RSD-TR-1-83

A GEOMETRIC APPROACH IN SOLVING THE INVERSE KINEMATICS OF PUMA ROBOTS

C. S. G. Lee

M. Ziegler

Department of Electrical and Computer Engineering

The University of Michigan

Ann Arbor, Michigan 48109-1109

December 1983

CENTER FOR ROBOTICS AND INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109-1109

TABLE OF CONTENTS

ABSTRACT

This paper presents a geometric approach to derive a consistent joint solution of a six-joint PUMA[1] robot. The approach calls for the definition of various possible arm configurations based on the link coordinate systems and human arm geometry. These arm configurations are then expressed in an exact mathematical way to allow the construction of arm configuration indicators and their corresponding decision equations. The arm configuration indicators are prespecified by a user for finding the joint solution. These indicators enable one to find a solution from the possible four solutions for the first three joints, and a solution from the possible two solutions for the last three joints. The solution is calculated in two stages. First a position vector pointing from the shoulder to the wrist is derived. This is used to derive the solution of the first three joints by looking at the projection of the position vector onto the $\mathbf{x}_{i-1}$-$\mathbf{y}_{i-1}$ ($i = 1,2,3$) plane. The last three joints are solved using the calculated joint solution from the first three joints, the orientation matrices, and the projection of the link coordinate frames onto the $\mathbf{x}_{i-1}$-$\mathbf{y}_{i-1}$ ($i = 4,5,6$) plane. From the geometry, one can easily find the arm solution consistently. Computer simulation study conducted on a VAX-11/780 computer demonstrated the validity of the arm solution.

This work was supported in part by the Robot Systems Division of the Center for Robotics and Integrated Manufacturing (CRIM) at The University of Michigan, Ann Arbor, MI. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agency.

[1] PUMA is a trademark of Unimation Inc.

1. INTRODUCTION

An industrial robot is a general purpose manipulator having several rigid links connected in series by revolute or prismatic joints driven by actuators. One end of the chain is attached to a supporting base while the other end is free and attached with a tool to manipulate objects or perform assembly tasks. The motion of the joints results in relative motion of the links. Since the robot servo system requires the reference inputs to be in joint coordinates and a task is generally stated in terms of the Cartesian coordinate system, controlling the position and orientation of the end-effector of a robot arm to reach its object requires the understanding of the kinematic relationship between these two coordinate systems.

The kinematics problem usually consists of two subproblems - the direct and inverse kinematics problems. The direct kinematics problem is to find the position and orientation of the end effector of a manipulator with respect to a reference coordinate system, given the joint variable vector $\mathbf{q}^T = (q_1, q_2, \cdots, q_n)$ of the robot arm and the various geometric link parameters, where superscript T on vectors and matrices denotes a transpose operation, and n is the number of degree-of-freedom. The inverse kinematics problem (or arm solution) is to calculate the joint variable vector $\mathbf{q}$ for positioning the end-effector of the robot arm at the desired position with the desired orientation, given the position and orientation of the end effector with respect to the reference coordinate system and the various geometric link parameters. This paper is concerned with the inverse kinematic analysis of simple manipulators consisting of six rotary joints.

In general, the inverse kinematics problem can be solved either by algebraic, iterative, or geometric approach. Several investigators have attempted to solve the problem for the PUMA and Stanford robot arms using the algebraic approach [1]-[6]. This approach suffers from the fact that the solution does not give a clear indication on how to select the correct solution from the several possible solutions for a particular arm configuration. The user often needs to rely on his/her intuition to pick the right answer. The iteration solution [7-8] often requires more computations and it does not guarantee convergence to the correct solution, especially in the singular and degenerate cases. Furthermore, there is no indication on how to choose the correct solution for a particular arm configuration.

If the manipulator under consideration is simple, that is the geometry of the first three joints has revolute or prismatic pairs and the last three joint axes intersect at a point [1], then the geometric approach presents a better approach for obtaining a closed form solution. This paper presents a geometric approach in solving the inverse kinematics problem of a simple robot arm with rotary joints. The approach calls for the definition of various possible arm configurations based on the link coordinate systems and human arm geometry. These arm configurations are then expressed in an exact mathematical way to allow the construction of three arm configuration indicators (ARM, ELBOW, and WRIST) and their corresponding decision equations. With the assistance of the configuration indicators and the arm geometry, one can easily find the arm solution consistently. The validity of the arm solution was simulated on a VAX-11/780 computer. With appropriate modification and adjustment, the user can generalize and extend the method to most present day industrial robots with rotary joints and obtain the arm solution easily.

2. LINKS, JOINTS, AND COORDINATE TRANSFORMATION

To describe the translational and rotational relationship between adjacent links, a Denavit-Hartenberg matrix representation [9] for each link is used and shown in Figure 1. From Figure 1, an orthonormal coordinate frame system ($\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i$) is assigned to link i, where the $\mathbf{z}_i$ axis passes through the axis of motion of joint $i+1$, and the $\mathbf{x}_i$ axis is normal to the $\mathbf{z}_{i-1}$ axis pointing away from it, while the $\mathbf{y}_i$ axis completes the right hand rule. With this orthonormal coordinate frame, link i is characterized by two parameters: a_i, the common normal distance between the $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ axes and α_i, the twist angle measured between the $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ axes in a plane perpendicular to a_i. Joint i which connects link i-1 to link i is characterized by a distance parameter d_i measured between the $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ axes and a revolute joint variable θ_i which is the joint angle between the normals and measured in a plane normal to the joint axis. If joint i is prismatic, then it is characterized by an angle parameter θ_i and a joint variable d_i.

Once the link coordinate systems have been established for each link, a homogeneous transformation matrix, $\mathbf{A}_{i-1}^{i}$, can easily be developed relating the i^{th} coordinate frame to the i-1^{th} coordinate frame. Using the $\mathbf{A}_{i-1}^{i}$ matrix, one can relate a point $\mathbf{p}_i$ at rest in link i and expressed in homogeneous coordinates with respect to the i^{th} coordinate system to the i-1^{th} coordinate system established at link (i-1) by:

$$\mathbf{p}_{i-1} = \mathbf{A}_{i-1}^{i}\ \mathbf{p}_i \tag{1}$$

where $\mathbf{p}_{i-1} = (x_{i-1}, y_{i-1}, z_{i-1}, 1)^T$; $\mathbf{p}_i = (x_i, y_i, z_i, 1)^T$;

$$\mathbf{A}_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \text{ for } \textit{a rotary joint} \tag{2}$$

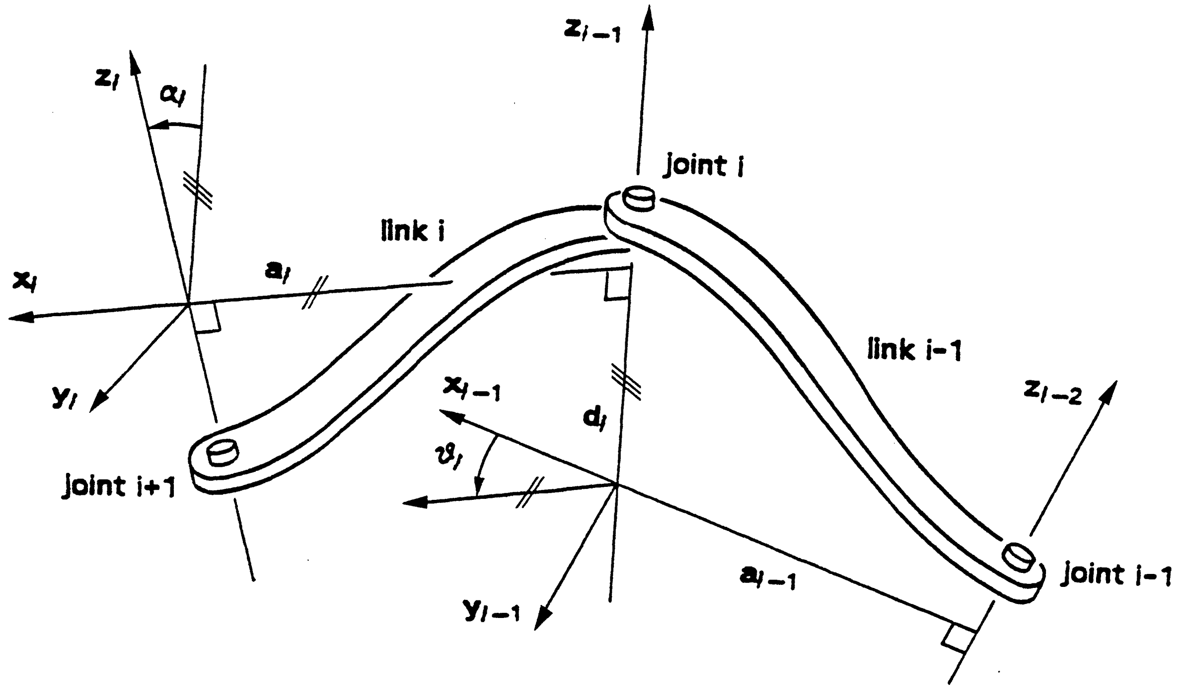
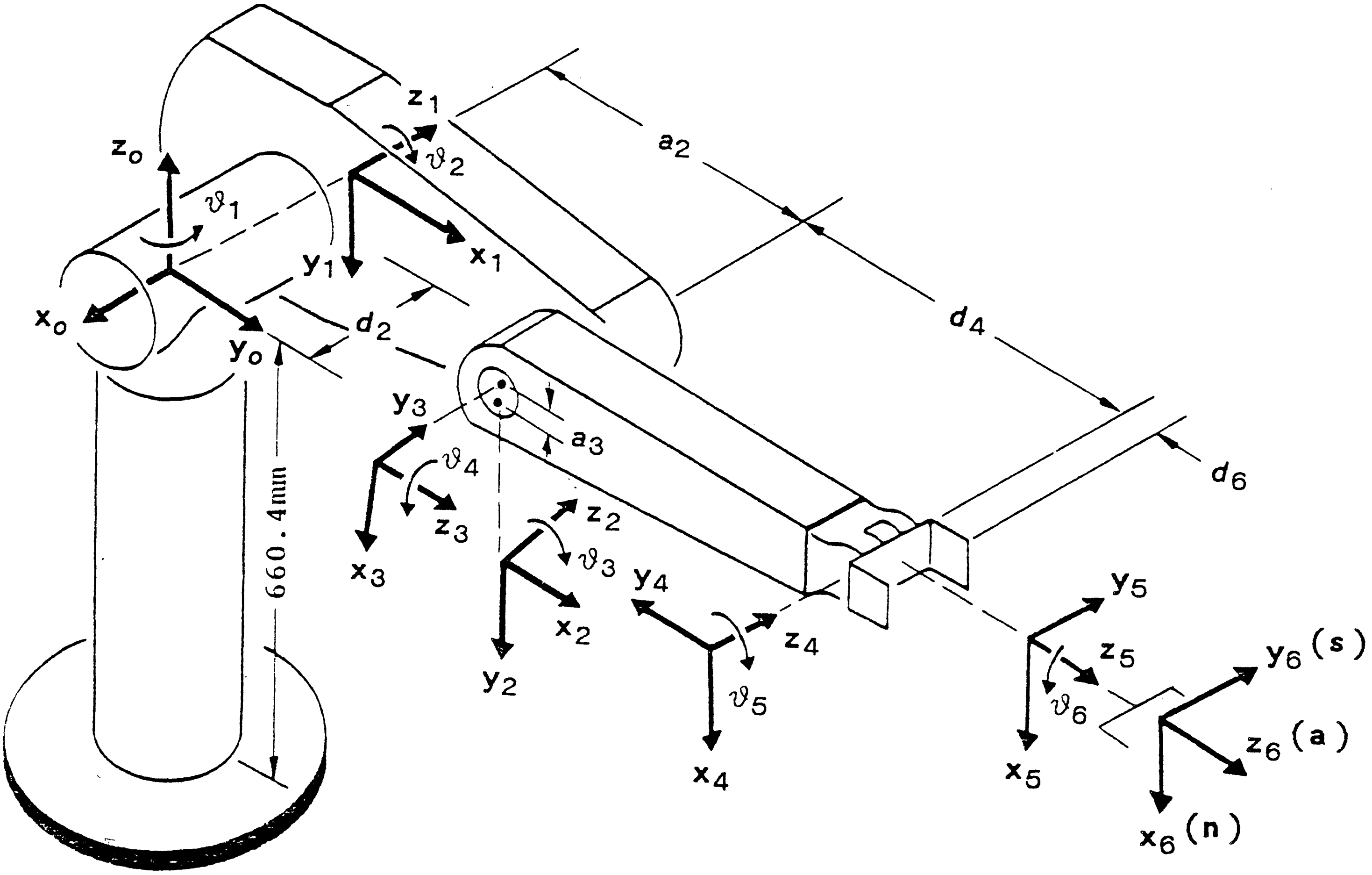


Figure 1 Link Coordinate System and Its Parameters

With the basic rules for establishing an orthonormal coordinate system for each link and the geometric interpretation of the joint and link parameters, a procedure for establishing *consistent* orthonormal coordinate systems for a robot is outlined in [5]. An example of applying this algorithm to a six-joint PUMA robot arm is given in Figure 2. The six $\mathbf{A}_{i-1}^{i}$ homogeneous transformation matrices for the PUMA robot shown in Figure 2 are listed in Figure 3.



PUMA Robot Link Coordinate Parameters

Joint i	ϑ_i	α_i	a_i	d_i	Range
1	90	-90	0	0	-160 to +160
2	0	0	431.8 mm	149.09 mm	-225 to +45
3	90	90	- 20.32 mm	0	-45 to +225
4	0	-90	0	433.07 mm	-110 to +170
5	0	90	0	0	-100 to +100
6	0	0	0	56.25 mm	-266 to +266

Figure 2 Link Coordinate Systems For A PUMA Robot

$$
\mathbf{A}_0^1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
\mathbf{A}_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
\mathbf{A}_2^3 = \begin{bmatrix} C_3 & 0 & S_3 & a_3C_3 \\ S_3 & 0 & -C_3 & a_3S_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\mathbf{A}_3^4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
\mathbf{A}_4^5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\quad
\mathbf{A}_5^6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

where $C_i \equiv \cos\theta_i$; $S_i \equiv \sin\theta_i$

Figure 3 Coordinate Transformation Matrices For The PUMA in Figure 2

3. KINEMATIC EQUATIONS FOR MANIPULATORS

The homogeneous transformation matrix $\mathbf{T}_0^i$ which specifies the position and orientation of the i^{th} coordinate frame with respect to the base coordinate system is the chain product of successive homogeneous transformation matrices of $\mathbf{A}_{i-1}^i$, expressed as:

$$
\mathbf{T}_0^i = \mathbf{A}_0^1\,\mathbf{A}_1^2 \cdots \mathbf{A}_{i-1}^i = \prod_{j=1}^{i} \mathbf{A}_{j-1}^j = \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \text{for } i = 1,2, \cdots, n \tag{3}
$$

Specifically for $i=n$, we obtain the $\mathbf{T}$ matrix, $\mathbf{T} = \mathbf{T}_0^n$, which specifies the position and orientation of the end-point of a manipulator with respect to the base coordinate system. This $\mathbf{T}$ matrix is used so frequently in the kinematic analysis of robot arm that it is called the "arm matrix". Consider the $\mathbf{T}$ matrix to be of the form:

$$
\mathbf{T} = \begin{bmatrix} \mathbf{x}_n & \mathbf{y}_n & \mathbf{z}_n & \mathbf{p}_n \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}
$$

where (see Figure 4):

n is the normal vector of the hand. Assuming parallel-jaw hand, it is orthogonal to the fingers of the robot arm.

s is the sliding vector of the hand. It is pointing in the direction of the finger motion as the gripper opens and closes.

a is the approach vector of the hand. It is pointing in the direction normal to the palm of the hand. (i.e., normal to the tool mounting plate of the arm.)

p is the position vector of the hand. It points from the origin of the base coordinate system to the origin of the hand coordinate system, which is usually located at the center point of the fully closed fingers.

The elements of the arm matrix for the PUMA robot arm shown in Figure 2 are found to be

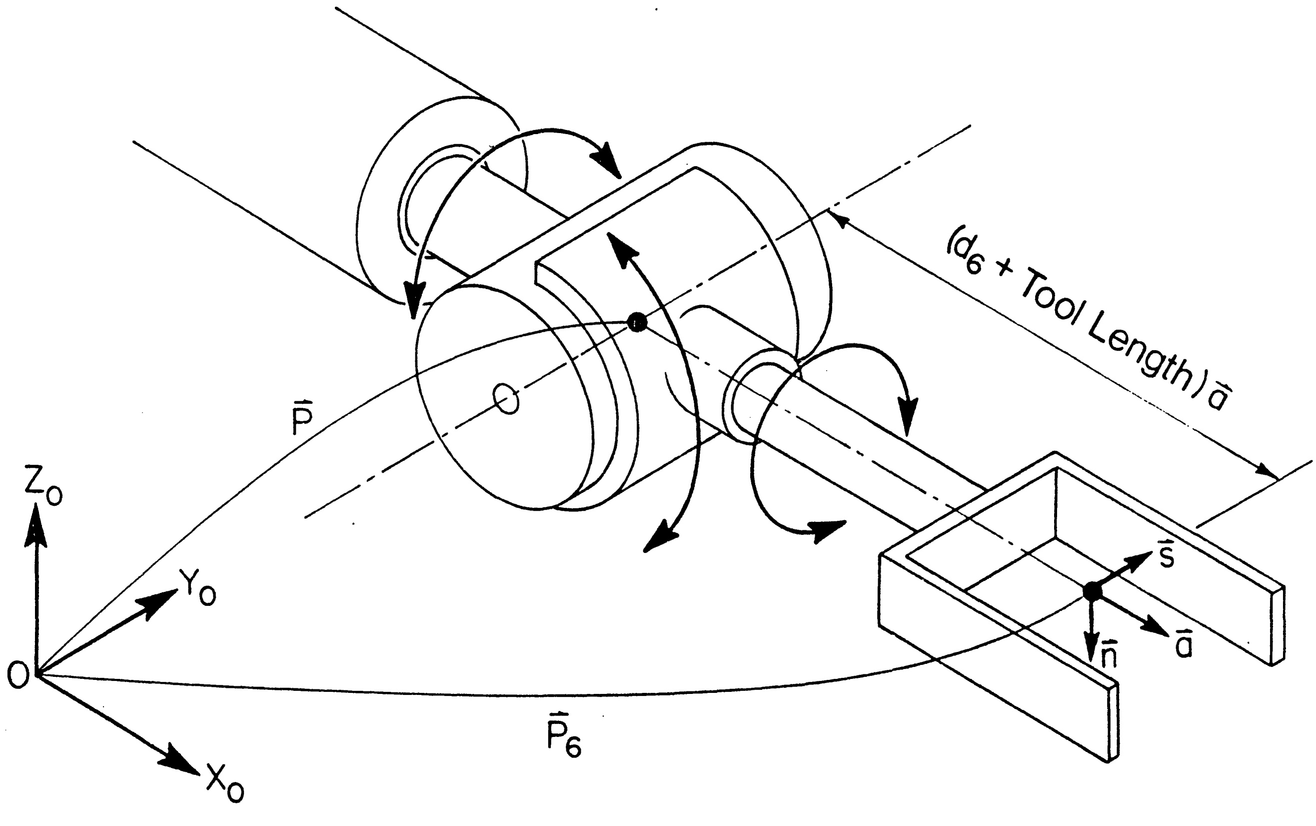


Figure 4 Hand Coordinate System and [n , s , a]

$$\begin{aligned}
n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1[S_4C_5C_6 + C_4S_6]\\
n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1[S_4C_5C_6 + C_4S_6]\\
n_z &= -S_{23}[C_4C_5C_6 - S_4S_6] - C_{23}S_5C_6
\end{aligned} \tag{5}$$

$$\begin{aligned}
s_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - S_1[-S_4C_5S_6 + C_4C_6]\\
s_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_1[-S_4C_5S_6 + C_4C_6]\\
s_z &= S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6
\end{aligned} \tag{6}$$

$$\begin{aligned}
a_x &= C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5\\
a_y &= S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5\\
a_z &= -S_{23}C_4S_5 + C_{23}C_5
\end{aligned} \tag{7}$$

$$\begin{aligned}
p_x &= C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2)\\
p_y &= S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2)\\
p_z &= d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2
\end{aligned} \tag{8}$$

where $C_i \equiv \cos\theta_i$; $S_i \equiv \sin\theta_i$; $C_{ij} \equiv \cos(\theta_i + \theta_j)$; $S_{ij} \equiv \sin(\theta_i + \theta_j)$.

