AN INTELLIGENT ROBOT CONTROL SYSTEM FOR PHYSIOTHERAPIC APPLICATIONS

by

Jun Yan, B.Sc., M.Sc.

This thesis is submitted as the fulfilment of the requirement for the award of the Degree of Doctor of Philosophy (Ph.D) by research to:

DUBLIN CITY UNIVERSITY

Sponsoring Establishment:

Dublin City University
School of Mechanical & Manufacturing Engineering
Glasnevin, Dublin 9
Ireland

AUGUST 1991

To LinLin and my family

DECLARATION

I hereby declare that all the work reported in this thesis were carried out by me at Dublin City University during the period of February 1989 to August 1991.

To the best of my knowledge, the results presented in this thesis originated from the present study, except where references have been made. No part of this thesis has been submitted for a degree at any other institution.

Signature of Candidate

J. Yan

JUN YAN

ACKNOWLEDGMENTS

The author wishes to express his sincere gratitude to Dr. M. A. El Baradie and Professor M. S. J. Hashmi for their supervision and for their kind guidance, helpful comments, valuable assistance and encouragement at all stages of this research.

The author is indebted to EOLAS, the Irish Science and Technology Agency, for their invitation of the author as a visiting researcher in EOLAS and for the provision of the grant.

The author is also indebted to Mr. J. J. Murhpy, head of the Manufacturing Consultancy Services, EOLAS, and Mr. Michael Fitzgibbon, Director of the Science and Technology Evaluation Unit, EOLAS, for their kind help and encouragement.

Thanks are also due to Mr. T. Walsh, Mr. J. Tracey, and Mr. I. Hooper, of the workshop staff at the School of Mechanical & Manufacturing Engineering, for their great work in manufacturing the designed robotic hands. And to G. Anderson for his valuable assistance.

Thanks are also extended to all the technical staff of the School of the Electronic Engineering for providing the necessary equipment during the manufacturing process of the designed PCB.

The author would like to express his sincere gratitute to Ms. L. Lawlor, the secretary of the School of Mechanical and Manufacturing Engineering, for typing all the papers which the author published during this research.

AN INTELLIGENT ROBOT CONTROL SYSTEM FOR PHYSIOTHERAPIC APPLICATIONS

Jun Yan, B.Sc., M.Sc.

ABSTRACT

An intelligent robot control system for physiotherapic applications has been developed. The intelligent robot control system consists of a specially designed robotic hand with built-in sensors, an interfacing module between the robot system and the computer, an intelligent path planning module and a fuzzy logic based intelligent control module.

The robotic hand with the integrated palm and two fingers has been used to perform the padding and kneading opeartions. The sensory information of the robotic hand have been used in the intelligent control process.

The intelligent path planning and control modules have been constructed with the knowledge bases (KBS) and the fuzzy logic based inference mechanism, which are able to deal with uncertainties by manipulating the fuzzy terms.

Thus, with the fuzzy/linguistic input terms, the required parameters can be generated for the path planning module. The massaging path can be planned by using the KBS in the intelligent path planning module.

While the task execution is monitored by the intelligent control module. The intelligent control module allows error-correction strategies to be formulated.
The required corrections can be carried out by using the on-line KBS and fuzzy inference mechanism in the intelligent control module.

Experimental results are presented, which show the feasibility and the effectiveness of the designed intelligent control system.

Contents　　　　Page

Table No.		Page

INTRODUCTION

Robots have found wide applications in manufacturing industry, remote exploration, etc. However, the developments of the Artificial Intelligence (AI), the sensing system and the dexterous hands for robots have remained as the key issues in the development of the intelligent generation robots.

Physiotherapic operations such as massaging the human body (arm, neck, back, etc.) in the health care process are monotonous and tedious tasks. And they are also time consuming and could be best carried out by a robot. However, massaging the human body is a difficult task and is usually carried out by highly skilled professionals.

A human massaging process consists of two basic actions: kneading and padding. Kneading is a process of applying a series of appropriate forces using the dexterous fingertips onto the muscles of the human body. While padding is a process of applying a series of appropriate forces using the palm onto the muscles of the human body.

The massaging process carried out by a skilled professional is an intelligent process of path planning and on-line path modification based on the human observations, which are usually in an imprecision/fuzzy form. This process cannot be achieved without using the knowledge acquired by the professional and the abilities to deal with the uncertainties which naturally exist in a massaging process.

To carry out the massaging process effectively, the physiotherapic robot system must be constructed with the knowledge bases (KBS) and the fuzzy inference mechanism to cope with any uncertainties. Therefore, the robot system with the AI is able to take the right actions for the given part, to apply appropriate force onto the part being massaged, and to make the necessary adjustments whenever required during a massaging process.

Aimed at developing a dexterous robotic hand and an AI control system for the physiotherapic robots, the objectives of this research project can be outlined as follows:

* Establish an experimental robotic massaging system with AI control modules

* Design a robotic hand with position/force sensing abilities to perform the massaging manipulations

* Design the digital controllers for the position servo control loop of the robotic hand

* Establish the mathematical model for the massaging path design and planning

* Incorporate the expertise knowledge bases of the massaging process into the path planning and parameter generating module

* Incorporate the expertise knowledge into the on-line fuzzy control rule bases and data bases

* Organize the task executions

* Perform the on-line error-corrections

A robotic hand with an integrated palm and two fingers has been developed and used to perform the padding and kneading operations. And the sensory information of the robotic hand have been used in the intelligent control process.

In a robotic massaging process, uncertainties and errors may occur due to wrongly specified part location, part deviations from its specified position and incorrectly planned path.

The intelligent path planning and control modules have been constructed with the KBS and the fuzzy logic inference mechanism, which are able to deal with uncertainties and errors by manipulating the fuzzy terms.

Thus, with the fuzzy/linguistic input terms, the required parameters can be generated for the path planning module. And the massaging path can be planned by using the KBS in the path planning module.

The task execution is monitored by the intelligent control module. The intelligent control module allows error-correction strategies to be formulated. The required corrections can be carried out by using the on-line KBS and fuzzy inference mechanism in the intelligent control module.

In this thesis, the literature survey is conducted in chapter 1, which is mainly concerned with the development of the robotic end-effectors, sensing, compliance control and AI control in robotics.

The robotic massaging process is studied in chapter 2. And a robotic massaging system with AI is also proposed.

The configuration of the robotic massaging system is described in chapter 3, which includes the robot arm and the robotic hand with their controllers, the interfaces and the computer. The mechanical design of the robotic hand is also given out.

The development of the end-effector's controller is presented in chapter 4 Where the sensors and amplifier, the DC motor drive circuit design, the position and force servo loop control over the robotic hand have been described.

The direct and inverse kinematics of the robot arm are analysed in chapter 5. And the robotic hand coordinates are presented. Also the massaging path design is described.

The intelligent control system is presented in chapter 6 in which the off-line and on-line KBS are described. The parameter generating and path planning using off-line KBS are established and the task execution module is introduced. The on-line error-corrections based on fuzzy logic are studied. Also the experimental results are presented.

Chapter One

Literature Survey

1-1 Introduction

A robot is defined as a reprogrammable multifunctional manipulator designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks.[1]

Robots have found wide applications in industrial flexible manufacturing, remote exploration, and daily life service.

The new generation of robots are characterized by their abilities of:

* intelligent sensing
* intelligent decision-making
* dealing with uncertainties
* intelligent path-planning & error-corrections
* performing delicate tasks using dexterous hand

One may always find an application area to verify or to develop the new generation robots. The physiotherapic robot is one of them which requires Artificial Intelligence (AI), sensing, and dexterous robotic hand.

In this chapter, the literature survey on robotic hand design, robotic sensing, compliance control, AI system and fuzzy logic control is carried out.

The development of the robotic hands with multiple fingers is presented in section 1-2. The force/tactile sensing in robotics is reviewed in section 1-3. The compliance control stategies and compliance devices used in robotics are outlined in section 1-4. The literature of the development of AI including fuzzy logic control are surveyed in section 1-5.

1-2 Robotic end-effectors

Robotic end-effectors perform all the tasks instructed by robots. The performance of the end-effectors decides what kind of jobs the robot can do.

Placed at the end of the robot arm and interacting with external objects, end-effectors are often equipped with their own sensors and actuators.

There are two types of end-effectors:

* Special-purpose end-effectors which are designed to adapt to the specific tasks, such as welding, grinding, etc.
* General-purpose end-effectors which are designed with the dexterity and versatility.

The general-purpose end-effectors are featured by the multifingers and the built-in sensors. While the special-purpose end-effectors are characterized with the simple mechanical configurations. The development of the special -purpose end-effectors is at the mature stage except the sensing ability.

The general-purpose end-effectors have long been dreamt of. And in the course of development of robots, many articulated robot hands have been developed to achieve high levels of dexterity and versatility in imitating the human hand with its 32 DOF and thousands of positional, force and temperature sensors. The development of dexterous robot hands with the built-in sensors holds considerable promise for advanced robot capabilities.

Depending on the applications, the dexterous robot hand is usually designed with the following functions for the industrial applications:

* The ability to perform advanced manipulation, such as grasping arbitrary objects and tools in the robot workspace.
* The ability to provide the required information to infer the properties of the environment.

Several research projects have been carried out to develop reasonable and practical mechanical design configuration for the robot hands with built-in sensors. The major developments on the multifingered robotic hand are listed as follows.

OKADA HAND (1977)

Research into multifingered robotic hands took its first major step forward with the development of a 3-fingered hand [2] by OKADA in 1977. The hand consists of three fingers: an equivalent thumb with 3 joints, an equivalent index finger with 4 joints and an equivalent middle finger with 4 joints.

Thus, the hand has 11 DOF in total. There are 11 tendons to the fingers, one per joint. The tendons are wires runing through a flexible but incompressible sheath off the arm to the DC motors with gearboxes. The joint torque is generated by controlling the DC motors. And the tendon length is measured by using the potentiometers mounted at the motors.

The OKADA hand was designed to handle objects in industrial applications. The problems with the OKADA hand are inadequate sensory feedback [3-4]. There are no force sensors on the tendons to measure real tendon force. The force inferred from the motor torque is grossly in error because of friction in the tendon sheaths. No account was taken of tendon stretch, and the finger positions are not precise.

HANAFUSA HAND (1977)

A three fingered planar gripper was developed for industrial assembly applications by HANAFUSA and ASADA [5-6] in 1977. Each finger is a single DOF level driven by a stepping motor. The frictionless rollers are mounted on the fingertips to prevent any tangential contact force.

The only sensing feedback was the position from the motor. Besides the mechanical design, a gripping theory in the grasping plane was also developed [5]. By defining a potential function arising from the fingertip forces, the stable grasps can be determined by using the knowledge of the object shapes.

Stanford/JPL HAND (1982)

The Stanford/JPL hand was designed for objects handling by SALISBURY [7-10] by following the design philosophy of achieving an arbitrary grasping ability with the fewest fingers, tendons, and sensors.

The hand has three fingers with three joints each, arranged as two fingers and an opposing thumb. There are 4 tendons per finger, following the (n+1) rule that n tendons are needed for n DOF plus extra one since tendons cannot push. The tendons are teflon-coated cables, running over pulleys at the joints, and traveling through flexible but incompressible sheaths.

The position and force sensors have been embedded into the design of the hand. The position sensors at the DC motors are used to measure tendon length. Where the force sensors on the tendon are used to measure the tendon force and to correct the tendon stretch.

Pennsylvania Articulated Mechanical HAND (PAMH) (1983)

Used in industrial assembly operations, PAMH [11] has two forefingers and an opposing thumb. Each finger has two links and two joints with a parallel action. A linear actuator is used to drive the fingers. And the passive springs provide the restoring joint torque. Also optical encoders are used to measure the rotations of the motor shaft.

CAPORALI HAND (1984)

A five fingered hand with four forefingers and a thumb has been designed for industrial applications by Caporali and Shahinpoor [12]. Each finger has three links and 3 DOF. Cables over pulleys are driven by stepping motors. The passive extension of each joint is achieved by using springs. No contact sensing is provided.

Utah/MIT HAND (1985)

The Utah/MIT hand [13-18] consists of four fingers, arranged as a thumb opposing three fingers. Each finger has three links and four joints. Each joint is actuated by 2 antagonistic ploymeric tendon tapes, which run over pulleys to a remote actuator package.

There are 32 tendon tension sensors and 16 joint position encoders. Each joint angle is measured directly by miniature position encoder. Each tendon force is detected at the knuckles by using strain gauges on idler pulleys. Incorporating tactile sensors on the fingertips are being investigated [22].

Characterized with large numbers of actuators and sensors, the Utah/MIT dexterous hand requires high servo rates. Thus powerful and flexible computer architectures are needed to carry out the computation and control. Five M68000 microprocessors on a multibus, connected to a VAX 11/750 through a parallel DMA interface, are used to control the Utah/MIT hand [16,18].

The hand control system tends to become more and more complicated and bulky to improve the computation speed [17]. Thus it is not easy for the users to incorporate this hand into their robot system.

Hitachi HAND (1985)

A three fingered, tension-driven hand was developed by Hitachi Ltd. [19,20]. A thumb is arranged to oppose two fore fingers. Every finger has three segments and four DOF. Each joint is driven by a novel Shape Memory Alloy (SMA) actuator through tendons. The restoring torque is provided by springs.

The Hitachi hand can lift 2 Kg weight. The maximum joint motion is 90 degree per second. The SMA actuator is compact and light. But the response time due to slow temperature changes does not meet the requirements of most current industrial applications.

YAMAFUJI HAND (1988)

A three fingered hand was developed for objects handling by YAMAFUJI and MAEDA [21]. The hand consists of a palm, a thumb and two fore fingers. The thumb is constructed with 2 joints. While each forefinger has three joints. The bending motion of each finger is realized by using a steel wire driven by a DC motor mounted on the palm. The rotation of the thumb is carried out by using the same driven system. Only one steel wire driven by a DC motor is used to rotate the forefingers.

Eight rotary encoders are mounted on the fingers to detect the motion of the joints, and 22 touch sensors are used to detect the grasped objects. One master microprocessor and three slave microprocessors are used to contruct the hand control system.

Table 1.1 gives a summary of the main features for the different robot hands.

Table 1.1 Robotic Hands Summary

Robotic Hands	Mechanism	Sensors
OKADA (1977)	3 fingers, 11 DOF	position
HANAFUSA (1977)	3 fingers, 3 DOF	position
Stanford/JPL (1982)	3 fingers, 9 DOF	position/force
PAMH (1983)	3 fingers, 6 DOF	position
CAPORALI (1984)	5 fingers, 15 DOF	position
Utah/MIT (1985)	4 fingers, 15 DOF	position/force
Hitachi (1985)	3 fingers, 12 DOF	position
YAMAFUJI (1988)	3 fingers, 11 DOF	position/touch

1-3 Sensing in Robotics

The use of sensors plays an important role in extending the capability of robots to deal with unknown environment and unexpected events. In general, sensors in robotics are mainly used for the following objectives:

* obtaining the on-line information about the worksp-ace and workpieces [23-27]
* detecting the interactions between robots and en-vironment [27-34]
* guiding the motion of the robots based on the sen-sory information [35-40]
* enhancing the performance of the robots

Sensors used in robotics can be classified into internal sensors and external sensors. The internal sensors, which are usually embedded in the drive systems of the robot to measure position and speed of robot joints and linkages, include encoders, potentiometers, and tachometers. This group of sensors has been well developed and widely used in robotics.

Opposed to the internal position sensors, the external sensors, which are subdivided into contact sensors and non-contact sensors, are under intensive development. Contact sensors detect force/torque and touch/pressure when physically contacting an object. While noncontact sensors sense images, range, and the presence of objects without making any physical contact.

<u>Non-contact sensors</u> are used mainly for:

* identifying and locating objects in an environment [23-25]
* visually inspecting the objects [26]
* guiding the manipulation of the robots [36,38,39]

However, using visual sensing, it is only possible to discover mechanical properties of the objects by deducing them from optical properties. Furthermore, the interaction properties, such as force and torque, between the robot and the environment, can not be detected by using visual sensing. Hence, the contact type sensors such as force and tactile sensors are required to provide the interaction information.

Contact sensors include:

* touch sensors
* force/torque sensors
* tactile/pressure sensors

Tactile sensors, which gives information only about whether or not contact has occurred, are widely used in robotics due to their simple configuration, and low cost. The touch sensors have been used to prevent damaging collisions with obstacles [41-43].

Force/torque sensors are used primarily for measuring the reaction forces developed due to the interactions between the robot and its environment. The measured forces can be used to guide the motion of the robots.

Different types of sensing materials have been used to construct the force/torque sensing devices, which include: [28,34]

* metal strain gauges
* semiconductor strain gauges
* conductive elastomers
* piezoelectric ceramics, etc.

According to the placement relative to the robotic manipulator, the force/torque sensing devices can be further subdivided into: [32,34]

* force sensing platform

* joint torque sensing devices
* force sensing wrist and fingers

The force sensing platform has been used by WATSON and DRAKE [33] in 1975 to carry out assembly work. The horizontal and vertical forces generated due to the interaction between robot and environment can be measured by using the platform, on which the object being manipulated is placed.

The joint torque sensing devices are usually mounted on the joints of the manipulator. Joint torque sensing has the added advantage of not only detecting forces and torques applied at robotic hand, but also those applied at other points on the manipulator. This is very usefule in providing feedback information if, for instance, some portion of the manipulator were to unexpectedly encounter an obstacle [32,44].

The disadvantages in using the joint torque sensors are:

* time consuming to convert joint torques to the equivalent forces and moment at the robotic hand frame
* uncertainties in measuring and controlling robotic hand forces

One of the solutions to reduce the uncertainties in measuring and controlling hand forces is to mount the force sensing devices either close to the robotic hand or on the robotic fingers, where they are subjected to a minimum of interference from the configuration of the manipulator.

Based on strain gauges and elastically flexing beams, the wrist sensors have been developed by many researchers [45-52] since 1973.

Among the wrist sensors, two mechanical configurations have been adopted to contruct the sensors:

* a hollow cylinder with 8 beams [45-50]
* a metallic frame with cross cantilever beams [51,52]

Strain gauges are placed at the high strain points. And the wrist sensor can measure all forces and torques in the Cartesian coordinates. Several industrial applications of such sensors have been reported, e.g., in the fields of deburring [53] and grinding [54].

However, some kind of compliance in the wrist is required for the delicate manipulations, such as assembly and massaging. The multifingered hands with the force or tactile sensing abilities may provide the required compliance.

The force sensors in robotic fingers are usually mounted on:

* fingertips to measure the normal or tangental forces on the fingertips [22,54,56]
* the finger joints to measure the tendon forces [7-10,13-16,21,22,57]

For the multijoint fingers driven by tendons, the strain gauges, which are mounted on idler pulleys, are used to measure the tendon forces. Examples can be found in the Stanford/JPL hand [7-10], the Utah/MIT hand [13-16,22], and the tendon-actuated finger [57].

Most force sensors on the fingertips are in an array form. For instance, a force sensor array (3 x 3) was mounted on each fingertip of the Stanford/JPL hand by LOUCKE et al [54]. A force sensor array (16 x 16) has been mounted on each finger of the Utah/MIT hand by ALLEN et al [22]. The array type force sensor is also called tactile sensor.

The tactile sensing is defined as continuous sensing of variable contact forces. Different from the force sensors which only yield the net forces and torques, the tactile sensors can detect both the geometrical information of the object and the forces generated between the robot hand and the object.

As suggested in [58-61], tactile sensors should be array sensors on thin and flexible materials with high sensitivity, fast response, continuously variable output, and good spatial resolution. Various tactile sensors have been developed, which include: [58-60]

conductive rubber sensors, piezoelectric sensors, solid-state sensors, fiber optic sensors,capacitance sensors,etc.

The tactile array sensors (4 x 8) using conductive rubber have been incorporated into the sensing fingers of a JPL/CURV manipulator for construction and maintenance in space by HEER and BEJCZY [56] in 1983. Each element of the array sensors can measure contact pressure from 2 to 50 Pa.

An architecture of integrated tactile sensors mounted on the PAMH hand was described by GOLDWASSER [62] in 1984. As a part of an entire active sensory processor expert system, the tactile sensor array incorporates an analog multiplexer, ADC, and single chip microprocessor on a hybrid circuit. The signals form tactile sensor arrays are processed by the finger tactile processors.

The optical tactile sensors have been incorporated into a sensory gripper [63] for object recognition, orientation control and stable manipulation. The tactile sensor, which contains 16 needles with 4 mm space, is used to acquire three dimensional information about object contours of interest.

To mimic the tactile functions of the human fingertips, DARIO et al [57] used the multilayered tactile sensor to increase the fingertip sensing ability to detect the pulse rate of the human wrist. The sensor comprises a superficial (epidermal) sensing layer, an intermediate compliant layer, and a deep (dermal) sensing layer. Both sensing layers are made of ferroelectric polymer (PVF2) material, while the compliant layer is natural rubber.

Usually, an intelligent robot is constructed with multiple contact and noncontact sensors. To upgrade robot intelligence using multiple sensors, the data from the sensors must be integrated and processed in a right way. The efficient fusion of data from different sources will enable the machine to respond promptly in dealing with the real world [29].

Several approaches for multi-sensor integration schemes have been developed, such as sensor fusion [29,64,65], active sensory processing [66], control and monitoring system [30]. The main aim is to understand the real world and to infer the necessary actions the robot should take by using all the sensed information during an operation. Hence, effective sensor data fusion is critical to increasing robot capability. The more effective and complete data from the sensor resources are compiled, the greater the robot's ability to accomplish complex tasks. This is closely related to the AI functions in the robotic systems.

1-4 Compliance control

Compliance motion control may be defined as the ability to modify the manipulator motion based on the sensed contact information during the execution process of the tasks. Dealing with the interactions between a robot and its surroundings even if uncertainties exist, the compliance control is required for most robotic tasks. The control objectives of the compliance are to comply with either the geometrical constaints or the force constraints.

Two basic strategies have been employed to achieve the compliance motion control: passive approach and active approach.

The passive compliance control is achieved through the inherent mechanical compliance of the manipulator joints, servos, or by the specially designed compliance fixture devices, such as the Remote Center Compliance (RCC) device.

The passive RCC device [67-71], originally designed to support cylindrical pegs for assembly into cylindrical holes, is widely used in industrial assembly now. The RCC device is designed as spring-like mechanism, in which a pure force applied causes mostly translation and a pure torque causes rotation about the tip. The passive RCC devices are characterized with simple and low cost, but lack of the active and programmable ability.

The active compliance motion control is achieved by providing the manipulator with a programmable capability to react to force stimuli by constructing a force feedback control loop in the controller. As more and more emphasis have been put on the development of the active compliance control since 1970's, a considerable number of control strategies have been developed.

In 1977, WHITNEY [72] developed a force feedback control strategy using the resolved motion rate control and a force feedback matrix in the feedback loop for servoing a mechanical manipulator in fine motion control. This is the earliest description of the generalized damper approach to compliance control. WHITNEY's work has been classified into velocity based accommodation control in Cartesian space by MAPLES & BECKER [73].

In 1980, SALISBURY [74] described a method for actively controlling the stiffness of an robot arm. Using this method, the three translational and three rotational stiffness of a frame located arbitrarly in the robotic hand coordinates can be programmed.

Using the resolved force vector from the wrist force sensor, PAUL & SHIMANO [75] proposed a simple joint compliance motion control method by selectively servoing several joints to complete the insertion peg tasks. The main idea in [75] is:
" control forces applied to the object by selecting a certain joint (or joints) in the manipulator whose action is most closely aligned with the desired direction of force. The selected joints are then force controlled while the remaining joints are left under position control."

In 1981, MASON's theoretical work [76] on compliance control grounded the base for hybrid position/force control architecture proposed by RAIBERT & CRAIG [77]. The kinematics constraints imposed on manipulator motion due to a particular task geometry was discussed in [76]. Hybrid control was proposed to address the issue of control in the presence of natural constraints imposed by task geometry and artificial constraints imposed by the performance of the task itself. The use of artificial constraints orthogonal to the natural constraints was suggested as well. Once the constraint frame is specified, the directions in which position and orientation is constrained by task geometry may be defined with respect to the cartesian space. Therefore, in these directions, constraint forces and torques can be controlled, while in other cartesian space directions, position and orientation is controlled.

Based on the theoretical framework described in [76], RAIBERT & CRAIG [77] proposed a hybrid position/force control architecture to satisfy simultaneous position and force constraints on manipulator motion. This architecture consists of separate position and force control loops. The hybrid controller servos each degree of motion freedom, position or force, at the cartesian space by a closed loop. The joint drive signal is a linear combination of all position/force errors in the cartesian space.

Several arguments have been made for the hybrid control architecture proposed in [77]:

a. high cost computation [78]
b. neglecting manipulator dynamics [79,80]
c. instability [81]

An improved method was proposed by ZHANG & PAUL [78] to speed up the computation and to simplify the control algorithm by combining the stiffness control [74] with the hybrid control [77]. The dynamic hybrid control of the manipulator was discussed by YOSHIKAWA et al [79,80] and MILLS et al [82]. The stability of the hybrid controller proposed in [77] was found unstable for revolute manipulator [81].

To achieve a robust controller and an effective system, ARONNE & YANG [83] proposed a force control scheme which incorporates both the active compliance control and the passive compliance control. The RCC device was mounted between the wrist sensor and the tool. The hybrid control idea was used to minimize disturbance of the position controller.

Several programmable compliance devices have been designed since 1983 for the industrial assembly applications [84,85]

Realizing that both the characteristic of the robot and its environment should be considered, HOGAN [86] proposed an approach which is called impedance control to the control of dynamic interaction between a manipulator and its environment. The impedance control considers the effects of impedance on robot/environment interactions, when performed in task space, a known impedance can be maintained for all configurations. It is considered, however, to be solely a position control scheme, with small adjustments made to react to contact forces. Positions are commanded, and impedance are adjusted to obtain the proper force response.

An unified control approach called hybrid impedance control was proposed by ANDERSON & SPONG [87] by combining the hybrid control [77] with the impedance control [86]. The main feature of the proposed method is its adaptability.[87]

Compliance control, as stated in [88], is one of the key issues of the research in robotics. Research on the compliance control have been focused on the following aspects:

* establishing compliance models
* developing control strategies
* implementations

1-5 AI in robotics

AI is an embryonic technology dealing with the structure, interpretation, and presentation of knowledge, judgements, and inferences. AI involves all elements of investigation that simulate the features, attributes, and behavior of the human brain and related functions. The primary goal of AI is to make machines smarter and more useful. An AI system is usually constructed with: [89]

* knowledge of the domain of interest
* methods for operating on the knowledge
* control structures for choosing the control actions and modifying the data base as required

Robotics is generally regarded as a bright area of application of AI. Robots should be intelligent enough to perform the delicate tasks. An intelligent robot is expected to be capable of: [90]

* Receiving high level communications
* Understanding its environment
* Formulating plans based on reasoning
* executing plans and monitoring its operation

Though there is a long way to go for the robots to reach the human' abilities, many efforts have been made to incorporate AI into the robotic systems.

AI Planning in Robotics

Using the hierarchical approach, a robot expert planning system called ABSTRIPS was developed by SACERDOTI [91] in 1974 to devise plans for a robot to move objects between rooms. The knowledge base was constructed with configuration of the rooms, objects properties in the domain, and heuristic search rules, etc.

Unlike ABSTRIPS, which orders the subgoal sequence strictly, some systems do not enforce subgoal sequence until sufficient information exists -- a technique known as least commitment. A hierarchy with least commitment technique can be found in the AI planning softwares [92-93].

A knowledge based planning system [94] for mechanical assembly using robots was proposed in 1988. The planning efficiency was improved due to two novel features: problem analysis and goal-oriented hierarchical operation representation.

A telerobot interactive planning system (TIPS) [95] was developed in 1980s to perform planning for the space telerobots. An AI planning has also been developed in 1980s for a planetary rover which is used to explore and sample planetary surfaces [96]. With the abilities to recover from planning errors, the AI control module embedded in the planetary rover system can reason about plans, terminate or suspend partions of plans, add patches, and retry plans.

A knowledge based task planning and execution system was developed for an assembly workcell [97] in 1985. The system is constructed with off-line and on-line modules. The off-line module includes various planners based on geometric reasoning in order to structure the workcell space, to synthesize the various actions that can be executed, and to provide rules for action selection and scheduling. The on-line module is a knowledge based system. It maps the task execution parameters into the execution actions and performs the on-line control.

Real-time Knowledge Based System in Robotics

ADAPTIWELD [98] is one of the first arc welding systems to incorporate knowledge of the skilled welders in its information and control base. A three dimensional vision system is used to detect the characteristics of a seam to be welded. These characteristics are stored in the computer memory and are manipulated by the expert system to infer a set of welding actions. The expert system allows the system to perform autonomous welding. Also the expertise knowledge can be added into the welding system's knowledge base.

A robot system with learning ability and knowledge base has been proposed for the meat cutting applications [99] in 1989. The knowledge base has been constructed with the three dimensional models of typical carcasses, the cutting strategies, etc. Two 2-D cameras have been used to provide the geometric properties of the carcass being cut. The force sensor in the cutting device provides feedback of the cutting force. The sensed information will be processed by the AI controller of the robot. Hence, the on-line error-correction can be realized. The learning unit is used to update the data and knowledge needed in the meat cutting process. Further work of this system is to deal with the uncertainties by using fuzzy logic control method.

The on-line error-correction using the real-time expert system can also be found in the sheep shearing robots [100-102]. A sheep shearing robot needs delicate yet fast tactile action, efficient vision, and a sophisticated control and planning system capable of operating under the pressure of a real-time environment. The surface models of the sheep have been built into the knowledge base of the AI system. The surface model provides advance warnings of changes in surface curvature and serves as a reference for planning robot movements. A machine vision system is used to generate geometric models of the sheep's surface. Based on the surface model, the robot arm trajectory and the cutter attitude are planned. The knowledge of the shearing techniques, combined with force sensing and monitoring of unusual conditions in the adaptation mechanism of the robot, provides the inputs to a real-time expert system embedded in the sheep shearing robot system. Incorporated with the on-line recovery strategies, the real-time expert system is able to replan the shear strategy when the lower level path and trajectory adaptation is not sufficient.

The developments of the intelligent control systems for robots have been focused on incorporating on-line expert system into real-time path planning and error-correction. Jet Propulsion Laboratory [103] is developing this kind of AI systems, which consist of a planner expert system, a system diagnostic module, and execution with error recovery module, for the space robot systems.

Fuzzy Logic Based Control in Robotics

To upgrade the level of the intelligence of robotic systems, fuzzy logic based control modules have been incorporated into the AI systems in robotics. Introduced and formulated by ZADEH [104-108] since 1965, fuzzy logic, on which the fuzzy control is based, is an effective means of dealing with uncertainties and linguistic terms. Linguistic terms such as 'small' and 'big' may be defined as fuzzy sets. A fuzzy set is characterized by a membership function that assigns to each element in a given class a grade of membership ranging between zero and one. Therefore, heuristic knowledge may be used as basis for logical inference. Moreover, lingustic rules may be used for specification of control laws in control problems. Fuzzy sets allow for qualitative and imprecise information to be expressed in an exact mathematical way.

Derived from the fuzzy set theory, fuzzy logic deals with relations between fuzzy sets. Fuzzy logic is much closer in spirit to human thinking and reasoning than the traditional logical systems. As an extension of traditional Boolean logic, Fuzzy logic allows partial truth and partial falseness.

Motivated by ZADEH's work, MAMDANI et al [109-115] have pioneered the research on the applications of fuzzy logic controllers to the industrial processes. Recently, fuzzy logic controller is getting intensively studied and applied in Japan and USA due to its ability to: [116]

* incorporate expert knowledge into the control system
* make tough problems much simpler to solve
* improve system performance radically
* make the control system more flexible by carrying out the inference under uncertainties

The basic configuration of a fuzzy logic controller (FLC) is shown in Fig. 1.1.

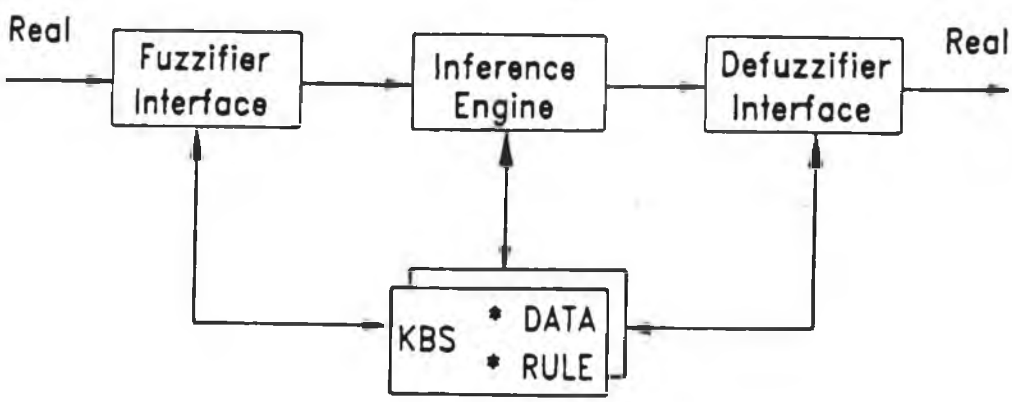


Fig. 1.1 Basic configuration of FLC

As shown in Fig. 1.1, a fuzzy logic controller consists of four major units: [116-118]

A. The fuzzifier interface which involves the following functions:

* measure the values of input variables
* perform a scale mapping to transfer the range of the values of input variables into corresponding universes of discourse
* perform fuzzification to convert input data into suitable linguistic values which may be viewed as labels of fuzzy sets

B. The knowledge base which consists of a data base and a lingustic fuzzy control rule base

* the data base provides necessary definitions, which are used to define linguistic control rules and fuzzy data manipulation in a FLC
* the rule base characterizes the control goals and control policy of the domain experts by means of a set of linguistic control rules

C. The inference engine which has the following capabilities:

* simulating human decision-making based on fuzzy concepts
* inferring fuzzy control actions employing fuzzy relations and the rules of inference

D. The defuzzifier interface which performs the following functions:

* scale mapping to convert the range of values of output variables into the corresponding universes of discourse
* defuzzification to yield a non-fuzzy control action from an inferred fuzzy control action

According to HUANG et al [116], the control rules in a FLC can be derived in several ways:

* based on the expertise experience/knowledge [119]

* based on the fuzzy model of the process [120]
* based on the operator's control action [121]
* based on learning algorithms [122,123]

The application of fuzzy logic to robotics was first conducted by URAGAMI et al [124] in 1976. The robot was able to move through a map space. The robot controls were based on fuzzy programmes. The fuzzy program [124] has been defined as an ordered sequence of fuzzy instructions. In the execution of a fuzzy program, fuzzy instructions are translated into machine instructions by the use of MAX-method and back-tracking.

The MAX-method is referred to the max-selection function used to select the machine instruction with the highest grade. The back-tracking is performed if the result of the interpretation of a fuzzy instruction is impossible to execute. The present state is replaced by the one step before. Then re-intepretation is carried out. Again, the machine instruction, which should be selected in the backtracking process, is the one with the highest grade among those which have never been selected.

A similar work on robots was also reported by GOGUAN [125]. Fuzzy linguistic hints were used to aid a robot running through a maze.

A robot with a knowledge base of movements was studied by HIROTA et al [126] in 1985. The knowledge base is mainly composed of control rules in terms of probabilistic sets in extended fuzzy expressions. The ambiguous instructions in terms of membership and vagueness are given to the robot. The robot is able to recognize these instructions and select an appropriate movement.

In 1985, SCHARF and MANDIC [122] presented a fuzzy Self-Organizing Controller (SOC) for a robot arm. The robot controller was contructed with the following features:

* the control rules are formulated through learning
* each algorithm needs to act in the direct forward path of its respective motor control loop
* the output of the controller is interpreted directly as the width of the motor drive pulse.

The SOC consists of the rule base, the performance matrix, the rule reinforcement and the history buffer. The learning function is realized by reference to an incremental performance matrix which has the same size and axes as the rule matrix. The performance matrix is derived from the fuzzy linguistic statements. Experiment shows that the performance of the SOC is superior to a conventional PID controller.

Further work on the SOC based on fuzzy logic was carried out by TANSCHEIT and SCHARF [123]. In the improved SOC, the input signals, which are mapped to one of the 13 discrete levels, are processed by using the rule-based control algorithm. The output signals, in a form of linguistics, will be mapped to a real value.

A fuzzy controller for a robot welding system was developed by KOUATLI et al [127]. The objective is to control the speed of the robot arm to carry out the weld in the same manner as the human welding operators. The fuzzy set shapes have been chosen as 'fuzzimetric arcs'. A scale for partitioning the universe of discourse is determined by using the expertise knowledge. The fuzzy reasoning is based on a compositional rule of inference. The speed of the robot arm controlled by the fuzzy logic controller varies with the cavity size of the workpiece being welded.

The fuzzy logic based controller has also been used by SARIDIS [128-130] to construct the linguistic decision modules for the intelligent robots.

Though not specially designed for robotic applications, the intelligent fuzzy logic controller proposed by RAY et al [131] will definitly have potential impact on the future intelligent robots. As suggested in [131], under normal operating conditions the controller will receive information of regular observations of plant data and select a suitable control strategy using compositional rule of inference. While under abnormal conditions, normal control actions are modified using knowledge based decision theoretic scheme.

The global analysis of fuzzy dynamical system was carried out by CHEN et al [132]. Using this method, the approximate prediction of the behavior of a FLC can be achieved.

To speed up the fuzzy inference processing, fuzzy logic chips and computers [133-136] have been developed since 1985.

The first fuzzy logic chip was designed by TOGAI and WATANABE [133] in 1985. The inference mechanism embedded in the VLSI chip is the max-min logic operation. A fuzzy logic accelerator (FLA) and fuzzy processor based on this chip are also available now [116,137].

YAMAKAWA et al [134] realized 9 basic fuzzy logic functions by the standard CMOS process in current-mode circuit systems in 1986.

As mentioned by LIM and TAKEFUJI [136] in 1990, incorporating reasoning system on hardware is significant because expert systems have to make decisions in real-time. Developing reasoning system hardware for an fuzzy processor system consists of two stages: specifying the fuzzy reasoning algorithm and designing special-purpose hardware.

The fuzzy chips and computers, on which the fuzzy inference speed is greatly enhanced, will speed up the applications of fuzzy logic controllers to the intelligent robot systems.

Chapter Two
Robotic Massaging Process

2-1 Introduction

Physiotherapic applications such as massaging the human body (arm,neck,back, etc.) are monotonous and tedious tasks. They are also time consuming and could be best carried out by a robot.

However, massaging the human body is a difficult task and is usually carried out by highly skilled professionals. The professional can take advantages of the well developed human coordination between the dexterous hands and eyes to locate the part to be massaged and to carry out the massaging manipulations. Also, he/she can utilize the knowledge about the human body and the trained knowledge about the massaging to perform the path planning and the necessary modifications based on his/her rough observations during a massaging process. To carry out the massaging operations, the robot system must be equipped with the necessary intelligence to meet the basic requirements of a massaging process.

In this chapter, the robotic massaging process has been defined in section 2-2. The basic construction of the robotic massaging system has been described in section 2-3.

2-2 Robotic massaging process

To simulate a massaging process which is carried out by a skilled professional, the robot system should be constructed with the massaging intelligent procedures to handle the complicated and difficult problems associated with part locating , parameter generating, path planning and on-line error corrections.

A robotic massaging process may be defined as a process of applying a series of predefined forces onto the part being massaged along a predefined massaging path, which may be modified by utilizing the sensory information.

Due to the geometrical difference of the part being massaged, different massaging strategies should be applied for different parts. Table 2.1 lists the required strategies for robotic massaging process.

Table 2.1 Robotic massaging strategies

Characteristics of the part	Massaging strategies for the part
Cylindrical or Conical shapes, such as Arm, Neck, Leg, etc.	Kneading by using the robotic fingertips
Flat surface shapes, such as Back, Chest, etc.	Padding by using the robotic palm

The following rules have been developed for the robotic massaging process.

Rule 2.1 For an unconstrained part Ω in the robot workspace , infinite geometrical massaging paths can be defined corresponding to the formulated massaging strategy.

Otherwise, finite geometrical massaging paths can be defined for a constrained part Ω in the robot workspace.

<u>Rule 2.2</u> For a given part Ω in the robot workspace, at least one massaging strategy can be formulated.

For example, for a segment of the human arm in the robot workspace, a massaging strategy can be formulated as: "kneading the arm along its axial direction."

<u>Rule 2.3</u> For any part Ω in the robot workspace, its geometrical properties can be represented by a virtual surface on which the geometrical massaging paths (axial and radial) are planned, the force is applied along the radial path.

<u>Rule 2.4</u> For a given part, the massaging force is proportional to its size.

The following functions have to be performed to carry out the robotic massaging:

a. Part locating -- Locate the part to be massaged in the robot workspace.

b. Parameter generating -- Generate the required parameters for the path planning.

c. Path planning -- Plan the massaging paths

d. Massaging execution -- Carry out the massaging

e. On-line error corrections -- Adjust the massaging path and force using the sensed information and the knowledge bases in the AI modules.

According to Rule 2.3, there are two types of massaging paths -- position path and force path, for a given part to be massaged in the robot workspace.

The position path is defined as a geometrical massaging path, which can be denoted by an axial path and a radial path. The force path is defined as a collection of the massaging forces exerted on the part surface along the radial path.

The virtual surface concept has been introduced into the path planning process. In general, a virtual surface may be defined as follows:

<u>Definition 2.1</u> Regardless of the local properties of the surface of a part being massaged, a virtual path surface can be constructed with the global properties of the surface to encompass the surface of the part being massaged.

Since the local properties of the part surface are not regarded, the part surface can be represented by a simpler form of virtual surface in a global range. For instance, the surface to encompass the fore arm of the human body can be denoted by either a conical or a cylindrical surface, which has been referred to as a virtual surface.

Once the virtual surface is defined for a given part, the position path can be formulated.

<u>Remark 2.1</u> For a part being massaged using the robotic fingers, an axial center line of the virtual path always exists. The axial center line is always followed by the finger grasp center while the opening of the robotic fingers complies with the radial path.

The massaging process carried out by a skilled professional is a process of intelligent path planning and intelligent on-line error corrections. This process cannot be achieved without using the knowledge acquired by the professional and the inference abilities of the human being.

To carry out the massaging process effectively, the robot system should be constructed with the knowledge base and the intelligent mechanism. Therefore, the robot system may be expected to be able to take the right actions for the given part, to apply appropriate forces onto the part being massaged, and to make the necessary adjustments whenever required during a massaging cycle.

An AI control system, characterized by the abilities to react in an uncertainty environment, generally comprises four components :

a. Man-machine interface
b. Sensing
c. Intelligent decision-making
d. Knowledge bases (KBS)

According to the time requirements of the robotic massaging system, the knowledge bases incorporated into the AI system can be divided into two types:

a. Off-line KBS
b. On-line KBS

The off-line KBS are referred to as the knowledge bases which are used to assist the operations without crucial time requirement. While the on-line KBS are referred to as the knowledge bases which are used to assist the operations with crucial time requirement. Hence, the off-line KBS are incorporated into the following modules:

a. Man-machine interface
b. Parameter generating

c. Path planning
d. Off-line fuzzy inference

And the on-line KBS are incorporated into the following modules:

a. Error correction
b. Path modifying
c. On-line fuzzy inference

2-3 Robotic massaging system

To develop the associated techniques for the physiotherapic applications, an experimental robotic massaging system has been constructed as shown in Fig. 2.1. The robotic massaging system includes:

A. Hardware

a. Robotic arm & its controller
b. Robotic hand & its controller
c. Position, force and tactile sensing units
d. Interfacing between the sensing units and the PC
e. Interfacing between the controller and the PC
f. Personal Computer (IBM-PC/AT)
g. A/D , D/A, and I/O boards on the PC

B. Software

a. Man-machine dialogue
b. Off-line KBS
c. On-line KBS
d. Path planning
e. Task execution
f. Error correction & path modifying
g. Sensing processing & interfacing
h. Servo loop controller

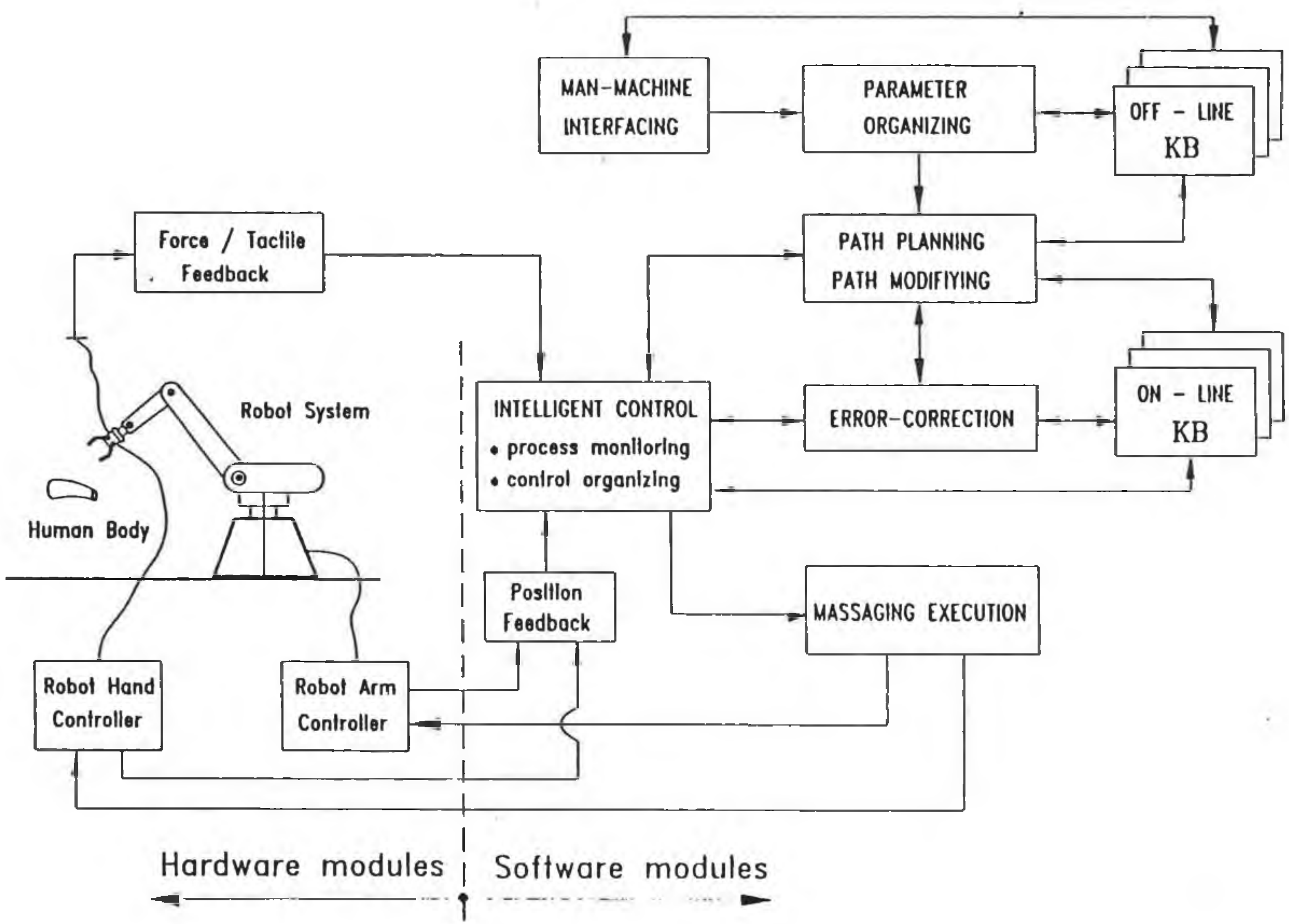


Fig. 2.1 Robotic massaging system

Hardware aspects

A Mitsubishi Movemaster robot arm with 5 DOF joints has been chosen for the experimental massaging system. The controller of the robot arm is able to communicate with any IBM compatible PC. Hence, the control over the robot arm can be achieved in a higher level control architecture, i.e., an intelligent control environment.

To simulate the human massaging operations, a robotic hand has been developed with the following features:

a. Two independent rotational fingers with position and force/tactile sensors.
b. One palm with force/tactile sensors
c. The fingertip is used to make contact with the part being massaged. It can provide a force upto 9 N.
d. The palm can provide a force upto 6 N.
e. The robotic hand controller with position and force control loop is able to interface with the PC.

An IBM compatible PC (80286) with maximum 12 MHz speed has been employed to deal with the sensing signals, computations, fuzzy inference and control.

The interfacing between the position/force sensors and the PC is carried out by the A/D (DAS8) and the D/A & I/O (DAC-06) boards which are inserted in the PC bus slots. The interfacing between the robotic arm controller and the PC is carried out by using the RS-232 port. And the interfacing between the robotic hand DC motors and the PC is performed by using the D/A (DAC-06) board.

Software aspects

There are two types of codes among the computer control and computation software:

a. Compiled BASIC code, which provides the control system with machine instructions.
b. Robot arm control code, which is provided by the manufacturer.

The motion of the robot arm can be realized by sending the robot arm control codes to the robot arm controller. For example, the code "NT" sent out by the PC will cause the robot arm to move back to its defined home position.

The compiled BASIC has been employed to develop and construct the following software:

a. Man-machine dialogue
b. KBS for off-line and on-line
c. Robotic kinematics computation
d. Intelligent path planning
e. Sensory information processing
f. Interfacing programming
g. Robotic hand digital controller
h. Fuzzy inference and decision-making
i. Intelligent control

When an anolog-to-digital conversion is performed by the A/D board, its driven software, which is written in assemblier language, can be incorporated into the compiled BASIC code by users.

The intelligent control software written in the complied BASIC can also be programmed with other computer languages such as C and LISP, provided that the language used can handle the information flows in the whole system.

Chapter Three

Configuration of the robotic massaging system

3-1 Introduction

In this chapter, the robotic massaging system configuration is presented in section 3-2. While the mechanical designs of the robotic hand and its sensing units are described in section 3-3.

3-2 System configuration

The main components of the robotic massaging system can be generated as follows:

* Robot arm (Mitsubishi type RV-M1), which moves the robotic hand to the specified position during a massaging process.

* Robotic hand (specially designed), which applies a series of predefined forces onto the part.

* Robot arm controller (RV-M1), which controls the motion of the robot arm.

* Robotic hand controller (specially designed), which controls the position and force of the robotic hand.

* PC (IBM PC/AT compatible), which performs the intelligent control and the sensory information processing.

Fig. 3.1 shows the general configuration of the robotic massaging system.

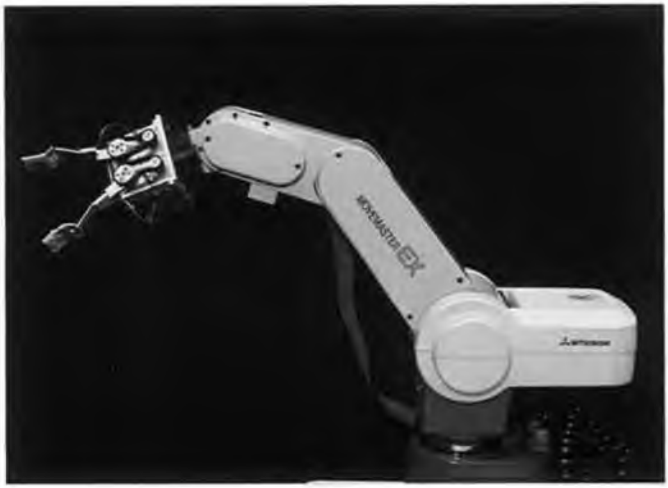


Fig. 3.1

General system configuration

3-2-1 Robot arm and its controller

The robot arm (Mitsubishi type RV-M1) is a revolute type manipulator with 5-DOF. Every joint of the robot arm is driven by a DC servo motor. The brake control systems have been applied to the J2 axis (shoulder) and J3 axis (elbow). The motion of the joint space of the robot arm is shown in Fig. 3.2.

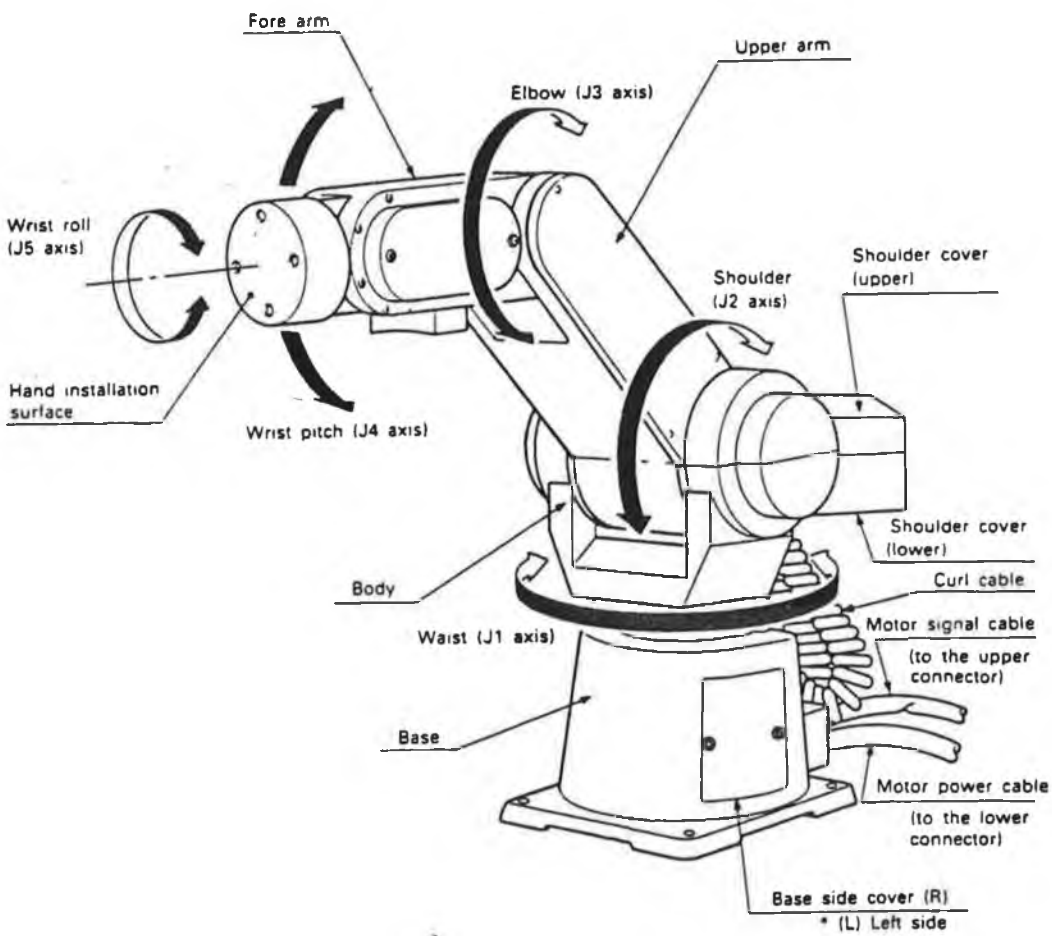


Fig. 3.2 Robot arm joint space motion

The robot arm together with its controller have been configured with the PC. The PC invokes the robot's joint motion by the intelligent commands provided in the robot arm controller. The intelligent commands for the robot arm are given in Appendix A-1. In this configuration, the PC has been used to control the robot to perform a variety of tasks.

