공학 연구논문집 제23권 제1호 pp. 1~12. 울산대학교
Journal of Engineering Research Vol. 23, No. 1, pp. 1~12, 1992.
University of Ulsan.

Development of general programming language system for robot using a low cost microcomputer.

Won Kyu Lee
Dept. of mech. Eng.

〈Abstract〉

One of the difficulties frequently met in the current automated manufacturing system which especilly uses many different types of robot is inconsistency of the programming environments for individual robot. These environments could include many factors such as the controller type of a robot, variety of function modules provided by the controller, its operating system, programming language system used, communication capability with other machines, etc..

These could lead to many problems in system efficiency, safety, integration, cost and other aspects in the system. Modifying the existing system's characteristics into rather standardised form could reduce somewhat problematic factors.

Thus in this work with the IRB-6 robot, attempts were made to solve one of the many problems met in those environments, i.e. to develop a general programming language system for many different kinds of robots with the least possible variation, material and time consuming conditions required in robot programming.

Development of general programming language system for robot using a low cost microcomputer.

이원규
기계공학과

〈요 약〉

많은 종류의 로보트들을 사용하는 요즈음의 자동화된 제조 시스템에서 종종 볼수있는 어려움 중의 하나는 개별적 로보트에 대한 프로그래밍 환경에 일관성이 없는 것이다. 이런 환경은, 로보트 콘트롤러 형태, 그 콘트롤러에 의해 제공되는 로보트 기능 module들의 다양성, 오퍼레이팅 시스템, 사용된 프로그래밍 언어 시스템, 다른 기계들과의 통신능력 등등과 같은 많은 인자들을 포함할 수 있다.

이런 환경은 시스템 효율, 안전성, 시스템의 통합, 비용및 여러 관점들에 있어서 많은 문제점을 가져 온다. 기존 시스템의 특성을 비교적 표준화된 형태로 수정함으로서 이런 문제 요소들을 어느정도 줄일 수 있을 것이다.

따라서 IRB-6 로보트를 사용한 본 연구에서는 이런 환경에서 볼 수 있는 중요한 문제들의 하나를 해결하려 하였다. 즉 기존의 다른 종류의 로보트에서 필요로 하는 프로그래밍의 변화 및 자료, 시간 소모를 최소화하는 일반적인 프로그래밍 언어 시스템을 개발하는 것이다.

(1) Introduction.

In the current automated manufacturing systems, many kinds of robots are very often employed even in single manufacturing processes, and if the robots are different each other, the controller type and its programming methods are usually different each other. In this case not only dividing the task to be assigned to each robot but also preparing different programmes of the robots for the task are required. Because of the differences in programming methods, the robot operator must generally be required to know how to control and programme every robot involved in manufacturing. It is not easy for one operator to get used to every individual robot programming and it would take relatively long time to prepare it. It could make the operator to be confused at the programming and sometimes it could cause the safety problems. In addition, it would also be difficult to train new personnel in controlling various kinds of robots.

For these reason, generalised programming methods have been devised and tried in many ways. For example, in the case of using the high-level robot language, many different types of interpreter or compiler have been devised to produce the required robot-depenednt programme or data for the individual robot controller from the same general source programme. In other words, when robot programmes are made, the way in which they are made is always same and the programme texts themselves are almost same for different robots, but by use of different interpreter or compiler, the resulant machine code programme or data that are different in different controllers could be

produced.

However, this method is not easy to develop because of the difficulties in creating the new interpreter or compiler for many different robots. Producing different interpreter or compiler could mean making new software system of the robot. Sometimes new manipulator modules must be created and accordingly the data format which the controller sofeware understand in communication with an external device must be newly designed in order to fit the new robot language to a new compiler. That is, vast amount of work such as designing new software system for a different robot controller should be required.

In a small to medium automated manufacturing system which usually prefers to make use of low cost microcomputers, such method as mentioned above cannot be justified and probably cannot be coped with. Instead, it would be more desirable to use high-level-language-like robot programming language system that is independent of various types of the robot controller software and can be used in any microcomputer with its existing language compiler.

From this point of view the aim of this work is directed to developing a high-level-language-like robot programming language system which utilises an existing general computer programming language like 'Pascal' that can be readily available to microcomputers.

(2) Brief introduction to the IRB-6 robot used.

