SANYO DENKI TECHNICAL REPORT No.7 May-1999
Feature

Development and Caracteristics of Advanced Motion Language(AML)

Hiroshi Yoshikawa

1. Introduction

Having developed AML for the PC based controller to further encourage open-architecture of FA, we are going to discuss AML's advantageous characteristics and design concept in this paper.

When writing a program with a language for describing "motion," the program is written either with reference to "position" or to "time." A typical example of a program written with reference to position is G Code (IEC RS274D): a motion descriptive language of the linear coodinates system. A typical example of a program written with reference to time is Ladder Diagram: a Programmable Logic Controller (PLC) language. While these two types of languages have been separately developed over the past 30 years, moves have recently started toward integrating G Code and Ladder Diagram in the course of the standardization of software PLC (IEC 1131-3). It is a method to call a G Code program within a sequence program. It is, however, no different from the existing method given the context that the program describing a coordinate (position) and a program describing the timing are written separately. This does not provide a real solution.

We have solved this integration problem with AML by adopting an event processing method. We have also developed an interpolation method called Drive Train. We can use Drive Train to write programs for synchronizing control and multiple axis control easily that have not been supported by any other programming languages up until AML.

2. Development Concept of AML

AML is one of the important components of the open architecture controller "S-MAC" system which we created here at Sanyo Denki.

The "S-MAC" system consists of Types A, B, and C. As the descriptive language of motion, Type A supports the robotic language, Type B supports G Code and Type C supports AML. In this paper we will discuss the advantageous characteristics of AML with a particular focus on the relation between the "S-MAC" system Type C and what provided the background of AML development.

We will also compare AML with other motion control languages.

2.1 Full Software Controller and AML

Fig. 1 shows how the three "S-MAC" systems ,type A, B, and C are positioned in term of application. In addition to being a motion descriptive language AML is also a run-time engine to execute SERCOS IDN. This function has made it possible to remove the motion card from Type C.

2.1.1 Development concept of "S-MAC" Type C

Basic "S-MAC" Type C development concepts include the following 7 items:

1. The time required for development was to be kept as short as possible, using a simple control language.

2. The international standard SERCOS would be used for the motion bus.
3. An object-oriented command system would be used.
4. Synchronizing operations would be controlled by position rather than by velocity, requiring a language capable of defining the rotary coordinate system.
5. Interpolation would be done by software rather than by hardware (motion cards).
6. An external sequencer would not be used. Software PLC would be used.
7. For error correction, the recipe method used by the metal forming machine controller (software correction) was preferred to the overriding method (hardware correction) used by the cutting machine controller.

The advantages of the full software controller could not be realized unless all 7 conditions were satisfied. We did not have any time, however, to create such a language from scratch as we were pressed by the scheduled commercialization of the product. This drove us to look all over the world for a likely candidate for such a language. We found AML which had been developed by Automation Intelligence Inc. of the USA for a packaging machine system. The functions that were missing for application to "S-MAC" Type C were added through a joint development effort which was concluded as the current AML. The history of the development is shown in Fig. 2.

Note: It took us 14 years when we started to write the program in the C language 4 years since we implemented the program into the Windows environment.

2.1.2 Design choice on the development concepts of AML and the "S-MAC" system

Both AML and "S-MAC" Type C were developed on the basis of certain choices made in the course of the development process.

The fact that these choices coincided between the language and the system, has resulted in AML being positioned as the most suitable language for the purpose of "S-MAC" Type C and have, eventually, enabled the system to be incorporated into a variety of application systems. Such design choices are as listed below:

1. PC based controller
   The selection of (IBM compatible) PC as the hardware platform
2. SERCOS motion bus
   The selection of the open standard SERCOS as the communication protocol between controler and servo driver
3. Real-time OS
   SERCOS target system was made a multi-platform system which is independent of a specific real-time OS.

2.2 Advantage of Using AML

The advantage in the use of an open architecture controller is the ability to shorten the development lead-time. We can see how the implementation of AML helped minimize the development cycle.

The normal development cycle for a new control system using a servo motor proceeds as follows:

1. System design (to decide how the burdens of the process should be shared between software and hardware.)
2. Software and hardware designs
3. Hardware Adjustment (manual operation)
4. Adjust of the servo system (single axis operation)
5. Behavioral verification (Linking test of the processing and sequence programs)
6. General adjustment (to verify the linkage to Human machine interface (HMI),

accuracy, processing tact time, etc.)

The development cycle of a new product is normally set at 6 months for a simple system and at 1 year for a complex system. It is not possible to shorten the lead time itself, but the number of man-hours can definitely be reduced to about 1/2 or 1/3 by using AML as it will help reduce the working time of the engineers concerned because of the improved efficiency of the debugging process.

