Control of an Industrial Robot using Ethernet - Application to a V+ Running Controller

Wassim Mansour[1], Khaled Jelassi[1], Imen Chaieb Memmi[2]
[1]Deptartment of Electrical Engineering
ENIT
[2]ESTI
[1,2]Tunis, Tunisia

ABSTRACT: *Control of flexible manufacturing systems requires the adaptation of the software components to the constraints imposed by the hardware components. As a key element of the control of flexible manufacturing systems, the control of an industrial robot is an essential first step towards the control of a whole cell. This step provides a wide field of investigation given the multitude of algorithmic combinations leading to the control of the robot. On the other hand, the optimization of such a command is an important task because of the effects it may have on the command of the whole cell. In this paper, we propose an algorithmic approach for the control of an industrial robot. Our approach is based on a dynamic data exchange (DDE) between the control and the operative part of the system. The exchanged data allows the control of the system and the information about its status. This exchange needs synchronization between the various software components of the system. The approach is implemented through a win32 software allowing the manipulation of an Adept Cobra 600 industrial robot.*

Keywords: Robotics, Robot Control using Ethernet, DDE, V+ Language

Received: 2 May 2013, Revised 14 June 2013, Accepted 20 June 2013

1. Introduction

In this work, we are interested in the control of a single robot arm via the Ethernet network. We present a model of communication between the control and the operative parts. The control part consists of a server machine running a win32 platform (Microsoft Windows). The operative part is a robot controller running a specific operating system (V+ system). The V + operating system has not been the subject of much research. Still, some work has been devoted to communication with controller running a V+ system. In [3], the authors present a method and related V+ algorithm for visual robot guidance in tracking objects moving on conveyor belts. Other authors [4] have presented a framework of an online knowledge based error recovery system for robotized assembly. The implementation of the system was performed on controllers running a V + operating system.

The model developed is implemented through an Adept Cobra 600 robot, which is a SCARA robot (Selective Compliance

Assembly Robot Arm). SCARA robots have usually four axes: Axis 1, 2 and 4 are rotations and axis 3 is a translation. Figure 1 presents a description of the positions of the Adept Cobra 600 robot's axes. On the industrial point of view, these robots are widely used in the operations of pickand- place, painting, brushing, drilling...

The data exchange is done via Ethernet using a TCP/IP protocol. DDE communication is established to allow such an exchange. DDE is a technology for interprocess communication under Microsoft Windows. It has existed since the early versions of Windows. It was first introduced in 1987 with the launch of Windows 2.0 as a method of interprocess communication so that a program can communicate with or control another program. This protocol is integrated to most software because of its respectable real-time benefits. This protocol is typically implemented as an exchange of data between different applications running under windows [1] [5].

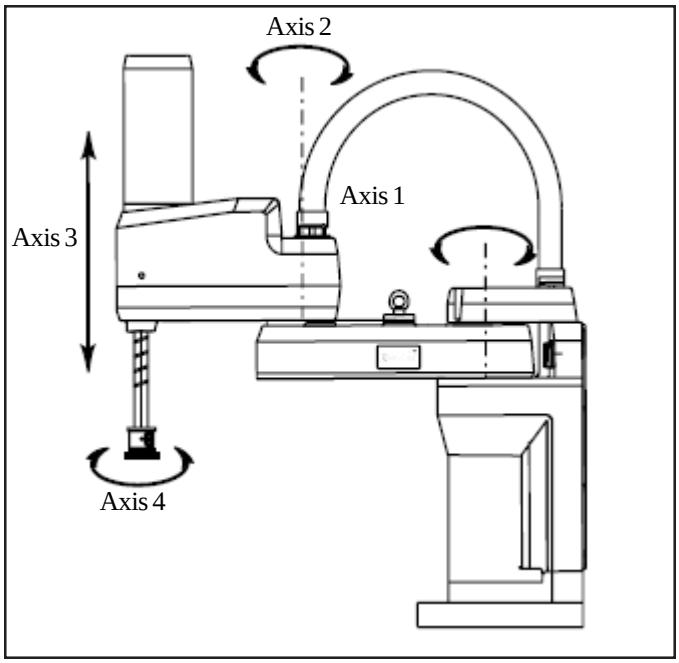

Figure 1. Axes positions of a SCARA robot (Adept Cobra 600)

2. Robot Control Model

The system considered in this model is essentially composed of two elements interacting in a client / server architecture:

• **The control part:** A personal computer operating in a Microsoft Windows environment and acting as a server.