The robot arm controller has a built-in arithmetic processing unit and a battery backed static RAM (Random Access Memory). Any commands from the PC will be stored in the RAM and executed under the control of the PC. The interfacing between the PC and the robot arm controller is made through RS232C.

The external dimensions of the robot arm and the mechanical interface (wrist mounting surface) between the robot wrist and the robot hand are shown in Fig. 3.3. And the specifications of the robot arm are given in Appendix A-2.

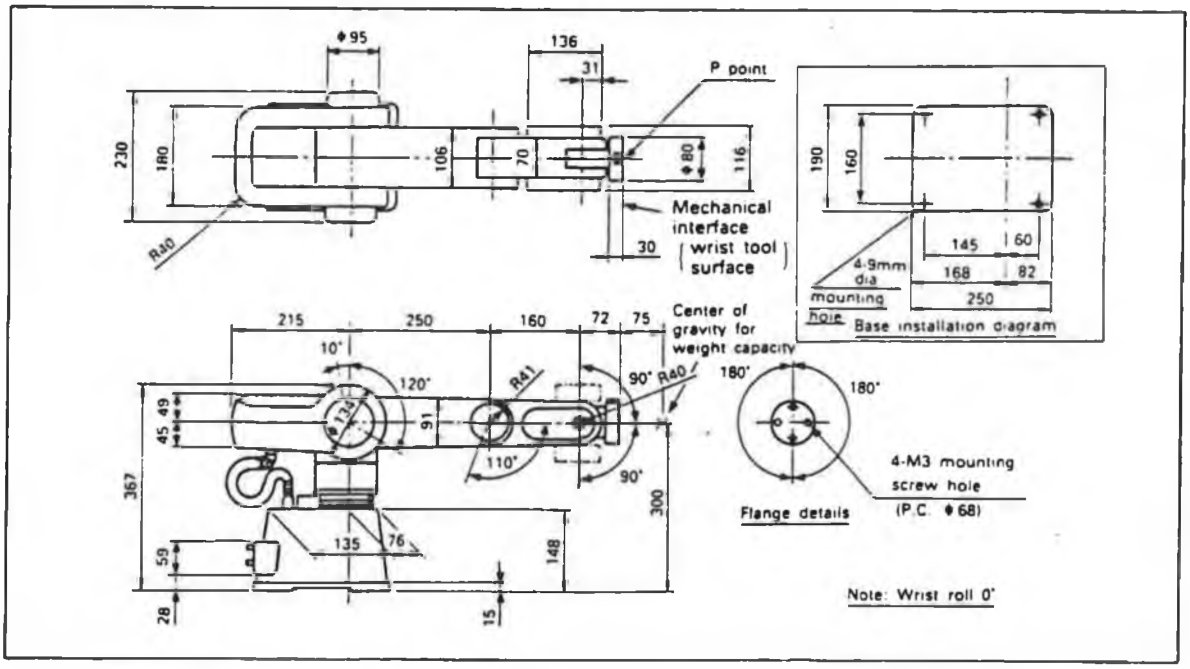


Fig. 3.3 Robot arm dimensions and mounting surface

3-2-2 Robotic hand and its controller

In this research project, two sets of robotic hands have been designed and developed, which are HAND-I and HAND-II.

The first hand (HAND-I) is an experimental model, where FSR (Force Sensing Resistor) sensors are used to constructed the force sensing fingertips and the palm. And potentiometers are used to construct the position sensing unit for the robotic fingers. Each robotic finger is driven by a DC-motor through a pulley-timing belt system.

The second hand (HAND-II) is a modified version of HAND-I. And HAND-II is intended to be a general purpose hand for different applications such as delicate material handling and massaging.

The drive system for HAND-II is almost the same as that of HAND-I. The differences between HAND-II and HAND-I in massaging applications are as follows:

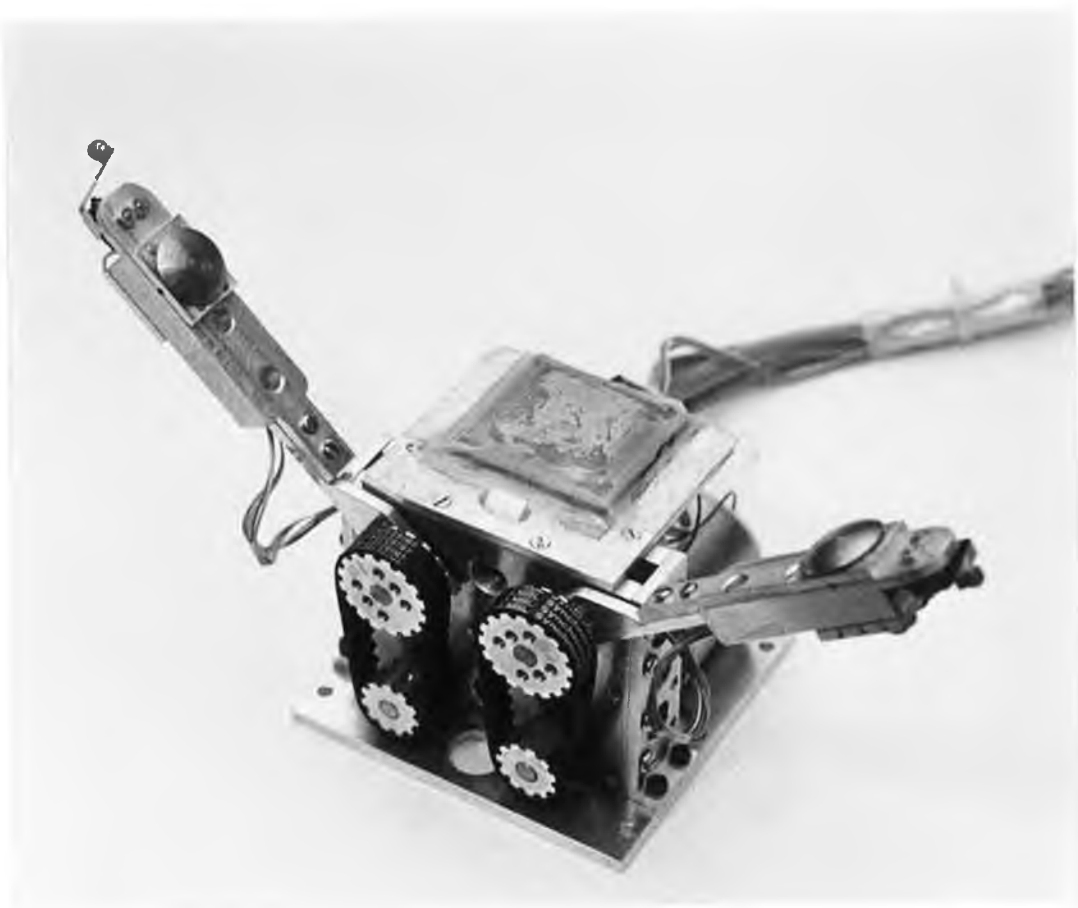
* A big size palm has been built in HAND-II.

* Load cells are used as force sensing units in the fingertips of HAND-II.

* Microswitches are mounted at the fingertips of HAND-II.

Fig. 3.4 shows the configuration of HAND-II.

The more detailed mechanical design of the robotic hands can be found in section 3-3.



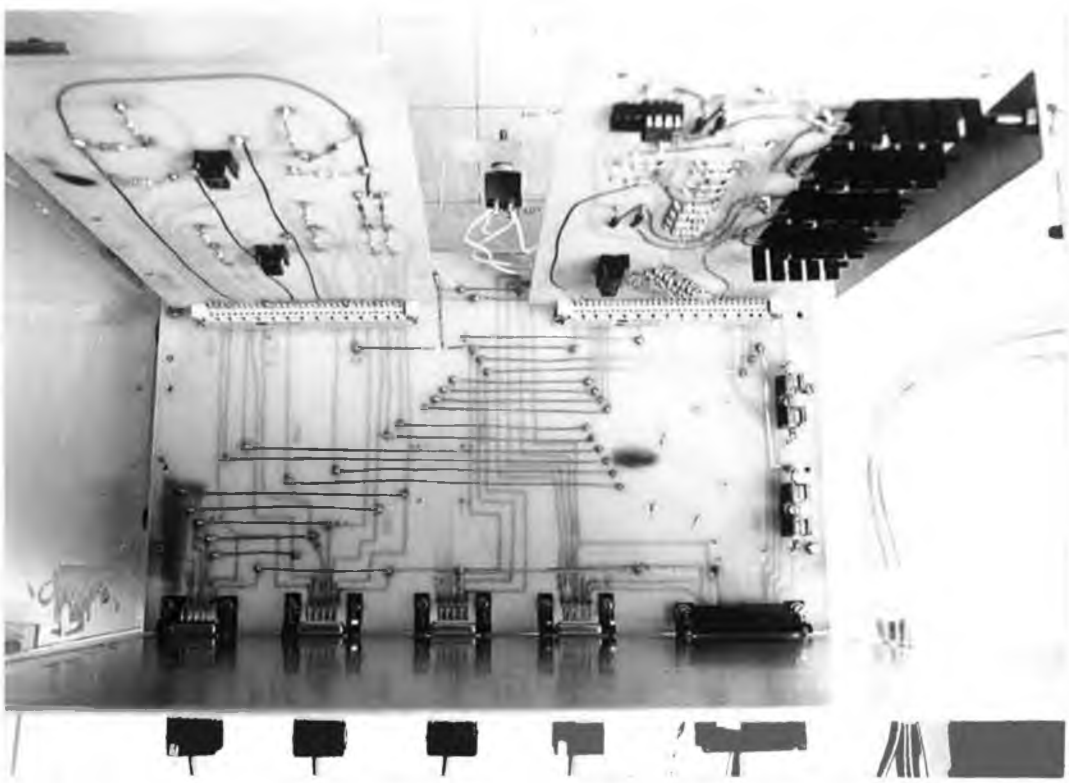
(photo)

Fig. 3.4 Robotic hand -- HAND-II

The robotic hand controller includes the following elements:

* DC-motor driver
* Sensor amplifiers
* Interfacing between the PC and the hand controller
* Computer control software

Fig. 3.5 shows HAND-II's hardware controller which includes motor drive PCB, sensor amplifier PCB, HAND-II signal port, power supply port, D/A port, A/D port and measurement port.



(photo)

Fig. 3.5 Hardware controller of HAND-II

3-2-3 Interfaces

A. Interfacing between robot arm controller and PC

The robot arm controller allows two types of interfaces for the link between the robot arm controller and the PC: Parallel and serial interfaces.

In the parallel interface mode, the PC sends 8 bits in parallel through the centronics port and the dedicated signal lines control the flow of data. The parallel transmission ensures the faster transmission speed and requires no special settings. But the following problems will discourage the use of the parallel interface:

* The data transmission distance is restricted to 1 to 2 meters.
* The data transfer is only one-way from the PC to the robot arm controller.
* Some intelligent commands such as position feedback of the robot arm cannot be used.

Thus, the parallel interface was not used in this study.

The serial interface, or RS232C interface, was originally the standard for data communication equipment using telephone lines and has evolved into the serial data transmission standard for the computers and their peripheral equipment.

In RS232C interface mode, the data are sent along a single wire (or channel), one bit at a time. Thus, it takes longer than in the parallel transmission if the baud rate is low. However, the capability of bidirectional data transfer enables the PC to read the robot's internal data such as position feedback. Also, the serial communication adapter permits a longer transmission distance than parallel communication (as long as 3 to 15 meters).

On the computer side, the serial port COM1 has been used to connect the RS232C connector embedded in the robot arm controller. The software key is used between the COM1 and the RS232C connector. The link between the COM1 in PC and the RS232C connector in robot is shown in Fig. 3.6.

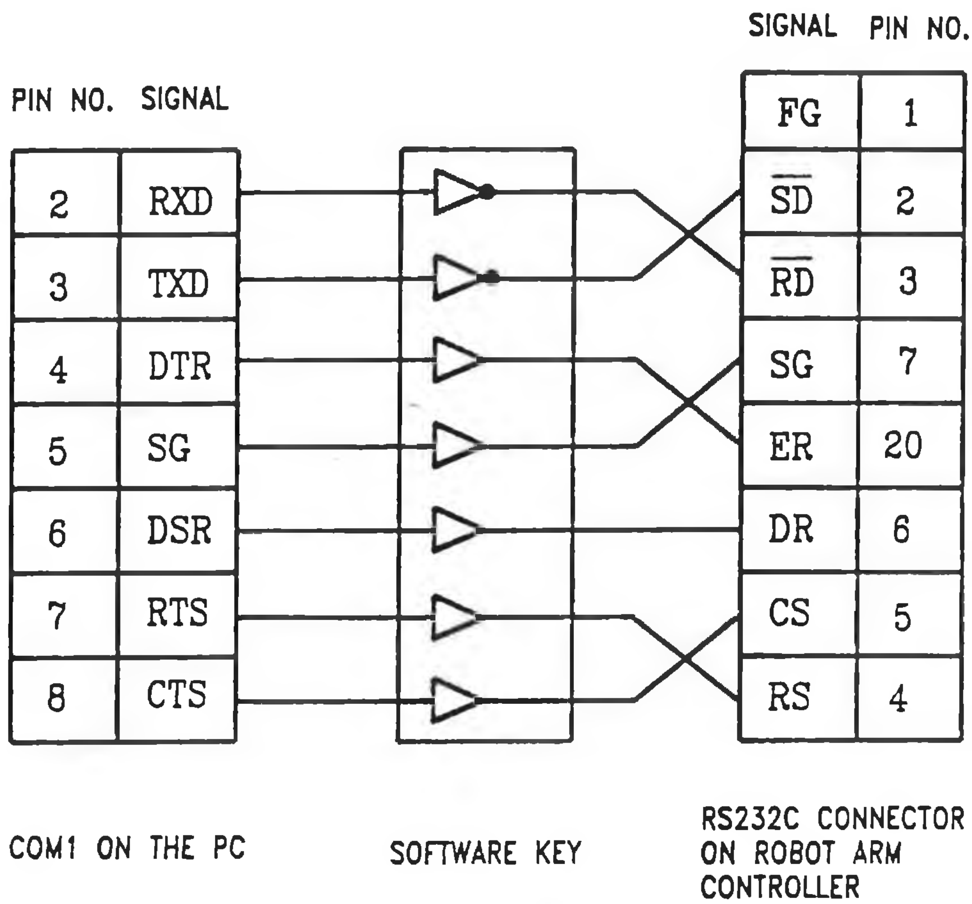


Fig. 3.6 Link between COM1 and RS232C connector

Table 3.1 shows the functions of the signals involved in the RS232C communication.

Table 3.1 Signal functions

Signals		Functions
RS232C	COM1	
$\overline{SD}$	TXD	TXD: transmit data $\overline{SD}$: the line on which the robot arm controller transfer data to the PC
$\overline{RD}$	RXD	RXD: receive data $\overline{RD}$: the line on which the PC transfer data to the robot arm controller
RS	RTS	RTS: request to send RS: the signal indicates the PC wishing to transmit data
CS	CTS	CTS: clear to send CS: the signal authorizes the robot arm controller to transmit data
DR	DSR	DSR: data set ready DR: the signal indicates that the PC is ready to transmit/receive data
ER	DTR	DTR: data terminal ready ER: the signal indicates that the arm controller is ready to transmit/ receive data
SG	SG	SG: signal ground for data lines
FG	--	FG: frame ground on the robot arm controller

To make the RS232C interface function efficiently, the communication condition settings must be made on the robot arm controller as well as the PC. The settings on the robot arm controller must be the same as those on the PC. Otherwise, the communication cannot be accomplished properly.

The serial communication port COM1 on the PC has been configured as follows:

```
Mode COM1:96,E,7,2,R
```

Where, the transmission rate is 9600 baud. The parity is EVEN. The number of data bits is 7. The number of stop bits is 2. And the COM1 port is in a return ready mode.

The settings on the robot arm controller have to be made to accomodate the settings of the COM1, as shown in Fig. 3.7.

The interface between the PC and the robot arm controller can be performed by opening a communication buffer in the PC. In the complied Basic, the communication buffer can be opened as:

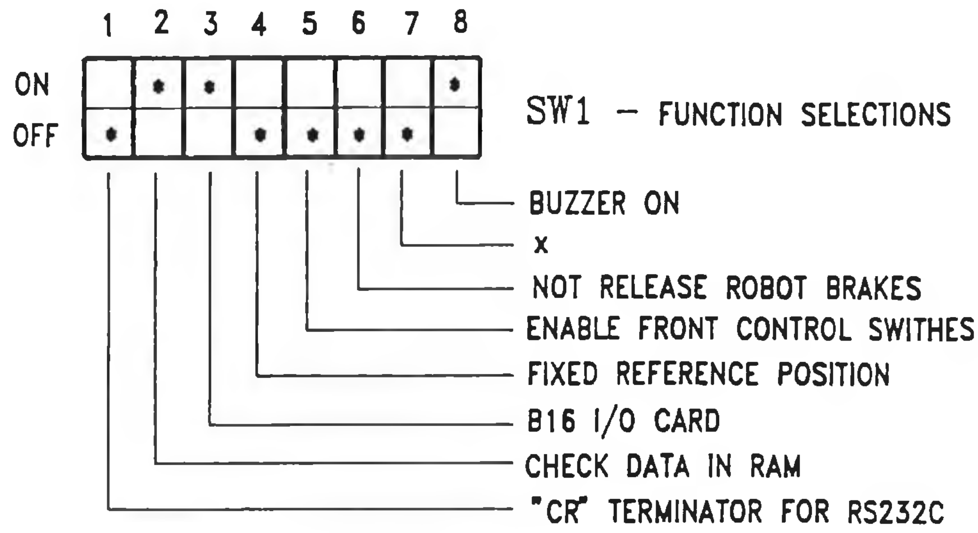
```
OPEN"COM1:9600,E,7,2,DS60000" AS #2
```

Thus, any control code for the robot arm can be sent out by this communication buffer as follows:

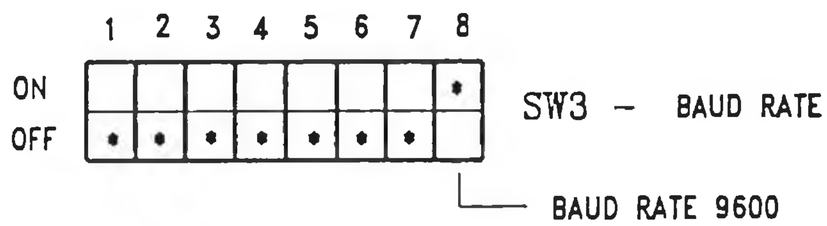
```
PRINT #2, "control codes"
```

For example, to move the robot to its home position, the following program is required:

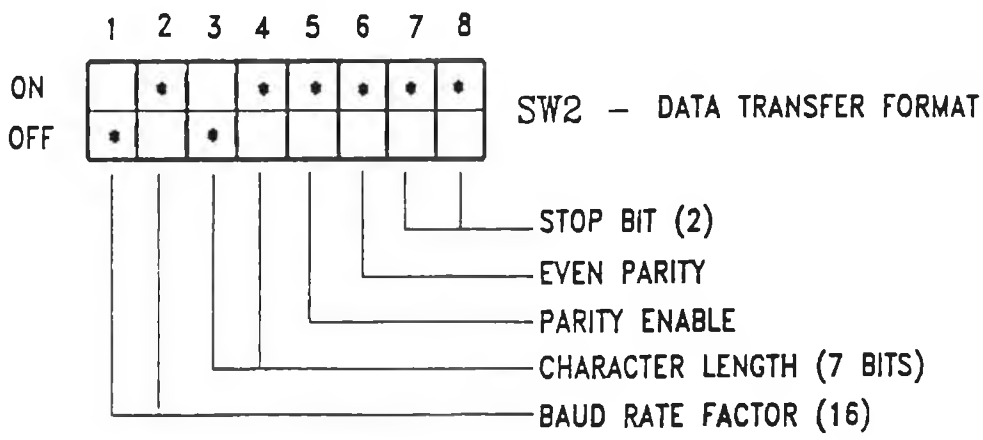
```
PRINT #2, "NT"
```



a). Function selection setting



b). Baud rate setting



c). Data transfer format setting

Fig. 3.7 Switch settings for robot arm controller

B. Interfacing between robot hand controller and PC

The robot hand controller consists of the motor drive circuit and sensor amplifier circuit. Thus, the D/A interface is required to supply the control voltages to the motor drive circuit. And the A/D and I/O interfaces are required to fetch the sensor signals into the PC.

Here, a DDA-06, which provides 6 channels of 12 bit analog output and 24 lines of digital I/O, is used to supply the control signals to the motor drive circuit and to fetch the microswitch detection signals. A DAS-8, which is an 8 channel 12 bit high speed A/D converter and timer/counter board, is used to fetch the position/force sensory information. Fig. 3.8 shows the functional arrangement of the interfacing between the hand controller and the PC.

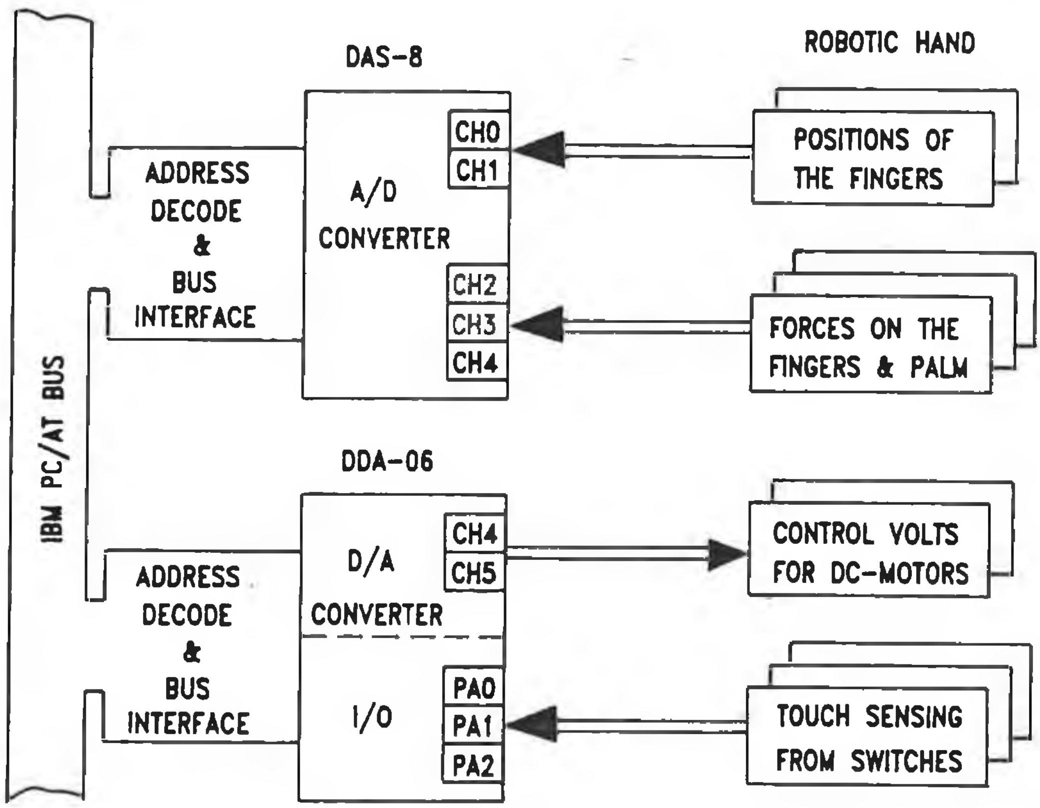


Fig. 3.8 Interfacing between the robot hand and the PC

The full scale input of each channel in DAS-8 is ±5 Volts with a resolution of 0.00244 Volts. A/D conversion time is typically 25 microseconds. The 8254 programmable counter timer embedded in the DAS-8 provides periodic interrupts for the A/D converter. The bus clock of the PC is used by the DAS-8 to drive the timer 8254. The base address of the DAS-8 has been set as &H300.

The output of the D/A channel in DDA-06 can be adjusted in a range of ±10 Volts. The I/O port may be independently programmed as an input or output and is TTL/CMOS compatible. Port A (PA) has been configured as an input port to fetch the signals from the microswitches. The base address of the DDA-06 has been set as &H310.

The interface signal connections are shown in Table 3.2

Table 3.2 Signal connections

Interface boards	Channel No.	Signals in robot hand
DAS-8	A/D CH0	θ_{F1} - Finger #1
	A/D CH1	θ_{F2} - Finger #2
	A/D CH2	F_1 - Finger #1
	A/D CH3	F_2 - Finger #2
	A/D CH4	F_3 - Palm
DDA-06	D/A CH4	V_{in} - Motor #1
	D/A CH5	V_{in} - Motor #2
	I/O PA0	Fingertip #1
	I/O PA1	Fingertip #2
	I/O PA2	Palm

3-2-4 Computer system

The IBM PC/AT compatible PC (Proturbo 286) used consists of a motherboard, 2 serial ports, 1 parallel port, a 33 MB hard disk, 2 floppy diskette drivers, a keyboard and an EGA monitor.

The motherboard, on which the function jumpers and switches are used to enable the various add-on features, has been constructed with the following components:

* CPU 80286
* Math coprocessor 80287-10
* 32 KB ROM
* 1 MB RAM
* Clock generator chip with speed selection switch on the front pannel (8MHz or 12 MHz)
* Bus controller chip
* Peripheral chips
* System unit expansion slots (5)

The speed of the PC is under the control of the speed button on the front panel. Normally, the system is running at 12 MHz, where the bus clock is selected as 12 MHz.

When the DAS-8 is inserted into the bus slot on the motherboard, the bus clock 12 MHz is selected to drive the timer in DAS-8.

3-3 Design & development of the robotic hands

3-3-1 Design specifications

The design objectives of the robot hand are as follows:

* The robotic fingers and robotic palm are integrated into the robotic hand

* The kneading operations can be carried out by using the fingers, while the padding operations can be carried out by the palm.

* The force applied by the hand should be programmable.

* The hand can sense its surroundings by using the touch sensors, position sensors, and force sensors embedded in the robotic hand.

The design specifications can be generalized as:

<u>A. For robotic fingers</u>

* Two rotational fingers driven by two DC-motors can be rotated independently.

* The maximum working torque provided by each finger is 0.9 Nm.

* The fingers can be either position controlled or force controlled

* The position sensor is mounted on the rotational center of each finger

* The force sensor is mounted on each fingertip

* The finger rotation range is from -5^0 to 95^0.

* The microswitch is mounted on each fingertip.

B. For robotic palm

* The maximum force provided by the palm is 5 N

* The force sensor is mounted on the top of the palm

* The physical size of the palm should be big enough with less weight

C. For robotic hand body

* The weight of the hand is less than 1.2 Kg

* The hand is easy to be mounted on the robotic wrist mounting surface.

3-3-2 Mechanical design of the robotic hands

As an experimental model for the massaging operations, HAND-I is featured by:

* FSR sensors are mounted on the robotic fingertips and the robotic palm to carry out the force sensing

* Potentiometers are mounted on the finger shafts to carry out the position sensing

* A microswitch is mounted on the palm to detect the

touch

* Each finger is driven by a DC-motor through a pulley-timing belt system

* Exchangable fingers

* Single mounting surface

As a modified version of HAND-I, HAND-II is featured by:

* The load cell are mounted on the fingertips to measure the forces

* The big size FSR is mounted on the palm to measure the force

* Potentiometers are mounted on the finger shafts to measure the rotational position of the fingers

* A microswitch is mounted on each fingertip

* Each finger is driven by a DC-motor through a pulley-timing belt system

* Multi mounting surfaces

* exchangable fingers

A. Selection of DC-motors

The DC motor (Maxon F-2140-934) with a gearbox (Maxon 2938.304-0100) has been selected as the drive unit for each robotic finger. The specifications of the DC motor can be found in Appendix B-1.

The gearbox construction employs a fibre wheel first stage followed by steel gears on bronze shafts. The gear-box reduction is $n_g = 1/100$.

The reversible motor employs an ironless rotor giving linear speed-torque performance. Considering that the torque constant of the DC motor is 28×10^{-6} Nm/mA and the maximum efficency of the DC motor is 81%, one may obtain the motor output torque constant K_T as:

$$K_T = 22.68 \times 10^{-6} \quad (\text{Nm / mA}) \qquad (3\text{-}1)$$

Since the allowable continuous output torque of the gearbox is $T_g = 0.6$ Nm, the maximum permissible DC motor armature current is given by:

$$I_{Lmax} = T_g \, n_g \, / \, K_T \qquad (3\text{-}2)$$

Where, $I_{Lmax} = 264.55$ mA

The weight of the DC motor with gearbox is 0.26 Kg. And the output drive shaft has a flat machined on it to simplify load coupling. Thus, a pulley can be easily mounted onto the output shaft of the gearbox.

B. Selection of the pulley-timing belt system

To increase the drive torque transmitted to the robotic fingers, a pulley-timing belt system has been used. The reduction, n_p, of the pulley-timing belt system is denoted by:

$$n_p = Z_1/Z_2 = T_p/T_F \qquad (3\text{-}3)$$

Where

n_p -- Reduction of the pulley-timing system

Z_1 -- Tooth number of the pulley on the gearbox shaft

Z_2 -- Tooth number of the pulley on the finger shaft

T_g -- Torque on the gearbox output shaft

T_F -- Torque on the finger rotational shaft

To meet the torque requirements of the robotic fingers, the pulley-timing belt (Mitsubishi, Synchrostar Timing belt) systems have been selected as shown in Table 3.3.

Table 3.3 Pulley-timing belt selections

items	HAND-I	HAND-II
Z_1	10 (10XL037)	11 (11XL037)
Z_2	16 (16XL037)	16 (16XL037)
belt	60XL037	60XL037
n_p	1/1.6	1.1/1.6

C. Placement of the position/force sensors

The conductive plastic servo potentiometers have been used as the position sensors to measure the rotational angle of the robotic finger. The specifications of the potentiometer can be found in Appendix B-2. To directly measure the rotational angle of the robotic finger, the potentiometer has been mounted on the robotic finger shaft.

The FSR sensors have been mounted on the fingertips and on the palm surface for HAND-I. While the load cells have been mounted on the fingertips and FSR sensor have been mounted on the palm for HAND-II. Due to the constant contact area requirement of the sensors, the special contact plates have been designed for the load cell and the FSR sensors (See the mechanical design section.).

The specifications of the FSR sensors can be found in Appendix B-3. And the specifications of the load cells can be found in Appendix B-4.

D. Structure of the robotic hand

Due to the limitation of the load capacity of the robotic hand, the weight of the robotic hand has been restricted to be under 1.2 Kg in the design process.

Where

$$W = \begin{cases} 1.100 \text{ Kg} & \text{for HAND-I} \\ 1.046 \text{ Kg} & \text{for HAND-II} \end{cases}$$

Considering the allowable gravity center for the weight capacity of the robotic hand, the robotic hand has been constructed with the weight center being close to the robotic wrist mounting surface. Thus, the motors have been arranged to be close to the wrist surface. And the robotic hand has been mounted on to the wrist through a mounting interface. The robotic palm has been designed to be parallel to the robotic wrist surface.

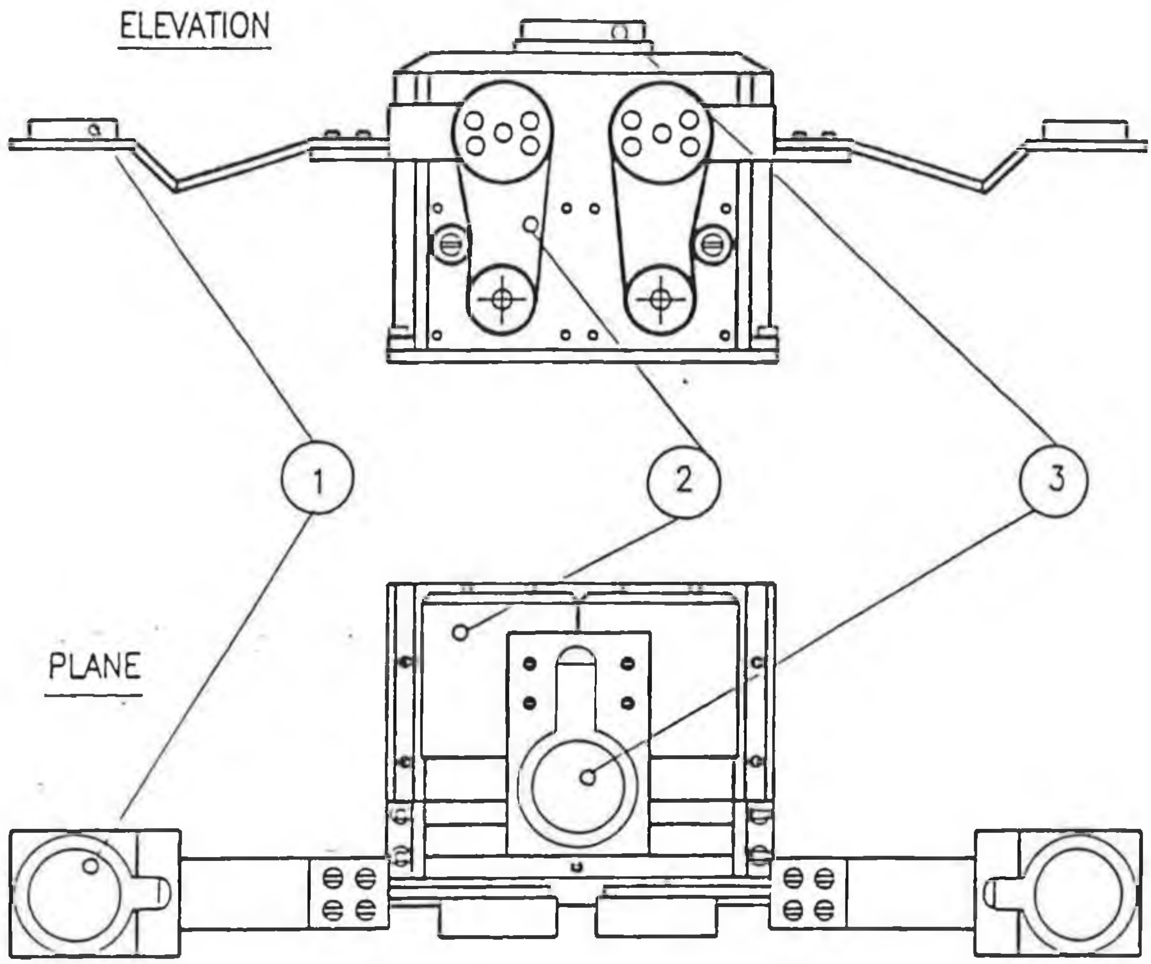
E. Mechanical design of the robotic hand

The mechanical design of HAND-I is shown in Fig. 3.9. HAND-I consists of three main parts: the robotic finger with FSR sensor, the body of the robotic hand with position sensors, and the robotic palm with FSR sensor. The design of the robotic finger with FSR sensor is shown in Fig. 3.10. The design of the body of the robotic hand with position sensors is shown in Fig. 3.11. And the design of the robotic palm with FSR sensor is shown in Fig. 3.12.

The detailed mechanical design drawings for HAND-I can be found in Appendix C-1.

The mechanical design of HAND-II is shown in Fig. 3.13. HAND-II consists of three main parts: the robotic finger with load cell as force sensing unit, the body of the robotic hand with position sensors, and the robotic palm with FSR sensor. The design of the robotic finger with load cell is shown in Fig. 3.14. The design of the body of the robotic hand with position sensors is shown in Fig. 3.15. And the design of the robotic palm with FSR sensor is shown in Fig. 3.16.

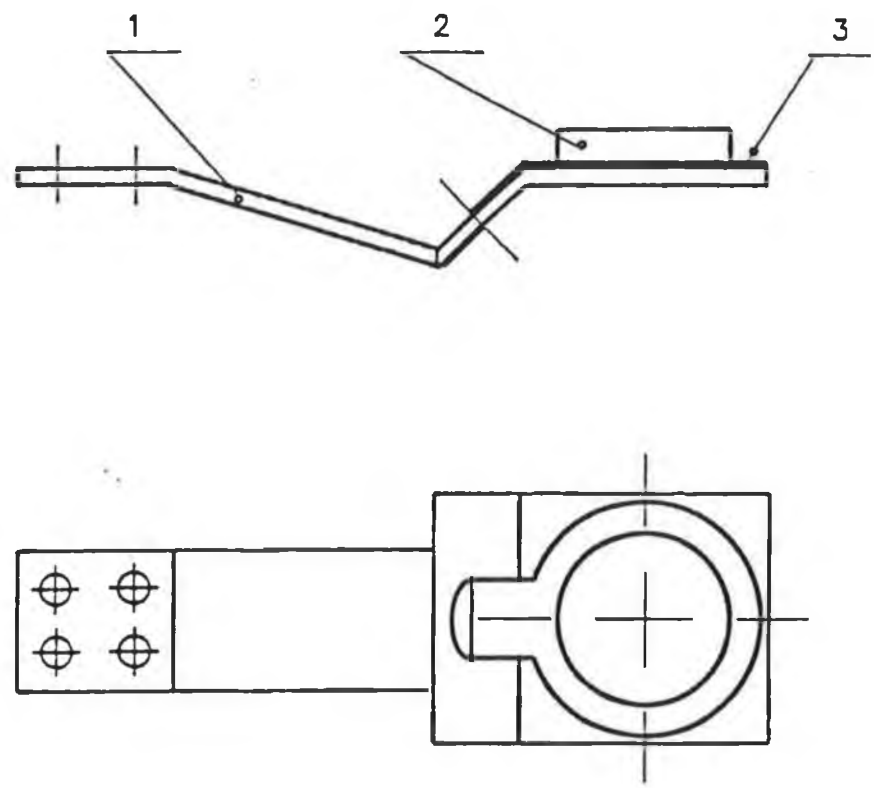
The detailed mechanical design drawings of HAND-II can be found in Appendix C-2.



No.	QNTY.	DESCRIPTION	DRG. No.
3	1	ROBOTIC PALM	H1-03
2	1	ROBOTIC HAND BODY	H1-02
1	2	ROBOTIC FINGER	H1-01

Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:2	Title: ROBOTIC HAND (HAND-i)
		DRG. NO. H1-00

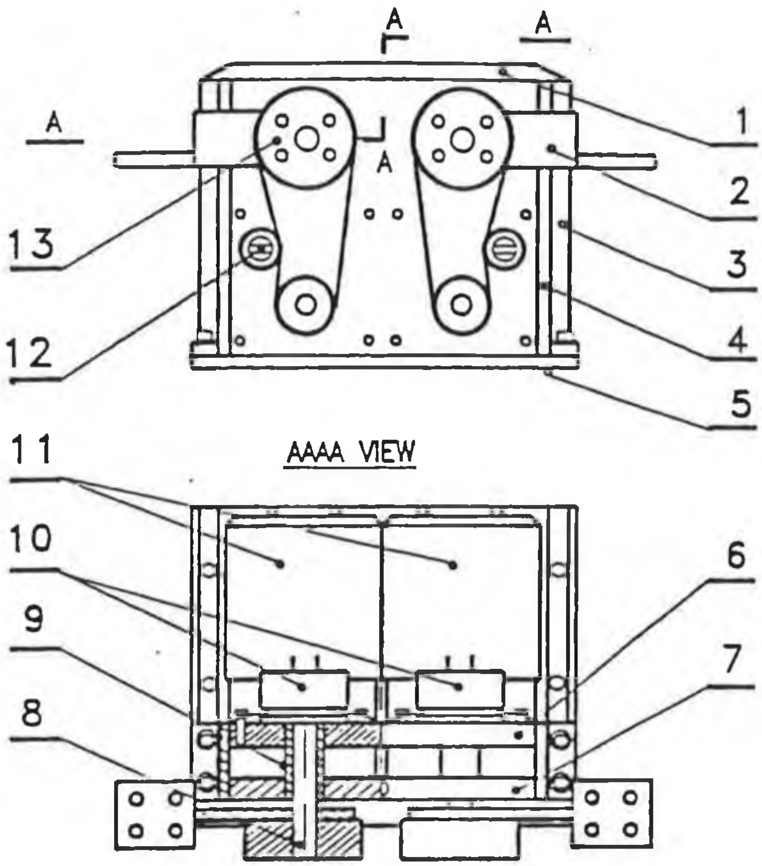
Fig. 3.9 Robotic hand design -- HAND-I



No.	QNTY.	DESCRIPTION	DRG. No.
3	1	FSR SENSOR	
2	1	CONTACT PLATE	H1-01-02
1	1	FINGER BODY	H1-01-01

Designed by: J. Yan	Dublin City University
Unit: mm — Scale: 1:1	Title: ROBOTIC FINGER
	DRG. NO. H1-01

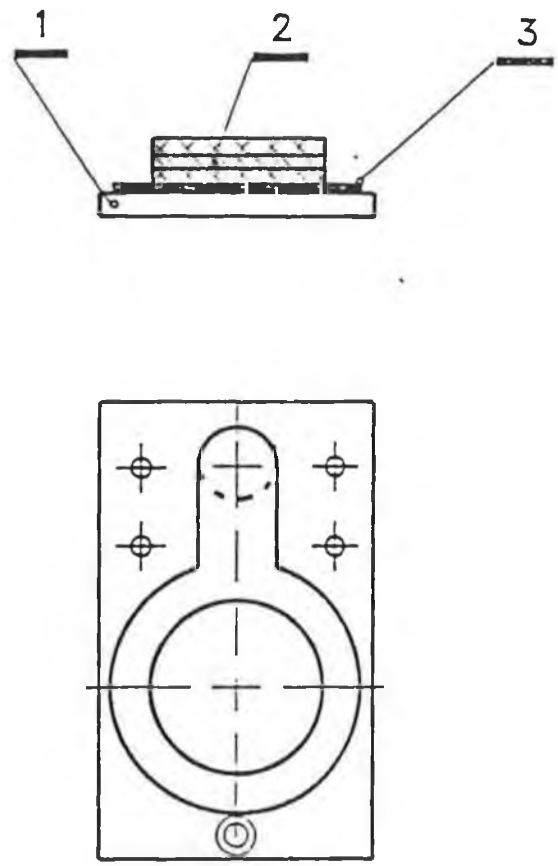
Fig. 3.10 Robotic finger design for HAND-I



No.	QNTY.	DESCRIPTION	DRG. No.
13	2	TIMING-BELT & PULLEY	
12	2	IDLER BEARING	
11	2	DC-MOTOR	
10	2	POTENTIOMETER	
9	2	BUSH	H1-02-09
8	2	FINGER SHAFT	H1-02-08
7	1	SUPPORT WALL B	H1-02-07
6	1	SUPPORT WALL A	H1-02-06
5	1	HAND BASE	H1-02-05
4	2	ENFORCEMENT WALL B	H1-02-04
3	2	ENFORCEMENT WALL A	H1-02-03
2	2	FINGER BASE	H1-02-02
1	1	PALM BASE	H1-02-01

Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:2	Title: ROBOTIC HAND BODY
	DRG. NO. H1-02

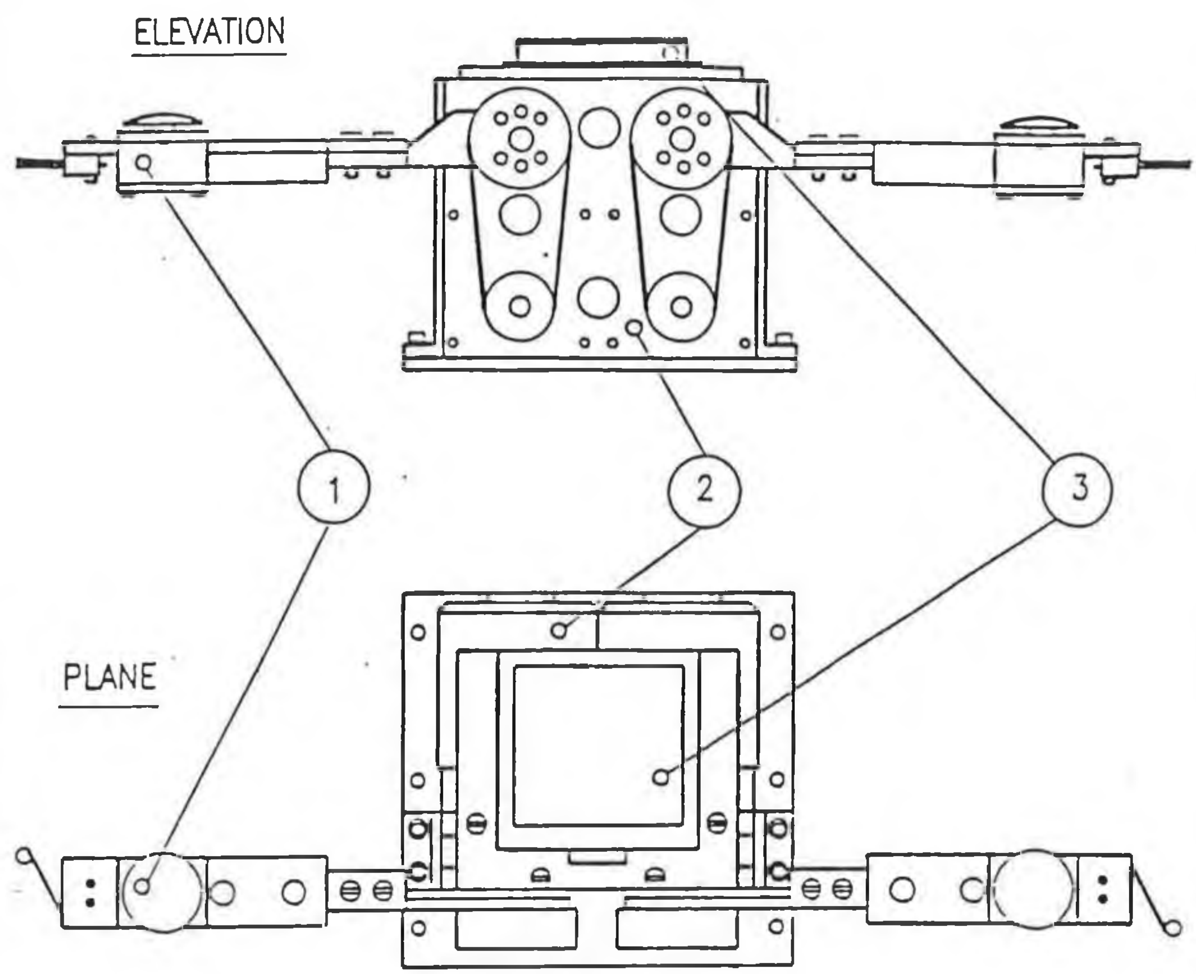
Fig. 3.11 Robotic hand body design for HAND-I



No.	QNTY.	DESCRIPTION	DRG. No.
3	1	FSR SENSOR	
2	1	CONTACT PLATE	H1-03-02
1	1	PALM BODY	H1-03-01

Designed by: J. Yan	Dublin City University
Unit: mm	Scale: 1:1	Title: ROBOTIC PALM
	DRG. NO. H1-03

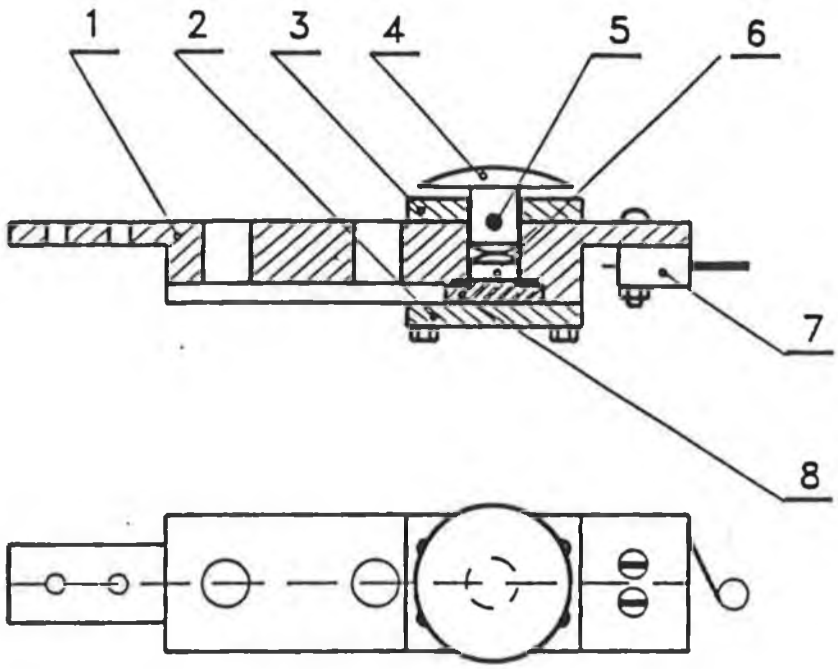
Fig. 3.12 Robotic palm design for HAND-I



No.	QNTY.	DESCRIPTION	DRG. No.
3	1	ROBOTIC PALM	H2–03
2	1	ROBOTIC HAND BODY	H2–02
1	2	ROBOTIC FINGER	H2–01

Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:2	Title: ROBOTIC HAND (HAND–II)
		DRG. NO. H2–00

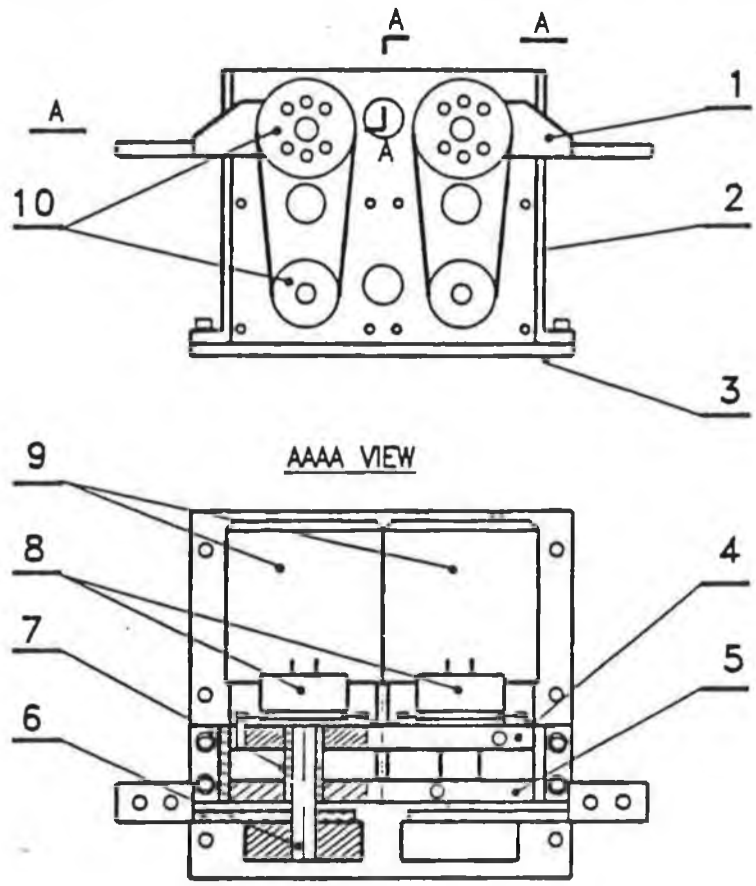
Fig. 3.13 Robotic hand design -- HAND-II



No.	QNTY.	DESCRIPTION	DRG. No.
8	1	LOAD CELL	
7	1	MICROSWITCH	
6	1	CONTACT CYLINDER	H2-01-06
5	1	STOP BAR	H2-01-05
4	1	TOUCH CAP	H2-01-04
3	1	FINGER TOP	H2-01-03
2	1	FINGER BOTTOM	H2-01-02
1	1	FINGER BODY	H2-01-01

Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: ROBOTIC FINGER
		DRG. NO. H2-01

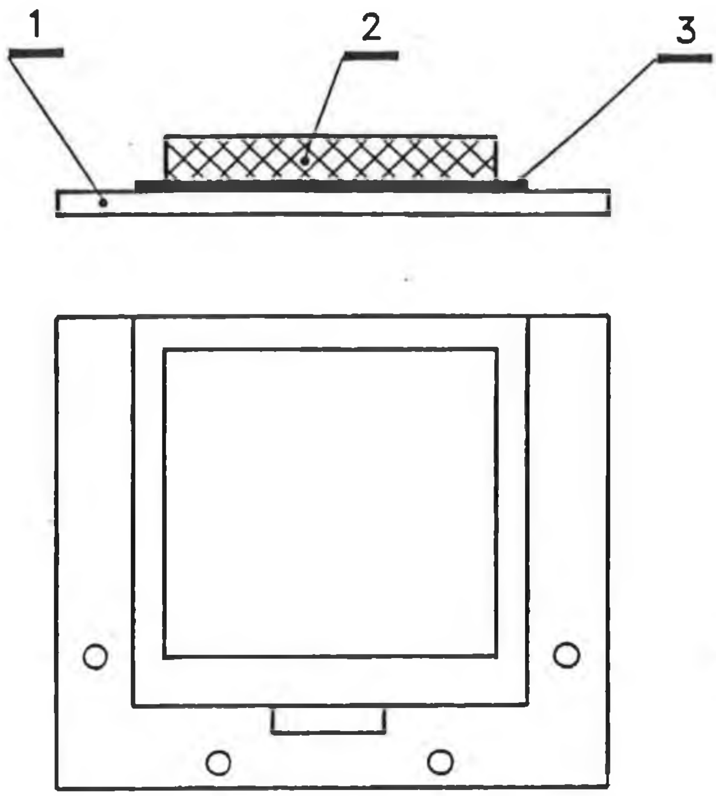
Fig. 3.14 Robotic finger design for HAND-II



No.	QNTY.	DESCRIPTION	DRG. No.
10	2	TIMING-BELT & PULLEY	
9	2	DC - MOTOR	
8	2	POTENTIOMETER	
7	2	BUSH	H2-02-07
6	2	FINGER SHAFT	H2-02-06
5	1	SUPPORT WALL B	H2-02-05
4	1	SUPPORT WALL A	H2-02-04
3	1	HAND BASE	H2-02-03
2	2	ENFORCEMENT WALL	H2-02-02
1	2	FINGER BASE	H2-02-01

Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:2	Title: ROBOTIC HAND BODY
	DRG. NO. H2-02

Fig. 3.15 Robotic hand body design for HAND-II



No.	QNTY.	DESCRIPTION	DRG. No.
3	1	FSR SENSOR	
2	1	CONTACT PLATE	H2-03-02
1	1	PALM BODY	H2-03-01

Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:1	Title: ROBOTIC PALM
	DRG. NO. H2-03

Fig. 3.16 Robotic palm design for HAND-II

Chapter Four

Development of the end-effector's controller -- Hardware and Software

4-1 Introduction

Two types of controllers have been developed for the robotic end-effector, which include:

a. Position servo loop controller
b. Force servo loop controller

Each of the controllers consists of the electronic hardware and computer control software. The electronic hardware includes the motor driver, sensor amplifiers and the interfacing boards. The computer control software includes the digital control algorithms and computation algorithms.