One programmer, for instance, can write both the motion and sequence control programs. This eliminates the necessity of an engineer to compose the sequence control program and also helps minimize the interface specifications to significantly cut down the design man-hours. Furthermore, AML development environment itself are structured by laying weight on its function as an on-line debugging tool rather than as a programming tool, so that the time required for the motion verification of the system can be substantially reduced.

2.3 Comparison with Other Languages

Table 1 Comparison table with other languages

	◎>○>□>△	AML	General purpose languages C, VB, ALGol	G_Code	PLC	Robot Language	Local Command
Motion	PTP	◎	○	◎	○	○	◎
	Multiple axis interpolation	△	○	○	△	◎	○
	Synchronization	◎	□	□	□	○	□
	Super multiple axis machine (16 axes or more)	◎	□	△	△	○	□
	Robot (articulated)	△	○	△	△	◎	□
	Control responses	□	○	□	△	○	◎
Programming levels	Program development lead-time	◎	□	○	□	○	□
	Network drive	◎	○	△	△	○	□
	Tuning and monitoring	◎	□	○	△	◎	○
	Upper layer communication (E-net, Internet)	◎	◎	△	△	◎	○
Mechanism (Object)	Electrical cam and gear	◎	○	○	□	○	○
	Electronic shaft (rotational cycle)	◎	○	□	□	○	□
	Master - Slave virtual master	◎	○	○	□	○	○

Generally speaking, it is quite difficult to directly compare control languages with each other as each language is quite closely targeted for a specific machine and objectives as it is developed. Table 1 is the result of our comparison which is done by assuming a control program written in the general purpose language C for a 5-axis system controller to be used as the reference. Scores are naturally higher for AML on the functions of the electronic cam and gear as they are not supported by the other languages.

3. Stracture of AML

3.1 Development and Target System Environments

The control system using AML is configured from a development environment PC and target PCs. The basic configuration of the system is shown in Fig. 3 below:

The development environment is mainly used for composing AML applications (a number of programs written in AML) or for setting up a development environment and target systems.

The target system is used for executing AML applications and for controlling equipment interconnected with the SERCOS ring.

The development environment and target system are interfaced with a RS-232C cable or Ethernet cable (10BASE-T) for the purpose of down/uploading AML applications or transmitting data required for monitoring the operating status.

Control with HMI (human machine interface) or other upper layer system will be required for composing a FA control system, but a linkage using RS-232C or Ethernet cable can be established between computers even in such instances.

Examples of the configuration of the development environment and target systems using the AML control system are introduced below:

Fig. 4a. indicates a development system built with a PC for a development environment and many PCs for the target system. As a single AML development environment can control a number of AML target systems, the one against many control system can be established by connecting all units with Ethernet cables. In this situation, a single development environment can carry out the development of an AML application for one target system while monitoring the operating status of another target system. It is not allowable, however, to simultaneously carry out a number of similar operations such as monitoring the execution status that is going on in more than one target system.

Fig. 4b. indicates a (stand alone) development system with only one PC in a target system. In a normal situation, a single unit control system can be established in the target system by separating the development environment when the monitoring of a target system is no longer needed, or when the development of an AML application is completed. It is possible to establish such a system because the target system of AML has a function which can automatically start up the system after the power supply is turned on.

Thus the advantage of the AML system is that it is possible to expand the system from the basic configuration of a development environment against a target system. It also enabled a centralized control of the AML program of the development environment and the program of the target system.

3.2 Development Environment

The development environment will be described in detail in this section.

3.2.1 Use

The development environment is used for the following purposes:

1. To set up development environment and target system
2. To edit and save the programming of AML application
3. To start and stop the AML application
4. To debug the AML application
5. To monitor the state of execution

3.2.2 Hardware

Hardware usable for the purpose of the development environment is the SMS-10 computer made by Sanyo Denki or commercial PC/AT compatible personal computers. As mentioned in this section, such personal computers must be provided with at least an RS-232C serial port or Ethernet port (10Base-T, RJ-45 jack) as it is necessary to communicate with the target system.

3.2.3 Operating System

The operating system used in the development environment shall be Windows 3.1, 95, or NT.

3.2.4 Software

In this section, we will explain about the group of AML software used in the development environment.

(See Fig. 5)

The following software may be directly operated by users:

1. Configuration Tool
   To be used for setting up development environment and target system as well as SERCOS device.
2. Maintenance Tool
   To be used for the motion verification of SERCOS device.
3. AML Development Environment
   To be used to develop (for programming and debugging), start up, and shut off AML applications.
4. LogViewer
   To display the state of the target system in error, warning, and user messages.
5. SERCOScope
   To graphically display SERCOS communication data (ex. command values or feedback values) in real-time.

The software to be used in the background is as follows:

1. MPP (Multi Purpose Protocol)
   To handle communication processes with the target system. All data to be transmitted and received by the AML software as mentioned above shall be made through MPP.
2. Pipeline
   In a control system, it may be necessary to exchange data directly with the target system. It is possible, for example, to operate a control system with a HMI composed in Visual Basic, or to make some kind of processing with Excel by acquiring the current position of the motor. Pipeline shall be made possible to exchange data with the application in the target system. DDE (Dynamic Data Exchange) shall be the protocol used when doing so.