4. THE INVERSE KINEMATICS SOLUTION OF A PUMA ROBOT ARM

This section presents a geometric approach to derive a consistent joint angle solution of a PUMA robot given the arm matrix as in Eq. 4. Based on the link coordinate systems and human arm geometry, various arm configurations of a PUMA robot can be identified with the assistance of three configuration indicators (ARM, ELBOW and WRIST) - two associated with the solution of the first three joints and the other with the last three joints. For a six-joint PUMA robot arm, there are four possible solutions to the first three joints and for each of these four solutions there are two possible solutions to the last three joints. The first two configuration indicators allow one to determine one solution from the possible four solutions for the first three joints. Similarly, the third indicator selects a solution from the possible two solutions for the last three joints. The arm configuration indicators are prespecified by a user for finding the inverse solution. The solution is calculated in two stages. First a position vector pointing from the shoulder to the wrist is derived. This is used to derive the solution of each joint i ($i = 1,2,3$) of the first three joints by looking at the projection of the position vector onto the $\mathbf{x}_{i-1}$-$\mathbf{y}_{i-1}$ plane. The last three joints are solved using the calculated joint solution from the first three joints, the orientation submatrices of $\mathbf{T}_0^i$ and $\mathbf{A}_{i-1}^i$ ($i = 4,5,6$), and the projection of the link coordinate frames onto the $\mathbf{x}_{i-1}$-$\mathbf{y}_{i-1}$ plane. From the geometry, one can easily find the arm solution consistently. As a verification of the joint solution, the arm configuration indicators can be determined from the corresponding decision equations which are functions of the joint angles.

4.1. DEFINITION OF VARIOUS ARM CONFIGURATIONS

For the PUMA robot arm shown in Figure 2 (and other rotary robot arms), various arm configurations are defined according to human arm geometry and the link coordinate systems which are established using the algorithm in [5] as: (Figure 5)

RIGHT (shoulder) ARM: Positive θ_2 moves the wrist in the ***positive*** $\mathbf{z}_0$ direction while joint 3 is not activated.

LEFT (shoulder) ARM: Positive θ_2 moves the wrist in the ***negative*** $\mathbf{z}_0$ direction while joint 3 is not activated.

ABOVE ARM (elbow above wrist): Position of the wrist of the $\left\{\begin{matrix} \mathit{RIGHT} \\ \mathit{LEFT} \end{matrix}\right\}$ arm with respect to the shoulder coordinate system has $\left\{\begin{matrix} \mathit{negative} \\ \mathit{positive} \end{matrix}\right\}$ coordinate value along the $\mathbf{y}_2$-axis.

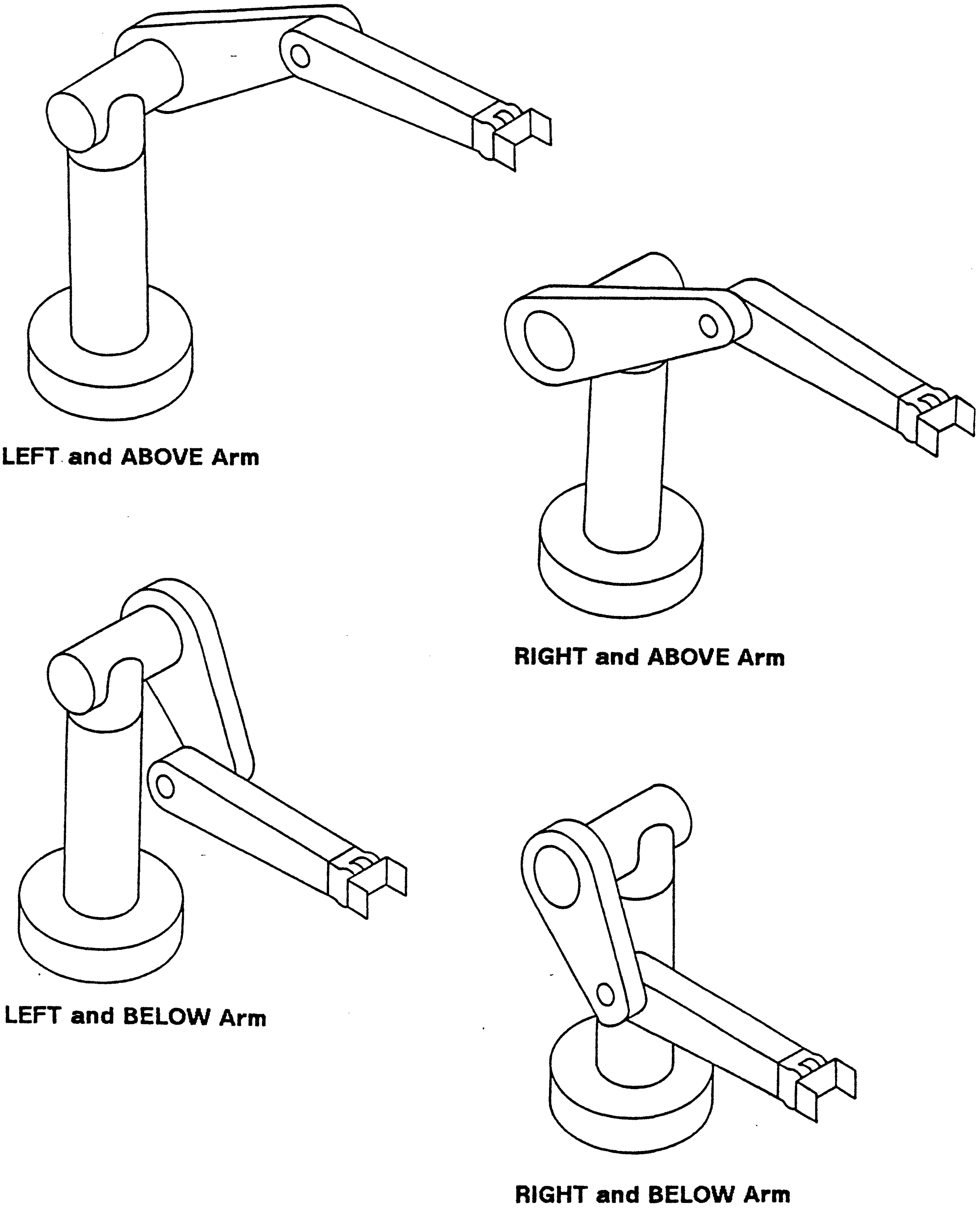
BELOW ARM (elbow below wrist): Position of the wrist of the $\left\{\begin{matrix} \mathit{RIGHT} \\ \mathit{LEFT} \end{matrix}\right\}$ arm with respect to the shoulder coordinate system has $\left\{\begin{matrix} \mathit{positive} \\ \mathit{negative} \end{matrix}\right\}$ coordinate value along the $\mathbf{y}_2$-axis.

WRIST DOWN: The $\mathbf{s}$ unit vector of the hand coordinate system and the $\mathbf{y}_5$ unit vector of the $(\mathbf{x}_5, \mathbf{y}_5, \mathbf{z}_5)$ coordinate system have a positive dot product.

WRIST UP: The $\mathbf{s}$ unit vector of the hand coordinate system and the $\mathbf{y}_5$ unit vector of the $(\mathbf{x}_5, \mathbf{y}_5, \mathbf{z}_5)$ coordinate system have a negative dot product.

(Note that the definition of the arm configurations with respect to the link coordinate systems may have to be slightly modified if one uses different link coordinate systems.)

With respect to the above definition of various arm configurations, two arm configuration ***indicators*** (ARM and ELBOW) are defined for each arm configuration. These two indicators are combined to give one solution out of



LEFT and ABOVE Arm

RIGHT and ABOVE Arm

LEFT and BELOW Arm

RIGHT and BELOW Arm

Figure 5. Definition of Various Arm Configurations

the possible four joint solutions for the first three joints. For each of the four arm configurations (Figure 5) defined by these two indicators, the third indicator (WRIST) gives one of the two possible joint solutions for the last three joints. These three indicators can be defined as:

$$ARM = \begin{cases} +1 & ; \; RIGHT\ arm \\ -1 & ; \; LEFT\ arm \end{cases} \tag{9}$$

$$ELBOW = \begin{cases} +1 & ; \; ABOVE\ arm \\ -1 & ; \; BELOW\ arm \end{cases} \tag{10}$$

$$WRIST = \begin{cases} +1 & ; \; WRIST\ DOWN \\ -1 & ; \; WRIST\ UP \end{cases} \tag{11}$$

In addition to these indicators, the user can define a "FLIP" toggle as:

$$FLIP = \begin{cases} +1 & ; \; Flip\ the\ wrist\ orientation \\ -1 & ; \; Do\ not\ flip\ the\ wrist\ orientation \end{cases} \tag{12}$$

The signed values of these indicators and the toggle are prespecified by a user for finding the inverse kinematics solution. These indicators can also be set from the knowledge of the joint angles of the robot arm using the corresponding decision equations. We shall later give the decision equations that determine these indicator values. The decision equations can be used as a verification of the inverse kinematics solution.

4.2. ARM SOLUTION FOR THE FIRST THREE JOINTS OF A PUMA ROBOT ARM

From the kinematics diagram of the PUMA robot arm as in Figure 2, we define a position vector $\mathbf{p}$ which points from the origin of the shoulder coordinate system $(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$ to the point where the last three joint axes intersect as (see Figure 4):

$$\mathbf{p} = \mathbf{p}_6 - d_6\mathbf{a} = (p_x, p_y, p_z)^T \tag{13}$$

which corresponds to the position vector of $\mathbf{T}_0^4$:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} C_1\left(a_2C_2 + a_3C_{23} + d_4S_{23}\right) - d_2S_1 \\ S_1\left(a_2C_2 + a_3C_{23} + d_4S_{23}\right) + d_2C_1 \\ d_4C_{23} - a_3S_{23} - a_2S_2 \end{bmatrix} \tag{14}$$

Joint One Solution. If we project the position vector **p** onto the $\mathbf{x}_0$-$\mathbf{y}_0$ plane as in Figure 6, we obtain the following equations for solving θ_1:

$$\theta_1^L = \phi - \alpha \ ; \ \theta_1^R = \pi + \phi + \alpha \tag{15}$$

$$r = \sqrt{p_x^2+p_y^2-d_2^2} \ ; \ R = \sqrt{p_x^2+p_y^2} \tag{16}$$

$$\sin\phi = \frac{p_y}{R} \ ; \ \cos\phi = \frac{p_x}{R} \tag{17}$$

$$\sin\alpha = \frac{d_2}{R} \ ; \ \cos\alpha = \frac{r}{R} \tag{18}$$

where the superscript L/R on joint angles indicates the LEFT/RIGHT arm configurations. From Eqs. 15-18, we obtain the sine and cosine functions of θ_1 for LEFT/RIGHT arm configurations:

$$\sin\theta_1^L = \sin(\phi - \alpha) = \sin\phi\cos\alpha - \cos\phi\sin\alpha = \frac{p_yr - p_xd_2}{R^2} \tag{19}$$

$$\cos\theta_1^L = \cos(\phi - \alpha) = \cos\phi\cos\alpha + \sin\phi\sin\alpha = \frac{p_xr + p_yd_2}{R^2} \tag{20}$$

$$\sin\theta_1^R = \sin(\pi + \phi + \alpha) = \frac{-p_yr - p_xd_2}{R^2} \tag{21}$$

$$\cos\theta_1^R = \cos(\pi + \phi + \alpha) = \frac{-p_xr + p_yd_2}{R^2} \tag{22}$$

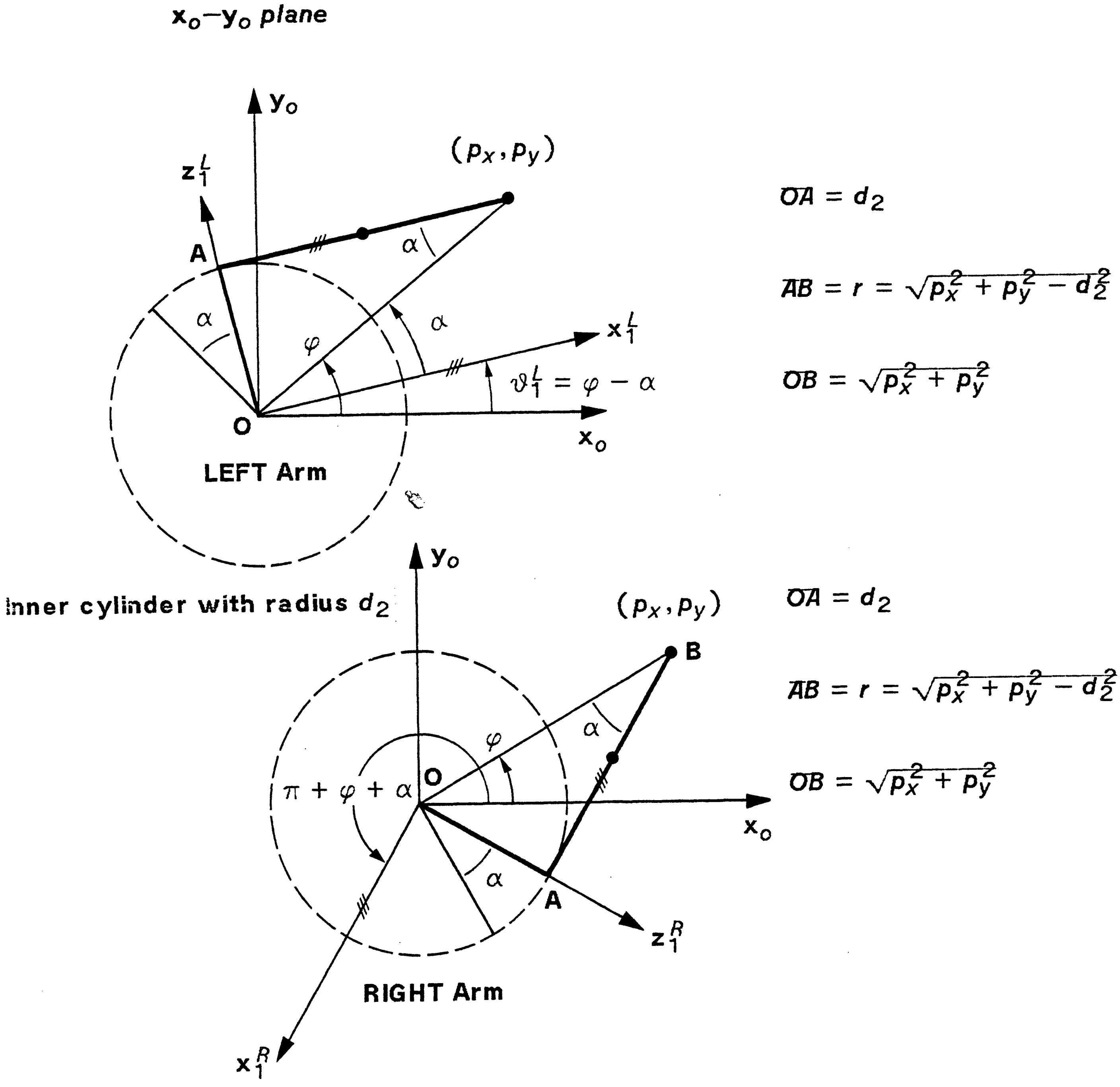


Figure 6. Joint One Solution

Combining Eqs. 19-22 and using the ARM indicator to indicate the LEFT/RIGHT arm configuration, we obtain the sine and cosine functions of θ_1 respectively:

$$\sin\theta_1 = \frac{-ARM \cdot p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_x d_2}{p_x^2 + p_y^2} \tag{23}$$

$$\cos\theta_1 = \frac{-ARM \cdot p_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_y d_2}{p_x^2 + p_y^2} \tag{24}$$

where positive square root is taken in these equations and ARM is defined as in Eq. 9. In order to evaluate θ_1 for $-\pi \leq \theta_1 \leq \pi$, an arc tangent function, $atan2(\frac{y}{x})$, which returns $\tan^{-1}(\frac{y}{x})$ adjusted to the proper quadrant will be used. It is defined as:

$$\theta = atan2(\frac{y}{x}) = \begin{cases} 0^o \leq \theta \leq 90^o & ; \text{ for } +x \text{ and } +y \\ 90^o \leq \theta \leq 180^o & ; \text{ for } -x \text{ and } +y \\ -180^o \leq \theta \leq -90^o & ; \text{ for } -x \text{ and } -y \\ -90^o \leq \theta \leq 0^o & ; \text{ for } +x \text{ and } -y \end{cases} \tag{25}$$

From Eqs. 23 and 24, and using Eq. 25, θ_1 is found to be:

$$\theta_1 = atan2\left[\frac{\sin\theta_1}{\cos\theta_1}\right] = atan2\left[\frac{-ARM \cdot p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_x d_2}{-ARM \cdot p_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_y d_2}\right] \quad ; \; -\pi \leq \theta_1 \leq \pi \tag{26}$$

Joint Two Solution. To find joint 2, we project the position vector **p** onto the $\mathbf{x}_1$-$\mathbf{y}_1$ plane as shown in Figure 7. From Figure 7, we have four different arm configurations. Each arm configuration corresponds to different values of joint two as:

Arm Configurations	θ_2	ARM	ELBOW	ARM · ELBOW
LEFT and ABOVE arm	$\alpha - \beta$	-1	+1	-1
LEFT and BELOW arm	$\alpha + \beta$	-1	-1	+1
RIGHT and ABOVE arm	$\alpha + \beta$	+1	+1	+1
RIGHT and BELOW arm	$\alpha - \beta$	+1	-1	-1

where $0^\circ \leq \alpha \leq 360^\circ$ and $0^\circ \leq \beta \leq 90^\circ$.