In this chapter, the design and development of the end-effector's controllers are described, together with the development of the required sensor's amplifiers. The overall configurations of the controllers are given out in section 4-2. The sensors and their amplifiers are described in section 4-3. The motor drive circuit design is presented in section 4-4. The PCBs designed are shown in section 4-5. And the position servo controllers of the robotic fingers are designed in section 4-6. While the force control of the robotic fingertips and the palm is investigated in 4-7.

4-2 Configurations of the controllers

The controllers of the robotic hands for position and force servo controls may be generalized as shown in Fig. 4.1.

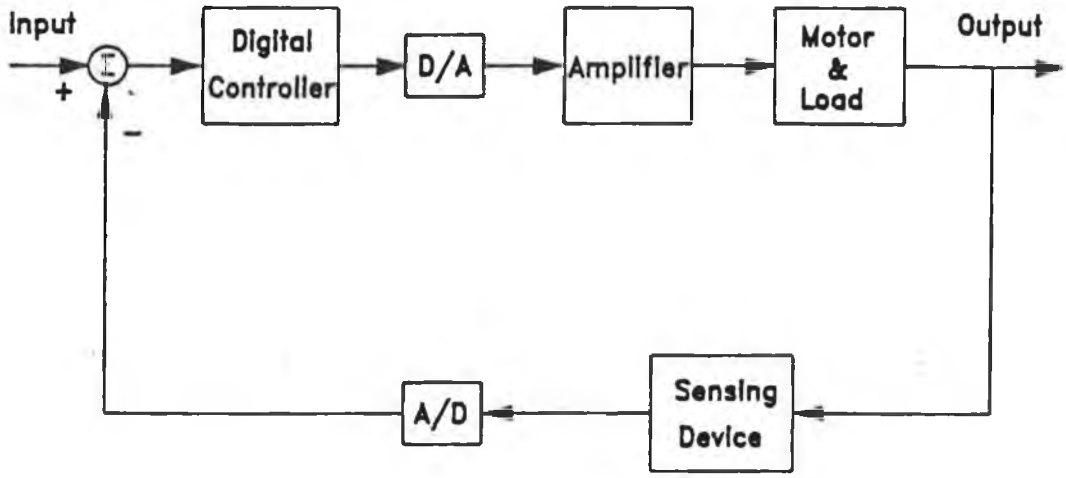


Fig. 4.1 General closed-loop controller

Here the input may be either desired position or desired force. And the response of the system is the output. The digital controller and the amplifier are used to drive the motor which then drives the load. The sensed output is compared with the desired input to produce an error signal, which, in turn, drives the controller/amplifier, and the motor.

The position control architecture for the robotic fingers is shown in Fig. 4.2. Where the current feedback amplifiers are used to adjust the current across the motor terminals. The servo potentiometers are used to detect the rotational angles of the robotic fingers.

The force control architecture for the robotic fingers is shown in Fig. 4.3. Where the Force Sensing Resistors (FSR) are mounted on the fingertips of Hand-I, and the Load Cells are mounted on the fingertips of Hand-II. Both the FSR and the Load Cell force sensors provide the direct measurements about the force generated between the fingertips and the part to be manipulated.

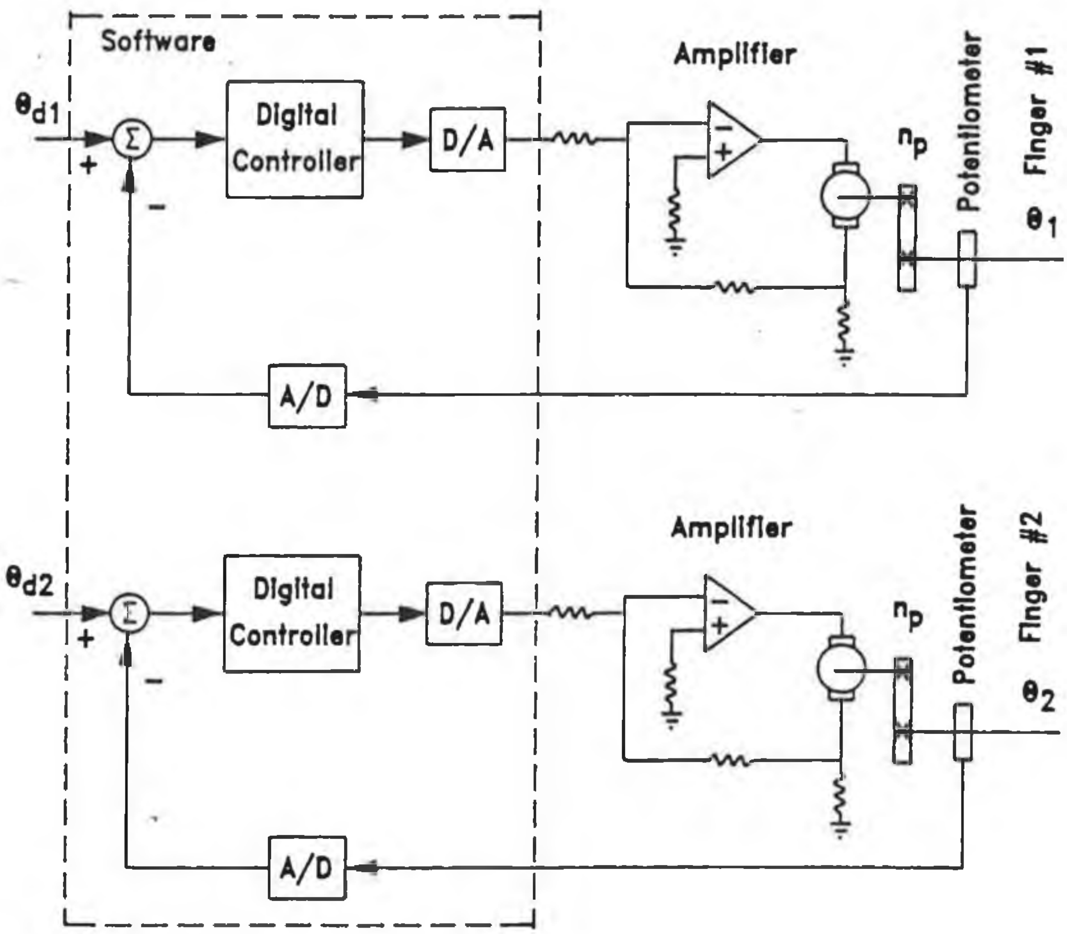


Fig. 4.2 Finger position control architecture

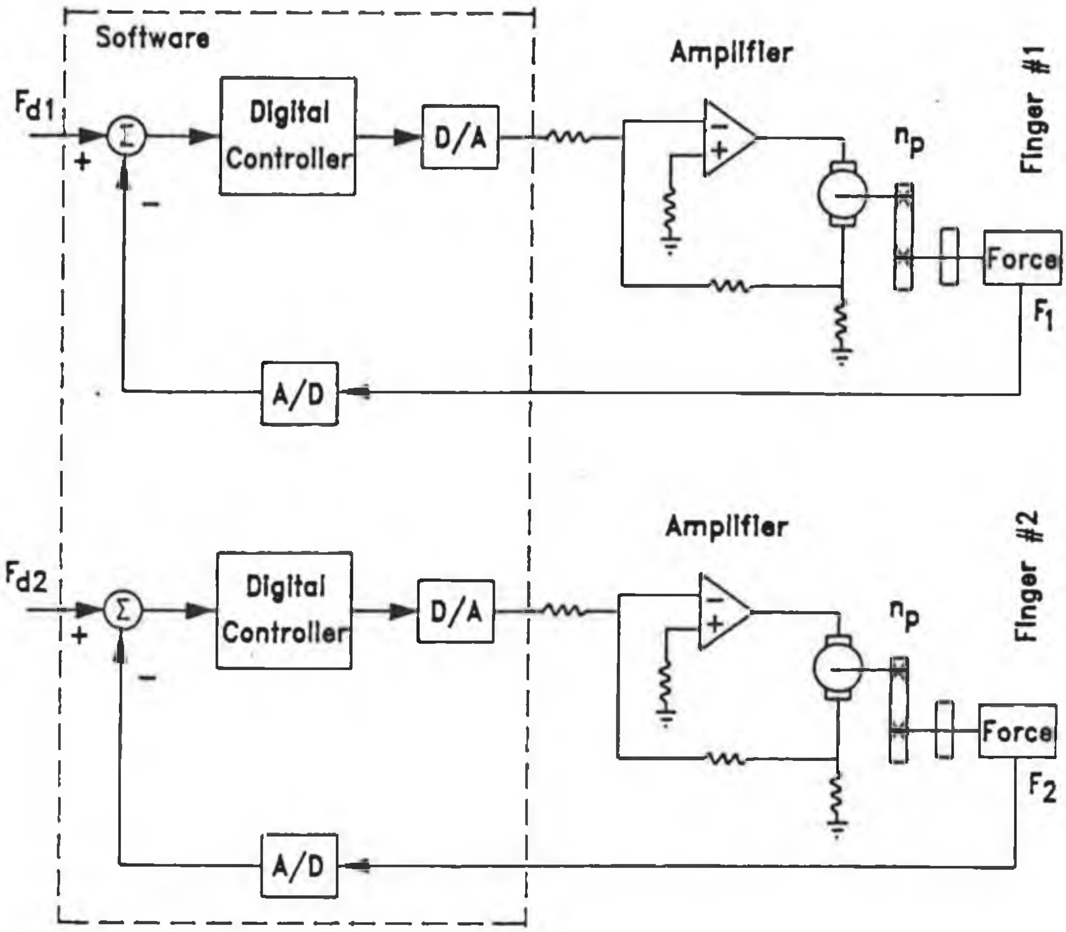


Fig. 4.3 Finger force control architecture

4-3 Sensors and their amplifiers

The following sensors have been incorporated into the designed robotic hands:

a. Servo potentiometers
b. FSR sensors
c. Load Cells
d. Microswitches

Table 4.1 Calibrations and pin connections

	Hand	Hand-I		Hand-II	
	Finger	No. 1	No. 2	No.1	No.2
Connections	+ 5V	Pin 3	Pin 1	Pin 3	Pin 1
	GND	Pin 1	Pin 3	Pin 1	Pin 3
	V_θ	Pin 2	Pin 2	Pin 2	Pin 2
Calibration	V_o	1.80		1.85	
	K_θ	1.34/90		1.21/90	

4-3-2 FSR sensor amplifiers and calibrations

The FSR is a ploymer film device that exhibits a decreasing resistance with increasing force. Since the change of the FSR is related with the force exerted on the surface and the contact area which is usually called force "footprint", the force contact area must be kept constant to obtain the repeatable force measurements under the same loads. Three FSR sensors with the size of 1" diameter have been used in Hand-I to detect the forces on the fingertips and on the palm. And one FSR sensor has been used in Hand-II to detect the force on the palm. More detailed specifications for FSR may be found in Appendix B-3.

To ensure that the contact area is constant, a flat plate coated with the silicon rubber is used as the medium between the FSR sensor and its environment.

To convert the resistance change on FSR sensor unit into the voltage signal for easy processing by the PC, a linear amplifier circuit shown in Fig. 4.4 has been employed.

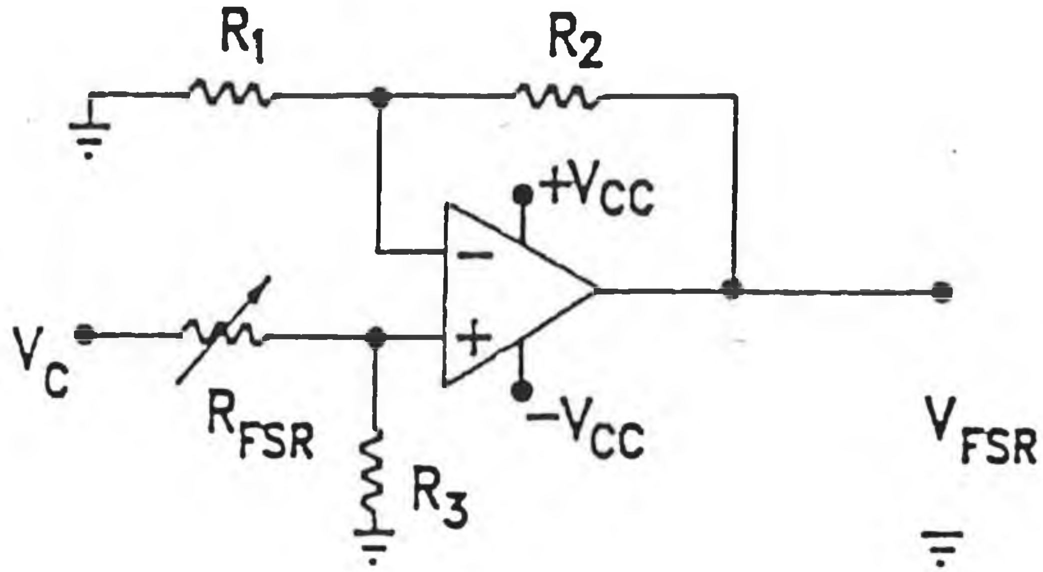


Fig. 4.4 Amplifier circuit for FSR

The relationship between the output of the amplifier and the resistance of the FSR is denoted by:

$$V_{FSR} = \frac{(R_1 + R_2)}{R_2} \frac{R_3}{(R_3 + R_{FSR})} V_c \qquad (4-2)$$

Where R_{FSR} is the resistance of the FSR sensor.

The selection of the resistors in the amplifier circuit for FSR is listed in Table 4.2.

<u>Table 4.2 Resistors selection</u>

Hand	Circuit for FSR on	R_1 (KΩ)	R_2 (KΩ)	R_3 (kΩ)
	Finger #1	5.5	1	20
Hand-I	Finger #2	5.5	1	10
	Palm	5.5	1	20
Hand-II	Palm	5.5	1	10

Using the amplifier circuit depicted in Fig. 4.4, the calibrations for the FSR sensors have been carried out. The calibration curves are shown in Fig. 4.5. And the calibration curve equations are listed in Appendix B-5.

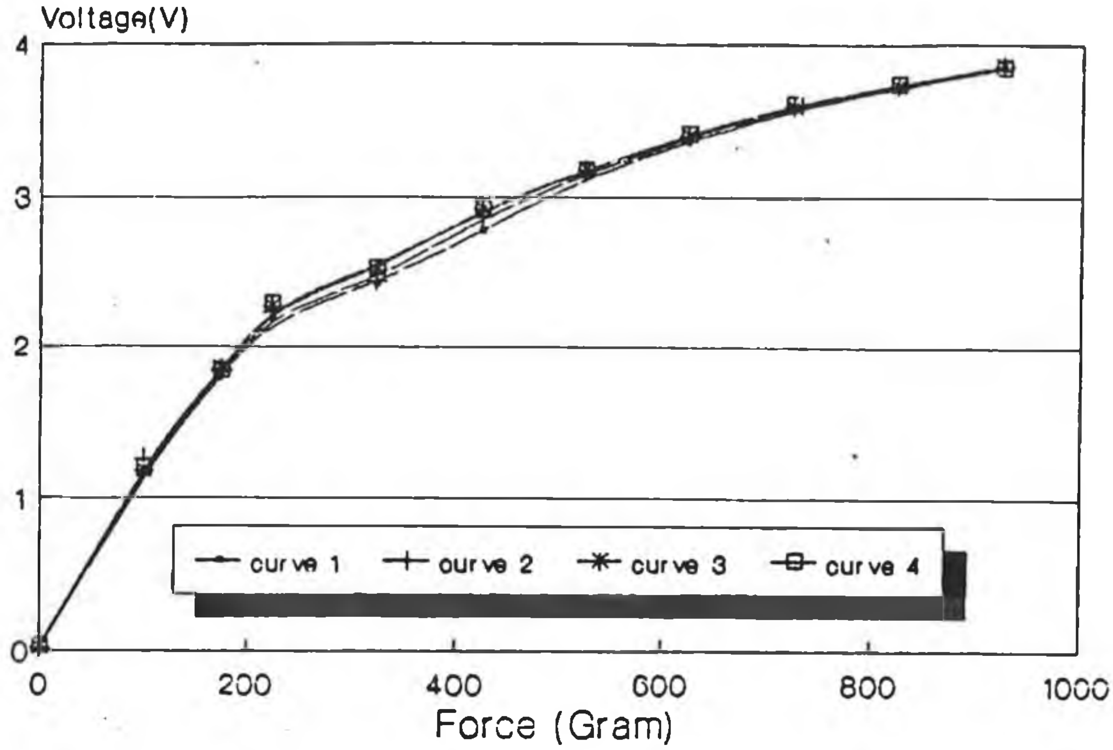


Fig.4.5 a). Calibration curve for FSR on Finger #1

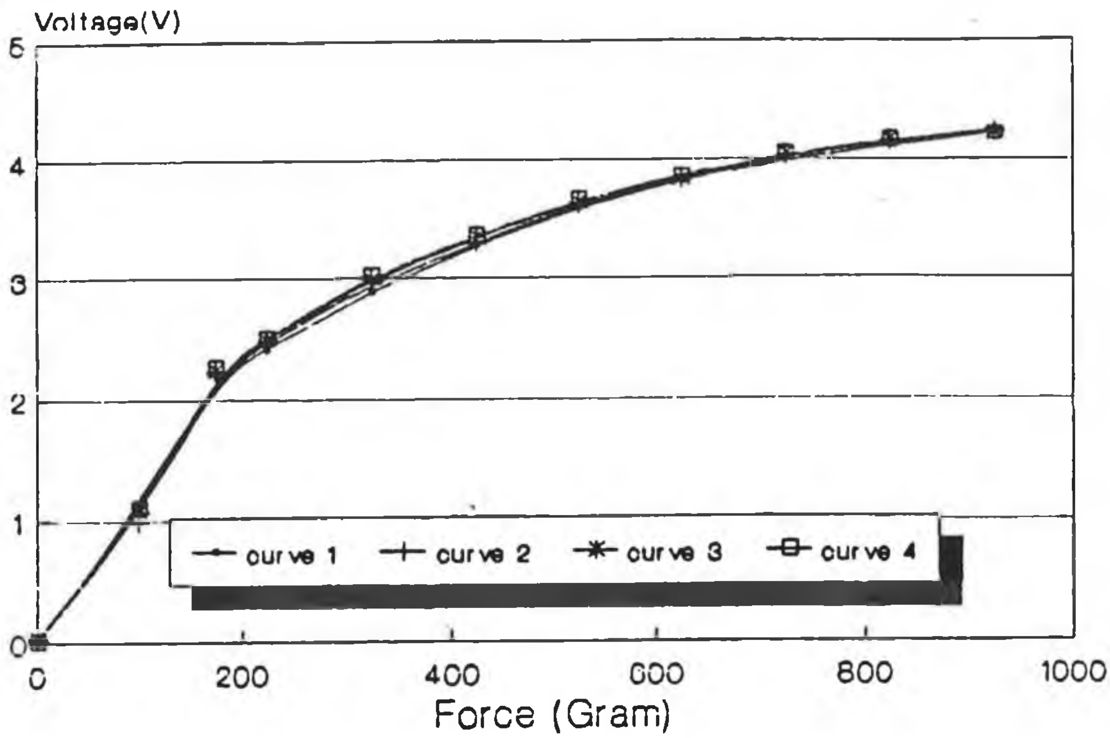


Fig.4.5 b). Calibration curve for FSR on Finger #2

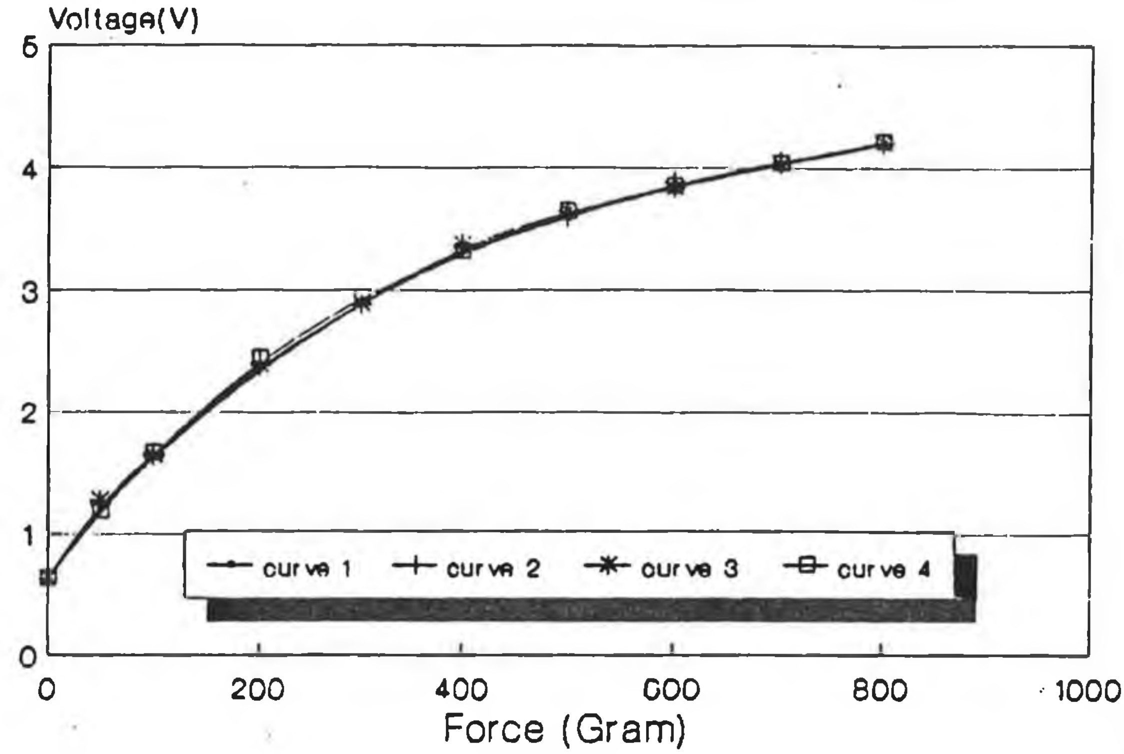


Fig.4.5 c). Calibration curve for FSR on Palm (Hand-I)

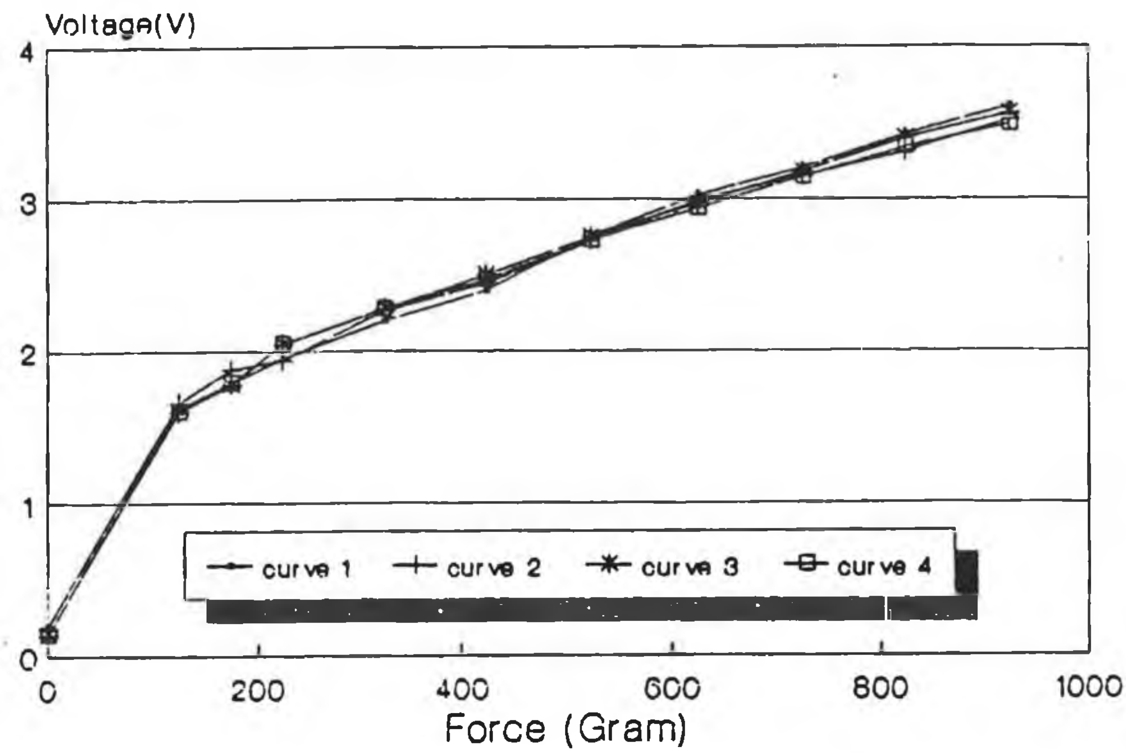


Fig.4.5 d). Calibration curve for FSR on Palm (Hand-II)

4-3-3 Load Cell amplifier and calibrations

To increase the force measurement accuracy and range, the load cells (Entran type ELF-500-5) have been used to construct the force sensing units on the fingertips of Hand-II. The load cells employ a fully active wheat-stone bridge consisting of semiconductor strain gages. The strain gauges are bonded to a thin circular diaphragm which is clamped along its circumference and which contains a load button in its center. Load applied to the button presents a distributed load to the diaphragm, which in turn provides bending stresses and resultant strains to which the strain gages react. This stress creates a strain proportional to the applied load which results in a bridge unbalance. With an applied voltage, this unbalance produces a mV deviation at the bridge output, which is proportional to the load acting upon the load button. More detailed information about the load cell ELF-500-5 can be found in Appendix B-4.

The output of the built-in load cell in the fingertips can reach up to 120 mV under the applied load of 10 N. To facilitate the force signal processing in the control process, the output of the load cell must be amplified by using a voltage amplifier. The voltage amplifier has been designed as shown in Fig. 4.6.

Let V_L denote the output of the load cell, and V_{out} denote the ouput of the amplifier. The relationship between V_{out} and V_L can be denoted by

$$V_{out} = (1 + R_F / R_L) * V_L \qquad (4 - 3)$$

Note that V_L is decided by

$$V_L = K * F + V_{OFF} \qquad (4 - 4)$$

Where

K -- Voltage - force coefficient

F -- Force applied to the load cell

V_{OFF} -- Offset Voltage of the load cell

From the view point of the design of the amplifier, the gain of the amplifier circuit has been chosen as 40. Hence, the maximum nominal output of the load cell built-in the fingertip after amplifying is 4.12 V. It is reasonable for the A/D conversion. Therefore, the resistances of R_F and R_L have been chosen as: R_F = 39 KΩ and R_L = 1 KΩ.

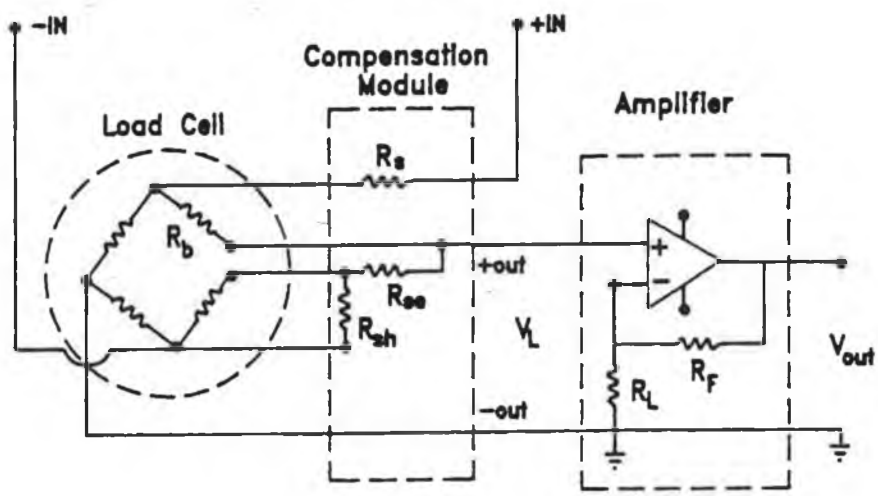


Fig. 4.6 Amplifier circuit for load cell

The calibrations for the load cells built-in the fingertips of Hand-II have been carried out by using the amplifier circuit depicted in Fig. 4.6. The calibration curves are shown in Fig. 4.7. And the calibration curve equations are listed in Appendix B-6. The scatter in Fig.4.7 a) is due to the variation of the applied force positions on the fingertip as shown in Fig 3.14.

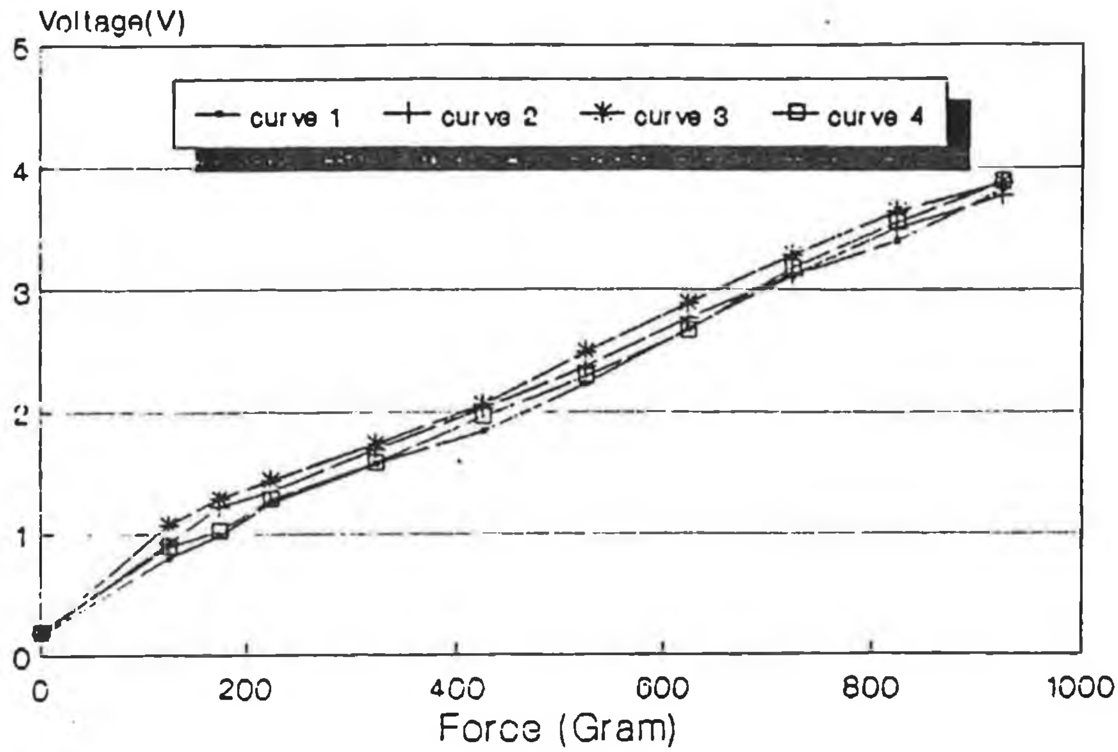


Fig. 4.7 a). Calibration for the load cell on Finger #1

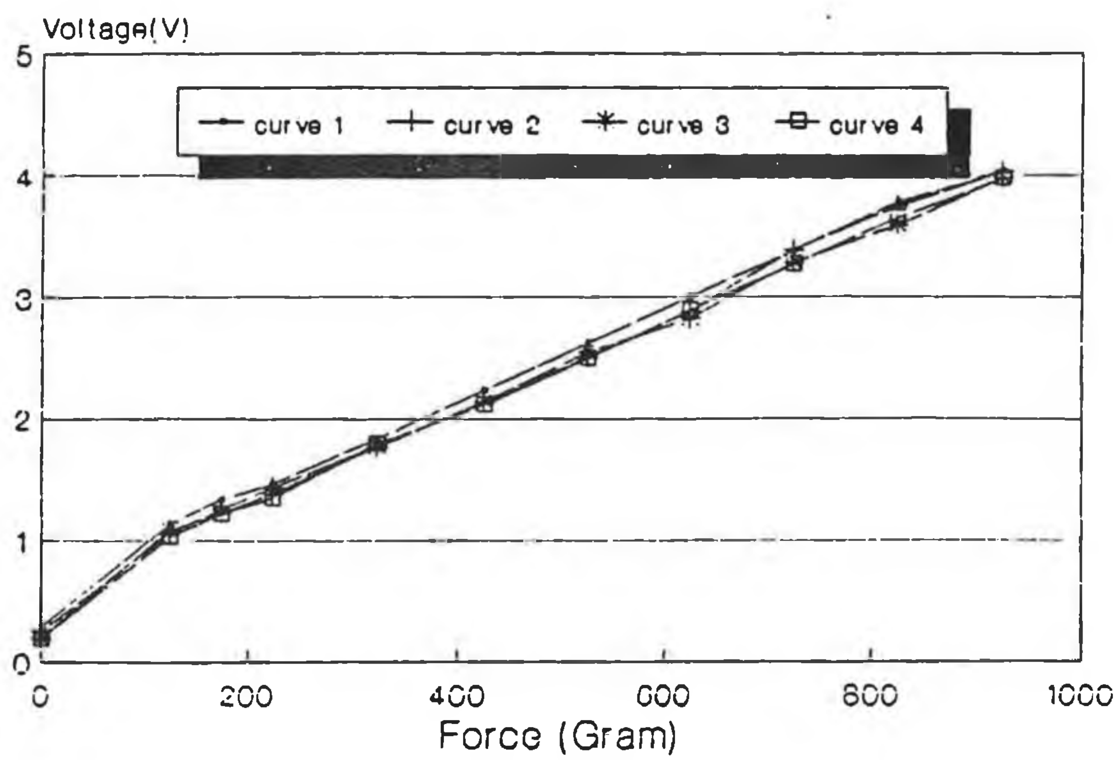


Fig. 4.7 b). Calibration for the load cell on Finger #2

4-3-4 Microswitch and sensing logic

The roller leaf sub-miniature microswitchs have been used to detect the contact (touch sensing) between the robotic hand and its environment. The electronic circuit for the touch sensing has been constructed as shown in Fig. 4.8. And the sensing logic is listed in Table 4.3.

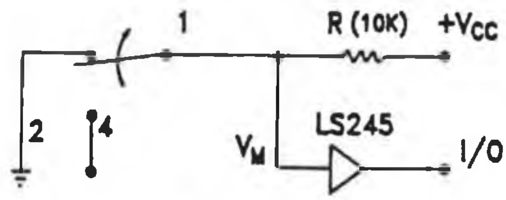


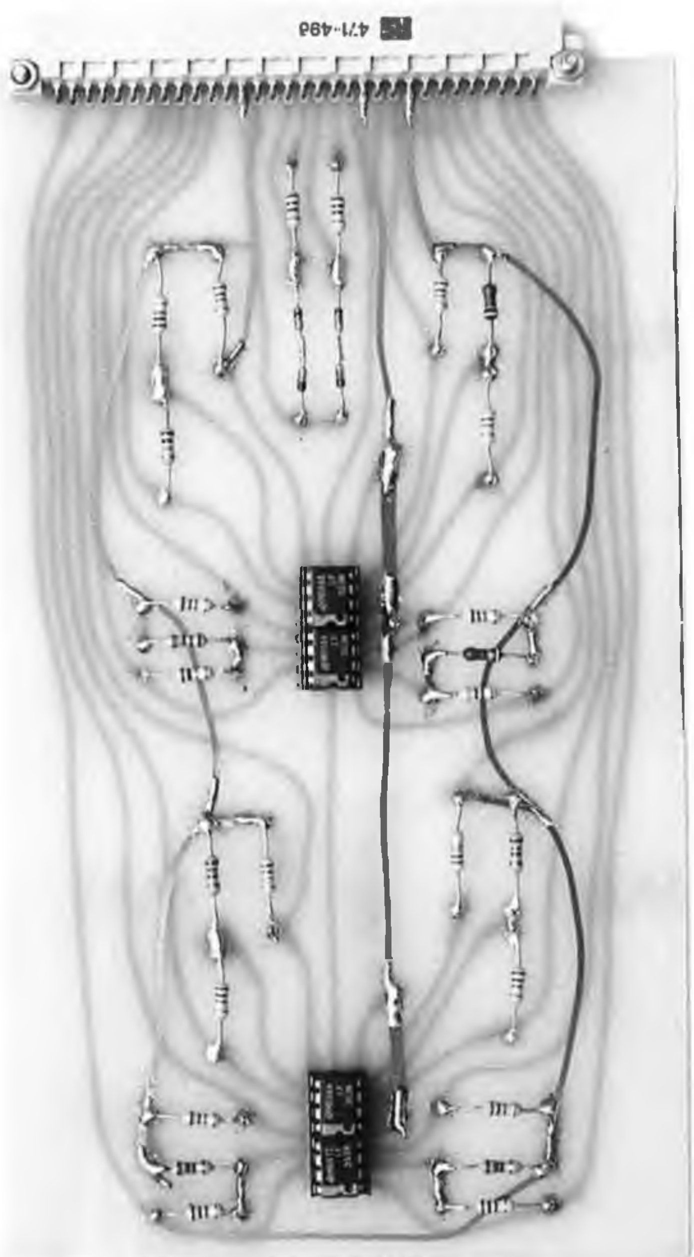
Fig. 4.8 Microswitch connections

Table 4.3 Sensing Logic

Contact	V_M (Logic voltage)	I/O Bit
YES	HIGH	1
NO	LOW	0

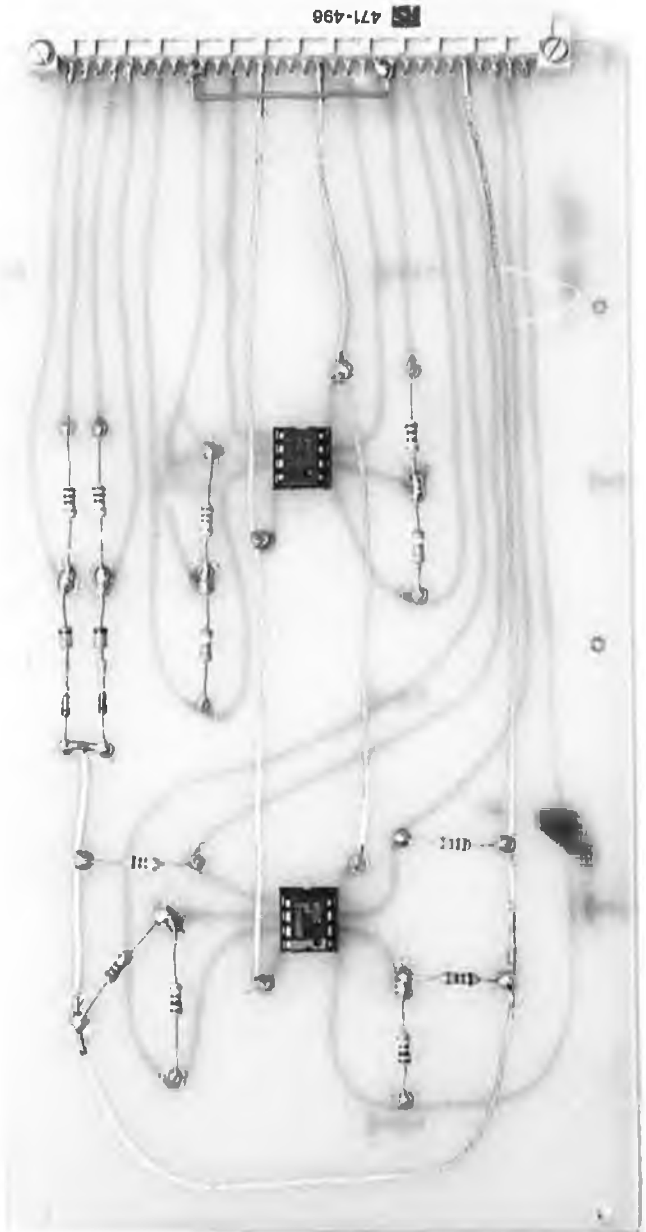
4-3-5 Sensor PCB

The layouts of the PCB of the sensor's amplifiers and protection circuits for Hand-I and Hand-II are shown in Fig. 4.9. And the detailed electronic connections of the PCB are listed in Appendix D-1.



(photo)

Fig. 4.9 a). Sensor PCB for Hand-I



(photo)

Fig. 4.9 b). Sensor PCB for Hand-II

§5-4 DC motor drive circuit design

The current amplifier has been used to drive the DC servo motor. Fig. 4.10 shows the DC motor drive circuit.

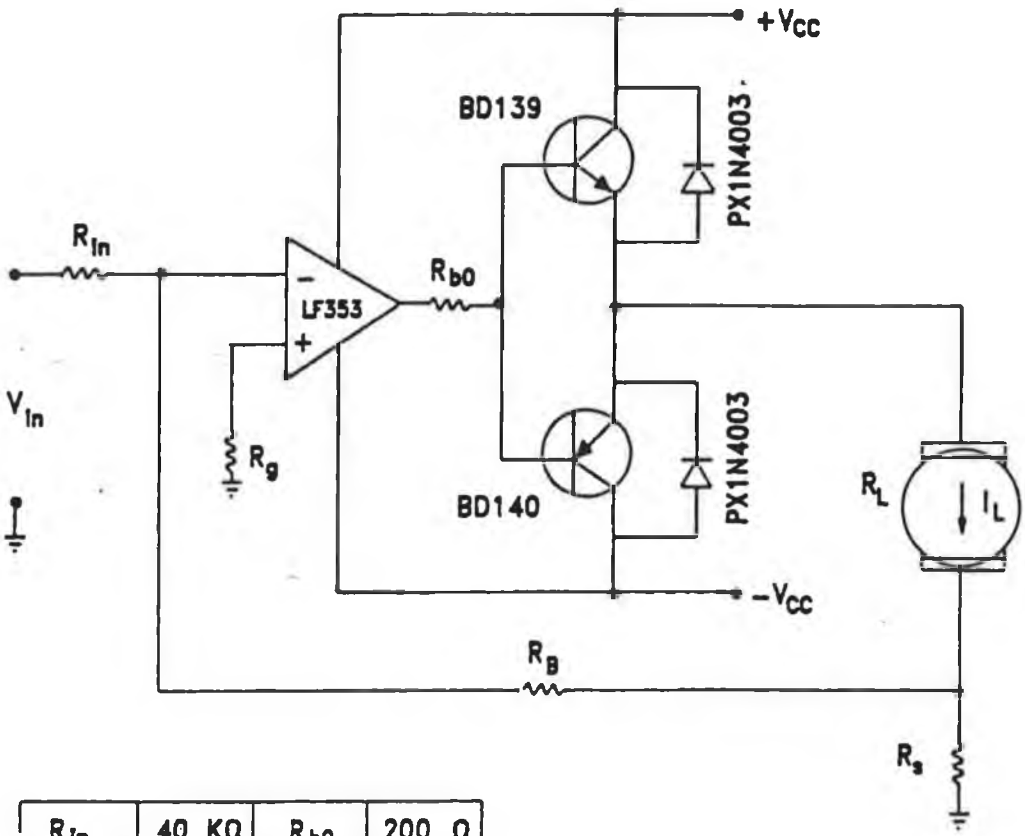
This type of amplifier is very popular when it is desired to adjust the current across the DC motors. The position/ force control of the DC motors can be achieved by using this type of amplifier.

One advantage of using such a device with a DC servo motor is the fact that the currect delivered will maintain the same regardless of changes in the motor's armture resistance which is a function of the armture temperature. In addition, the voltage drops inherent in the wiring from the amplifier to the motor will not affect the power delivered to the motor.

The diodes (PX1N4003) function as flyback protection. The inductance in the servo motor armture can produce an inductive kick when the power amplifier transistors are either suddenly all turned off or when the motor is reversed. Hence, the flyback diodes must be placed across the collect-emitter terminals of the output transistors. Otherwise, a short circuit between the collector and emitter may occur.

The current, I_L, across the motor can be directly adjusted by the input control voltage V_{in}. The relationship between them can be denoted by

$$I_L = [R_B / (R_s * R_{in})] * V_{in} \qquad (4-5)$$



R_{in}	40 KΩ	R_{b0}	200 Ω
R_B	1 KΩ	R_s	0.55 Ω
R_L	10 Ω	V_{CC}	10 V
R_g	1 KΩ		

Fig. 4.10 DC motor drive circuit

The nominal relationship between I_L and V_{in} can be inferred from Eq. (4 - 5) as :

$$I_L = (1000/22) * V_{in} \qquad (4-6)$$

However, the experimental measurement is slightly different from the nominal one:

$$I_L = K_I * V_{in} \qquad (4-7)$$

Where K_I is the current constant. And

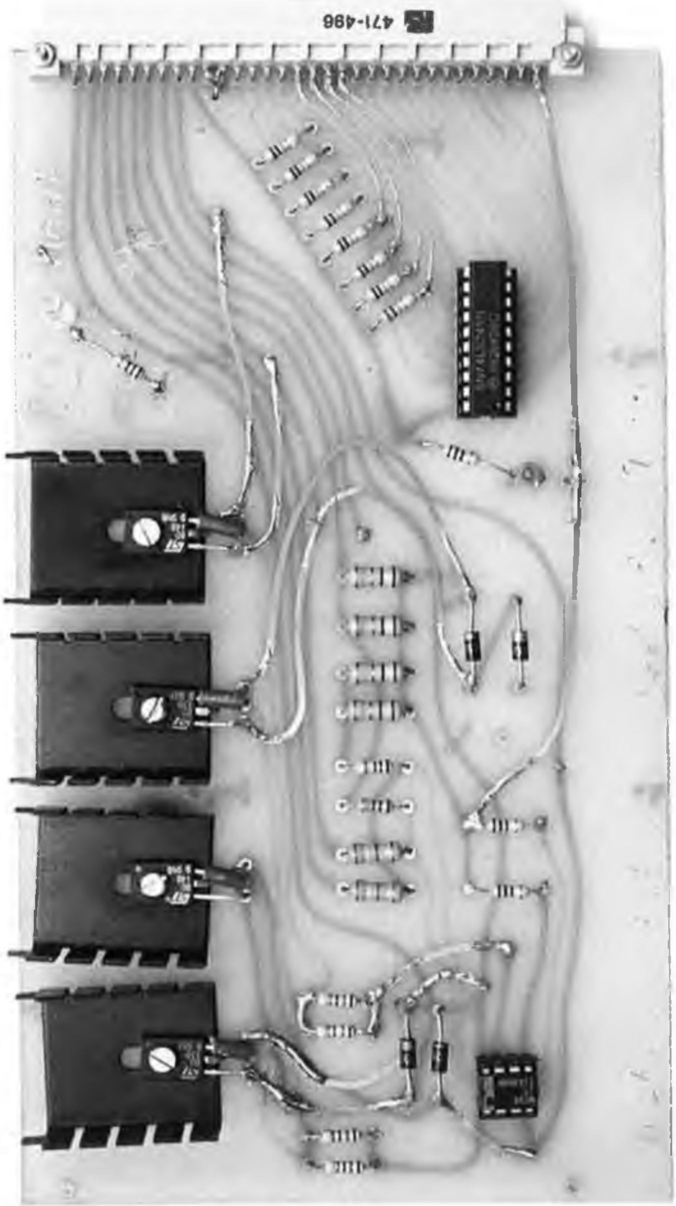
$$K_I = 1000/20.1 \quad (mA / V)$$

The PCB design for motor drive circuit is shown in Fig. 4.11. And the detailed electronic connections are listed in Appendix D-2.

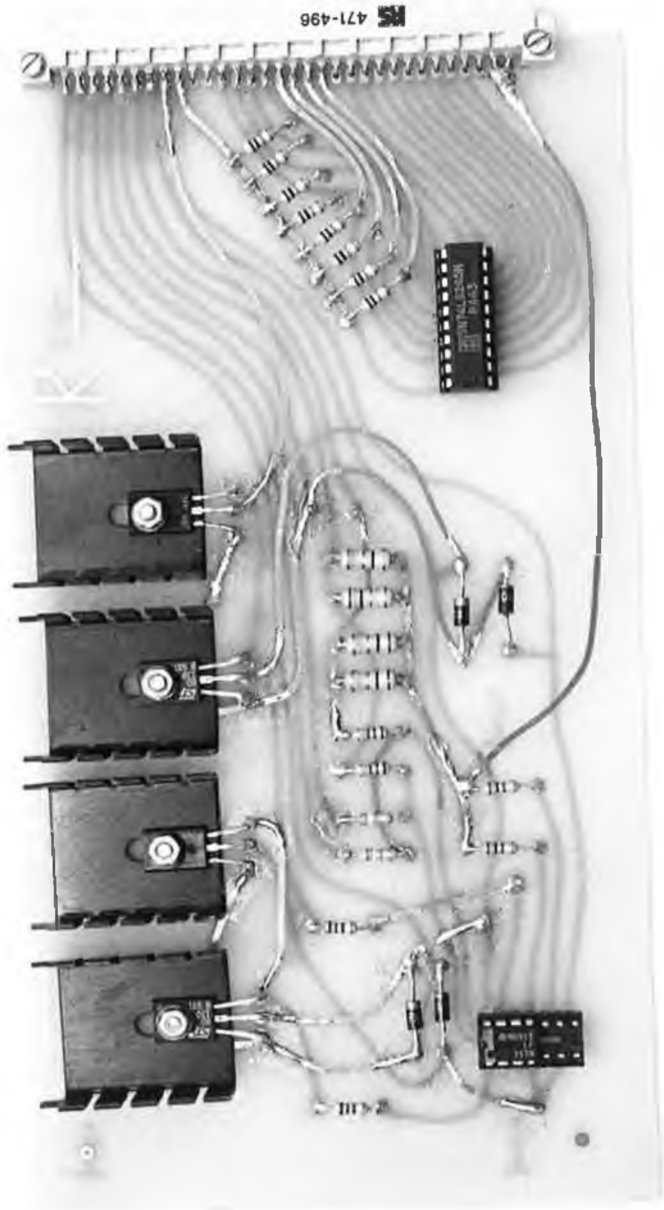
4-5 Robotic hand interfacing with the PC

To facilitate the signals distribution and interfacing among the motor/PCB, the sensor/PCB, the robotic hand, the A/D cable, the D/A cable, the signal measurement cable and the power supply, a interfacing PCB has been designed for Hand-II to handle the signals distribution and interfacing.

Fig. 4.12 shows the layout of the interfacing PCB. And the detailed electronic connections are listed in Appendix D-3.



(photo)
Fig. 4.11 a). DC motor PCB for Hand-I



(photo)

Fig. 4.11 b). DC motor PCB for Hand-II

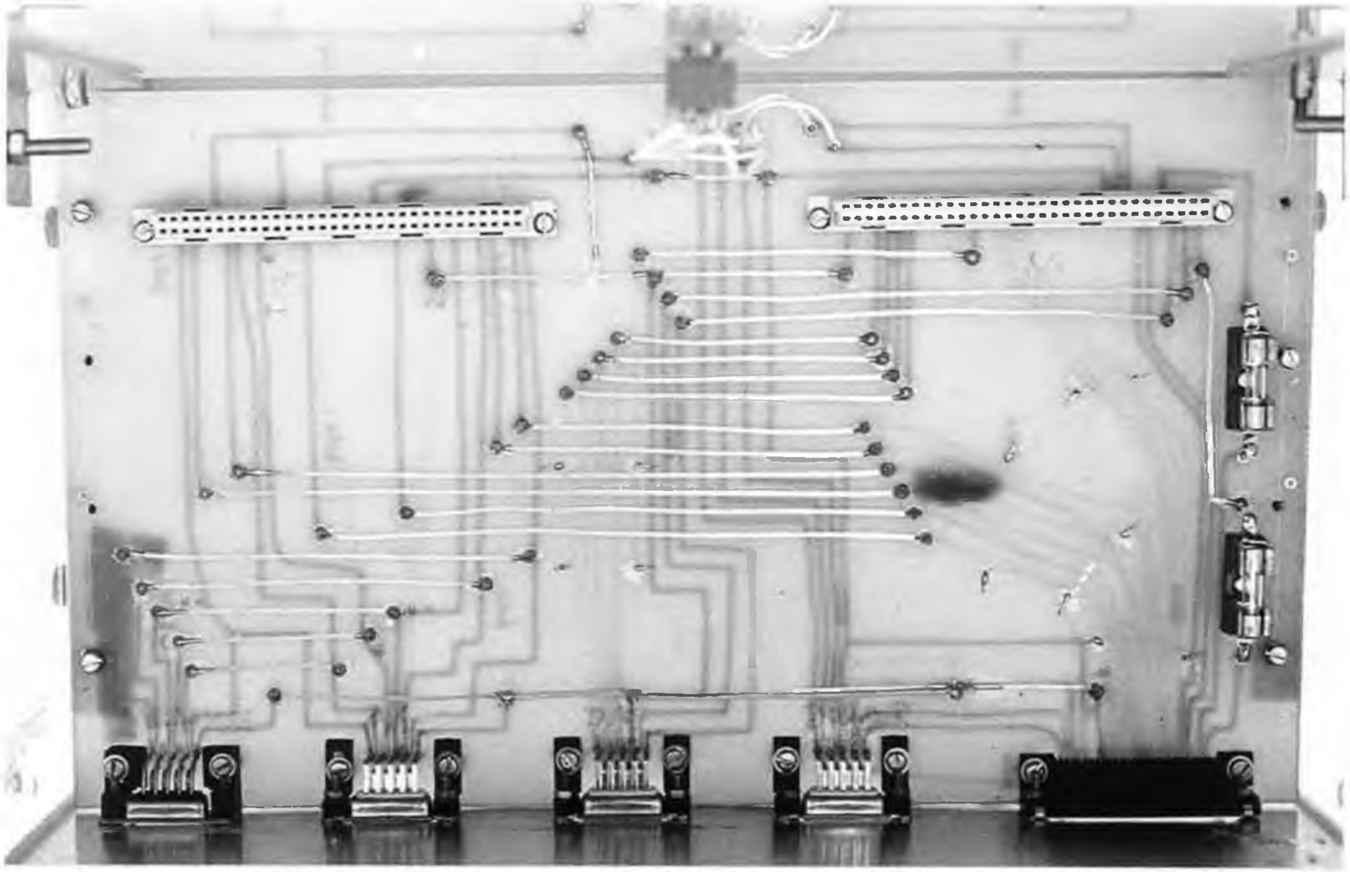


Fig. 4.12 Interfacing PCB

4-6 position servo control of robotic fingers

The position servo control architecture has been proposed in Fig. 4.2. The current amplifier has been used to supply a current proportional to its input control voltage. Since the torque generated by the DC motor is proportional to the supplied current, the control over a DC motor using a current amplifier is also termed as torque control approach.