3.3 Target System

The target system will be described in detail in this section.

3.3.1 Use

Target system will be mainly used for the following purposes:

1. To execute AML applications
2. To control connected equipment and devices

3.3.2 Hardware platform

Hardware used in the target system is "S-MAC PC", model "SMS-10" which are

"S-MAC" components. This is a PC/AT compatible FA computer, which has an excellent ability to withstand severe environments and is provided with SERCOS interface as a default.

3.3.3 Operating system

The operating system used in the target system is iRMX, a real-time OS of Radisys Inc. We plan to release VxWorks and Windows NT (RTX) versions soon.

3.3.4 SRX

SRX (SERCOS Runtime eXecutive) is the generic designation of AML software running in the target system that serve as the runtime engine to run AML applications and control connected peripherals. Fig. 6 below depicts the configuration of the target system software.

While Fig. 6 indicates focusing on the task to communicate with external devices, there is also an interpreter and a task group related to motor control. These tasks communicate with each other by using a mailbox or semaphore in coordination to carry out system processes.

Now, we will describe MCU (Motion Control Unit) which is a task group related to motor control.

One of the most important tasks of MCU is the SERCOS interrupt task to be processed by cyclic interrupts (which can be set in increments of 1 msec within a range from 1 to 20 msec). The main processes done by the interrupt is to transmit/receive cyclic data through communication with the SERCOS device. In addition, it may calculate and transmit command values or receive feedback values at the interrupt cycle by using a motion profile generated from the commanded velocity or acceleration/deceleration. It will also generate an event to notify other tasks when the commanded velocity is reached or a position command is completed. One advantageous functions of AML is the drive train. This function electronically realizes a gear or cam movement by synchronizing master and slave axes and is also carried out by this task.

There are other tasks to execute a homing process or to perform pretreatment for Drive Train.

3.3.5 Multi tasks

It was explained in the previous paragraph that a number of system tasks are coordinating to carry out processes on the real-time multi-task system in the target PC.

A multitask system is also implemented in AML applications built by users. An AML application is composed from a number of modules. (See Fig. 7)

The term "AML application" used here indicates a unique software system built by a user, provided to a control system and composed from a number of modules.

The term "module" used here means a program written in AML and processed as an independent task on the SRX. In other words, each module is processed as a separate task when an AML application consists of more than one module.

The task priority of each module is fixed and it is restricted in that no different priority may be assigned to each module. It follows that task switching is used when a number of modules must be run at one time. This is the round robin method where the execution of a module will be suspended if the maximum allotted time is exceeded and moves the processing to the next module. It helps avoid a situation where a module occupies the position indefinitely to prevent the next module on the schedule from being processed.

Generally speaking, it is a difficult task even for experts to set different priorities for every task to streamline the execution of the entire system so called "task scheduling". The reason for implementing the round robin method for the execution of an AML application is to enable a user who is unfamiliar with the multi-task system to build a system.

Although modules are tasks that are independent from each other, certain mechanisms are provided to enable them to be processed in coordination.

One example of such mechanisms is a function for one module to control another module. The control includes the loading, startup, shut off, and unloading. Using these functions, it will become possible to efficiently process a system possessing multiple modules in a target system with a limited CPU resources.

Another of such functions is the event processing function described in the next section.

By effectively utilizing these functions, it will become possible to build a foreseeable program with functions assigned separately by module.

3.3.6 Event processing function

An event processing function is another advantageous features of AML applications. It offers the following advantages as compared with the other general scanning (or polling) processes.

1. CPU resources can be effectively utilized.
2. The response time following an event occured is always constant.

Let us explain the differences between event processing and scanning processing as follows: (See Fig. 8)

(1)Scanning process (Fig. 8a.)

1. The module on standby monitors the system at regular intervals until the state of the system changes. The module remains as a task in an active state while on standby, consuming CPU resources.
2. As the event source changes its state, the module on standby detects it and finishes the scanning process.
3. The module on standby will resume processing.

(2)Event processing (Fig. 8b.)

Events are controlled and monitored by the system (OS).

Module on standby waits for a startup signal from the system. The task on standby is in a sleep state and consumes no CPU resources.

1. As an event source generates an event, the system changes the state of the task from sleep to active.
2. The module on standby will resume processing.

3.4 AML Interpreter

AML uses an interpreter method, meaning that it translates the source code (program in making) line by line (sequential processing) and runs the program at the same time. Generally, the interpreter method has the drawback of slower processing speed than the method where the program is compiled in an executable form, but on the other hand, the interpreter method has the advantage that the source code is easy to handle as it does not need any compiling process. In order to maintain the advantage of the method while covering the shortcomings, AML uses the pseudo code rather than the source code.