The IRB-6 robot system consists of 4 main parts: robot manipulator, teach pendant as a programming unit, tape recorder or external computer and robot controller. The first 3 parts are all connected to the robot controller. This system uses teach-by-showing method in producing a robot programme by use of the teach pendant which can be fitted to the control cubicle. It has orignally no other ways to produce the robot programme except the one already mentioned.

The controller panel contains only operational switches and the teach pendant extension for an operator interface to this robot. The controller can be fitted with a communication board(card) so that a robot programme can be transferred between the controller and an external device. The external device can be a tape recorder for programme storage only, or a computer which can perform communication functions with the controller as well as storing functions. This capability of the controller opens another way to develop a robot programme in addition to the teach-by-showing method.

(3) Basic requirements of robot language system and approach.

Although there are some different aspects between a robot programming language and computer language, general characteristics as a programming language are very similar to each other.

Generally drawn, required characteristics(requirements) of any robot language system can be summarised as follows.

1) Easiness of understanding the language.

2) Various computing and debugging power.

3) Adaptability and portability.

4) Extendability to other machines.

5) Various possibility of futher development of language components.

6) Generalities of language application.

With these requirements, it was tried to give maximum satisfaction to the robot language system in design by adopting one of existing computer languages and by trading-off some of them.

In other to make language design simple while achieving these requirements, one high-level computer programming language as a parent one was chosen.

The main reason why to choose one high-level computer programming language which might be considered as most appropriate is that the first 3 of these requirements could be readily satisfied by the inherent characteristics of the chosen language.

Futher 3 requirements could be satisfied by flexible adapting of the robot system software to a specific individual(different) robot.

This type of approach would reduce considerable amount of time that might have been required in developing the whole language system from a basic language component to a compiler with variation for different computers. In fact, one of the most critical and important reasons why this approach was chosen is the difficulties in developing whole language system and difficulties in finding any futher generalities of the result.

In this work, 'Pascal' was chosen as a parent language system and by utilising its structural characteristics, robot language system was designed and developed.

(4) Development of overall structure of the robot language system.

The robot language system developed comprises of largely two parts.

One is programming system part and the other is communication part.

Programming system contains a parent('Pascal') system, editor programme, data preparation programme, combining programme whose role is to generate pascal user programme from system components and user programme written in this system language and a simple debugging programme

as well as computer operating system used. Communication system comprises of uploading, downloading programmes and, robot-dependently, robot programme executing programme and some status programmes which usually depend on the robot controller communication capability.

System components existing as system files are Kernel 1(K1), Kernel 2(K2), user created files, data files, matrix data files, final user Pascal file, its compiled file and the final robot program code file. These are shown in figure 1.