Table 1. Various Arm Configurations for Joint Two

From the above table, θ_2 can be expressed in one equation for different arm and elbow configurations using the ARM and ELBOW indicators as:

$$\theta_2 = \alpha + (ARM \cdot ELBOW)\,\beta = \alpha + K \cdot \beta \tag{27}$$

where the combined arm configuration indicator $K = ARM \cdot ELBOW$ will give an appropriate signed value and the "dot" represents a multiplication operation on the indicators. From the arm geometry in Figure 7, we obtain:

$$R = \sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2} \quad ; \quad r = \sqrt{p_x^2 + p_y^2 - d_2^2} \tag{28}$$

$$\sin\alpha = -\frac{p_z}{R} = -\frac{p_z}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{29}$$

$$\cos\alpha = -\frac{ARM \cdot r}{R} = -\frac{ARM \cdot \sqrt{p_x^2 + p_y^2 - d_2^2}}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{30}$$

$$\cos\beta = \frac{a_2^2 + R^2 - (\, d_4^2 + a_3^2 \,)}{2a_2R} = \frac{p_x^2 + p_y^2 + p_z^2 + a_2^2 - d_2^2 - (\, d_4^2 + a_3^2 \,)}{2a_2\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{31}$$

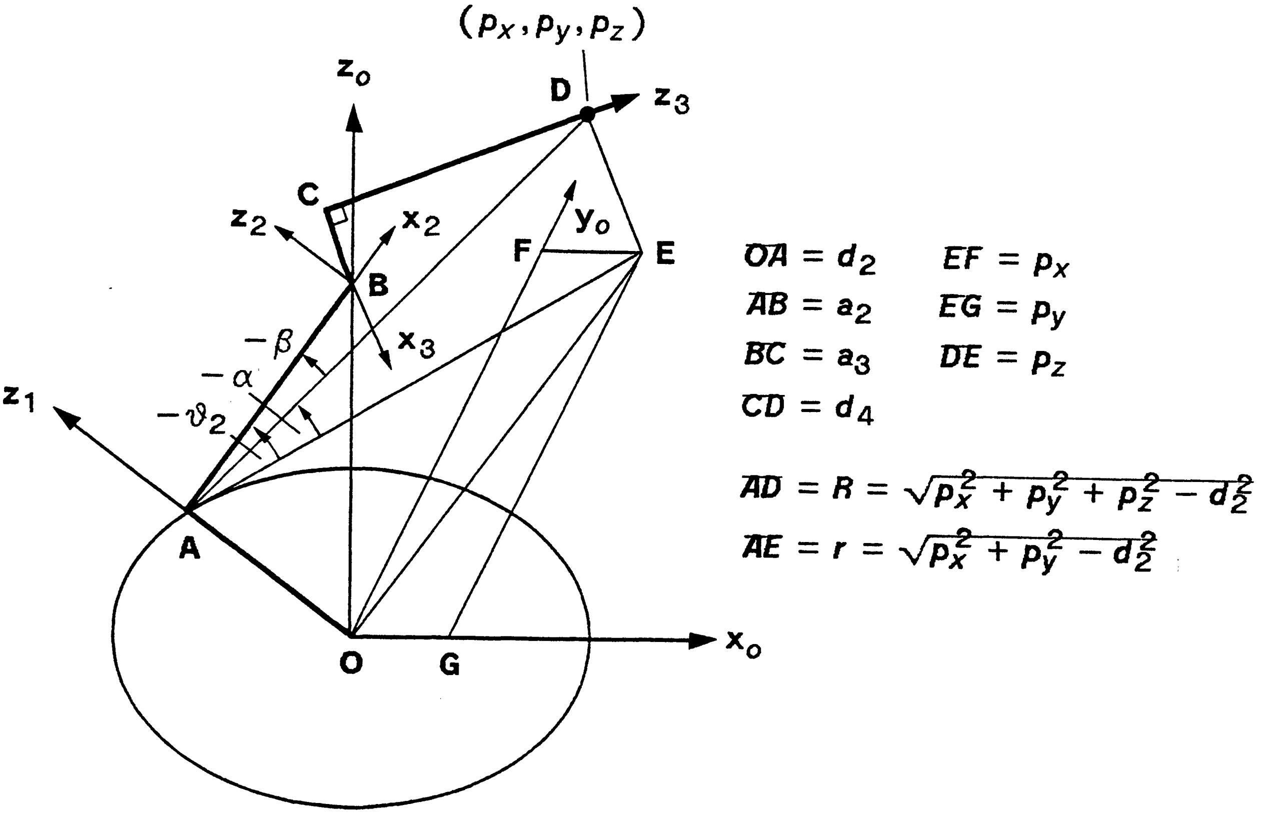


Figure 7. Joint Two Solution

$$\sin\beta = \sqrt{1 - \cos^2\beta} \tag{32}$$

From Eqs. 27-32, we can find the sine and cosine functions of θ_2:

$$\sin\theta_2 = \sin(\alpha + K\cdot\beta) = \sin\alpha\cos\beta + (ARM \cdot ELBOW)\cos\alpha\sin\beta \tag{33}$$

$$\cos\theta_2 = \cos(\alpha + K\cdot\beta) = \cos\alpha\cos\beta - (ARM \cdot ELBOW)\sin\alpha\sin\beta \tag{34}$$

From Eqs. 33 and 34, we obtain the solution for θ_2:

$$\theta_2 = atan2\left[\frac{\sin\theta_2}{\cos\theta_2}\right] \quad ; \quad -\pi \leq \theta_2 \leq \pi \tag{35}$$

Joint Three Solution. For joint 3, we project the position vector **p** onto the x_2-y_2 plane as shown in Figure 8. From Figure 8, we again have four different arm configurations. Each arm configuration corresponds to different values of joint three as:

Arm Configurations	$(p_2^4)_y$	θ_3	ARM	ELBOW	ARM · ELBOW
LEFT and ABOVE arm	≥ 0	$\phi - \beta$	-1	+1	-1
LEFT and BELOW arm	≤ 0	$\phi - \beta$	-1	-1	+1
RIGHT and ABOVE arm	≤ 0	$\phi - \beta$	+1	+1	+1
RIGHT and BELOW arm	≥ 0	$\phi - \beta$	+1	-1	-1

Table 2. Various Arm Configurations for Joint Three

where $(p_2^4)_y$ is the y-component of the position vector from the origin of (x_2, y_2, z_2) to the point where the last three joint axes intersect.

From the arm geometry in Figure 8, we obtain the following equations for finding the solution for θ_3:

$$R = \sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2} \tag{36}$$

$$\cos\phi = \frac{a_2^2 + (d_4^2 + a_3^2) - R^2}{2a_2\sqrt{d_4^2 + a_3^2}} \quad ; \quad \sin\phi = ARM \cdot ELBOW\sqrt{1 - \cos^2\phi} \tag{37}$$

$$\sin\beta = \frac{d_4}{\sqrt{d_4^2 + a_3^2}} \quad ; \quad \cos\beta = \frac{|a_3|}{\sqrt{d_4^2 + a_3^2}} \tag{38}$$

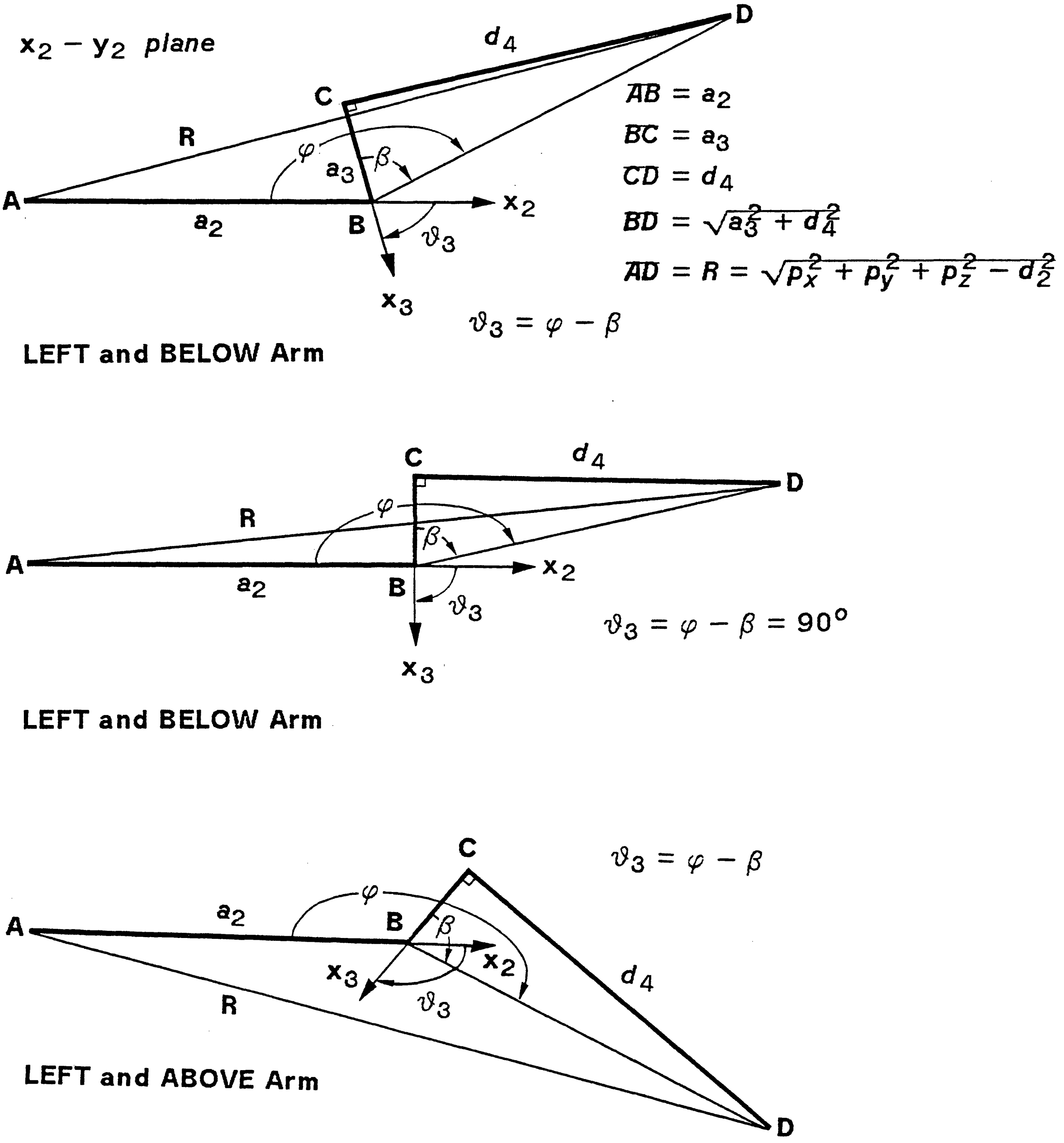


Figure 8. Joint Three Solution

From Table 2, we obtain the equation for θ_3:

$$\theta_3 = \phi - \beta \tag{39}$$

From Eq. 39, the sine and cosine functions of θ_3 are, respectively:

$$\sin\theta_3 = \sin(\phi - \beta) = \sin\phi\cos\beta - \cos\phi\sin\beta \tag{40}$$

$$\cos\theta_3 = \cos(\phi - \beta) = \cos\phi\cos\beta + \sin\phi\sin\beta \tag{41}$$

From Eqs. 40 and 41, and using Eqs. 36-38, we find the solution for θ_3:

$$\theta_3 = atan2\left[\frac{\sin\theta_3}{\cos\theta_3}\right] \quad ; \; -\pi \leq \theta_3 \leq \pi \tag{42}$$

4.3. ARM SOLUTION FOR THE LAST THREE JOINTS OF A PUMA ROBOT ARM

Knowing the first three joint angles, we can evaluate the $\mathbf{T}_0^3$ matrix which is used extensively to find the solution of the last three joints. The solution of the last three joints of a PUMA robot arm can be found by setting these joints to meet the following criteria:

(1) Set joint 4 such that a rotation about joint 5 will align the axis of motion of joint 6 with the given approach vector ($\mathbf{a}$ *of* $\mathbf{T}$)

(2) Set joint 5 to align the axis of motion of joint 6 with the approach vector.

(3) Set joint 6 to align the given orientation vector (or sliding vector or $\mathbf{y}_6$) and normal vector.

Mathematically the above criteria respectively mean:

$$\mathbf{z}_4 = \frac{\pm(\mathbf{z}_3 \times \mathbf{a})}{\| \mathbf{z}_3 \times \mathbf{a} \|} \quad ; \; given\ \mathbf{a} = (a_x, a_y, a_z)^T \tag{43}$$

$$\mathbf{a} = \mathbf{z}_5 \quad ; \; given\ \mathbf{a} = (a_x, a_y, a_z)^T \tag{44}$$

$$\mathbf{s} = \mathbf{y}_6 \quad ; \quad given\ \mathbf{s} = (s_x, s_y, s_z)^T \text{ and } \mathbf{n} = (n_x, n_y, n_z)^T \tag{45}$$

In Eq. 43, the vector cross product may be taken to be positive or negative. As a result, there are two possible solutions for θ_4. If the vector cross product is zero (i.e. $\mathbf{z}_3$ is parallel to $\mathbf{a}$), it indicates the degenerate case. This happens when the axes of rotation for joint 4 and joint 6 are parallel. It indicates that at this particular arm configuration, a five-axis robot arm rather than a six-axis one would suffice.

Joint Four Solution. Both orientations of the wrist (UP and DOWN) are defined by looking at the orientation of the hand coordinate frame ($\mathbf{n}$,$\mathbf{s}$,$\mathbf{a}$) with respect to the ($\mathbf{x}_5$,$\mathbf{y}_5$,$\mathbf{z}_5$) coordinate frame. The sign of the vector cross product in Eq. 43 cannot be determined without referring to the orientation of either the $\mathbf{n}$ or $\mathbf{s}$ unit vector with respect to the $\mathbf{x}_5$ or $\mathbf{y}_5$ unit vector, respectively, which have a fixed relation with respect to the $\mathbf{z}_4$ unit vector from the assignment of the link coordinate frames. (From Figure 2, we have the $\mathbf{z}_4$ unit vector pointing at the same direction as the $\mathbf{y}_5$ unit vector)

We shall start with an assumption that the vector cross product in Eq. 43 has a positive sign. This can be indicated by an orientation indicator Ω which is defined as:

$$\Omega = \begin{cases} 0 & ; \text{ if } in\ the\ degenerate\ case \\ \mathbf{s}\cdot\mathbf{y}_5 & ; \text{ if } \mathbf{s}\cdot\mathbf{y}_5 \neq 0 \\ \mathbf{n}\cdot\mathbf{y}_5 & ; \text{ if } \mathbf{s}\cdot\mathbf{y}_5 = 0 \end{cases} \tag{46}$$

From Figure 2, $\mathbf{y}_5 = \mathbf{z}_4$ and using Eq. 43, the orientation indicator Ω can be rewritten as:

$$\Omega = \begin{cases} 0 & ; \text{ if } in\ the\ degenerate\ case \\ \mathbf{s}\cdot\dfrac{(\mathbf{z}_3 \times \mathbf{a})}{\|\mathbf{z}_3 \times \mathbf{a}\|} & ; \text{ if } \mathbf{s}\cdot(\mathbf{z}_3 \times \mathbf{a}) \neq 0 \\ \mathbf{n}\cdot\dfrac{(\mathbf{z}_3 \times \mathbf{a})}{\|\mathbf{z}_3 \times \mathbf{a}\|} & ; \text{ if } \mathbf{s}\cdot(\mathbf{z}_3 \times \mathbf{a}) = 0 \end{cases} \tag{47}$$

If our assumption of the sign of the vector cross product in Eq. 43 is not correct, it will be corrected later using the combination of the WRIST indicator and the orientation indicator Ω. The Ω is used to indicate the initial orientation of the $\mathbf{z}_4$ unit vector (positive direction) from the link coordinate systems assignment, while the WRIST indicator specifies the user's preference of the orientation of the wrist subsystem according to the definition given in Eq. 11. If both the orientation Ω and the WRIST indicators have the same sign, then the assumption of the sign of the vector cross product in Eq. 43 is correct. Various wrist orientations resulting from the combination of the various values of the WRIST and orientation indicators are tabulated in Table 3.

Wrist Orientation	$\Omega = \mathbf{s} \cdot \mathbf{y}_5$ or $\mathbf{n} \cdot \mathbf{y}_5$	WRIST	$M = WRIST \cdot sign(\Omega)$
DOWN	≥ 0	+1	+1
DOWN	< 0	+1	-1
UP	≥ 0	-1	-1
UP	< 0	-1	+1

Table 3. Various Orientations for The Wrist

Again looking at the projection of the coordinate frame $(\mathbf{x}_4, \mathbf{y}_4, \mathbf{z}_4)$ on the $\mathbf{x}_3$-$\mathbf{y}_3$ plane and from the Table 3 and Figure 9, it can be shown that the followings are true (see Figure 9):

$$\sin\theta_4 = -M \cdot (\mathbf{z}_4 \cdot \mathbf{x}_3) \quad ; \quad \cos\theta_4 = M \cdot (\mathbf{z}_4 \cdot \mathbf{y}_3) \tag{48}$$

where $\mathbf{x}_3$ and $\mathbf{y}_3$ are the x and y column vector of $\mathbf{T}_0^3$ respectively, $M = WRIST \cdot sign(\Omega)$, and the sign function is defined as:

$$sign(x) = \begin{cases} +1 & ; \text{if } x \geq 0 \\ -1 & ; \text{if } x < 0 \end{cases} \tag{49}$$

Thus the solution for θ_4 with the orientation and WRIST indicators is:

$$\theta_4 = atan2\left[\frac{\sin\theta_4}{\cos\theta_4}\right] = atan2\left[\frac{M \cdot (C_1 a_y - S_1 a_x)}{M \cdot (C_1 C_{23} a_x + S_1 C_{23} a_y - S_{23} a_z)}\right] ; -\pi \leq \theta_4 \leq \pi \tag{50}$$

If the degenerate case occurs, any convenient value may be chosen for θ_4 as long as the orientation of the wrist (UP/DOWN) is satisfied. This can always be ensured by setting θ_4 equals to the current value of θ_4. In addition to this,

$$\sin \vartheta_4 = -(z_4 \cdot x_3)$$

$$\cos \vartheta_4 = (z_4 \cdot y_3)$$

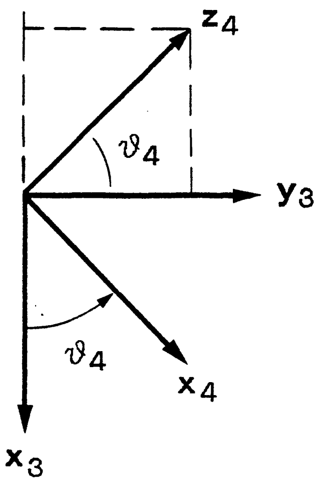


Figure 9 Joint Four Solution

the user can turn on the FLIP toggle to obtain the other solution of θ_4, that is $\theta_4 = \theta_4 + 180^\circ$.

