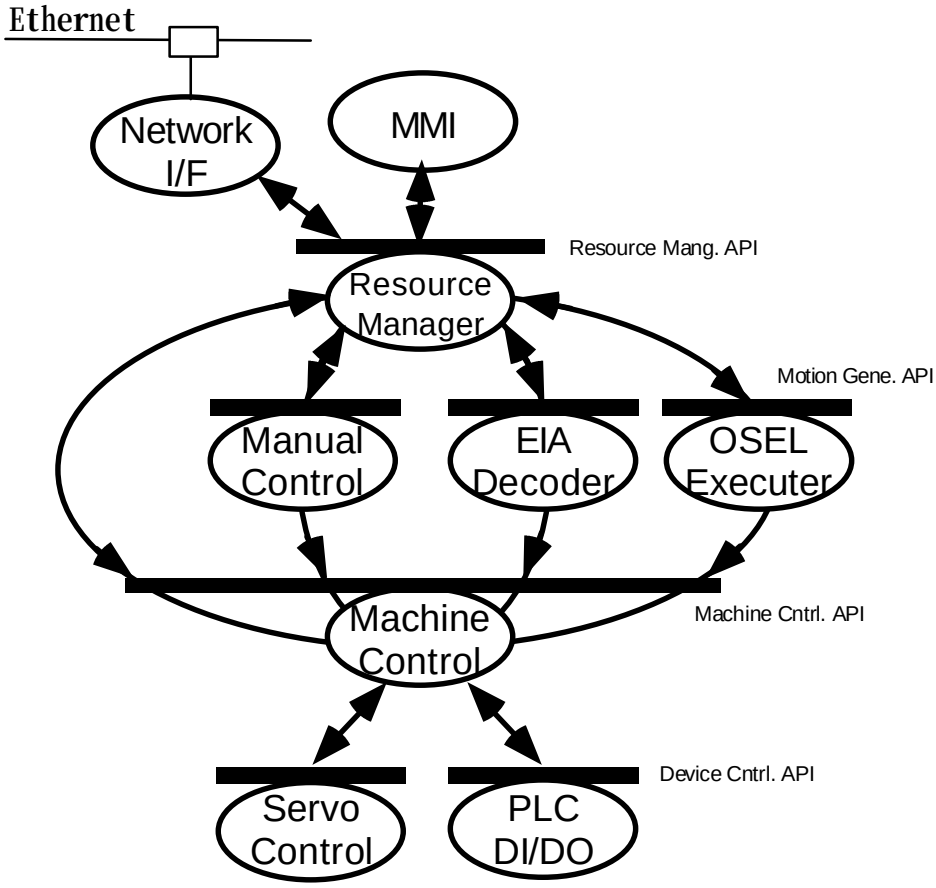

Figure 4-1 Implementation Model for CNC

4.2 Human Machine Interface for Machine Operation

HMI serves as the operation interface of open CNC through which external process can make approaches to various resources of machine tool.
HMI software parts are build by Application framework for Graphic User Interface. Various

resources of machine tools (Machine resources) are defined as abstract object models of classes. By assigning attributes and methods to respective objects, it become possible to make approaches to machine resources in an abstract way. Through building the experimental system, simultaneously with the implementation of Machine Resource Objects, an application with basic operation functions is also developed and the effectiveness of Machine Resource Objects as software part is verified.

The connection between display parts and function models is explained here. The next example shows the connection between MpsAxesView part, which displays information about spindle and movable axes, and function model.

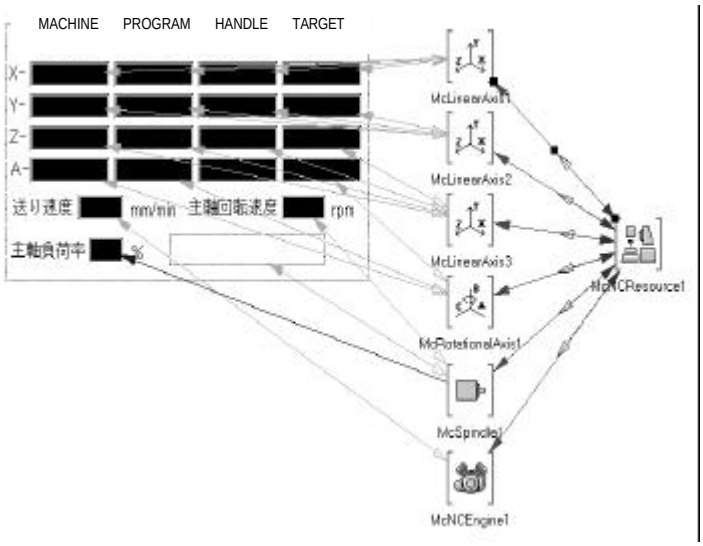

Figure 4-2 Connection between MpsAxesView Parts and Function Models

The lines connecting display parts and function models, represent messages between objects. For example, the current position attributes of movable axes in function models are connected to the machine coordinates display box in display parts, It means the contents of machine coordinates display boxes will be renewed upon the current positions changed.

During the development of experimental system, three different operation panel applications are developed by the member companies of OSEC HMI work group. Although they are oriented to the same machine (a Toshiba-made machining center) ,the actually completed operation panel applications are characteristic of each member, reflecting their respective views of operation panel functions.

4.2 Resources Management

Resource Management is arranged on architecture for coordination Machine Resource Objects ,model of machine tools and actual control system.

The Machine Resource Objects separates applications from control functions and defines distinctly the interface between them. The important role of Machine Resource Objects is to provide an environment upon which the machine tool manufacturer or users can easily integrate their own applications with individual functions.

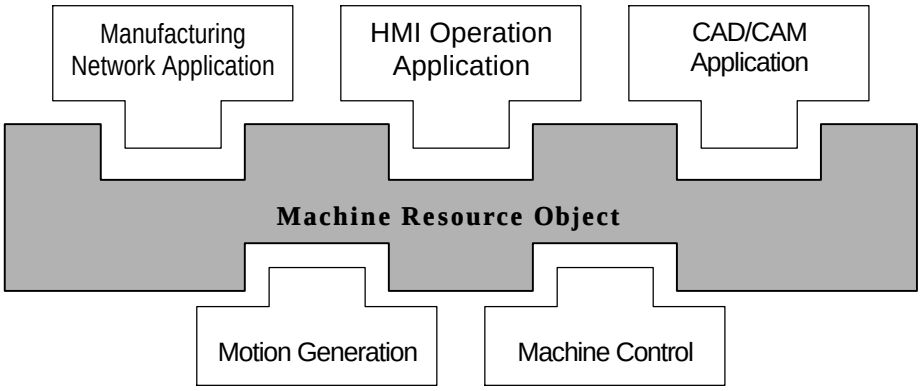

Figure 4-3 Resources Management around MRO

(1) Summary of Machine Resource Objects

To model the Machine Resource Objects, the controller's control functions are classified and modeled into several function groups called "Domain". The domains are defined as following.

- Basic Function Domain

  Basic function domain defines the classes of machine tool and alarm.

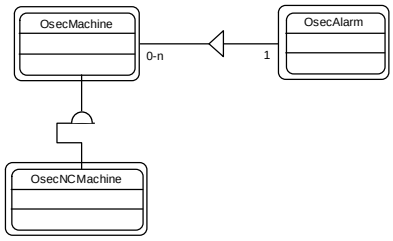

Figure 4-4 Basic Function Domain Object Model (Cord Yourdon Notation)

- Operation Domain

  Operation domain provides the operation interface to machine tools. Many of objects included in this domain are derived from modeled machine parts such as movable axis, spindle, etc.

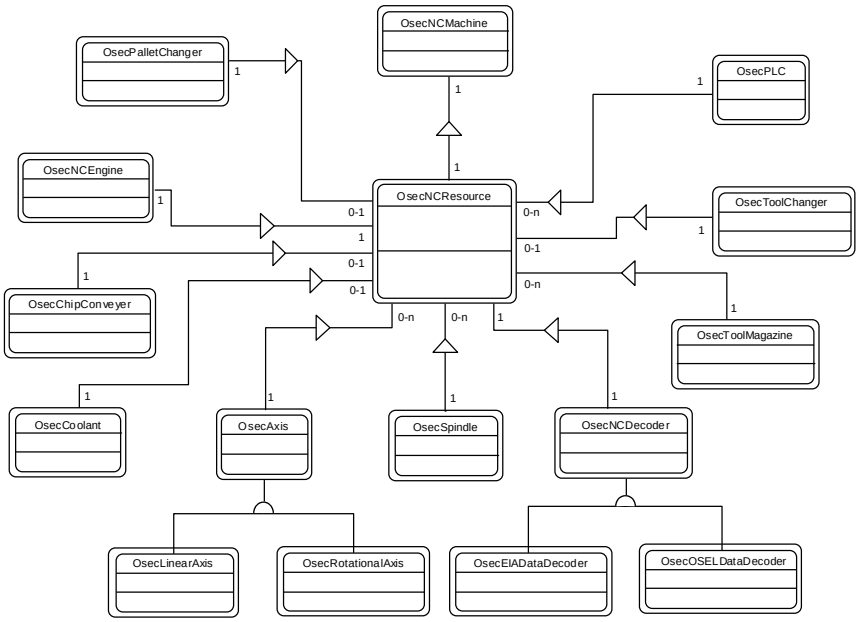

Figure 4-5 Operation Domain Object Model

- Control Service Domain

Control Service Domain connects Machine Resource Objects to the controller's control functions. In OSEC-II, a function block called "Resources Control " is provided to do the mapping from Machine Resource Objects to motion generation function groups. This resources control function can be considered as an implementation form of control service domain.

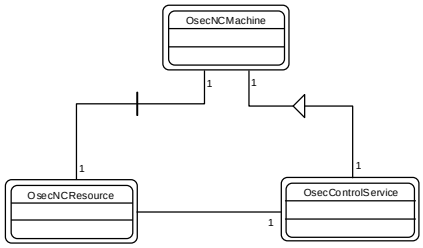

Figure 4-6 Control Service Domain Object Model

(2) Implementation

HMI and network applications' standardization is implemented by connecting to the standardized Machine Resource Objects.

Nevertheless, to actual control systems, there are various forms for the implementation of machine control and motion generation, so an identical definition is rather difficult.

Considering this, we call the section which reflects the feedback information like current positions etc., the instruction information like execution commands, destination position etc. and the information on machine's operation state to the Machine Resource Objects, converts the Machine Resource Objects' messages like program selection, language processor's start/stop, manual operations etc. occurring from HMI into instructions to motion generation and machine control, as Resources Control.

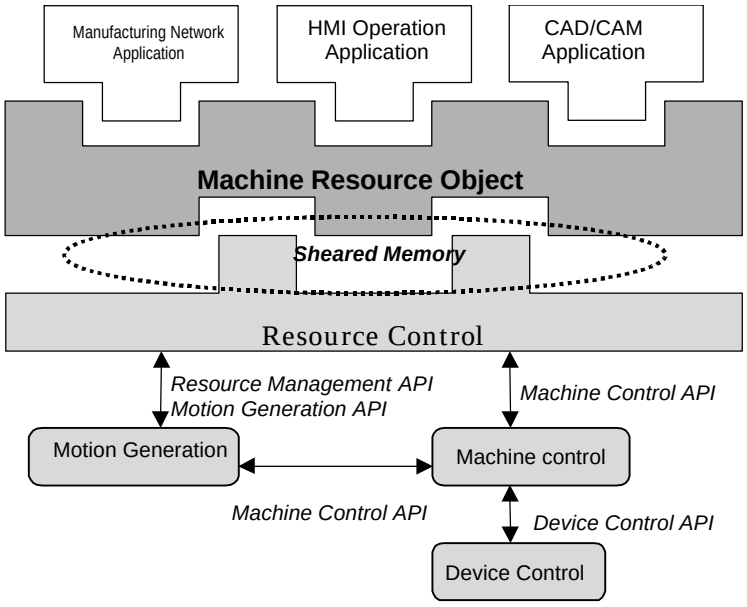

Figure 4-7 Resource Controll Implemention on Shared Memory and APIs

The connection to Machine Resource Objects, can be implemented as the interface to all object models, but the interface to machine control, motion generation of actual control systems is not compatible with object models. The resources control is installed as a section which makes the resources management of Machine Resource Objects possible by connecting object models to machine resources models and connecting to motion generation, machine control by static API.

■Machine Resource Objects

C++ is selected as the programming language, using IBM Japan's development tools for C++. The Machine Resource Objects is installed as a software part utilizable within the development tools. Using the software parts provided by the Machine Resource Objects, the function model for the target machine tool can be constructed.

■Virtual NC Card

Virtual NC Card (VNCCard) implements the interface functions to the resources management layer through common memory. It converts commands issued from each object of the Machine Resource Objects into command interface in common memory, reads various data for control functions from the periodical watching data section in common memory and write attribute members of the corresponding object among the Machine Resource Objects.

■Common Memory & Resources Control

Common Memory serve as the interface between virtual NC card (VNCCard) and resources control functions. 2 blocks of common memory are available for each communication direction respectively, two types of data transactions are carried out as following

- Data, like commands or alarms, which must be transferred certainly between the Machine Resource Objects and the resources functions.
- Data, like current positions of moving axes, which need only to be watched at necessary time. These data are mapped to the periodical watching data section.