An important advantage of the torque control approach is that a desired force or torque can be maintained. Another advantage is that no additional power will be drawn from the electrical source even when the fingers encounter resistance during position servo control. Thus, the safety of the human body can be ensured.

The basic ideas embedded in the closed-loop control, which will be discussed later, may be generalized as follows:

a. If the error is large and the velocity is small, apply a large drive signal

b. If the error is small and the velocity is high, apply a negative drive signal

c. If the error is within the required limit, apply a lock signal to stop the motor being controlled.

4-6-1 Plant modelling

The robotic fingers are driven by two DC-motors with built-in gearbox. Since the maximum continuous torque permitted by the gearbox is 0.6 Nm, it does not meet the massaging force requirement. Hence, a set of pulley-timing belt system has been used to increase the torque supplied by the motor (The ratio is 1.6 for Hand-I). Due to the similarity of the drive systems of the robotic fingers, only one drive system for one finger will be discussed. A drive system for one finger can be illustrated in Fig. 4.13.

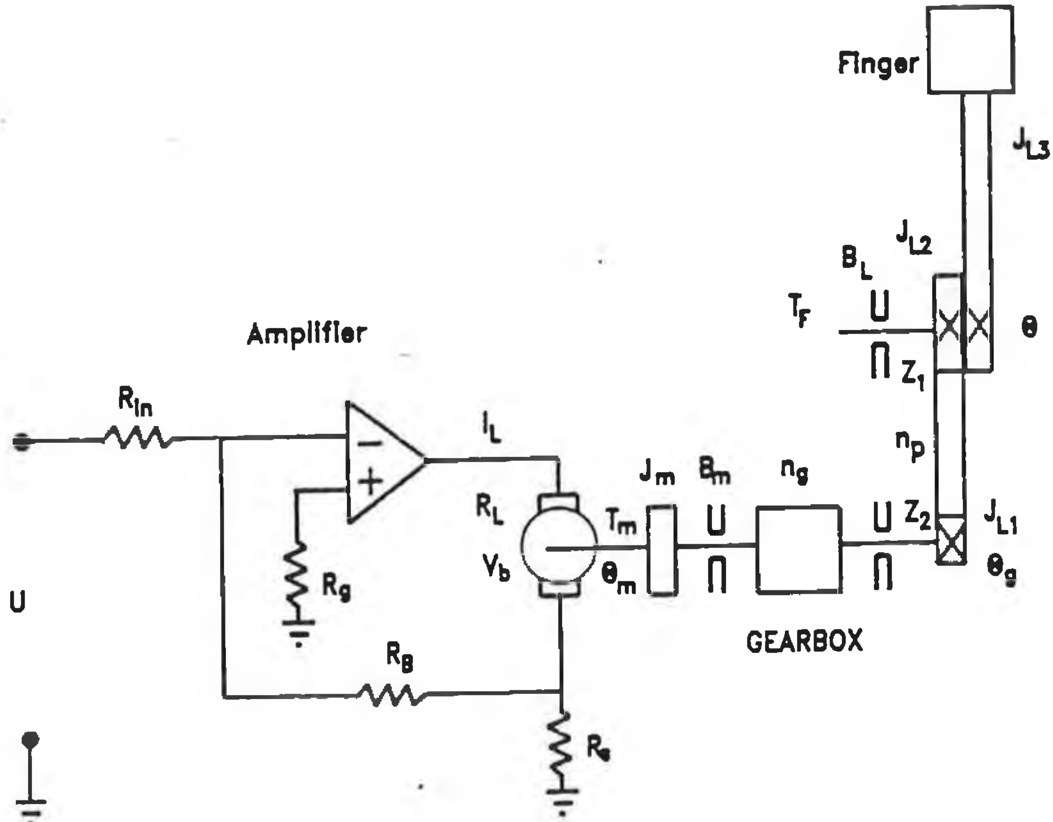


Fig. 4.13 DC motor drive system for robotic fingers

Where

I_L	--	DC motor armature current
R_L	--	DC motor armature winding resistance (10 KΩ)
V_b	--	Back emf voltage
U	--	Control voltage from D/A (V)
Θ_m	--	DC motor rotational angle
Θ_g	--	Gearbox rotational angle at output shaft
Θ	--	Robotic finger rotational angle
B_m	--	DC motor friction constant
B_L	--	Robotic finger shaft friction constant
J_m	--	Inertia of the DC motor
J_{L1}	--	Inertia of the pulley 1 on the gearbox shaft
J_{L2}	--	Inertia of the pulley 2 on the finger shaft
J_{L3}	--	Inertia of the finger on the finger shaft
Z_1	--	Tooth number of the pulley 1
Z_2	--	Tooth number of the pulley 2
T_F	--	Load torque on the finger rotational shaft
T_m	--	Drive torque on the DC motor shaft
n_g	--	Reduction of the DC motor gearbox
n_p	--	Reduction of the pulley-timing belt system

Mechanical Characteristics of the drive system

A. Reductions

The reduction of the gearbox is defined as:

$$n_g = \Theta_g / \Theta_m \qquad (4 - 8)$$

And

$$n_g = 1/100$$

The reduction of the pulley-timing belt system is defined as:

$$n_p = \Theta / \Theta_g \qquad (4 - 9)$$

And

$n_p = 1/1.6$ for Hand-I

$n_p = 1.1/1.6$ for Hand-II

Thus the relationship between the robotic finger rotational angle and the DC motor rotational angle is denoted by

$$\Theta = (n_p n_g)\ \Theta_m \qquad (4 - 10)$$

B. Equivalent load inertia (J)

The equivalent load inertia, J, at the DC motor shaft is denoted by:

$$J = J_m + (n_g)^2 J_{L1} + (n_p n_g)^2 (J_{L2} + J_{L3}) \qquad (4 - 11)$$

And

$J = 2.325 \times 10^{-6}$ (Kgm^2) for Hand-I

$J = 2.332 \times 10^{-6}$ (Kgm^2) for Hand-II

C. Equivalent load friction constant (B)

The equivalent load friction, B, at the DC motor shaft is denoted by:

$$B = B_m + (n_p n_g)^2 B_L \qquad (4 - 12)$$

And

$$B = 0 \qquad \text{for Hand-I and Hand-II}$$

D. Motion equation for the drive system

The motion equation for the drive system can be given by

$$T_m - (n_p n_g) T_F = J\ddot{\theta}_m + B\dot{\theta}_m \qquad (4 - 13)$$

From eq.(4 - 13), two cases can be derived:

Case I -- when the finger moves in the free space
In this case, $T_F = 0$. This is the situation of position control.

Case II -- When the finger applies a force onto a part
In this case, $T_F > 0$. And the angle speed and the angle acceleration are very low. This is the situation of force control.

Electrical characteristics of the plant

For a given current, the output torque of the DC motor may be denoted by:

$$T_m = K_T I_L \qquad (4 - 14)$$

Where K_T is the DC motor output torque constant. And

$$K_T = 22.68 \times 10^{-6} \qquad (\text{Nm} / \text{mA})$$

Considering the allowable working range of the gearbox, the maximum permissible DC motor armature current is:

$$I_{Lmax} = 264.55 \quad (\text{ mA })$$

Combining eq.(4 - 14) with eq.(4 - 7), one may obtain

$$T_m = K_T K_I U \qquad (4 - 15)$$

<u>Modelling of the plant</u>

From eq.(4 - 10), eq.(4 - 13) and eq.(4 - 15), the Laplace transfer function for the finger position and the control voltage can be obtained:

$$\frac{\Theta(s)}{U(s)} = \frac{K_I K_T (n_p n_g)}{s(Js + B)} \qquad (4 - 16)$$

A proportional-derivative (PD) controller has been used to generate the control voltage U. The position servo loop for the fingers can be illustrated in Fig. 4.14.

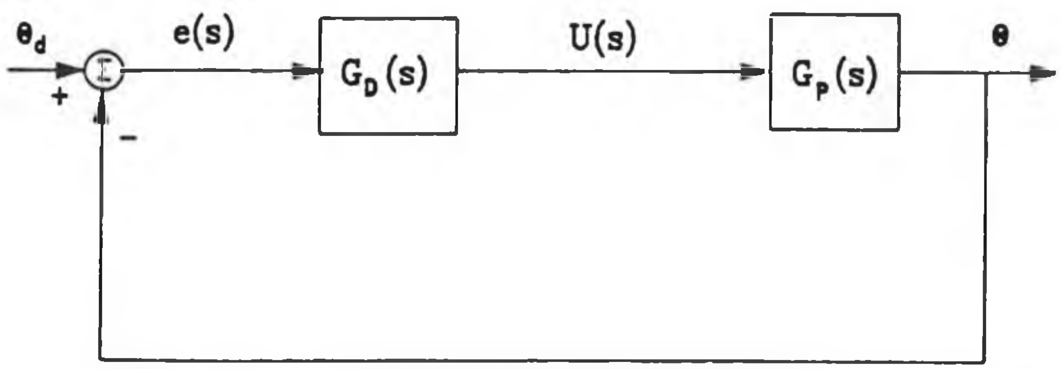


Fig. 4.14 Position servo loop for fingers

Where

$$G_p = \frac{K_0}{s(s + a)} \qquad (4 - 17)$$

$$K_0 = K_I K_T n_p n_g / J \qquad (4 - 18)$$

$$a = B / J \qquad (4 - 19)$$

Furthermore,

$K_0 = 3.048$ for Hand-I
$K_0 = 3.343$ for Hand-II
$a = 0$

4-6-2 Digital controller design

The design of the digital controller is carried out by using the well-developed analogue design techniques. The analogue controller designed is transformed into the discrete form to obtain the digital controller.

Referring to Fig. 4.14, the analogue PD controller in the position servo loop can be expressed by :

$$G_D(s) = K_p + K_d s \qquad (4 - 20)$$

Hence, the Laplace transfer function of the servo loop may be denoted by

$$\frac{\Theta(s)}{\Theta_d(s)} = \frac{K_0 K_p + K_0 K_d s}{s^2 + K_0 K_d s + K_0 K_p} \qquad (4 - 21)$$

The step input of $\theta_d(s)$ is expressed as:

$$\theta_d(s) = \theta_d/s \qquad (4-22)$$

In the process of analysing the characteristics of the position servo loop, two parameters, ω_n and ζ, are often used [138-140]. Where, ω_n is referred as natural undamped freqency and ζ is referred as damping ratio. And

$$\omega_n = (K_0K_p)^{\frac{1}{2}} \qquad (4-23)$$

$$\zeta = 2K_d(K_0/K_p)^{\frac{1}{2}} \qquad (4-24)$$

The response equations of the position servo loop to a desired position step input are given in Table 4.4.

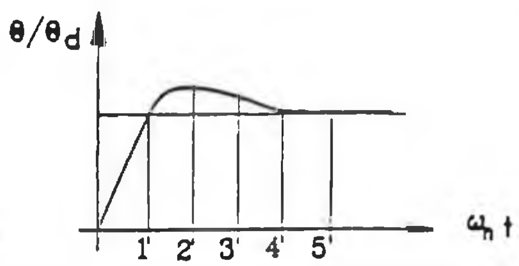
<u>Table 4.4 Servo loop response</u>

ζ	Servo loop response
$0<\zeta<1$	$\theta(t) = \theta_d\{1 + \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\mathrm{Sin}[\sqrt{1-\zeta^2}\,\omega_n t - \psi]\}$ $\psi = \mathrm{Tan}^{-1}\frac{\sqrt{1-\zeta^2}}{\zeta}$
$\zeta = 1$	$\theta(t) = \theta_d\{1 - (1 - \omega_n t)e^{-\omega_n t}\}$
$\zeta > 1$	$\theta(t) = \theta_d\{1 - \frac{1}{B_1 - B_2}[B_1 e^{-B_1 t} - B_2 e^{-B_2 t}]\}$ $B_{1,2} = (\zeta \pm \sqrt{\zeta^2 - 1})\omega_n$

For the position servo control of the fingers, the damp ratio has been selected as $\zeta = 1$.

Table 4.5 gives out the transient response performance for the servo loop under the condition of $\zeta = 1$.

<u>Table 4.5 System transient performance</u>

Performance	Expressions
Rise Time t_r	$t_r = 1/\omega_n$
Max. Overshoot M_p	$M_p = 13.5\%$ $t_p = 2/\omega_n$
Settling time t_s	$t_s = 5/\omega_n$ For $e_{ss} < 2.7\%$
Diagram Illustration	

Thus, for a system settling time t_s, a set of PD controller parameters, K_p and K_d, can be designed under certain steady-state error e_{ss}.

For $e_{ss} \leq 2.7\%$, the K_p and the K_d may be denoted by:

$$K_p = (1/K_0)*(25/t_s^2) \qquad (4-25)$$

$$K_d = (1/K_0)*(10/t_s) \qquad (4-26)$$

For different t_s , the K_p and the K_d can be obtained by using eq.(4-25) and eq.(4-26). Table 4.6 shows two groups of designed parameters for the PD controller.

Table 4.6 K_p and K_d design

t_s	PD	Hand-I	Hand-II
1 sec	K_p	8.20	7.48
	K_d	3.28	3.00
2 sec	K_p	2.05	1.87
	K_d	1.64	1.50

The designed analogue PD controller can be converted into a discrete PD algorithm by using the approximation techniques [141-143]. Here, the PD controller can be denoted by:

$$G_D = \frac{K_1 + K_2 z^{-1}}{1 + z^{-1}} \qquad (4 - 27)$$

Where

$$K_1 = K_p + (2/T)K_d \qquad (4 - 28)$$

$$K_2 = K_p - (2/T)K_d \qquad (4 - 29)$$

Thus, the discrete PD algorithm may be expressed as:

$$U(n)=U(n-2)+K_1e(n)+(K_2-K_1)e(n-1)-K_2e(n-2) \qquad (4 - 30)$$

And the digital servo control system for the finger position control is in a form shown in Fig. 4.15.

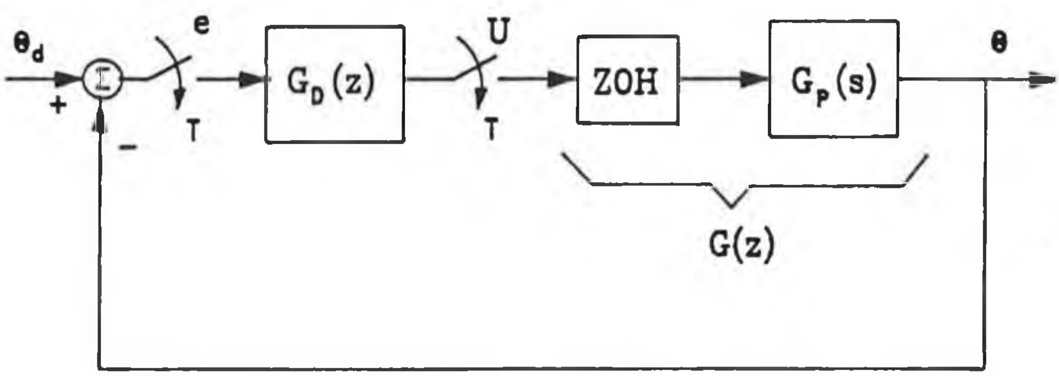


Fig. 4.15 Digital servo control system

Where

$$G(z) = \frac{K_0 T^2 (1+z^{-1}) z^{-1}}{2(1-z^{-1})^2} \qquad (4-31)$$

Hence, the transfer function of the digital servo control system can be obtained:

$$\frac{\Theta(z)}{\Theta_d(z)} = \frac{G_D(z)G(z)}{1 + G_D(z)G(z)}$$

$$= \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \qquad (4-32)$$

Where

$$\begin{aligned} a_1 &= 2 - T^2 K_0 K_1/2 \\ a_2 &= -1 - T^2 K_0 K_2/2 \\ b_1 &= T^2 K_0 K_1/2 \\ b_2 &= T^2 K_0 K_2/2 \end{aligned} \qquad (4-33)$$

Therefore, the system response of the digital control system can be denoted by:

$$\Theta(n) = a_1\Theta(n-1) + a_2\Theta(n-2) + b_1\Theta_d(n-1) + b_2\Theta_d(n-2) \qquad (4-34)$$

Using the designed digital controller , the position servo control over the robotic fingers has been carried out. Fig. 4.16 shows some of the experimental results under the sampling rate of T=0.01 Sec.

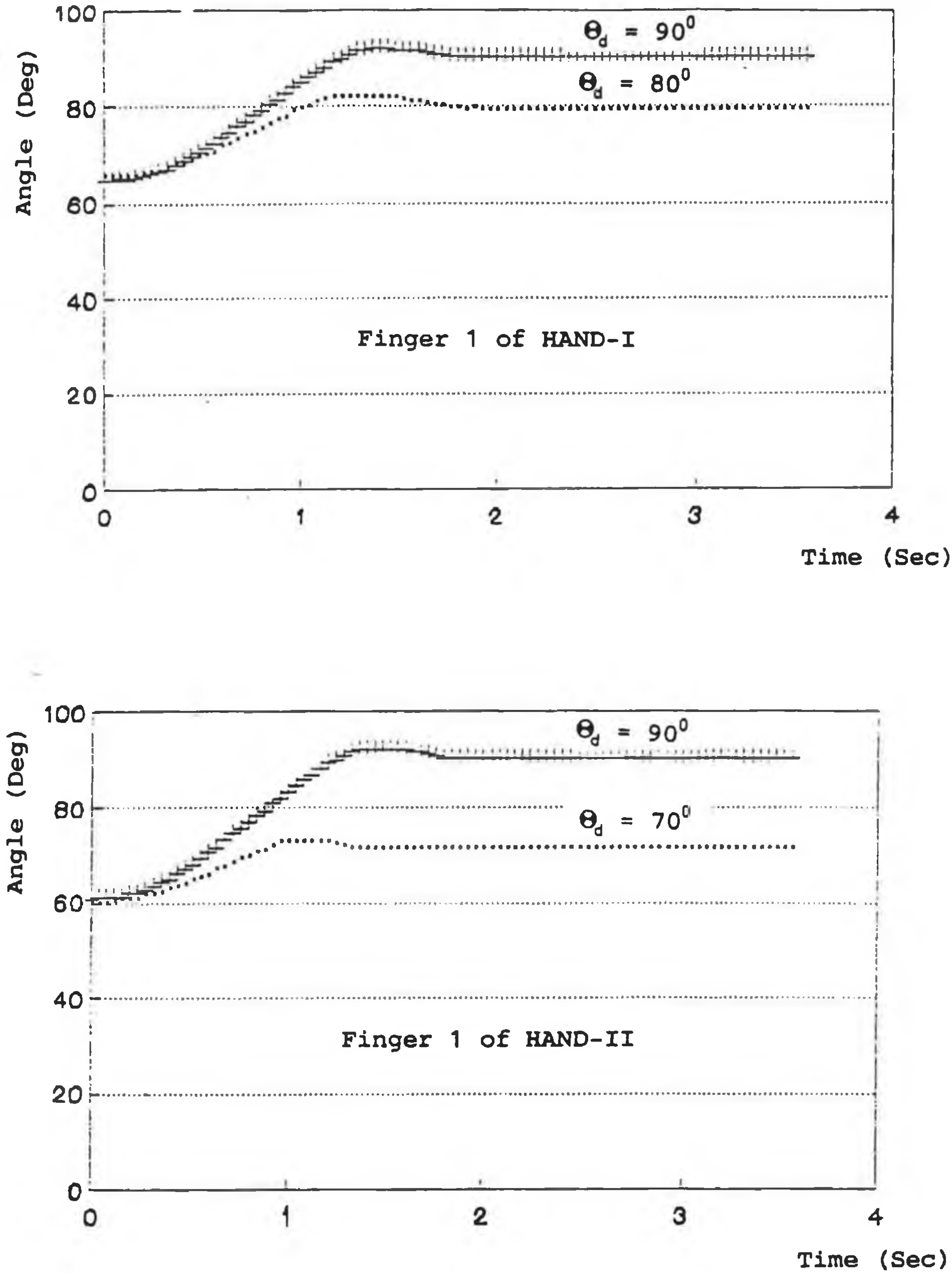


Fig. 4.16 Position servo control results

4-7 Force control of the end-effectors

There are two types of forces applied by the robotic end-effectors: the kneading force applied by the finger-tips and the padding force applied by the palm.

The control of the kneading force is realized by regulating the current across the DC motors. While control of the padding force is realized by controlling the motion of the robotic arm.

4-7-1 Kneading force control

From eqs.(4-10), (4-13) and (4-14), the torque, $T_F(s)$, applied by the DC motor on the finger can be derived as:

$$T_F(s)=(K_T K_I/n_p n_g)U(s)-[s(Js+B)/(n_p n_g)^2]\Theta(s) \qquad (4-35)$$

Since the kneading force can only be produced when the robotic fingers contact the part being massaged, the angular speed and acceleration of the finger are very small at this stage. Hence, the force generated due to angular speed and acceleration can be neglected in this case. Thus, the kneading torque applied by one finger can be controlled by a linear equation:

$$T_F = K_F U \qquad (4-36)$$

Where

$$K_F = K_T K_I/n_p n_g \qquad (4-37)$$

The force on the fingertip can be denoted by

$$F = T_F / L_F \qquad (4-38)$$

Where, L_F is the distance from the force exerting point on the fingertip to the finger rotational center.

Hence, the force control equation can be generalized from eq.(4-36) and eq.(4-38):

$$K_F U = F L_F \qquad (4-39)$$

For a given kneading force, F, on one fingertip, the required input control voltage, U, can be decided by using eq.(4-39), that is:

$$U = FL_F/K_F \qquad (4-40)$$

For a given control voltage, U, the sensitivity of the force control equation is denoted by:

$$\frac{dL}{L_F} = -\frac{dF}{F} \qquad (4-41)$$

To ensure the desired force to be applied onto the part being massaged, the contact between the fingertips and the part must be maintained. In most situations, the forces applied by both fingertips are required to be the same. Hence, the part being massaged is required to be centralized in the robotic hand frame, so that the force exerting distance, L_F, can be kept the same. The centralizing process of the part can be performed by detecting the forces and the positions of the fingertips. Any decentralization of the part out of the limitations should be corrected. Once the part has been centralized, the next step is to check the contact situation. The fingertips must make full contact with the part because only the forces on the fingertips can be reported by the force sensors. Using the forces and positions of the fingertips the contact situation can also be assessed. And a certain correction strategy can be formulated. The more detailed error-correction will be discussed in chapter 6. Here it is assumed that the contact between the fingertips and the part being massaged is perfect and that the part has been centralized.

The parameters used in the force control process are shown in Table 4.7.

<u>Table 4.7</u> K_F and L_F

Items	Hand-I	Hand-II
K_F	0.18144	0.16495
L_F (m)	0.115	0.092

Fig. 4.17 illustrates the force control scheme for the robotic fingers. One may notice that the finger position control is also involved in a force control process.

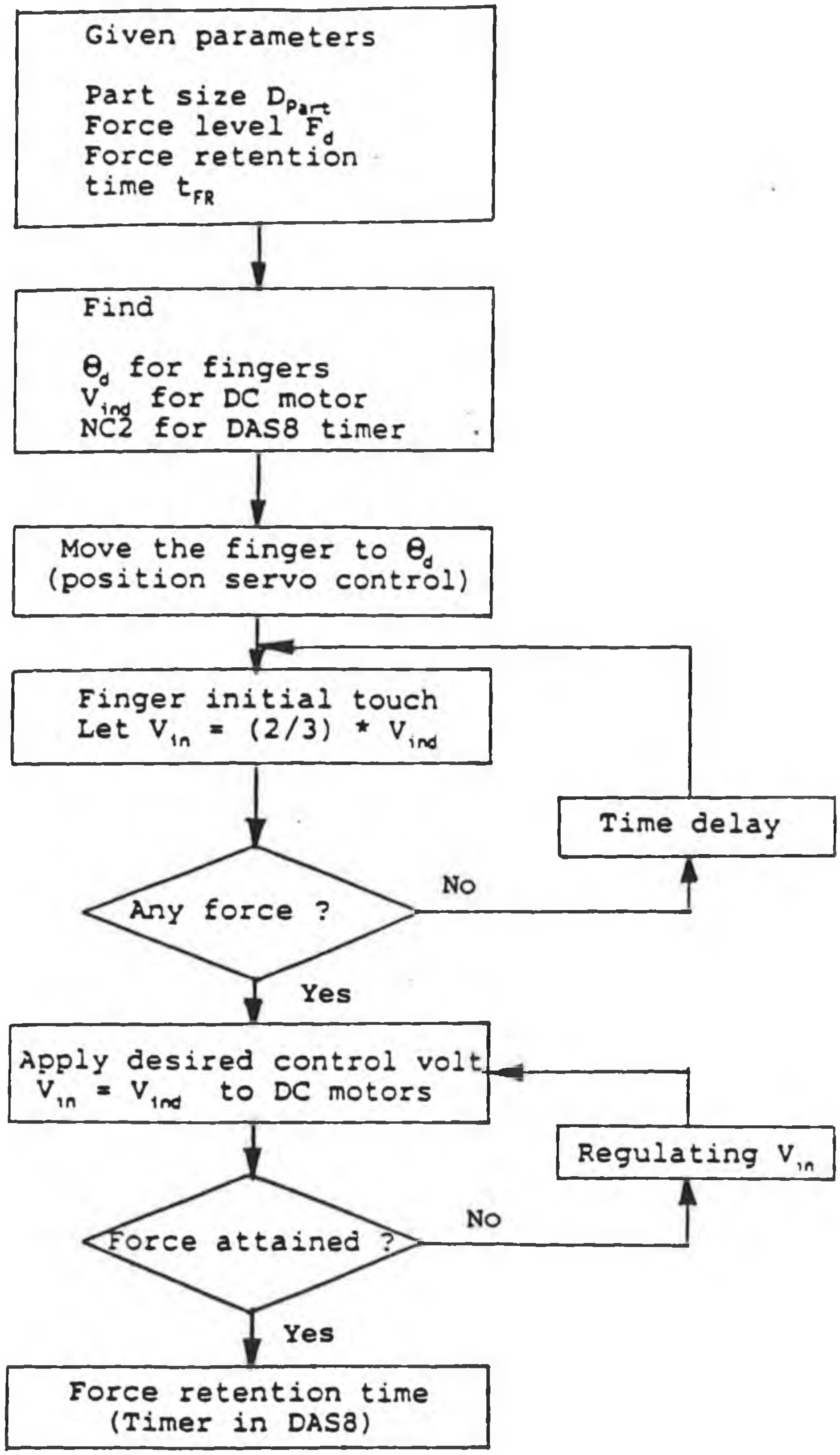


Fig. 4.17 Flow Chart for force control scheme

By sending different control voltages to the DC motors, different force levels of the robotic fingertips can be achieved. Fig. 4.18 shows the kneading force levels of the fingertip 2 in HAND-II. Where the fingertips of HAND-II are commanded to knead a soft rubber ball with the diameter of 45 mm.

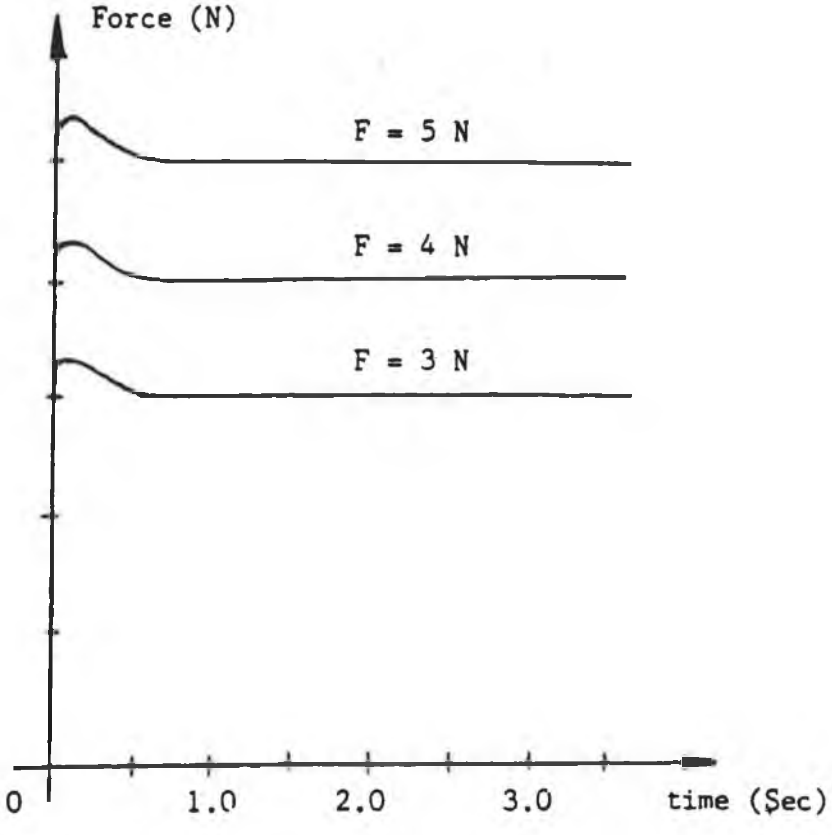


Fig. 4.18 Kneading force levels (HAND-II)

4-7-2 Padding force control

The padding force control can be achieved by controlling the fine motion of the robotic palm, which moves against the part being massaged.

A general force control equation has been embedded in the palm fine motion control algorithm, which is denoted by

$$[F] = [K]\ [\delta X] \qquad (4-42)$$

Where

- $[F]$ -- the padding force vector
- $[K]$ -- the stiffness matrix of the part
- $[\delta X]$ -- the palm fine motion vector

Since the stiffness of the part being massaged varies from one person to another, the stiffness matrix is not easy to be formulated. Hence, a trial and error method has been implemented to obtain the desired padding force by regulating the fine motion of the palm against the part being massaged. The force feedback of the palm gives the contact situation between the palm and the part. And the position feedback of the palm provides the palm motion status. If the motion covers a long range, a quick approaching distance must be formulated to speed up the force control process (The fuzzy inference has been used in Chapter 6). Once the initial contact is detected, a fine motion control of the palm must be initiated, while the force should be assessed in every motion cycle. The speed of the palm motion can be adjusted by setting the robot arm speed and the force retention time.

Where Fig. 4.19 shows the padding force control scheme using the trial and error method. And Fig. 4.20 shows an experimental result.

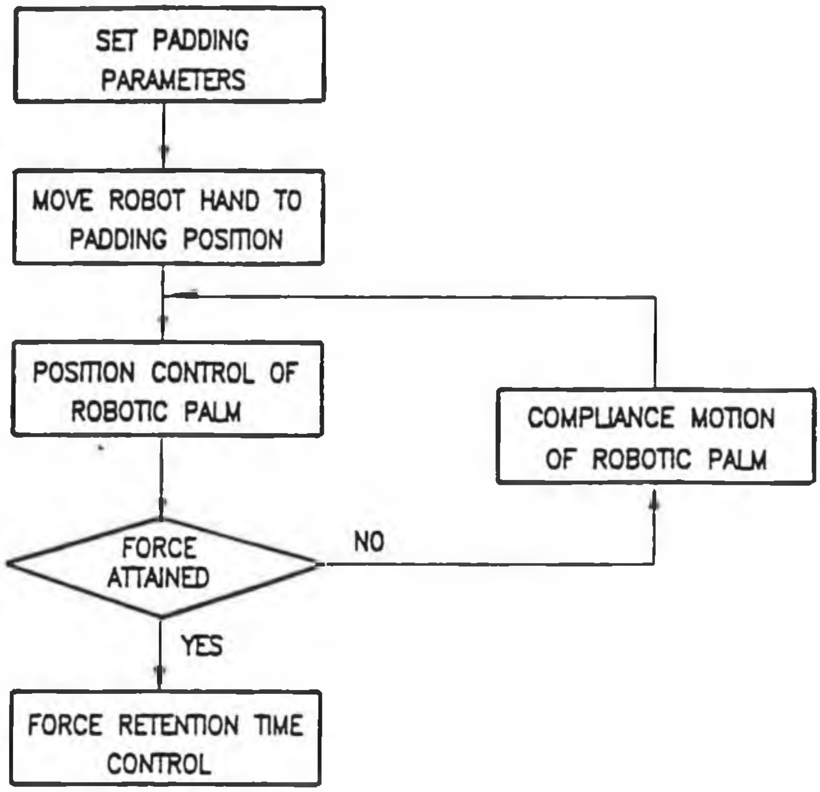


Fig. 4.19 Padding force control scheme

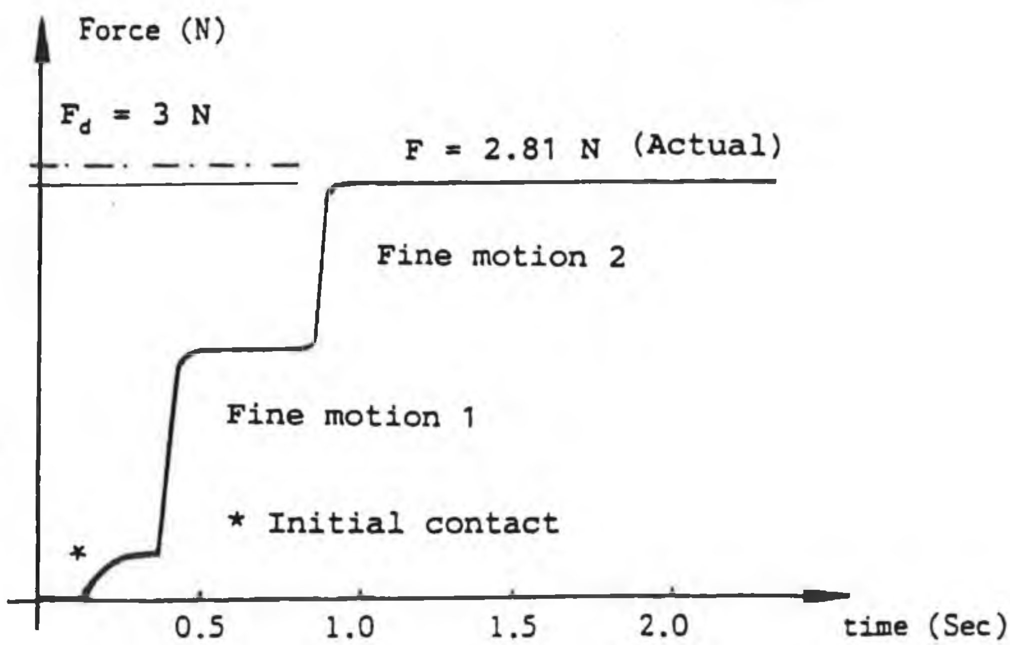


Fig. 4.20 Padding force and compliance motion

Chapter Five

Robotic kinematics and path design

5-1 Introduction

This chapter is mainly concerned with the geometry motion of the robotic arm with respect to a fixed reference coordinate system (robot base system). The geometry motion of the robot is a function of time without regard to the forces/torques that cause the motion. Thus, the spatial configuration of the robot as a function of time, in particular the relationship between the joint-variable space and the position/orientation of the end-effector of the robot arm will be studied. This is usually referred to as the kinematics of robots. The robot kinematics usually consists of two subproblems [144]:

a. direct kinematics
b. inverse kinematics

The direct kinematics problem is to find the position and orientation of the end-effector of a robotic manipulator with respect to a reference coordinate system, given the joint angle vector $\Theta = (\Theta_1, \Theta_2, \ldots \Theta_n)^T$ of the robot arm.

The inverse kinematics problem is to calculate the joint angle vector Θ given the position and orientation of the end-effector with respect to the reference coordinate system.

Computer-based robots are usually servoed in the joint space while objects to be manipulated are usually expressed in the Cartesian space. In order to control the position/orientation of the end-effector of the robot arm as it follows a predefined path, the inverse kinematics solutions are required.

Since the link of a robot arm may rotate with respect to a reference coordinate frame, the total spatial displacement of the end-effector is a result of the angular rotations of the links. The Denavit and Hartenberg (D-H) method [145] has been used to describe the spatial relationship between two adjacent rigid mechanical links. And the direct and inverse kinematics of the Mitsubishi robot arm with five DOF are analysed in 5-2.

The coordinate frames attached to the specially designed robotic hand are defined in 5-3. The path design for a given part to be massaged in the Cartesian space is described in 5-4. While the motion control of the robot arm is outlined in 5-5.

5-2 Kinematics of the robot arm

The physical construction of the robot arm has been shown in Fig. 3.2. A reference frame $O_0X_0Y_0Z_0$, which is usually called world frame, has been attached at the robot base as shown in Fig. 3.2. The world frame is the reference frame for position control and feedback of the robot arm. It is also the reference frame in which the position and the orientation of the part to be massaged are defined.

To study the kinematics of the robot arm, it is assumed that an ideal robotic hand with a fixed grasping center has been mounted onto the wrist mounting surface. A hand frame can be attached on the grasping center of the ideal hand. The hand frame is denoted by $O_5X_5Y_5Z_5$. In the hand frame, a set of orientation vectors (n o a) can be always defined by using the right-hand rule [144].

5-2-1 Direct kinematics

To describe the translational and rotational relationships between adjacent links, Denavit and Hartenberg [145] proposed a matrix method of establishing a coordinate system to each link of an articulated chain. The D-H representation results in a 4x4 homogeneous transformation matrix representing each link's coordinate system at the joint with respect to the previous link's coordinate system. Thus, through sequential transformations, the position and orientation of the end-effector can be expressed in the world frame.

By using the D-H representation, every coordinate frame can be determined and established on the basis of three rules:

a. The Z_{i-1} axis lies along the motion axis of the ith joint

b. The X_i axis is normal to the Z axis, pointing away from it

c. The Y_i axis completes the right hand coordinate system ($X_iY_iZ_i$)

By applying these rules, the coordinate system of the robot arm can be established as shown in Fig. 5.1. Where

$O_0X_0Y_0Z_0$ is the world frame
$O_1X_1Y_1Z_1$ is attached to the shoulder frame
$O_2X_2Y_2Z_2$ is attached to the elbow frame
$O_3X_3Y_3Z_3$ is attached to the wrist pitch frame
$O_4X_4Y_4Z_4$ is attached to the wrist roll frame
$O_5X_5Y_5Z_5$ is attached to the robot hand frame

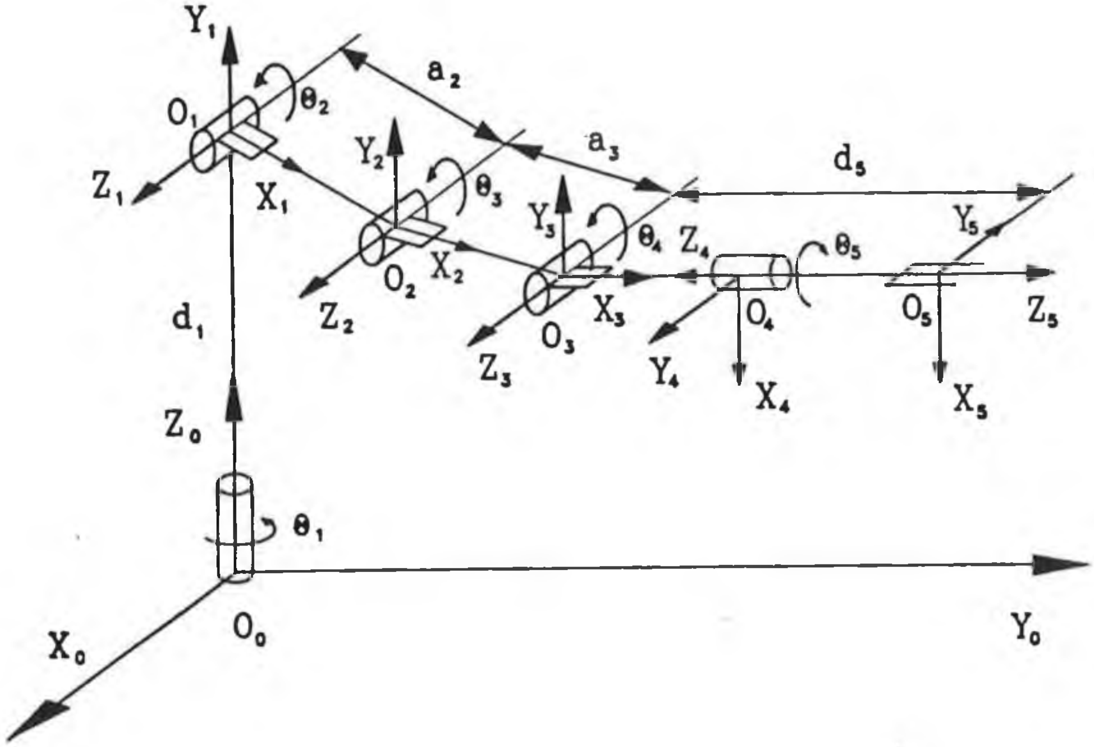


Fig. 5.1 Robot coordinate frames

Once the D-H coordinate system for each link is established, the homogeneous transformation matrix can be developed. Thus, the complete transformation of joint i with respect to joint i-1 can be given by:

$$A_{i-1}^{i} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5 - 1)$$

Where

$$C\theta_i = \cos\theta_i\ , \quad S\theta_i = \sin\theta_i$$

$$C\alpha_i = \cos\alpha_i\ , \quad S\alpha_i = \sin\alpha_i$$

And

θ_i - The joint angle from the X_{i-1} axis to the X_i axis about the Z_{i-1} axis

α_i - The offset angle from the Z_{i-1} axis to the Z_i axis about the X_i axis

d_i - The distance from the origin of the (i-1)th coordinate frame to the intersection of the Z_{i-1} axis with the X_i axis along the Z_{i-1} axis

a_i - The offset distance from the intersection of the Z_{i-1} axis with the X_i axis to the origin of the ith frame along the X_i axis

Hence, the position and orientation of the end-effector with respect to the world frame may be expressed in terms of the total transformation matrix T_o^a as follows:

$$T_o^a = A_0^1\ A_1^2\ A_2^3\ A_3^4\ A_4^5 \qquad (5 - 2)$$

Also

$$T_o^a = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5 - 3)$$

$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where

n - the normal vector of the robotic hand
o - the sliding vector of the robotic hand
a - the approach vector of the robotic hand
p - the position vector of the robotic hand

Fig. 5.2 shows the position and orientation of the robotic hand with respect to the world frame.

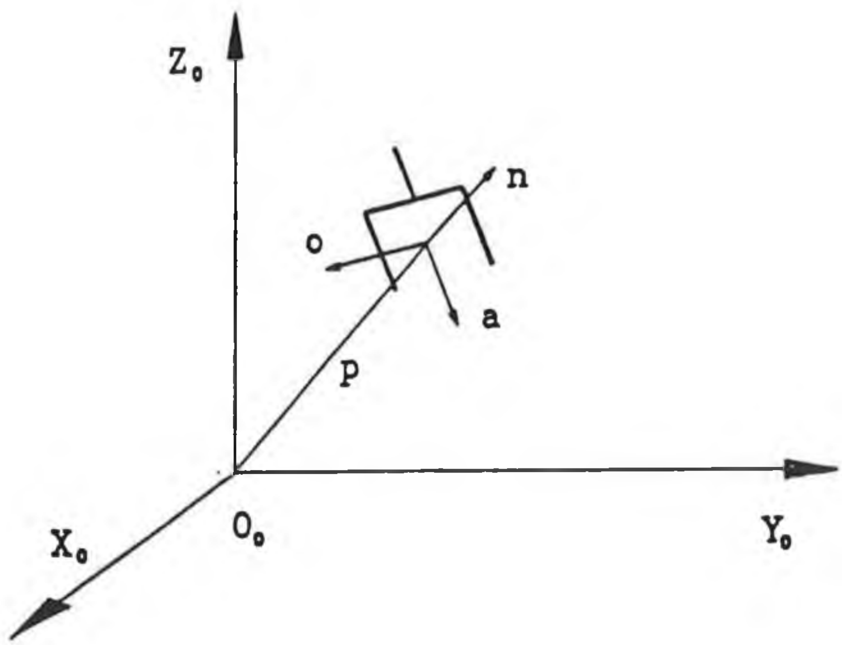


Fig. 5.2 Robotic hand position and orientation

While the approach vector points to the object to be grasped, the sliding vector together with the approach vector specifies the orientation of the hand. And the normal vector is orthogonal to the other two. One of the properties of the homogeneous transformations can be derived as:

$$
\begin{aligned}
o \times a &= n \\
a \times n &= o \qquad (5 - 4) \\
n \times o &= a
\end{aligned}
$$

Table 5.1 shows the Mitsubishi robot arm link coordinate parameters.

<u>Table 5.1 Robot arm link coordinate parameters</u>

Joint	θ_i	α_i	a_i	d_i	θ_i Range
1	0	90^0	0	d_1	$(-60^0, +240^0)$
2	0	0	a_2	0	$(-30^0, +100^0)$
3	0	0	a_3	0	$(-110^0, 0^0)$
4	270^0	90^0	0	0	$(-90^0, +90^0)$
5	0	180^0	0	$-d_5$	$(-180^0, +180^0)$

Substituting these parameters into eq. (5-1), the following homogeneous transformation matrice can be obtained:

$$A_0^{\,1} = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5\text{-}5\text{-}1)$$

$$A_1^{\,2} = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5\text{-}5\text{-}2)$$

$$A_2^{\,3} = \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5\text{-}5\text{-}3)$$

$$A_3^{\,4} = \begin{bmatrix} S_4 & 0 & -C_4 & 0 \\ -C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5\text{-}5\text{-}4)$$

$$A_4^{\,5} = \begin{bmatrix} C_5 & S_5 & 0 & 0 \\ S_5 & -C_5 & 0 & 0 \\ 0 & 0 & -1 & -d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5\text{-}5\text{-}5)$$

Where

$C_1 = \text{Cos } \theta_1$

$S_1 = \text{Sin } \theta_1$

Substituting eqs.(5-5-1) to (5-5-5) into eq.(5-2), and combining eq.(5-2) with eq.(5-3), one may obtain the direct kinematics of the robot arm.

$$n_x = C_1 \; S_{234} \; C_5 \; + \; S_1 \; S_5 \qquad (5\text{-}6\text{-}1)$$

$$n_y = S_1 \; S_{234} \; C_5 \; - \; C_1 \; S_5 \qquad (5\text{-}6\text{-}2)$$

$$n_z = - \; C_{234} \; C_5 \qquad (5\text{-}6\text{-}3)$$

$$o_x = C_1 \; S_{234} \; S_5 \; - \; S_1 C_5 \qquad (5\text{-}6\text{-}4)$$

$$o_y = S_1 \; S_{234} \; S_5 \; + \; C_1 C_5 \qquad (5\text{-}6\text{-}5)$$

$$o_z = - \; C_{234} \; S_5 \qquad (5\text{-}6\text{-}6)$$

$$a_x = C_1 \; C_{234} \qquad (5\text{-}6\text{-}7)$$

$$a_y = S_1 \; C_{234} \qquad (5\text{-}6\text{-}8)$$

$$a_z = S_{234} \qquad (5\text{-}6\text{-}9)$$

$$p_x = a_2 \; C_1 \; C_2 \; + \; a_3 \; C_1 \; C_{23} \; + \; d_5 \; C_1 \; C_{234} \qquad (5\text{-}6\text{-}10)$$

$$p_y = a_2 \; S_1 \; C_2 \; + \; a_3 \; S_1 \; C_{23} \; + \; d_5 \; S_1 \; C_{234} \qquad (5\text{-}6\text{-}11)$$

$$p_z = d_1 \; + \; a_2 \; S_2 \; + \; a_3 \; S_{23} \; + \; d_5 \; S_{234} \qquad (5\text{-}6\text{-}12)$$

Where

$$S_{ijk} = \mathrm{Sin}(\;\Theta_i + \Theta_j + \Theta_k\;)$$
$$C_{ijk} = \mathrm{Cos}(\;\Theta_i + \Theta_j + \Theta_k\;)$$

As an expample the following parameters were selected such that

$$\Theta_1=30^0,\; \Theta_2=45^0,\; \Theta_3=-30^0,\; \Theta_4=60^0,\; \Theta_5=45^0$$

Which gives the following results under the condition of $d_5 = 72$ mm.

$$T_o^a = \begin{bmatrix} 0.945 & 0.238 & 0.224 & 303.074 \\ -0.271 & 0.954 & 0.129 & 174.980 \\ -0.183 & -0.183 & 0.966 & 587.734 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5-2-2 Inverse kinematics

Given the position and orientation of the end-effector, it is required to find the corresponding joint space vector θ of the robot arm so that the end-effector can be positioned as required.

Referring to eqs. (5-6-1) to (5-6-12), the inverse kinematics problem can be solved as follows:

A. Pitch angle θ_{234}

From eq. (5-6-9), one may obtain

$$\theta_{234} = \mathrm{Sin}^{-1}(a_z) \qquad (5\text{-}7\text{-}1)$$

B. Joint angle θ_1

From eqs. (5-6-10) and (5-6-11), one obtains:

$$\theta_1 = \mathrm{Tan}^{-1}(p_y/p_x) \qquad (5\text{-}7\text{-}2)$$

C. Roll angle θ_5

If C_{234} <> 0, then from eqs. (5-6-3) and (5-6-6), one finds that

$$\theta_5 = \mathrm{Tan}^{-1}(-o_z/-n_z) \quad (5\text{-}7\text{-}3)$$

But if $C_{234} = 0$, then $\theta_{234} = \pm(2n-1)\pi/2$. Thus $S_{234} = \pm 1$. For $S_{234} = 1$, eq.(5-6-1) and eq.(5-6-4) can be expressed as

$$n_x = \mathrm{Cos}(\theta_1 - \theta_5)$$
$$o_x = -\mathrm{Sin}(\theta_1 - \theta_5)$$

Hence, the θ_5 is denoted by

$$\theta_5 = \theta_1 - \mathrm{Tan}^{-1}(-o_x/n_x) \quad (5\text{-}7\text{-}4)$$

For $S_{234} = -1$, eq.(5-6-1) and eq.(5-6-4) become

$$n_x = -\mathrm{Cos}(\theta_1 + \theta_5)$$
$$o_x = -\mathrm{Sin}(\theta_1 + \theta_5)$$

Thus, the θ_5 is denoted by

$$\theta_5 = -\theta_1 + \mathrm{Tan}^{-1}(-o_x/-n_x) \quad (5\text{-}7\text{-}5)$$

D. Joint angle θ_3

From eqs.(5-6-10) and (5-6-11), one may show that:

$$C_1p_x + S_1p_y = a_2C_2 + a_3C_{23} + d_5C_{234} \quad (5\text{-}7\text{-}6)$$

And from eqs.(5-6-12) and (5-7-6), one may get:

$$a_2C_2 + a_3S_{23} = \alpha \quad (5\text{-}7\text{-}7)$$

$$a_2C_2 + a_3C_{23} = \beta \quad (5\text{-}7\text{-}8)$$

Where

$$\alpha = p_z - d_1 - d_5S_{234}$$
$$\beta = C_1p_x + S_1p_y - d_5C_{234} \quad (5\text{-}7\text{-}9)$$

From eqs. (5-7-7) and (5-7-8), one may get

$$\alpha^2 + \beta^2 = a_2^2 + a_3^2 + 2a_2a_3C_3 \quad (5\text{-}7\text{-}10)$$

Such that

$$\Theta_3 = \cos^{-1} \frac{\alpha^2+\beta^2-a_2^2-a_3^2}{2a_2a_3} \quad (5\text{-}7\text{-}11)$$

E. Joint angle Θ_2

Expanding eqs.(5-7-7) and (5-7-8), one obtains:

$$a_2S_2 + a_3(S_2C_3 + C_2S_3) = \alpha \quad (5\text{-}7\text{-}12)$$

$$a_2C_2 + a_3(C_2C_3 - S_2S_3) = \beta \quad (5\text{-}7\text{-}13)$$

Multiply eq.(5-7-12) by S_2 and eq.(5-7-13) by C_2, and add to obtain:

$$a_2 + a_3C_3 = \alpha S_2 + \beta C_2 \quad (5\text{-}7\text{-}14)$$

Now multiply eq.(5-7-12) by C_2 and eq.(5-7-13) by S_2, and subtract to obtain:

$$a_3S_3 = \alpha C_2 - \beta S_2 \quad (5\text{-}7\text{-}15)$$

Multiply eq.(5-7-14) by ß and eq.(5-6-15) by α, and add to obtain:

$$\beta(a_2 + a_3C_3) + \alpha a_3S_3 = (\alpha^2 + \beta^2)C_2 \qquad (5\text{-}7\text{-}16)$$

Multiply eq.(5-7-14) by α and eq.(5-7-15) by ß, and subtract to obtain:

$$\alpha(a_2 + a_3C_3) - \beta a_3C_3 = (\alpha^2 + \beta^2)S_2 \qquad (5\text{-}7\text{-}17)$$

Now from eqs.(5-7-16) and (5-7-17), one finds that

$$\Theta_2 = \mathrm{Tan}^{-1} \frac{\alpha(a_2+a_3C_3)-\beta a_3S_3}{\beta(a_2+a_3C_3)+\alpha a_3S_3} \qquad (5\text{-}7\text{-}18)$$

F. Joint angle Θ_4

Thus the Θ_4 can be denoted by

$$\Theta_4 = \Theta_{234} - \Theta_2 - \Theta_3 \qquad (5\text{-}7\text{-}19)$$

The required inputs for the inverse computation are the position and orientation of the robotic hand, and the outputs of the inverse computation are the joint angles $\Theta_1 - \Theta_5$.

The inverse kinematics solutions for the robot arm are listed in Table 5.2. And Fig. 5.3 shows the computation algorithm of the inverse kinematics, which has been incorporated into the path planning and modifying modules.