Joint Five Solution. To find θ_5, we use the criterion that aligns the axis of rotation of joint six with the approach vector (or $\mathbf{a} = \mathbf{z}_5$). Looking at the projection of the coordinate frame $(\mathbf{x}_5, \mathbf{y}_5, \mathbf{z}_5)$ on the $\mathbf{x}_4$-$\mathbf{y}_4$ plane, it can be shown that the followings are true (see Figure 10):

$$\sin\theta_5 = \mathbf{a} \cdot \mathbf{x}_4 \quad ; \quad \cos\theta_5 = -(\mathbf{a} \cdot \mathbf{y}_4) \tag{51}$$

where $\mathbf{x}_4$ and $\mathbf{y}_4$ are the x and y column vector of $\mathbf{T}_0^4$ respectively and $\mathbf{a}$ is the approach vector. Thus the solution for θ_5 is:

$$\sin\vartheta_5 = \mathbf{a}\cdot\mathbf{x}_4$$

$$\cos\vartheta_5 = -(\mathbf{a}\cdot\mathbf{y}_4)$$

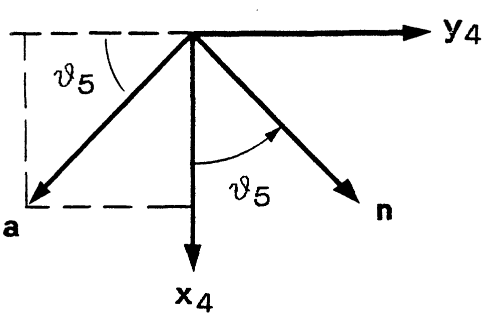


Figure 10 Joint Five Solution

$$\begin{aligned}\theta_5 &= atan2\left[\frac{\sin\theta_5}{\cos\theta_5}\right] \quad ; \quad -\pi \leq \theta_5 \leq \pi \\ &= atan2\left[\frac{(C_1C_{23}C_4-S_1S_4)a_x+(S_1C_{23}C_4+C_1S_4)a_y-C_4S_{23}a_z}{C_1S_{23}a_x+S_1S_{23}a_y+C_{23}a_z}\right]\end{aligned} \tag{52}$$

If $\theta_5 \approx 0$, then the degenerate case occurs.

Joint Six Solution. Up to now, we have aligned the axis of joint 6 with the approach vector. Next we need to align the orientation of the gripper to ease picking up the object. The criterion for doing this is to set $\mathbf{s} = \mathbf{y}_6$. Looking at the projection of the hand coordinate frame $(\mathbf{n},\mathbf{s},\mathbf{a})$ on the $\mathbf{x}_5$-$\mathbf{y}_5$ plane, it can be shown that the followings are true (see Figure 11):

$$\sin\theta_6 = \mathbf{n}\cdot\mathbf{y}_5 \quad ; \quad \cos\theta_6 = \mathbf{s}\cdot\mathbf{y}_5 \tag{53}$$

where $\mathbf{y}_5$ is the y column vector of $\mathbf{T}_0^5$ and $\mathbf{n}$ and $\mathbf{s}$ are the normal and sliding vectors of $\mathbf{T}_0^6$ respectively. Thus the solution for θ_6 is:

$$\begin{aligned}\theta_6 &= atan2\left[\frac{\sin\theta_6}{\cos\theta_6}\right] \quad ; \quad -\pi \le \theta_6 \le \pi \\ &= atan2\left[\frac{(-S_1C_4-C_1C_{23}S_4)n_x+(C_1C_4-S_1C_{23}S_4)n_y+(S_4S_{23})n_z}{(-S_1C_4-C_1C_{23}S_4)s_x+(C_1C_4-S_1C_{23}S_4)s_y+(S_4S_{23})s_z}\right]\end{aligned} \tag{54}$$

If the degenerate case occurs, then $(\theta_4 + \theta_6)$ equals to the total angle required to align the sliding vector ($\mathbf{s}$) and the normal vector ($\mathbf{n}$). If the FLIP toggle is on

$$\sin\vartheta_6 = \mathbf{n}\cdot\mathbf{y}_5$$

$$\cos\vartheta_6 = \mathbf{s}\cdot\mathbf{y}_5$$

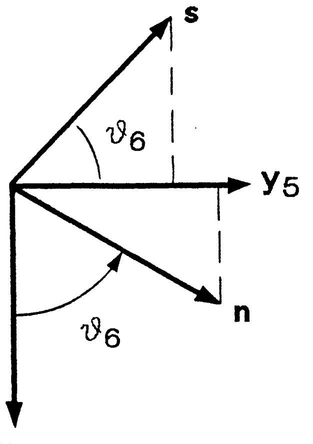


Figure 11 Joint Six Solution

(i.e. FLIP=1), then $\theta_4 = \theta_4 + \pi, \theta_5 = -\theta_5$, and $\theta_6 = \theta_6 + \pi$.

In summary, there are eight solutions to the inverse kinematics problem of a six-joint PUMA robot arm. There are four solutions for the first three joint solutions - two for the right shoulder arm configuration and two for the left shoulder arm configuration. For each arm configuration, Eqs. 26, 35, 42, 49, 52, and 54 give one set of solution $(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$ and $(\theta_1,\theta_2,\theta_3,\theta_4+\pi,-\theta_5,\theta_6+\pi)$ (with the FLIP toggle on) give another set of solution.

5. DECISION EQUATIONS FOR THE ARM CONFIGURATION INDICATORS

In the previous section, the arm solution of a PUMA robot arm has been derived. The solution is not *unique* and depends on the arm configuration indicators specified by the user. These arm configuration indicators (ARM, ELBOW and WRIST) can also be determined from the joint angles. In this section, we shall derive the respective decision equation for each arm configuration indicator. The signed value of the decision equation (positive, zero, or negative) provide an indication of the arm configuration as defined in Eqs. 9-11.

For the ARM indicator, following the definition of the RIGHT/LEFT arm, a decision equation for the ARM indicator can be found to be:

$$g(\theta,\mathbf{p}) = \mathbf{z}_0 \cdot \frac{\mathbf{z}_1 \times \mathbf{p}'}{\|\mathbf{z}_1 \times \mathbf{p}'\|} = \mathbf{z}_0 \cdot \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ p_x & p_y & 0 \end{vmatrix} \cdot \frac{1}{\|\mathbf{z}_1 \times \mathbf{p}'\|} \tag{55}$$

$$= \frac{-p_y\sin\theta_1 - p_x\cos\theta_1}{\|\mathbf{z}_1 \times \mathbf{p}'\|}$$

where $\mathbf{p}' = (p_x,p_y,0)^T$ is the projection of the position vector $\mathbf{p}$ (Eq. 14) onto the x_0-y_0 plane, $\mathbf{z}_1 = (-\sin\theta_1,\cos\theta_1,0)^T$ from the third column vector of $\mathbf{T}_0^1$, and $\mathbf{z}_0 = (0,0,1)^T$.

If $g(\theta,\mathbf{p}) > 0$, then the arm is in the RIGHT arm configuration.

If $g(\theta,\mathbf{p}) < 0$, then the arm is in the LEFT arm configuration.

If $g(\theta,\mathbf{p}) = 0$, then the criterion for finding the LEFT/RIGHT arm configuration cannot be uniquely determined. The arm is within the inner cylinder of radius d_2 in the workspace (see Figure 6). In this case, it is default to the RIGHT arm (ARM = +1).

Since the denominator of the above decision equation is always positive, the determination of the LEFT/RIGHT arm configuration is reduced to checking the sign of the numerator of $g(\theta,\mathbf{p})$:

$$ARM = sign(g(\theta,\mathbf{p})) = sign(-p_x\cos\theta_1 - p_y\sin\theta_1) \tag{56}$$

where the sign function is defined in Eq. 49. Substituting the x and y components of $\mathbf{p}$ from Eq. 14, Eq. 56 becomes:

$$ARM = sign(g(\theta,\mathbf{p})) = sign(g(\theta)) = sign(-d_4S_{23} - a_3C_{23} - a_2C_2) \tag{57}$$

Hence from the decision equation in Eq. 57, one can relate its signed value to the ARM indicator for the RIGHT/LEFT arm configuration as:

$$ARM = sign(-d_4S_{23} - a_3C_{23} - a_2C_2) = \begin{cases} +1 & ==> \ RIGHT\ arm \\ -1 & ==> \ LEFT\ arm \end{cases} \tag{58}$$

For the ELBOW arm indicator, we follow the definition of ABOVE/BELOW arm to formulate the corresponding decision equation. Using $(\mathbf{p}_2^4)_y$ and the ARM indicator in the Table 2, the decision equation for the ELBOW indicator is based on the sign of the y-component of the position vector of $\mathbf{A}_2^3 \cdot \mathbf{A}_3^4$ and the ARM indicator:

$$ELBOW = ARM \cdot sign(d_4C_3 - a_3S_3) = \begin{cases} +1 & ==> \ ELBOW\ above\ wrist \\ -1 & ==> \ ELBOW\ below\ wrist \end{cases} \tag{59}$$

For the WRIST indicator, we follow the definition of DOWN/UP wrist to obtain a positive dot product of the $\mathbf{s}$ and $\mathbf{y}_5$ (or $\mathbf{z}_4$) unit vectors:

$$WRIST = \begin{cases} +1 \ ; & \text{if } \mathbf{s}\cdot\mathbf{z}_4 > 0 \\ -1 \ ; & \text{if } \mathbf{s}\cdot\mathbf{z}_4 < 0 \end{cases} = sign(\mathbf{s}\cdot\mathbf{z}_4) \tag{60}$$

If $\mathbf{s}\cdot\mathbf{z}_4 = 0$, then the WRIST indicator can be found from:

$$WRIST = \begin{cases} +1 \ ; & \text{if } \mathbf{n}\cdot\mathbf{z}_4 > 0 \\ -1 \ ; & \text{if } \mathbf{n}\cdot\mathbf{z}_4 < 0 \end{cases} = sign(\mathbf{n}\cdot\mathbf{z}_4) \tag{61}$$

Combining Eqs. 60 and 61, we have

$$WRIST = \begin{cases} sign(\mathbf{s}\cdot\mathbf{z}_4) \ ; \ \text{if } \mathbf{s}\cdot\mathbf{z}_4 \neq 0 \\ sign(\mathbf{n}\cdot\mathbf{z}_4) \ ; \ \text{if } \mathbf{s}\cdot\mathbf{z}_4 = 0 \end{cases} = \begin{cases} +1 \ ; \ \mathit{WRIST\ DOWN} \\ -1 \ ; \ \mathit{WRIST\ UP} \end{cases} \tag{62}$$

These decision equations provide a verification of the arm solution. We use them to preset the arm configuration in the direct kinematics and then use the arm configuration indicators to find the inverse kinematics solution. (See Figure 12)

6. COMPUTER SIMULATION

A computer program was written to verify the validity of the inverse solution of the PUMA robot arm shown in Figure 2. The software initially generates all the locations in the workspace of the robot within the joint angles limits. They are inputed into the direct kinematics routine to obtain the arm matrix **T**. These joint angles are also used to compute the decision equations to obtain the three arm configuration indicators. These indicators together with the arm matrix **T** are fed into the inverse solution routine to obtain the joint angle solution which should agree to the joint angles fed into the direct kinematics routine previously. A computer simulation block diagram is shown in Figure 12 and a list of the computer program written in PASCAL is given in the APPENDIX.

7. CONCLUSION

The kinematics and inverse kinematics problems of a PUMA robot arm have been discussed. The inverse solution is determined with the assistance of three arm configuration indicators (ARM, ELBOW, and WRIST). There are eight solutions to a six-joint PUMA robot arm - four solutions for the first three joints and for each arm configuration two more solutions for the last three joints. Computer simulation of the direct and inverse kinematics showed that the above derived arm solution is correct. This approach, with appropriate modification and adjustment, can be generalized to other simple industrial robots with rotary joints.

8. ACKNOWLEDGEMENT

The authors would like to thank Robert Horner who wrote a "C" program to verify the above direct and inverse kinematics equations together with their corresponding decision equations. The authors also would like to thank Richard Jungclas who wrote the above kinematic equations in PASCAL and verified

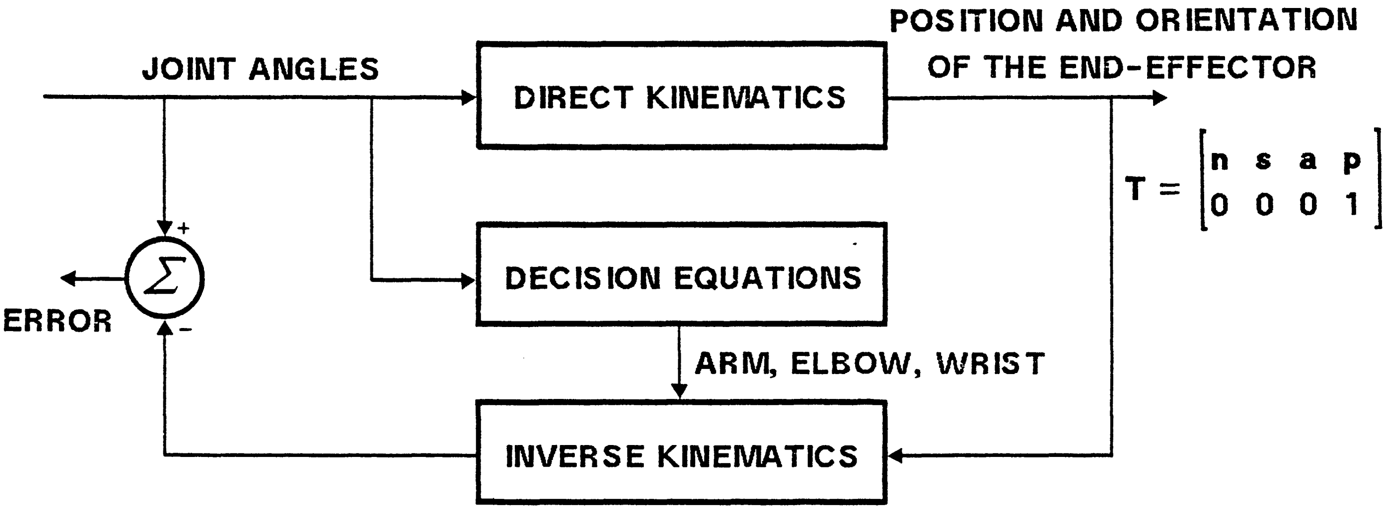


Figure 12. Computer Simulation of Joint Solution

them by controlling a PUMA robot arm from an IBM PC.

9. REFERENCES

1. D. L. Pieper, "The Kinematics of Manipulators Under Computer Control," Computer Science Department, Stanford University, Artificial Intelligence Project Memo No. 72, October 24, 1968.
2. R. P. Paul, "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm," Stanford Artificial Intelligence Laboratory Memo AIM-177, November 1972.
3. R. A. Lewis, "Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions," Technical Memo 33-679, Jet Propulsion Lab, March 15, 1974.
4. R. P. Paul, B. E. Shimano, and G. Mayer, "Kinematic Control Equations for Simple Manipulators," *IEEE Transactions of Systems, Man, and Cybernetics,* Vol. SMC-11, No. 6, June 1981, pp 449-455.
5. C. S. G. Lee, "Robot Arm Kinematics, Dynamics, and Control," *Computer,* Vol. 15, No. 12, December 1982, pp 62-80.
6. R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control,* M.I.T. Press, 1981, pp 50-84.
7. J. J. Uicker, Jr., J. Denavit, and R. S. Hartenberg, "An Iterative Method for the Displacement Analysis of Spatial Mechanisms," *Trans. of ASME, Journal of Applied Mechanics,* Vol. 31, Series E, 1964, pp. 309-314.
8. V. Milenkovic and B. Huang, "Kinematics of Major Robot Linkages," *Proceedings of the 13th International Symposium on Industrial Robots,* April 17-19, 1983, Chicago, Illinois, pp 16-31 to 16-47.
9. J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Journal of Applied Mechanics,* June 1955, pp. 215-221.

```
JG IC Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
 00           {$linesize:132,$pagesize:60}
              {$title:'Main PUMA Program (main.pas)',$subtitle:'Last change 4-3-84'}
              {$SPEED,$DEBUG-,$list+,$INDEXCK-,$NILCK-,$RANGECK-,$STACKCK-,$OCODE-}

              {
              ------------------------------------------------------------------------

                                      University of Michigan
                                      College of Engineering
                          Center for Robotics and Integrated Manufacturing
                                      Robot Systems Division
                                  2510 East Engineering Building
                                        Ann Arbor, MI 48109
                                          (313) 764-4343

                     Copyright 1984, The University of Michigan, All Rights Reserved


                                  Written by:  Richard M. Jungclas

              ------------------------------------------------------------------------

                   This system is used for the verification and development of the
                   direct kinematics and the inverse kinematics solutions for a six
                   jointed PUMA robot.  The system uses "A Geometric Approach in
                   Solving the Inverse Kinematics of PUMA Robots" developed by
                   C.S.G. Lee and M. Ziegler of the Electrical and Computer Engineering
                   department of the University of Michigan.

                   The actual PUMA routines were developed for an offline robot
                   programming project using the IBM PC being developed at the
                   Robot Systems Division of CRIM.  As a result these solutions
                   have been extensively tested and compared with the soutions
                   reported by VAL II.  We have found the solutions from the IBM PC
                   are within +/- 0.005 of a degree for the angles and within
                   +/- 0.005 of a millimeter for positions reported by VAL II. Generally,
                   the IBM PC gives solutions within +/- 0.001 of a degree or of a
                   millimeter.

                   The actual interface given here is a bit simplistic, but serves to
                   illustrates how the PUMA routines are used. While the interface
                   given below does not allow specifying of either the tool to mount
                   transformation or the robot reference to world transformation, the
                   solutions implemented allow for these transformations. The interface
                   assumes an identity transformation for the tool to mount transformation
                   and assumes that the world coordinate frame is a the base of link0
                   but oriented the same as the "shoulder" (link1) coordinate frame.

                   The interface uses a menu driven by single character inputs. The
                   menu is displayed on the top, left part of the screen. The key
                   used to select the menu item is given in parenthesis. The current
                   system status is diplayed on the top, right portion of the screen.
                   Data, various prompts and error messages are display on the lower
```

```
        half of the screen.

        Locations can be specified in either world cartesian, robot cartesian
        or robot joint coordinates frames. The default is the world cartesian
        coordinate frame. The robot cartesian coordinate frame is exactly the
        as the VAL II "trans" type locations. The joint coordinate frame is
        exactly the same as the VAL II "precision point" locations. Locations
        reported by all three types. The current type is display in the
        status area of the menu.

        The interface allows you to assign symbolic names of up to 12 characters
        to any location. The "type" of the location is determined by the
        current type setting at the time the symbol is defined and cannot be
        changed. There is no method at the moment of preserving the symbolic
        names between sessions.

        The current PUMA arm configuration is also display in the status area.



        The main commands are:

           Move             Moves the robot to location specified by either a
                            symbolic location or directly from the keyboard.
                            Entries from the keyboard use the "type" from the
                            current setting. The location in all three types is
                            reported for valid solutions.

           Name             Names the current location. The type of the symbolic
                            location is the "type" from the current setting.

           Robot Config.    Starts a submenu allowing changes of all the
                            Robot Configuration settings.

           Where            Reports the current location in all three types

           Exit             Terminates the program.



        The Robot Configuration commands are:

           Left/Right       Changes the PUMA arm configuration to Left arm or
                            Right arm.

           Above/Below      Changes the PUMA arm configuration to elbow Above
                            the wrist or elbow Below the wrist.

           Up/Down          Changes the PUMA arm configuration to wrist Up or
                            wrist Down.

           Flip/Noflip      Allows the wrist configuration to changed or not.