■Resources Management API

API to the Machine Resource Objects through common memory, serving as the

interface to the object model, makes the access to resources and attributes of the Machine Resource Objects possible.

(3) Investigation Subjects
a. OSEC-II's object model is a definition to the static structure of system. To decide the system specifications, there is also a necessity of defining the dynamic structure of system additionally.
b. The Machine Resource Objects is a model expected to adaptable to all machine tools, but only the adaptability to machine centers and 3-Dimension measurers are verified through an actually implemented system. The adaptability of this model to machines different from machine center, such as lathe or laser cutting machine, also needs to be verified .

4.3 Motion Generation

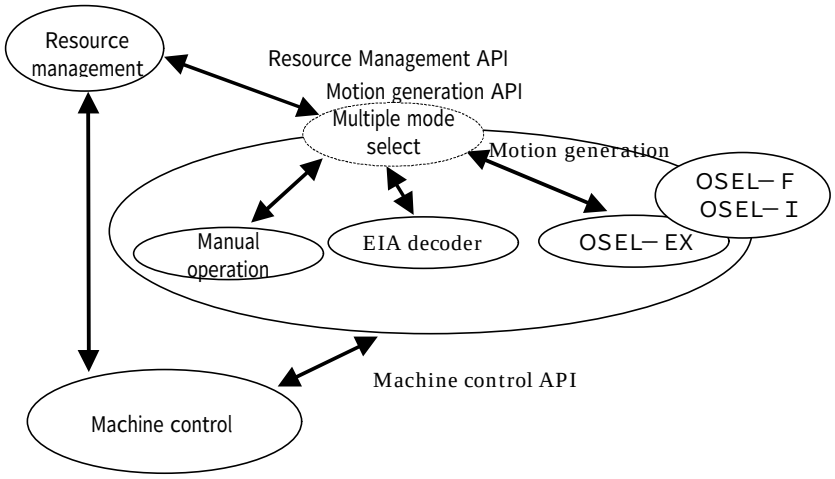

Figure 4-8 Resources Managemerment, Motion Generation, Machine Control, and APIs

The motion-generation function group, interprets processing-related instructions such as program operations, manual operations, etc. from the resources manager, generates processing strategies and then converts it into consecutive instructions to machine control functions. The instructions from the resources manager means CNC processing programs, manual operation instructions. The instructions to machine control functions handles motion instructions such as line, arc processing or coordinate operations, etc. The processing programs are considered as compatible with multiple languages, so it is also possible to have multiple language interpreter installed.

(1) Motion Generation API

API managing the execution of motion-generation programs with the functions of initialization, starting/stopping/restarting program, setting program name, etc. The motion-generation section generates the actual motions for machine tools through machine control API after receiving commands from the resources manager. For the instructions to all language interpreters are processed through this API, the implementation of multiple language interpreters can be fulfilled through a single API. According to the attributes of the designated work program, a proper language interpreter will be selected and activated. OSEL language and EIA are implemented this time.

(2) OSEL-Decoder（OSEL-EX）

API managing the execution of OSEL programs with the functions of initializing OSEL decoder, starting/stopping/restarting program, setting program name, etc. It will be called as motion-generation API when OSEL program is executed.

(3) EIA-Decoder

API managing the execution of EIA programs with the functions of initializing EIA decoder, starting/stopping/restarting program, setting program name, etc. It will be called as motion-generation API when EIA program is executed.

4.4 Machine Control

4.4.1 Function

Machine Control generates driving commands to servo motors like X,Y,Z axis or spindle, discrete commands to I/O units like valve or lamp, by operational commands from the upper layer. Still more, This module gets the status of a servo system which is the position or speed or angle data and so on, the information of PLC which is on/off data of I/O units and the result of relay ladder logic. Their status and information are used in Machine Control module itself for synchronization processing and transferred to upper modules.

4.4.2 Aims and Policy

In the phase of the construction of prototyping system after the decision of system architecture and interfaces of the function groups, it is most right way to implement the functional module in PC as software from a open oriented point of view. Its development was considered that this software modules would be the model for the future because of there was no public example to be referred.

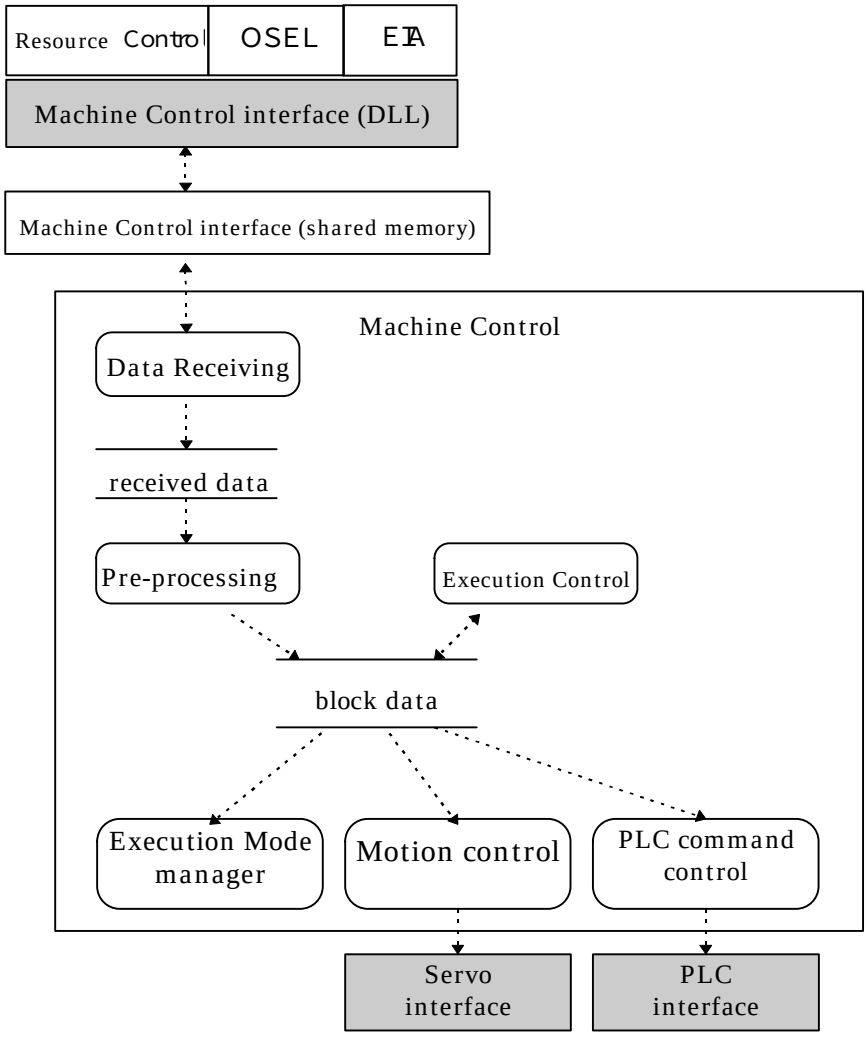

Figure 4-9 Example of Machine Control

The real-time ability of general Operating System for PC is some ten milliseconds

at best. While the real-time ability for position feedback loop of servo system need some milliseconds. There is the gap not to fill up between them. There are several ideas about the functional division for Machine Control and Device Control module. Because of the above reason, it will be the best way that the generation of motion commands as position or speed control is in Machine Control module, and the feedback loop control is in Device Control module. And because of the difference of the real-time ability of Machine Control module and servo devices, it is difficult to synchronize them completely. To cancel the influence of this difference, buffers are provided between Machine Control module and servo devices.

4.4.3 Internal Make-Up

The inside of Machine Control is made of 4 functions as below.

(1) an interface of upper layer
(2) generation of data for servo system and its interface
(3) generation of data for PLC system and its interface
(4) synchronization of servo and PLC

Figure 4-9 is an example of implementation.

4.4.4 Machine Control API

Machine Control API is an interface specification of Machine Control module. They are classified in six groups.

(1) Machine Control command
(2) Mode Control command
(3) Axis Control command
(4) DI/DO Control command
(5) Get information command
(6) Alarm command

4.4.5 Problems

From the point of view of functional classification, it is necessary to construct an architecture which can build in an interpolation process and acceleration/deceleration patterns, and to make module of them. Still more, it is necessary to construct an architecture which can make a subsystem of machine motion like turning of arm and spindle control in ATC (Automatic Tool Changing) motion.

From the point of view of compatibility, it is necessary to verify to handle servo systems or PLC systems with common interfaces. And it is necessary to apply the other machine except machining center.

4.5 Device Control

In a Device Control, the control information in the computer is converted to real world information for the machine tool equipment. In this specification, we specified the specification for the servo and PLC devices.
This specification is designed with the following strategy.

(a) Specification Level

We specify the interface not at the H/W level but the function level, i.e. , at C or C++ language level. We specify only the actual parts required to handle various devices, and we allow specialized tools and settings for the parameters and the development/debugging environment.

(b) Real-time

It is impossible for the personal computer with windowing operating systems to guarantee real-time functions, such as servo control, as in some real-time operating system. The Device Control part has the connection function between the soft real-time world on the personal computer and the hard real-time world on the control target such as the servo system.

(c) Multi-Vendor

This specification allows a multi-vendor environment, which is important when we want to optimize the assembly of our controller.

4.5.1 Servo Device

(1) General Description

Many operating systems running on PCs, especially windowing systems, do not have real-time functions sufficient for controlling servo motors or other servo devices directly. This specification covers the APIs of the real-time servo control part, including current , speed, and position feedback loop. The modules will be supported by device vendors.

(2) Model

The OSEC servo device model has three layers: a servo axis, a servo group, and servo interface as shown in Figure 4-10.

(3) API Functions

The OSEC servo device API has following functions.

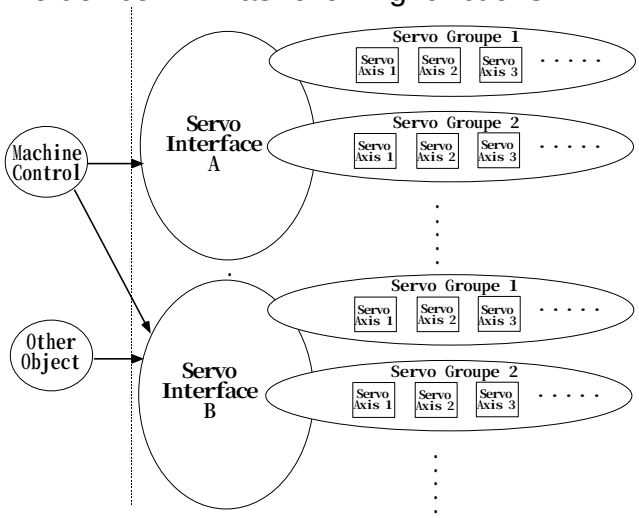

Figure 4-10 OSEC Servo Device Model

(a) Servo Interface Management : Connection between other modules, initialization

and configuration management functions with this specification.

(b) Servo Group Management : Initialization and configuration management of each servo group.

(c) Servo Axis Management : Initialization and configuration management of each axis.

(d) Servo Control : Controls servo devices. It contains cyclic servo commands, feedback, synchronization and some functions for servo control such as SERVO-ON/OFF.

(4) Future Problems

In the next step we will investigate the following themes.

(a) Multi-Vendor Interface : The interface specification for using materials made by different vendors will be discussed.

(b) Multi-Function Command : OSEC-II specification supports only cyclic commands for servo control. If we want advanced control functions such as motor torque control or accelerate/decelerate pattern control, we must add some upper level control interface.

4.5.2 PLC Device

(1) General Description

PLCs (Programmable Logic Controller) are generally used to control the sequence logic of peripheral devices in our machine tools. This specification provides the PLC I/O access functions. From the control program compatibility point of view, the OSEC PLC device manages the I/O data not as contact number but as a symbolic name.

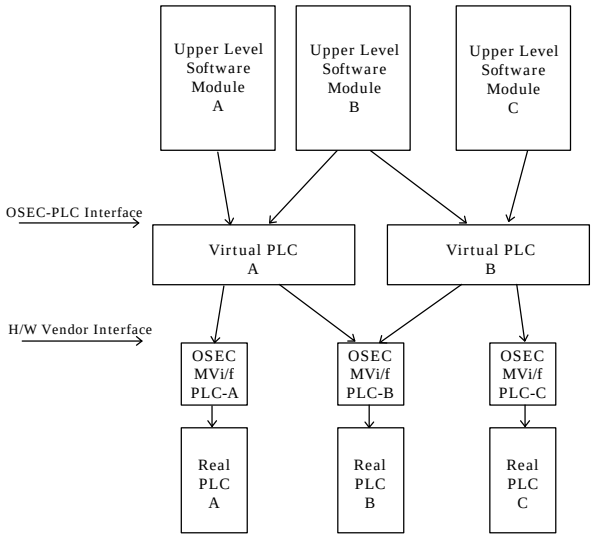

Figure 4-11 OSEC PLC Device Model

(2) Model

The OSEC model is shown in Figure 4-11. The introduction of a logical PLC guarantees independence from physical PLCs.

(3) API Functions

This API is the interface specification for using PLCs from upper level software modules.

(a) PLC Interface Management : Contains interface functions including connections to upper modules and management of some PLC parameters.

(b) State Management : Manages the states of PLC.