Table 5.2 Inverse kinematics solutions

Joint	θ_i	θ_i Range
1	$\theta_1 = \mathrm{Tan}^{-1}(p_y/p_x)$	$(-60^0, +240^0)$
2	$\theta_2 = \mathrm{Tan}^{-1}\dfrac{\alpha(a_2+a_3C_3)-\beta a_3S_3}{\beta(a_2+a_3C_3)+\alpha a_3S_3}$	$(-30^0, +100^0)$
3	$\theta_3 = \mathrm{Cos}^{-1}\dfrac{\alpha^2+\beta^2-a_2^2-a_3^2}{2a_2a_3}$	$(-110^0, 0^0)$
4	$\theta_4 = \theta_{234} - \theta_3 - \theta_2$	$(-90^0, +90^0)$
5	$\theta_5 = \mathrm{Tan}^{-1}\dfrac{-o_z}{-n_z}$ if $C_{234}<>0$ $\theta_5 = \theta_1 - \mathrm{Tan}^{-1}\dfrac{-o_x}{n_x}$ if $S_{234}=1$ $\theta_5 = \mathrm{Tan}^{-1}\dfrac{-o_x}{-n_x} - \theta_1$ if $S_{234}=-1$	$(-180^0, +180^0)$
pitch	$\theta_{234} = \mathrm{Sin}^{-1}(a_z)$	

Where $\alpha = p_z - d_1 - d_5S_{234}$

$\beta = C_1p_x + S_1p_y - d_5C_{234}$

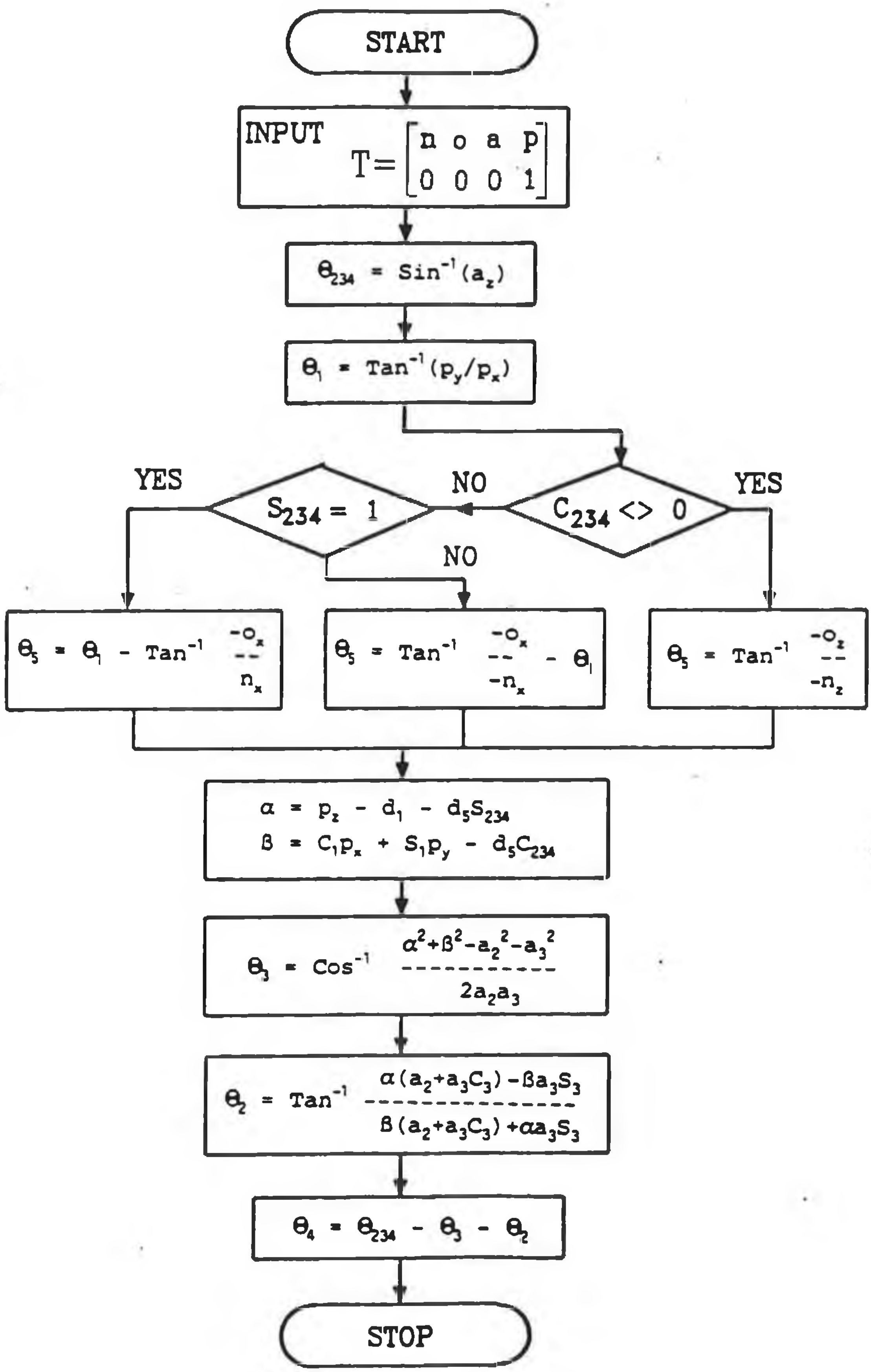


Fig. 5.3 Computation algorithm of Inverse Kinematics

As an example the following position and orientation were selected:

$$n = (\ 0,\ \ 0,\ -1)^T$$
$$o = (-1,\ \ 0,\ \ 0)^T$$
$$a = (\ 0,\ \ 1,\ \ 0)^T$$
$$p = (\ 0,\ 480, 300)^T$$

Which gave the following results under the condition of the tool length being of 0 mm (thus the d_5 = 72 mm).

$$\Theta_1 = 90^0$$
$$\Theta_2 = 4.53^0$$
$$\Theta_3 = -11.61^0$$
$$\Theta_4 = 7.08^0$$
$$\Theta_5 = 0^0$$

5-3 Coordinates of the end-effector

With two rotational fingers and a flat palm, the robotic hand has different contact (grasping) points with the environment.

For a kneading operation, the contacts with the part are made by the fingertips. As stated in Remark 2.1, the grasping center of the fingertips is required to follow the axial center line of the part being kneaded and the opening of the robotic fingers should comply with the radial path. Thus, the opening of the robotic fingers can be decided by the diameter of the part being massaged.

And the grasping distance varies with the openings of the fingers. Hence, for the kneading operation, the grasping center position together with the openings of the fingertips must be controlled.

For a padding operation by the palm, the fingers are required to be fully open and the contact is made by a fixed point (or a fixed area) in the center of the palm force sensor. Thus, for the padding operation, the contact point position of the robotic palm must be controlled.

To study the kinematics of the robotic hand, two coordinate frames have been established for the robotic hand according to the massaging modes (kneading or padding):

a. Kneading frame $O_K X_K Y_K Z_K$, which is located at the grasping point of the fingertips.

b. Padding frame $O_P X_P Y_P Z_P$, which is located at the contact point of the palm.

Since the massaging process of the robotic hand is a compliance process, hence the kneading frame and the padding frame are also termed as compliance frames.

Here the robotic hand frame $O_5 X_5 Y_5 Z_5$, which has been defined in 5-2, has been attached to the center of the robotic wrist mounting surface. Thus, $O_5 X_5 Y_5 Z_5$ can also be considered as a robotic wrist frame. Therefore, the motion of the robotic fingers and the palm can be analysed with respect to the robotic wrist frame. And the kinematics solutions obtained in 5-2 can be used directly to find the position and the orientation of the compliance frame with respect to the world frame.

Fig. 5.4 shows the coordinates of the robotic hand.

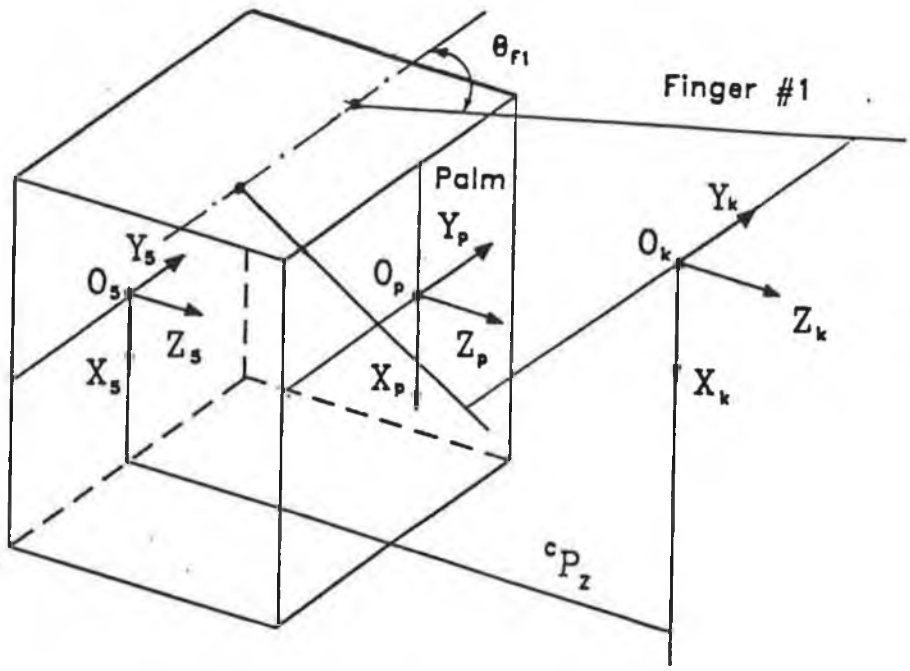


Fig. 5.4 Coordinates of the robotic hand

Where, O_K has been defined as the grasping center for kneading operations, and O_P has been defined as the contact point for padding operations. The position of the compliance frame (kneading or padding) with respect to the wrist frame $O_5X_5Y_5Z_5$ is denoted by a set of offset distances $(^cP_x, ^cP_y, ^cP_z)$. And the orientation of the compliance frame with respect to the world frame maintains the same orientation as the wrist frame.

Note that

^cP_x - offset distance along X_5
^cP_y - offset distance along Y_5
^cP_z - offset distance along Z_5

Thus, the position and orientation of the kneading and the padding frames with respect to the robotic wrist frame $O_5X_5Y_5Z_5$ can be denoted by

$$T_a^c = \begin{bmatrix} 1 & 0 & 0 & {}^cP_x \\ 0 & 1 & 0 & {}^cP_y \\ 0 & 0 & 1 & {}^cP_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-8}$$

Here T_a^c is termed as geometry compliance matrix. By assigning the desired values to $({}^cP_x, {}^cP_y, {}^cP_z)$, the kneading frame and the padding frame can be denoted respectively.

Let the virtual diameter of the part being massaged be denoted by D_R and the finger length be denoted by L_F. The massaging status of the robotic hand is shown in Fig 5.5.

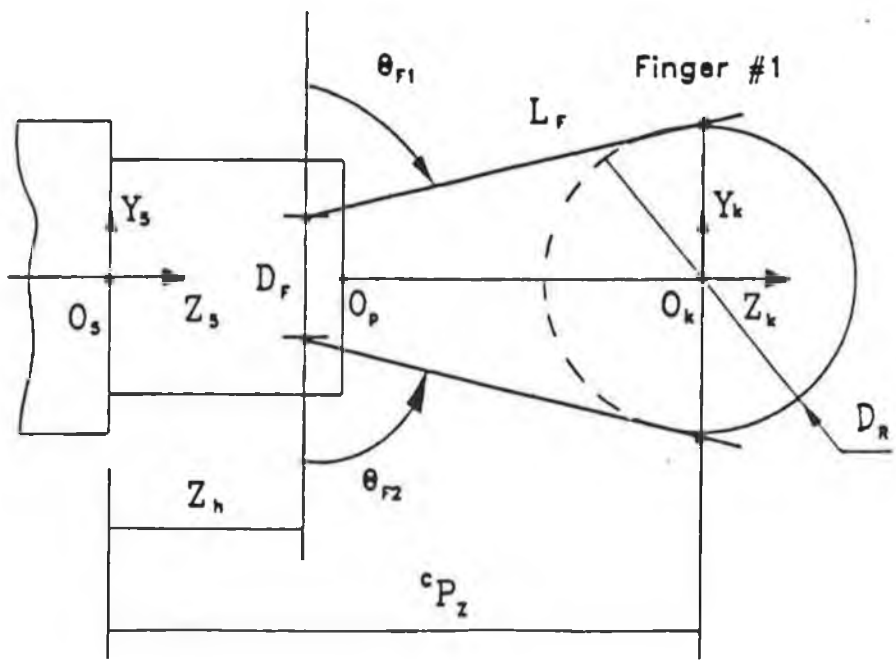


Fig. 5.5 Robotic hand in kneading operations

Thus, for the kneading opeartion, the finger rotational angles, θ_{F1} and θ_{F2}, are given by:

$$\theta_{F1} = \theta_{F2} = \mathrm{Cos}^{-1}[(D_R-D_F)/2L_F] \qquad (6\text{-}9)$$

And the grasping distance ${}^cP_z$ along Z_5 axis is given by:

$${}^cP_z = Z_h + (L_F^2-(D_R-D_F)^2/4)^{\frac{1}{2}} \qquad (6\text{-}10)$$

Where

Z_h = 65 mm

D_F = 40 mm

L_F = 115 mm for Hand-I; 95 mm for Hand-II

Table 5.3 lists the offset distances for Hand-I and Hand-II.

Table 5.3 Offset distances

Hand	Frame	${}^cP_x$	${}^cP_y$	${}^cP_z$
Hand-I	padding	-15	0	95
	Kneading	-34	0	${}^cP_z$
Hand-II	Padding	-5	0	95
	Kneading	-37	0	${}^cP_z$

The position and orientation of the compliance frame (kneading or padding frame) with respect to the world frame is denoted by:

$$T_o^c = T_o^a \, T_a^c \tag{5-11}$$

Where

T_o^c - the position/orientation of the compliance frame with respect to the world frame

T_o^a - the position/orientation of the robotic wrist frame with respect to the world frame

T_a^c - the position/orientation of the compliance frame with respect to the robotic wrist frame.

Combining eqs.(5-3), (5-8) and (5-11), one may obtain:

$$T_o^c = \begin{bmatrix} n_x & o_x & a_x & p_{xc} \\ n_y & o_y & a_y & p_{yc} \\ n_z & o_z & a_z & p_{zc} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-12}$$

Where

$$p_{xc} = p_x + ({}^cp_x n_x + {}^cp_y o_x + {}^cp_z a_x)$$

$$p_{yc} = p_y + ({}^cp_x n_y + {}^cp_y o_y + {}^cp_z a_y)$$

$$p_{zc} = p_z + ({}^cp_x n_z + {}^cp_y o_z + {}^cp_z a_z)$$

5-4 The massaging path design

The position path of the massaging should be designed according to the task specifications, i.e., kneading and/or padding.

For a given part to be massaged in the robot workspace, its axial path may be represented by a space curve along which the grasping center of the robotic hand should follow while the robotic fingers are rotated to comply with the radial path of the part.

Let the space curve be denoted by

$$\mathbf{r}(t) = X(t)\, i + Y(t)\, j + Z(t)\, k \qquad (5\text{-}13)$$

Any point, P, which moves along the space curve, can be defined here as:

$$P^t = (P_x^{\ t}, P_y^{\ t}, P_z^{\ t})^T$$

$$= (X(t), Y(t), Z(t))^T \qquad (5\text{-}14)$$

To obtain the feasible path for the robot hand to follow, two planes, P_m and P_n, have been employed to generate the orientation of the robotic hand along the space curve. Fig. 5.6 shows the basic principle of the trajectory which is followed by the robotic hand.

Where

P_m - the tangential plane at point P vertical to $O_0X_0Y_0$ plane

P_n - the normal plane with respect to the tangental line at point P along the space curve

n^t - vector along the tangental line at point P

o^t - vector along the intersection line of P_m and P_n at point P

a^t - vector parallel to $O_0X_0Y_0$ plane

ß - angle between the tangental plane P_m and $O_0X_0Z_0$ plane at point P. And

$$\beta = \mathrm{Tan}^{-1}(dY/dX)$$

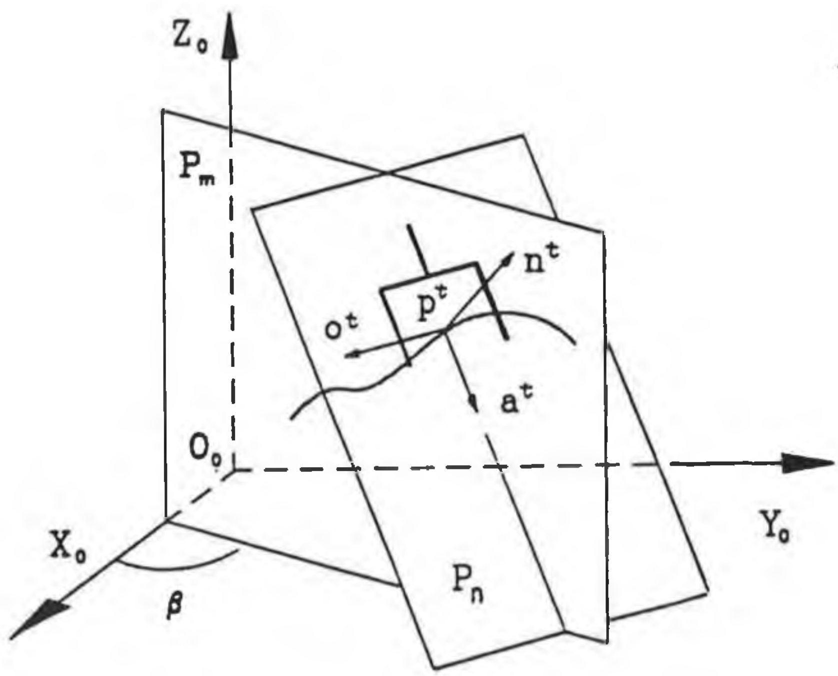


Fig. 5.6 Robot hand follows space curve

By using the geometry analysis method [146], the feasible orientation of the robot hand at point P can be obtained as follows.

a. $n^t = (n_x^t \; n_y^t \; n_z^t)^T$

$$n^t = dr/dt = \{ X'(t), \; Y'(t), \; Z'(t)\}^T \tag{5-15}$$

b. $o^t = (o_x^t \; o_y^t \; o_z^t)^T$

$$o^t = \{ \mathrm{Sin}\beta, \; -\mathrm{Cos}\beta, \; 0 \}^T \tag{5-16}$$

c. $a^t = (a_x^t \; a_y^t \; a_z^t)^T$

$$\begin{aligned} a^t &= n^t \times o^t \\ &= \{n_z^t \mathrm{Cos}\beta, \; n_z^t \mathrm{Sin}\beta, \; -(n_x^t \mathrm{Cos}\beta + n_y^t \mathrm{Sin}\beta)\}^T \end{aligned} \tag{5-17}$$

Hence, the position/orientation of the robotic hand at point P along the space curve can be denoted by a transformation matrix, T_0^t. Where,

$$T_0^t = \begin{bmatrix} n_x^t & o_x^t & a_x^t & p_x^t \\ n_y^t & o_y^t & a_y^t & p_y^t \\ n_z^t & o_z^t & a_z^t & p_z^t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-18}$$

T_0^t is also termed as position task matrix. For any given task, the position and the orientation of the axial path of the part in the world frame can be denoted by the task matrix as shown in eq.(5-18).

Case study

In this case the space curve has been simplified into a straight line in the world frame as shown in Fig. 5.7.

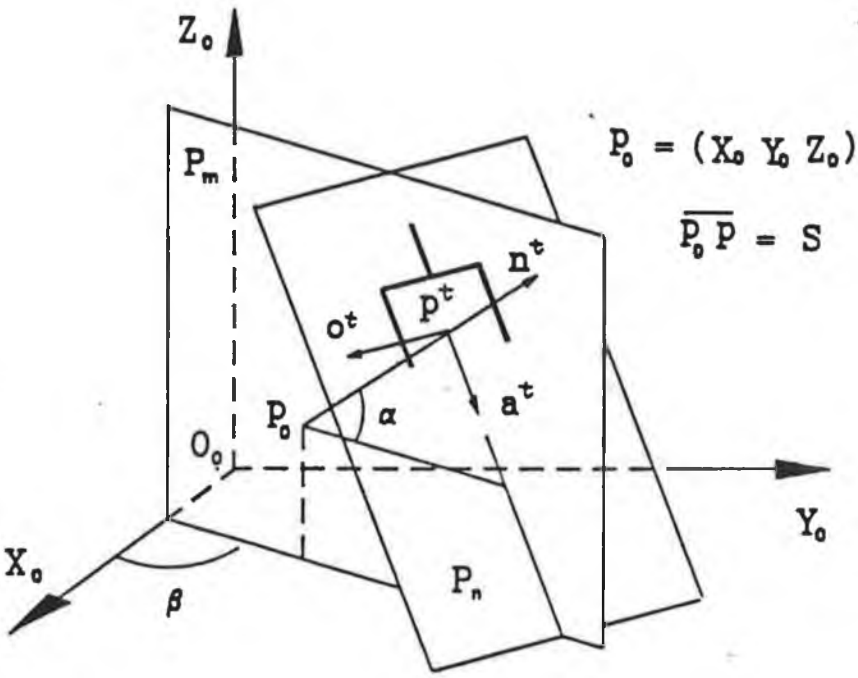


Fig. 5.7 Massaging path along a straight line

The straight line in the world frame can be denoted by

$$X(S) = X_0 + S \, Cos\alpha \, Cos\beta$$

$$Y(S) = Y_0 + S \, Cos\alpha \, Sin\beta \qquad (5\text{-}19)$$

$$Z(S) = Z_0 + S \, Sin\alpha$$

Where

S is the part length parameter

α is the angle between the line and $O_0X_0Y_0$ plane
(X_0, Y_0, Z_0) is the initial position of the part

Hence, from eqs.(5-15) - (5-17), the orientation of the robot hand can be obtained as follows:

$$\begin{aligned} n^t &= (\text{Cos}\beta\ \text{Cos}\alpha,\ \text{Sin}\beta\ \text{Cos}\alpha,\ \ \text{Sin}\alpha)^T \\ o^t &= (\ \ \text{Sin}\beta,\qquad -\text{Cos}\beta,\qquad 0\ )^T \qquad (5\text{-}20) \\ a^t &= (\text{Cos}\beta\ \text{Sin}\alpha,\ \text{Sin}\beta\ \text{Sin}\alpha,\ -\text{Cos}\alpha)^T \end{aligned}$$

And the position is:

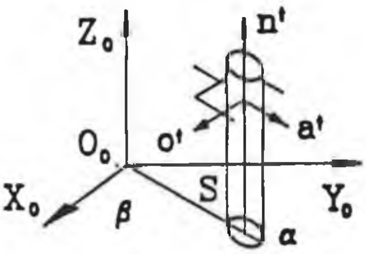
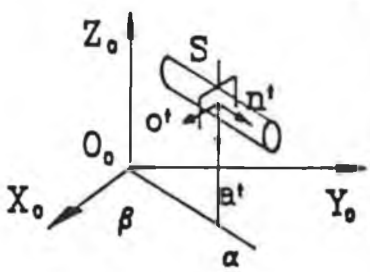
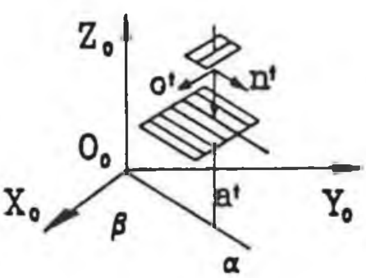
$$p^t = \{X_0 + S\text{Cos}\alpha\text{Cos}\beta,\ Y_0 + S\text{Cos}\alpha\text{Sin}\beta,\ Z_0 + S\text{Sin}\alpha\}^T \qquad (5\text{-}21)$$

Using the space line defined in eq.(5-19), one may specify the position/orientation of any part to be kneaded in the robotic workspace. And the massaging path along the axial direction of the part can be obtained by using eqs. (5-20) and (5-21).

For the padding operation, the massaging path can also be designed by using the defined space line in Fig.5.7. In this case, the straight line can be understood as a axial line on the padded plane. And the angle, α , can be understood as the angle between the padded plane and the $O_0X_0Y_0$ plane.

Table 5.4 shows three examples for the massaging path design. Where the kneading paths have been designed for a vertical part and a parallel part with respect to $O_0X_0Y_0$. And the padding path has been designed for a parallel plane with respect to $O_0X_0Y_0$.

Table 5.4 Special cases

Part in space	S	α	β	n^t	o^t	a^t	p^t
Vertical part for kneading $P_0 = (X_0\ Y_0\ Z_0)^T$ 	S_i	90^0	ß	0 0 1 0	Sß -Cß 0 0	Cß Sß 0 0	X_0 Y_0 Z_0+S_i 1
Parallel part for kneading $P_0 = (X_0\ Y_0\ Z_0)^T$ 	S_i	0^0	ß	Cß Sß 0 0	Sß -Cß 0 0	0 0 -1 0	$X_0+S_iC\beta$ $Y_0+S_iS\beta$ Z_0 1
Parallel plane for padding $P_0^* = (X_0\ Y_0\ Z_0)^T$ 	S_i	0	ß *	Cß Sß 0 0	Sß -Cß 0 0	0 0 -1 0	$X_0+S_iC\beta$ $Y_0+S_iS\beta$ Z_0 1

* varies with the radial paths

5-5 Motion control

For a given massaging task (kneading or padding), the massaging path can be designed by using the methods given in 5-4. To follow the designed path, the robot hand must be controlled with the required position and orientation. This can be achieved by controlling the joint space of the robot.

Motion control for kneading operations

For kneading operations using the robotic fingers, both the robotic finger joint space and the robotic arm joint space should be calculated and controlled.

The robotic finger rotational angles, Θ_{F1} and Θ_{F2}, can be calculated by using eq.(5-9). The required input for the computation of the rotational angles is the diameter, D_R, of the part being massaged.

The kneading geometry compliance matrix, T_a^c, which is related with the size of the part being massaged, can be obtained as shown in eq.(5-8) by referring to Table 5.3.

To follow the designed path, the motion control equation for the robot hand at any point P along the space curve must be maintained as follows:

$$T_0^c = T_0^t \qquad (5\text{-}22)$$

Where

T_0^c is the compliance matrix as defined in eq.(5-11).

T_0^t is the task matrix as defined in eq.(5-18).

Note that the robotic arm joint space parameters, θ_1 - θ_5, can be obtained by performing the inverse kinematics computations over the position/orientation matrix , T_0^a, of the robotic wrist frame with respect to the world frame.

Referring to eqs.(5-11) and (5-22), one may obtain:

$$T_0^a = T_0^t \, (T_a^c)^{-1} \tag{5-23}$$

Where

$(T_a^c)^{-1}$ is the inverse matrix of T_a^c

Furthermore, referring to eqs.(5-8), (5-18) and (5-23), T_0^a can be expressed as follows:

$$T_0^a = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-24}$$

Where

$$\begin{array}{lll} n_x = n_x^t & o_x = o_x^t & a_x = a_x^t \\ n_y = n_y^t & o_y = o_y^t & a_y = a_y^t \\ n_z = n_z^t & o_z = o_z^t & a_z = a_z^t \end{array} \tag{5-25}$$

and

$$\begin{aligned} p_x &= p_x^t - ({}^cp_x n_x + {}^cp_y o_x + {}^cp_z a_x) \\ p_y &= p_y^t - ({}^cp_x n_y + {}^cp_y o_y + {}^cp_z a_y) \\ p_z &= p_z^t - ({}^cp_x n_z + {}^cp_y o_z + {}^cp_z a_z) \end{aligned} \tag{5-26}$$

Motion control for padding operations

For padding operation using the robotic palm, only the robotic arm joint space is required to be controlled. Different from the kneading geometry compliance matrix, the padding geometry compliance matrix, T_a^c, is only related with the configuration of the robotic hand as defined in Table 5.3.

The motion control equation for the robot palm can also be denoted by eq.(5-22). Thus, the position and the orientation of the robotic wrist with respect to the world frame at any point P along the padding path can be obtained by using eqs.(5-24) - (5-26). And the robot arm joint parameters, θ_1 - θ_5, can be calculated by using the inverse kinematics of the robot arm.

Fig. 5.8 shows the required kinematics computations in the motion control process for the robot system.

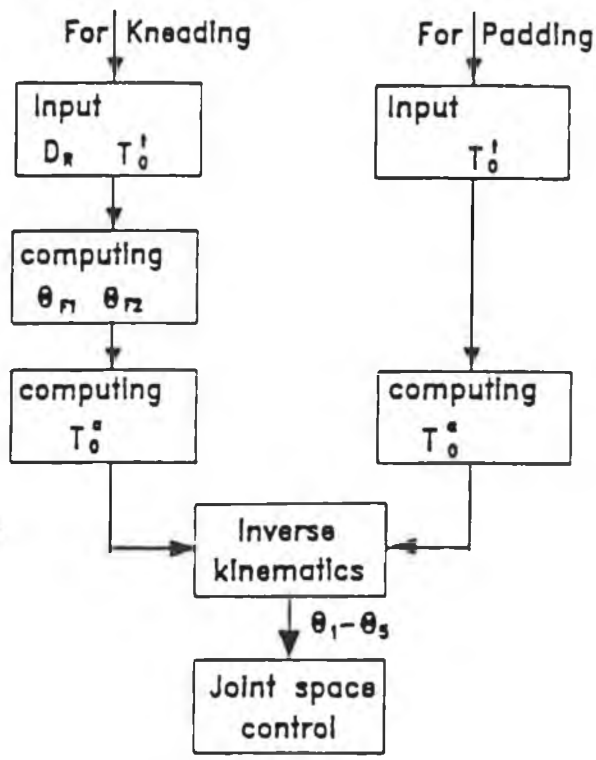


Fig. 5.8 Computations in motion control process

Chapter Six

Intelligent Control System

6-1 Introduction

In this chapter, the intelligent control system for the robotic massaging operations has been developed.

The overall AI system is described in section 6-2. The parameter organizing and path planning using the off-line KBS are given out in section 6-3. While the on-line error-corrections using the on-line KBS are constructed in section 6-5. The robot control organizing and task execution are outlined in section 6-4. And the software development of the AI control system together with the experimental results are presented in section 6-6.

6-2 AI Control System

There are many uncertainties or fuzziness in a robotic massaging process due to:

* The characteristics of the part to be massaged varies from one person to another.

* The unpredictable deviations of the part being massaged from its planned path

* The configuration limitations of the robotic sensing system.

To carry out the task execution using the robot, AI is required for the robotic massaging system. The AI embedded in the control system is capable of handling the imprecision (fuzzy) knowledge by using the fuzzy sets theory and fuzzy logic [109-116 & 123-128].

In general, the robotic system with AI is able to perform the following functions:

a. Path planning
b. Sensory information interpretation
c. Knowledge manipulation
d. Uncertainty/fuzzy processing
e. Intelligent inference
f. Conditional adaptive control
g. Process monitoring
h. Automatic error-correction

The design objectives of the intelligent robot system depend on the application fields. In this investigation, the intelligent robot control system has been developed to achieve the following objectives:

a. friendly man-machine dialogue
b. parameter organizing by using the fuzzy sets
c. path planning by using the robotic kinematics KB
d. automatic motion coordinating
e. process execution and monitoring
f. intelligent inference based on the fuzzy logic
g. automatic error-correction by using the on-line KB

For a given part in the robot workspace, the man-machine module provides the system with the fuzzy descriptions of the part and its environment.

Two types of knowledge bases, off-line KB and on-line KB, have been established in the AI system according to their applications.

The off-line KB will be interfaced and used by the following modules:

a. man-machine module
b. parameter organizing module
c. path planning module

And the on-line KB will be interfaced and used by the following modules:

a. intelligent control module
b. error-correction module
c. path modifying module

Once the path planning has been completed, a massaging can be carried out by the execution module under the supervision of the intelligent control module.

The massaging operation will be closely watched by the intelligent control module. The error-correction module will be initiated by the intelligent control module once any error is detected from the sensing feedback.

Fig. 6.1 shows the AI control system for the robot system.

A specially designed robotic hand with the position/force sensors has been used to carry out the massaging operations. The force/tactile sensors are mounted on the fingertips and the palm of the robotic hand.

The operation procedure of the robotic massaging system with AI is illustrated in Fig. 6.2.

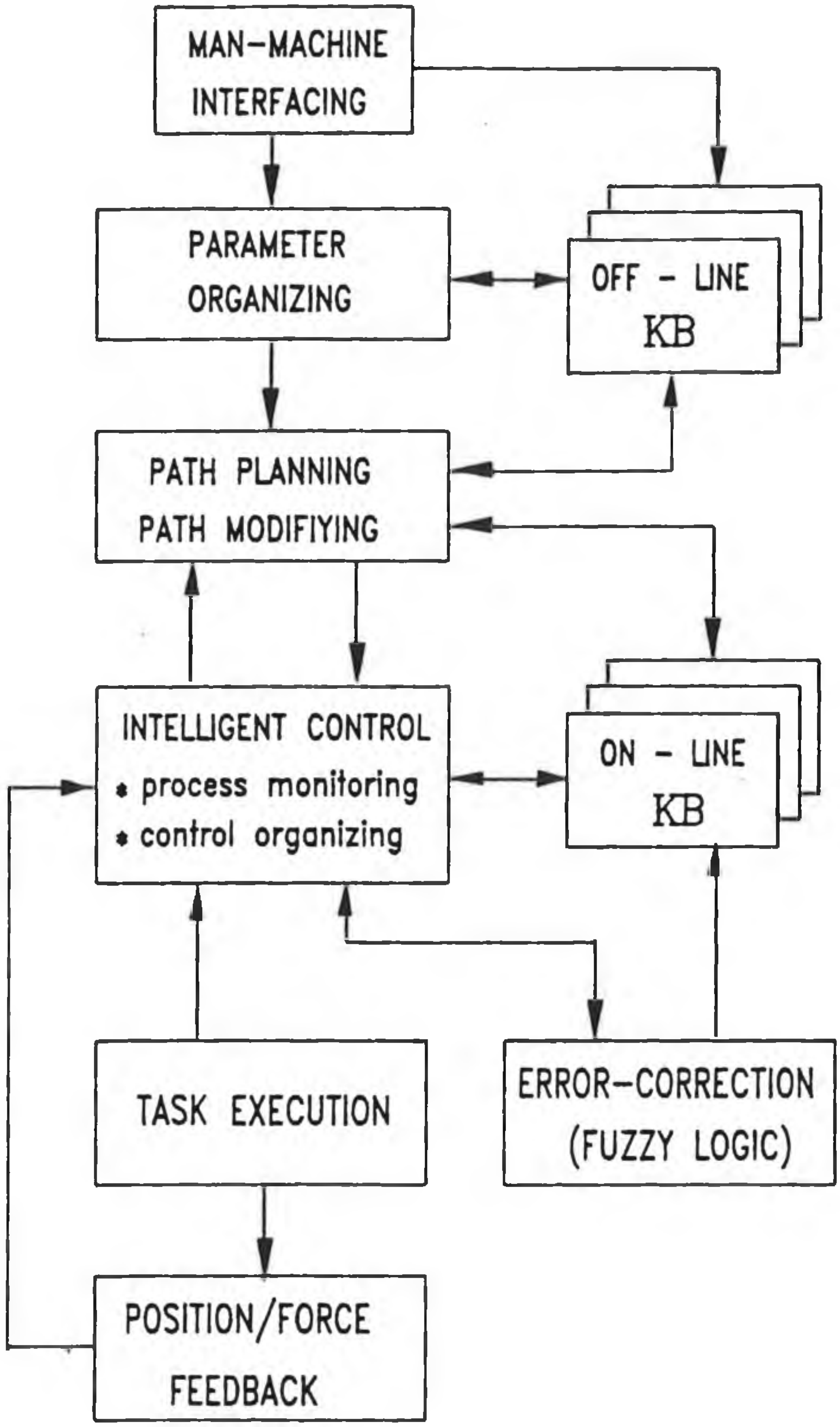


Fig. 6.1 Schematic representation of the AI system

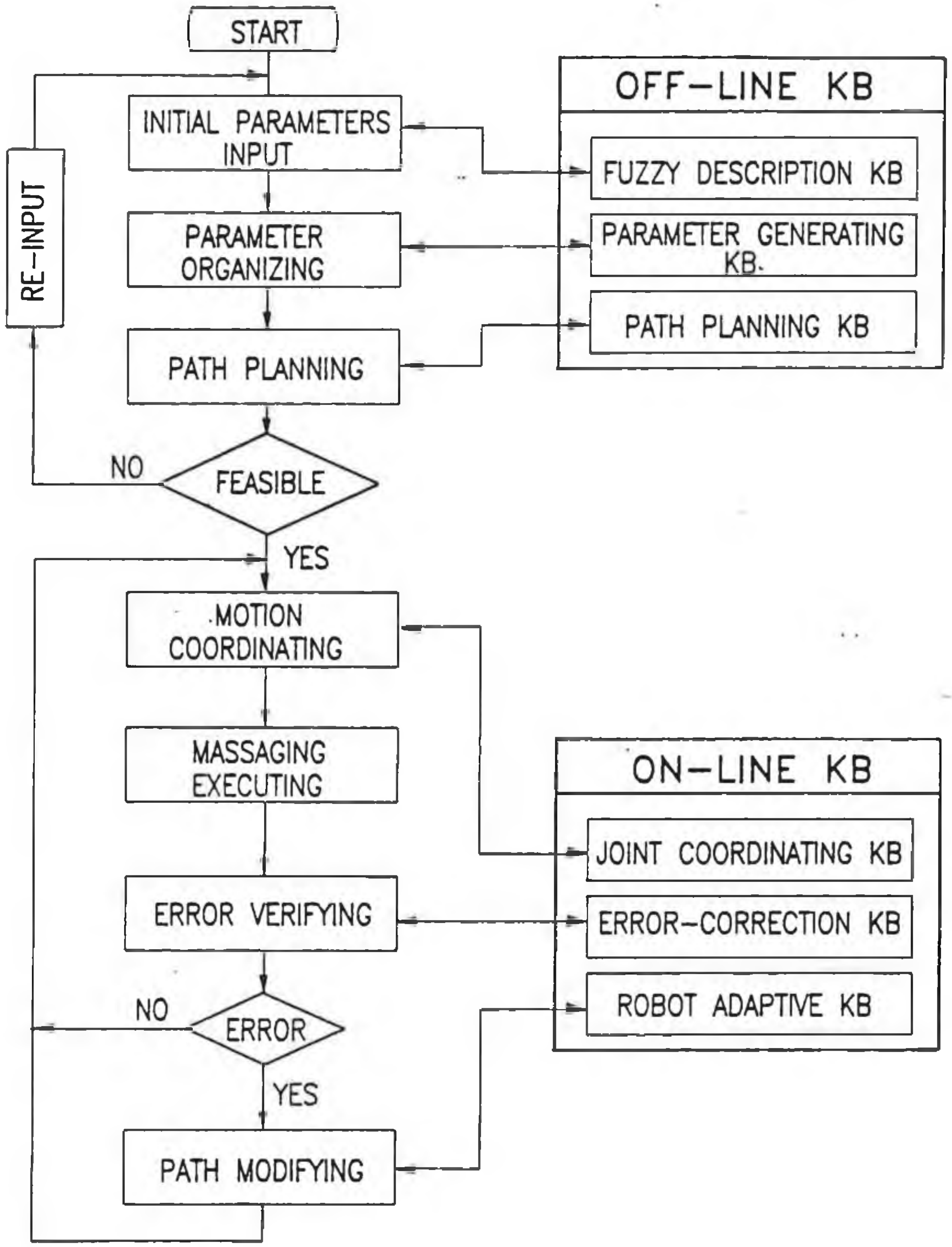


Fig. 6.2 Robotic massaging operation procedure

6-3 Parameter organizing and path planning

In the robot workspace, the human's observations provide the quickest way to describe the parts to be massaged provided that the location constraints have been imposed on the parts. Nevertheless, the human assessments are usually in a term of fuzziness. For example, the geometry size of an arm to be massaged in a predefined location may be described as "large", which is a fuzzy description [110-114 & 124-127].

Furthermore, the massaging force and massaging path have to be decided by using the human's massaging knowledge.

Hence, a off-line knowledge base can be constructed to assist the interpretation of the fuzzy inputs, the parameter generating and the path planning.

6-3-1 Off-line KB

The off-line KB consists of three parts: fuzzy description KB, parameter generating KB and path planning KB, as shown in Fig. 6.3.

A. Fuzzy description KB

The fuzzy/linguistic description of a task include:

a. the part to be massaged (linguistic terms)

arm, neck, back, etc.

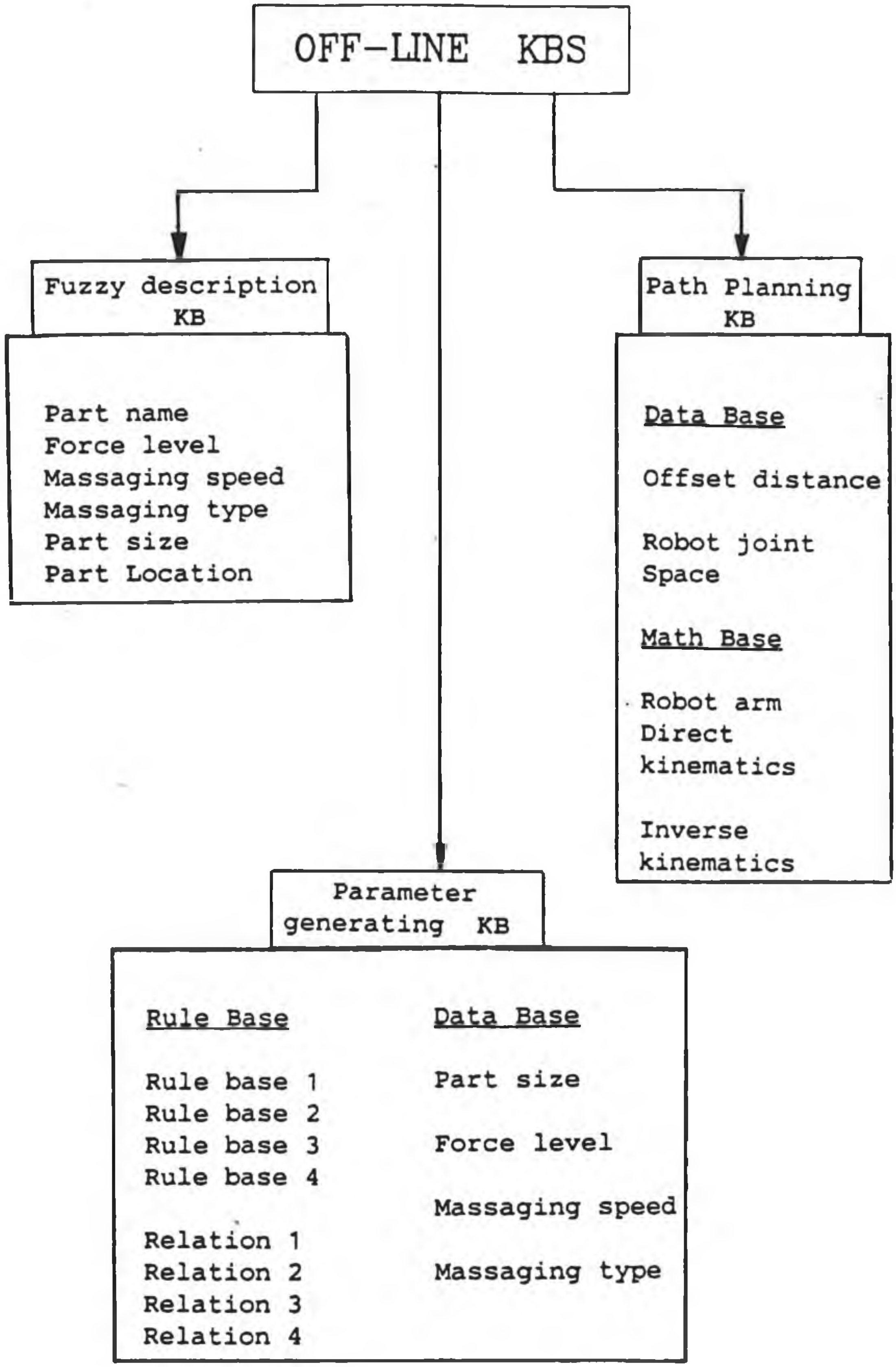


Fig. 6.3 Off-line KB

b. the force level (fuzzy terms)

[smaller, small, medium, big, bigger]

or

[SME, SM, ME, BG, BGE]

c. the speed of massaging (fuzzy terms)

[lower, low, medium, high, higher]

or

[LWE, LW, ME, HG, HGE]

d. the massaging type (fuzzy terms)

[coarse, standard, fine]

or

[CRS, STD, FIN]

The fuzzy/linguistic description of an environment include:

a. the part size (fuzzy terms)

[smaller, small, medium, large, larger]

or

[SME, SM, ME, LG, LGE]

b. the location of the part (linguistic terms or crisp)

The location of the part can be specified either in a crisp manner or in a linguistic manner [126]. When specified by linguistic terms, the part is assumed to be located in the predefined positions with predefined orientations. Otherwise, the position and orientation of the part is measured and then specified either automatically or manually.

B. Parameter generating KB

B.1. Rule base

The rule base can be established by considering the following correlations generalized from the human massaging knowledge:

* the massaging force level related with the part size

* the robot arm speed and the force retention time related with the massaging speed

* the number of the massaging points and the number of the radial path related with the massaging type

* the length and diameter (or the height and width) of the part related with the size of the part.

Based on the above correlations, the rule base has been constructed in the form of fuzzy conditional statements:

IF (a set of conditions are satisfied)

THEN (a set of consequences can be inferred)

Thus, the following fuzzy relations (Rule base 1 - Rule base 4) have been incorporated into the rule base for the parameter generating module.

Table 6.1 Rule base 1

Rules	IF Part size	THEN Massaging force
R11	SME (Smaller)	SME (Smaller)
R12	SM (Small)	SM (Small)
R13	ME (Medium)	ME (Medium)
R14	LG (Large)	BG (Big)
R15	LGE (Larger)	BGE (Bigger)

Table 6.2 Rule base 2

Rules	IF Massaging speed	THEN Arm speed	 Force retention time
R21	LWE (Lower)	LWE (Lower)	LNE (Longer)
R22	LW (Low)	LW (Low)	LN (Long)
R23	ME (Medium)	ME (Medium)	ME (Medium)
R24	HG (High)	HG (High)	SH (Short)
R25	HGE (Higher)	HGE (Higher)	SHE (Shorter)

Table 6.3 Rule base 3

Rules	IF	THEN	
	Part size	Diameter (Width)	Length (Height)
R31	SME (Smaller)	SME (Smaller)	SHE (Shorter)
R32	SM (Small)	SM (Small)	SH (Short)
R33	ME (Medium)	ME (Medium)	ME (Medium)
R34	LG (Large)	LG (Large)	LN (Long)
R35	LGE (Larger)	LGE (Larger)	LNE (Longer)

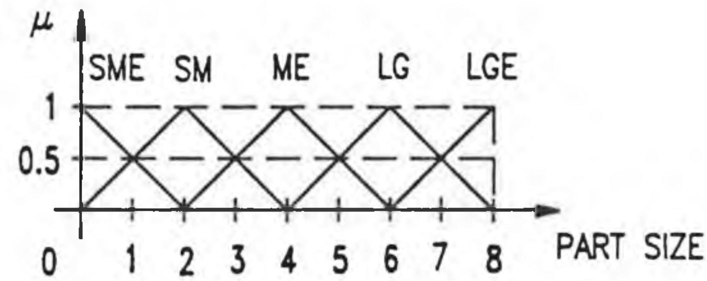
Table 6.4 Rule base 4

Rules	IF	THEN	
	Massaging type	Path number	Point number
R41	CRS (Coarse)	SM (Small)	SM (Small)
R42	STD (Standard)	ME (Medium)	ME (Medium)
R43	FIN (Fine)	BG (Big)	BG (Big)

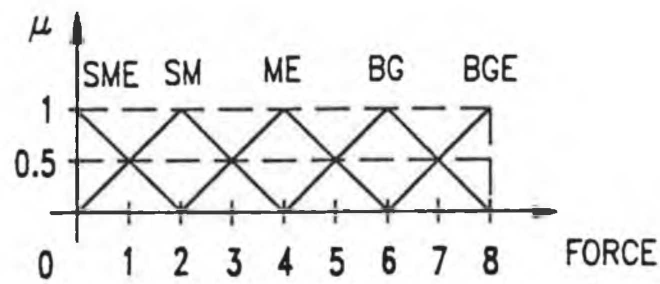
B.2. Membership function

During the process of construction of the data base, the triangular shape has been employed to describe the fuzzy sets. The universe of the input/output has been partitioned according to the assigned range of the fuzzy variables. And different membership values are assigned to each element of the discrete universe.

Fig 6.4 shows two examples of membership functions for part size and massaging force.



a). Part size membership



b). Force membership

Fig. 6.4 Membership functions

From Fig. 6.4, the discretized universes [117] of the fuzzy variables (part size and massaging force) can be derived as shown in Table 6.5 and Table 6.6.

Table 6.5 Universe of part size

Fuzzy	Discrete universe of part size								
terms	0	1	2	3	4	5	6	7	8
SME	1	0.5	0	0	0	0	0	0	0
SM	0	0.5	1	0.5	0	0	0	0	0
ME	0	0	0	0.5	1	0.5	0	0	0
LG	0	0	0	0	0	0.5	1	0.5	0
LGE	0	0	0	0	0	0	0	0.5	1

Table 6.6 Universe of massaging force

Fuzzy	Discrete universe of massaging force								
terms	0	1	2	3	4	5	6	7	8
SME	1	0.5	0	0	0	0	0	0	0
SM	0	0.5	1	0.5	0	0	0	0	0
ME	0	0	0	0.5	1	0.5	0	0	0
BG	0	0	0	0	0	0.5	1	0.5	0
BGE	0	0	0	0	0	0	0	0.5	1

The same procedure can be applied to define the fuzzy membership functions and to derive the discretized universes of the fuzzy variables: massaging speed, robot arm speed, force retention time.

The discretized universes for massaging speed, robot arm speed and the force retention time can be found in Appendix E-1.

B.3. Fuzzy relations

Due to the difficulty of having a control rule for every possible situation, a composition rule of inference may be used to obtain an output subset which belongs to the output fuzzy set from an fuzzy input term using the fuzzy relationship between the object in the condition section (known as "IN_PUT") and the object in the consequence section (Known as "OUT_PUT").

For example, the object in the condition section (or IN_PUT) in Rule Base 1 (see Table 6.1) is referred to "PART SIZE", while the object in the consequence section (or OUT_PUT) in Rule Base 1 is referred to "MASSAGING FORCE".

Let the object in the condition section of the jth rule base be denoted by IN_PUT^j, and the object in the consequence section of the jth rule be denoted by OUT_PUT^j.

Thus, from Table 6.1 to Table 6.4, the IN_PUT^j and OUT_PUT^j can be outlined as shown in Table 6.7.

Table 6.7 INPUT and OUTPUT terms

Rule base	IN_PUT^j	OUT_PUT^j No. 1	OUT_PUT^j No. 2
1	part size	massaging force	
2	massaging speed	arm speed	force retention time
3	part size	diameter	length
4	massaging type	path number	point number

For the ith rule in the jth rule base, R_{ji}, the fuzzy relations between the IN_PUT and the OUT_PUT can be denoted by:

$$R_{ji} = [IN\_PUT^j]_i^T * [OUT\_PUT^j]_i \quad (6\text{-}1)$$

Where, * denotes the operator for fuzzy relations

The membership function, μR_{ji}, for the fuzzy relationship is given by:

$$\mu R_{ji} = \text{MIN} \{ \mu[\text{IN\_PUT}^j]_i^T, \mu[\text{OUT\_PUT}^j]_i \} \quad (6\text{-}2)$$

Where

$\mu[\text{IN\_PUT}^j]_i$ -- the membership in the discrete universe corresponding to the ith fuzzy input term in the condition section of the jth rule base.

$\mu[\text{OUT\_PUT}^j]_i$ -- the membership in the discrete universe corresponding to the ith fuzzy output term in the consequence section of the jth rule base.

By combining all the rules in the jth rule base using the fuzzy operator "OR", the membership function for the relationship between the IN_PUT and the OUT_PUT of the jth rule base is given by:

$$\mu R_j = \text{MAX} \{ \mu R_{j1}, \mu R_{j2}, \mu R_{j3}, \ldots, \mu R_{jn} \} \quad (6\text{-}3)$$

Thus, the fuzzy relations between the IN_PUT and the OUT_PUT for all the rule bases can be established by using eq.(6-2) and eq.(6-3).

Example: Procedure to establish the fuzzy relations between the IN_PUT^1 and the OUT_PUT^1 for rule base 1 by using eq.(6-2) and eq.(6-3).

Referring to Table 6.1, one may know that

IN_PUT^1 = " part size"
OUT_PUT^1 = " massaging force"

For the first rule, R_{11}, in rule base 1, the fuzzy input term for IN_PUT^1 is "SME" and the fuzzy output for OUT_PUT^1 is "SME".

Thus, referring to Table 6.5 and Table 6.6, one may obtain:

$$\mu[IN\_PUT^1]_1 = 1/0 + 0.5/1 + 0/2 + 0/3 + 0/4 + 0/5 + 0/6 + 0/7 + 0/8 \quad (6\text{-}4)$$

$$\mu[OUT\_PUT^1]_1 = 1/0 + 0.5/1 + 0/2 + 0/3 + 0/4 + 0/5 + 0/6 + 0/7 + 0/8 \quad (6\text{-}5)$$

Substituting eqs.(6-4) and (6-5) into eq.(6-2), one may obtain:

Table 6.8 μR_{11}

μR_{11}		Discrete universe of massaging force								
		0	1	2	3	4	5	6	7	8
universe of part size	0	1	0.5	0	0	0	0	0	0	0
	1	0.5	0.5	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0

Using the same method, the rest membership functions, μR_{12}, μR_{13}, μR_{14} and μR_{15}, can be obtained.

Hence, using eq.(6-3), one can obtain the membership function, as shown in Table 6.9, for the relations between the "part size" and the "massaging force" in rule base 1.

Table 6.9 Membership function for rule base 1

R_1		Universe of massaging force								
		0	1	2	3	4	5	6	7	8
universe of part size	0	1	0.5	0	0	0	0	0	0	0
	1	0.5	0.5	0.5	0.5	0	0	0	0	0
	2	0	0.5	1	0.5	0	0	0	0	0
	3	0	0.5	0.5	0.5	0.5	0.5	0	0	0
	4	0	0	0	0.5	1	0.5	0	0	0
	5	0	0	0	0.5	0.5	0.5	0.5	0.5	0
	6	0	0	0	0	0	0.5	1	0.5	0
	7	0	0	0	0	0	0.5	0.5	0.5	0.5
	8	0	0	0	0	0	0	0	0.5	1

Using the same principles, the fuzzy relations in rule base 2, rule base 3 and rule base 4 can be expressed by the fuzzy membership functions, which are listed in Appendix E-2.