```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          107           Joint/Cartesian Specifies either Joint or Cartesian coordinate
          108                           frames. When cartesain coordinates are specified
          109                           a World/Robot menu item will appear to allow
          110                           selection of the type of cartesian coordinate frames.
          111
          112           Robot/World     Specifies the type of cartesian coordinates. Only
          113                           present if Cartesian coordinate are choosen.
          114
          115           Trace/Notrace   Permits the tracing of valid location in a file
          116                           named PUMA.DBG on the current directory.
          117
          118           Debug/Production Permits the tracing of debugging information in
          119                           a file named PUMA.DBG on the current directory.
          120
          121
          122
          123       The system uses a standard device call sercom as a mean of collecting
          124       debugging information, data, etc.. By default this the file PUMA.DBG
          125       is assigned to this device during initialization.
          126
          127   }
          128
            0   {$INCLUDE:'global.inc'}
   10     157   {$LIST+}
            0   {$INCLUDE:'debug.inc'}
   10      23   {$LIST+}
            0   {$INCLUDE:'menu.inc'}
   10      48   {$LIST+}
          132
          133   {$title:'Main PUMA Program (main.pas)',$subtitle:'Last change 4-3-84'}
          134
          135
   10     136   program robot(input,output);
          137
   10     138   uses debug;
   10     139   uses globals;
   10     140   uses menu_functions;
          141
          142
          143
   10     144   var
   10     145           last_move,                      !flag for last move
   10     146           invalid_command,                !flag for bad command
   10     147           leave_pgm: boolean;             !flag for program termination
   10     148           command,                        !char command from user
   10     149           spec: char;
          150
          151
          152
   10     153   var [public]
   10     154           sercom: file1;                  !Master debugging file
          155
          156
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   10    157    procedure initialize; external;

 Symtab  157    Offset Length   Variable - INITIALIZE
                -    2      2   Return offset, Frame length

   10    158    Procedure config_robot; external;

 Symtab  158    Offset Length   Variable - CONFIG_ROBOT
                -    2      2   Return offset, Frame length

   10    159    Procedure writeloc(var dev: file1); external;

 Symtab  159    Offset Length   Variable - WRITELOC
                -    4      2   Return offset, Frame length
                +    6      2   DEV                                 :File    VarP

   10    160    Procedure nameloc; external;

 Symtab  160    Offset Length   Variable - NAMELOC
                -    2      2   Return offset, Frame length

   10    161    function  move(var tracefil:file1): boolean; external;

 Symtab  161    Offset Length   Variable - MOVE
                -    4      4   Return offset, Frame length
                -    2      1   (function return)                   :Boolean
                +    6      2   TRACEFIL                            :File    VarP

 MOVE
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          162   {$PAGE+}
    10    163   begin          { MAIN }
          164
          165   {
          166           Here is where we wake up.  The first things that have to be done is
          167           to initialze the system.
          168   }
          169
    11    170   initialize;
          171
          172   {
          173           This is the root level of the menu.
          174
          175           leave_pgm is a flag for program termination.  When it is set to
          176           true, program execution is done.
          177
          178           Invalid_command is a flag that is used within each menu in the
          179           system. When it is true, it indicates that the PREVIOUS command the
          180           user typed was invalid. This flag is used in determining whether or
          181           not the prompt error should be erased and the menu reprinted. If the
          182           last user command was invalid, there is probably an error message in
          183           prompt area, meaning the user should be prompted without clear that
          184           area first.
          185
          186   }
          187
    11    188   last_move := false;
    11    189   leave_pgm := false;
    11    190   invalid_command := false;
          191
    11    192   while not leave_pgm do begin
          193
    12    194           if (not invalid_command)
    12    195              then begin
    13    196                   display_menu(menu_flag,'PUMA');
    13    197                   menu_item('Robot configuration');
    13    198                   menu_item('Move');
    13    199                   menu_item('Name this location');
    13    200                   menu_item('Where');
    13    201                   menu_item('Exit the program');
    13    202                   end
    12    203              else invalid_command := false;
          204
    12    205           if last_move
    13    206              then begin;
    13    207                   data_prompt;
  / 13    208                   writeloc(output);
    13    209                   last_move := false;
    12    210                   end;
          211
    12    212           command_prompt;
    12    213           write('Enter PUMA command: ');
    12    214           repeat until getc(command,spec);
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   12    215            data_prompt;
         216
   13    217            case command of
         218
         219
   13    220               'm','M': last_move := move(sercom);  !Move the robot
         221
         222
   13    223               'n','N': nameloc;                    !Name to current location
         224
         225
   13    226               'r','R': config_robot;               !Change robot configuration
         227
         228
   14    229               'w','W': begin;                      !Report robot location
   14    230                    writeln('Robot location:');
 / 14    231                    writeloc(output);
   13    232                    end;
         233
         234
   13    235               'e','E' : leave_pgm := true;         !Program termination
         236
         237
         238
   13    239               otherwise begin                      !Invalid commands
   14    240                    invalid_command := true;
   14    241                    write('(',command,') is an invalid command')
   13    242                    end;
         243
   12    244               end;
         245
   11    246            end;
         247
   11    248    cls;
         249
   00    250    end.

 Symtab   250   Offset Length   Variable
                     0    666   Return offset, Frame length
                     0     64   TO                            :Array   Static Extern
                     0     64   H                             :Array   Static Extern
                     0     64   HI                            :Array   Static Extern
                     0     64   TOI                           :Array   Static Extern
                    28      1   SPEC                          :Char    Static
                     0     14   CONFIG                        :Array   Static Extern
                     0     24   THETA                         :Array   Static Extern
                    30    636   SERCOM                        :File    Static Public
                     0      8   VERSION                       :Array   Static Extern
                     0     24   ROB_XYZ                       :Array   Static Extern
                    26      1   COMMAND                       :Char    Static
                     0      2   FIRST_STR                     :Pointer Static Extern
                     0      1   MENU_FLAG                     :Boolean Static Extern
                    20      1   LAST_MOVE                     :Boolean Static
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
                     24       1   LEAVE_PGM                        :Boolean Static
                      0       2   MENU_CURSOR                      :Integer Static Extern
                      0       2   COORDS_TYPE                      :Integer Static Extern
                     22       1   INVALID_COMMAND                  :Boolean Static


                Errors  Warns  In Pass One
                     0      0
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    00      1   {$linesize:132,$pagesize:60}
            2   {$title:'Main program routines(routines.pas)',$subtitle:'Last change: 4-3-84'}
            3   {$SPEED,$DEBUG-,$list+,$INDEXCK-,$NILCK-,$RANGECK-,$STACKCK-,$OCODE-}
            4
            5   {
            6   -----------------------------------------------------------------------
            7
            8                              University of Michigan
            9                              College of Engineering
           10                  Center for Robotics and Integrated Manufacturing
           11                              Robot Systems Division
           12                          2510 East Engineering Building
           13                                Ann Arbor, MI 48109
           14                                  (313) 764-4343
           15
           16           Copyright 1984, The University of Michigan, All Rights Reserved
           17
           18
           19                         Written by:  Richard M. Jungclas
           20
           21   -----------------------------------------------------------------------
           22   }
           23
           24
            0   {$INCLUDE:'global.inc'}
   10     157   {$LIST+}
            0   {$INCLUDE:'debug.inc'}
   10      23   {$LIST+}
            0   {$INCLUDE:'menu.inc'}
   10      48   {$LIST+}
            0   {$INCLUDE:'puma.inc'}
   10      35   {$LIST+}
           29
           30   {$title:'Main program routines(routines.pas)',$subtitle:'Last change: 4-3-84'}
           31
   00      32   Module main_routines;
           33
   10      34   uses globals;
   10      35   uses debug;
   10      36   uses menu_functions;
   10      37   uses puma;
           38
   10      39   var
   10      40           sercom [extern]: file1;
           41
           42
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           43   {$PAGE+}
    10     44   procedure initialize;
           45
           46   {
           47           PROCEDURE INITIALIZE;
           48
           49           Purpose:        Performs all the initializations necessary for the
           50                           user to begin running the system.
           51
           52   }
           53
    20     54   var
    20     55           i,                      !loop counters
    20     56           j: integer;
    20     57           invertible: boolean;
    20     58           temp_STR: STR_ptr;
           59
           60
    20     61   begin
           62
  = 21     63   sercom.trap := true;            !Setup standard debugging file
  / 21     64   assign(sercom, 'PUMA.dbg');
  / 21     65   rewrite(sercom);
    21     66   if sercom.errs > 0
    21     67      then writeln('Unable to open sercom file! Code=', sercom.errs:1);
           68
  = 21     69   version := 'AR044.0';           !System version
           70
           71
           72   {
           73           Initialize the robot data structures
           74   }
    21     75   init_robot;
           76
  = 21     77   config[0]  :=  1;       !right arm
  = 21     78   config[1]  :=  1;       !above elbow
  = 21     79   config[2]  := -1;       !wrist up
  = 21     80   config[3]  := -1;       !noflip (wrist)
  = 21     81   config[4]  :=  0;       !initially  valid solution
  = 21     82   config[5]  :=  0;       !production
  = 21     83   config[6]  :=  0;
           84
           85   {       Robot(Shoulder) coords w/ Null Tool    }
  = 21     86   xyz[1]  :=  -20.32;
  = 21     87   xyz[2]  :=  149.09;
  = 21     88   xyz[3]  :=  921.12;
  = 21     89   xyz[4]  :=   90.0;
  = 21     90   xyz[5]  :=  -90.0;
  = 21     91   xyz[6]  :=    0.0;
  = 21     92   coords_type := robot_type;
  = 21     93   theta[1]  :=   0.0;
  = 21     94   theta[2]  := -90.0;
  = 21     95   theta[3]  :=  90.0;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
  = 21     96   theta[4] :=   0.0;
  = 21     97   theta[5] :=   0.0;
  = 21     98   theta[6] :=   0.0;
           99
          100   {
          101           Find the initial position of robot
          102   }
    21    103   homotran;
    21    104   inverse;
          105
          106
    21    107   if not joint_check
    21    108      then writewt('Initial robot configuration bad');
          109
          110
  = 21    111   first_STR := nil;       !No symbols to start
          112
          113   {
          114           Predefined symbols
          115   }
    21    116   new(temp_STR);
    21    117   temp_STR@.symname := 'ready';
    21    118   temp_STR@.data := theta;
    21    119   temp_STR@.ctype := joint_type;
    21    120   temp_STR@.used := true;
    21    121   temp_STR@.next_STR := first_STR;
  = 21    122   first_STR := temp_STR;
          123
          124
  = 21    125   menu_flag := true;              !Full menu to start
          126
  = 21    127   coords_type := world_type;      !default user to world coords
          128
    10    129   end;

Symtab    129   Offset Length   Variable - INITIALIZE
                -     2    10   Return offset, Frame length
                -     2     2   I                               :Integer
                -     4     2   J                               :Integer
                -     6     1   INVERTIBLE                      :Boolean
                -     8     2   TEMP_STR                        :Pointer
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          130   {$PAGE+}
   10     131   Procedure config_robot;
          132
          133   {
          134           PROCEDURE CONFIG_ROBOT;
          135
          136           Purpose:        Acts as the driver for the robot configuration
          137                           commands.
          138
          139   }
          140
   20     141   var
   20     142           command,                        !to hold user's command
   20     143           spec: char;
   20     144           leave,                          !flag for returning up one level
   20     145           invalid_command: boolean;       !valid command flag
          146
          147
   20     148   begin
          149   {
          150           This menu contains the robot configuration programming commands.
          151   }
          152
   21     153   leave := false;
   21     154   invalid_command := false;
          155
   21     156   while not leave do
   21     157           begin
   22     158           if not invalid_command
   22     159              then begin
          160
 / 23     161                   display_menu(menu_flag,'PUMA/config');
   23     162                   bmenu_item('PUMA menu');
   23     163                   if config[0] = 1
   23     164                      then menu_item('Left arm ')
   23     165                      else menu_item('Right arm');
   23     166                   if config[1] = 1
   23     167                      then menu_item('Below elbow')
   23     168                      else menu_item('Above elbow');
   23     169                   if config[2] = 1
   23     170                      then menu_item('Up  wrist ')
   23     171                      else menu_item('Down wrist');
   23     172                   if config[3] = 1
   23     173                      then menu_item('Noflip wrist')
   23     174                      else menu_item('Flip  wrist ');
   24     175                   case coords_type of
   24     176                      world_type,robot_type:
   25     177                            begin;
   25     178                            menu_item('Joint coordinates');
   25     179                            if coords_type = world_type
   25     180                               then gmenu_item('M','Robot coordinates')
   25     181                               else menu_item('World coordinates');
   24     182                            end;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    24    183                      joint_type:
    24    184                           menu_item('Cartesian coordinates');
    24    185                      otherwise
    24    186                           ;
    23    187                      end;
    23    188                   if config[5] = 0
    23    189                      then gmenu_item('E','Debug mode')
    23    190                      else menu_Item('Production mode');
    23    191                   if config[6] = 0
    23    192                      then menu_item('Trace moves')
    23    193                      else gmenu_item('O', 'No trace');
          194
    23    195                   display_status;
    22    196                   end;
          197
    22    198           invalid_command := false;
    22    199           command_prompt;
    22    200           write('Enter robot programming command: ');
    22    201           repeat until getc(command,spec);
    22    202           data_prompt;
          203
    23    204           case command of
          205
          206
    23    207              '-': leave := true;                  !Back to PUMA menu
          208
          209
          210
 = 23    211              'l','L': config[0] := -1;            !Left Arm
          212
 = 23    213              'r','R': config[0] := 1;             !Right arm
          214
          215
          216
 = 23    217              'b','B': config[1] := -1;            !Below elbow
          218
 = 23    219              'a','A': config[1] := 1;             !Above elbow
          220
          221
          222
 = 23    223              'd','D': config[2] := 1;             !Wrist down
          224
 = 23    225              'u','U': config[2] := -1;            !Wrist up
          226
          227
          228
 = 23    229              'f','F': config[3] := 1;             !Flip of wrist allowed
          230
 = 23    231              'n','N': config[3] := -1;            !Noflip of wrist allowed
          232
          233
          234
 = 23    235              'c','C': coords_type := robot_type;  !punch robot cartesian coords
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
         236
 = 23    237                'j','J': coords_type := joint_type;  !punch joint coords.
         238
         239
         240
         241
 = 23    242                'm','M': coords_type := robot_type;  !Coordinates
         243
 = 23    244                'w','W': coords_type := world_type;
         245
         246
         247
 = 23    248                'e','E': config[5] := 1;     !Debug Mode
         249
 = 23    250                'p','P': config[5] := 0;     !Production mode
         251
         252
         253
   23    254                't','T': begin               !Trace on
 = 24    255                     config[6] := 1;;
   24    256                     writeln(sercom);
   23    257                     end;
         258
 = 23    259                'o','O': config[6] := 0;     !Trace off
         260
         261
         262
   23    263                otherwise begin
   24    264                     write('(',command,') is an invalid command');
   24    265                     invalid_command := true;
   23    266                     end;
         267
   22    268                end;
   21    269           end;
   10    270    end;

Symtab   270    Offset Length   Variable - CONFIG_ROBOT
                -     2     10   Return offset, Frame length
                -     2      1   COMMAND                        :Char
                -     4      1   SPEC                           :Char
                -     6      1   LEAVE                          :Boolean
                -     8      1   INVALID_COMMAND                :Boolean

CONFIG_ROBOT
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          271   {$PAGE+}
    20    272   procedure writeloc(var dev: file1);
          273
          274   {
          275           PROCEDURE WRITELOC(var DEV: file1);
          276
          277           Purpose:        Writes out the robot location in robot coords.,
          278                           world coords. and joint angles.
          279
          280   }
          281
    20    282   var
    20    283           T2, T3, T4, T5, T6: matrix;
          284
          285
    20    286   begin
          287
  = 21    288   itheta := theta;
    21    289   tmats(T2, T3, T4, T5, T6);
  = 21    290   rob_xyz := get_xyzoat(T6);
          291
          292   {add base and hand displacements}
  = 21    293   tmatrix := mat_mult(T6,T0);
  = 21    294   tmatrix := mat_mult(H,tmatrix);
  = 21    295   xyz := get_xyzoat(tmatrix);
          296
    21    297   writeln(dev,'world  X=',xyz[1]:8:3,' Y=',xyz[2]:8:3,' Z=',xyz[3]:8:3,
    21    298                   ' O=',xyz[4]:8:3,' A=',xyz[5]:8:3,' T=',xyz[6]:8:3);
    21    299   writeln(dev,'robot  X=',rob_xyz[1]:8:3,' Y=',rob_xyz[2]:8:3,' Z=',rob_xyz[3]:8:3,
    21    300                   ' O=',rob_xyz[4]:8:3,' A=',rob_xyz[5]:8:3,' T=',rob_xyz[6]:8:3);
    21    301   writeln(dev,'joint  1=',theta[1]:8:3,' 2=',theta[2]:8:3,' 3=',theta[3]:8:3,
    21    302                   ' 4=',theta[4]:8:3,' 5=',theta[5]:8:3,' 6=',theta[6]:8:3);
          303
    21    304   writeln(dev);
    10    305   end;

Symtab    305   Offset Length   Variable - WRITELOC
                -    4    386   Return offset, Frame length
                +    6      2   DEV                            :File    VarP
                -   64     64   T2                             :Array
                -  128     64   T3                             :Array
                -  192     64   T4                             :Array
                -  256     64   T5                             :Array
                -  320     64   T6                             :Array

