16 Apr 1992

Computer Software Control of a Three-Joint Model Robot

John W. Fierke

COMPUTER SOFTWARE CONTROL OF A THREE-JOINT MODEL ROBOT

John W. Fierke
Department of Electrical Engineering

ABSTRACT

The general scope of the computer program to control a three-joint model robot is presented. Several existing and potential sources of difficulty are also mentioned. For all specific explanations the reader is referred to the thesis papers of Richard Wainwright [1] and Eric Stelzer [2]. This presentation merely adheres to the description of the software to control the robot using the existing information contained in these two papers

INTRODUCTION

The purpose of this project was to make the Electrical Engineering Department's Fishertechnik model robot functional through the use of an IBM PC. It was desired to have it operate as it was originally designed to operate on an Apple microcomputer, as described by Richard Wainwright [1] and Eric Stelzer [2].

The software control of the robot is the primary concern of this paper. However, the software greatly hinges upon the design of the existing auxiliary interface card, which has seemed to pose the majority of the difficulty of this project. This card was not accompanied by any documentation of any sort, no schematic, no description. The greatest concern of the author was to discover how the edge connectors of the card interfaced with the connectors on the card slots of the IBM, and also to learn the actual addresses of the slot connectors so as to be able to communicate with the card (This has been postponed as the author has been on co-op, away from all the materials).

The card's primary component is an ADC0808CCN Analog-to-Digital converter made by National Semiconductor Corporation [3]. Its function is to convert the voltages across the potentiometers on the joints of the robot into numerical values corresponding to positions of the arms.

Along with these pieces of information and the descriptions by Wainwright[1] and Stelzer[2], it was decided to move on and devise a "skeleton" program to handle the desired operations of the robot without actually knowing the true addresses of the interface card. It essentially "steps through" all the processes for the sake of having a base to build upon once the addressing problem is conquered. Some small, specific subroutines have been created to manipulate non-critical, unused memory locations,

treating them as the actual output and input addresses. These subroutines simulate the robot's motion and provide all the necessary values that would be experienced, were there an actual robot attached, so that the various calculations can be performed and verified. This simulation is still under development and is useful only for the present until time can be devoted to the robot itself at the lab in Rolla. Only then can the pertinent modifications be made and the true results observed.

Turbo Pascal was chosen as the programming language for its structure and high level language characteristics in order that the program could be more easily altered or updated in the future.

Although the primary function of the model robot is to act in a "machine vision" capacity; identify an object on a conveyor belt, calculate the point of intersection, move to pick it up, and deposit it in a corresponding bin; it should also be capable of performing several specific operations and the user should have control of most of its parameters such as speed, direction, and position and be able to either provide a finite set of points through which to move the robot or have the computer execute a calculated set of moves to reach a desired position.

Following is a general description of the design and operation of the Robotics Command System (RCS) and some of its existing and potential disadvantages.

THE ROBOTICS COMMAND SYSTEM
ROBOT STATUS WINDOW

In keeping with Stelzer's design, most of the features he mentions as being part of his Robotics Operating System (i.e. MOVE, LEARN, CALIBRATE, Path Segments, Path Buffers, and Speed Control) have been implemented with various modifications. This program is not be command-line based but resembles the display in Figure 1.

This is the operating environment for the user and allows for complete visualization of all of the robot's present parameters and the demands put upon it. The top portion, the Robot Status Window (RSW), remains fixed at the top of the screen and is constantly updated to relate the robot's present values. Following are the parameters displayed in the RSW :

Mode

Mode depicts that the hardware controller switch is either in the Computer (C) position, which transfers control to the computer, or in the Manual (M) position, allowing manipulation of the robot with the controls on the hardware controller. The particular mode that it is in is checked upon every attempt to output or input information so as to determine if the computer can proceed with its operation or not. The corresponding letter in the display will be highlighted.

```
*****************************************
*                                       *
*    ROBOTICS COMMAND SYSTEM (RCS)      *
*                                       *
*****************************************

*************************************************************************
*                                                                       *
*  MODE : C/M  POSITION  Cart : (x)___ (y)___ (z)___                    *
*                        Joint : (1)___ (2)___ (3)___    MAGNET : ON/OFF *
*               SPEED (%)      : (1)___ (2)___ (3)___                  *
*                                                                       *
*************************************************************************
```

#	Command	Coordinates	System	Speed	Options	Elbow	Time
1	MOVE	0,0,0	J	100		RU	
2		12,5,-10	C	50	H400		
3	LPICK	0,0,0	J				
4		15,4,15	C			LD	
5		0,0,0	J	75			
6	E						
7	CALIB						
8	A						

Figure 1. The Robot Command System Window

Position

This portion of the RSW displays the position of the manipulator as it moves in both Cartesian and joint coordinates. The Cartesian coordinates; x, y, z; are expressed in centimeters and the joint coordinates; J1, J2, J3; in degrees. The joint coordinates are obtained directly from reading the position of the three pots and converting the resulting values (each can range from 0 to 255) into degrees by applying a conversion factor of 340 degrees/255. (340 degrees due to the physical limitations of the joints as explained by Stelzer[2]. This RSW feature has the definite potential of slowing down the program and thus the smooth operation of the robot, because upon each sampling of the robot's position, the computer must convert the positions into degrees, convert the degrees into Cartesian values, and print all six values on the screen.