B.4. Data base

The data base has been established to assist the parameter generating. It must be mentioned that the data in the data bases are given out based on the observations of the author and the considerations of the robotic massaging system's configuration and limitations. For the practical usages, they are subject to modifications to meet the requirements of the massaging environment and the system.

Table 6.10 Data base of part size (mm)

Part		Discrete universe of the part size								
		0	1	2	3	4	5	6	7	8
ARM	D	60	70	80	90	100	110	120	130	140
	L	80	90	100	110	120	130	140	150	160
NECK	D	80	85	90	95	100	105	110	115	120
	L	30	35	40	45	50	55	60	65	70
BACK	W	80	90	100	110	120	130	140	150	160
	H	80	90	100	110	120	130	140	150	160

Where

D -- diameter of the part
L -- Length of the part
W -- Width of the back
H -- Height of the back

Table 6.11 Data base of force level (N)

Part	Discrete universe of massaging force 0	1	2	3	4	5	6	7	8
Arm	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
Neck	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
Back	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0

Table 6.12 Data base of massaging speed

	Discrete universe of massaging speed 0	1	2	3	4	5	6	7	8
Arm speed (Speed$)	1	2	3	4	5	6	7	8	9
Force retention time (t_{FR})	2	1.5	1.5	1	1	1	0.75	0.75	0.5

Table 6.13 Data base of massaging type

	Discrete universe of massaging type				
	0	1	2	3	4
Massaging points (m)	2	3	4	5	6
Massaging paths (N)	4	6	8	10	12

C. Path planning KB

C.1. Data base of the offset distances

The offset distances for the compliance frame, which has been defined in Table 5.3, have been incorporated into the path planning KB as one of the data bases.

C.2. Data base of the robot joint space

The valid ranges of the robot joint space variables, $[\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_{F1}, \Theta_{F2}]$, have been incorporated into the path planning KB as one of the data bases.

The valid ranges of the robot joint space variables are shown in Table 6.14.

Table 6.14 Robot valid joint space

Joints	Valid ranges
θ_1	[-60^0, $+240^0$]
θ_2	[-30^0, $+100^0$]
θ_3	[-110^0, 0^0]
θ_4	[-90^0, $+90^0$]
θ_5	[-180^0, $+180^0$]
θ_{F1}	[-15^0, $+95^0$]
θ_{F2}	[-15^0, $+95^0$]

C.3. Math base for robot arm

The robot arm position matrix, T_0^a, has been incorporated into the the path planning KB as one of the math bases.

The position matrix, T_0^a, has been defined in eq.(5-23). For a given task matrix, the position and orientation of the robot arm are denoted by eq.(5-24).

C.4. Math base for inverse kinematics

The computation algorithm of inverse kinematics, which has been shown in Fig. 5.3, has been incorporated into the path planning KB as one of the math bases.

6-3-2 Knowledge based parameter organizing

The procedure of the knowledge based parameter organizing is shown in Fig. 6.5.

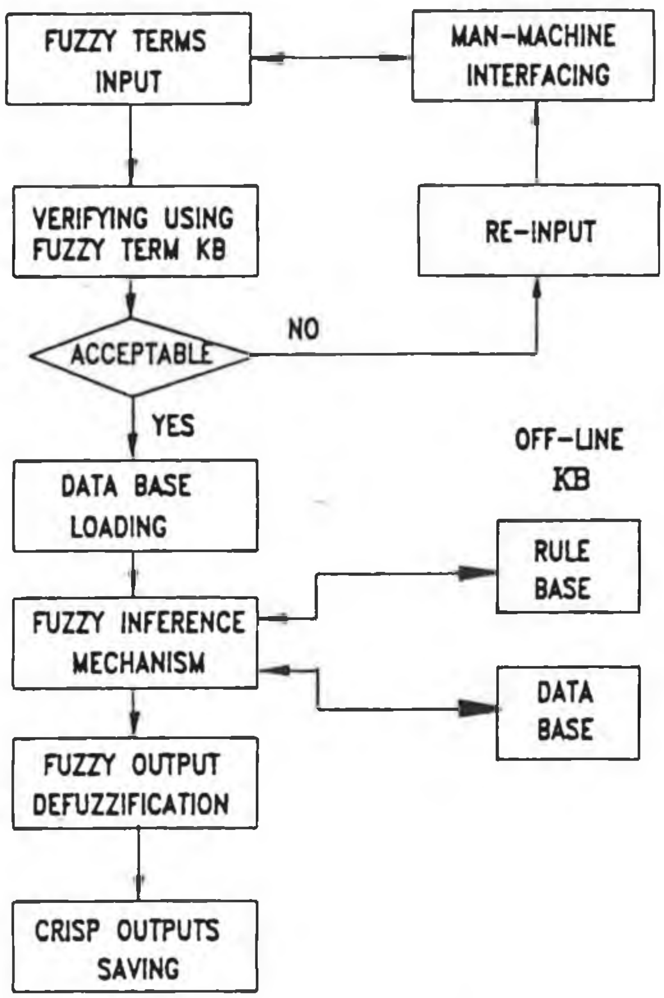


Fig. 6.5 Knowledge based parameter organizing

The input to the parameter generating module are:

* part to be massaged (linguistic)
* part size (fuzzy)
* massaging speed (fuzzy)
* massaging type (fuzzy)
* part location (crisp/linguistic)

The output of the parameter generating module are;

* the robot arm speed (Speed$)
* the massaging force (FForce)
* the force retention time (t_{FR})
* the diameter (width) of the part (Dpart)
* the length (height) of the part (Lpart)
* the No. of the radial massaging paths (N)
* the massaging points along the radial path (m)
* the initial position of the part (X_0 Y_0 Z_0)
* the angle ß for the part
* the angle α for the part

The fuzzy inference is carried out by manipulating the rule bases. For a fuzzy input terms, the output can be inferred by using the fuzzy relations which have been obtained in parameter generating KB. And the data base in the KB will also be manipulated to obtain the crisp values of the inferred fuzzy output.

Example: Assume that the "arm" size is "small". What is the massaging force? (The massaging force is related with the part!)

Solution:

a). Referring to Fig. 6.4, the "small" part size is corresponding to the universe "2".

b). Referring to Table 6.9, the fuzzy membership function for the relation between the massaging force and the part size corresponding to the part size "2" is:

$$[\mu Force] = 0/0 + 0.5/1 + 1/2 + 0.5/3 + 0/4 + 0/5 + 0/6 + 0/7 + 0/8$$

c). Referring to Table 6.11, the massaging force distribution for the "arm" along the discrete universe of the massaging force is:

$$[Force] = 1.5/0 + 2/1 + 2.5/2 + 3/3 + 3.5/4 + 4/5 + 4.5/6 + 5/7 + 5.5/8$$

d). The defuzzfied output of the massaging force can be obtained :

$$FForce = [\mu Force]*[Force]^T/\Sigma \mu Force_j$$

$$= \frac{0.5*2 + 1*2.5 + 0.5*3}{0.5 + 1 + 0.5} = 2.5\ (N)$$

6-3-3 Knowledge based path planning

Before massaging is performed, the following massaging paths must be planned based on the generated parameters by the planning system with the aid of the path planning KB:

* the radial/axial position path
* the force path

Once the position/force paths have been planned, two groups of the control data should be generated for the robot system at every massaging point:

a. the robot joint space parameters

* robot arm joint space variables θ_1 - θ_5
* robotic hand joint space parameters: θ_{F1} - θ_{F2}
* force supplied by the hand: F_1 - F_3

b. the robot Cartesian space parameters

* robot hand position [X Y Z θ_p θ_r θ_{F1} θ_{F2}]
* robot hand orientation [n o a]

Position path planning

The mathematical analysis of the massaging path design for a given part in the robot workspace has been given in section 5-4. And the motion control for the robotic hand has been studied in section 5-5.

A discrete massaging path has been employed to plan the path. For a part with conical shape (arm, neck, etc.), the discrete massaging path is shown in Fig. 6.6. For a part with flat surface, the discrete massaging path is shown in Fig. 6.7.

For the kneading operation, the part size is denoted by its diameter (Dpart) and length (Lpart). And the part is required to be centralized in the robotic hand frame. For the padding operation, the part size is denoted by its width (Dpart) and height (Lpart). For both operations, the number of the axial paths is denoted by N and the massaging points along the radial path is denoted by m.

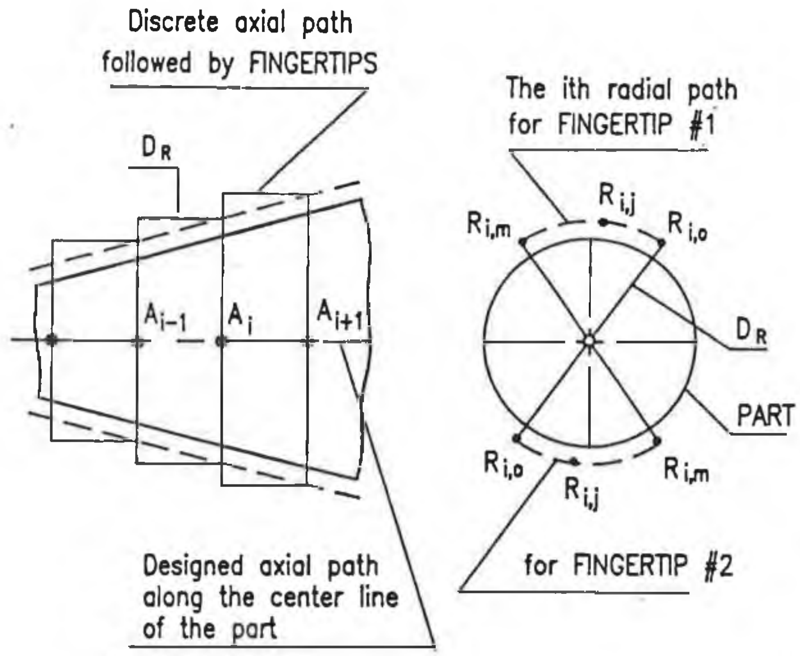


Fig. 6.6 Discrete path for conical part

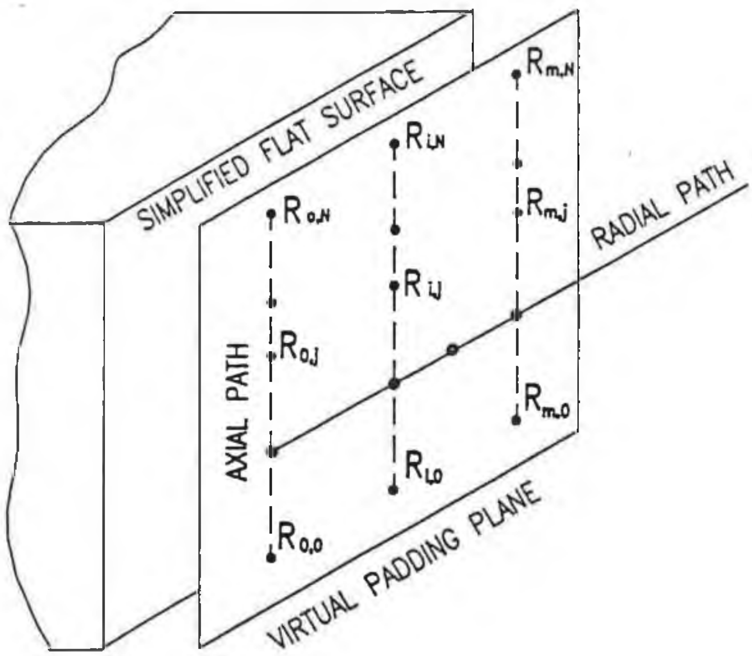


Fig. 6.7 Discrete path for flat part

In the Cartesian space, the initial position of the part is known as (X_0 Y_0 Z_0). And the orientation of the part is specified by α and ß. Hence, the position and orientation of the part can be determined.

At the ith massaging path, the part length parameter, S, along the axial path can be denoted by

$$S = (Lpart/N)*i \qquad (6\text{-}6)$$

Thus, the axial massaging path can be denoted by eq.(5-19). And the position and orientation, T_0^t, of the robotic hand to follow the massaging path are given by eqs.(5-20) and (5-21).

Using the math bases in the path planning KBS, the position/orientation of the robot arm can be obtained. The inverse kinematics computations can be performed. And the robot joint space variables can be obtained.

Force path planning

For the kneading operation by the fingers, the kneading forces, F_1 and F_2, can be specified in the robot fingers joint space. At every massaging point, both robotic fingertips should apply the required forces (FForce) onto the part surface. And the forces will be retained for a certain period of time (t_{FR}).

For the padding by the palm, the padding force, F_3, is specified along the the approach vector of the robotic hand. The padding force can be achieved by controlling the compliance motion of the robot arm along the force direction in the Cartesian space.

Since a big force will damage the robot hand or hurt the part. While a small force will not meet the task requirements. Hence, the planned forces should be evaluated agaist the valid work range of the robot hand by using the path planning KB.

Example: path planning for kneading operation

a. Parameter generating

In the parameter generating process, the location of the part is specified by the user. While the other parameters such as massaging force, robot arm speed, etc. are generated by using the parameter generating KB.

For the fuzzy/linguistic inputs:

Part to be massaged	=	"ARM"
Part size	=	"SM"
Massaging type	=	"CRS"
Massaging speed	=	"HG"
Robot hand used	=	"HANDNEW"

The following parameters can be inferred by using the parameter generating KB:

Massaging action	(ACT$)	=	"KNEAD"	
Massaging paths	(N)	=	6	
Massaging points	(m)	=	3	
Robot arm speed	(Speed$)	=	7	
Force retention time	(t_{FR})	=	0.81	(Sec)
Massaging force	(FForce)	=	2.50	(N)
Diameter of the arm	(Dpart)	=	80	(mm)
Length of the arm	(Lpart)	=	100	(mm)

The position of the part has been specified as:

$$X_0 = -500$$
$$Y_0 = 0$$
$$Z_0 = 300$$
$$\alpha = 90^0$$
$$\beta = 180^0$$

b. Path planning

Using the above parameters, the orientation of the robot hand in Cartesian space has been planned as:

$$[\ n \quad o \quad a\ ] = \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

And the positions of the robot arm, together with the robotic hand joint space angle, have been planned as:

N	X	Y	Z	θ_p	θ_r	θ_{F1} (θ_{F2})
0	-347.25	0	263.00	0	0	77.16
1	-347.25	0	283.00	0	0	77.16
2	-347.25	0	303.00	0	0	77.16
3	-347.25	0	323.00	0	0	77.16
4	-347.25	0	343.00	0	0	77.16
5	-347.25	0	363.00	0	0	77.16

6-4 Control organizing module

The robot control organizing module includes:

* Identify the task and go to right control module

* map the planned parameters into the robot joint space or Cartesian space according to the task executed.

* execute the task planned

* monitor and display the operation of the system

* continue the task execution if the error occured is within the specified range

* Branch the control into the error-correction module if the error occurred is intolerable.

Fig. 6.8 shows the functions of the control organizing module.

a. <u>identification of the task</u>

The identification of the task is carried out by checking the contents in the planned parameters -- ACT$ and HAND$.

The massaging action is defined in ACT$. For kneading opeartion, ACT$="KNEAD". While for padding operation, ACT$="PAD".

The robotic hand is specified by HAND$. The HAND-I should be used when HAND$="HANDOLD". While the HAND-II should be used when HAND$="HANDNEW".

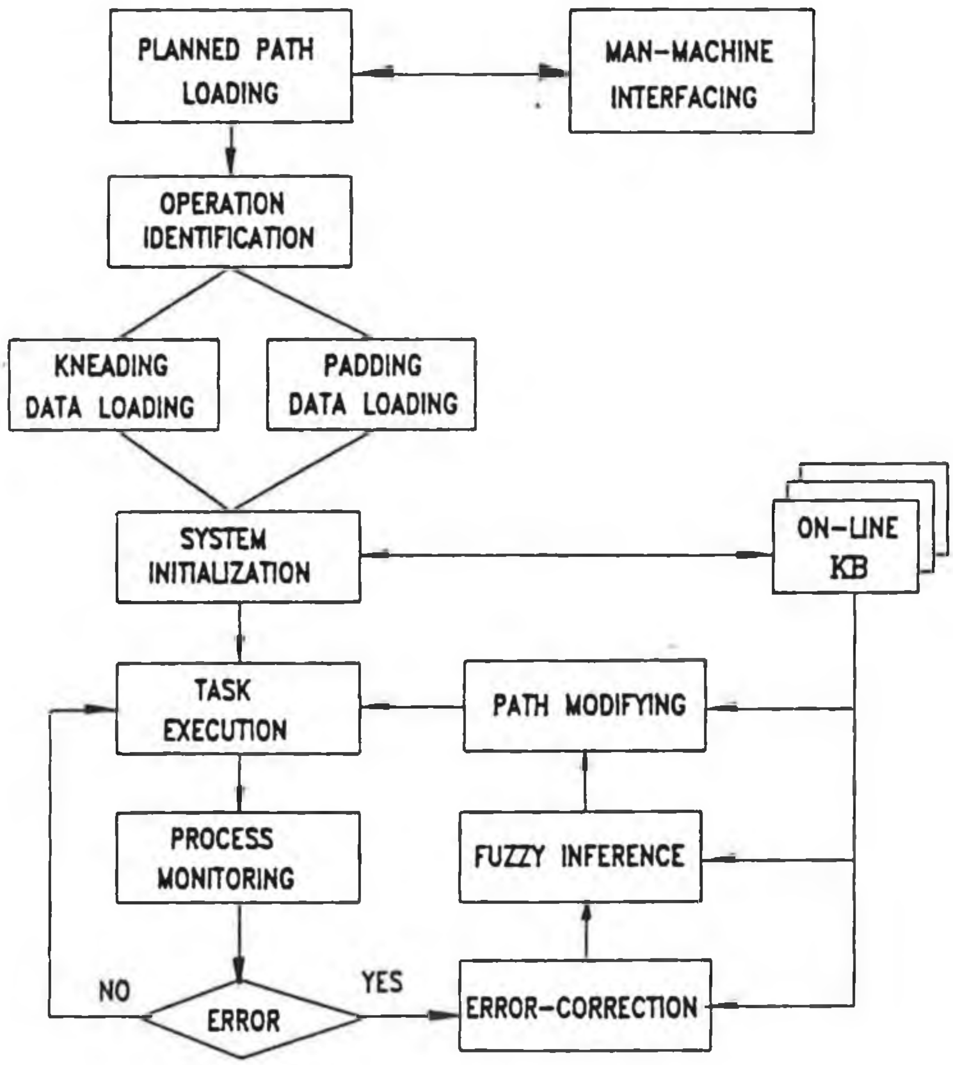


Fig. 6.8 Functions of the control organizing module

b. <u>planned parameters mapping</u>

The planned parameters must be mapped into the buffer of the control module. For kneading operation, the position parameters [X Y Z θ_p θ_r θ_{F1} θ_{F2}] at the ith massaging path for the robotic hand are mapped into the control buffers of:

* robotic finger position control space [θ_{F1} θ_{F2}]
* robotic arm position control space [X Y Z θ_p θ_r]

The robot arm speed (Speed$) is mapped into the control buffer of the robot arm speed controller.

The massaging force (FForce) is mapped into the control buffer of the robotic finger force control space.

The massaging path number , together with the massaging point number, are loaded into the control loop. And the number of the massaging points along a radial path in a kneading operation means the massaging repeat times of the robotic fingertips at the same massaging position.

The orientation vectors of the robotic hand in the cartesian space are mapped into the buffer of the error-correction module. Once error-correction is required, the orientation vectors of the robotic hand will be used to find the new position of the robotic hand.

For the padding operation, the robotic palm are used to apply the required force onto the part. The position parameters [X Y Z θ_p θ_r] are still mapped into the robot arm position control buffer. The massaging force (FForce) is mapped into the compliance motion control buffer as a condition to be evaluated. The fine motion to achieve the massaging force will be commanded by the compliance loop according to the sensed information.

c. <u>Task execution</u>

The task execution is carried out by using the data in the control buffers. By interfacing with the robot controller and the robotic hand controller, the PC controls and monitors the whole massaging system.

For the motion control of the robot arm, the built-in codes, such as "MP", etc., are used to initiate and control the motion of the robot arm. While the motion control of the robot fingers is carried out by activating the robot hand position servo loop.

The force control of the robotic fingertips is realized by activating the robotic hand force servo loop.

Fig. 6.9 shows the task execution for the kneading operation. While the padding operation is shown in Fig. 6.10.

During the task execution, the operation is also under close watch by the PC. The current position/force of the robot hand, together with the fuzzy inference process, are displayed on the screen.

Fig. 6.11 shows the on-line display of the sensory information.

The task execution will continue if there is not intolerable errors during the operation process.

However, the on-line error-correction based on fuzzy logic will be activated if the error exceeds the tolerable limitations during a task execution process.

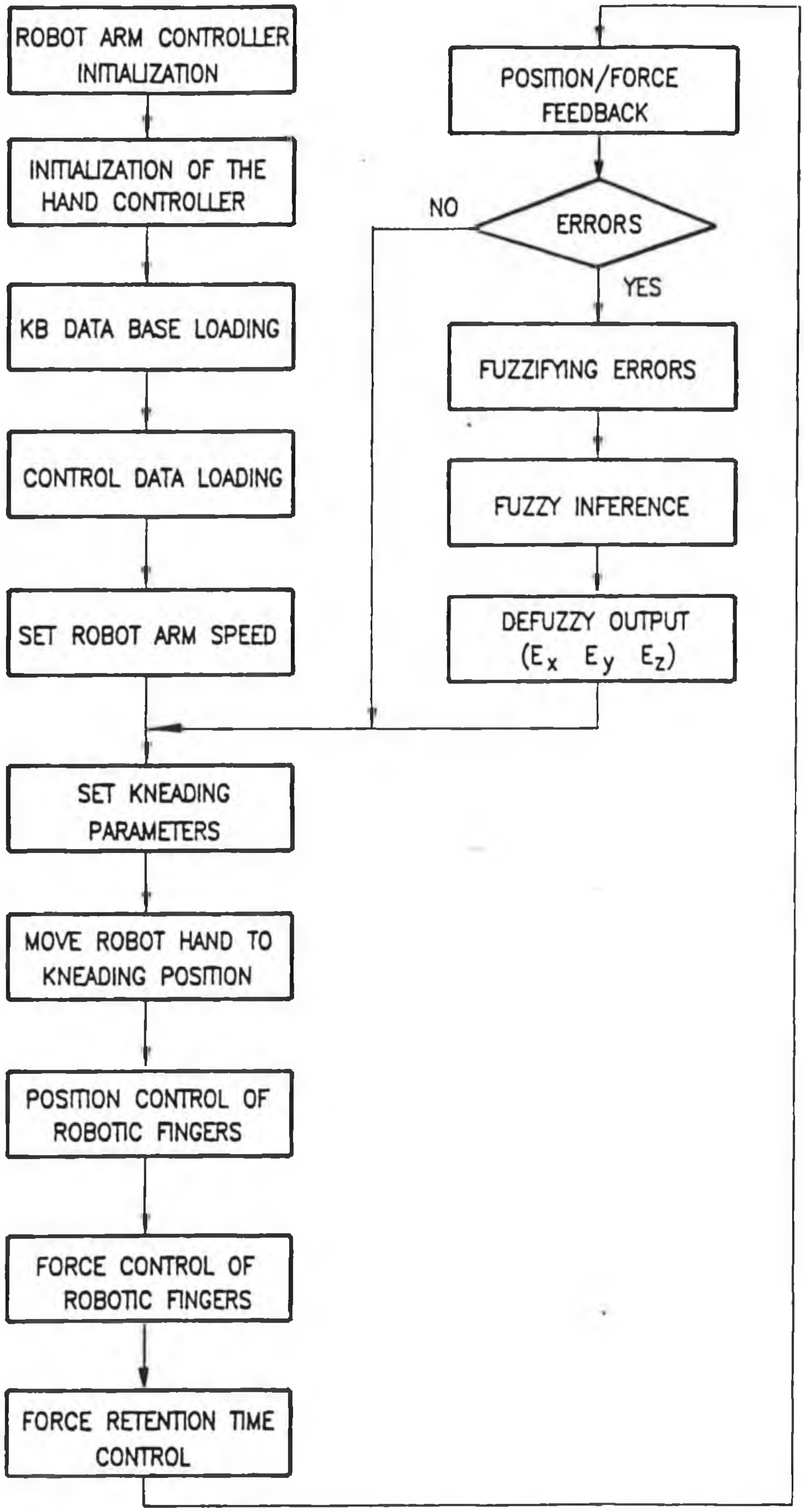


Fig. 6.9 Task execution for kneading operations

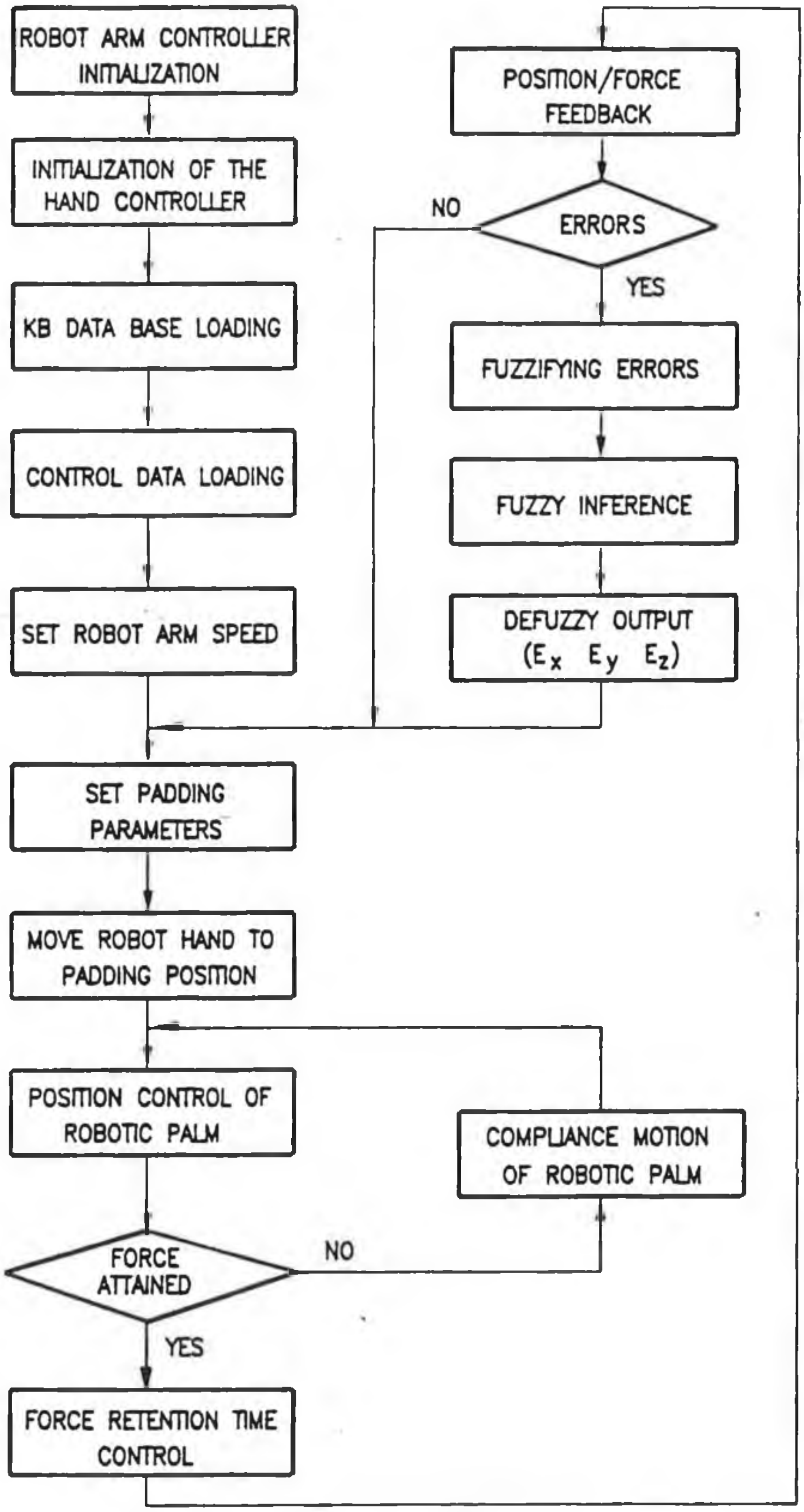


Fig. 6.10 Task execution for padding operations

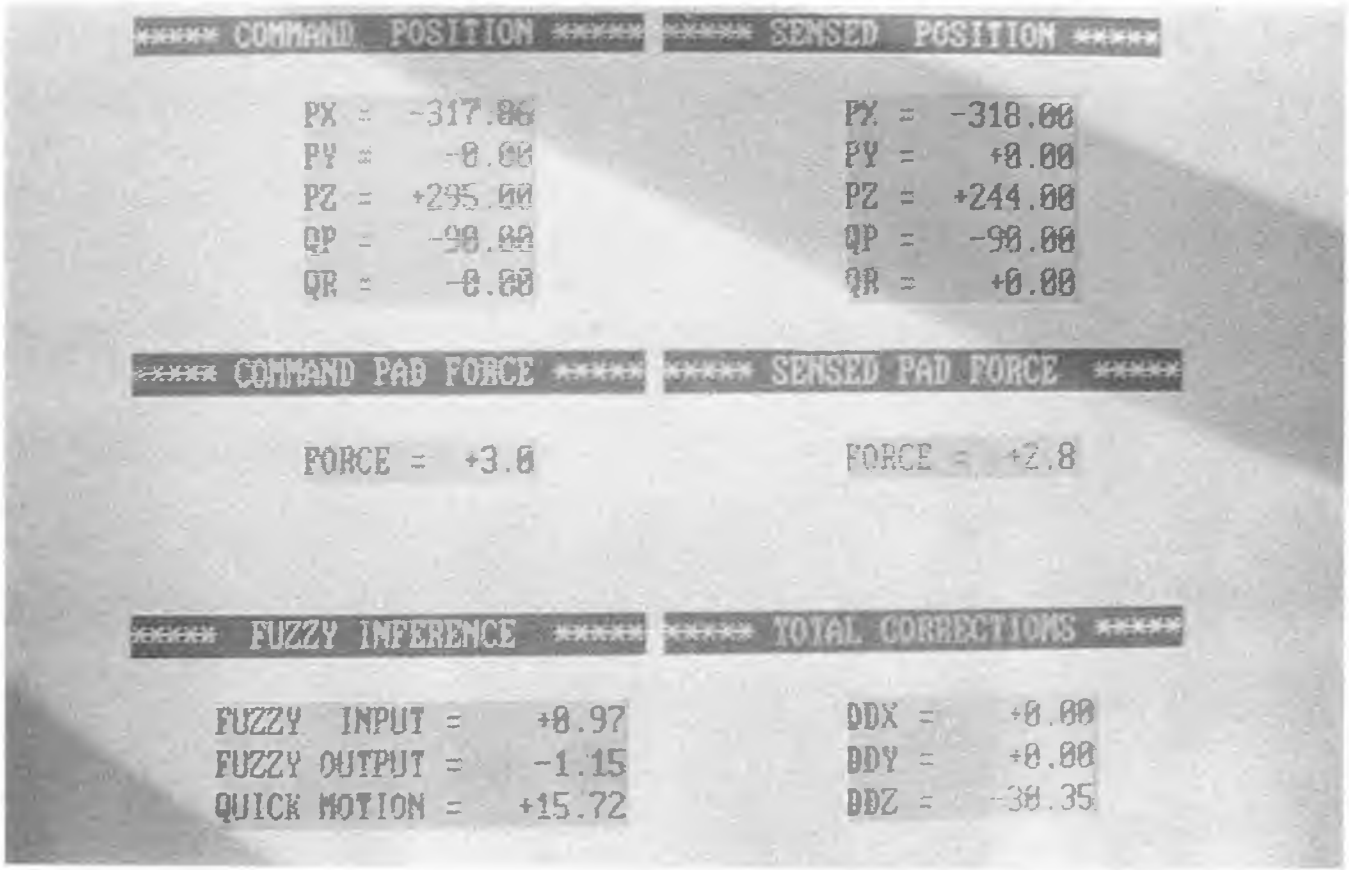


Fig. 6.11 On-line display of the sensory information

6-5 On-line error-correction

The robotic massaging performs well provided that:

a. The location of the part being massaged is accurately specified
b. The massaging path is well planned
c. The part being massaged does not deviate from its original position during the massaging process
d. There is no unpredictable obstacle during the massaging operations

However, these conditions can not be guaranteed in the practical operations. The robotic massaging system must be endowed with the abilities to carry out on-line error-corrections.

The on-line error-corrections include two tasks:

* error-detections
* error-corrections

An error-correction process is defined as a process of adjusting the robot hand to the actual massaging position of the part being massaged from its planned position, if there is an intolerable error between the planned position and the actual position of the part.

The feedback of the sensory information of the robot system is used to perform the error-detections, while the error-corrections are carried out by manipulating the on-line KB, the fuzzy inference module, the path modifying module and the intelligent control module.

Incorporated with the error-correction module, the on-line intelligent control system for the robotic massaging system can be organized as shown in Fig. 6.12.

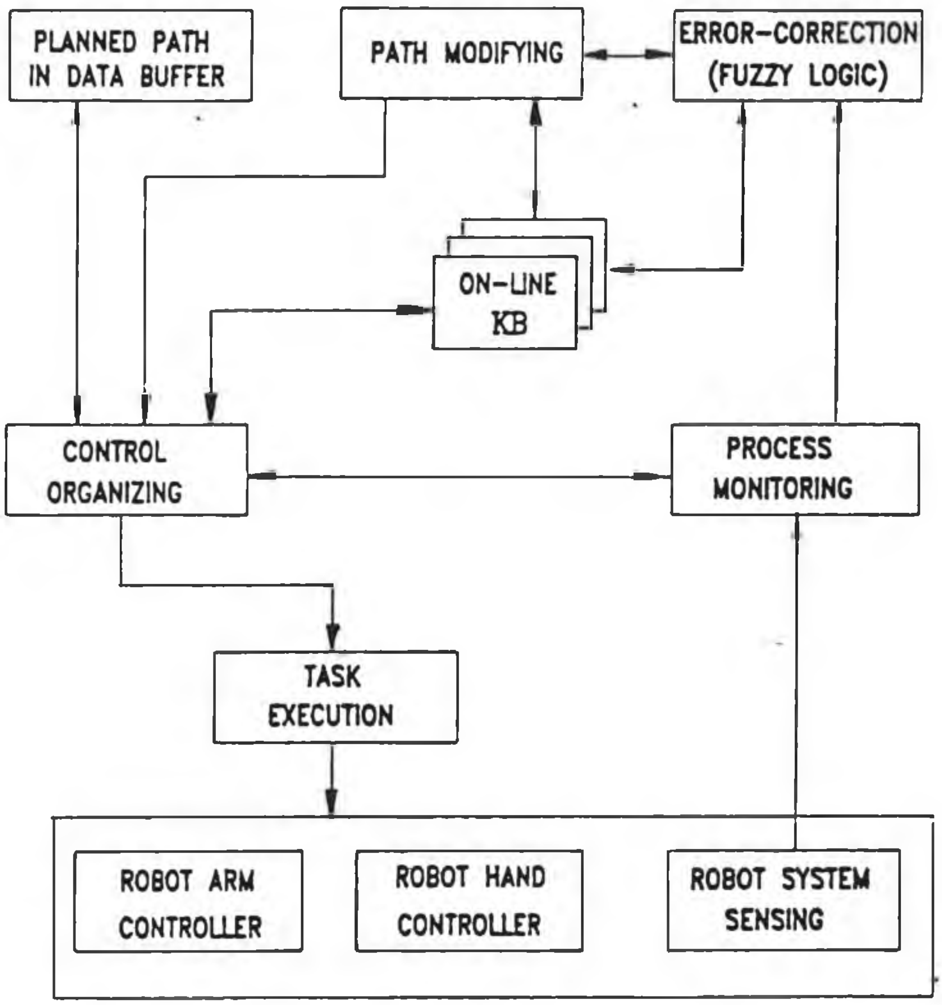


Fig. 6.12 On-line intelligent control system

6-5-1 Error types and correction equations

The error between the actual massaging position and the planned massaging position can be classified as path misplanning errors.

Depending on its extent, a path misplanning error may cause several problems:

* damaging the robot arm due to collisions
* hurting the part being massaged
* massaging the part in a wrong position

The path misplanning errors may be caused by any one of the factors, such as:

a. the location of the part is wrong specified
b. the part deviates from its specified position
c. the path is incorrectly planned

<u>A. Analysis of the errors in Cartesian space</u>

For a massaging operation using the robotic hand, the position and orientation of the robotic hand with respect to the world frame has been specified by a position task matrix T_0^t. The motion control for the robotic hand to follow the specified path has been discussed in section 6-5. Where the compliance (kneading or padding) frame is required to maintain the same position and orientation as the task frame.

The robotic massaging will be carried out smoothly if there is no error occured during the massaging process. However, certain errors may exist during a massaging process, such as path misplanning errors.

Let the position and orientation of the robotic hand along the planned axial path be denoted by a task matrix T_0^t, and the position and orientation of the robotic hand along the actual axial massaging path be denoted by a matrix T_0^m. Also, a task frame $O_tX_tY_tZ_t$ is attached to the planned grasping center O_t and a massaging frame $O_mX_mY_mZ_m$ is attached to the actual grasping center O_m.

Thus, the path misplanning errors can be illustrated in Fig. 6.13.

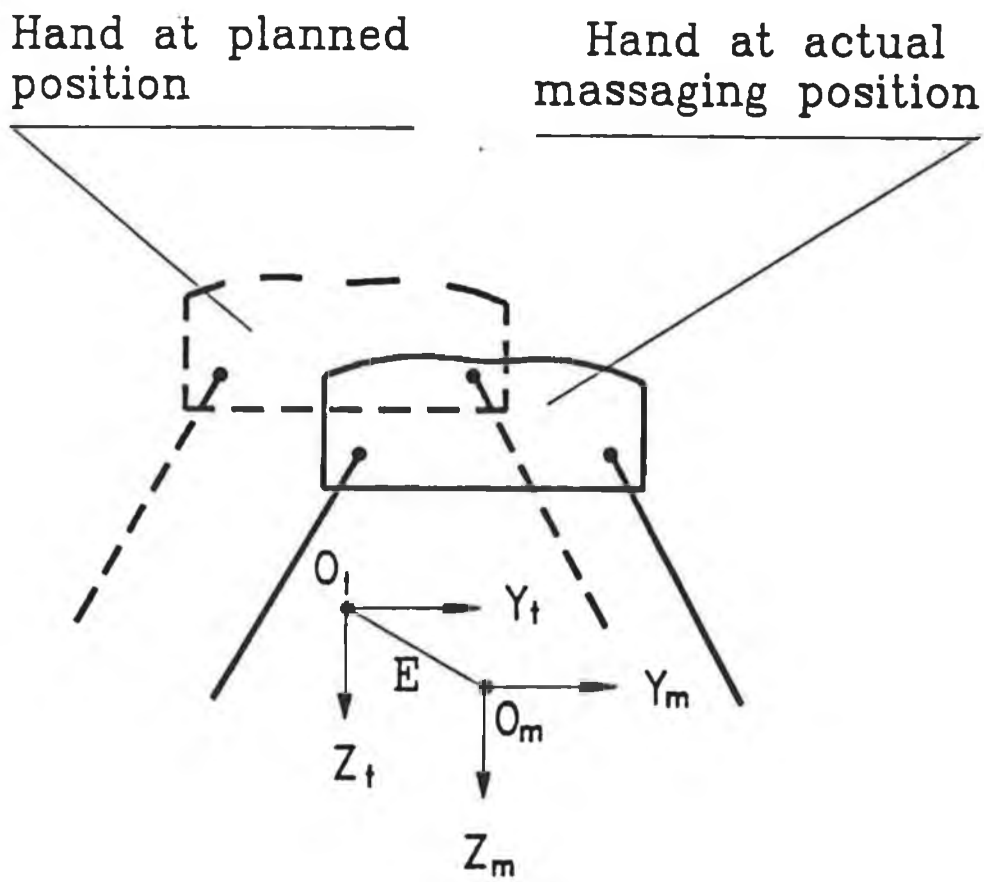


Fig. 6.13 Path misplanning errors

Where

O_t - origin of the task frame, $O_tX_tY_tZ_t$, along the planned axial path

O_m - origin of the massaging frame, $O_mX_mY_mZ_m$, along the actual axial path

E_X - error between O_c and O_m along X_t

E_Y - error between O_c and O_m along Y_t

E_Z - error between O_c and O_m along Z_t

To carry out the massaging operation, the robotic hand should be moved from the planned position (O_t) to the actual massaging position (O_m) so that the errors occured can be corrected. Hence, the following motion control equation must be maintained:

$$T_0^c = T_0^m \qquad (6\text{-}7)$$

Where, T_0^c is the position and orientation of the compliance frame of the robotic hand with respect to the world frame. And T_0^m is the position and orientation of the massaging frame with respect to the world frame.

Assume that the position and orientation of the robotic hand along the planned axial path are given by:

$$T_0^t = \begin{bmatrix} n_x^t & o_x^t & a_x^t & p_x^t \\ n_y^t & o_y^t & a_y^t & p_y^t \\ n_z^t & o_z^t & a_z^t & p_z^t \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6\text{-}8)$$

When the robotic hand is moved to the actual massaging position O_m, its position and orientation with respect to the task frame, $O_tX_tY_tZ_t$, may be denoted by:

$$T_t^m = \begin{bmatrix} 1 & 0 & 0 & E_X \\ 0 & 1 & 0 & E_Y \\ 0 & 0 & 1 & E_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6\text{-}9)$$

Thus, the position/orientation of the robotic hand at the actual massaging position with respect to the world frame is given by

$$T_0^m = T_0^t \, T_t^m \qquad (6\text{-}10)$$

Recall that

$$T_0^c = T_0^a \, T_a^c \qquad (6\text{-}11)$$

Where, T_0^a and T_a^c have been defined in eq.(5-11).

Referring to eq.(6-7), and combining eq.(6-10) with eq.(6-11), one may obtain

$$T_0^a = T_0^t \, T_t^m \, (T_a^c)^{-1} \qquad (6\text{-}12)$$

Hence, for the path misplanning errors (E_X E_Y E_Z), the robotic wrist position after error-corrections can be denoted by:

$$T_0^a = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6\text{-}13)$$

Where

$$\begin{aligned} n_x &= n_x^t & o_x &= o_x^t & a_x &= a_x^t \\ n_y &= n_y^t & o_y &= o_y^t & a_y &= a_y^t \\ n_z &= n_z^t & o_z &= o_z^t & a_z &= a_z^t \end{aligned} \qquad (6\text{-}14)$$

And

$$\begin{aligned} p_x &= p_x^t - ({}^cp_x n_x + {}^cp_y o_x + {}^cp_z a_x) + (E_X n_x + E_Y o_x + E_Z a_x) \\ p_y &= p_y^t - ({}^cp_x n_y + {}^cp_y o_y + {}^cp_z a_y) + (E_X n_y + E_Y o_y + E_Z a_y) \\ p_z &= p_z^t - ({}^cp_x n_z + {}^cp_y o_z + {}^cp_z a_z) + (E_X n_z + E_Y o_z + E_Z a_z) \end{aligned} \qquad (6\text{-}15)$$

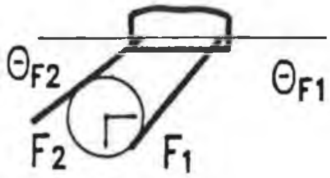
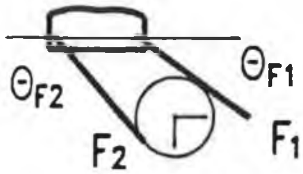
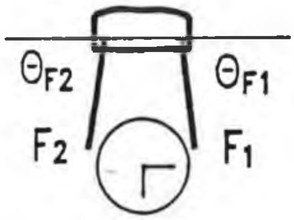
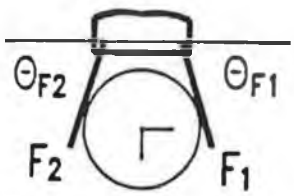
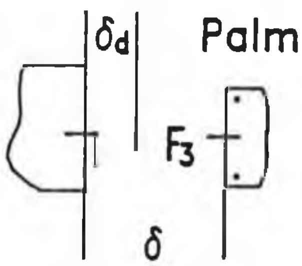
B. Error types and error-detections

According to the error directions, the errors can be classified into three types:

a. offset along X_t axis (E_X)
b. offset along Y_t axis (E_Y)
c. offset along Z_t axis (E_Z)

Table 6.15 shows several typical errors and their detections.

Table 6.15 Errors and detections

Error type	Illustration	Conditions detected	Corrections E_X	E_Y	E_Z
Knead $+Y_t$		$\theta_{F1} >> \theta_{F2d}$ $\theta_{F2} << \theta_{F2d}$ $F_1 <> F_2$	0	-E	0
Knead $-Y_t$		$\theta_{F1} << \theta_{F1d}$ $\theta_{F2} >> \theta_{F2d}$ $F_1 <> F_2$	0	+E	0
Knead $-Z_t$		$\theta_{F1} >> \theta_{F1d}$ $\theta_{F2} >> \theta_{F2d}$ $F_1 = F_2 \geq 0$	0	0	+E
Knead $+Z_t$		$\theta_{F1} << \theta_{F1d}$ $\theta_{F2} << \theta_{F2d}$ $F_1 = F_2 \geq 0$	0	0	-E
Pad $-Z_t$		$\delta >> \delta_d$ $F_3 = 0$	0	0	+E
Pad $+Z_t$		$\delta << \delta_d$ $F_3 > 0$	0	0	-E

For the kneading operations using the robotic fingers, there exist uncertainties in the error detection process due to:

* The contact points between the robotic fingers and the part being contacted are not exactly known, since the force/tactile sensors are mounted on the fingertips.

* The part size may be different from the specified size.

To deal with the uncertainties, the fuzzy inference and the human massaging knowledge are employed in this case. By manipulating the detected information and the expert knowledge base embedded in the fuzzy inference rule base and data base, the required control strategies and error-correction distances (E_X E_Y E_Z) can be obtained. And the error-correction can be carried out on-line.

6-5-2 Fuzzy logic based error-corrections

Based on the human massaging knowledge and the human inference process, the fuzzy error-correction strategies may be developed, which include:

<u>Criterion 1.</u>

IF { the offset (E_X E_Y E_Z) is very small }

THEN { no correction is required }

Criterion 2.

IF { the offset (E_X E_Y E_Z) is [Small, Medium, Big] }

THEN { move the robot hand a [Small, Medium, Big] distance along the opposite direction of the offset (to reduce the offset) }

A. Universes of discourse

Let the angle errors of the robotic fingers be denoted by

$$E\Theta_1 = \Theta_{F1d} - \Theta_{F1}$$
$$E\Theta_2 = \Theta_{F2d} - \Theta_{F2} \qquad (6\text{-}16)$$

For kneading operations, the inputs to the fuzzy error-correction inference mechanism are $E\Theta_1$ and $E\Theta_2$. And the input to the fuzzy inference mechanism is δ for padding operation.

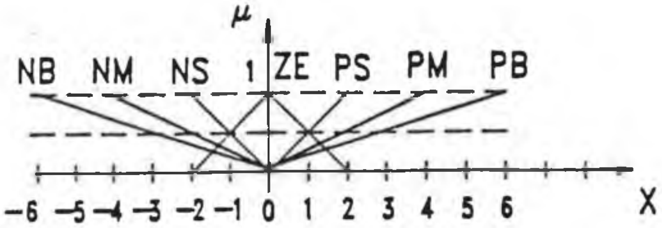
The outputs of the fuzzy inference mechanism are a set of correction distances, i.e., (E_X E_Y E_Z).

Hence, the universes of discourse can be classified into two types:

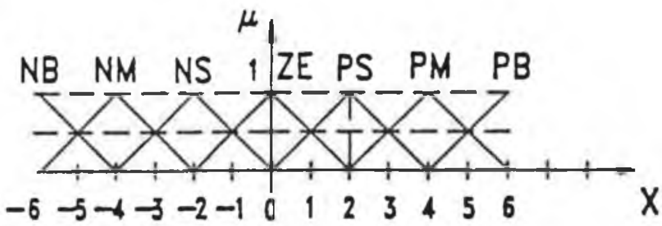
* process input variables -- ($E\Theta_1$ $E\Theta_2$ δ)

* control output variables -- (E_X E_Y E_Z)

For the universes of discourse ($E\Theta_1$ $E\Theta_1$ δ E_X E_Y E_Z), the universe partition and the membership function are defined as shown in Fig. 6.14.



a). X = [E_X E_Y E_Z]



b). X = [$E\theta_1$ $E\theta_2$ δ]

Fig. 6.14 Membership function and universe partition

The mapping scaling factors are shown in Table 6.16.

Table 6.16 Mapping scalers

Universe of discourse	Mapping scaler	Range
$E\theta_1$	$FK\theta = 3^0$	$[-18^0, +18^0]$
$E\theta_2$	$FK\theta = 3^0$	$[-18^0, +18^0]$
δ	$FK\delta$ = 5 mm	[-30, +30] mm
E_X	4 mm (knead)	
E_Y	FKO =	[-30, +30] mm
E_Z	5 mm (pad)	

B. Fuzzification of input variables

Before the fuzzy inference process is carried out, the measured input variables must be fuzzified into suitable linguistic values, which may be viewed as labels of fuzzy sets.

Table 6.17 shows the fuzzifications for the input variables ($E\Theta_1$ $E\Theta_2$ δ). Where, $FX = E\Theta_i/FK\Theta$ for kneading operation. And $FX = \delta/FK\delta$ for padding operation.

Table 6.17 Input variable fuzzifications

Universe of discourse after scaling map	Primary fuzzy sets (Fuzzy terms)
$FX < -5$	NB (Negative Big)
$-5 \leq FX < -3$	NM (Negative Medium)
$-3 \leq FX < -1$	NS (Negative Small)
$-1 \leq FX \leq +1$	ZE (Zero)
$+1 < FX \leq +3$	PS (Positive Small)
$+3 < FX \leq +5$	PM (Positive Medium)
$+5 < FX$	PB (Positive Big)

C. Fuzzy control rules

By using Criterion 1 and Criterion 2, the fuzzy control rules for padding operations have been formulated as shown in Table 6.18. And the fuzzy control rules for kneading operations have been formulated as shown in Table 6.19.

Table 6.18 Fuzzy control rules for padding

Rule No.	Rule base for padding operations	
RP1	IF δ = PS	THEN (E_X =ZE E_Y =ZE E_Z = NS)
RP2	IF δ = PM	THEN (E_X =ZE E_Y =ZE E_Z = NM)
RP3	IF δ = PB	THEN (E_X =ZE E_Y =ZE E_Z = NB)
RP4	IF δ = NS	THEN (E_X =ZE E_Y =ZE E_Z = PS)
RP5	IF δ = NM	THEN (E_X =ZE E_Y =ZE E_Z = PM)
RP6	IF δ = NB	THEN (E_X =ZE E_Y =ZE E_Z = PB)

From Table 6.18, the basic ideas behind the fuzzy control rules may be generalized as:

* If the palm surface is far from the actual surface of the part being padded, move the palm closer to the part surface.
* If the palm surface is too close to the part surface, move the palm away from the surface.

Table 6.19 Fuzzy control rules for kneading

Rule No.	Rule base for kneading operations
RK1	IF $E\Theta_1$=NM AND $E\Theta_2$=PM THEN (E_X=ZE E_Y=NM E_Z=ZE)
RK2	IF $E\Theta_1$=NB AND $E\Theta_2$=PB THEN (E_X=ZE E_Y=NB E_Z=ZE)
RK3	IF $E\Theta_1$=PM AND $E\Theta_2$=NM THEN (E_X=ZE E_Y=PM E_Z=ZE)
RK4	IF $E\Theta_1$=PB AND $E\Theta_2$=NB THEN (E_X=ZE E_Y=PB E_Z=ZE)
RK5	IF $E\Theta_1$=NM AND $E\Theta_2$=NM THEN (E_X=ZE E_Y=ZE E_Z=PM)
RK6	IF $E\Theta_1$=NB AND $E\Theta_2$=NB THEN (E_X=ZE E_Y=ZE E_Z=PB)
RK7	IF $E\Theta_1$=PM AND $E\Theta_2$=PM THEN (E_X=ZE E_Y=ZE E_Z=NM)
RK8	IF $E\Theta_1$=PB AND $E\Theta_2$=PB THEN (E_X=ZE E_Y=ZE E_Z=NB)

From Table 6.19, the basic ideas behind the fuzzy control rules for kneading operations can be generalized as:

* If the part is not at the planned position and the robotic hand is still at the planned position, move the robotic hand to the actual massaging position.

* The robotic hand moves in such a way that the part is always to be centralized in the compliance frame of the robotic hand (to ensure the fully contacts between the fingertips and the part being massaged).

D. Fuzzy reasoning and defuzzification strategy

In contrast to a classical inference system, all fuzzy control rules are considered to be fired with different strength in the fuzzy reasoning process. Of course, rules that fire strongly will contribute significantly to the final control action.

In this study, the fuzzy reasoning based on fuzzy logic is employed. For any fuzzy input term, the fire strength for condition fuzzy terms in the control rule bases, which are given in Table 6.18 and Table 6.19, can be designed as shown in Table 6.20.

Table 6.20 Fire strength for control rules

Input fuzzy terms	Fire strength for condition fuzzy terms 0 NB	1 NM	2 NS	3 ZE	4 PS	5 PM	6 PB
NB	1.0	0.3	0.0	0.0	0.0	0.0	0.0
NM	0.3	1.0	0.3	0.0	0.0	0.0	0.0
NS	0.0	0.3	1.0	0.3	0.0	0.0	0.0
ZE	0.0	0.0	0.3	1.0	0.3	0.0	0.0
PS	0.0	0.0	0.0	0.3	1.0	0.3	0.0
PM	0.0	0.0	0.0	0.0	0.3	1.0	0.3
PB	0.0	0.0	0.0	0.0	0.0	0.3	1.0

Hence, any element of the fire strength in Table 6.20 can be denoted by SFIRE(IN_FUZZ, CON_FUZZ).