The pseudo code will be loaded onto RAM in the target system by using the AML software in the development or auto-loading function in the target system.

Procedures from the development to the execution of an AML application are shown in Fig. 9 to clarify how the pseudo code is handled.

(1)Edit

Modules configuring the AML application are made and edited in AML (language) by using AML software in the development environment. The pseudo code (extension .AML) is automatically generated and saved on the hard disk.

(2)Download

The pseudo code will be downloaded from the development environment to the target system by using AML software in the development environment, which results in copying the pseudo code in the compact flash memory in the target system.

(3)Load

The pseudo code will be loaded onto RAM in the target system by using the AML software in the development environment or auto-loading function in the target system. Comments and other elements unnecessary for execution will be removed during the process.

(4)Execution

The pseudo code is executed by software in the development environment or by the auto-load function in the target system. The pseudo code is in an executable form by the interpreter (sequential processing) when edited.

In the case of a controller such as PLC, the process normally follows a flow where (a) the source code is edited in the development environment, (b) the execution code is generated by compiling, (c) the program is then downloaded into the target system, and (d) executed and edited in the target system. Troubles then arise after repeated debugging when the source code and the running code do not match. It is advantageous that the running code is modifiable while in the field, but then this makes it difficult to be properly maintained.

AML, however, uses the same pseudo code for both editing and running as seen in the explanation above so that mismatches will never occur.

4. AML's Advantageous Features

We discussed the advantages points of AML as seen from the AML software system in the preceding chapter. We will now introduce the advantageous characteristics of the motion programming system.

4.1 Program

AML is a high level programming language resembling PASCAL. It is, therefore, easier to read and to write programs written in AML than those written in the ladder diagram and G code. It is also easier for debugging, too.

The basic syntax is same as other general high level programming languages. The program is composed with statements that begin with a ″DECLARE″ command and executable statements are sandwiched between ″BEGIN″ and ″END″.

Function calls are made with calls used in ordinary programming and also with unique AML calls.

There are also other objects provided that summarize necessary functions for controlling purposes. (See next section.)

4.2 Object

AML is an object based programming language. In other words, functions frequently used for controlling purposes are defined as objects, implementing the object principle. Object is a software block composed of data members and methods. Data member refers to setup data required for executing the method and method refers to a routine to process the object.

AML does not, in principle, allow users to define a new object (also called a class). Experienced users of high level language, especially of structured programming languages may feel that it is restrictive in comparison with other object-oriented languages like C++ or Java. It is, however, a consideration made for the benefit of users inexperienced in structured programming so that they can write easy and safe control programs. New objects can be added to the system as needed.

Available objects in AML are indicated below:

4.2.1 MOTOR Object

Use:
    MOTOR object is used to arrange motors defined in system configuration to make them usable in AML application and control motors.
Data members (a part of)
    DefaultAccel: Default acceleration
    DefaultSpeed: Default velocity
Method (a part of)
    On(): servo on
    Off(): servo off
    Start(): To move a motor in defined motion

4.2.2 ABS_MOVE Object

Use:
    To set up an absolute move:
Data member (a part of)
    Endpoint: Endpoint
    Mode: Direction and method of movement
    Speed: Velocity of movement
Method (a part of)
    NotifyCompleteAs(): To generate an event at the end of a motion
    NotifySpeedAs(): To generate an event upon reaching the target velocity

4.2.3 Other important objects

DRIVE_TRAIN
    To arrange and control a drive train
EVENT
    To be used for event handling
IO_BOOLEAN

To arrange and control I/O.

PLS

To set up and execute a programmable limit switch

4.3 AML Programming Examples Using Objects

Programming examples are shown in Fig. 10

```
   // Declaration section
DECLARE MoveComplete EVENTID
DECLARE MyEvent EVENT
DECLARE count WORD
DECLARE MainAxis(″Axis_1″) MOTOR
DECLARE MyMove() ABS_MOVE

   // Execution section
BEGIN

   // To make a specified event receivable with an object.
LOOKFOR(MyEvent)

   // To set up a data member of a MOTOR object.
MainAxis.DefaultAccel = 500
MainAxis DefaultDecel = 500

   // To set up a data member of an ABS_MOVE object.
MyMove.Mode = MOVE_FORWARD
MyMove.Speed = 100
MyMove.Endpoint = 0
MyMove.NotifyCompleteAs (MoveComplete)

   // Servo on
MainAxis.On()

   // To move a motor to the initial position by an absolute move.
MainAxis.Start(MyMove)

   // Standing by at the end of a motion
MyEvent.Wait()
FOR count FROM 1 to 10 DO
IF (count%2 == 1) THEN

   // To set up a distance in the normal direction of movement
MyMove.Endpoint = 500
ELSE

   // To set up a distance in the reverse direction of movement
MyMove.Endpoint = 0
ENDIF

   // To execute in an absolute move
MainAxis.Start(MyMove)

   // Standing by at the end of a motion
MyEvent.Wait()
ENDFOR

   // To stop a motion
MainAxis.Stop()
```

```
   // Servo off
MainAxis.Off()

   // To display a message
LogMsg("Abs Motion Complete")
END
```

Fig. 10 Program examples

5. Actual Application Examples

5.1 Application to a Packaging Machine

An example of a pillow type packaging machine application is shown below.