(c) PLC Data Control : Executes data management, access control, and synchronization functions.

(4) Future Problems

In the next step we will investigate the following themes.

(a) Multi-Vendor Interface : The interface specification for using materials made by different vendors will be studied.

(b) PLC Data Identifiers : Standard symbols and identifiers for PLC data will be studied.

(c) PLC Programming/Maintenance Environment : This specification doesn't yet provide the programming/maintenance environment.

5 OSEL

5.1 Objective

OSEL (OSE Language) provides an open framework for describing machining know-how related to processing as a set of software components. The know-how on processing is generated and saved by organizations and workers related to the machine tool industry at the levels of engineering documentation and technical skills. The processing know-how can be captured as an object, and we focus on the accumulation of the processing know-how through the construction of class libraries. If a different material or processing method is used, or processing conditions differ, a user will be able to eliminate an unnecessary accumulated set of libraries, replacing it with a new processing library. Other libraries will exist somewhere on the network, and will be available, for substitution if necessary.

We will construct a framework as a numerical control(NC) language, and will focus on establishing an interface among functional modules in the OSEC architecture. Here we will not propose a whole set of processing class libraries, which would expand in every direction. We believe that it is the duty of manufacturers who are interested in providing and realizing processing libraries to do so. Instead, we will create a demonstration version, to show the validity of the framework and the interface.

5.2 Background

5.2.1 Problems of Present NC Language and Language Handling

Today, the format of NC data used to control machine tools is prescribed by the ISO, and is concise descriptive. However, since this ISO standard was established over 20 years ago, it is difficult to say that is adequately reflects the performance of current computers, which have brought about remarkable engineering innovations. Furthermore, there is a tendency nowadays to use macros, in order to reduce the amount of input data, and also to use complicated functions added by individual CNC manufacturers, As a result, it has been becoming more difficult to guarantee interchangeability. Some of the current problems regadrding NC data are as follows:

- The NC data specification was defined when memory was still expensive, in such a way that a set of NC data must be described by using a limited number of characters. Thus, the description is not always easy to read.
- NC data does not contain functions that can describe free form curves and free form surfaces. Therefore, the movements for these shapes have been transformed into many small linear movements.
- NC data is normally described on several levels: there is a complex operation description and a simple operation description for use in, say, compound macros and linear movement, respectively.
- Custom macros are functions expanded by CNC manufacturers, and they have no compatibility with macros created by other manufacturers. In addition, they include components that depend on the machine tool type, and identical NC data cannot be used with to other machine types.
- The dependence on the machine tool type is strong, and the flexibility and expandability of the descriptions are poor. Consequently, it is difficult to construct libraries.

In addition to these problems, the growing expectation of an opening up in the area of FA control equipment has led to demands for ease of integration among CAD/CAM systems,

database systems, and other information-processing tools. If these demands are met, it would be easy to construct an adequate system by integrating software packages and software components provided by multiple vendors.

5.2.2 Required Functions

For the above reasons, the following features are required in data used to control machine tools:

1. To provide flexibility and expandability in the description:
   - The language used to describe the operation of machine tools should be a script-type language that is easy to understand.
   - A direct specification by numerical equations of free-form curves and free-form surfaces should be achieved. Defining these curves and surfaces, such as Bezier function, before processing will reduce the amount of data required give a better process quality than the conventional approach, which uses a set of NC data points.
   - The ability to use libraries of macro functions for expansion should be considered.
2. To increase the abstraction level and generality of machine tools, and to reduce the dependence on machine types:
   - The data should be able to handle information on machine-type dependence in abstraction at the operation description level, such as the specifications of machine tools (accuracy, speed, etc.), tools for tool handling, jigs for workpiece offset, and so forth.
   - ASCII description should be used, so that components depending on computational hardware and operating systems can be eliminated.
   - The movement description level should be unified to a primitive one, so that dependence on machine types can be reduced.
3. To ensure compatibility with existing data:
   - Compatibility with existing data should be ensured by preparing an NC data decoder to convert existing data described in G-code and/or CL data.

5.2.3 Details

In OSEC-I, a process program for the operation control was interpreted and then forwarded in order to control commands as shown in Figure 5-1. The sequence control is considered to be included in the process.

In OSEC-II, on the other hand, the language name is changed to OSEL, and the positioning of the process is also changed. The operation control in OSEC-I gives the impression that it controls the whole NC. In OSEC-II, however, a "resource management" part is added, which monitors and controls all the resources of an NC. Hence, OSEL can describe the characteristics of process, and it is now positioned in such a way that it concentrates on generating a tool path and process conditions based on its description.

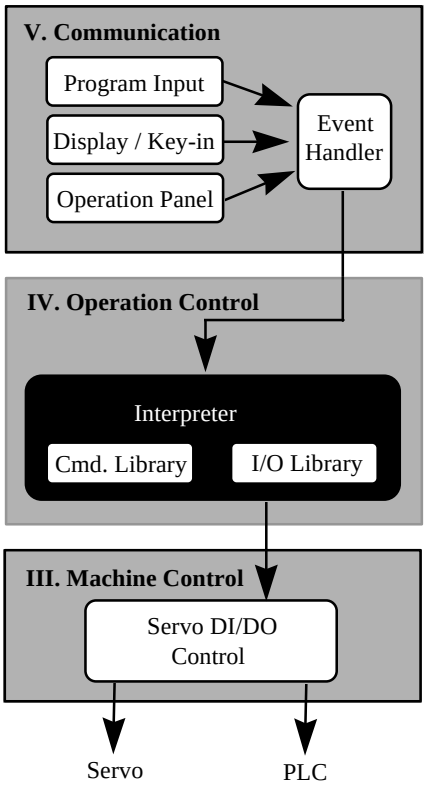

Figure 5-1 Positioning of the Machinery Description Language for OSEC-I

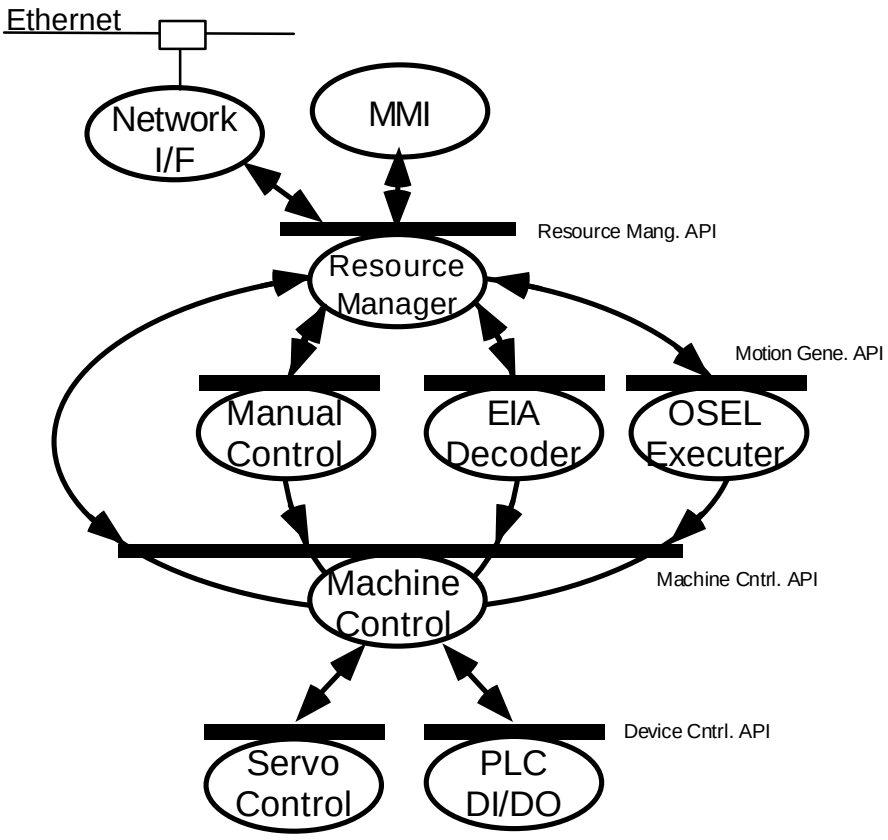

Figure 5-2 Positioning of OSEL Based on OSEC-II

5.2.4 Scope of OSEL

In the form of OSEL being investigated for implementation in OSEC-II, the process shape, process material, and process order are assumed to be defined. The scope of OSEL will handle everything from this information up to outputting to NC controls, among other functional modules that would be required for the processes, tool paths, and process conditions (Figure 5-3).

It will be possible to separate the language processor from machine tools as a software component, and to exchange it, if necessary. The process class libraries will be also considered to be exchangeable with corresponding specified processes. Hence, it will be possible to configure the OSEL processor in response to performance improvements in hardware, advances in tools, and variations in process materials.

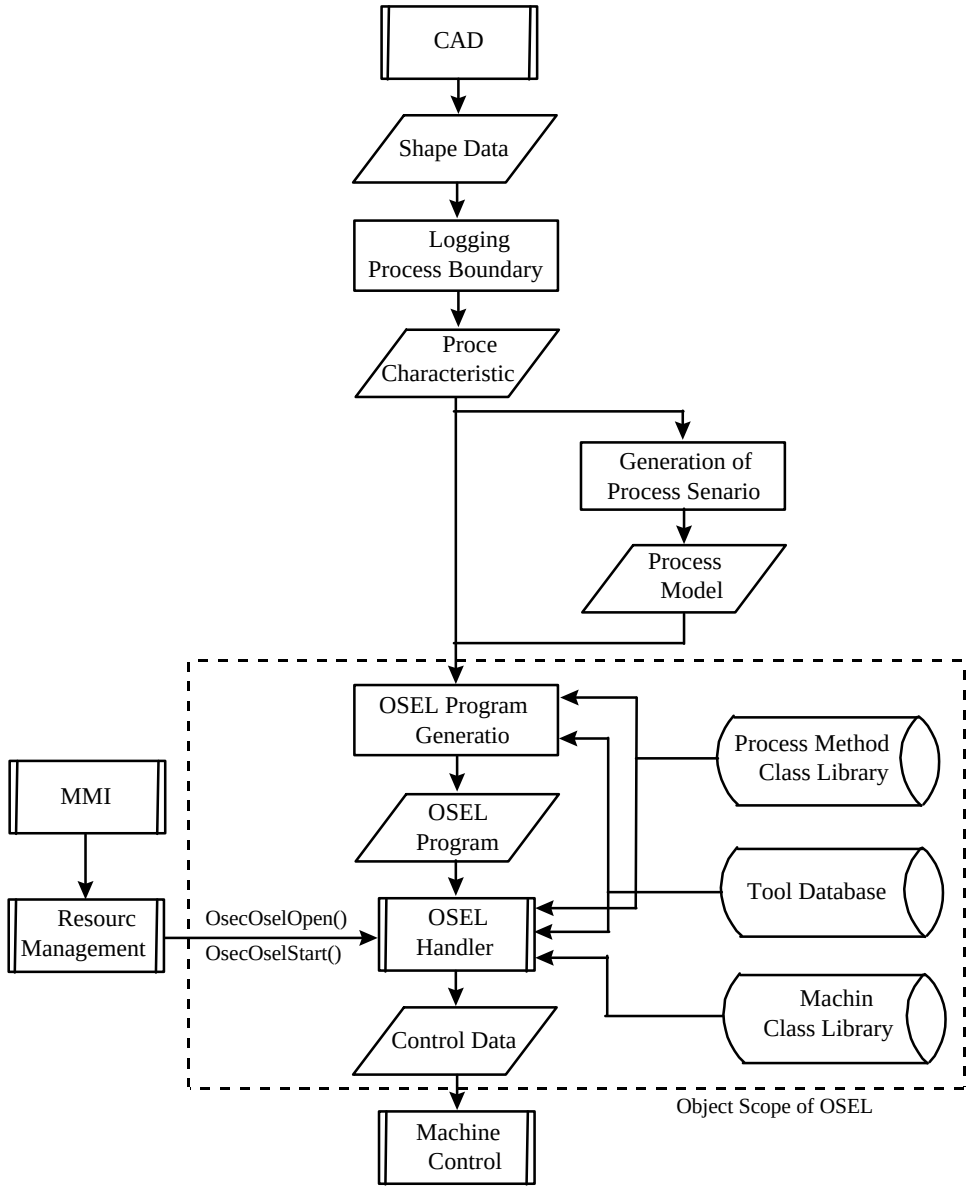

Figure 5-3 Handling Scope of OSEL

5.3 Concept

A computer language system for NC programming should consist of at least two parts: a language for programmers, in which they describe their machining intentions, and a control command language, which is interpreted by NC machine controllers to yield step-by- step machine actions.

The former of these should be able to define the objects to be machined and the method of machining. When a programmer gives these definitions, he/she may want to concentrate on the description of the shape to be made and the precision of machining, for example, rather than the details of machining, such as the assignment of an actual machine to be used. Moreover, the written program is required to be executable for various actual machine tool selections and in various conditions. Accordingly, it is desirable that the program should describe the shapes of the objects and the method of machining, but not the control details such as the step-by-step machining sequence or the actual tool id.