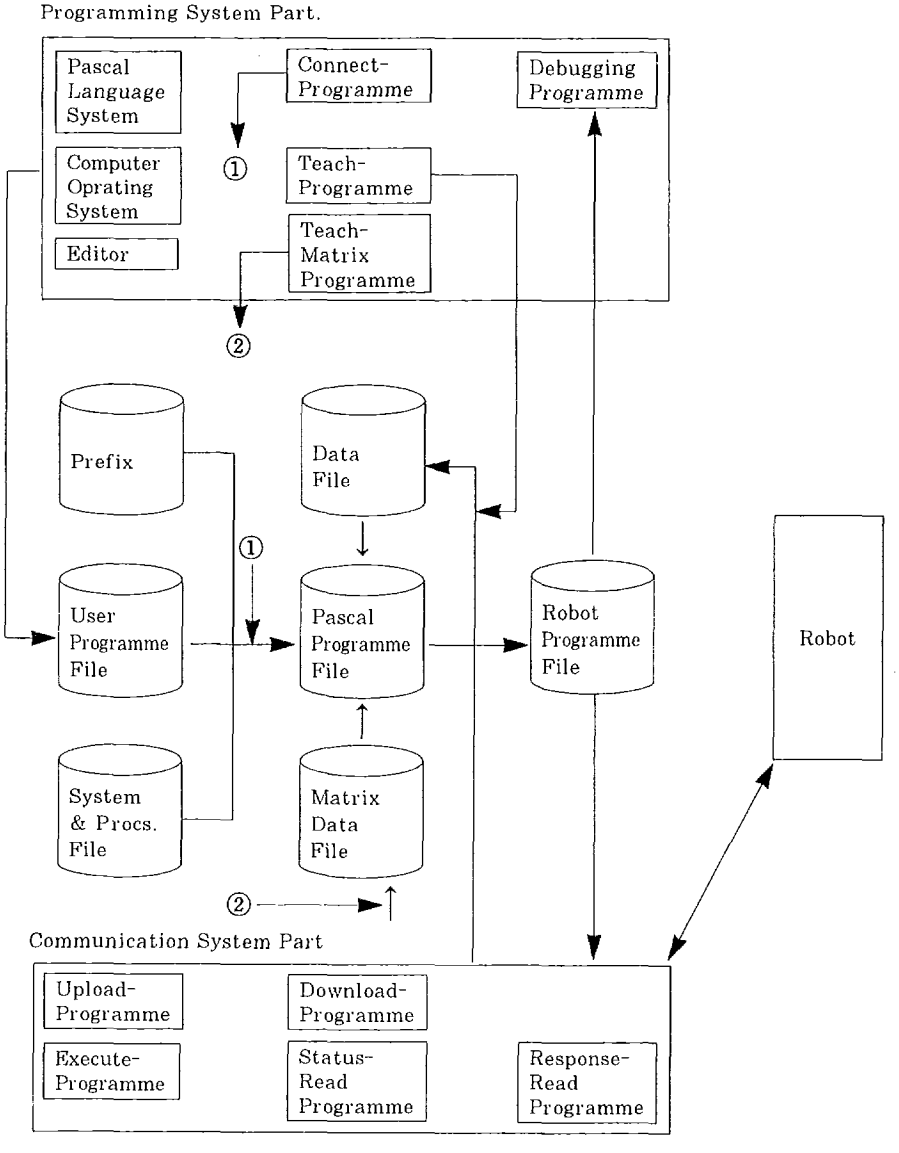

Figure 1. Overall Structure of a robot Programming system.

(5) Discriptions of system elements developed.

1. Programming system part.

1) Kernel 1. (K1)

Kernel 1 is to contain the 'Prefix' (which is specific in 'Pascal' language system that is implemented on Cortex Computer used in this work) and Constant and Type definitions which are inherent to IRB-6 robot. This part can be changed according to different robot.

2) Kernel 2. (K2)

Kernel 2 contains robot programming language components for a specific robot, i.e. robot command-like function or identifiers with its variable (according to different robot) contents. These are the very robot-dependent part which, when linked with the user programme and compiled, would enable to produce the final robot programme code.

Therefore this provides the robot programmer with great flexibility to produce different code for different robots.

Generally, it was designed to performe largely 7 different functions. They are as follows.

1. Operator interface function.
2. File interface function.
3. (Location) Data loading function.
4. (Location) Data calculation function.
5. (Location) Data transformation function.
6. Robot instruction function.
7. Prefixing and surfixing function.

3) User programme part.

This is a file in which a user produces a robot programme by use of this language system. This contains a textual robot programme. Even though this programme is written in exactly same format as Pascal programme, it cannot be understood by the Pascal compiler because it is not yet combined with other Kernels of this language system to form a final user programme.

4) Final user programme.

This programme is produced from K1, K2 and user programme part by use of the system programme called 'Connect' programme. This is a complete programme and can be understood by the Pascal compiler, and when executing its compiled file, it will produce an actual robot programme codes.

5) Connect programme.

The roll of this programme is to combine K1, K2 and user programme into one final user programme.

6) Teach programme.

By use of this programme, a user can freely produce reference points data in link with the robot controller.

7) Teach matrix programme.

This programme can produce a transformation matrix of a certain reference frame within a robot coordinate system. The product of this programme can be used by several functions listed in K2.

8) Read programme.

This is used for final robot programme code debugging.

2. Communication system part.

1) Upload-programme.

This programme is used to receive a robot programme stored in the robot controller memory through communication protocol.

2) Download-programme.

This programme is used to transmit to the robot controller a robot programme codes created or modified by an external device through communication protocol.

3) Execute-programme.

This is used to initiate the execution of the robot programme already stored in the controller.

4) Read-response or request status programmes.

These programmes can be used to receive the robot controller response in communication or can be used to request the current status of the robot. These are very limited and are considered to be robot-dependent.

These communication programmes can be used either as a part of the programming system programmes or as their own sake seperately.

(6) Application and examples.

Following figures such as figure 2 to 5 show the user programme format and simple robot programme part examples together with data preparation procedures by use of 'Teach' and 'Teach Matrix' programmes.

As seen in the format in the figures, the user programme has a complete Pascal programme format and its syntax is also Pascal syntax.

Therefore once the user is accustomed to Pascal language, it would be very easy to understand as well as to make use of this system programmes.