The size of the AML program is about 3,000 lines (including comments) for a 3-axis system.

5.1.1 Axis configuration

The axis configuration of the system is shown in Fig. 11. The function of each axis is as shown below:

Lug Chain Axis: Synchronized feeding control of goods

Fin Seal Axis: Fin sealing control

Crimp Axis: Crimp sealing control

5.1.2 Outside view

Fig. 12 indicates a outside view of the system.

5.2 Wire Winding Machine Application

A wire winding machine for the segment core of motors where the core is rotated for winding wires is shown below.

The size of the AML program is about 2,000 lines (including comments) for a 5 axes system.

5.2.1 Axis configuration

The axis configuration is shown in Fig. 13. The functions of the respective axes are as shown below:

q axis: to control the core rotation synchronization

X axis: to control the offset (movement) in direction X.

Y axis: to control the offset (movement) in direction Y.

Z axis: to control the offset (movement) in direction Z.

S axis: to control the supporting roller of the wire

5.2.2 Outside view and fabricated samples

Fig. 14 indicates a outside view of the system and fabricated samples.

5.3 Application to a Thread Rolling Machine

This is an example of application to a thread rolling machine where threading dies on the left and right are synchronized and moved closer toward each other for a threading process.

The size of the AML program is about 12,000 lines (including comments) for a 5-axis system.

5.3.1 Axis configuration

The axis configuration is shown in Fig. 15. The functions of the respective axes are as shown below:

SR axis: the synchronous control of the right spindle

SL axis: the synchronous control of the left spindle

TR axis: the synchronous control for the slanting axis (RH)

TL axis: the synchronous control for the slanting axis (LH)

M axis: the synchronous control for moving the axis closer

5.3.2 Outside view and a fabricated sample

Fig. 16 indicates a outside view of the system and fabricated samples.

6. Conclusion

In response to the demands of the FA industry for open structuring and software control, we have developed a powerful next generation language for full software controllers called AML.

Although the software needs further provisions for additional functions such as high speed and high precision circular interpolation for controlling laser cutting machines and a high speed coordinate conversion function required for a multiple joint robot controller, we are ready to provide sufficient solutions to cover needs that could not be fulfilled by the existing programming languages such as G code or Ladder diagram with the basic functions of our current system. We are committed to further promote the open structuring of AML in coordination with our customers together with the development of multi-platform systems for a target system to make AML the de facto standard in this sector of the industry (packaging, thread rolling and wire winding machines, etc.).

The author wishes to hereby acknowledge Mr. Eichi Kobayashi's valuable contributions and would like to thank him for his help, especially for the composition of Chapters 3 and 4.

Hiroshi Yoshikawa
Joined company in 1996
Control Systems Division
Worked on development of controllers

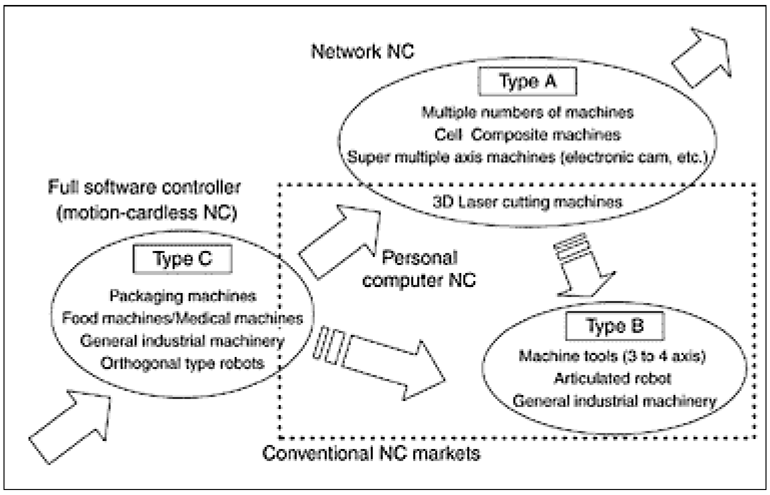
Network NC
Type A
Multiple numbers of machines
Cell Composite machines
Super multiple axis machines (electronic cam, etc.)
3D Laser cutting machines
Full software controller
(motion-cardless NC)
Type C
Packaging machines
Food machines/Medical machines
General industrial machinery
Orthogonal type robots
Personal
computer NC
Type B
Machine tools (3 to 4 axis)
Articulated robot
General industrial machinery
Conventional NC markets