On the other hand, the control command language should address the individual machine on which the machining is to be executed, because this language is expected to be directly executable by an actual machine controller. Typically, the language should be able to optimize the sequence of machine actions, and to address actual tools in the magazine, tooling axes, the type of tool path interpolation, the possible feed rate and spindle rate, and the control of auxiliary mechanisms. Of course, the syntax of the program should be directly parsed and executed by machine controllers.

In the OSEL language system, the description language is named OSEL-F (Feature) and the executable language is named OSEL-X (eXecutable). In most cases, a language-processing system can be developed by using these two languages as input and output language, respectively. However, if only these two languages are specified, the system may become quite rigid, and may not be flexible enough, for example to create several different executable language programs from a single descriptive language program. To ensure this kind of multi-executable-language flexibility, an intermediate language called OSEL-I is introduced as a pivot language. It is mostly for the use of the language-processing system, and may not be visible for end users. However, it is an essential mechanism for realizing the flexibility and commonality in OSEL language systems.

The OSEL language system is preferably realized as an assembly of functional components so that the system can keep up with variations and changes in machines, machining methods, tools, and so on. In the OSEC-II project, object-oriented language technology was tested to determine its usability for this purpose.

Fig. 5.4 shows an overview of the language processing.

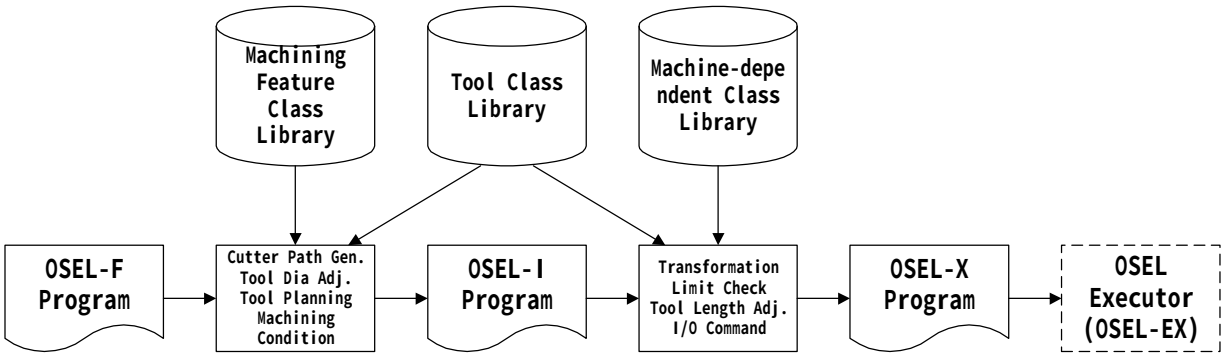

Figure 5-4 Overview of OSEL processing

OSEL-F, which is the top-level language, is used to describe machining features corresponding to machined geometry such as "make a hole with a drill." OSEL-F is transformed into OSEL-I, which is an intermediate level, independent of machine tools, by using a machining feature class library and a tool class library. OSEL-I in turn is transformed into OSEL-X, which depends on the a machine tool and is executable, by using a machine-dependent class library.

5.4 OSEL-F

This chapter explains OSEL-F, which is a top-level description language. Since the framework of a programming language is subject to that of Java, the portion proper to OSEL is described here.

The following are described in an OSEL-F machining program:

- Definitions of machining features, which are objects of machining
- Designation of the machining methods performed to machining features
- Designation of a machining sequence

Machining features are definitions of features such through holes or grooves to be machined. Machining feature objects are created as objects of machining feature classes such as the through hole class and the groove class in the machining feature class library described later. An object's position with respect to a workpiece, its geometry such as the diameter or the depth of a through hole class, and geometric tolerance information are defined as its attributes. Subject to a form of the Java language, a conceptual object creation description is as follows:

```
MachiningFeature machining_feature_object = new MachiningFeature(attributes);
```

A machining method is designated by calling a machining method of a defined machining feature object. Machining methods are defined in each machining feature as object methods. A conceptual call is described as follows:

```
machining_feature_object . machining_method(variable_for_machining, ...)
```

Machining methods like these are called in accordance with a machining sequence.

A machining program written in OSEL-F for drilling is shown in Figure 5-5. In this example, some parts are omitted for ease of understanding

```
public class Sample {                                    // A sample program
    public static void main(String args[]) {
        //
        // Position, diameter, depth are defined here.
        //

        ThroughHole hole = new ThroughHole(pos, dia, depth);// Definitions of the machining
feature Through Hole"
        Hole.centerdrilling();
        Hole.drilling();
    }
}
```

Figure 5-5 Sample of OSEL-F Machining Program

5.5 Machining Feature Class Library

In this chapter, a machining feature base class library and a geometry base class library are explained as OSEL specifications. A user can define his/her own derived classes and methods in accordance with the Java language syntax. Machining feature objects have machined geometry and tolerance information, and define machining methods as their methods. There are geometry objects for geometry representation and geometric calculation, and vector objects for three-dimensional coordinates and calculation of coordinates and vectors. Tolerance objects have not been adequately discussed as of the time of writing. The geometry is defined at the time a machining feature object is created. When a machining method is called, a geometry object associated with the machining feature object generates a cutter path for machining. The machining feature object then adds approach/departure motions of a tool to the cutter path, sets machining conditions with reference to a tool object, calls methods of a machining tool object for controlling axes and outputs data. The relationship of method-calling among objects is shown in Figure 5-6.

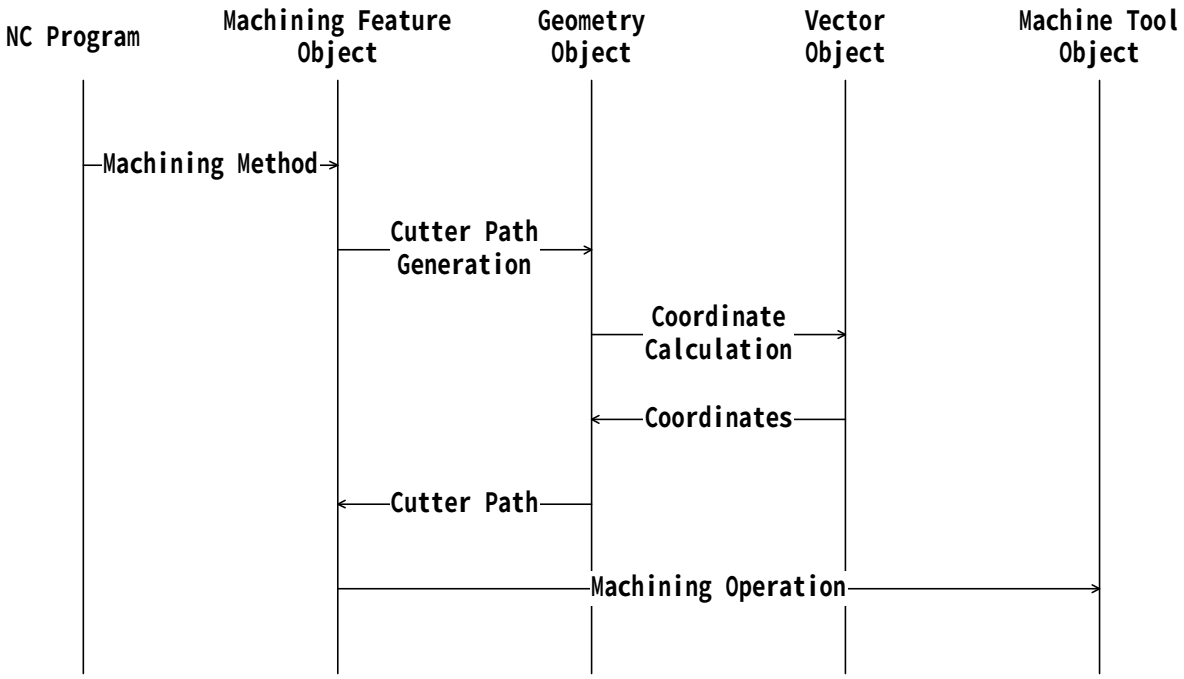

Figure 5-6 Relationship of Method Calling among Objects

5.5.1 Machining Feature Class Overview

Machining feature classes are categorized into three groups: holes, profiles, and surfaces. The attributes of the classes are the geometry objects of the machining features and tolerances, and the methods of the classes are machining methods. The overall structure of the classes is shown in Figure 5-7.

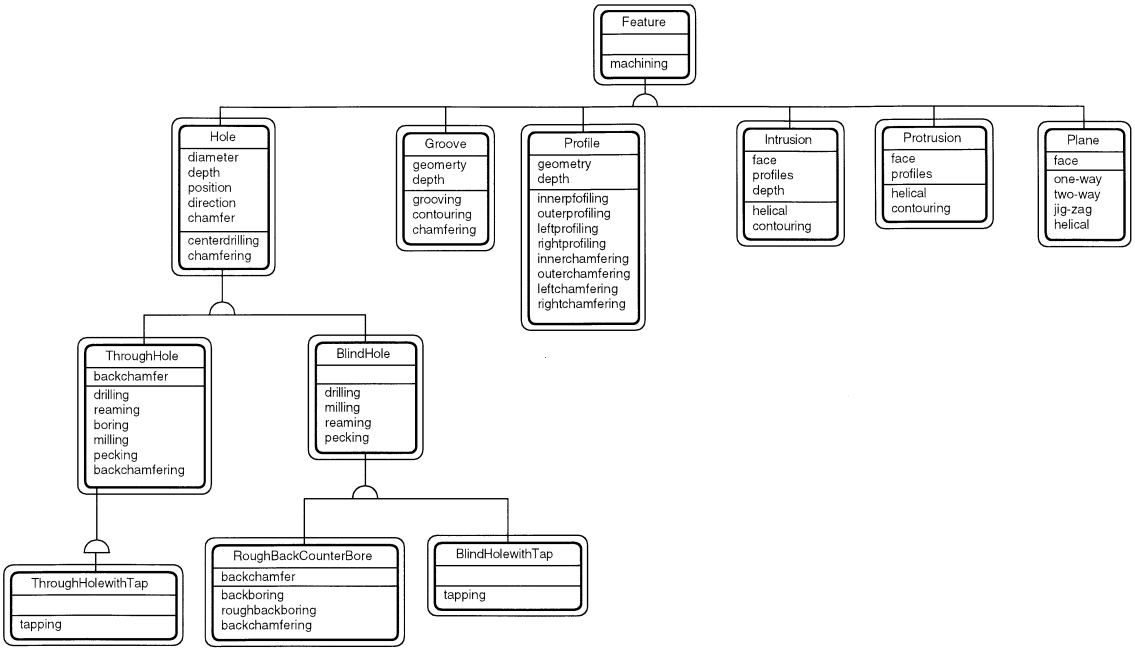

Figure 5-7 Hierarchy of Machining Feature Class

5.5.2 Geometry Related Class Library

This is a library of classes for geometric representation and its calculation. Attributes of the classes are parameters for geometric representation and methods of them are calculation of cutter paths. Overall structure of the classes are shown in Figure 5-8.

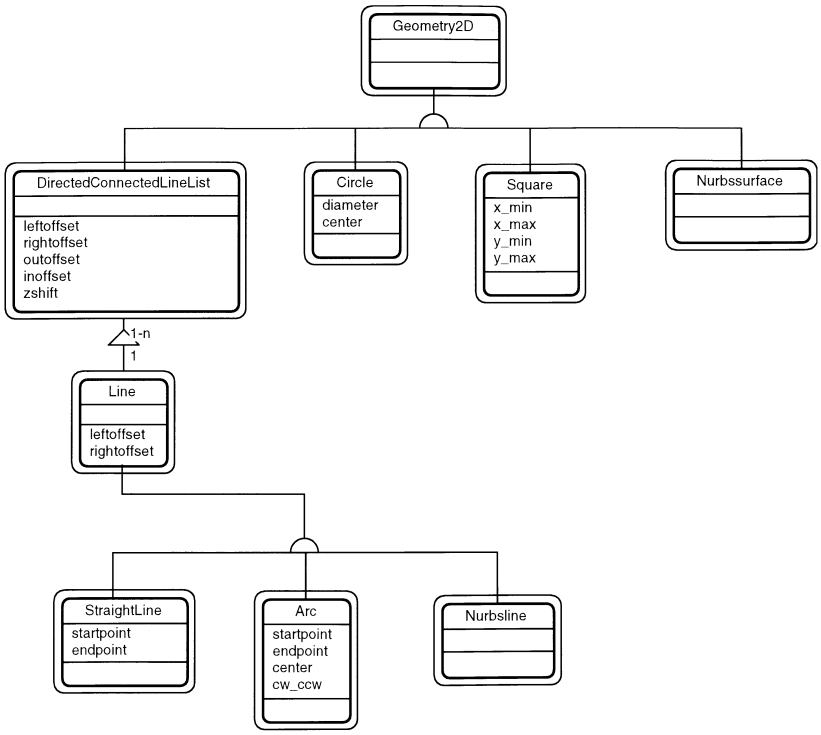

Figure 5-8 Hierarchy of Geometric Representation Classes

5.5.3 Sample Code

```
public class ThroughHole extends Hole { //Definition of machining feature "Through Hole"
   Vect3 position;        //attribute: position
   double diameter;       //attribute: diameter
   double depth;          //attribute: depth
   public ThroughHole(Vect3 pos, double dia, double d) {    //constructor of "through hole"
   position = pos;
   diameter = dia;
   depth = d;
   }

    public void drilling() {                    //Definition of drilling method
    Tool tool = toolfordrilling(diameter);    //Selection of tool for drilling
    machining_center.toolchange(tool);    //Tool change
    Vect3 initial = getinitial();                    //Initial point
    Vect3 reference = getreference();      //Reference point
    Vect3 target = gettarget();     //Target position of through hole
    machining_center.toolmove(initial);              //Move to initial point
    machining_center.toolmove(reference);          //Move to reference point
```

```
machining_center.spindle_on(Osel.CW);      //Spindle on (clockwise)
machining_center.coolant_on();                    //Coolant on
machining_center.toolline(target);         //Machining to target point
machining_center.toolmove(reference);      //Return to reference point
machining_center.spindle_off();                   //Spindle off
machining_center.coolant_off();                   //Coolant off
}
```

5.6 Tool Class Library

5.6.1 Outline of the tool class library

A tool class library is a library of tools that is referred to by a machining feature class library or a machine tool class library. The machine feature class library refers to the tool class library to obtain the name of the tool that is actually installed in the ATC tool station. The machine tool class library refers to the tool class library to identify the tool size, geometry, and machining conditions selecting tools whose characteristics match those of the machining feature.