• **The operative part:** The controller of the industrial robot running a specific operating system (depending on the manufacturer of the robot). In this work, we consider a controller having Adept V+ as an operating system.

In this model, the data exchanged are of two types:

• **Control data:** These data are supposed to cause the starting or the stopping the movement of the robot along one or more axes. These data are transmitted to the robot controller as control variables (*CVi*). These variables represent the input data for a set of algorithms responsible for communicating instructions directly to the robot (Figure 2).

• **Status data:** These data are transmitted from the controller to the computer in the form of DDE requests. These variables (*SVi*), among others, allow the validation of the control data reception (Figure 2).

All the data exchanged between the control and operative parts is defined in the algorithm managing the DDE communication. This algorithm is written in the V+ language and has to be running on the controller's operating system during the communication.

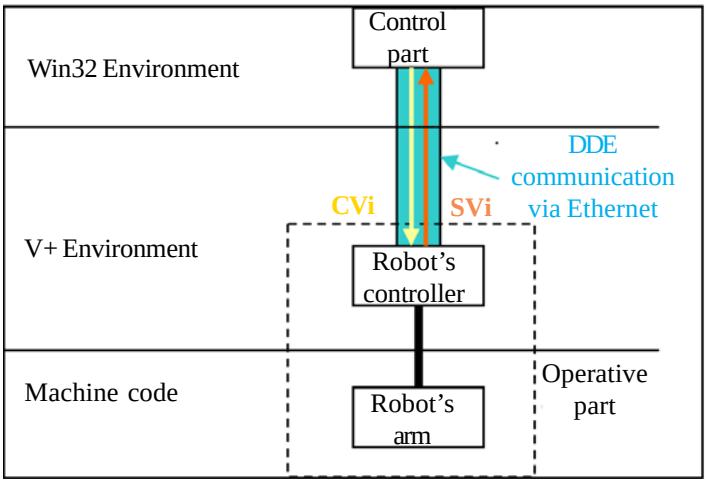

Figure 2. Robot's command model

DDE communication is achieved through two key elements (see Figure 3):

• A server application that acts as a link between the TCP/IP protocol and the DDE service (running as a task in the Windows environment): The application AdeptWindows DDE.

• A V+ algorithm that transcripts data received via the Ethernet port into values, as well as sending variables (runs as a task on the controller's operating system): V+ custom program (DDE.V2).

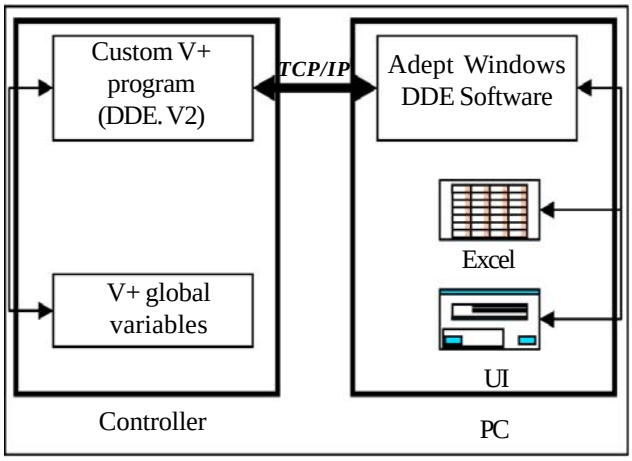

Figure 3. DDE mechanism for Adept controllers

2.1 Model's control part

The control part, which consists of software components, is responsible for the transmission of various instructions from the user to the robot controller. It allows, via the user interface, the visualization of system's status (position, speed, time, etc) and the returning of the status of the system.

A user interface is developed using VBA language. This interface allows the user to have full control over the device, record the coordinates of specific points and edit sequences of actions. Functions offered to the user such us moving the robot arm towards a particular direction or changing the robot's speed consist of assigning values to some DDE variables which causes the intended. DDE variables are declared in the program DDE.V2 using a table with 4 columns (4 one-dimensional tables in V+ language) (Table 1).

Data transmitted to the controller trigger the corresponding actions upon receipt by programs running on the controller.