fig. 1 ″S-MAC″ system

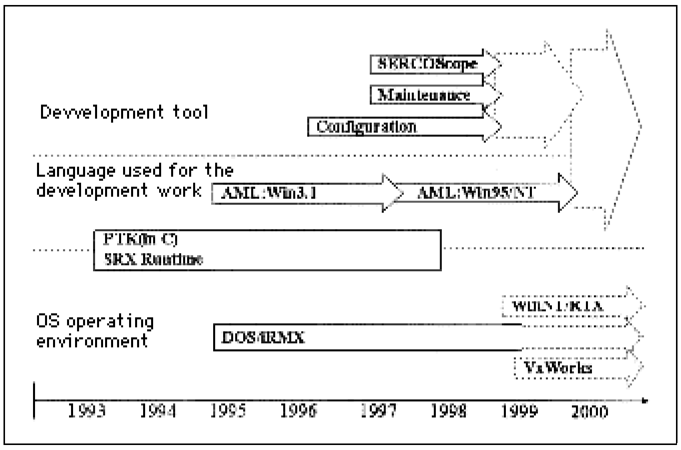

fig. 2 AML development history

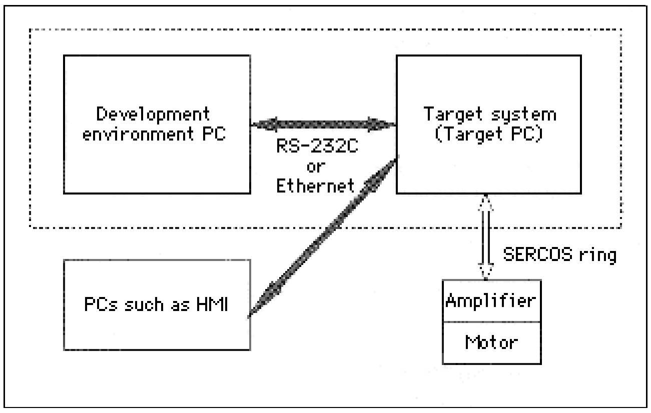

fig. 3 AML control system basic configuration

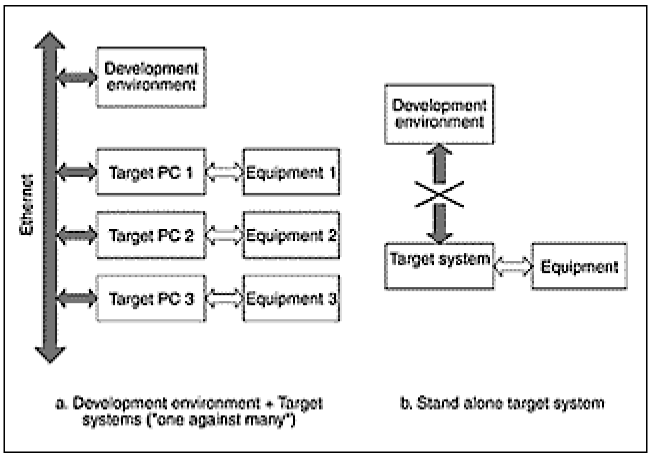

fig. 4 Example of a system configuration

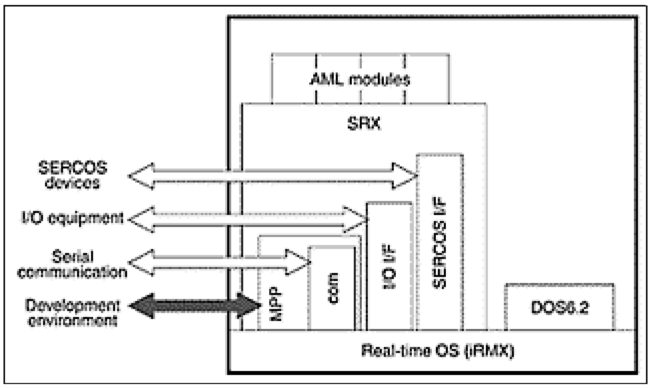

fig. 5 Development environment software group

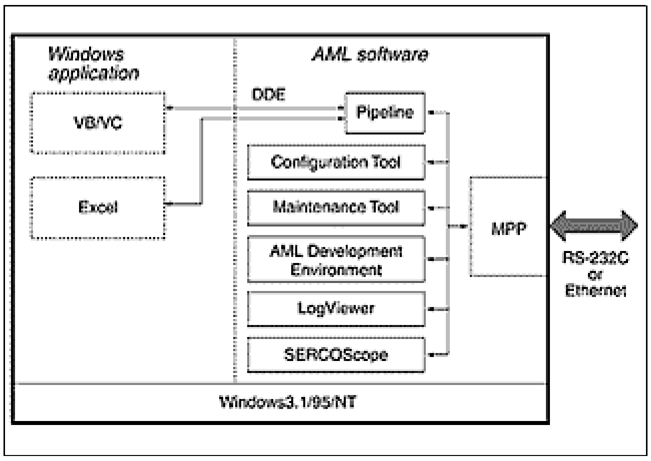

fig. 6　Target system software group

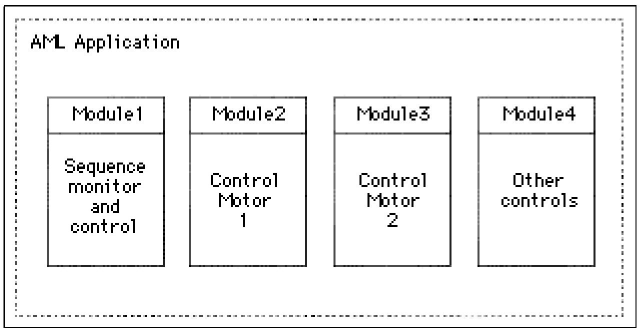

fig. 7 AML applications and modules

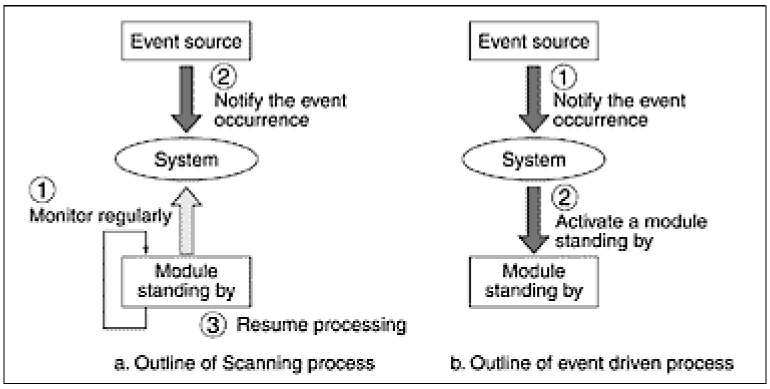
Event source
② Notify the event occurrence
System
① Monitor regularly
Module standing by
③ Resume processing
a. Outline of Scanning process
Event source
① Notify the event occurrence
System
② Activate a module standing by
Module standing by
b. Outline of event driven process