WRITELOC
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          306   {$PAGE+}
    10    307   Procedure nameloc;
          308
          309   {
          310           PROCEDURE NAMELOC;
          311
          312           Purpose:        Gives the current valid robot location a symbolic
          313                           name.
          314
          315   }
          316
    20    317   var
    20    318           command,
    20    319           spec:           char;
    20    320           temp_STR:       STR_ptr;
    20    321           name:           name_lstr;
    20    322           input_line:     consol_input_lstr;
    20    323           temp_type:      integer;
          324
    20    325   begin
    21    326   write('Enter robot location name? ');
    21    327   readln(input_line);
    21    328   trim(input_line);
    21    329   name:=input_line;
          330
    21    331   temp_STR := find_STR(name);
          332
    21    333   if temp_STR <> nil
    22    334      then begin;          !Matches existing name
    32    335           if temp_STR@.used
    22    336              then write('Overwrite existing used location? ')
    22    337              else write('Overwrite existing unused location? ');
    22    338           repeat until getc(command,spec);
    22    339           if (command <> 'Y') and (command <> 'y')
    23    340              then begin;
    23    341                   data_prompt;
    23    342                   write('Location not changed!');
 *  23    343                   return;
    23    344                   end
    32    345              else if temp_STR@.used
    22    346                   then begin
    23    347                        data_prompt;
    23    348                        write('Previous references use the redefined location!');
    23    349                        end
    22    350                   else data_prompt;
    22    351           temp_type := temp_STR@.ctype;   !Used the existing coords type
    22    352           end
          353
    22    354      else begin;          !allocate new symbol record
    22    355           temp_type := world_type;
    23    356           case coords_type of
          357
    23    358              world_type, robot_type:
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    23    359                   temp_type := coords_type;
          360
    23    361              joint_type:
    23    362                   temp_type := joint_type;
          363
    22    364              end;
          365
    22    366         new(temp_STR);
    22    367         temp_STR@.symname := name;
    22    368         temp_STR@.ctype := coords_type;          !current default setting
    22    369         temp_STR@.used := false;
    22    370         temp_STR@.next_STR := first_STR;
  = 22    371         first_STR := temp_STR;
    21    372         end;
          373
    22    374   case temp_type of
          375
    22    376      world_type:
    22    377         begin                          {add base and hand displacements}
  = 23    378         xyz := rob_xyz;
    23    379         homotran;
  = 23    380         tmatrix := mat_mult(tmatrix,T0);
  = 23    381         tmatrix := mat_mult(H,tmatrix);
    23    382         temp_STR@.data := get_xyzoat(tmatrix);
    22    383         end;
          384
    22    385      robot_type:
    22    386         temp_STR@.data := rob_xyz;
          387
    22    388      joint_type:
    22    389         temp_STR@.data := theta;
          390
    21    391      end;
          392
    10    393   end;

 Symtab   393   Offset Length   Variable - NAMELOC
                -     2    170   Return offset, Frame length
                -     2      1   COMMAND                          :Char
                -     4      1   SPEC                             :Char
                -    20     14   NAME                             :Array
                -     6      2   TEMP_STR                         :Pointer
                -   102     82   INPUT_LINE                       :Array
                -   104      2   TEMP_TYPE                        :Integer
```

NAMELOC

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          394   {$PAGE+}
    20    395   function move(var tracefil: file1): boolean;
          396
          397   {
          398           FUNCTION MOVE(var TRACEFIL: file1): boolean;
          399
          400           Purpose:        Moves robot to new location returning true
          401                           if the location was within robot's workspace.
          402
          403   }
          404
    20    405   var
    20    406           command,                                !to hold user's command
    20    407           spec:           char;
    20    408           leave,                                  !flag for returning up one level
    20    409           full:           boolean;                !command menu flag
    20    410           x,                                      !Generalize  robot coordinates
    20    411           y,
    20    412           z,
    20    413           o,
    20    414           a,
    20    415           t:              real;
    20    416           temp_type,
    20    417           temp_coord_type,
    20    418           i:              integer;
    20    419           temp_STR:       STR_ptr;
    20    420           name:           name_lstr;
    20    421           input_line:     consol_input_lstr;
          422
    20    423   begin
          424
    21    425   leave := false;
    21    426   full := false;
  = 21    427   menu_flag := true;
    21    428   cls;
          429
    21    430   while not leave do
    21    431           begin
          432
    22    433           while not leave do
    22    434                   begin
    23    435                   display_menu(full,'Move command');
    23    436                   bmenu_item('PUMA menu');
    23    437                   menu_item('Direct from Keyboard');
    23    438                   menu_item('Named Location');
          439
    23    440                   leave := true;
    23    441                   command_prompt;
    23    442                   write('Enter move input selection: ');
    23    443                   repeat until getc(command,spec);
    23    444                   data_prompt;
          445
    24    446                   case command of
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          447
 *  24    448                       '-': return;                 !Backout
          449
    25    450                       'd','D': begin;              !Keyboard selection
          451
    25    452                           input_line := NULL;
    25    453                           temp_type := coords_type;
    26    454                           case coords_type of
          455
    27    456                              world_type, robot_type: begin;
    27    457                                   If coords_type = world_type
    27    458                                      then writeln('World XYZOAT location')
    27    459                                      else writeln('Robot XYZOAT location');
    27    460                                   write('Enter X position: ');
    27    461                                   readln(x);
    27    462                                   write('Enter Y position: ');
    27    463                                   readln(y);
    27    464                                   write('Enter Z position: ');
    27    465                                   readln(z);
    27    466                                   write('Enter O angle: ');
    27    467                                   readln(o);
    27    468                                   write('Enter A angle: ');
    27    469                                   readln(a);
    27    470                                   write('Enter T angle: ');
    27    471                                   readln(t);
          472
 =  27    473                                   xyz[1]  := x;
 =  27    474                                   xyz[2]  := y;
 =  27    475                                   xyz[3]  := z;
          476
 =  27    477                                   xyz[4]  := o;
 =  27    478                                   xyz[5]  := a;
 =  27    479                                   xyz[6]  := t;
    26    480                                   end;
          481
    27    482                             joint_type: begin;
    27    483                                   writeln('Joint position');
    27    484                                   write('Enter J1 angle: ');
 /  27    485                                   readln(itheta[1]);
    27    486                                   write('Enter J2 angle: ');
 /  27    487                                   readln(itheta[2]);
    27    488                                   write('Enter J3 angle: ');
 /  27    489                                   readln(itheta[3]);
    27    490                                   write('Enter J4 angle: ');
 /  27    491                                   readln(itheta[4]);
    27    492                                   write('Enter J5 angle: ');
 /  27    493                                   readln(itheta[5]);
    27    494                                   write('Enter J6 angle: ');
 /  27    495                                   readln(itheta[6]);
    26    496                                   end;
          497
    25    498                              end;
          499
```

```
JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   24    500                           end;
         501
   25    502                      'n','N': begin;
   25    503                           write('Enter robot location name? ');
   25    504                           readln(input_line);
   25    505                           trim(input_line);
   25    506                           name:=input_line;
         507
   25    508                           if (name = NULL)
   25    509                              then begin
   26    510                                   leave := false;
   26    511                                   data_prompt;
   26    512                                   end
   26    513                              else begin;
   26    514                                   temp_STR := first_STR;
   26    515                                   while temp_STR <> nil and then temp_STR@.symname <> name do
   26    516                                           temp_STR := temp_STR@.next_STR;
         517
   26    518                                   if temp_STR <> nil
   27    519                                      then begin;          !Matches existing name
   27    520                                           temp_STR@.used := true;
   27    521                                           temp_type := temp_STR@.ctype;
   28    522                                           case temp_type of
   28    523                                              world_type, robot_type:
 = 28    524                                                   xyz := temp_STR@.data;
   28    525                                              joint_type:
 = 28    526                                                   Itheta := temp_STR@.data;
   27    527                                              end;
   27    528                                           end
   27    529                                      else begin;          !Symbol not found
   27    530                                           data_prompt;
   27    531                                           leave := false;
   27    532                                           writeln('Location ''',name, ''' not found');
   26    533                                           end;
   25    534                                   end;
   24    535                           end;
         536
   25    537                      otherwise begin;
   25    538                           writeln('(',command,') is an invalid selection');
   25    539                           leave := false;
   24    540                           end;
   23    541                      end;
         542
   22    543                  end;
         544
   22    545          clear_upper;
   22    546          data_prompt;
   22    547          position_cursor(15,0);
         548
   23    549          case temp_type of
         550
   23    551             world_type, robot_type:
   23    552                  begin
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   00       1   {$LINESIZE:132 $PAGESIZE:60}
            2   {$Title:'PUMA robot routines',$subtitle:'Last change: 4-3-84 rmj'}
            3   {$SPEED,$DEBUG-,$LIST+,$INDEXCK-,$NILCK-,$RANGECK-,$STACKCK-,$OCODE-}
            4
            5   {$MESSAGE:'Enter 1 for debugging information, 0 for normal operation'
            6   $INCONST:puma_debug,$INCONST:tmats_debug,$INCONST:homo_debug}
            7
            8   {
            9   -----------------------------------------------------------------------
           10
           11                         University of Michigan
           12                         College of Engineering
           13                Center for Robotics and Integrated Manufacturing
           14                          Robot System Division
           15                       2514 East Engineering Building
           16                           Ann Arbor, MI 48109
           17                             (313) 764-4343
           18
           19         Copyright 1984, The University of Michigan, All Rights Reserved
           20
           21
           22                      Written by:  Richard M. Jungclas
           23
           24   -----------------------------------------------------------------------
           25   }
           26
            0   {$INCLUDE:'global.inc'}
   10     157   {$LIST+}
            0   {$INCLUDE:'debug.inc'}
   10      23   {$LIST+}
            0   {$INCLUDE:'puma.inc'}
   10      35   {$LIST+}
           30
           31   {$Title:'PUMA robot routines',$subtitle:'Last change: 4-3-84 rmj'}
           32
   10      33   implementation of puma;
           34
   10      35   Uses globals;
   10      36   Uses debug;
           37
   10      38   function a2srqq(consts a,b: real): real; external;

 Symtab    38   Offset Length   Variable - A2SRQQ
                -   12      2   Return offset, Frame length
                +    6      4   (function return)                  :Real
                +   12      4   A                                  :Real    Const  VarsP
                +    8      4   B                                  :Real    Const  VarsP

           39
           40
   10      41   var
   10      42           sercom [extern]: file1;
   10      43           max_degree: j_matrix;          !Joint limits
```

```
JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   10     44           min_degree: j_matrix;
   10     45           model_verts: rmat_ptr;
   10     46           A10,                                !link i to link i-1 transformations
   10     47           A21,
   10     48           A32,
   10     49           A43,
   10     50           A54,
   10     51           A65: matrix;
          52
   10     53   const
   10     54           D2 = 149.09;            !PUMA robot parameters
   10     55           A2 = 431.80;
   10     56           A3 = -20.32;
   10     57           D4 = 433.07;
   10     58           D6 =  56.25;
          59   {       Most of these constants are pre-calculated to maximize numerical
          60           accurray, which at best is limited to 7 decimal digits}
   10     61           f2_a2 = 863.60;         !2.0 * A2
   10     62           D2sq = 22227.83;        !D2 * D2
   10     63           A2sq = 186451.2;        !A2 * A2
   10     64           A3sq = 412.9024;        !A3 * A3
   10     65           D4sq = 187549.6;        !D4 * D4
   10     66           l1 = 433.5465;          !sqrt( l2 )
   10     67           l2 = 187962.5;          !D4*D4 + A3*A3
   10     68           l5 = 0.04686926;        !abs( A3 ) / l1
   10     69           l6 = 0.9989010;         !D4 / l1
   10     70           f2_a2_l1 = 374410.7;    !2.0 * A2 * l1
   10     71           A2_l2 = 374413.8;       !A2sq + l2
   10     72           A2_D4_A3 = -1511.287;   !A2sq - D4sq - A3sq
          73
   10     74           PI = 3.141593;
   10     75           twoPI = 6.283185;       !2.0 * PI
   10     76           f180_pi = 57.29578;     !180.0 / PI
   10     77           pi_180 = 0.01745329;    !PI / 180.0
          78
   10     79           epsilon = 0.001;
          80
```

A2SRQQ

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          81    {$PAGE+}
   10     82    procedure inverse;
          83
          84    {
          85            PROCEDURE INVERSE;
          86
          87            Purpose:        Calculates the inverse kinematics for the PUMA robot
          88                            arm. It is given the transformation matrix for the
          89                            position, and calculates the joint angles for that
          90                            position.
          91
          92            Calling convention:
          93                            inverse;
          94
          95            Global Variables:
          96                            xyz     Contains the proposed xyzoat position of
          97                                    the puma arm.
          98                            tmatrix The homogenous transformation matrix
          99                                    describing the proposed arm position.
         100                            itheta  Returns the inverse solution of the arm
         101                                    (ie.  each of the six joint angles).
         102
         103
         104    }
         105
   20    106    var
   20    107            i: integer;             !index
   20    108            px,py,pz,               !position of the hand
   20    109            ax,ay,az,               !approach vector components
   20    110            sx,sy,sz,               !sliding vector components
   20    111            nx,ny,nz,               !normal vector componets
   20    112            s1,s2,s3,s4,s5,s6,s23,  !Joint sines and cosines
   20    113            c1,c2,c3,c4,c5,c6,c23,
   20    114            pxsq,                   !px * px
   20    115            pysq,                   !py * py
   20    116            pzsq,                   !pz * pz
   20    117            rsq,                    !px*px + py*py + pz*pz - D2*D2
   20    118            r,                      !sqrt( rsq )
   20    119            sal,cal,sbt,cbt,        !Misc sine and cosines  theta 2 solution
   20    120            13,14,                  !Misc   "      "        theta 3    "
   20    121            omega,                  !orientation indicator
   20    122            z3ax,z3ay,z3az,         !z3 cross a components
   20    123            k1,k2,k3,k4,            !Misc constant terms
   20    124            t1,t2,t3,t4: real;
   20    125            arm,                    !ARM is negative(left)
   20    126            arm below,              !ARM * BELOW is negative
   20    127            k: boolean;             !WRIST * sign(Omega) is negative
         128
   20    129    begin
         130
 = 21    131    config[4] := 0;                 !assume no error
         132
         133    {
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
         134            First find t06 matrix, eg T06 := BI * T * HI where B is our T0 and T
         135            is the "world" transformation!
         136    }
    21   137    if coords_type = world_type
    22   138       then begin;                  !Base coords differs from robot coords.
  = 22   139            tmatrix := mat_mult(HI, tmatrix);
  = 22   140            tmatrix := mat_mult(tmatrix, TOI);
  = 22   141            xyz := get_xyzoat(tmatrix);
    21   142            end;
         143
    21   144    nx := tmatrix[1,1];
    21   145    ny := tmatrix[1,2];
    21   146    nz := tmatrix[1,3];
         147
    21   148    sx := tmatrix[2,1];
    21   149    sy := tmatrix[2,2];
    21   150    sz := tmatrix[2,3];
         151
    21   152    ax := tmatrix[3,1];
    21   153    ay := tmatrix[3,2];
    21   154    az := tmatrix[3,3];
         155
    21   156    px := tmatrix[4,1] - D6 * ax;
    21   157    py := tmatrix[4,2] - D6 * ay;
    21   158    pz := tmatrix[4,3] - D6 * az;
         159
    21   160    pxsq := px * px;
    21   161    pysq := py * py;
    21   162    pzsq := pz * pz;
         163
    21   164    k3 := pxsq + pysq;              !k3 = px*px + py*py
    21   165    k1 := k3 - d2sq;                !k1 = px*px + py*py - D2*D2
    21   166    if (k1 < 0)
    21   167       then begin
    22   168            k1 := -k1;
    22   169            if k1 > epsilon
    23   170               then begin;
    23   171                     writeln('Warning: Invalid Position (k1)');
  = 23   172                     config[4] := 1;
    22   173                     end;
    21   174            end;
    21   175    k2 := sqrt(k1);                 !k2 = sqrt(px*px + py*py - D2*D2)
    21   176    rsq := k1 + pzsq;               !rsq = px*px + py*py + pz*pz - D2*D2
    21   177    if (rsq < 0)
    21   178       then begin
    22   179            rsq := -rsq;
    22   180            if rsq > epsilon
    23   181               then begin;
    23   182                     writeln('Warning: Invalid Position (rsq)');
  = 23   183                     config[4] := 1;
    22   184                     end;
    21   185            end;
    21   186    r := sqrt(rsq);                 !r = sqrt(px*px + py*py + pz*pz - D2*D2)
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          187
   21     188   arm := (config[0] = -1);                         !Set ARM flag
   21     189   arm_below := (arm) xor (config[1] = -1);         !Set ARM * BELOW
          190
          191   {
          192           find theta sub 1
          193   }
   21     194   t1 := py*k2;
   21     195   t2 := px*k2;
   21     196   if not arm
   21     197      then begin
   22     198           t1 := -t1;
   22     199           t2 := -t2;
   21     200           end;
   21     201   s1 := t1 - px*d2;
   21     202   c1 := t2 + py*d2;
 = 21     203   itheta[1] := a2srqq(s1,c1);
   21     204   if abs(k3) < epsilon
   22     205      then begin;                    !division by zero (invalid position)
   22     206           s1 := 0;
   22     207           c1 := 0;
   22     208           end
   22     209      else begin;
   22     210           s1 := s1 / k3;
   22     211           c1 := c1 / k3;
   21     212           end;
   21     213   if abs(s1*s1 + c1*c1 - 1.0) > epsilon
   21     214      then writeln('Warning: Illegal Position (1) ');
          215   {
          216           find theta sub 2
          217   }
   21     218   if abs(r) < epsilon
   21     219      then t1 := -1.0                !division by zero (invalid position)
   22     220      else begin;
   22     221           sal := -pz / r;                               !sine alpha
   22     222           cal := k2 / r;                                !cosine alpha
   22     223           if not arm
   22     224              then cal := -cal;
   22     225           cbt := (A2_D4_A3 + rsq) / (f2_a2 * r);  !cosine beta
   22     226           t1 := 1.0 - cbt*cbt;
   21     227           end;
   21     228   if (t1 < 0)
   21     229      then begin
   22     230           t1 := -t1;
   22     231           if t1 > epsilon
   23     232              then begin;
   23     233                   writeln('Warning: Invalid Position (2)');
 = 23     234                   config[4] := 1;
   22     235                   end;
   21     236           end;
   21     237   sbt := sqrt(t1);                                !sine beta
   21     238   t2 := cal * sbt;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    21    240   if arm_below
    21    241      then begin
    22    242           t2 := -t2;
    22    243           t3 := -t3;
    21    244           end;
    21    245   s2 := sal*cbt + t2;
    21    246   c2 := cal*cbt - t3;
  = 21    247   1theta[2] := a2srqq(s2, c2);
    21    248   if abs(s2*s2 + c2*c2 - 1.0) > epsilon
    22    249      then begin;
    22    250           writeln('Warning: Invalid Position (2)');
  = 22    251           config[4] := 1;
    21    252           end;
          253
          254   {
          255           find theta sub 3
          256   }
    21    257   13 := (A2_12 - rsq) / f2_A2_11;                      !cosine phi
    21    258   t1 := 1.0 - 13*13;
    21    259   if (t1 < 0)
    21    260      then begin
    22    261           t1 := -t1;
    22    262           if t1 > epsilon
    23    263              then begin;
    23    264                   writeln('Warning: Invalid Position (3)');
  = 23    265                   config[4] := 1;
    22    266                   end;
    21    267           end;
    21    268   14 := sqrt(t1);                                      !sine phi
    21    269   if arm_below
    21    270      then 14 := -14;
    21    271   s3 := 14*15 - 13*16;
    21    272   c3 := 13*15 + 14*16;
  = 21    273   1theta[3] := a2srqq(s3, c3);
    21    274   if abs(s3*s3 + c3*c3 - 1.0) > epsilon
    22    275      then begin;
    22    276           writeln('Warning: Invalid Position (3)');
  = 22    277           config[4] := 1;
    21    278           end;
          279
          280   {
          281           Now for the wrist solution
          282   }
    21    283   t1 := 1theta[2] + 1theta[3];
    21    284   while t1 < 0.0 do t1 := t1 + twoPI;     !Needed by PC sin() and cos() fcns.
    21    285   s23 := sin(t1);
    21    286   c23 := cos(t1);
    21    287   if abs(s23*s23 + c23*c23 - 1.0) > epsilon
    22    288      then begin;
    22    289           writeln('Warning: Illegal Position (23)');
  = 22    290           config[4] := 1;
    21    291           end;
    21    292   t2 := s1 * c23;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    21    293   t3 := c1 * c23;
    21    294   omega := 0.0;
    21    295   z3ax := s1*s23*az - c23*ay;               !z3 x a components
    21    296   z3ay := c23*ax - c1*s23*az;
    21    297   z3az := c1*s23*ay - s1*s23*ax;
    21    298   if ((z3ax <> 0.0) or (z3ay <> 0.0) or (z3az <> 0.0))
    22    299      then begin;
    22    300           omega := sx*z3ax + sy*z3ay + sz*z3az;           !s * (z3 x a)
    22    301           if (abs(omega) < epsilon)
    22    302              then omega := nx*z3ax + ny*z3ay + nz*z3az;   !n * (z3 x a)
    21    303           end;
          304
    21    305   k := (config[2] = -1) xor ( omega < 0.0);              !WRIST * sign(omega)
          306
          307
    21    308   if (not k) and (config[3] = 1)         !Necessary to flip wrist!!!
    22    309      then begin;
 = 22    310           config[2] := - config[2];
    22    311           k := not k;
    21    312           end;
          313
          314   {
          315           find theta sub 4
          316   }
    21    317   s4 := c1*ay - s1*ax;
    21    318   c4 := t3*ax + t2*ay - s23*az;
    21    319   if k
    22    320      then begin;
    22    321           s4 := -s4;
    22    322           c4 := -c4;
    21    323           end;
    21    324   if (abs(s4) < epsilon) and (abs(c4) < epsilon)  !Degenerate case
 = 21    325      then itheta[4] := theta[4] * pi_180  !theta 4 already alligned, use current value
 = 21    326      else itheta[4] := a2srqq(s4,c4);
          327
    21    328   t1 := itheta[4];
    21    329   while t1 < 0.0 do t1 := t1 + twoPI;     !Needed by PC sin() and cos() fcns.
    21    330   s4 := sin(t1);
    21    331   c4 := cos(t1);
          332
          333   {
          334           find theta sub 5
          335   }
    21    336   s5 := ax*(t3*c4 - s1*s4) + ay*(t2*c4 + c1*s4) - az*c4*s23;
    21    337   c5 := ax*c1*s23 + ay*s1*s23 + az*c23;
 = 21    338   itheta[5] := a2srqq(s5, c5);
          339
          340   {
          341           find theta sub 6
          342   }
    21    343   t4 := -s1*c4 - t3*s4;
    21    344   t3 := c1*c4 - t2*s4;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    21    346   s6 := t4*nx + t3*ny + t2*nz;
    21    347   c6 := t4*sx + t3*sy + t2*sz;
  = 21    348   itheta[6] := a2srqq(s6, c6);
          349
          350
          351
    21    352   for i:=1 to 6 do
  = 21    353         itheta[i] := itheta[i] * f180_pi;
          354
    21    355   if puma_debug  or wrd(config[5]) = 1  then
    21    356         for i:=1 to 6 do begin
    22    357               writeln(sercom,'Joint ',i:1, ' has an angle of ',itheta[i]:8:3);
    21    358         end;
          359
    10    360   end;