Where

IN_FUZZ is the input fuzzy term which belongs to (NM,NB,NS,ZE,PS,PM,PB).

CON_FUZZ is the condition fuzzy term in the rule base.

Let the condition fuzzy terms, (NB,NM,NS,ZE,PS,PM,PB), be denoted by (0,1,2,3,4,5,6). Thus, for any input fuzzy term IN_FUZZ, a fire strength vector for the basic condition fuzzy terms used in a rule base can be obtained from Table 6.20:

$$[V\_SFIRE] = \begin{bmatrix} SFIRE(IN\_FUZZ,0) \\ SFIRE(IN\_FUZZ,1) \\ SFIRE(IN\_FUZZ,2) \\ SFIRE(IN\_FUZZ,3) \\ SFIRE(IN\_FUZZ,4) \\ SFIRE(IN\_FUZZ,5) \\ SFIRE(IN\_FUZZ,6) \end{bmatrix} \qquad (6\text{-}17)$$

Depending on how many control rules have been constructed in a rule base, the dimension of a strength vector for the rule base is decided by the number of the rules.

Thus, the fire strength vector for the padding control rule base can be denoted by:

$$[W\_SFIRE] = \begin{bmatrix} SFIRE(IN\_FUZZ,4) \\ SFIRE(IN\_FUZZ,5) \\ SFIRE(IN\_FUZZ,6) \\ SFIRE(IN\_FUZZ,2) \\ SFIRE(IN\_FUZZ,1) \\ SFIRE(IN\_FUZZ,0) \end{bmatrix} \qquad (6\text{-}18)$$

Where, [W_SFIRE] is the fire strength for all the control rules in the padding rule base.

The fire strength vector for kneading control rule base can be denoted by:

$$[W\_SFIRE] = MIN\{ [W\_SFIRE\_\theta_1], [W\_SFIRE\_\theta_2] \} \qquad (6\text{-}19)$$

Where

$$[W\_SFIRE] = \begin{bmatrix} W\_SFIRE(0) \\ W\_SFIRE(1) \\ W\_SFIRE(2) \\ W\_SFIRE(3) \\ W\_SFIRE(4) \\ W\_SFIRE(5) \\ W\_SFIRE(6) \\ W\_SFIRE(7) \end{bmatrix} \qquad (6\text{-}20)$$

$$[W\_SFIRE\_\theta_1] = \begin{bmatrix} SFIRE(IN\_FUZZ\_\theta_1,1) \\ SFIRE(IN\_FUZZ\_\theta_1,0) \\ SFIRE(IN\_FUZZ\_\theta_1,5) \\ SFIRE(IN\_FUZZ\_\theta_1,6) \\ SFIRE(IN\_FUZZ\_\theta_1,1) \\ SFIRE(IN\_FUZZ\_\theta_1,0) \\ SFIRE(IN\_FUZZ\_\theta_1,5) \\ SFIRE(IN\_FUZZ\_\theta_1,6) \end{bmatrix} \tag{6-21}$$

$$[W\_SFIRE\_\theta_2] = \begin{bmatrix} SFIRE(IN\_FUZZ\_\theta_2,5) \\ SFIRE(IN\_FUZZ\_\theta_2,6) \\ SFIRE(IN\_FUZZ\_\theta_2,1) \\ SFIRE(IN\_FUZZ\_\theta_2,0) \\ SFIRE(IN\_FUZZ\_\theta_2,1) \\ SFIRE(IN\_FUZZ\_\theta_2,0) \\ SFIRE(IN\_FIRE\_\theta_2,5) \\ SFIRE(IN\_FUZZ\_\theta_2,6) \end{bmatrix} \tag{6-22}$$

And

$IN\_FUZZ\_\theta_i$ is the ith finger angle error fuzzy input.

$[W\_SFIRE\_\theta_i]$ is the fire strength for the ith finger condition fuzzy terms in the rule base.

Thus, the control action [Y] can be expressed as:

$$[Y] = \frac{[W\_SFIRE]^T * [E]}{\Sigma\ W\_SFIRE(i)} * FKO \qquad (6\text{-}23)$$

Where

[Y] -- defuzzified control action, and

$$[Y] = [E_X\ E_Y\ E_Z]$$

[E] -- crisp outcome of the control rule base

FKO -- output mapping scaler

[W_SFIRE] -- fire strength for the control rule base

E. Example

As an example, let's find the correction action [Y] for a kneading operation under fuzzy inputs:

$IN\_FUZZ\_\Theta_1$ = "NM"
$IN\_FUZZ\_\Theta_2$ = "PM"

Here the kneading control rule base is used.

Solution:

For the inputs $IN\_FUZZ\_\Theta_1$ = "NM" and $IN\_FUZZ\_\Theta_2$ = "PM", referring to the fire strength given in Table 6.20, the fire strength for the kneading control rule base can be obtained by using eqs.(6-21) and (6-22).

Where

$$[W\_SFIRE\_\theta_1] = [1,\ 0.3,\ 0,\ 0,\ 1,\ 0.3,\ 0,\ 0]^T \qquad (6\text{-}24)$$

$$[W\_SFIRE\_\theta_2] = [1,\ 0.3,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0.3]^T \qquad (6\text{-}25)$$

Thus, referring to eq.(6-19), one may obtain:

$$\begin{bmatrix} W\_SFIRE(0) \\ W\_SFIRE(1) \\ W\_SFIRE(2) \\ W\_SFIRE(3) \\ W\_SFIRE(4) \\ W\_SFIRE(5) \\ W\_SFIRE(6) \\ W\_SFIRE(7) \end{bmatrix} = \begin{bmatrix} 1 \\ 0.3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad (6\text{-}26)$$

Note that for the kneading operation, [E] is denoted by:

$$[E] = \begin{bmatrix} 0 & -5 & 0 \\ 0 & -6 & 0 \\ 0 & 5 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 5 \\ 0 & 0 & 6 \\ 0 & 0 & -5 \\ 0 & 0 & -6 \end{bmatrix} \qquad (6\text{-}27)$$

Substituting eqs.(6-26) and (6-27) into eq.(6-23), one may obtain the correction actions:

$$[Y] = \begin{bmatrix} E_X \\ E_Y \\ E_Z \end{bmatrix}^T = \begin{bmatrix} 0 \\ -26.15 \\ 0 \end{bmatrix}^T \qquad (6\text{-}28)$$

6-5-3 On-line KB

The on-line knowledge base consists of three parts: Robot joint space coordinating KB, error-correction KB and robot adaptive KB.

A. Robot joint space coordinating KB

For the planned path (position/force), a coordinating or mapping from the cartesian space to the robot joint space is required. A coordinating KB can be constructed as shown in Table 6.21.

Table 6.21 Coordinating KB

Operations	Position	Force
Kneading	$[X,Y,Z,\theta_{234},\theta_5,\theta_{F1},\theta_{F2}]^T$	$[F_1, F_2]^T$
Padding	$[X,Y,Z,\theta_{234},\theta_5]^T$	$[F_3]$

B. Error-correction KB

The error-correction KB includes:

a. error correction strategies
b. fire strength vectors for rule bases
c. rule base and data base for fuzzy inference

The error correction strategy KB is shown in Table 6.22.

Table 6.22 Error Correction strategy KB

Fuzzy mode	Fuzzy terms [E]			Crisp terms [E]		
Rule No.	E_X	E_Y	E_Z	E_X	E_Y	E_Z
RK1	ZE	NM	ZE	0	-5	0
RK2	ZE	NB	ZE	0	-6	0
RK3	ZE	PM	ZE	0	+5	0
RK4	ZE	PB	ZE	0	+6	0
RK5	ZE	ZE	PM	0	0	+5
RK6	ZE	ZE	PB	0	0	+6
RK7	ZE	ZE	NM	0	0	-5
RK8	ZE	ZE	NB	0	0	-6
RP1	ZE	ZE	NS	0	0	-3
RP2	ZE	ZE	NM	0	0	-4
RP3	ZE	ZE	NB	0	0	-5
RP4	ZE	ZE	PS	0	0	+3
RP5	ZE	ZE	PM	0	0	+4
RP6	ZE	ZE	PB	0	0	+5

The fire strength vector KB for padding rule base can be constructed by using eq.(6-18). And the fire stength vector KB for kneading rule base can be formulated by using eq.(6-19).

The rule bases in Table 6.18 and Table 6.19 can be incorporated into the rule base KB. And the data base KB can be established by using Table 6.16 and Table 6.17.

C. Robot adaptive KB

For any inferred correction (E_X E_Y E_Z), there are two ways to carry out the corrections:

a. Direct modifying.
   Where no verifying is involved. The path is directly modified.

b. Verifying and modifying.
   Where the feasibility of the inferred new position is verified first by using the kinematics of the robot. If feasible, the path will be modified accordingly. Otherwise, the inferred corrections will be either discarded or modified.

Since the position of the robot hand is modified to adapt the position change of the part being massaged, the path modifying process is also termed as robot adaptive process. And the KB used to assist the path modifying is referred to robot adaptive KB.

For the direct modifying strategy, the correction equation of the robotic hand can be incorporated into the robot adaptive KB. While for the verifying and modifying strategy, both the correction equation and the inverse kinematics of the robotic hand can be incorporated into the robot adaptive KB. Table 6.23 lists the robot adaptive KB.

Table 6.23 Robot adaptive KB

Correction equation (for hand grasping center):

$$X = P_x^t + (E_X n_x + E_Y o_x + E_Z a_x)$$

$$Y = P_y^t + (E_X n_y + E_Y o_y + E_Z a_y)$$

$$Z = P_z^t + (E_X n_z + E_Y o_z + E_Z a_z)$$

$$\Theta_{234} = \Theta_{234}$$

$$\Theta_5 = \Theta_5$$

Inverse kinematics math base (for robotic wrist)

$$[\Theta_1 \ \Theta_2 \ \Theta_3 \ \Theta_4 \ \Theta_5]^T = \text{Inverse kinematics} \ (T_0^a)$$

Comparing to the inverse kinematics math base in the robot adaptive KB, the correction equation requires less computation time.

6-5-4 Realization of the on-line error-corrections

The procedure of on-line error-corrections during a massaging process can be generalized as follows:

a. move the robotic hand to the massaging position

b. move the robotic fingers into the required positions

c. apply the desired force onto the part being massaged

d. detect the massaging position and force of the robotic hand

e. check the errors. If there is error, go to step (f). Otherwise, go back to step (a) to carry out next massaging operation.

f. fuzzify the crisp inputs into the fuzzy sets

g. fuzzy inference

h. defuzzy the fuzzy output

i. calculate the new position of the robotic hand at the corrected position

j. control the robot hand to move to the correction position

k. go to step (a) to carry out next massaging operation.

Fig. 6.15 shows a general error-correction module using the fuzzy inference mechanism.

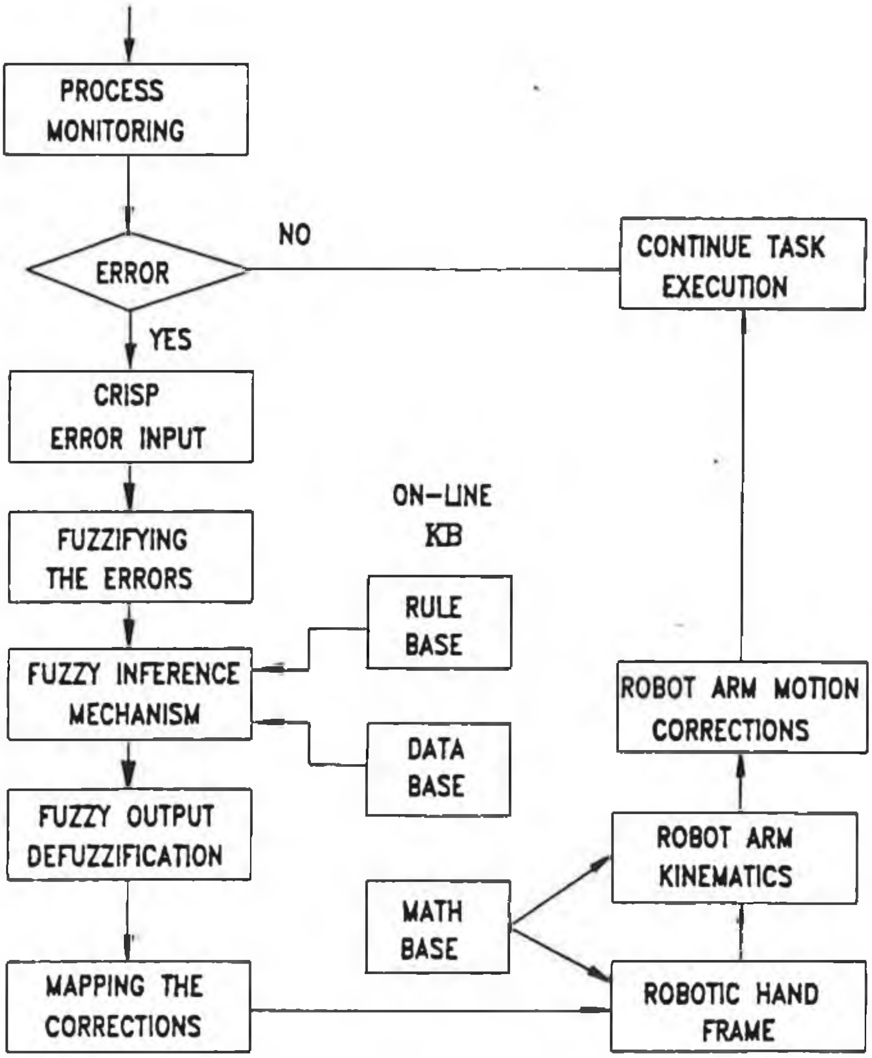


Fig. 6.15 Error-correction module

6-6 Software development and experimental results

6-6-1 Software development

The software for the intelligent robot control system have been developed, which include:

* Parameter generating and path planning software for the robot system in which both HAND-I and HAND-II are included. The source codes of the developed software, named as EXPERTP.BAS, can be found in APPENDIX F-1.

* Task execution and intelligent control software for the robot system using HAND-I. The source codes of the developed software, named as EXPERTO.BAS, can be found in APPENDIX F-2.

* Task execution and intelligent control software for the robot system using HAND-II. The source codes of the developed software, named as EXPERTN.BAS, can be found in APPENDIX F-3.

6-6-2 Experimental results

A. Padding operation

Both HAND-I and HAND-II can be used to perform the padding operations. Followed is one of the experiments carried out by using HAND-II to pad a flat foam which simulates the human back.

a. Parameter generating

For the fuzzy/liguistic inputs:

Part to be massaged	=	"BACK"
Part size	=	"SM"
Massaging type	=	"CRS"
Massaging speed	=	"ME"
Robot hand used	=	"HANDNEW"

The following parameters can be inferred by using the parameter generating KB:

Massaging action (ACT$)	=	"PAD"	
Massaging paths (N)	=	6	
Radial massaging points (m)	=	3	
Robot arm speed	=	5	
Force retention time	=	1.0	(Sec)
Massaging force	=	3.0	(N)
Width of the back	=	100	(mm)
Length of the back	=	100	(mm)

The position of the back has been specified as:

X_0	Y_0	Z_0	α	β
-280	0	190	0^0	180^0

b. Path planning

Using the generated parameters, the orientation of the robotic hand in Cartesian space has been planned as:

$$[\, n \quad o \quad a \,] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

And the positions of the robot arm, together with the control angles, have been planned as shown in Table 6.24. Where N=6 has been changed into N=8.

c. Parameter mapping

Before the task execution, the intelligent control organizing module will map all the planned parameters into the control buffer.

In this experiment, the padding has been performed by the robotic palm along the axial padding path. When the padding along the first axial path is completed, the padding will be continued in the next axial padding path.

Table 6.24 Planned positions

Position No. (m, N)	X	Y	Z	θ_p	θ_r
(0,0)	-274.9	-49.9	295	-90	-10.3
(0,1)	-289.2	-50.0	295	-90	-9.8
(0,2)	-303.5	-50.0	295	-90	-9.3
(0,3)	-317.8	-50.0	295	-90	-8.9
(0,4)	-332.1	-50.0	295	-90	-8.5
(0,5)	-346.4	-50.0	295	-90	-8.2
(0,6)	-360.7	-50.0	295	-90	-7.8
(0,7)	-374.9	-50.0	295	-90	-7.5
(1,0)	-275.0	0	295	-90	0
(1,1)	-289.2	0	295	-90	0
(1,2)	-303.5	0	295	-90	0
(1,3)	-317.8	0	295	-90	0
(1,4)	-332.1	0	295	-90	0
(1,5)	-346.4	0	295	-90	0
(1,6)	-360.7	0	295	-90	0
(1,7)	-375.0	0	295	-90	0
(2,0)	-275.0	50.0	295	-90	10.3
(2,1)	-289.2	49.9	295	-90	9.8
(2,2)	-303.5	49.9	295	-90	9.3
(2,3)	-317.8	49.9	295	-90	8.9
(2,4)	-332.1	49.9	295	-90	8.5
(2,5)	-346.4	49.9	295	-90	8.2
(2,6)	-360.7	49.9	295	-90	7.8
(2,7)	-375.0	49.9	295	-90	7.5

d. Task execution

Using the control variables in the control buffer, the padding operations can be carried out.

To ensure the safe operation, a constant approaching distance, DABOVE, above the padding surface has been introduced into the padding process. Where, DABOVE=20 mm. To increase the padding speed, a quick approach motion has also been employed in the padding control process.

To achieve the desired padding force, the compliance motion control of the palm is required. Experiment shows that two or three times of fine motions are required in every padding cycle to attain the desired force. Where the fine motion distance is 1.0 mm.

Fig. 6.16 shows the padding force-time history of the robotic palm. Where, the robotic palm moves from the position (1,0) to the position (1,7) (See Table 6.24).

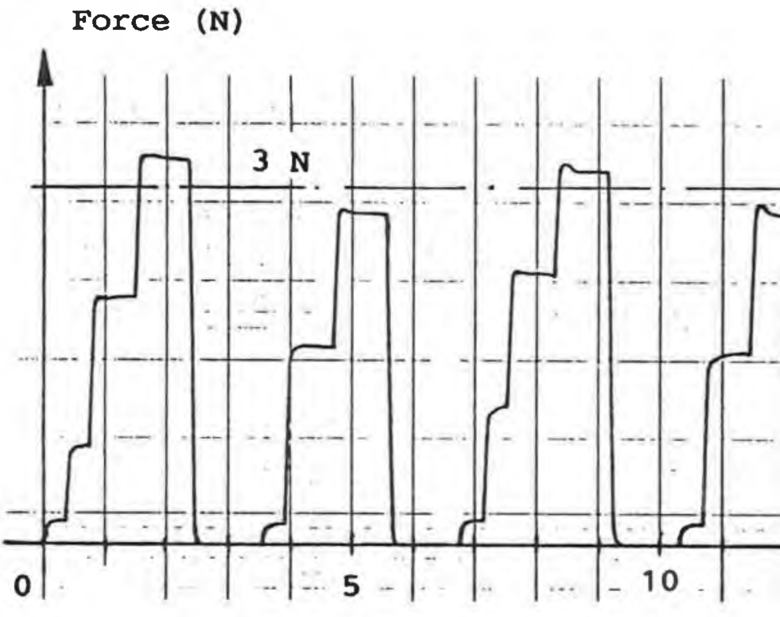
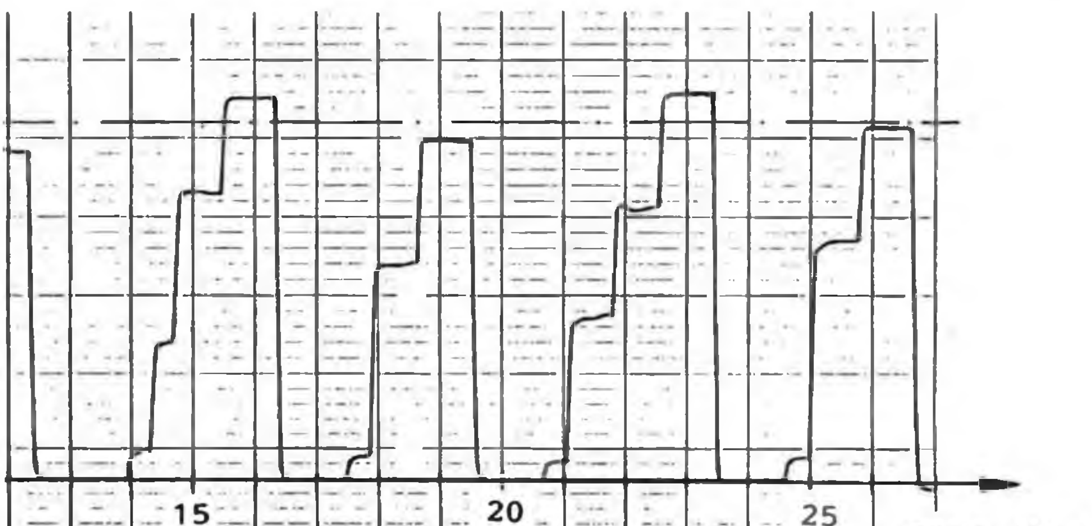


Fig. 6.16



Padding force-time history

B. Kneading operation with on-line error-correction

Both HAND-I and HAND-II can be used to perform the kneading operations. Followed is one of the experiments carried out by HAND-II. Where HAND-II is commanded to knead a cylinder foam which simulates the human arm.

a. Parameter generating

For the fuzzy/linguistic inputs:

Part to be massaged	=	"ARM"
Part size	=	"ME"
Massaging type	=	"CRS"
Massaging speed	=	"ME"
Robot hand used	=	"HANDNEW"

The following parameters can be inferred by using the parameter generating KB:

Massaging action (ACT$)	=	"KNEAD"	
Massaging paths (N)	=	6	
Radial massaging points (m)	=	3	
Robot arm speed	=	5	
Force retention time	=	1.0	(Sec)
Massaging force	=	3.5	(N)
Diameter of the arm	=	100	(mm)
Length of the arm	=	120	(mm)

The position of the arm has been specified as:

X_0	Y_0	Z_0	α	β
-500	0	150	90^0	180^0

b. Path planning

Using the generated parameters, the orientation of the robotic hand in Cartesian space has been planned as:

$$[\, n \quad o \quad a \,] = \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

And the positions of the robot arm, together with the control angles, have been planned as shown in Table 6.25.

c. Parameter mapping

Before the task execution, the intelligent control organizing module will map all the planned parameters into the control buffer.

In this experiment, the kneading actions performed by the robotic fingers along the radial path have been reduced into 1. Thus, at every radial path, only one kneading will take place. Along the axial direction of the arm, 6 kneading cycles will be carried out.

Table 6.25 Planned positions

N	X	Y	Z	θ_p	θ_r	θ_{F1} (θ_{F2})
0	-350.1	0	112.9	0	0	70.5
1	-350.1	0	136.9	0	0	70.5
2	-350.1	0	160.9	0	0	70.5
3	-350.1	0	184.9	0	0	70.5
4	-350.1	0	208.9	0	0	70.5
5	-350.1	0	232.9	0	0	70.5

d. Task execution

Using the control variables in the control buffer, the kneading operations can be carried out. The procedure of the kneading operation in this experiment is:

* move the robot hand to the kneading start position
* input commands to start the kneading operation
* robot hand performs the kneading operation
* analyse the sensory feedback information
* perform error-correction if error occurs
* otherwise, continue the kneading operation

Fig. 6.17 shows the kneading force-time history of the fingertip 1 (HAND-II) for the first 4 kneading cycles in one kneading process. Where, the robot hand moves from position N=5 to position N=0 (See Table 6.25). The kneading force is 3.5 N. And the force retention time is 1.0 Sec. The force-time curve was recorded by the X-Y plotter.

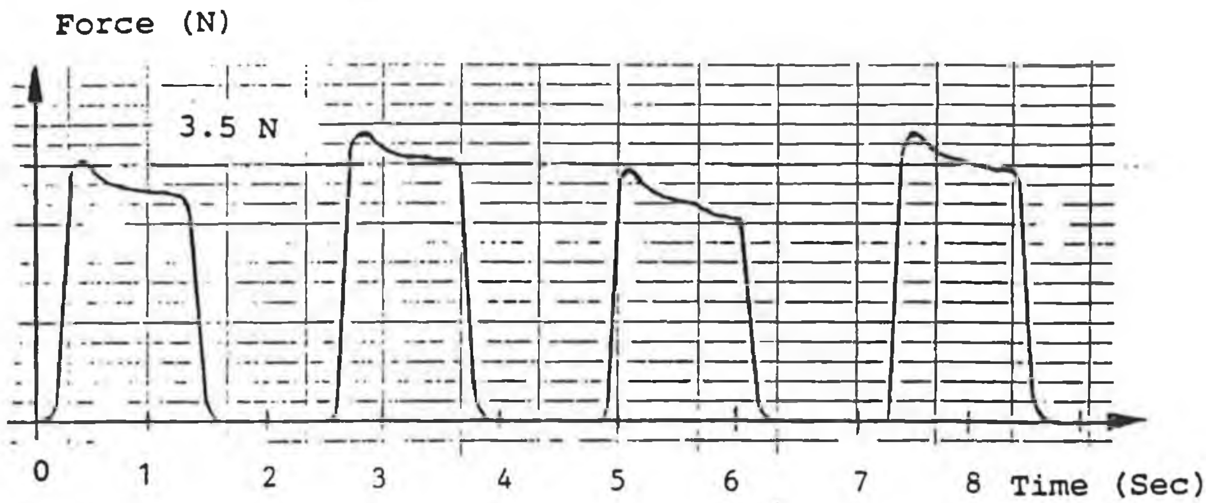


Fig. 6.17 Kneading force-time history

e. On-line error-corrections

Experiment on on-line error-corrections has been carried out. During the kneading opeartion, the cylinder foam has been moved deliberately in a range of ±30 mm along Y axis. Once the position change of the arm is detected, a on-line error-correction will be carried out. Using the sensed positions of the robotic fingers, the correction strategies are inferred from the fuzzy inference mechanism. Table 6.26 shows a fuzzy inference process.

Where the error has occured in position N=2 when the arm was moved about -25 mm along Y axis. The robot hand moves from position N=0 to position N=5. The force-time history of the fingertip 1 is shown in Fig. 6.18.

Table 6.26 Fuzzy inference process

Position	Fuzzy inputs		Inferred corrections		
N	$E\theta_1$	$E\theta_2$	EX	EY	EZ
0	-1.7	-4.0	0	0	0
1	-5.7	-1.2	0	0	0
2	+18.6	-22.5	0	-23.1	0
3	-0.2	-6.1	0	0	0
4	+2.7	-3.9	0	0	0
5	-2.5	-3.8	0	0	0

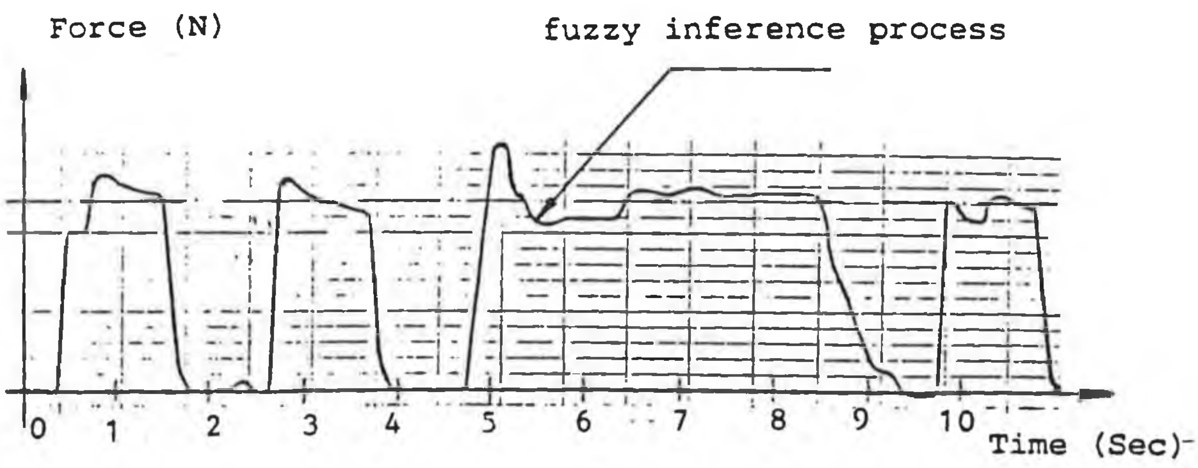


Fig. 6.18 Force-time history in error-correction process

When the robot hand moves back from position N=5 to position N=0, a position change was made for the arm at N=4. Where the arm was moved about -24 mm along Y axis. An error was detected by the intelligent control module. And a on-line error-correction was carried out. Table 6.27 shows the fuzzy inference process in this case. And the force-time history of the fingertip 1 (HAND-II) is shown in Fig. 6.19.

From the experiments, it can be found that the fuzzy inference result is very close to the actual error. Some tests using this system to massage the human arms were also carried out. It has been found that the functions of the system are satisfactory. Hence, it can be concluded that the designed intelligent control module is feasible and effective.

Table 6.27 Fuzzy inference process

Position	Fuzzy inputs		Inferred corrections		
N	$E\Theta_1$	$E\Theta_2$	EX	EY	EZ
5	-4.6	-0.7	0	0	0
4	+16.5	-20.8	0	-23.1	0
3	-3.8	-1.6	0	0	0
2	-2.7	-3.5	0	0	0
1	-1.6	-4.2	0	0	0
0	-2.1	-3.6	0	0	0

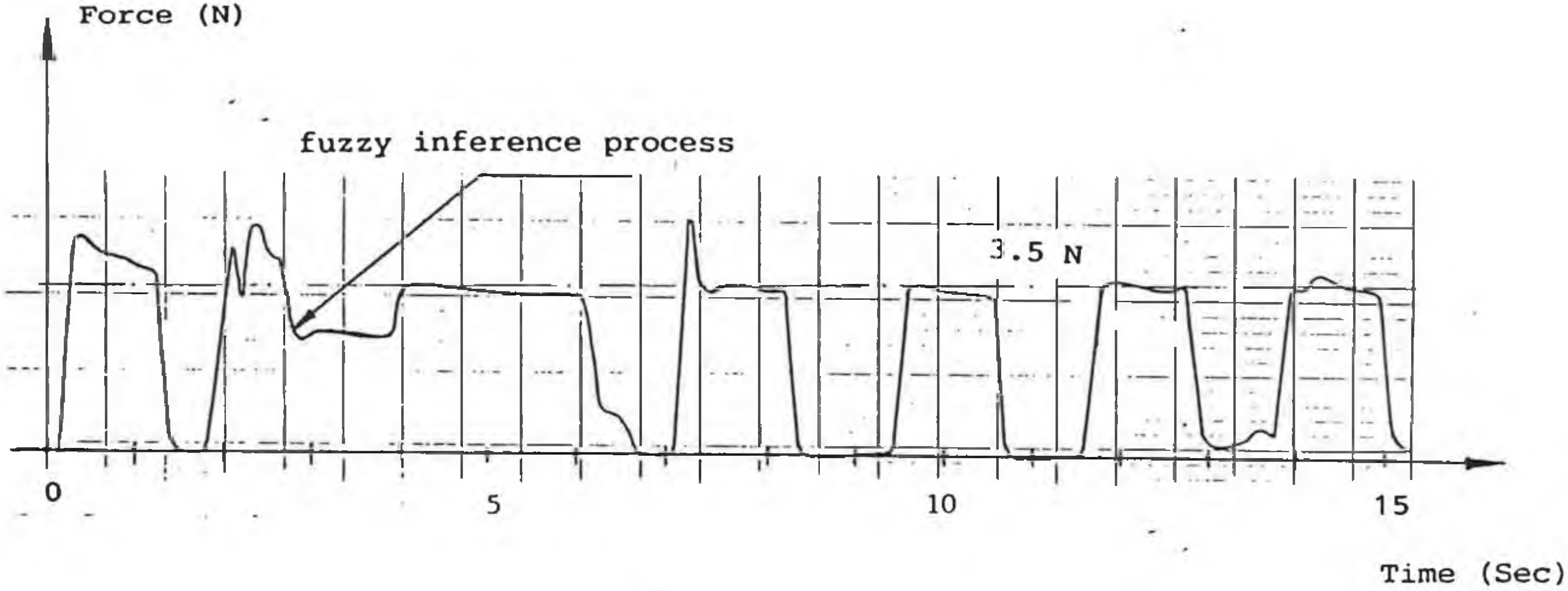


Fig. 6.19 Force-time history in error-correction process

Chapter Seven

Conclusions and Recommendations for further work

7-1 Conclusions

In this research, the physiotherapic robot system has been constructed with the KBS and the fuzzy inference mechanism to cope with any uncertainties and errors in the real-time control process. And a dexterous robotic hand with an integrated palm and two fingers has been employed to perform the padding and kneading opeartions.

To carry out the massaging process effectively, an intelligent robot control system for physiotherapic applications has been developed. The intelligent robot control system consists of a specially designed robotic hand with built-in sensors, an interfacing module between the robot system and the computer, an intelligent path planning module and a fuzzy logic based intelligent control module.

Two hands, HAND-I and HAND-II, have been developed with different types of force sensing units -- FSR and load cell. For the FSR sensor on each fingertip of HAND-I, a constant contact area was maintained by using a layer of silicone rubber between the contact piece and the FSR sensing surface. And a good response of the FSR force sensor has been achieved. While for the load cell on each fingertip of HAND-II, a constant contact area was maintained by using a cylinder bar with a contact cap on top of it. Comparing with FSR sensors, load cell sensing units are more sensitive, robust and compact.

The force level of the robotic fingers can be controlled directly by sending the control voltages to the designed

DC motor drive circuit. The maximum working torque provided by each finger is 0.9 Nm.

The force level of the robotic palm can only be achieved by controlling the fine motion of the robot palm after contact is made. The maximum working force provided by the palm is 5 N.

The designed robot hands (HAND-I & HAND-II) with the integrated palm and two fingers can be used to perform the padding and kneading operations.

However, the weight of the robotic hands was limited by the load carrying capability of the robot arm. From the design of HAND-I and HAND-II, it has been concluded that the weight of the robotic hands cannot be greatly reduced due to the necessity of integrating the two DC motors into the body of the hand.

The massaging speed is proportional to the robot arm speed and the force retention time. The higher the massaging speed, the higher the robot arm speed. While, the higher the massaging speed, the shorter the force retention time. For the safe operation, the robot arm speed should be under 7 (speed level).

The required force retention time can be realized by using the timer in the A/D conversion board (DAS8). As it has been found that the timer in the robot arm controller was not suitable for generating the required time delay.

Constructed with the expertise knowledge bases (KBS) of the massaging process and the fuzzy logic based inference mechanism, the intelligent path planning module can deal with uncertainties by manipulating the fuzzy/linguistic terms. Thus, With the fuzzy/linguistic input terms, the required parameters can be generated and the massaging path can be planned off-line. From the path planning examples, it can be concluded that the designed intelligent path planning module is effective and feasible. However, due to the limitation of the configuration of the robot arm, not all positions and orientations of the part being massaged can be attained by the robotic hand. To make the robot system more dexterous, a 6 DOF robot arm should be used.

In robotic massaging process, the wrong specified part location, part deviations from its specified position and incorrectly planned path have been identified as the causes of uncertainties and errors.

Hence, it can be concluded that to carry out the massaging process effectively, the intelligent control module has to be constructed with the KBS and the fuzzy logic inference mechanism to cope with any uncertainties or errors. Furthermore, by using the error threshold in the intelligent control software, a more effective massaging operation has been achieved. When the detected errors are within the threshold, the errors will be ignored and no error-correction is taken place. Otherwise, fuzzy logic inference will be initiated and error-correction must be carried out.

Experimental results have shown that the fuzzy inferred correction distances are very close to the actual errors. Thus, it can be concluded that the designed fuzzy inference mechanism is feasible and effective.

Furthermore, using the designed intelligent control module, the complicated methematical model and dynamical analysis of the control system can be avoided. And the expertise knowledge can be incorporated into the control process. Also the AI control can be realized in a real-time control process. Thus, the developed fuzzy inference mechanism and the AI control system can be applied to other similar application areas.

7-2 Recommendations for further work

-- To speed up the part locating process and to ensure the safe operation of the robotic system, robotic vision sensors should be incorporated into the robotic massaging system. Using robotic visions, the part size and location in the robot workspace can be roughly observed. These observations may be in a form of fuzziness and can be employed by the parameter generating module to organize the parameters for the path planning.

-- To give the host PC more time to handle higher level control organizing, the slave microcomputers should be used to perform the position/force servo loop control of the robotic hand. Speed/acceleration sensors should be mounted onto the DC motor shafts to realize the speed and acceleration control over the robotic fingers.

-- To apply the developed robot control system into other applications such as delicate material handling and industrial assembly, a three fingered hand with multi-joints should be developed. Different types of force sensors should be used. A 6 axial robot wrist force sensing unit might be required to carry out the complicated industrial assembley operations.

-- To speed up the fuzzy inference, fuzzy logic computer should be used in the next research stage. Furthermore, Using the fuzzy logic computer, the expert knowledge can be easily incorporated into the control system and the on-line control system will be more robust.

REFERENCES

[1] Shahinpoor, M., " A robot engineering textbook ", Harper & Row Publishers, 1987

[2] Okada, T., " On a versatile finger system ", Proc. 7th Int. Symp. on Industrial Robots, Tokyo, 1977, PP345-352

[3] Okada, T., " Object handling system for manual industry ", IEEE Trans. Syst., Man, Cybern., SMC-9, 1979, PP79-89

[4] Okada, T., " Computer control of multijointed finger system for precise object-handling", IEEE Trans. Syst., Man, Cybern., SMC-12, 1982, PP289-299

[5] Hanafusa, H. and Asada, H., " Stable prehesion by a robot hand with elastic fingers ", Proc. 7th Int. Symp. on Industrial Robots, Tokyo, 1977, PP361-368

[6] Hanafusa, H. and Asada, H.," A robot hand with elastic fingers and its application to assembly process ", Proc. IFAC Symp. on Inform. and Control Problems in Manufact. Tech., Tokyo, 1977, PP127-138

[7] Salisbury, J.K. and Craig, J.J., "Articulated hands: force control and kinematic issues ", Int. J. of Robotics Research, Vol.1, No.1, 1982

[8] Salisbury, J.K. and Roth, B., " Kinematic and force analysis of articulated hands ", J. Mechanisms, Transmissions, and Automation in Design, 1983, PP35-41

[9] Salisbury, J.K., " Interpretation of contact geometries from force measurements ", Robotics Research

(eds.) by Brady, M. and Paul, R., MIT Press, 1984, PP565-577

[10] Salisbury, J.K., et al, " Integrated lamguage, sensing, and control for a robot hand ", Proc. 3rd Int. Symp. of Robotics Research, France, 1985, PP54-61

[11] Abramowitz, J., et al, " Pennsylvania articulated mechanical hand ", Proc. ASME Conf. on Robotics, Chicago, USA, 1983

[12] Caporali, M. and Shahinpoor, M., " Design and construction of a five-fingered robotic hand", Robotics Age, Vol.6, No.2, 1984, PP14-20

[13] Jacobsen, S.C., et al, " The version I Utah/MIT dexterous hand ", Robotic Research (eds.) by Hanafusa & Inoue, MIT Press, 1985, PP301-308

[14] Siegel, D.M., et al, "Computational architecture for the Utah/MIT hand ", Proc. IEEE Int. Conf. Robotics and Automation, St. Louis, USA, 1985, PP1016-1021

[15] Jacobsen, S.C., et al, " Design of the Utah/MIT dexterous hand ", Proc. IEEE Int. Conf. Robotics and Automation, San Francisco, USA, 1986, PP1520-1532

[16] Narasimhan, S., et al, " Implementation of control methodologies on the computational architecture for the Utah/MIT hand ", Proc. IEEE Int. Conf. Robotics & Automation, San Francisco, USA, 1986, PP1884-1889

[17] Narasimhan, S., et al, " CONDOR: An architacture for controlling the Utah-MIT dexterous hand", IEEE Trans Robotics and Automation, Vol.5, No.5, Oct. 1989, PP616-627

[18] Biggers, K.B., et al, " Low-level control of the Utah-MIT dexterous hand ", Proc. IEEE Int. Conf. on Robotics and Automation, CA, USA, 1986, PP61-66

[19] Edson, D.V., " Giving robot hands a human touch ", High Technology, 1985, PP31-35

[20] Hitachi, Ltd., "Hitachi's SMA robot hand ", 1985

[21] Yamafuji, K. and Maeda, T., "Development of a multi-processor controlled robot hand ", Adv. Manuf. Eng., Vol.1, Oct. 1988, PP21-25

[22] Allen, P.K., et al, " A system for programming and controlling a multisensor robotic hand ", IEEE Trans on Syst., Man, Cybern., Vol.20, No.6, 1990, PP1450-1456

[23] Koivo, A.J. and Houshangi,N., " Real-time vision feedback for servoing robotic manipulator with self-turning controller", IEEE Trans. Syst., Man, Cybern. Vol.21, No.1, 1991, PP134-142

[24] Dreyfus, M.G., " Visual Robots ", Industrial Robot, Dec., 1974

[25] Fairhurst, M.C., " Computer vision for robotic systems ", Englewood Cliffs, 1988

[26] Trivedi, M.M., et al, " A vision system for robotic inspection and manipulation ", IEEE Comput., Vol.22, 1989, PP91-98

[27] Beni, G. and Hackwood, S., " Recent advances in robotics ", John wiley & sons, Inc., 1985

[28] Webster, J.G., " Tactile sensors for robotics and medicine ", John wiley & sons, Inc., 1988

[29] Luo, R.C., et al, " Dynamic multi-sensor data fusion system for intelligent robots", IEEE Trans. Robotics & Automation, Vol.4, No.4, 1988, PP386-396

[30] Lee, M.H., et al, " A control and monitoring system for multiple-sensor industrial robots " , Proc. 3rd Int. Conf. on Robot Vision & Sensory Controls, USA, 1983

[31] Hillis, D., " A high resolution imaging tactile sensor ", Int. J. Robotics Research, Vol.1, No.2 1982, PP33-44

[32] Shimano, B. and Roth, B., " On force sensing information and its use in controlling manipulators ", Proc. 8th Int. Symp. on Industrial Robots, 1978, PP119-126

[33] Watson, P.C. and Drake, S.H., " Pedestal and wrist force sensors for automatic assembly ", Proc. 5th Int. Symp. on Industrial Robots, 1975, PP501-511

[34] Coiffet, P., " Interaction with the environment ", (Robot Technology, Vol.2), Kogan page Ltd., 1983

[35] Hill, J.W. and Sword, A.J. " Manipulation based on sensor directed control: an integrated end-effector and touch sensing system ", Proc. 17th Annual Human Factor Society Convention, USA, 1973

[36] Feddema, J.T. and Mitchell, O.R., " Vision-guided servoing with feature-based trajectory generation", IEEE Trans. Robotics & Automation, Vol.5, 1989, PP691-700

[37] Elmaraghy, H.A. and Payandeh, S., " Contact prediction and reasoning for compliant robot motions ", IEEE Trans. Robotics & Automation, Vol.5, No.4, 1989, PP533-538

[38] Kwoh, Y.S., et al, " A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery ", IEEE Trans. Biomedical Eng., Vol.35 No.2, 1988, PP153-160

[39] Andre, G., " A multiproximity sensor system for the guidance of robot end effectors ", Proc. Int. Conf. Robot Vision and Sensory Controls, The Netherlands, 1985

[40] Hirzinger, G., et al, " Multisensory robots and sensor based path generation ", Proc. IEEE Int. Conf. Robotics and Automation, CA, USA, 1986

[41] Critchlow, A.J., " Introduction to Robotics ", New York: Macmillan, 1985

[42] Russel, R.A., " Closing the sensor-computer-robot control loop ", Robotics Age, Vol.6, No.4, 1984, PP15-20

[43] Raibert, M.H., " An all digital VLSI tactile array sensor", Proc. IEEE Int. Conf. Robotics, 1984, PP314-319

[44] Nakano, E., et al, " Cooperational control of the anthropomorphous manipulator", Proc. 4th Int. Symp. on Industrial Robots, 1974, PP251-260

[45] Binford, T.D., " sensor system for manipulation ", Proc. 1st Conf. On Remotely Manned Systems, 1973, PP283-291

[46] Rosen, C.A. and Nitzan, D., " Development in programmable automation ", Manufact. Eng., 1975, PP26-30

[47] Nitzan, D., et al, " The measurement and use of registered reflectance and range data in scene analysis ", Proc. IEEE, Vol.65, 1977, PP206-220

[48] Nevins, J.L. and Whitney, D.E., " Computer controlled assembly ", Science of American, Vol.238, No.2, 1978, PP62-73

[49] Van Brushell, H. and Simons, J., " Automatic assembly by active force feeadback accommodation ", Proc. 8th Int. Symp. Indutrial Robots, 1978, PP181-193

[50] Warnecke, H.J., et al, " Programmable assembly with tactile sensors and visual inspection ", Proc. 1st Int. Conf. Assembly Automation, UK, 1980, PP23-32

[51] Bejczy, A.K., " Smart sensors for smart hands ", AIAA/NASA Conf. on Smart Sensors, USA, 1978, PP17

[52] Bejczy, A.K., " Effect of hand-based sensors on manipulator control performance ", Mechanism & Machine Theory, Vol.12, 1977, PP547-567

[53] Hirzinger, G. and Plank, G., "Controlling a Robot's motion speed by a force-torque-sensor for deburring problems ", Proc. 4th IFAC-IFIP Symp. Infrom. Contr. Problems in Manufact. Tech., 1982

[54] Loucks, C.S., et al, " Modeling and control of the Stanford/JPL hand ", Proc. IEEE Int. Conf. Robotics Automation, 1986, PP1520-1532

[55] Whitney, D.E., " Elements of intelligent robot grinding systems", Proc. 3rd Int. Symp. Robotics Research, France, 1985

[56] Heer, E. and Bejczy, A.K., " Control of robot manipulators for handling and assembling in space ", Mech. Mach. Theory, Vol.18, No.1, 1983, PP23-35

[57] Dario, P., et al, " An advanced robot system for automated diagnostic tasks through palpation", IEEE Trans. Biomedical Eng., Vol.35, No.2, 1988, PP118-126

[58] Ogorek, M., " Tactile sensors ", Manufact. Eng., Vol.94, No.2, 1985, PP69-77

[59] Pennywitt, K.E., " Robotic tactile sensing ", BYTE, 1986, PP177-200

[60] Dario, P. and Rossi, D.D., "Tactile sensors and the gripping chanllenge ", IEEE Spectrum, Vol.22, No.8, PP46-52

[61] Harmon, L.D., " Automated tactile sensing ", Int. J. Robotics Research, Vol.1, No.2, 1982, PP3-32

[62] Goldwasser, S.M., " Computer architecture for grasping ", Proc. IEEE Int. Conf. on Robotics, 1984, PP320-325

[63] Mehdian, M. and Rahnejat, H., " A sensory gripper using tactile sensors for object recognition, orientation control, and stable manipulation ", IEEE Trans syst., Man, Cybern., Vol.19, No.5, 1989, PP1250-1261

[64] Harmon, S.Y., et al, " Sensor data fusion through a distributed blackboard ", Proc. IEEE Int. Conf. on Robotics & Automation, USA, 1986, PP1449-1454

[65] Moravec, H.P. and Elfes, A.E., " High resolution maps from wide angle sonar ", Proc. IEEE Int. Conf. on Robotics & Automation, USA, 1985, PP116-121

[66] Lee, I. and Goldwasser, S.M., " A distributed test for active sensory processing", Proc. IEEE Int. Conf on Robotics and Automation, USA, 1985, PP925-930

[67] Watson, P.C., " A multidimensional system analysis of the assembly process as performed by a manipulator ", 1st North American Robot Conf., USA, 1976

[68] Whitney, D.E. and Nevins, J.L., " What is the romote centre complinace and what can it do? ", Proc. 9th Int. Symp. on Industrial Robots, USA, 1979, PP135-152

[69] Nevins, J.L. and Whitney, D.E., "Assembly research", Automatica, Vol. 16, 1980, PP595-613

[70] Whitney, D.E., " Quasi-static assembly of compliantly supported rigid parts ", J. Dyn. Syst., Meas., Contr., Vol.104, 1982, PP65-77

[71] El Baradie, M.A., et al, "The design and development of a romote center compliance, for automatic assembly ", Proc. IMC-6, Ireland, 1989, PP1268-1283

[72] Whitney, D.E., "Force feedback control of manipulator fine motions ", J. Dyn. Syst., Meas., Contr., June 1977, PP91-97

[73] Maples, J.A. and Becker, J.H., " Experiments in force control of robotic manipulators ", Proc. IEEE Robotics and Automation, 1986, PP695-702

[74] Salisbury, J.K., " Active stiffness control of a manipulator in cartesian coordinates ", IEEE Conf. on Decision & Control, USA, 1980

[75] Paul, R.P. and Shimano, B., "Compliance and control" Proc. Joint Automatic Control Conf., CA, USA, 1976, PP694-699

[76] Mason, M.T., " Compliance and force control for computer controlled manipulators ", IEEE Trans. Syst., Man, and Cybern., Vol SMC-11, No.6, 1981, PP418-432

[77] Raibert, M.H. and Craig, J.J, " Hybrid position/ force control of manipulators ", J. Dyn. Syst., Meas., Contr., Vol 102, 1981, PP126-133

[78] Zhang, H. and Paul, R.P., " Hybrid control of robot manipulators ", Proc. IEEE Robotics and Automation, 1985

[79] Yoshikawa, T., et al, "Dynamic hybrid position/force control of robot manipulators -- controller design and experiment ", IEEE Trans. on Robotics and Automation, Vol.4, No.6, 1988, PP699-705

[80] Yoshikawa, T., et al, "Dynamic hybrid position/force control of robot manipulators -- description of hand constraints and calculation of joint driving force", Proc. IEEE Robotics & Automation, 1986, PP1393-1398

[81] An, C.H. and Hollerbach, J.M., " The role of dynamic models in cartesian force control of manipulators ", Int. J. Robot Research, Vol.8, No.4, 1989, PP51-72

[82] Mills, J.K. and Goldenberg, A.A., " Force and position control of manipulators during constrained motion tasks ", IEEE Trans. on Robotics & Automation, Vol.5, No.1, 1989, PP30-46

[83] Aronne, E.J. and Yang, J.C.S., " A force control system for robotic manipulators ", 8th Int. Conf. on Offshore Mechanica and Arctic Eng., 1989, PP143-155

[84] Asakawa, K., et al, " A variable complinace device and its application for automatic assembly ", Proc. 5th Conf. Auto. Fact., USA, 1983, PP10.1-10.17

[85] Kazerooni, H., "Direct-drive active compliant end-effector (active RCC) ", IEEE Trans. on Robotics & Automation, Vol.4, No.3, 1988, PP324-333

[86] Hogan, N., " Impedance control: An approach to manipulation ", J. Dyn. Syst., Meas., Contr., Vol.107, 1985, PP1-24

[87] Anderson, R.J. and Spong, M.W., " Hybrid Impedance control of robotic manipulators ", IEEE Trans. on Robotics & Automation, Vol.4, No.5, 1988, PP549-556

[88] Loozano-Perez, T., " Compliance in robot manipulation", Artificial Intelligence, Vol.25, 1985, PP5-12

[89] Gevarter, W.B., " Intelligent machines: An introductory perspective of artificial intelligence and robotics ", Prentic-Hall, Inc., 1985

[90] Bick, J.R. and Kelley, R.B., " An overview of the basic research needed to advance the state of knowledge in robotics ", IEEE Trans. on Syst., Man, & Cybern., Vol. SMC-11, No.8, 1981, PP575-579

[91] Sacerdoti, E.D., " Planning in a hierarchy of abstraction spaces ", Artificial Intelligence, Vol.5, 1974, PP115-135

[92] Sacertodi, E.D., " The nonlinear nature of plan ", Proc. 4th Int. Joint Conf. AI, USA, 1975, PP206-214

[93] Tate, A., " Generating project networks ", Proc. 5th Int. Joint Conf. AI, USA, 1977, PP888-893

[94] Chang, K.H. and Wee, W.G., " A Knowledge-based planning system for mechanical assembly using robots ", IEEE Expert, Spring, 1988, PP18-30

[95] Rokey, M. and Grenander, S., " Planning for space telerobotics: The remote mission specialist ", IEEE Expert, June, 1990, PP8-15

[96] Locke, C., " Pushing the envelope ", IEEE Expert, June, 1990, PP2-7

[97] Chochon, H. and Alami, R., " NNS, A knowledge-based on-line system for an assembly workcell ", IEEE Int. Conf. on Robotics & Automation, USA, 1986

[98] Kerth, Jr. and William, J., " Knowledge-based expert welding ", Proc. SME 9th Conf. on Robots, USA, 1985, PP5-110

[99] Khogabandehloo, K., "Getting down to the bare bones" The Industrial Robot, Vol.16, No.3, 1989, PP160-165

[100] Trevelyan, J.P., " Skills for a shearing robot: dexterity and sensing ", Proc. 2nd Int. Synp. on Robotics Research, Japan, 1984

[101] Trevelyan, J.P., et al, "Adaptive motion sequencing for process robots ", Proc. 4th Int. Symp. on Robotics Research, USA, 1986

[102] Trevelyan, J.P., " Sensing and control for sheep-shearing robots ", IEEE Trans. on Robotics & Automation, Vol.5, No.6, 1989, PP716-727

[103] Walker, T.C. and Miller, B.K., " Expert systems handbook ", The Fairmont Press, Inc., 1990

[104] Zadeh, L.A., "Fuzzy sets ", Information & Control, Vol.8, 1965, PP338-353

[105] ____, "Fuzzy algorithm ", Information & Control, Vol.12, 1968, PP94-102

[106] ____, " A rational for fuzzy control ", J. Dyn. Syst., Meas., Contr., Vol.94, 1072, PP3-4

[107] ____, " Outline of a new approach to the analysis complex systems and decision processes ", IEEE Trans. Syst., Man, & Cybern., Vol. SMC-3, 1973, PP28-44

[108] ____, " The role of fuzzy logic in the management of uncertainty in expert system ", Fuzzy set and Systems, Vol.11, 1983, PP199-227

[109] Mamdani, E.H., " Applications of fuzzy algorithm for simple dynamic plant ", Proc. IEEE, Vol.121, No12, 1974, PP1585-1588

[110] Mamdani, E.H. and Assilian, S., " An experiment in linguistic synthesis with a fuzzy logic controller" Int. J. Man & Mach. Studies, Vol.7, No.1, 1975, PP1-13

[111] Mandani, E.H., " Advances in linguistic synthesis of fuzzy controllers", Int. J. Man & Mach. Studies, Vol.8, No.6, 1976, PP669-678

[112] ____, " Application of fuzzy logic to approximate reasoning using linguistic synthesis ", IEEE Trans. Computer, Vol. C-26, No.12, 1977, PP1182-2291

[113] King, P.J. and Mamdani, E.H., " The application of fuzzy control systems to industrial processes ", Automatica, Vol.13, No.3, 1977, PP235-242

[114] Mamdani, E.H. and Gaines, B.R., " Fuzzy reasoning and its applications ", London:Academic, 1981

[115] Kickert, W.J.M. and Mamdani, E.H., " Analysis of fuzzy logic controller ", Fuzzy Sets and Systems, Vol.1, 1978, PP29-44

[116] Williams, T., " Fuzzy logic simplifies complex control problems ", Computer Design, March 1991, PP90-102

[117] Lee, C.C., " Fuzzy logic in control systems: Fuzzy logic controller - Part I ", IEEE Trans. Syst., Man, & Cybern., Vol.20, No.2, 1990, PP404-418

[118] Lee, C.C., " Fuzzy Logic in control Systems: Fuzzy logic controller - part II ", IEEE Trans. Syst., Man, & Cybern., Vol.20, No.2, 1990, PP419-435

[119] Mamdani, E.H., et al, " Use of fuzzy logic for implementation rule-based control of industrial processes ", Fuzzy Sets & Decision Analysis, NY: North Holland, 1984, PP429-445

[120] Tokagi, T. and Sugeno, M., " Fuzzy identification of systems and its applications to modelling and control ", IEEE Trans. Syst., Man, & Cybern., Vol. SMC-15, No.1, 1985, PP116-132

[121] ____, " Derivation of fuzzy control rules from human operator's control actions", Proc. IFAC Symp. Fuzzy Information, Knowledge Representation & Decision Analysis, France, 1983

[123] Tanscheit, R. and Scharf, E.M., " Experiments with the use of a rule-based self-organizing controller for Robotics Application ", Fuzzy Sets and Systems, Vol.26, 1988, PP195-214

[124] Uragami, M., et al, " Fuzzy robot controls ", J. Cybernetics, Vol.6, 1976, PP39-64

[125] Goguen, J., " On fuzzy robot planning ", in Fuzzy Sets & Application to Cognitive Decision Processes, (eds.) by Zadeh, L., et al, 1977, PP429-447

[126] Hirota, K., et al, " Robot Control based on membership and vagueness ", in Approximate Reasoning in Expert Systems, (eds.) by Gupta, M.M., et al, 1985, PP621-635

[127] Kouatli, I. and Jones, B., " An improved design procedure for fuzzy control systems", Int. J. Mach. Tools & Manufact., Vol.31, No.1, 1991, PP107-122

[128] Saradis, G.N., " Intelligent Robotic Control ", IEEE Trans. on AC, Vol. AC-28, No.5, 1983, PP547-557

[129] ____, " Foundations of the theory of intelligent controls ", IEEE Workshop on Intelligent Control,

USA, 1986, PP23-28

[130] Saridis, G.N. and Graham, J.H., " Linguistic decision scemata for intelligent robots ", Automatica, Vol.20, NO.1, 1984, PP121-126

[131] Ray, K.S., et al, " Structure of an intelligent fuzzy logic controller and its behaviour ", in Approximate Reasoning in Expert Systems, (eds.) by Gupta, M.M., et al, 1985, PP593-619

[132] Chen, Y.Y. and Tsao, T.C., " A description of the dynamical behavior of fuzzy systems ", IEEE Trans. on Syst., Man, & Cybern., Vol.19, No.4, 1989, PP745-755

[133] Togai, M. and Watanabe, H., " Expert system on a chip : An engine for real-time approximate reasoning ", IEEE Expert, Fall 1986, PP55-62

[134] Yamakawa, T. and Miki, K., " The currect mode fuzzy logic integrated circuits fabricated by the standard CMOS process ", IEEE Trans. on Computer, Vol. C-35, No.2, 1986, PP161-167

[135] Eshera, M.A. and Barash, S.C., " Parallel Rule-based fuzzy inference on mesh-connected systolic arrays", IEEE Expert, Winter, 1989, PP27-35.