	Table 1	Table 2	Table 3	Table 4
Name	$Name[]	$Read[]	$Write[]	Poll[]
Type	String	String	String	Real
Description	Nom of the link item (DDE variable)	Name of the V+ variable or expression that is passed back to the $name Link Item	Name of V+ variable that the Link Item value is written to	Polling time, in seconds
Example (time since boot)	Time_since _boot	TIMER(-3)	-	0

Table 1. DDE Variables' Structure in DDE.V2

Data transmission requires first opening a communication channel with the DDE server:

Application.DDEInitiate app: = *"Adept "*,
topic: = *MV_Name*

Data transmission is done using the following VBA statement:

Application.DDEPoke channel: = *"Channel number"*,
Item: = *"DDE variable" data*: = *"value"*

On the other hand, the control part receives data about the status of the system such as the position of the robot, the configuration status; the time since boot ... These data are obtained from the variables declared in the program DDE.V2. Calling these data is done using the following VBA statement:

VBA.variable = Application.DDERequest
channel: = "*Channel number*", *Item*: = "*DDE variable*"

The architecture of the control part is shown in Figure 4.

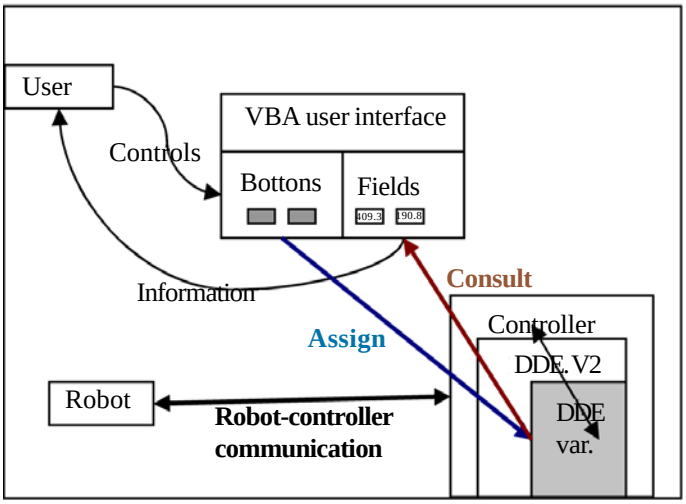

Figure 4. Architecture of the command part

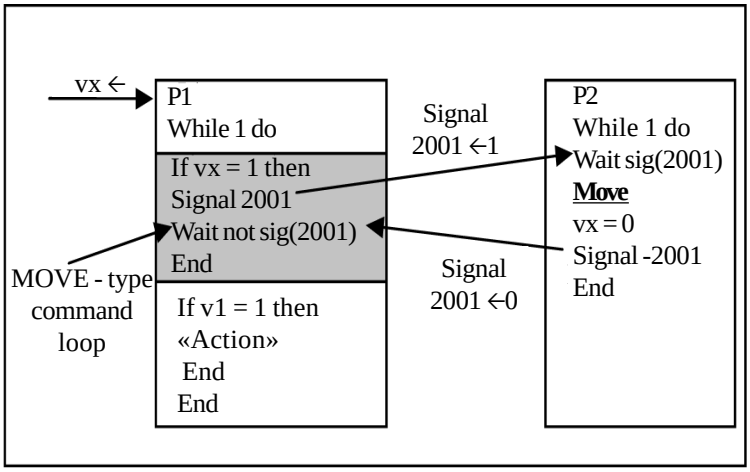

Figure 5. Interaction between the operative part programs

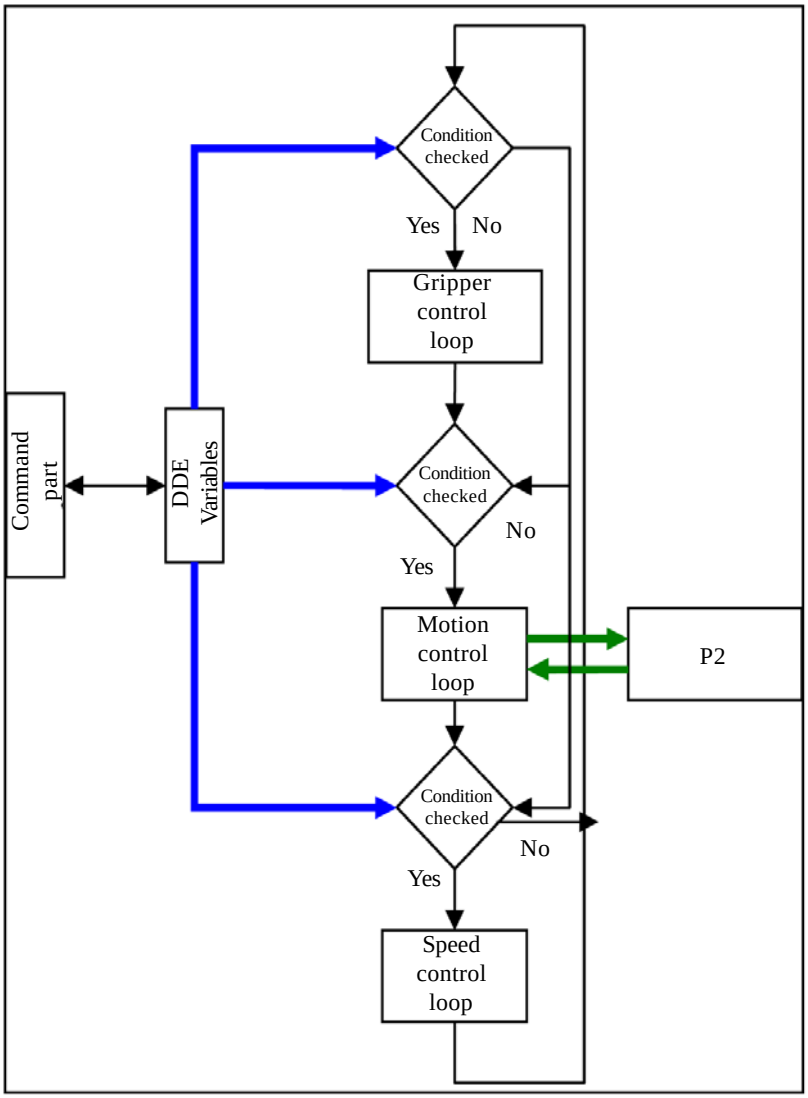

Figure 6. Flowchart of "*P*1"

2.2 Model's operative part

The considered operative part consists of hardware and software components interacting in order to execute user's commands. Otherwise, the components that transform data transmitted to the controller into actions executed by the robot's arm.

Data transmitted to the controller consist of assigning values to DDE variables. In the operative part, this event should trigger the corresponding action. This is achieved through two programs running in parallel (the V+ environment is multitasking):

• The first program "*P*1" consists of an infinite loop that continuously checks the status of DDE variables. Upon each difference with the reference state, the program executes an action corresponding to the new state of the variable.

• The second program "*P*2" is an algorithm in the standby state (V+ instruction *WAIT*). An analog signal is set to active state by the "*P*1" program when a MOVE-type command is received (moving the robot arm). At this point the program "*P*2" actually performs the *MOVE* command and returns the signal to idle state. Figure 5 shows the model of interaction between these two programs.