 Symtab   360   Offset Length   Variable - INVERSE
                -    2    270   Return offset, Frame length
                -    2      2   I                                   :Integer
                -    6      4   PX                                  :Real
                -   18      4   AX                                  :Real
                -  126      4   R                                   :Real
                -  204      1   K                                   :Boolean
                -   22      4   AY                                  :Real
                -   26      4   AZ                                  :Real
                -   42      4   NX                                  :Real
                -   82      4   C1                                  :Real
                -   86      4   C2                                  :Real
                -   90      4   C3                                  :Real
                -   94      4   C4                                  :Real
                -   98      4   C5                                  :Real
                -  102      4   C6                                  :Real
                -  146      4   L3                                  :Real
                -  170      4   K1                                  :Real
                -  174      4   K2                                  :Real
                -  178      4   K3                                  :Real
                -  182      4   K4                                  :Real
                -  150      4   L4                                  :Real
                -   46      4   NY                                  :Real
                -   50      4   NZ                                  :Real
                -   10      4   PY                                  :Real
                -   14      4   PZ                                  :Real
                -   30      4   SX                                  :Real
                -   54      4   S1                                  :Real
                -   58      4   S2                                  :Real
                -   62      4   S3                                  :Real
                -   66      4   S4                                  :Real
                -   70      4   S5                                  :Real
                -   74      4   S6                                  :Real
                -   34      4   SY                                  :Real
                -   38      4   SZ                                  :Real
                -   78      4   S23                                 :Real
                -  106      4   C23                                 :Real
```

```
                - 186      4   T1                             :Real
                - 190      4   T2                             :Real
                - 194      4   T3                             :Real
                - 198      4   T4                             :Real
                - 200      1   ARM                            :Boolean
                - 122      4   RSQ                            :Real
                - 134      4   CAL                            :Real
                - 142      4   CBT                            :Real
                - 110      4   PXSQ                           :Real
                - 130      4   SAL                            :Real
                - 138      4   SBT                            :Real
                - 114      4   PYSQ                           :Real
                - 118      4   PZSQ                           :Real
                - 154      4   OMEGA                          :Real
                - 158      4   Z3AX                           :Real
                - 162      4   Z3AY                           :Real
                - 166      4   Z3AZ                           :Real
                - 202      1   ARM_BELOW                      :Boolean

 INVERSE
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          361   {$PAGE+}
   10     362   procedure homotran;
          363
          364   {
          365           PROCEDURE HOMOTRAN;
          366
          367           Purpose:        Finds the homogeneous transformation matrix
          368                           specifying the current position of the end-effector
          369                           of the robot from the XYZOAT description of the
          370                           robot location.
          371
          372           Calling convention:
          373                           homotran;
          374
          375           Global variables:
          376                           xyz     xyzoat configuration of robot
          377
          378                           tmatrix returns the homogeneous tranformation of the
          379                                   current position of the end effector of the
          380                                   robot arm.
          381
          382
          383   }
          384
   20     385   var
   20     386           o,                      !OAT angles in radians
   20     387           a,
   20     388           t,
   20     389           cosO,
   20     390           sinO,
   20     391           cosA,
   20     392           sinA,
   20     393           cosT,
   20     394           sinT: real;
          395
   20     396   begin
   21     397   o := xyz[4];
   21     398   a := xyz[5];
   21     399   t := xyz[6];
          400
   21     401   cosO := dcos(o);
   21     402   cosA := dcos(a);
   21     403   cosT := dcos(t);
   21     404   sinO := dsin(o);
   21     405   sinA := dsin(a);
   21     406   sinT := dsin(t);
          407
 = 21     408   tmatrix[1,1] := (cosO * sinT) - (sinO * sinA * cosT);
 = 21     409   tmatrix[1,2] := (cosO * sinA * cosT) + (sinO * sinT);
 = 21     410   tmatrix[1,3] := - (cosA * cosT);
 = 21     411   tmatrix[2,1] := (sinO * sinA * sinT) + (cosO * cosT);
 = 21     412   tmatrix[2,2] := - (cosO * sinA * sinT) + (sinO * cosT);
 = 21     413   tmatrix[2,3] := cosA * sinT;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
  = 21    414   tmatrix[3,1] := sinO * cosA;
  = 21    415   tmatrix[3,2] := - (cosO * cosA);
  = 21    416   tmatrix[3,3] := - sinA;
  = 21    417   tmatrix[4,1] := xyz[1];
  = 21    418   tmatrix[4,2] := xyz[2];
  = 21    419   tmatrix[4,3] := xyz[3];
          420
    21    421   if homo_debug  or wrd(config[5]) = 1
    21    422      then begin
    22    423           writeln(sercom,'tmatrix:');
    22    424           pr_mat(tmatrix);
    21    425           end;
          426
    10    427   end;

 Symtab   427   Offset Length   Variable - HOMOTRAN
                -    2     42   Return offset, Frame length
                -    4      4   O                                  :Real
                -    8      4   A                                  :Real
                -   12      4   T                                  :Real
                -   16      4   COSO                               :Real
                -   24      4   COSA                               :Real
                -   20      4   SINO                               :Real
                -   28      4   SINA                               :Real
                -   32      4   COST                               :Real
                -   36      4   SINT                               :Real

 HOMOTRAN
```

```
          {$PAGE+}
   10     function joint_check;

          {
                  FUNCTION JOINT_CHECK: boolean;

                  Purpose:          Determines if all of the joint angles given are
                                    within the PUMA's tolerable limits. It returns true
                                    if everything is okay. It returns false if any of the
                                    angles are bad.   When no errrors are found the
                                    position of arm is saved.

                  Calling convention:
                                    good := joint_check;

                  Global Variables:
                                    xyz     Contains the proposed xyzoat position of
                                            the puma arm.
                                    tmatrix The homogenous transformation matrix
                                            describing the proposed arm position.
                                    itheta  Contains the inverse solution of the arm
                                            (ie.  each of the six joint angles).
                                    rob_xyz Returns last valid xyzoat (robot coords)
                                            position of the puma arm.
                                    theta   Returns last valid inverse solutions of the
                                            arm (ie.  each of the six joint angles).

          }

   20     var
   20             i,                        !incrementor
   20             error: integer;           !error flag

   20     begin

   21     error := 0;

   21     for i :=1  to 6 do
   21             begin

                  {Place into range of -360.0 to 360.0}
   22             if (itheta[i] > max_degree[i])
   22                then while itheta[i] > max_degree[i] do
 = 22                     itheta[i]  := itheta[i] - 360.0

   22                else while (itheta[i] < min_degree[i]) do
 = 22                     itheta[i]  := itheta[i] + 360.0;

                  {outside legal joint limits}
   22             if ( (itheta[i]<min_degree[i]) or ( itheta[i]>max_degree[i]) )
   22                then begin
   23                     if itheta[i] < 0        !choose closest limit
```

```
 JG IC  Line#  Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    24   481                      then begin;
    24   482                           if abs(itheta[i]-min_degree[i]) > abs(itheta[i]+360.0-max_degree[i])
  = 24   483                              then itheta[i] := itheta[i] + 360.0;
    24   484                           end
         485
    23   486                      else if abs(itheta[i]-360.0-min_degree[i]) < abs(itheta[i]-max_degree[i])
  = 23   487                                      then itheta[i] := itheta[i] - 360.0;
         488
    23   489                   writeln('Joint ',i:1,' at ',itheta[i]:8:3,
    23   490                           ' degrees out of ', min_degree[i]:8:3,
    23   491                           ' to ',max_degree[i]:8:3,' range');
         492
    23   493                   error := 1;
         494
    22   495                   end;
         496
    22   497           if itheta[i] > 180.0
  = 22   498              then itheta[i] := itheta[i] - 360.0
    22   499              else if itheta[i] < -180.0
  = 22   500                      then itheta[i] := itheta[i] + 360.0;
         501
    21   502           end;
         503
    21   504   error := error + config[4];
  = 21   505   config[4] := error;
         506
    21   507   if error = 0
    21   508      then begin                           !if no errors
  = 22   509           rob_xyz[1] := tmatrix[4,1];    !robot coords
  = 22   510           rob_xyz[2] := tmatrix[4,2];
  = 22   511           rob_xyz[3] := tmatrix[4,3];
  = 22   512           rob_xyz[4] := xyz[4];
  = 22   513           rob_xyz[5] := xyz[5];
  = 22   514           rob_xyz[6] := xyz[6];
    22   515           for i := 1 to 6 do              !save valid joint angles
  = 22   516                   theta[i] := itheta[i];
    21   517           end;
         518
  = 21   519   joint_check := (error = 0);
         520
 *  21   521   return;
    10   522   end;

Symtab   522   Offset Length   Variable - JOINT_CHECK
               -    2      8   Return offset, Frame length
               -    2      1   (function return)               :Boolean
               -    4      2   I                               :Integer
               -    6      2   ERROR                           :Integer
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          523   {$PAGE+}
   10     524   procedure robot_config;
          525
          526   {
          527           PROCEDURE ROBOT_CONFIG;
          528
          529           Purpose:        Displays the configuration of current attempted robot
          530                           location.  This routines expects that the current i
          531                           valid joint angle solution is in the array ITHETA.
          532
          533
          534   }
          535
   20     536   var
   20     537           T2,                     !Transformations from links back to robot ref
   20     538           T3,
   20     539           T4,
   20     540           T5,
   20     541           T6: matrix;
   20     542           darm,
   20     543           delbow,
   20     544           dwrist,
   20     545           t23: real;
          546
   20     547   begin   {ROBOT_CONFIG}
          548
          549   {
          550           First compute the necessary transformations
          551   }
   21     552   tmats(T2,T3,T4,T5,T6);
          553
          554
   21     555   t23 := itheta[2]+itheta[3];
   21     556   darm := -D4 * dsin(t23) - A3 * dcos(t23) - A2 * dcos(itheta[2]);
   21     557   if darm < -epsilon
   21     558      then write('Left, ')
   21     559      else write('Right, ');
          560
   21     561   delbow := darm * (-A3*dsin(itheta[3]) + D4*dcos(itheta[3]));
   21     562   if (delbow < -epsilon)
   21     563      then write('Below, ')
   21     564      else write('Above, ');
          565
   21     566   dwrist := t4[3,1]*t6[2,1] + t4[3,2]*t6[2,2] + t4[3,3]*t6[2,3];  !S * Z4
   21     567   if abs(dwrist) < epsilon
   21     568      then dwrist := t4[3,1]*t6[1,1] + t4[3,2]*t6[1,2] + t4[3,3]*t6[1,3];  !N * Z4
   21     569   if dwrist < -epsilon
   21     570      then write('Up, ')
   21     571      else write('Down, ');
   21     572   if itheta[5] < -epsilon
   21     573      then writeln('Flip ')
   21     574      else writeln('Noflip ');
          575
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          576
   10     577   end;

 Symtab   577   Offset Length   Variable - ROBOT_CONFIG
                -    2    350   Return offset, Frame length
                -   64     64   T2                                :Array
                -  128     64   T3                                :Array
                -  192     64   T4                                :Array
                -  256     64   T5                                :Array
                -  320     64   T6                                :Array
                -  324      4   DARM                              :Real
                -  336      4   T23                               :Real
                -  328      4   DELBOW                            :Real
                -  332      4   DWRIST                            :Real

 ROBOT_CONFIG
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          578   {$PAGE+}
    10    579   procedure tmats;
          580
          581   {
          582           PROCEDURE TMATS(var T2, T3, T4, T5, T6: matrix);
          583
          584           Purpose:        Finds the transformations from link i coordinate
          585                           system back to robot reference (base) coordinate
          586                           system. Note: uses the transformation notation
          587                                   x' = x A   (graphics)
          588                           instead of usual
          589                                   x' = A x   (robotics).
          590                           The result is that matrix A (graphics) is transpose
          591                           of matrix A (robotics).
          592
          593   }
          594
    20    595   var
    20    596           s,              !temporary holds sine and cosine values
    20    597           c: real;
          598
    20    599   begin
          600
    21    601   s := dsin(itheta[1]);   !link 1 to link 0
    21    602   c := dcos(itheta[1]);
  = 21    603   A10[1,1] := c;
  = 21    604   A10[1,2] := s;
  = 21    605   A10[3,1] := -s;
  = 21    606   A10[3,2] := c;
          607
    21    608   s := dsin(itheta[2]);   !link 2 to link 1
    21    609   c := dcos(itheta[2]);
  = 21    610   A21[1,1] := c;
  = 21    611   A21[1,2] := s;
  = 21    612   A21[2,1] := -s;
  = 21    613   A21[2,2] := c;
  = 21    614   A21[4,1] := A2 * c;
  = 21    615   A21[4,2] := A2 * s;
          616
    21    617   s := dsin(itheta[3]);   !link 3 to link 2
    21    618   c := dcos(itheta[3]);
  = 21    619   A32[1,1] := c;
  = 21    620   A32[1,2] := s;
  = 21    621   A32[3,1] := s;
  = 21    622   A32[3,2] := -c;
  = 21    623   A32[4,1] := A3 * c;
  = 21    624   A32[4,2] := A3 * s;
          625
    21    626   s := dsin(itheta[4]);   !link 4 to link 3
    21    627   c := dcos(itheta[4]);
  = 21    628   A43[1,1] := c;
  = 21    629   A43[1,2] := s;
  = 21    630   A43[3,1] := -s;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
  = 21    631   A43[3,2] := c;
          632
    21    633   s := dsin(itheta[5]);   !link 5 to link 4
    21    634   c := dcos(itheta[5]);
  = 21    635   A54[1,1] := c;
  = 21    636   A54[1,2] := s;
  = 21    637   A54[3,1] := s;
  = 21    638   A54[3,2] := -c;
          639
    21    640   s := dsin(itheta[6]);   !link 6 to 5
    21    641   c := dcos(itheta[6]);
  = 21    642   A65[1,1] := c;
  = 21    643   A65[1,2] := s;
  = 21    644   A65[2,1] := -s;
  = 21    645   A65[2,2] := c;
          646
          647   {
          648           Finally, find transformations from link 1 back to world.  Note: T0
          649           is transformation from "robot world" to "real world".
          650   }
    21    651   T2 := mat_mult(A21,A10);
    21    652   T3 := mat_mult(A32,T2);
    21    653   T4 := mat_mult(A43,T3);
    21    654   T5 := mat_mult(A54,T4);
    21    655   T6 := mat_mult(A65,T5);
          656
          657   {$IF tmats_debug $THEN}
          658           writeln(sercom,'A21');
          659           pr_mat(A21);
          660           writeln(sercom,'A32');
          661           pr_mat(A32);
          662           writeln(sercom,'A43');
          663           pr_mat(A43);
          664           writeln(sercom,'A54');
          665           pr_mat(A54);
          666           writeln(sercom,'A65');
          667           pr_mat(A65);
          668           writeln(sercom,'T0');
          669           pr_mat(T0);
          670           writeln(sercom,'T1 or A10');
          671           pr_mat(A10);
          672           writeln(sercom,'T2');
          673           pr_mat(T2);
          674           writeln(sercom,'T3');
          675           pr_mat(T3);
          676           writeln(sercom,'T4');
          677           pr_mat(T4);
          678           writeln(sercom,'T5');
          679           pr_mat(T5);
          680           writeln(sercom,'T6');
          681           pr_mat(T6);
          682   {$END}
```

```
JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   10    684   end;

Symtab   684   Offset Length   Variable - TMATS
               -   12     74   Return offset, Frame length
               +   14      2   T2                                :Array   VarP
               -    4      4   S                                 :Real
               -    8      4   C                                 :Real
               +   12      2   T3                                :Array   VarP
               +   10      2   T4                                :Array   VarP
               +    8      2   T5                                :Array   VarP
               +    6      2   T6                                :Array   VarP