Format of a user programme part.

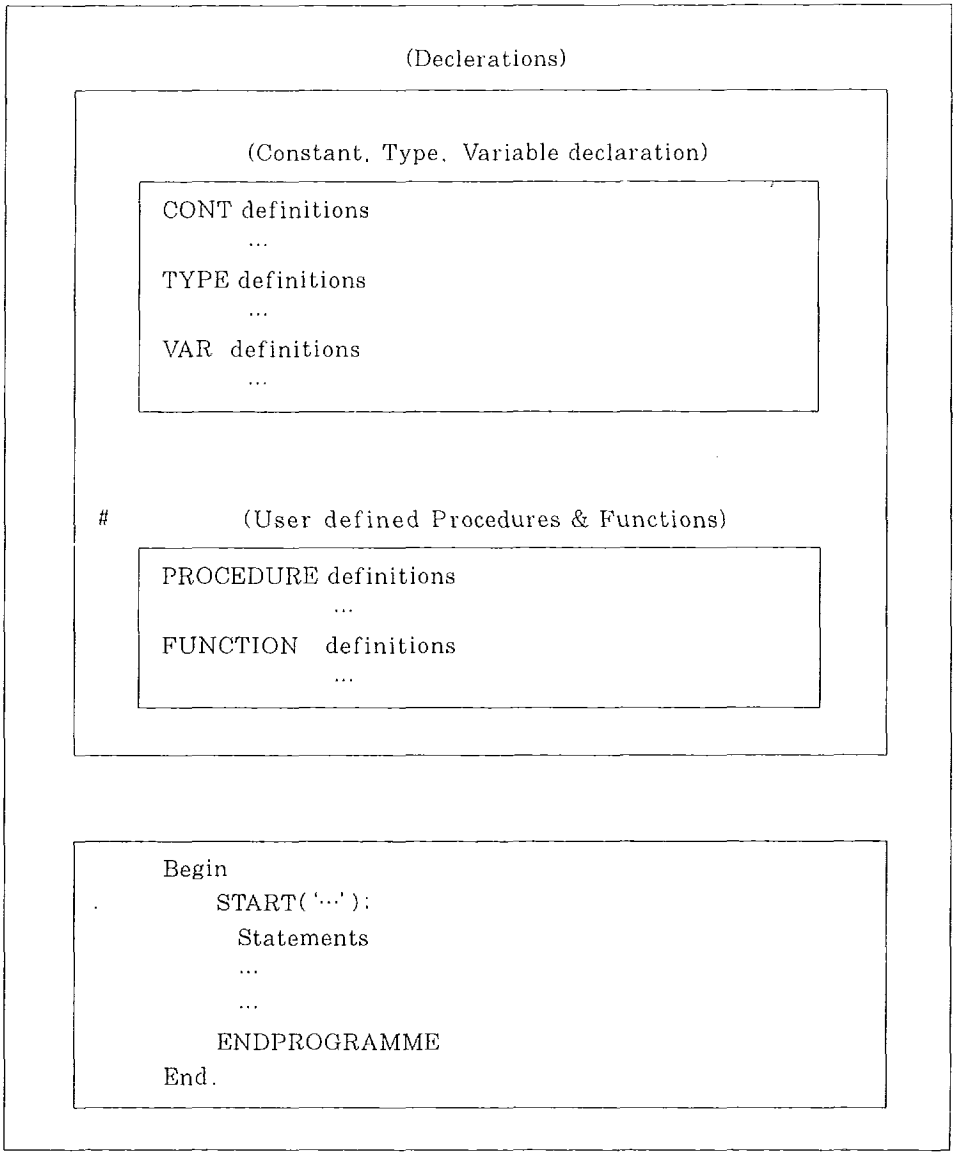

Figure 2. Format of a user(main) programme part including declerations.

```
var
  .
  .
  Station1, Station2 : Loc-Name ;
  .
```

```
Begin
  .
  .
  Station1.Name := 'Station1   ';
  Station2.Name := 'Station2   ';
  Loc_Data('2/RobotAreal', Station1);
  Loc_Data('2/RobotAreal', Station2);
  .
  .
```

Preparation Section

```
  .
  .
  MOVE(10, Station1, 5) ;
  MOVE(20, Station2, 2) ;
  .
  .
```

Robot Command Section

```
End
```

(Main Programme Block)

Figure 3. Example of a detailed user programme block devided into two large sections.

```
var
   .
   .
   location : Loc_Name;
   .
   .                          (Main Programme Block)
```