The "*P*1" program can also run other commands offered by the VBA user interface:

- Changing the robot's speed.
- Controlling the robot's gripper (pneumatic action).

The flowchart of the program "*P*1" is illustrated by Figure 6. The program "*P*1" consists of number of loops triggering specific actions each one (Figure 5). Other authors [1] propose other kind of loops (based on "*Case OF* " structure) to achieve the same purpose. This solution has been compared to ours and gave us similar results.

The operative part consists essentially of semaphores that can make tasks faster. Triggering an event having the robot arm as resource indirectly sets other events into idle status. Thus, it is not possible that several events can have the robot arm as a resource at the same time. Such techniques have been widely used for the control of industrial robots as they guarantee a simple synchronization between tasks using the same resource. Bison & Gini [2] have used such a technique to manage access to a common space (single resource) between two robots. This allowed them to avoid possible collisions between two robots.

3. Conclusion

In this paper we have presented a communication model with an industrial robot using Ethernet. The trigger controls is made through DDE variables that allow interaction between the control and operative parts of the system.

This work is as a basis for the development of a multi-agent approach to control a flexible manufacturing cell.

References

[1] Norberto Pires, J., Sá da Costa, J. M. G. (2000). Object-oriented and distributed approach for programming robotic manufacturing cells, *Robotics and Computer Integrated Manufacturing*, 16, 29-42.

[2] Bison, P., Gini, M. (1989). An object-oriented approach to robot programming, *Computer-Integrated Manufacturing Systems*, 2 (1) 29-34 February .

[3] Borangiu, Th., Anton, F. D., Anamaria Dogar. Visual Robot Guidance in Conveyor Tracking with Belt Variables, *International Conference on Automation, Quality and Testing, Robotics*. 1, 1-6.

[4] Wenhan, Q., Lie, M. (1991). Knowledge based error recovery in robotized assembly, *IEEE/RSJ International Workshop on Intelligent Robots and Systems lROS* '91. Nov. 3-5, p. 1344-1349.

[5] Mo, J. P. T., Tang, B. C. K. (1998). Petri net modelling and design of task oriented messaging system for robot control, *Computers & Industrial Engineering*, 34 (4) 729–742, September.