TMATS
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
          685   {$PAGE+}
   10     686   procedure init_robot;
          687
          688   {
          689           PROCEDURE INIT_ROBOT;
          690
          691           Purpose:        Initializes most of the robot specific data
          692                           strucutures.  This procedure loads the data for the
          693                           robot figure from the file called "PUMA.DAT" into the
          694                           "robot" object data structure. However, it should be
          695                           noted that this object is really a collection of
          696                           seven objects, one for each link.
          697
          698   }
          699
   20     700   var
   20     701           invertible: boolean;
          702
          703
   20     704   begin   {ROBOT_INIT}
          705
 = 21     706   tmatrix := identity_matrix;             !robot to world transform matrix
          707
          708   {
          709           Specify the robot to world transformation.  By the correct rotation
          710           transformation, table mounts, ceiling mounts and side mounts can be
          711           handled as well as translation  between the coordinate systems.
          712   }
 = 21     713   TO  := identity_matrix;                 !Robot base to shoulder
 = 21     714   TO[4,3] := 669.4;
 = 21     715   TOI := invert_matrix(TO,invertible);
          716
 = 21     717   H := identity_matrix;                   !Robot to tool transformation (null)
 = 21     718   HI := identity_matrix;
          719
          720   {
          721           Set the joint limits
          722   }
 = 21     723   max_degree[1] := 160.0;
 = 21     724   max_degree[2] := 45.0;
 = 21     725   max_degree[3] := 225.0;
 = 21     726   max_degree[4] := 170.0;
 = 21     727   max_degree[5] := 100.0;
 = 21     728   max_degree[6] := 266.0;
          729
 = 21     730   min_degree[1] := -160.0;
 = 21     731   min_degree[2] := -225.0;
 = 21     732   min_degree[3] :=  -45.0;
 = 21     733   min_degree[4] := -110.0;
 = 21     734   min_degree[5] := -100.0;
 = 21     735   min_degree[6] := -266.0;
          736
          737
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
  = 21    738   A10 := identity_matrix;
  = 21    739   A10[2,2] := 0.0;
  = 21    740   A10[2,3] := -1.0;
  = 21    741   A10[3,3] := 0.0;
          742
  = 21    743   A21 := identity_matrix;
  = 21    744   A21[4,3] := D2;
          745
  = 21    746   A32 := A10;                     !Most similar to A10
  = 21    747   A32[2,3] := 1.0;
          748
  = 21    749   A43 := A10;                     !Most similar to A10
  = 21    750   A43[4,3] := D4;
          751
  = 21    752   A54 := A32;                     !Most similar to A32
          753
  = 21    754   A65 := A21;                     !Most similar to A21
  = 21    755   A65[4,3] := D6;
          756
          757
    10    758   end;    {ROBOT_INIT}

 Symtab   758   Offset Length   Variable - INIT_ROBOT
                -    2     68   Return offset, Frame length
                -    2      1   INVERTIBLE                       :Boolean
```

PUMA

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
         759    {$PAGE+}
    00   760    end.

 Symtab  760    Offset Length   Variable
                     0    570   Return offset, Frame length
                     4     24   XYZ                              :Array   Static Public
                     0     64   TO                               :Array   Static Extern
                     0     64   H                                :Array   Static Extern
                     0     64   HI                               :Array   Static Extern
                     0     64   TOI                              :Array   Static Extern
                   186     64   A10                              :Array   Static
                   250     64   A21                              :Array   Static
                   314     64   A32                              :Array   Static
                   378     64   A43                              :Array   Static
                   442     64   A54                              :Array   Static
                   506     64   A65                              :Array   Static
                     0     24   THETA                            :Array   Static Extern
                    92     24   ITHETA                           :Array   Static Public
                     0     14   CONFIG                           :Array   Static Extern
                     0    636   SERCOM                           :File    Static Extern
                    28     64   TMATRIX                          :Array   Static Public
                     0     24   ROB_XYZ                          :Array   Static Extern
                     0      8   VERSION                          :Array   Static Extern
                     0      2   FIRST_STR                        :Pointer Static Extern
                     0      1   MENU_FLAG                        :Boolean Static Extern
                   132     24   MAX_DEGREE                       :Array   Static
                   156     24   MIN_DEGREE                       :Array   Static
                     0      2   COORDS_TYPE                      :Integer Static Extern
                     0      2   MENU_CURSOR                      :Integer Static Extern
                   180      6   MODEL_VERTS                      :Pointer Static

                Errors  Warns  In Pass One
                     0      0
```

```
JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   00      1   {$linesize:132,$pagesize:60}
           2   {$title:'Global include file (global.inc)',$subtitle:' '}
           3   {$debug-,$list+}
           4
           5   {
           6   ---------------------------------------------------------------------
           7
           8                              University of Michigan
           9                              College of Engineering
          10                Center for Robotics and Integrated Manufacturing
          11                              Robot Systems Division
          12                           2510 East Engineering Building
          13                               Ann Arbor, MI 48109
          14                                 (313) 764-4343
          15
          16          Copyright 1984, The University of Michigan, All Rights Reserved
          17
          18
          19                          Written by:  Richard M. Jungclas
          20
          21   ---------------------------------------------------------------------
          22   }
          23
           0   {$include:'GLOBAL.INC'}
           1   {$LIST+}
           2   {$title:'Global Include file (global.inc)',$subtitle:'Last change: 4-3-84'}
           3
  00       4   interface;
           5   {
           6     GLOBAL.INC
           7          This file contains all declarations of types global to the system.
           8          It also includes variables and some matrix manipulation routines
           9          used throughout the system.
          10   }
          11
  00      12   unit globals(
  00      13                version,                    !system's version number
          14
  00      15                r_matrix,                   !Generic type two-dimensional arrays
  00      16                i_matrix,
  00      17                j_matrix,
  00      18                g_matrix,                   !general one dimensional matrix type
  00      19                matrix,                     !4 x 4 homogeneous matrix type
  00      20                c_matrix,                   !coordinate matrix type
  00      21                rmat_ptr,                   !two super array pointer types
  00      22                imat_ptr,
          23
  00      24                SUCCESS,                    !return code from serial I/O routines
  00      25                NO_SUCCESS,                 !return code from serial I/O routines
  00      26                SPECIAL,                    !return code meaning special key
          27
  00      28                world_type,                 !world cartesian coordinates type
  00      29                robot_type,                 !robot cartesian coordinates type
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   00      30                joint_type,                     !joint coordinates type
           31
   00      32                file1,                          !A text file type
   00      33                consol_input_lstr,              !lstring type for consol input.
   00      34                name_lstr,                      !lstring type for names
   00      35                file_lstr,                      !lstring standard file name type.
           36
   00      37                STRS,                           !Symbol Table Record structure
   00      38                STR_ptr,                        !pointer type to symbol table record
   00      39                first_STR,                      !Pointer to symbol list
           40
   00      41                menu_cursor,                    !Menu item cursor
   00      42                menu_flag,                      !Flag to redraw menu
           43
   00      44                config,                         !Robot configuration status
   00      45                coords_type,                    !Type of robot coordinates
   00      46                TO,                             !Robot base to world transformation
   00      47                TOI,                            !World to robot base  transformation
   00      48                H,                              !Robot to tool transformation
   00      49                HI,                             !Tool to robot transformation
   00      50                rob_xyz,                        !generalized coords (xyzoat) for robot
           51
           52                                                    !(Null tool to robot reference)
   00      53                theta,                          !Robot joint angles
           54
   00      55                find_STR,                       !function to find symbol table entry
   00      56                rotate_matrix,                  !function to generate rotation matrix
   00      57                dsin,                           !sine functiegrees)
   00      58                dcos,                           !cosine function (in degrees)
   00      59                mat_mult,                       !matrix manipulation routines
   00      60                identity_matrix,
   00      61                invert_matrix,
   00      62                get_xyzoat,
   00      63                getc,
   00      64                trim);
           65
   00      66          type
   10      67             consol_input_lstr     = lstring(80);
   10      68             name_lstr = lstring(12);
   10      69             file_lstr = lstring(63);
   10      70             file1     = text;
           71
   10      72             r_matrix = super array [1..*,1..*] of real;
   10      73             i_matrix = super array [1..*,1..*] of integer;
   10      74             g_matrix = super array [1..*] of real;
           75
   10      76             c_matrix = g_matrix(3);
   10      77             matrix   = r_matrix(4,4);
   10      78             j_matrix = g_matrix(6);
           79
   10      80             rmat_ptr = @r_matrix;
   10      81             imat_ptr = @i_matrix;
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
   10      83           STR_ptr  = @STRS;
           84
           85
           86   {
           87    Symbol Table Record
           88
           89           symname:        The symbolic name string
           90           data:           The coordinates of the location
           91           ctype:          The coordinates type
           92           used:           A boolean flag which (if true) indicates that the
           93                           location has been used by the programmer.
           94           next_STR:       A link to the next entry in the symbol table (or nil)
           95
           96   }
   10      97           STRS = record
   20      98                           symname:        name_lstr;
   20      99                           data:           j_matrix;
   20     100                           ctype:          integer;
   20     101                           used:           boolean;
   20     102                           next_STR:       STR_ptr;
   10     103                       end;
          104
          105
          106
   10     107   const
   10     108           SUCCESS    = 0;         {return code from serial I/O routines }
   10     109           NO_SUCCESS = 1;         {return code from serial I/O routines }
   10     110           SPECIAL    = 2;         {return code meaning special key }
   10     111           world_type = 0;         {type for world cartesian coordinates}
   10     112           robot_type = 1;         {type for robot cartesian coordinates}
   10     113           joint_type = 2;         {type for joint coordinates}
   10     114   var
   10     115           version:        string(7);      {system version number}
          116
   10     117           first_STR:      STR_ptr;
          118
   10     119           menu_flag:      boolean;
          120
   10     121           coords_type,            {type of robot coordinates
          122                                           0 = world cartesian
          123                                           1 = robot shoulder cartesian
          124                                           2 = joint }
          125
   10     126              menu_cursor: integer;
   10     127           config: array[0..6] of integer;
          128                                   {configuration of the robot arm
          129                                           [0]  -1=lefty,   1=righty
          130                                           [1]  -1=below,   1=above
          131                                           [2]  -1=up,      1=down
          132                                           [3]  -1=noflip,  1=flip
          133                                           [4]   1=error,   0=valid solution
          134                                           [5]   1=debug,   0=production
          135                                           [6]   (not used) }
```

```
 JG IC   Line#  Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           136
    10     137          rob_xyz,
    10     138             theta:       j_matrix;
    10     139          TO,
    10     140             TOI,
    10     141             H,
    10     142             HI:          matrix;
           143
           144
    20     145  function  find_STR(const name: lstring): STR_ptr [pure];
    20     146  function  rotate_matrix(const x,y,z: real): matrix [pure];
    20     147  function  dsin(const x: real): real [pure];
    20     148  function  dcos(const x: real): real [pure];
    20     149  function mat_mult(const m1,m2: matrix): matrix [pure];
    20     150  function identity_matrix: matrix [pure];
    20     151  function invert_matrix(source: matrix; var invertible: boolean): matrix [pure];
    20     152  function get_xyzoat(const m: matrix): j_matrix [pure];
    20     153  function getc(var letter,spec: char): boolean [pure];
    20     154  procedure trim(var s: lstring);
           155
    10     156  end;  {*****************************************************************}
           157  {$LIST+}
```

TRIM

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           25   {$title:'Debug Include file (debug.inc)',$subtitle:' ',$PAGE+}
            0   {$include:'DEBUG.INC'}
            1   {$LIST+}
            2   {$title:'Debug Include file (debug.inc)  Last Change: 3-2-84 rmj'}
            3
    00      4   interface;
            5
            6   {
            7      DEBUG
            8            This interface contains routines to aid in debugging PASCAL programs.
            9   }
           10
    00     11   unit debug(pr_mat, heap_space, writewt, breakpt);
           12
    10     13   uses globals;
           14
           15
    20     16   procedure pr_mat(const m1: matrix);
    10     17   procedure heap_space;
    20     18   procedure writewt(const line: consol_input_lstr);
    20     19   procedure breakpt(const line: consol_input_lstr);
           20
           21
    10     22   end;  {***************************************************************}
           23   {$LIST+}
```

BREAKPT

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           27   {$title:'Menu Function Include file (menu.inc)',$subtitle:' ',$PAGE+}
            0   {$include:'MENU.INC'}
            1   {$LIST+}
            2   {$title:'Menu Function include file (menu.inc)  Last change: 4-3-84 rmj '}
            3
    00      4   interface;
            5   {
            6      MENU
            7
            8           This interface contains routines to control the cursor as well as
            9           maintain the menu system.
           10   }
           11
    00     12   unit menu_functions(
    00     13           cls,
    00     14           clear_lower,
    00     15           clear_upper,
    00     16           command_prompt,
    00     17           data_prompt,
    00     18           display_menu,
    00     19           bmenu_item,
    00     20           gmenu_item,
    00     21           menu_item,
    00     22           display_status,
    00     23           highlight,
    00     24           position_cursor,
    00     25           print_border);
           26
    10     27   uses globals;
           28
    10     29   type
    10     30           pitem = lstring(30);
    10     31           pkey  = lstring(3);
           32
    10     33   procedure cls;
    10     34   procedure clear_lower;
    10     35   procedure clear_upper;
    10     36   procedure command_prompt;
    10     37   procedure data_prompt;
    20     38   procedure display_menu(var full_flag: boolean; const level:consol_input_lstr);
    20     39   procedure bmenu_item(const item: pitem);
    20     40   procedure gmenu_item(const key: pkey; const item: pitem);
    20     41   procedure menu_item(const item: pitem);
    10     42   procedure display_status;
    20     43   procedure highlight(line:consol_input_lstr);
    20     44   procedure position_cursor(row,column: integer);
    10     45   procedure print_border;
           46
    10     47   end;  {****************************************************************}
           48   {$LIST+}
```

PRINT BORDER

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           29   {$title:'PUMA Robot Routines Include file (puma.inc)',$subtitle:' ',$PAGE+}
            0   {$include:'PUMA.INC'}
            1   {$LIST+}
            2   {$title:'PUMA Robot Routines Include file (puma.inc)  Last change: 4-3-84'}
            3
   00       4   interface;
            5   {
            6      PUMA
            7            This interface contains PUMA routines
            8   }
            9
   00      10   unit puma(init_robot, tmats, inverse, homotran, joint_check, robot_config, xyz, tmatrix, itheta);
           11
   10      12   uses globals;
           13
           14
   10      15   var
   10      16            itheta,                 {Current joint angles solution for the robot}
   10      17            xyz: j_matrix;          {Generalized coords (xyzoat) for robot for
           18                                     the last request. Either tool to world or
           19                                     joint 6 to robot reference. Possibly invalid
           20                                     (out of reach) }
   10      21            tmatrix: matrix;        {Robot Transformation matrix for the last
           22                                     request. Either tool to world (usually) or
           23                                     joint 6 to robot reference, depending on
           24                                     context. Possibly invalid (out of reach). }
           25
   10      26   procedure init_robot;
   20      27   procedure tmats(var T2,T3,T4,T5,T6: matrix);
   10      28   procedure inverse;
   10      29   procedure homotran;
   20      30   function joint_check: boolean;
   10      31   procedure robot_config;
           32
           33
   10      34   end;  {************************************************************}
           35   {$LIST+}

DUMMY
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
           31   {$title:'Dummy routine to list include files',$subtitle:' ',$PAGE+}
   10      32   program dummy(input,output);
           33
   10      34   begin   {DUMMY procedure}
           35
   11      36           writeln('Dummy procedure');
           37
   00      38   end.

 Symtab    38   Offset Length   Variable
                     0     20   Return offset, Frame length


                Errors  Warns  In Pass One
                     0      0
```

```
 JG IC  Line#    Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
    24    553                    if config[5] = 1 and then coords_type = world_type
    24    554                       then writeln(sercom, 'World Cartesian coordinates')
    24    555                       else writeln(sercom, 'Robot Cartesain coordinates');
    24    556                    temp_coord_type := coords_type;
  = 24    557                    coords_type := temp_type;
    24    558                    if config[5] = 1
    24    559                       then writeln(sercom, 'Calling homotran');
    24    560                    homotran;
    24    561                    if config[5] = 1
    24    562                       then writeln(sercom, 'Calling inverse');
    24    563                    inverse;
  = 24    564                    coords_type := temp_coord_type;
    23    565                    end;
          566
    23    567                joint_type:
    23    568                    If config[5] = 1
    23    569                       then writeln(sercom, 'Joint coordinates');
          570
    22    571                end;
          572
    22    573            if joint_check
    22    574                then begin
  = 23    575                    move := true;
    23    576                    if config[6] = 1
  / 23    577                       then writeloc(sercom);
    23    578                    end
    23    579                else begin;
    23    580                    writeln('Robot can''t reach this location!');
    23    581                    leave := false;
  = 23    582                    move := false;
    22    583                    end;
          584
    21    585            end;
          586
    10    587    end;

Symtab    587    Offset Length   Variable - MOVE
                 -    4    140   Return offset, Frame length
                 -    2      1   (function return)                  :Boolean
                 +    6      2   TRACEFIL                           :File    VarP
                 -    4      1   COMMAND                            :Char
                 -    6      1   SPEC                               :Char
                 -   10      1   FULL                               :Boolean
                 -   14      4   X                                  :Real
                 -   26      4   O                                  :Real
                 -   30      4   A                                  :Real
                 -   40      2   I                                  :Integer
                 -   34      4   T                                  :Real
                 -   18      4   Y                                  :Real
                 -   22      4   Z                                  :Real
                 -   56     14   NAME                               :Array
                 -    8      1   LEAVE                              :Boolean
                 -   42      2   TEMP_STR                           :Pointer
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
                -    36       2   TEMP_TYPE                              :Integer
                -    38       2   TEMP_COORD_TYPE                        :Integer
                -   138      82   INPUT_LINE                             :Array

 MAIN_ROUTINES
```

```
 JG IC  Line#   Microsoft MS-Pascal Compiler, MS-DOS 8086 Version 3.11, 05/83
         588    {$PAGE+}
   00    589    end.

 Symtab   589   Offset Length   Variable
                     0     24   Return offset, Frame length
                     0      8   VERSION                          :Array   Static Extern
                     0     64   TO                               :Array   Static Extern
                     0     64   H                                :Array   Static Extern
                     0     64   HI                               :Array   Static Extern
                     0     64   TOI                              :Array   Static Extern
                     0     24   XYZ                              :Array   Static Extern
                     0     14   CONFIG                           :Array   Static Extern
                     0     24   THETA                            :Array   Static Extern
                     0     24   ITHETA                           :Array   Static Extern
                     0     24   ROB_XYZ                          :Array   Static Extern
                     0    636   SERCOM                           :File    Static Extern
                     0     64   TMATRIX                          :Array   Static Extern
                     0      2   FIRST_STR                        :Pointer Static Extern
                     0      1   MENU_FLAG                        :Boolean Static Extern
                     0      2   MENU_CURSOR                      :Integer Static Extern
                     0      2   COORDS_TYPE                      :Integer Static Extern

                Errors  Warns  In Pass One
                     0      0
```