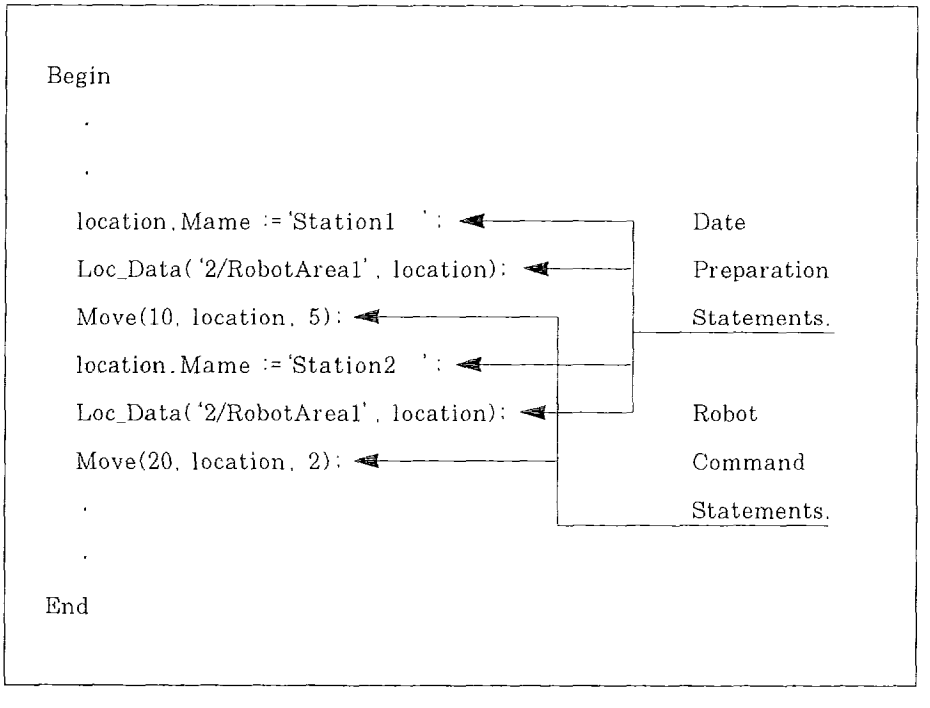

Figure 3. Example of a user programme part which has the same structure as that of the Pascal programme.

```
var
    .
    .
    location : Loc_Name;
    .
    .                          (Main Programme Block)
```

```
    .                                                    .
Begin
  START('2/RobotProg');                          Robot Command
    .
    .
      location.Mame := 'Station1   ';            )   (  Date
      Loc_Data('2/RobotArea1', location);        )   (  Preparation
    Move(10, location, 5);                           Robot Command
      location.Mame := 'Station2   ';            )   (  Data
      Loc_Data('2/RobotArea1', location);        )   (  Preparation
    Move(20, location, 2);                           Robot Command
    .
    .
  ENDPROGRAM                                     Robot Command
End
```

Figure 5. Example of another format of a user programme part.

(7) Conclusion.

It is estimated that most of the requirement in a robot language system could be met by this approach. In addition, this is a low cost microcomputer based system and parent language based system, the language system's cost effectiveness and portability to another micro-computers can be easily achieved. And because system's Kernels can be developed for different robot, adaptability as well as extend-ability independent of many different robot types could be also achieved, while the language syntax and its programming patterns remains almost same, which provides eventually vital importance of robot programming language generality. This could reduce considerable amount of user efforts in various programming environments of a complex manufacturing system.

Incoporating with a computer graphics system would enhance the merit of this system.

References.

1. ASEA Industrial robot system. Information YB 110-317E YFB. June 1977. Reg. 6397. Edition 2.
2. ASEA Computer Link Industrial robot system. Information YB 110-317E YFB. Sept 1979. Reg. 6397.
3. Brinch Hansen, Per. The Architecture of concurrent programmes. Prentice Hall, Englewood Cliffs, New Jersey, 1977.
4. Rodnay Zaks. Introduction to Pascal(including UCSD Pascal) Second edition revised. Sybex 1981.
5. MDEX for the Cortex-User manual. Rev. 1.1 Microprocessor Engineering Ltd. June 1983.
6. John Walker. Marinchip 9900 Pascal User Guide. Marichip System. Mill Valley. Oct 1980.
7. Peter B. Scott. The robotics revolution. Basil Blackwell. 1984.