The main difference between the data in the tool class library and those in general tool catalogues is the tooling dimensions. The tooling dimensions given in the tool class library are the dimensions in actual machining conditions, such as the dimensions of tools with chuck and tool holders. Machining conditions can be modified in accordance with the actual operation experience, and this experiences can also be registered in the library.

The tool class library can also be extended when new tools are introduced into the machining system. The tool class library itself is described in the object-oriented language: Jave. This means that a new tool class having a function that generates an optimum cutting condition subject to the latest cutting theory can also be easily constructed in the library. In this case, a user only has to add the new functions to the most similar existing tool class. However, user may want to generate a new machining feature class library based on the existing library when he/she uses a completely new type of tool class.

5.6.2 Future focus

The data stored in the tool class library cannot describe all tool dimensions. At present, it roughly describes only three dimensions such as an edge, a shank, and a holder. Because of this limitation, it is impossible to calculate a perfect tool collision problem. It is still possible to apply a rough collision check by using the maximum area except for the tool edge diameter. If a three-dimensional tool feature is registered in the library, it becomes possible to realize a more precise collision check. The library may be able to manage tool wear compensation values. These values can also be referred to by other class libraries, and can be applied to real-time tool wear control and a forcast of tool changing time.

5.6.3 Specification of the tool class library

The specification of the tool class library is as follows. This is a temporary specification, and will be modified and extended in the near future.

```
package osel;
import java.util.*;
import java.awt.*;
import java.io.*;
```

```
public class Tool {
// Fields
                    // Names
          protected String toolName;              Tool name (tool ID)
          protected String toolCode;
                    // tool size
          protected double toolLength;            Distance from holder reference (taper shank
reference) to tool edge
          protected double availableLength;       Effective tool length
          protected double diameter;              Machining hole diameter
          protected double pointAngle;            Angle between workpiece and tool edge
          protected double shankDiameter;         Tool shank diameter
          protected double shankLength;           Tool shank length
          protected double holderDiameter;        Tool holder diameter
          protected double holderLength;          Distance from holder reference to the end-point of tool
holder
          protected int teeth;                    Number of tool teeth
          protected double supplementToolLength;  Compensation for Z-axis length error caused by tool
length
          protected double supplementDiameter;    Compensation for tool diameter
protected double freeFeedClearance;               Clearance for oblique approach
          protected double rapidFeedClearance;    Clearance for Z-axis direction approach
                    // Conditions
          protected String material;              Workpiece material
          protected double spindleSpeed;          Spindle speed (revolution)
          protected double axialFeedRate;axial direction feed rate
          protected double radialFeedRate;        Radial direction feed rate
          protected double roughAxialDepth;       Axial direction rough cutting depth
          protected double finishAxialDepth;      Axial direction fine cutting depth
          protected double roughRadialDepth;      Radial direction rough cutting depth
          protected double finishRadialDepth;     Axial direction fine cutting depth

                    // Constructors
          public Tool();
          public Tool(String tn);
          public Tool(String mt, String tn);
          public Tool(String mt, String tn, String tc);
                    mt          :Workpiece material
                    tn          :Tool name
                    tc          :Tool ID
          %In cases where the workpiece material is not specified, the highest priority material listed in the
cutting conditions selected

                    // Methods
                    // Sizes
          public String getToolName();            Get tool name
          public String getToolCode();            Get tool ID
          public double getToolLength();          Get tool length from holder reference (taper shank
reference) to tool edge
          public double getAvailLength();         Get effective tool length
          public double getDiameter();            Get tool diameter
          public double getPointAngle();          Get tool edge angle
          public double getShankDiameter();       Get tool shank diameter
          public double getShankLength();Get tool shank length
          public double getHolderDiameter();      Get tool holder diameter
          public double getHolderLength();        Get distance from tool shank reference to tool holder's
front end-point
          public int getTeeth();                  Get tool edge numbers
          public void setSuppLen(double l);       Set tool wear compensation value(Z axis direction)
          public double getSuppLen();             Get tool wear compensation value(Z axis direction)
```

```
        public void setSuppDia(double d);          Set tool wear compensation (radial direction)
        public double getSuppDia();                Get tool wear compensation (radial direction)
        public double getFreeFeedC();              Get clearance for oblique approach
        public double setRapidFeedC();  Get Z axis approach clearance
                // Conditions
        public String getMaterial();               Get workpiece material
        public double getSpindleSpeed();           Get spindle speed
        public double getAxialFeedRate();          Get axial direction feed rate
        public double getRadialFeedRate();         Get radial direction feed rate
        public double getRoughAxialDepth();        Get axial direction rough cut depth
        public double getFinishAxialDepth();       Get axial direction finish cut depth
        public double getRoughRadialDepth();Get radial direction rough cut depth
        public double getFinishRadialDepth();      Get radial direction finish cut depth
}
```

5.6.4 Example of a tool class library application

```
// example
package osel;
import java.util.*;
import java.awt.*;
import java.io.*;
import osel.*;

    // Extend the tool class and identify new end milling class created by the user
class EndmillEx extends Tool {
    // Identify the constructor when the tool name, tool ID, and machining method are specified
    public EndmillEx(String mt, String tn, String tc, int hc) {
        super(mt, tn, tc);                          // Call super-class constructor
        if ((tn.compareTo("D15") == 0) &&
          ((tc.compareTo("") == 0) || (tc.compareTo("EDM15") == 0))) {
                toolName = tn;                      // Tool name
                toolCode = "EDM15";     // Tool code (ID)
                        // Tool dimension
                toolLength = 210.5;       // Tool length
                availableLength = 65.0;  // Effective tool length
                diameter = 15;                      // Machining diameter
                pointAngle = 180.0;       // Tool point angle
                shankDiameter = 16.0;   // Tool shank diameter
                shankLength = 5.0;        // Tool shank length
                holderDiameter = 70.0;  // Tool holder diameter
                holderLength = 140.5;    // Tool holder length
                teeth = 2;                          // Tool edge number
                supplementToolLength = 0.0;// Z-axis offset error compensation
                supplementDiameter = 0.0;// Tool diameter error compensation
                        // Machining condition
                if    ((mt.compareTo("") == 0) || (mt.compareTo("S45C") == 0)) {
                        material = "S45C";              // Workpiece material
                        if (hc == 0) {
                                        // Slotting
                                        spindleSpeed = 560.0;// Spindle revolution
                                        axialFeedRate = 10.0;// Axial direction feed rate
                                        radialFeedRate = 40.0;// Radial direction feed rate
                                        roughAxialDepth = 7.5;// Axial direction rough cutting depth
                                        finishAxialDepth = 3.75;// Axial direction finish cutting depth
                                        roughRadialDepth = 15.0;// Radial direction rough cutting depth
                                        finishRadialDepth = 15.0;// Radial direction finish cutting depth
                        } else if (hc == 1) {
                                        // Side cut
                                        spindleSpeed = 560.0;// Spindle revolutions
```

```
                                    axialFeedRate = 190.0;// Axial direction feed rate
                                    radialFeedRate = 190.0;// Radial direction feed rate
                                    roughAxialDepth = 22.5;// Axial direction rough cutting depth
                                    finishAxialDepth = 11.25;// Axial direction finish cutting depth
                                    roughRadialDepth = 2.25;// Radial direction rough cutting depth
                                    finishRadialDepth = 0.75;// Radial direction finish cutting depth
                            }
                        }
                    }
                }
                    // Identify the constructor when the tool name is specified
                public EndmillEx(String tn) {
                    this("", tn, "", 0);                    // Call same-category constructor
                }
                    // Identify the constructor when tool name and machining method are specified
                public EndmillEx(String tn, int hc) {
                    this("", tn, "", hc);                   // Call same-category constructor
                }
                    // Identify the constructor when workpiece material and tool name are specified
                public EndmillEx(String mt, String tn) {
                    this(mt, tn, "", 0);                    // Call same-category constructor
                }
                    // Identify the constructor when workpiece material, tool name, and machining method are specified
                public EndmillEx(String mt, String tn, int hc) {
                    this(mt, tn, "", hc);// Call same category constructor
                }
                    // Identify the constuctor when workpiece material, tool name and tool ID are specified
                public EndmillEx(String mt, String tn, String tc) {
                    this(mt, tn, tc, 0);                    // Call same-category constructor
                }
}
```

5.7 Machine Class Library

The machine class library has the following functions:

1. Method for OSEL-X output
   - Output a command corresponding to MSTB control in EIA code
   - Output a command for axis movement control
2. Machine-dependent method
   - Limit check for axis stroke
   - Limit check for axis speed
   - Limit check for spindle speed
   - Wear compensation

The machine object called by the machining feature class library exchanges work coordinates with reference to a machine coordinate system. At this time, the library compensates for the tool length, checks the limit of axis stroke, the limit of axis speed, and the limit of spindle speed., and then outputs some commands MSTB and axis movement control. This class library transforms machining program written by using a machine-independent machining feature class library into actual machining control commands.

5.8 Other Standardization Activities

This section outlines TC184/SC1/WG7, which is one of the deliberation organizations of the ISO, and OPTIMAL, which is part of the ESPRIT project as concrete standardization activities.

5.8.1 OPTIMAL

OPTIMAL, which is short for "Optimized Preparation of Manufacturing Information with Multi-Level CAM-CNC Coupling," is part of ESPRIT project 8643. The participants are six companies and one research organization. The goal of OPTIMAL is to achieve consistent flow in CAD-CAM-CNC by means of a module structure, while maintaining the interface between the product geometric model data and the CNC device. Plans was made for the development of the programming systems with a modular structure and a data exchange interface between multiple data groups and the CNC device (Figure 5-9).

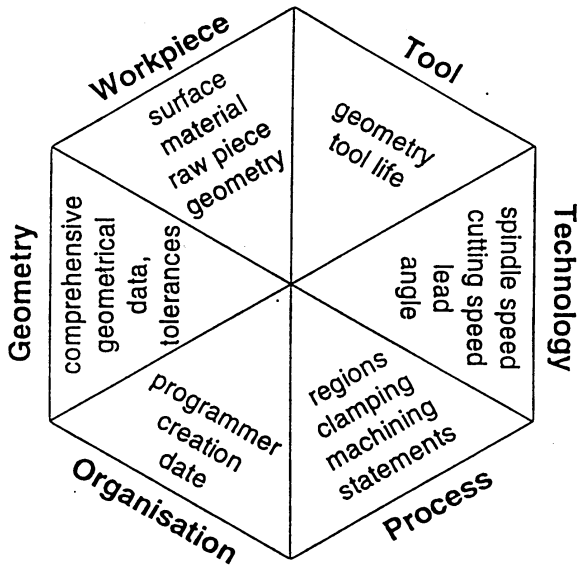

Figure 5-9 Data Groups in the OPTIMAL Interface
(The source : Definition of the Interface and the Interface Levels)

5.8.2 TC184/SC1/WG7

WG7 (data modeling for integration of physical devices) was formed under ISO/TC184/SC1 in November 1994. Using mainly the research results of ESPRIT, it is working toward standardization of the modeling of the data structure for CNC controllers and usage of the data.