Speed

It is intended to also display the actual speed of each joint as small compensations are made to correct the position of each joint due to the effects of friction and gravity. Constantly updated values such as speed and position may have to be changed to be updated only once for a certain number of inputs to speed up operation. The speed will initially be determined by the user, but in given time intervals throughout the movement of the manipulator the computer will compare the robot's present position (obtained by reading each joint's potentiometer voltage) to its expected position (calculated using the equation $p_k = v_k*t + c_k$, in the manner described by Stelzer[2]; p is the

expected position of joint k, v_k is the newly calculated velocity of joint k, t is the time elapsed, and c_k is the starting position of joint k.) and make appropriate speed adjustments to each motor. In this manner the robot will best reach its destination smoothly and as quickly as possible.

Magnet

This information is merely a means of identifying at a certain moment if the manipulator's electromagnet is activated or not, as is instructed by the user.

THE ROBOTICS COMMAND SYSTEM
ROBOT COMMAND WINDOW

In this portion of the screen the user is allowed to command the robot to complete a certain action while also providing various options. Each request is contained on one line of the RCS table as shown in Figure 1. For example, for command #1, the user would enter 'MOVE', tab to the next column, enter a set of coordinates '0, 0, 0', tab, signify which coordinate system was to be used 'J' for Joint coordinates, tab, provide the desired percent of full speed '100', tab, any other available options, tab, the elbow configuration 'RU'-Right elbow Up, then press Enter to execute the command or tab to the next line, in order to make a list of commands to execute later.

At this point the most important aspects of the RCW are Command, Coordinates, System, and Speed. Options, Elbow, and Time will be developed at a later time.

Command

Currently, the user can enter one of four instructions in the Command column-Move, L(earn), or Calib(rate) and A(uto).

Move

The Move command is used to initiate a move of the manipulator from its current position to a designated point specified in either Cartesian or Joint coordinates. If desired, the Move command may be omitted, as is shown on lines 2, 4, and 5 in Figure 1, and CS will take for granted that a move is desired since this will be the most commonly used command.

Learn

The Learn command is used in such a manner as to create a file on disk in which to store a series of commands to execute at any later time. The user merely follows the L with the desired filename in which to store the upcoming commands. Some or all the moves may be designated by switching the pulse-width modulator to the manual mode and positioning it using the controls on the box. By merely pressing enter, the present coordinates of the manipulator will be stored and printed on the appropriate line. The user may end the Learning session by entering an 'E' for Exit in the command column, upon which the file is stored on disk. Then to initiate those stored moves, the user must enter the name of the file in the command column.

Calibrate

This command is used for designating the "zero" position of the robot. When the command is entered the user must manually position the robot in the desired "zero" position and press enter to identify the robot's position to the computer. This position will replace the previous calibrated position and will be referenced upon each startup until it is re-calibrated.

Auto

When the user enters an 'A' in the command column this sends the robot into fully automatic operation, or "Machine Vision" mode as mentioned previously, in that the robot is only actuated upon the breaking of the light beam to the first photo cell on the conveyor belt. In this case, the inputs from the photo cells are considered and the path to the pickup point and the drop bins are stored in a special predetermined file.

Coordinates and System

The user may specify his coordinates for a move in either Cartesian or Joint coordinates by merely placing either a C or a J in the Coords column. The program deals directly with coordinates in degrees, so no conversion is necessary for Joint coordinates. If Cartesian coordinates are specified, these x-y-z values are converted to Joint coordinates through the use of the solutions of the Kinematic equations as described by Stelzer [2].

Speed

This option allows the user to set the overall operation speed of all three motors. For example, if the speed of motor 1 is 100% of full speed, motor 2 is 30%, and motor 3 is 80%, a speed value of 100 would maintain these speeds, but, for instance, a value of 50 would slow each motor to half its set speeds (i.e., motor 1-50%, motor 2-15%, motor 3-40%). When omitted, the default value used will be 50%.

The author regrets how unspecific this report may appear, but the majority of the material has already been stated by either Wainwright [1] or Stelzer [2]. The extent of this author's work was in designing and programming the software to accomplish the tasks mentioned. It was believed that discussing the program code would not prove to be useful in a report of this nature. Hopefully, the program will soon speak for itself.

ACKNOWLEDGMENTS

I wish to express my thanks to Dr. Randy Moss for the time and effort he offered. I would also like to thank Mr. Bob Dopher and Mr. Jim Ross for their extensive technical advice.

REFERENCES

1. Stelzer, E. H., "A Control Program and Operating System for a Three Joint Robot Arm on an Apple Microcomputer," M.S. Thesis, University of Missouri-Rolla, 1982.

2. Wainwright, R.E., "Microprocessor Control of a Model Robot System using Programmable Pulse Width Modulation," M.S. Thesis, University of Missouri-Rolla, 1982.

3. National Semiconductor Corporation, Semiconductor Databook, The ADC0808CCN Analog-to-Digital Converter, p. 48-58, 1989.