fig. 8 Differences between scanning process and event process

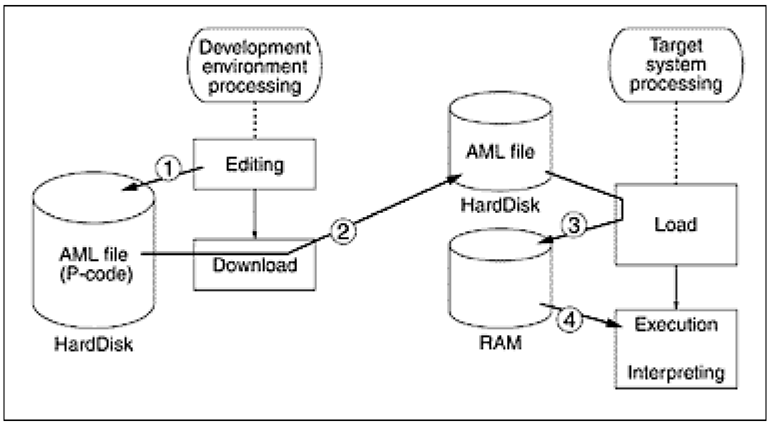

fig. 9 Development and execution procedures of an AML application

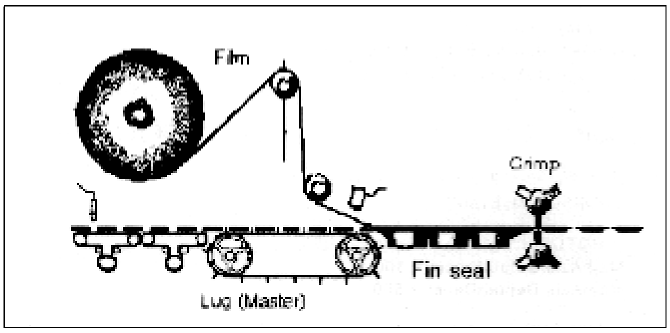

fig. 11 Packaging machine axis configuration

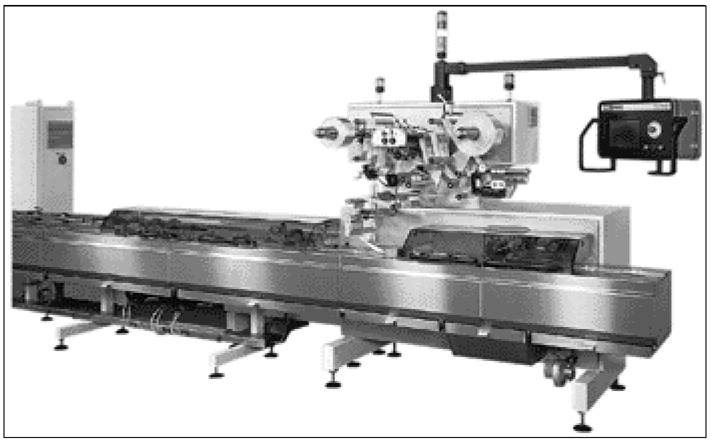

fig. 12 Outside view of packaging machine system

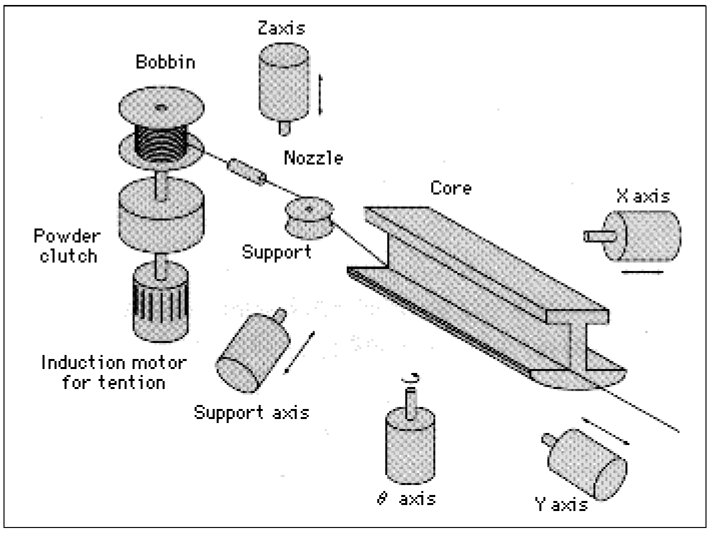

fig. 13 Wire winding machine axis configuration