The aims of the standardization are

- To multi-vendorize
- To establish a standardized technique from CAD data through NC programs
- To develop a new open NC language,

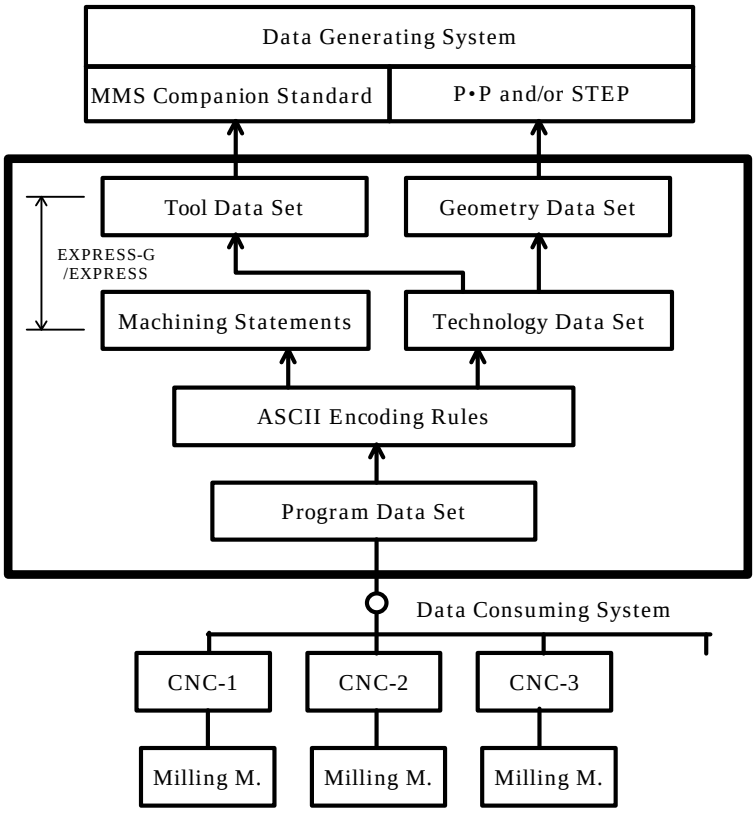

Figure 5-10 Working Concept of WG7
(Source: Japan Society of Mechanical Engineers, seminar text, No. 96-50 "Activities and Expectations for Open Architecture CNC" (in Japanese) p-10, '96-7-24,Tokyo)

Under the following conditions:

- The target controllers are CNC.
- The target machine is a milling machine tool.
- the target geometric model has 2-2.5 dimensions.

The group is considering the reference model of a CNC device constructed from four independent data-sets: - geometry, technology, tool, and program - described in EXPRESS-G. Product data and tool data described according to the international standards such as STEP are input to the data model (Figure 5-10).

The final output is an NC program that refers to the tool data-set and the technology data-set. The geometry data-set is further referred to by the technology data-set. The geometry data-set consists of data directly gathered from the STEP file. Each data-set except for the program data-set is described in EXPRESS. In the first stage, the group is working on confirming a static model. A dynamic model, which is necessary for real-time processes, will be the subject of the second step. The dynamic model is used to regenerate the machining process when obstructions (such as changes of geometry models, tools, machining processes, and machine tools) occur, as they surely will in practice.

References:

■Japan Society for Precision Engineering: 216th seminar text, "The long-awaited open architecture of CNC" (in Japanese)
■Japan Management Association: Proceedings of the Machine Tool Engineering Conference, July 1995
■Japan Society of Mechanical Engineers: seminar text No. 96-50, "Activities and Expectation to Open Architecture CNC, " p-10, 24 July 1996, Tokyo
■ESPRIT Project: Definition of the Interface and the Interface Levels, ESPRIT 8643, (1995).

5.9 Open Issues

In OSEC-II, a machining center with three axes is considered to be primary and holes, grooves, and surfaces are objects of processing. This is because it is currently important to validate the rationality of the OSEL concept in a simple case rather than in complex machining. There is a discussion of criteria for defining a hierarchical structure of a machining feature class. One criterion is to take account of actual cases in making, and another is to pay maximum consideration to expandability from the viewpoint of information processing. For example, there are three machining processes for a hole. One requires simple drilling, one requires profiling, and one requires pocket machining. If we take the position that machining features should be closely related to machining methods, there should be three types of machining feature. If we take information processing as our standpoint, there is a feature whose section is a circle and that is a subclass of concave geometry, and there are three methods for each machining method. This is based on the concept that maintainability is secured by an approach in which machining features are classified by geometry, and machining methods are defined as object methods of the features. The current OSEL adopts the former position, but the merits and demerits of these stand points have not been considered sufficiently thoroughly. Furthermore, it is necessary to think out a method whereby the two positions can coexist.

6 Prototyping Systems

We would define the OSEC specifications, including the function-group-based architecture, the function-block-based API, OSEL, and MRO, but we think the implementation should be let free. We suppose that personal computers are basically selected to implement the open controller and the FA equipment. The OSE exhibition demonstration systems in JIMTOF 1996 and their S/W and H/W modules are explained in the 5.1 and in the 5.2 section respectively.

6.1 OSE Demonstration Systems in JIMTOF 1996

The objective of this project is to define an open system architecture, which can be realized with present technologies. It is very important to implement the open control systems using the above system architecture and the OSEC-API in order to confirm their practicability and to discover any mistakes in the specifications. Through this implementation process, it can be also confirmed whether flexibility of combination with other systems will be achieved by replacing a conventional NC by a PC. For these purposes, the OSE partners developed and exhibited the following prototyping systems for JIMTOF, 1996. Through this experience, more concrete guidelines for open architecture controllers will be derived. The prototyping systems were connected to each other by a local network at the JIMTOF site, and also to world wide network by way of Internet (Figure 6-1).

- Station-A: Machining center consisting of the OSEC modules except a servo and a PLC board
- Station-B: Machining center consisting of an OSEC HMI module and conventional PC-based CNC
- Station-C: Instrument machine consisting of the full OSEC modules
- OSEL Station: Station for CAD/CAM integration using OSEL
- Network Station: Stations for "Internet Cyber Factory" networking OSEC control systems
- HMI Station: HMI stations developed with a MTF (Machine Tool Frame Work) tool

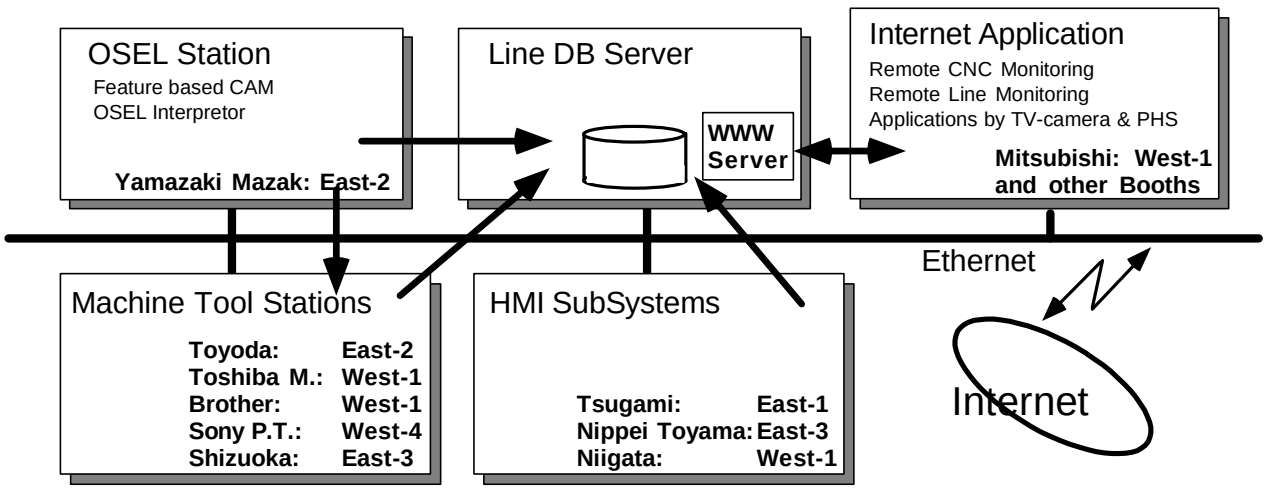

Figure 6-1 OSE Demonstration Systems in JIMTOF 1996

Figure 6-2 shows a comparison of implementation among Station-A, B, and C. The big three blocks describing Station-A, B, and C show the CNC basic parts and the shading and horizontal lines indicate the function blocks using the common API among the stations.

The shaded areas mean that their corresponding functions are implemented by additional control boards.
Station-A implements the device control function group with a CNC board and other function blocks with a PC-based software control. Station-B is a conventional PC-based CNC but uses an OSEC specification HMI module and a software-logic for monitoring. It has servo drives with SERCOS specification so that it is very close to a practical product. Two stations control the machine tools and can machine the workpieces. Station-C is a pure software CNC and uses the software-logic for a sequence control. It controls an instrument machine.

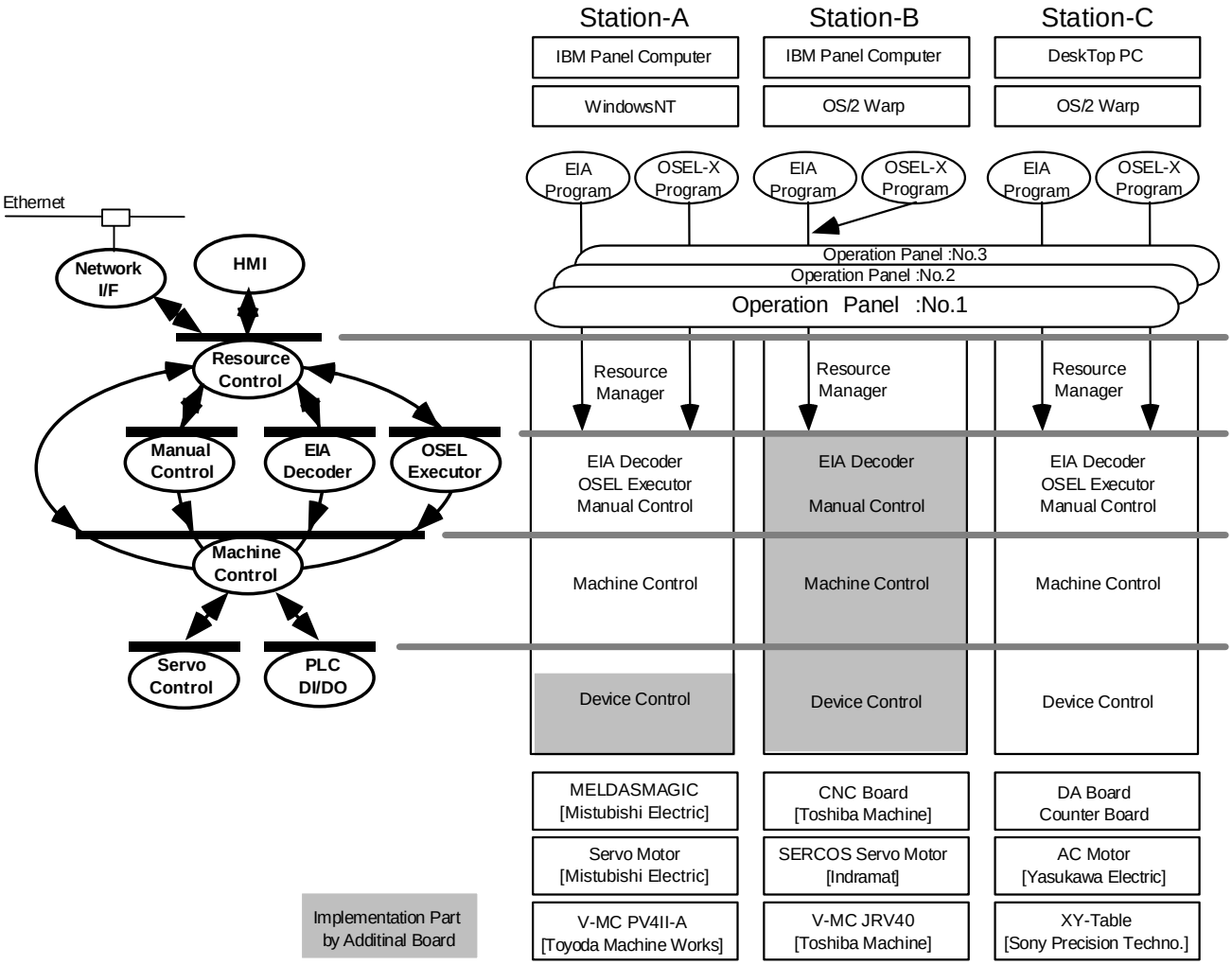

Figure 6-2 OSEC Implementation Model and Construction of Each Station

From the experience of the exhibition systems, we were able to prove the following possibilities:

- Software PC-based CNC, by implementing control functions with PC software and machining experiment using it
- Controller components by confirming compatibility and reusability of modules with OSEC specifications such as HMI and servo

- Construction of realtime control systems using only off-the-shelf parts
- Implementation of open controllers on malti-OSs and multi-platforms
- Description of userÕs machining know-how and its business on Internet
- Internet applications for open manufacturing systems at low cost and using only WWW and PC standard S/Ws

6.1.1 Station-A

Station-A has two targets.

■Constructing a machine tool using a PC-based controller, and cutting a work piece
■Making use of PCÕs resource as much as possible

Many of the controllerÕs function are made of the software modules of a PC, so necessary hardware to add to the PC should become very simple. Thus it will be possible to realize the open, modular and scaleable controller.

(1) Implementation of function module

The fruits of OSEC activity and off-the-shelf products were used in this controller as the function modules. This structure enables machine builders, users, and researchers to build in original functions to a controller.