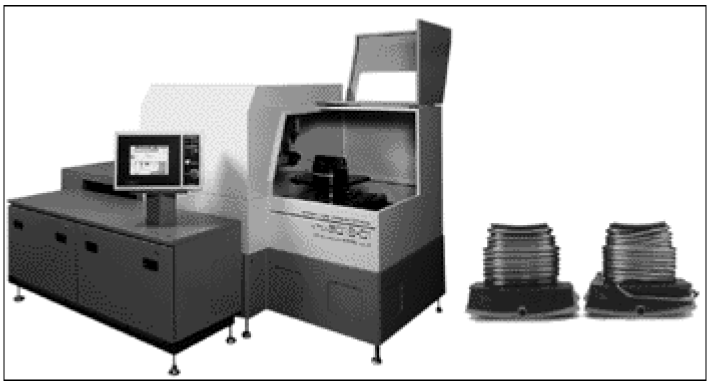

fig. 14 Outside view and fabricated samples of wire winding machine

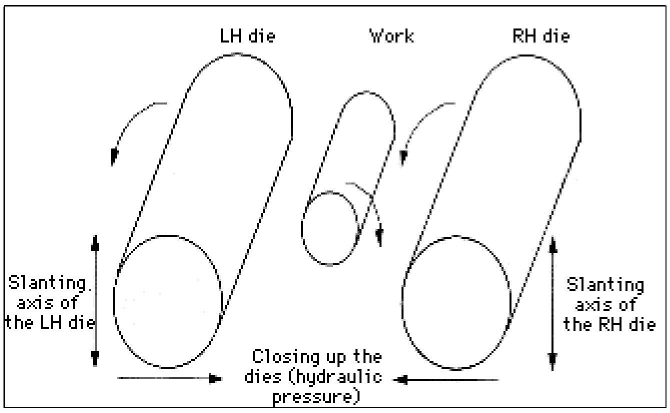
LH die
Work
RH die
Slanting axis of the LH die
Slanting axis of the RH die
Closing up the dies (hydraulic pressure)

fig. 15 Thread rolling machine configuration

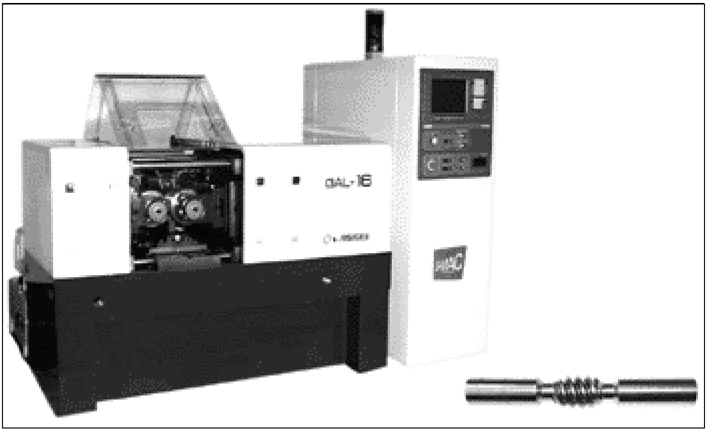

fig. 16 Outside view and a fabricated sample of thread rolling machine