Table 6-1 Platform Structure

PC	IBM-7344	Pentium (100MHz), HDD 500MB, RAM 32MB, Touch screen
Servo Drive System	MELDASMAGIC (Mitsubishi Electric)	
	Feed system (X,Y,Z)	Drive Units : MDS-A-V Type (Mitsubishi Electric) Motor : HA Type (Mitsubishi Electric)
	Spindle (SP)	Drive Units : MDS-A-SP Type (Mitsubishi Electric) Motor : Built-in Motor (Mitsubishi Electric)
I/O Units	Remote I/O : FCUA-DX (Mitsubishi Electric)	

Figure 6-3 Overall View of Station-A

(2) Conclusion

We achieved the above two targets in this Station-A. Not only ATC (Automatic Tool Change) and spindle control but also milling, drilling, and tapping was performed. Many machining methods were processed by a part program described by OSEL. And it was confirmed that many functions of controller could be performed by PC software. Some problems remain. The way to control in real time with certain by combinations of off-the-shelf products is needed. And the general way to send and receive a large quantity of information at high speed between NC and PLC is needed. Thus, tool selecting, tool changing, and interlock processing function, which needed NC and PLC connection, were carried out with low performance.

6.1.2 Station-B

At Station-B, a panel computer (IBM7344, OS/2) was attached to a vertical machining center (JRV40 Toshiba Machine) and PC-based CNC was constructed (Figure 6-4).

The following test items were selected for implementation.

■Reusability and sharability of a HMI module by using a MTF tool
■Connection of off-the-shelf servo drive systems through a SERCOS interface
■Connection of an OSEL program through an OSEL-EIA converter
■Real time NC data monitoring through network

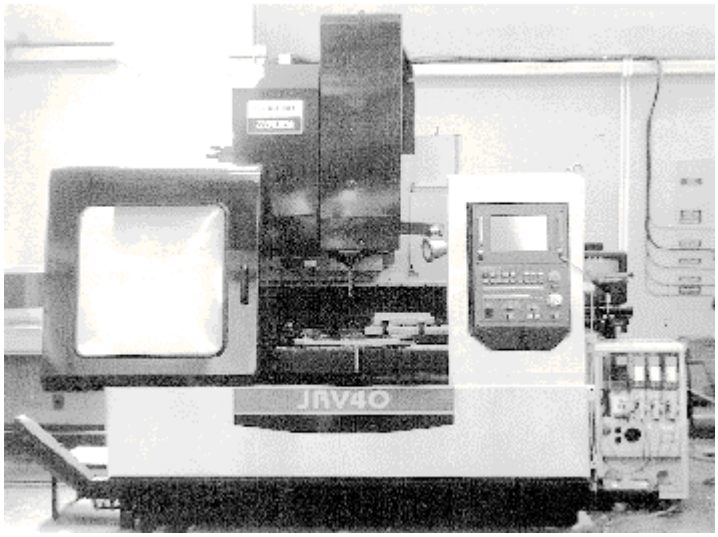

Figure 6-4 Overall View of Station-B

(1) Implementation

Station-B was equipped with a servo interface board in conformance with SERCOS, and controlled the off-the-shelf servo amplifier and motors (supplied from Indramat) through fiber optical cable. The panel computer and the CNC board exchanged data through a resource manager based on the MTF. Table 6-2 shows hardware construction.

Table 6-2 Platform Structure

PC	Pentium 100MHz, HDD 500MB, RAM 24MB Touch Panel Display (800×600 dots)
Servo Drive System	In Conformance to IEC1491(SERCOS) Optical Fiber Ring (2Mbps) Servo Amplifiers and Motors (Indramat)
NC Board	NC Board : Made by Toshiba Machine (TOSNUC-P) ISA Bus Interface , PLC Processor Inside

(2) Conclusion

Many human-machine operational software made separately by different members were installed within the panel computer as software parts. The ability to select and execute freely the software parts was confirmed. In addition, the ability to execute basic milling operations including ATC, while sending the NC data to a network server, was confirmed.

6.1.3 Station-C

Station-C, as an application example outsides the field of machine tools, controls a non-contact measuring machine developed by Sony Precision Technology Inc. based on the OSEC-II architecture. It fulfills 5 function groups with software on PCs with standard interfaces, and is completely controlled by a software CNC.

Figure 6-5 Overall View of Station-C

(1) Implementation

As the interface to sensors and electrical devices, the general-purposed counting, D/A and DI/DO boards are inserted into a PC. The calculation of the positioning loop is carried out by a process which is activated by an operating system at an interval of 8 msec. A software logic tool (ISaGRAF) conformed to IEC-1131 is also used.

This system, using the OS/2 Warp as its OS, is designed as an instance model for fulfilling 5 function groups based on the OSEC-II architecture with software.

Table 6-3 Platform Structure

PC	Pentium 133MHz, RAM 32MB, HDD 1.6GB
Servo Drive System	Velocity Control, Analog Input
PC-Inserted Boards	4Ch A/B Phase Counting Board 4Ch 16Bit D/A Board 32 Inputs 32 Outputs DI/DO Board

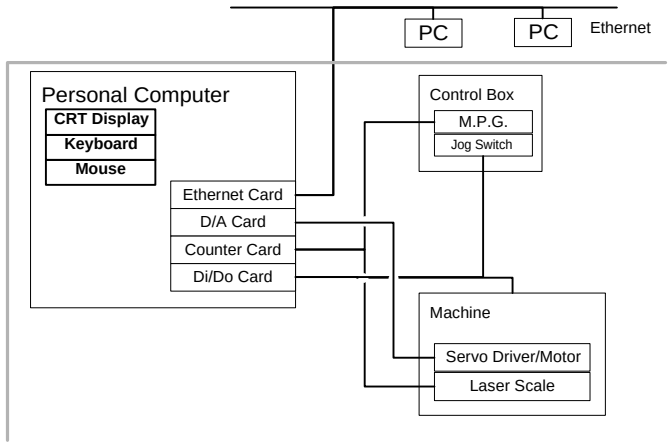

Figure 6-6 Connection of Hardware Blocks

(2) Conclusion

Through a design and a trial manufacture of the instance example with standard interfaces and 5 function groups which are defined in OSEC-II, It can be said that the effect of Ò widening the available choice of methods responding to objects Ó coming from controllerÕs openness has been verified. It was confirmed that a full-closed positioning loop servo of 8 msec sampling interval could be constructed on the OS/2 Warp operating system using the general-purposed inserted boards and servo drivers. It also shows the feasibility of openness.

6.1.4 OSEL Station

The objectives of demonstration on an OSEL station are to show

- the whole flow of information processing when OSEL is used,
- features of OSEL by a practical demonstration, especially the ease with which a user can incorporate his/her machining know-how into a program.

Significant information in OSEL machining programs includes machining sequence, cutter paths, and machining conditions. To make this information understandable, a shape viewer and some process sheets are used. A CAD tool (CADKEY) is used as the shape viewer and a spreadsheet (Excel) is used for displaying the process sheets. A workpiece and the cutter paths are displayed by using CADKEY, and the machining conditions, the tools, and the machining sequence are displayed on the spreadsheet. Excel is also used for process optimization.
A demonstration scenario is shown below.
(1) Workpiece is designed by using CADKEY.
(2) Machining features are designated in series by CADKEY.
(3) An OSEL-F machining program is created by using the above data. At this stage, instance creation of machining features and general machining specifications are described only in the program.

```
public class Sample
  public static void main(String args[])
  // Machining features are defined here.
  // m10_4:4-M10 Tap,  hole20: D20 through hole
  m10_4.machining();
  hole20.machining();
  }
```

Figure 6-7 OSEL-F Machining Program Created by Using CADKEY（Extract）

(4) A process sheet is displayed on Excel to confirm the machining features to be machined in the OSEL-F program.
(5) The OSEL processing system creates a plan of tools and expands the machining program written in OSEL-F by using a machining feature class library and a tool class library.
(6) Since wasteful tool change processes are included in the expanded program, the number of tool changes is minimized in the machining sequence on the basis of tool information and Z coordination of tool at which machining starts. The machining program written in OSEL-F is optimized on the basis of optimized sequence data.

```
public class Sample
   public static void main(String args[])
        // Machining features are defined here.
        // m10_4:4-M10 tap, hole20:D20 through hole
        // Machining by the center drill
        m10_4.centerdrilling();
        hole20.centerdrilling();
        // Machining by drill D8.6
        Tool drill8 = toolfordrilling(8.6); // Select from a tool library
        m10_4.drilling(drill8);
```

```
            // Machining by drill D15.8
            Tool drill15 = toolfordrilling(15.8);// Select from a tool library
            hole20.drilling(drill15);
            // Machining by endmill D15
            Tool endmill15 = toolforendmilling(15);// Select from a tool library
            hole20.milling(endmill15);
            // Tapping
            m10_4.tapping();
 }
}
```

Figure 6-8 Optimized OSEL-F Process Program (Extract)

(7) An OSEL-X program is generated from the OSEL-F program using the machining feature class library, the tool class library, and a machine-dependent machining library. The cutter paths of OSEL-X program are verified on CADKEY.

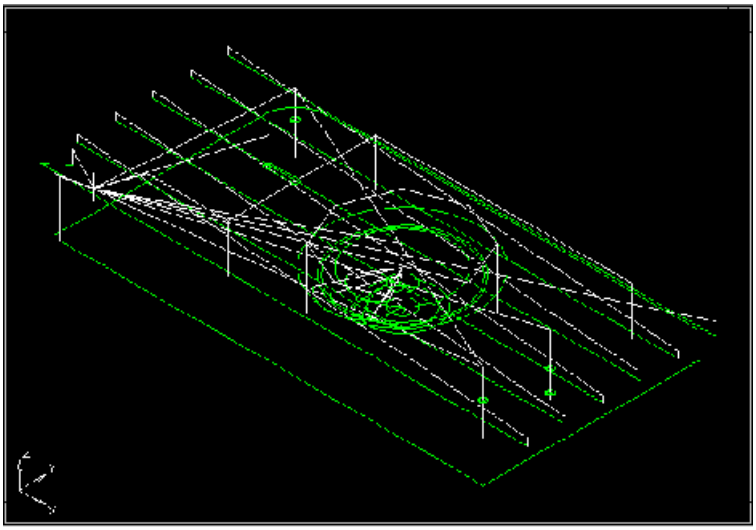

Figure 6-9 Verificaiton of Cutter Paths by CADKEY

Next, to illustrate the merits of OSEL, it is shown that tools can be modified easily. OSEL allows users to describe their know-how as a program and reuse it. For example, when a user selects a tool, he/she has to consider many conditions. Once the tools and their usage conditions have been programmed, an appropriate tool can be selected automatically afterwards. Moreover, the machining conditions and the cutter paths will be changed according to the tool selected. In the demonstration, the user changes a tool in a program and then a simulator written in Java shows that cutter paths are changed according to the tool.
The demonstration shows that users' machining know-how can be described in a library by using OSEL and that the use of Java as an implementation language for OSEL allows machining know-how to be disseminated on networks.

6.1.5 Network Stations

We also study about the utilization of open controllers in both Internet and Intranet environment. We name the space created from computers and networks ÏInternet Cyber Factory".

(1) Implementation

We built trial applications such as a remote monitoring, an operation supervision, an application for TV conference system, and so on. The network applications were constructed by applying general-purpose S/W and H/W for PCs and Internet technologies. Figure 6-10 shows the network system configuration. The stations with the OSEC controllers, a line database, PCs, and a mobile terminal were networked. Each station transmitted operating status to the line data base. Simultaneously, WWW browsers or a spread sheet software at the PCs or the mobile terminal read and displayed information in HTML format from the line data base. In addition, there were connected line monitoring TV-s or video communications between floors with a TV conference system. The status transmission from the controller was done by OFMP (OSEC Floor Management Protocol).

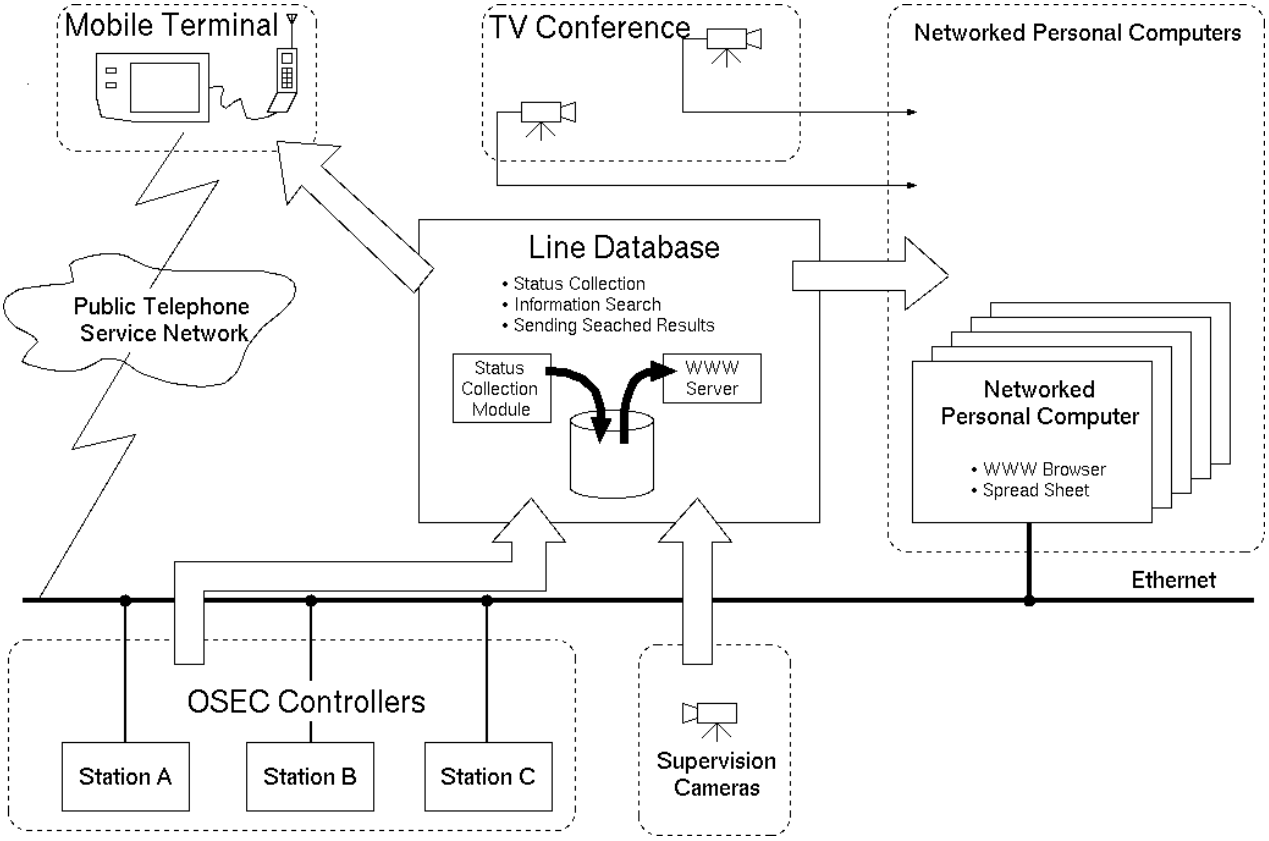

Figure 6-10 Network System Configuration

(2) Conclusion

As a result, we can develop the network applications in a short period and at low cost. The following problems should be considered in the "Internet Cyber Factory".

- Security countermeasures
- Mechanism of commercial transactions via Internet
- Administration of network equipment
- Connection of existing NCs in networks

7 Results and Issues

There are four main working groups in OSE (Open System Environment), and OSE has been able to configure modules, analyze interfaces among them, and to perform a test for evidence. Then, OSE could gain the following results for OSEC-II.

(1)Development of Human Machine Interface (HMI) System with Machine Frame:
Each element of machine tools is summarized in an Object Relational Drawing (ORD), and operation / display class libraries and software development system (Machine Framework) are developed based on these abstract models (Machine Resource Object). Also, several examples of machine operation displays are tested and evaluated. Then, it is proved that we can develop indication and operation displays in a short time period with a high efficiency by adapting Object Relational Programming.

(2)Development of OSEL Language:
Language specifications of OSEL, the new NC language, which is available to be added using process know-how, is analyzed, and three class libraries (tool class library, class library which depends upon each machine type, and class library which describes cutting method corresponding to process feature) are tested. Then, an OSEL program was performed on an actual machine, and the functionality was verified.

(3)Interface Specifications for Machine Control and Creation of Language Processor:
Interface specifications to machine control for generating tool paths is defined as an OSEC API, and then they are implemented with libraries.

(4)Creation of Interface Specifications for Servo Controller:
The interface with the servo controller is defined as an OSEL-API, and then software for machine controls and servo controls, meeting the specifications, is developed. Then, its movement for connection with a servo drive is verified.

(5)Creation of network system using internet technology:
The network system is tested. This system uses the internet technology.

There are some more issues that need to be resolved:

(1)Since a fusion has become free, inconvenient problems, such as machine suspension caused by combinations of different parts and modules provided by multiple vendors, are not addressed here.

(2)Responsibility of accidents caused by system inconvenience; that is to say, responsibility of products, is not addressed here, either.

(3)It is considered that a merit by reducing production lead-time would be created by gaining free access to resources on the networks (CAD / CAM, information of production management, electronic trading, and so on) with connections to internet or intranet. However, problems on illegal invasion and security matters are not mentioned here.

As future issues, it is necessary to investigate standards of interface between networks and open controls and to encourage HMI widely.

Appendix: Machine Resource Object and Function Block API of OSEC-II

Machine resource object and OSEC-II API based on function blocks are shown in the following tables. Please refer to the OSEC-II report for the detail information and their parameters. The API is first version and not yet finish one. Therfore it might be modified accroding with the progress of OSEC-III project.

1. Machine Resource Objects

1. Basic Domain Class Objects	
OsecAlarm	Alarm of machine tool class object.
OsecMachine	Generic equipment class object It has the methods for starting and stopping. It has also an alarm list and can provide a method of alarm reset .
OsecNCMachine	NC machine tools class object. It has methods for system mode control such as jog and program machining and remote control.
2. Operation Domain Class Objects	
OsecNCResource	Object like a holder to bind many resource objects. Its sub-class can be defined functions associated with some resource objects. It is also possible to define the resource configuration each machine tool.
OsecAxis	X, Y, Z axis class object. It has sub-class esof translate and rotation axis
OsecLinearAxis	Line Axis object class
OsecRotationalAxis	Rotational axis class object
OsecSpindle	Spindle motor class object
OsecNCDecoder	NC decoder class object
OsecEIADataDecoder	EIA decoder class object
OsecOSELDecoder	OSEL interpreter class object
OsecToolChanger	Tool changer class object
OsecToolMagazine	Tool magazine class object
OsecPalletChanger	Pallet changer class object
OsecCoolant	Coolant class object
OsecChipConveyor	Chip conveyor class object
OsecPLC	PLC class object. It provides functions of download of plc programs and input/output X/Y signals. Devices except standard resource objects are defined by way of X/Y signals.
OsecNCEngine	Class object for motion control. It coordinatesmovement of more than two axis
3. Basic Data Domain Class Objects	
OsecRecipeManager	Management class object Of database such as tools, tool adjustment, and NC programs
OsecNCProgram	NC program class object.
OsecTool	Tool class object
OsecToolTable	Tool table class object
OsecToolTableItem	Element of tool table class object
OsecToolOffsetTable	Tool offset table class object
OsecToolOffsetTableItem	Element of tool offset table class object

2. Resource Management API

API name	Description
OsecRmOpen()	Open machine resource
OsecRmClose()	Close machine resource
OsecRmGetRunCommand()	Get run command
OsecRmGetHoldCommand()	Get hold command
OsecRmGetEiaProgram()	Get Eia program
OsecRmSetRunStatus()	Set run status
OsecRmSetAlarm()	Set Alarm
OsecRmGetSingle()	Get single block stop
OsecRmGetBlockSkip()	Get block skip status
OsecRmGetOptStop()	Get optional block skip status
OsecRmSetStopStatus()	Set stop status
OsecRmGetReady()	Get ready
OsecRmSetFileName()	Set program file name
OsecRmGetNcAutoStatus()	Get auto-mode status of NC

3. Motion Generation API

API name	Description
OsecMgenOpen()	Open motion generation
OsecMgenClose()	Close motion generation
OsecMgenInit()	Initialize motion generation
OsecMgenReset()	Reset motion generation
OsecMgenReady()	Motion generation ready
OsecMgenStart()	Start motion generation
OsecMgenStop()	Stop motion generation
OsecMgenStep()	Step execution
OsecMgenResume()	Resume
OsecMgenSetProgram()	Set program name
OsecMgenGetProgram()	Get program name
OsecMgenSearchBlock()	Search program block

(1)OSEL-EX Decoder API

API name	Description
OsecOselOpen()	Open osel-ex decoder
OsecOselClose()	Close osel-ex decoder
OsecOselInit()	Initialize osel-ex decoder
OsecOselRun()	Run osel-ex program
OsecOselFileOpen()	Open osel program and interpreter
OsecOselStep()	Step execution
OsecOselExDebug()	Set debug display mode
OsecOselSetBreakPoint()	Set break point
OsecOselReleaseBreakPoint()	Reset break point

(2)EIA Decoder API

API name	Description
OsecEiaOpen()	OpenEia decoder
OsecEiaClose()	CloseEia decoder
OsecEiaInit()	initialize decoder
OsecEiaReset()	ResetEia decoder
OsecEiaReady()	Eia decoder Ready
OsecEiaStart()	StartEia Program
OsecEiaStop()	StopEia Program
OsecEiaStep()	Step execution of Eia program
OsecEiaResume()	ResumeEia Program
OsecEiaSetProgram()	Set Eia program
OsecEiaGetProgram()	Get Eia program
OsecEiaSearchBlock()	Search Eia program block

4. Machine Control API

(1)Machine Control API

API name	Description
OsecMctlOpen()	open an interface of Machine Control module
OsecMctlClose()	close an interface of Machine Control module
OsecMctlReady()	on/off a ready mode of Machine Control module
OsecMctlCancelBuf()	cancel data in command buffer
OsecMctlStart()	start interpretation of commands
OsecMctlReset()	initialize of Machine Control module
OsecMctlStop()	stop interpretation of commands
OsecMctlProgramEnd()	set command end

(2)Mode Control API(Option)

API name	Description
OsecMctlModeSingleBlock()	on/off a single block mode
OsecMctlModeDryrun()	on/off a dry run mode
OsecMctlModeExactStop()	on/off an exact stop mode
OsecMctlModeAxisCancel()	on/off a cancel mode of axis motion
OsecMctlOverCut()	set override value of feed rate
OsecMctlOverPart()	set partial override value of feed rate
OsecMctlOverMove()	set override value of rapid feed rate
OsecMctlAccurate()	set high accuracy machining mode

(3)Axis Control API

API name	Description
OsecMctlAxisProgramOrigin()	set origin point at program coordinates

OsecMctlAxisWorkOffset()	set offset data at work coordinates
OsecMctlAxisMove()	rapid feed move command to target position
OsecMctlAxisLine()	linear cutting feed move command
OsecMctlAxisCps()	cutting feed move command on the line of position data
OsecMctlAxisCpsAdd()	add data on the line of position data
OsecMctlAxisWait()	wait command for given time
OsecMctlAxisVelocity()	move command with directed velocity
OsecMctlAxisOrigin()	return command to origin point

(4)Axis Control API(Option)

API name	Description
OsecMctlAxisArc()	move command with an arc interpolation
OsecMctlAxisArcVector()	move command with an arc interpolation (3 axis control at the same time)
OsecMctlAxisHelical()	move command with a helical interpolation
OsecMctlAxisTap()	tapping command
OsecMctlAxisSkip()	linear cutting feed move command with skip function

(5)DI/DO API

API name	Description
OsecMctlSetDIDO()	set ON/OFF of I/O

(6)Get Command API

API name	Description
OsecMctlGetFreeBufCount()	get the size of free area of command buffer
OsecMctlGetUserBufCount()	get the size of command buffer
OsecMctlGetAxisStatus()	get the axis status
OsecMctlGetPosition()	get the present position
OsecMctlGetProgramOrigin()	get the origin point of program coordinates
OsecMctlGetWorkOffset()	get the offset value of work coordinates
OsecMctlGetBlockNo()	get the block number
OsecMctlGetInfo()	get the information
OsecMctlGetTargetPosition()	get the target position
OsecMctlGetStrokeLimit()	get the stroke limit data
OsecMctlGetSoftStrokeLimit()	get the soft stroke limit data
OsecMctlGetMAXVelocity()	get the data of maximum velocity
OsecMctlGetDiDoStatus()	get the status of DI/DO
OsecMctlGetDiDoData()	get the data of DI/DO
OsecMctlGetStatus()	get the present Machine Control status
OsecMctlGetTheoryPosition()	get the theoretical position data

(7)Alarm Command API

API name	Description
OsecMctlSetAlarm()	set the alarm information
OsecMctlGetAlarm()	get the alarm information

5. Servo Control API

(1)I/F Management API

API name	Description
OsecSrvOpen()	Open serve control
OsecSrvClose()	Close serve control
OsecSrvInit()	Initialize servo
OsecSrvSetInfo()	Set servo information
OsecSrvGetInfo()	Get servo information

(2)Group Management API

API name	Description
OsecSrvGrpInit()	Initialize servo group
OsecSrvGrpSetInfo()	set servo group information
OsecSrvGrpGetInfo()	get servo group information

(3)Servo Axis Management API

API name	Description
OsecSrvAxisInit()	Initialize servo axis
OsecSrvAxisSetInfo()	set servo axis information
OsecSrvAxisGetInfo()	get servo axis information

(4)Buffer Management API

API name	Description
OsecSrvGetFreeBufCount()	Get free buffer count
OsecSrvGetUsedBufCount()	Get used buffer count
OsecSrvCancelBuf()	Cancel buffer

(5)Servo Control API

API name	Description
OsecSrvErrorReset()	Reset servo error
OsecSrvSetSrvOn()	Set servo on
OsecSrvSetSrvOff()	Set servo off
OsecSrvSetCyclicCmd()	Set cyclic command
OsecSrvGetCyclicData()	Get cyclic data
OsecSrvSyncroCyclicData()	Synchronize cyclic data
OsecSrvWaitEvent()	Wait servo event

6. PLC Input / Output API

(1)I/F Management API

API name	Description
OsecPlcOpen()	Open plc
OsecPlcClose()	Close plc
OsecPlcInit()	Initialize plc
OsecPlcGetInfo()	Get plc information
OsecPlcSetInfo()	Set plc information

(2)PLC Input / Output API

API name	Description
OsecPlcGetIOData()	Get plc-io data
OsecPlcSetIOData()	Set plc-io data
OsecPlcGetIDbyName()	Get ID by name