[136] Lim, M.H. and Takefuji, Y., " Implementing fuzzy rule-based system on silicon chips ", IEEE Expert, Feb. 1990, P31-45

[137] Waller, L., " Fuzzy logic: It's comprehensible, It's practical -- and it's commercial ", Electronics, March 1989

[138] Kuo, B.C., " Automatic control systems ", Prentice-Hall, Inc., 1977

[139] Koren, Y., " Robotics for engineers ", McGraw-Hill, Inc., 1985

[140] Lee, C.S.G., etc., "Hierarchical control structure using special purpose processors for the control of robot arms ", Proc. IEEE Conf. Pat. Recog. Image Process., USA, 1982

[141] Ogata, K., " Discrete-time control systems ", Prentice-Hall, Inc., 1987

[142] Kaze, P., "Digital control using microprocessors", Prentice-Hall, Inc., 1981

[143] Bibbero, R.J., " Microprocessors in instrumentation and control ", Wiley Ltd., 1977

[144] Lee, C.S.G., " Robot arm kinematics, dynamics, and control", Computer, Vol.15, No.12, 1982, PP62-80

[145] Denavit, J. and Hartenberg, R.S., " A Kinematic notation for lower-pair mechanisms based on matric-es", J. Applied Mechanics, June 1955, PP215-221

[146] Lennox, S.C. and Chadwick, M., " Mathematics for engineers and applied scientists", Heinemann, 1979

[147] Self, K., " Designing with fuzzy logic ", IEEE Spectrum, Nov. 1990, PP42-44

APPENDIX A-1 Intelligent commands of robot arm

A Position/Motion Control Instructions

Program yes ······ Possible
no ······ Not possible

	Name	Input Format	Function	Program	Remarks
1	Decrement Position	DP	Moves robot to a predefined position with a position number smaller than the current one.	yes	
2	Draw	DW x, y, z	Moves hand end to a position away from the current one covering the distance specified in X-, Y-, and Z-axis directions.	yes	
3	Here	HE a	Defines the coordinates of the current position by assigning position number (a) to it.	yes	$1 \leq a \leq 629$
4	Home	HO	Establishes the reference position in the cartesian coordinate system.	yes	
5	Increment Position	IP	Moves robot to a predefined position with a position number greater than the current one.	yes	
6	Move Approach	MA a_1, a_2 [, O/C]	Moves hand end from the current position to a position away from position (a_1) in increments as specified for position (a_2).	yes	$1 \leq a_1, a_2 \leq 629$ O: Hand opened; C. Hand closed
7	Move Continuous	MC a_1, a_2	Moves robot continuously through predefined intermediate points between position numbers (a_1) and (a_2).	yes	$1 \leq a_1, a_2 \leq 629$
8	Move Joint	MJ w, s, e, p, r	Turns each joint the specified angle from the current position	no	
9	Move	MO a [, O/C]	Moves hand end to position (a)	yes	$1 \leq a \leq 629$ O: Hand opened; C. Hand closed
10	Move Position	MP x, y, z, p, r	Moves hand end to a position whose coordinates (position and angle) are specified as x, y, z, p and r.	no	
11	Move Straight	MS a, n [, O/C]	Moves robot to position (a) through n intermediate points on a straight line.	yes	$1 \leq a \leq 629$ $1 \leq n \leq 99$ O. Hand opened; C. Hand closed
12	Move Tool	MT a, b [, O/C]	Moves hand end from the current position to a position away from a specified position (a) in incremental distance b in the tool direction.	yes	$1 \leq a \leq 629$ O: Hand opened; C. Hand closed
13	Nest	NT	Returns robot to mechanical origin.	yes	
14	Origin	OG	Moves robot to the reference position in the cartesian coordinate system.	yes	
15	Pallet Assign	PA i, j, k	Defines the number of grid points (j, k) in the column and row directions for pallet (i).	yes	$1 \leq i \leq 9$ $1 \leq j, k \leq 255$
16	Position Clear	PC a_1, [, a_2]	Clears all position data from position a_1 to a_2.	no	$a_1 \leq a_2$ $1 \leq a_1, a_2 \leq 629$ (or $a_1 = 0$)
17	Position Define	PD a, x, y, z, p, r	Defines the coordinates (x, y, z, p, r) of position (a).	no	$1 \leq a \leq 629$

	Name	Input Format	Function	Program	Remarks
18	Position Load	PL a_1, a_2	Assigns the coordinates of position (a_2) to position (a_1).	yes	$1 \leq a_1, a_2 \leq 629$
19	Pallet	PT a	Calculates the coordinates of a grid point on pallet (a) and identifies the coordinates as position (a).	yes	$1 \leq a \leq 9$
20	Position Exchange	PX a_1, a_2	Exchanges the coordinates of position (a_1) for those of position (a_2).	yes	$1 \leq a_1, a_2 \leq 629$
21	Shift	SF a_1, a_2	Shifts the coordinates of position (a_1) in increments representing the coordinates of position (a_2) and redefines the new coordinates.	yes	$1 \leq a_1, a_2 \leq 629$
22	Speed	SP a [, H/L]	Sets the operating velocity and acceleration/deceleration time for robot. 0: Minimum speed; 9: Maximum speed	yes	$0 \leq a \leq 9$ H: High acceleration/deceleration time; L: Low acceleration/deceleration time
23	Timer	TI a	Halts motion for time (a). (Unit: 0.1 second)	yes	$0 \leq a \leq 32767$
24	Tool	TL a	Establishes the distance between hand mounting surface and hand end.	yes	$0 \leq a \leq +300.0$ Unit: mm

[8] Program Control Instructions

	Name	Input Format	Function	Program	Remarks
25	Compare Counter	CP a	Loads value in counter (a) into the internal register	yes	$1 \leq a \leq 99$
26	Disable Act	DA a	Disables interrupt by a signal through bit (a) of external input terminal.	yes	$0 \leq a \leq 7$ (15)
27	Decrement Counter	DC a	Decrements counter (a) by 1.	yes	$1 \leq a \leq 99$
28	Delete Line	DL a_1 [, a_2]	Deletes contents of line numbers from a_1 to a_2.	no	$a_1 \leq a_2$ $1 \leq a_1, a_2 \leq 2048$
29	Enable Act	EA a_1, a_2	Enables interrupt by a signal through bit (a_1) of external input terminal and specifies line number (a_2) to which the program jumps when interrupt occurs.	yes	$(-15) \quad (+15)$ $-7 \leq a_1 \leq +7$ +: ON; −: OFF $1 \leq a_2 \leq 2048$
30	End	ED	Ends the program.	yes	
31	If Equal	EQ a_1 (or &b), a_2	Causes a jump to occur to line number (a_2) if external input data or counter data equals a_1 (or &b).	yes	(−32767) (32767) $0 \leq a_1 \leq 255$ (decimal) $0 \leq b \leq$ &FF (hex.) (&8001) (&7FFF) $1 \leq a_2 \leq 2048$
32	Go Sub	GS a	Permits the instruction sequence to jump to sub-routine which starts with line number (a)	yes	$1 \leq a \leq 2048$
33	Go To	GT a	Permits the program sequence to jump to line number (a) unconditionally.	yes	$1 \leq a \leq 2048$
34	Increment Counter	IC a	Increments counter (a) by 1.	yes	$1 \leq a \leq 99$

	Name	Input Format	Function	Program	Remarks
35	If Larger	LG a_1 (or &b), a_2	Causes a jump to occur to line number (a_2) if external input data or counter data is greater than a_1 (or &b).	yes	(−32767) (32767) $0 \leq a_1 \leq 255$ (decimal) $0 \leq b \leq$ &FF (hex.) (&8001) (&7FFF) $1 \leq a_2 \leq 2048$
36	If Not Equal	NE a_1 (or &b), a_2	Causes a jump to occur to line number (a_2) if external input data or counter data does not equal a_1 (or &b).	yes	(−32767) (32767) $0 \leq a_1 \leq 255$ (decimal) $0 \leq b \leq$ &FF (hex.) (&8001) (&7FFF) $1 \leq a_2 \leq 2048$
37	New	NW	Deletes all program and position data in RAM.	no	
38	Next	NX	Specifies the range of a loop in a program executed by command RC.	yes	
39	Repeat Cycle	RC a	Repeats the loop specified by command NX (a) times.	yes	$1 \leq a \leq 32767$
40	Run	RN a_1 [, a_2]	Executes line numbers from (a_1) to (a_2). (a_2) not included.	no	$1 \leq a_1, a_2 \leq 2048$
52	Return	RT	Completes subroutine activated by command GS and returns to main program.	yes	
42	Set Counter	SC a_1, [a_2]	Loads (a_2) into counter (a_1).	yes	$1 \leq a_1 \leq 99$ $-32767 \leq a_2 \leq 32767$
43	If Smaller	SM a_1 (or &b), a_2	Causes a jump to occur to line number (a_2) if external input data or counter data is smaller than a_1 (or &b).	yes	(−32767) (32767) $0 \leq a_1 \leq 255$ (decimal) $0 \leq b \leq$ &FF (hex.) (&8001) (&7FFF) $1 \leq a_2 \leq 2048$

[C] Hand Control Instructions

	Name	Input Format	Function	Program	Remarks
44	Grip Close	GC	Closes hand grip.	yes	
45	Grip Flag	GF a	Defines the open/close state of hand grip, used in conjunction with command PD.	yes	a = 0 (open), 1 (closed)
46	Grip Open	GO	Opens hand grip.	yes	
47	Grip Pressure	GP a_1, a_2, a_3	Defines gripping force and gripping force retention time.	yes	$0 \leq a_1, a_2 \leq 15$ $0 \leq a_3 \leq 99$ (Unit: 0.1 second)

[D] I/O Control Instructions

	Name	Input Format	Function	Program	Remarks
48	Input Direct	ID	Fetches external signal unconditionally from input port.	yes	
49	Input	IN	Fetches external signal synchronously from input port.	yes	

	Name	Input Format	Function	Program	Remarks
50	Output Bit	OB a	Sets the output state of bit (a) of external output terminal	yes	$-7 \leq a \leq +7$ (-15) $(+15)$ +: ON; −: OFF
51	Output Direct	OD a (or &b)	Outputs data a (or &b) unconditionally through output port.	yes	(−32767) (32767) $0 \leq a \leq 255$ (decimal) 00 ≤ b ≤ &FF (hex.) (&8001) (&7FFF)
52	Output	OT a (or &b)	Outputs data a (or &b) synchronously through output port.	yes	(−32767) (32767) $0 \leq a \leq 255$ (decimal) 00 ≤ b ≤ &FF (hex.) (&2701) (&7FFF)
53	Test Bit	TB a_1, a_2	Causes a jump to occur to line number a_2 by means of bit (a_1) in external input terminal.	yes	$-7 \leq a_1 \leq +7$ (-15) $(+15)$ +: ON; −: OFF $1 \leq a_2 \leq 2048$

E RS232C Read Instructions

	Name	Input Format	Function	Program	Remarks
54	Counter Read	CR a	Reads contents of counter (a)	yes	$1 \leq a \leq 99$
55	Data Read	DR	Reads data in external input terminal, used in conjunction with commands ID and IN	yes	
56	Error Read	ER	Reads status of error (no error:0; error mode I: 1; error mode II: 2).	no	
57	Line Read	LR a	Reads contents of line number (a).	no	$1 \leq a \leq 2048$
58	Position Read	PR a	Reads coordinates of position (a).	yes	$1 \leq a \leq 629$
59	Where	WH	Reads coordinates of current position.	yes	

F Miscellaneous

	Name	Input Format	Function	Program	Remarks
60	Reset	RS	Resets error mode II.	no	
61	Transfer	TR	Transfers contents of EPROM to RAM.	no	
62	Write	WR	Writes contents of RAM into EPROM.	no	
63	Comment		Allows programmer to write a comment following '.	yes	

APPENDIX A-2 Robot arm specifications

Item		Specifications	Remarks
Mechanical Structure		5 degrees of freedom, vertical articulated robot	
Operation range	Waist rotation	300° (max. 120°/sec)	J1 axis
	Shoulder rotation	130° (max. 72°/sec)	J2 axis
	Elbow rotation	110° (max. 109°/sec)	J3 axis
	Wrist pitch	±90° (max. 100°/sec)	J4 axis
	Wrist roll	±180° (max. 163°/sec)	J5 axis
Arm length	Upper arm	250mm	
	Fore arm	160mm	
Weight capacity		Max. 1.2kgf (including the hand weight)	75mm from the mechanical interface (center of gravity)
Maximum path velocity		1000mm/sec (wrist tool surface)	Speed at point P in Fig. 1.3.4
Position repeatability		0.3mm (roll center of the wrist tool surface)	Accuracy at point P in Fig. 1.3.4
Drive system		Electrical servo drive using DC servo motors	
Robot weight Motor capacity		Approx. 19kgf J1 to J3 axes: 30W; J4, J5 axes: 11W	

Item	Specifications
Teaching method	Programming language system (63 commands), MDI (using a personal computer)
Control method	PTP position control system using DC servo motors
Number of control axes	5 axes (+1 optional axis)
Position detection	Pulse encoder system
Return to Origin Origin setting	Limit switches and pulse encoders (Z phase detection method)
Interpolation function	Articulation interpolation, linear interpolation
Speed setting	10 steps (max. 1000mm/sec)
Number of positions	629 (8KB)
Number of program steps	2048 (16KB)
Data storage	Write to EP-ROM using the built-in EP-ROM writer or storage in the battery-backed static RAM (the battery is optional and backs up the RAM for about 2 years).
Position teaching equipment	Teaching box (option) or personal computer
Programming equipment	Personal computer*2
External I/O	General-purpose I/O, 8 points each (16-point type available) General-purpose synchronous signals (STB, BUSY, ACK, RDY) No dedicated I/O (dedicated I/O of 3 points each available) Power for external I/O should be prepared by the user (12V to 24V DC)
Interface	1 parallel interface (conforming to Centronics) 1 serial interface (conforming to RS-232C)
Emergency stop	Using any of the front control switch, teaching box switch, and rear terminal block (N/C contact terminal)
Hand control	Motor-operated hand or pneumatically-operated hand (using AC solenoid)
Brake control	J2 axis (shoulder), J3 axis (elbow)
Power source	120V/220V/230V/240V AC, 0.5KVA
Ambient temperature	5°C to 40°C
Weight	Approx. 23kgf
Size	380 (W) x 331 (D) x 246 (H) mm

APPENDIX B-1 DC Motor specifications

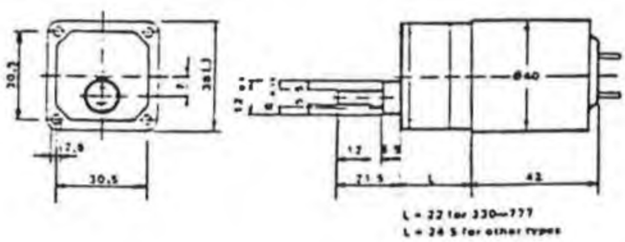
<u>B-1-1 Maxon DC Motor (2140) characteristics</u>

Winding	2140	934
Nominal voltage	V	12
No load speed	rpm	4090
Max. power output	mW	3410
Max. continuous operating current	mA	493
Max. efficiency	%	81
No load current	mA	12
Rotor inertia	gcm^2	23.2
Terminal resistance	Ohm	10
Torque constant	$mNmA^{-1}$	28
Mechanical time constant	ms	32
Max. permissible rotor temperature	^0C	85
Weight	g	187

B-1-2 Maxon gearhead (2938) Specifications

Gear number		2938.804-0100
Reduction		1:100
Number of stages		4
Max. Cont. Torque	(Nm)	0.6
Max. Peak Torque	(Nm)	1.8
Length	(mm)	28.4
Weight	(g)	70

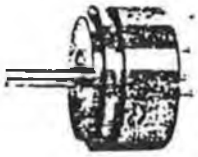
B-1-3 Dimensions of the DC motor with the gearbox



APPENDIX B-2 Potentiometer specifications

potentiometers

conductive plastic servo



Body dia 22 22
Spigot dia 19 05 (3/4 in)
H 13 1 (excl terminals)
Shaft dia 3 17 (1/8 in)
L 12 7

A range of high quality precision servo mount potentiometers, particularly suitable for use with **RS** precision d.c. motor systems as position transducers (refer to the Motors section). The screened conductive plastic element is trimmed to a close tolerance linearity and multifinger wipers provide a low output smoothness with virtually infinite resolution. Two servo bearings afford low shaft torque and long life. As standard with many servo potentiometers the shaft dia. is 1/8 in, set inside a rugged anodised aluminium housing machined to give accurate location of the shaft (with minimal runouts) directly into drive systems. A mounting kit* consisting of three clamps, nuts, screws and mounting instructions is supplied with each potentiometer.

technical specification	
electrical/thermal	
Resistance tolerance	±20%
Linearity (independent)	±0 5%
Output smoothness (max)	0 1%
Power rating	1 W at 40 °C
Derate power to	zero at 125 °C
Wiper current (max)	10 mA
Insulation resistance	10^9 Ω at 500 V d.c.
Dielectric strength	1000 V r m s.
Electrical rotation	340° ±4°
Temperature range	−55 °C to +125 °C
Temperature coeff.	±600 ppm/°C
mechanical	
Rotation	360° continuous
Torque (max.)	
starting	28 Nm. 10^{-4}
running	21 Nm.10^{-4}
Mechanical runouts (max.)	
shaft runout (eccentricity)	0 05
pilot runout (eccentricity)	0 05
lateral runout (parallel difference from the centre line)	0·05
shaft end play	0·13
shaft radial play	0·05
Rotational life**	> 10^7 shaft revolutions

APPENDIX B-3 FSR characteristics

1" dia. circular FSR characteristics

Items	characteristics
Max. applied Volts	5 Volts DC
Max. current	0.25 mA / cm^2
Power dissipation	0.1 W / cm^2
Force range	0 - 10,000 grams
Impedance	>1 MΩ (force: 0) 2.0 kΩ (force: 10Kg)

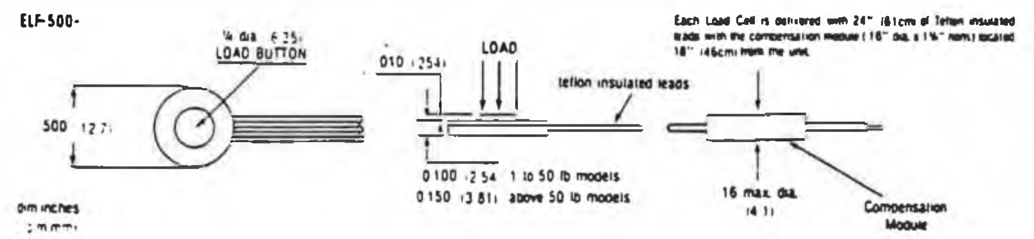
APPENDIX B-4 Load cell specifications

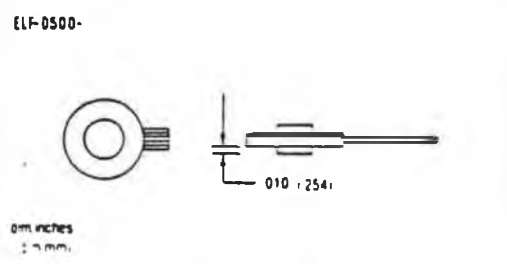
SPECIFICATIONS

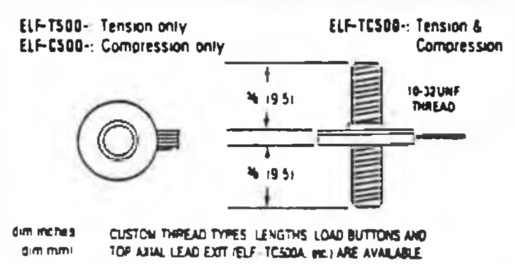
MODEL	ELF-500 -1	ELF-500 -2	ELF-500 -5	ELF-500 -10	ELF-500 -20	ELF-500 -30	ELF-500 -40	ELF-500 -50	ELF-500 -75	ELF-500 -100
[6]RANGE lbs.	1	2	5	10	20	30	40	50	75	100
[2,6]OVERRANGE lbs.	2	4	10	15	30	45	60	75	100	125
[6]SENSITIVITY mV/lb nom.	75	50	40	25	12.5	8	6	5	3	2.5
[1,3]USEFUL FREQ. nom.	0-900Hz	0-1200Hz	0-2800Hz	0-4000Hz	0-6000Hz	0-8000Hz	0-10000Hz	0-12000Hz	0-15000Hz	0-20000Hz

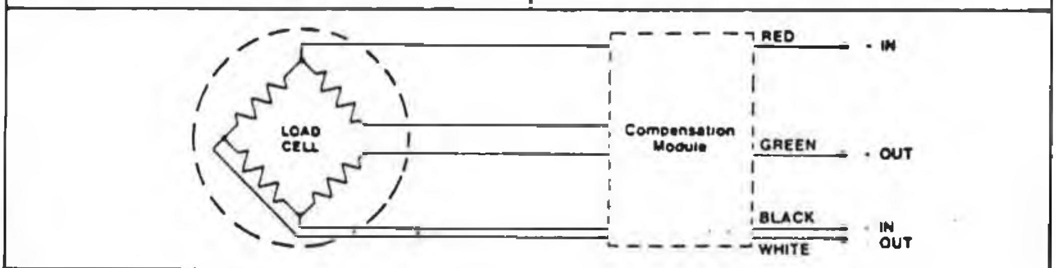
[1]Useful frequency range is 20% of Resonant Frequency [2]Overrange for use within Useful Frequency [3]Valid for ELF-500 basic unit unloaded. Different for other housing styles. [4]Zero offset of ±15mV max at 80°F after warm-up. Lower values available on request. [5]Other Excitations and Temperature Ranges available on request. [6]Must be loaded perpendicular and on center, distributed over loading surface

NON-LINEARITY	±1% F.S.	INPUT IMPEDANCE nom.	2000 Ω nom. typ (1000 Ω min.)
HYSTERESIS	±1% F.S.	OUTPUT IMPEDANCE nom.	1000 Ω nom.
THERMAL ZERO	[4] ±1%F.S./100°F	EXCITATION	[5] 15VDC
THERMAL SENSITIVITY	±2½%/100°F	COMPENSATED TEMP.	[5] 70°F to 170°F (21°C to 77°C) OPTION "Z" 32°F to 140°F (0°C to 60°C)
MODE	Compression except for ELF-T500 & ELF-TC500	OPERATING TEMP.	[5] −40°F to 250°F (−40°C to 121°C)









* "OFF-THE-SHELF" STOCK IN ELF-500-5, -20, -50 and ELF-TC500-5, -10, -20, -100

Specifications subject to change without notice.

APPENDIX B-5 Calibration Equations for FSR sensors

Here $V = V_{FSR}$ (V)

$NC = 9.81/1000$ (N/g)

B-5-1 For FSR on finger #1 of Hand-I

a. V [0.03, 2.30]

$$F = NC*[224 + 224/2.27*(V - 2.30)] \quad (N)$$

b. V [2.30, 3.41]

$$F = NC*[624 + 400/1.11*(V - 3.41)] \quad (N)$$

c. V [3.41, 3.74]

$$F = NC*[824 + 200/0.33*(V - 3.74)] \quad (N)$$

B-5-2 For FSR on finger #2 of Hand-I

a. V [0.01, 2.50]

$$F = NC*[224 + 224/2.49*(V - 2.50)] \quad (N)$$

b. V [2.50, 3.65]

$$F = NC*[524 + 300/1.15*(V - 3.65)] \quad (N)$$

c. V [3.65, 4.16]

$$F = NC*[824 + 300/0.51*(V - 4.16)] \quad (N)$$

<u>B-5-3</u> For FSR on palm of Hand-I

a. V [0.63, 2.45]

F = NC*[200 + 200/1.82*(V - 2.45)] (N)

b. V [2.45, 3.64]

F = NC*[500 + 300/1.19*(V - 3.64)] (N)

c. V [3.64, 4.22]

F = NC*[800 + 300/0.58*(V - 4.22)] (N)

<u>B-5-4</u> For FSR on palm of Hand-II

a. V [0.20, 1.60]

F = NC*[124 + 124/1.40*(V - 1.60)] (N)

b. V [1.60, 2.28]

F = NC*[324 + 200/0.68*(V - 2.28)] (N)

c. V [2.28, 3.56]

F = NC*[924 + 600/1.28*(V - 3.56)] (N)

APPENDIX B-6 Calibration Equations for Load Cells

Here $V = V_{out}$ (V)

$NC = 9.81/1000$ (N/g)

B-6-1 For load cell on finger #1 of Hand-II

a. V [0.18, 0.92]

$$F = NC*[124 + 124/0.74*(V - 0.92)] \quad (N)$$

b. V [0.92, 3.89]

$$F = NC*[924 + 800/2.97*(V - 3.89)] \quad (N)$$

B-6-2 For load cell on finger #2 of Hand-II

a. V [0.20, 1.08]

$$F = NC*[124 + 124/0.88*(V - 1.08)] \quad (N)$$

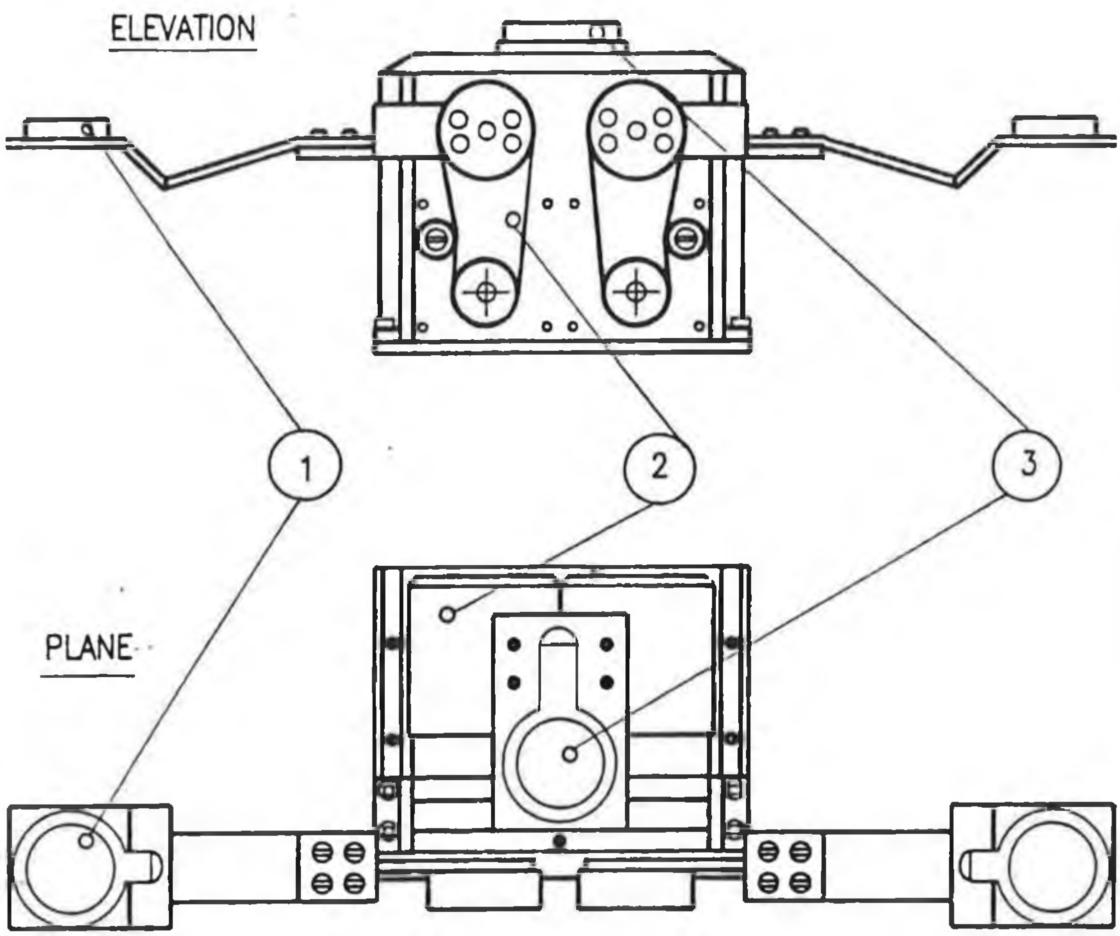
b. V [1.08, 4.03]

$$F = NC*[924 + 800/2.95*(V - 4.03)] \quad (N)$$

APPNEDIX C-1 Mechanical drawings for HAND-I

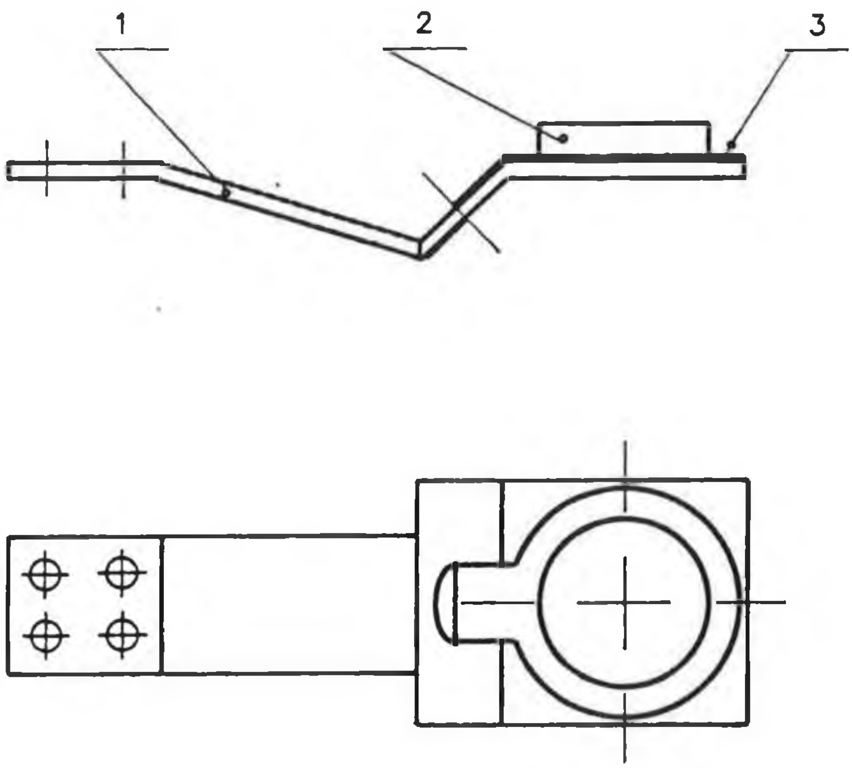
INDEX TO DESIGN DRAWINGS for HAND-I

Drawing No.	Title
H1-00	ROBOTIC HAND-I
H1-01	ROBOTIC FINGER
H1-01-01	FINGER BODY
H1-01-02	CONTACT PLATE
H1-02	ROBOTIC HAND BODY
H1-02-01	PALM BASE
H1-02-02	FINGER BASE
H1-02-03	ENFORCEMENT WALL A
H1-02-04	ENFORCEMENT WALL B
H1-02-05	HAND BASE
H1-02-06	SUPPORT WALL A
H1-02-07	SUPPORT WALL B
H1-02-08	FINGER SHAFT
H1-02-09	BUSH
H1-03	ROBOTIC PALM
H1-03-01	PALM BODY
H1-03-02	CONTACT PLATE



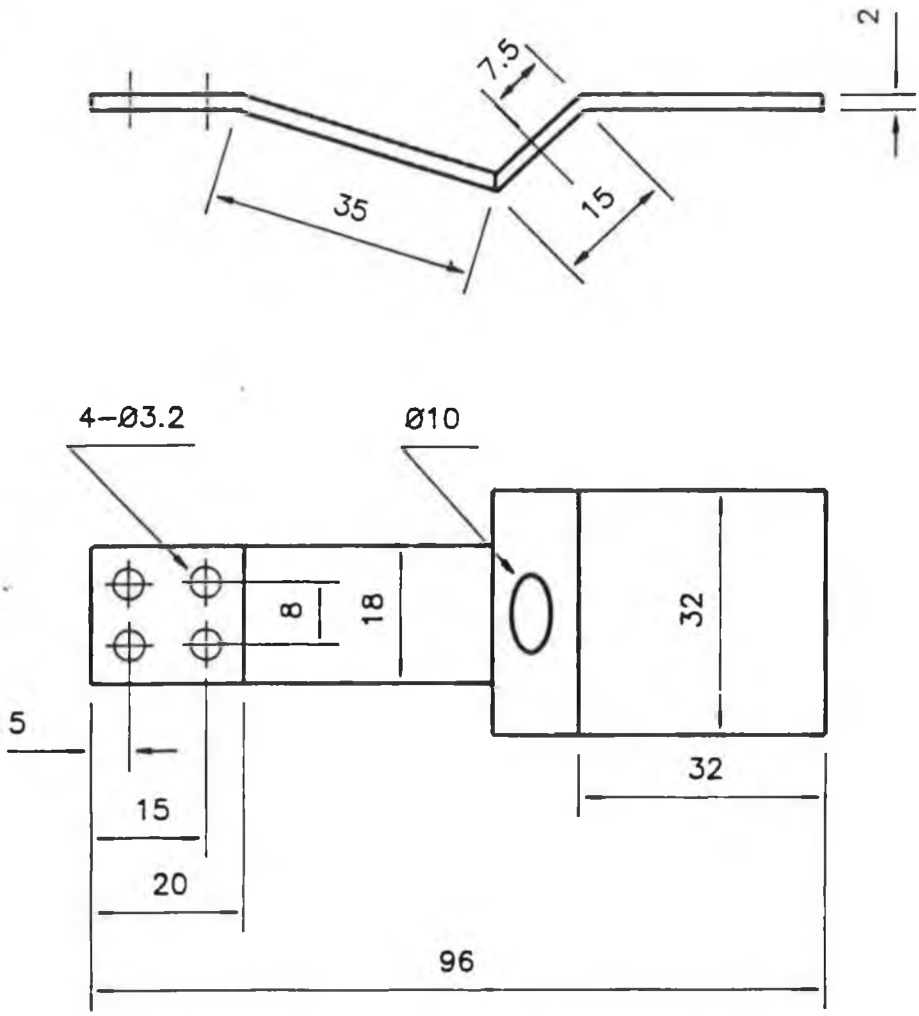
3	1	ROBOTIC PALM	H1-03
2	1	ROBOTIC HAND BODY	H1-02
1	2	ROBOTIC FINGER	H1-01
No.	QNTY.	DESCRIPTION	DRG. No.

Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:2	Title: ROBOTIC HAND (HAND-I)
	DRG. NO. H1-00

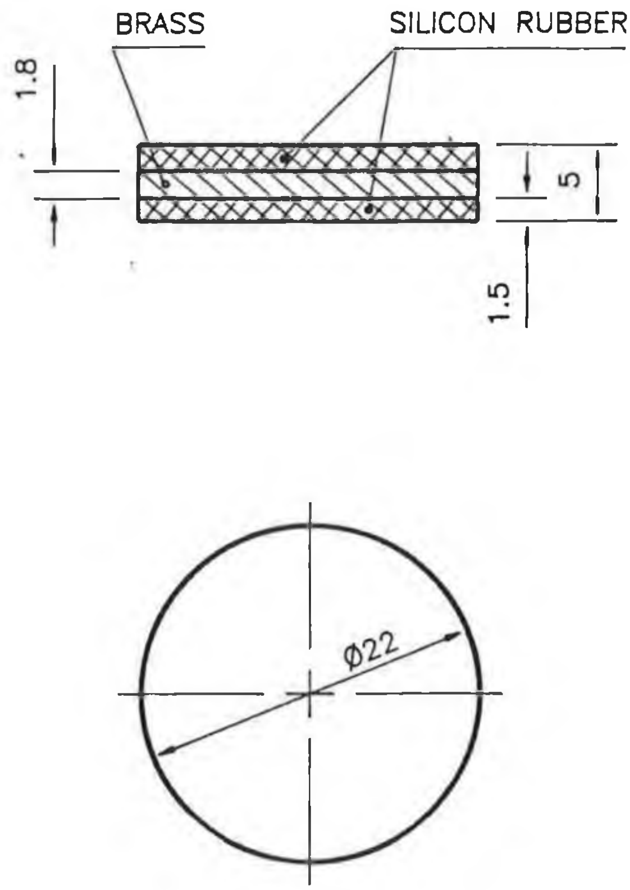


No.	QNTY.	DESCRIPTION	DRG. No.
3	1	FSR SENSOR	
2	1	CONTACT PLATE	H1-01-02
1	1	FINGER BODY	H1-01-01

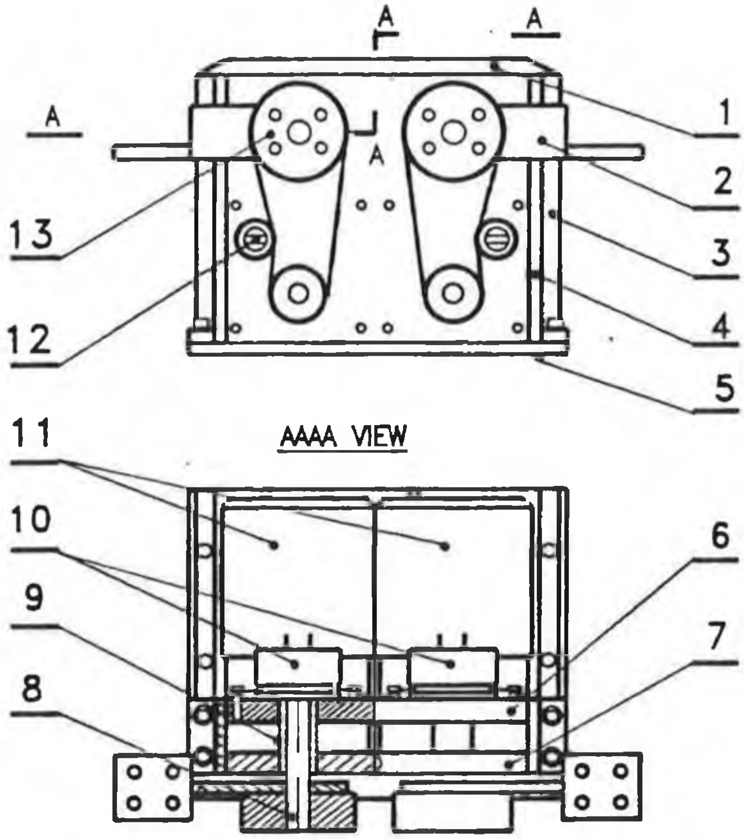
Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:1	Title: ROBOTIC FINGER
	DRG. NO. H1-01



Designed by: J. Yan	Dublin City University
Unit: mm · Scale: 1:1	Title: FINGER BODY
	MATERIAL: ALUMINIUM
	DRG. No. H1–01–01

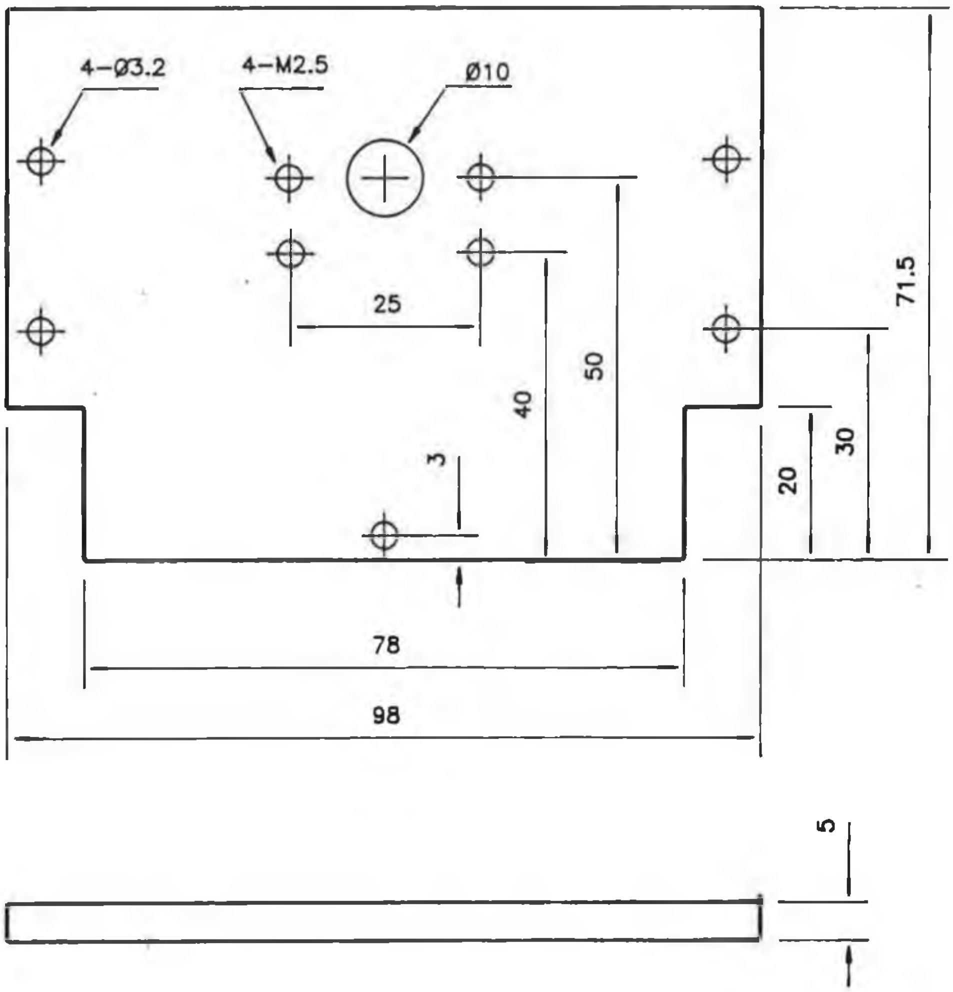


Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 2:1	Title: CONTACT PLATE
	MATERIAL: BRASS, RUBBER
	DRG. No. H1-01-02

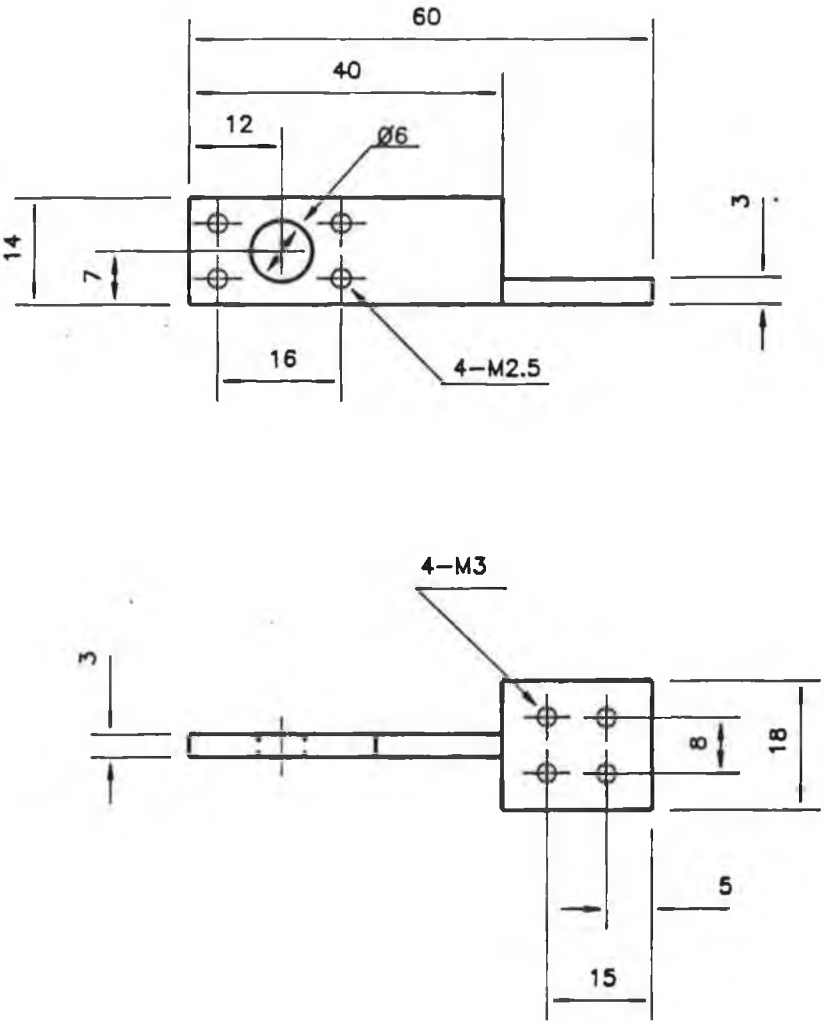


13	2	TIMING—BELT & PULLEY	
12	2	IDLER BEARING	
11	2	DC—MOTOR	
10	2	POTENTIOMETER	
9	2	BUSH	H1—02—09
8	2	FINGER SHAFT	H1—02—08
7	1	SUPPORT WALL B	H1—02—07
6	1	SUPPORT WALL A	H1—02—06
5	1	HAND BASE	H1—02—05
4	2	ENFORCEMENT WALL B	H1—02—04
3	2	ENFORCEMENT WALL A	H1—02—03
2	2	FINGER BASE	H1—02—02
1	1	PALM BASE	H1—02—01
No.	QNTY.	DESCRIPTION	DRG. No.

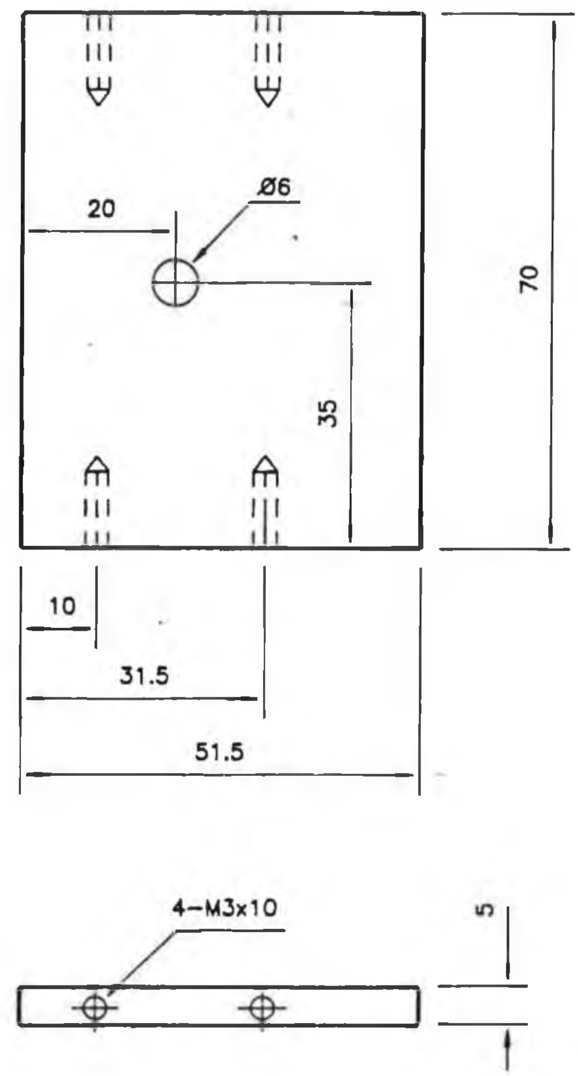
Designed by: J. Yan	Dublin City University
Unit: mm	Scale: 1:2
Title: ROBOTIC HAND BODY	DRG. NO. H1—02



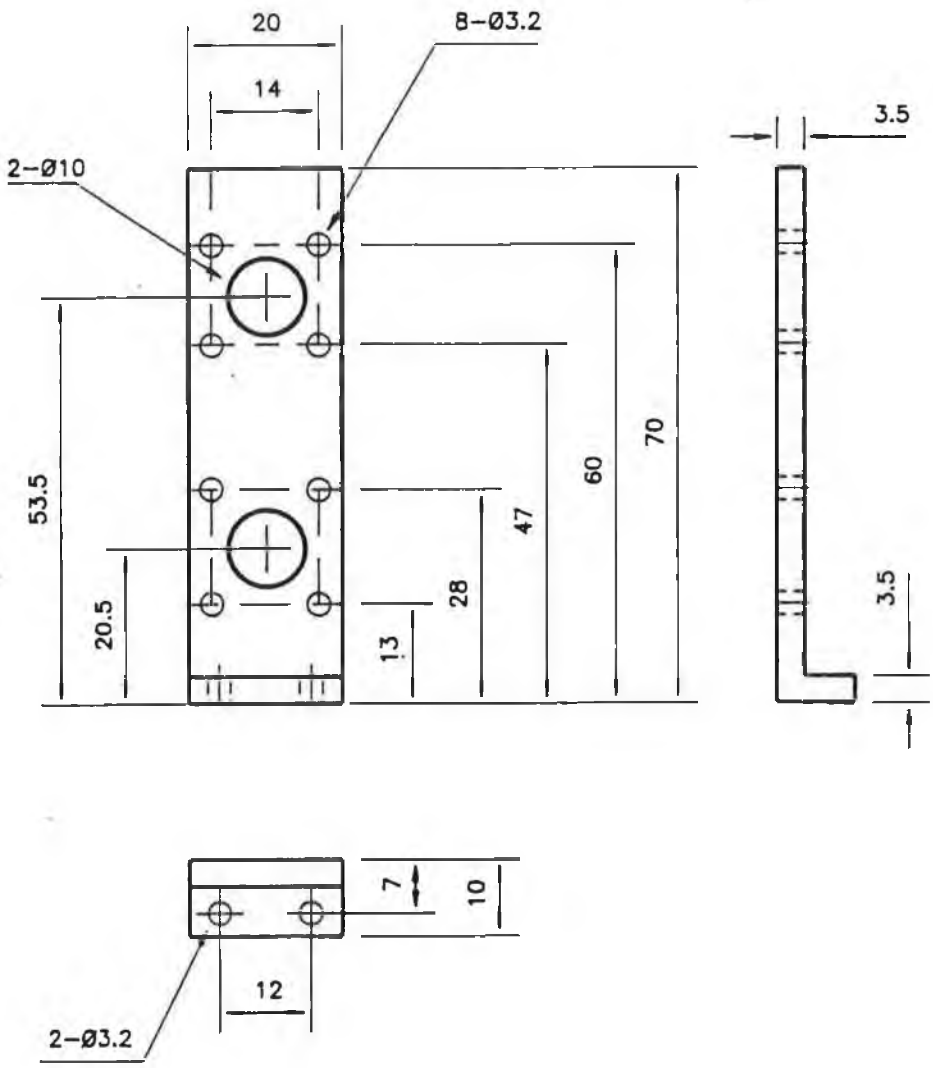
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: PALM BASE
		MATERIAL: PLASTIC
		DRG. No. H1–02–01



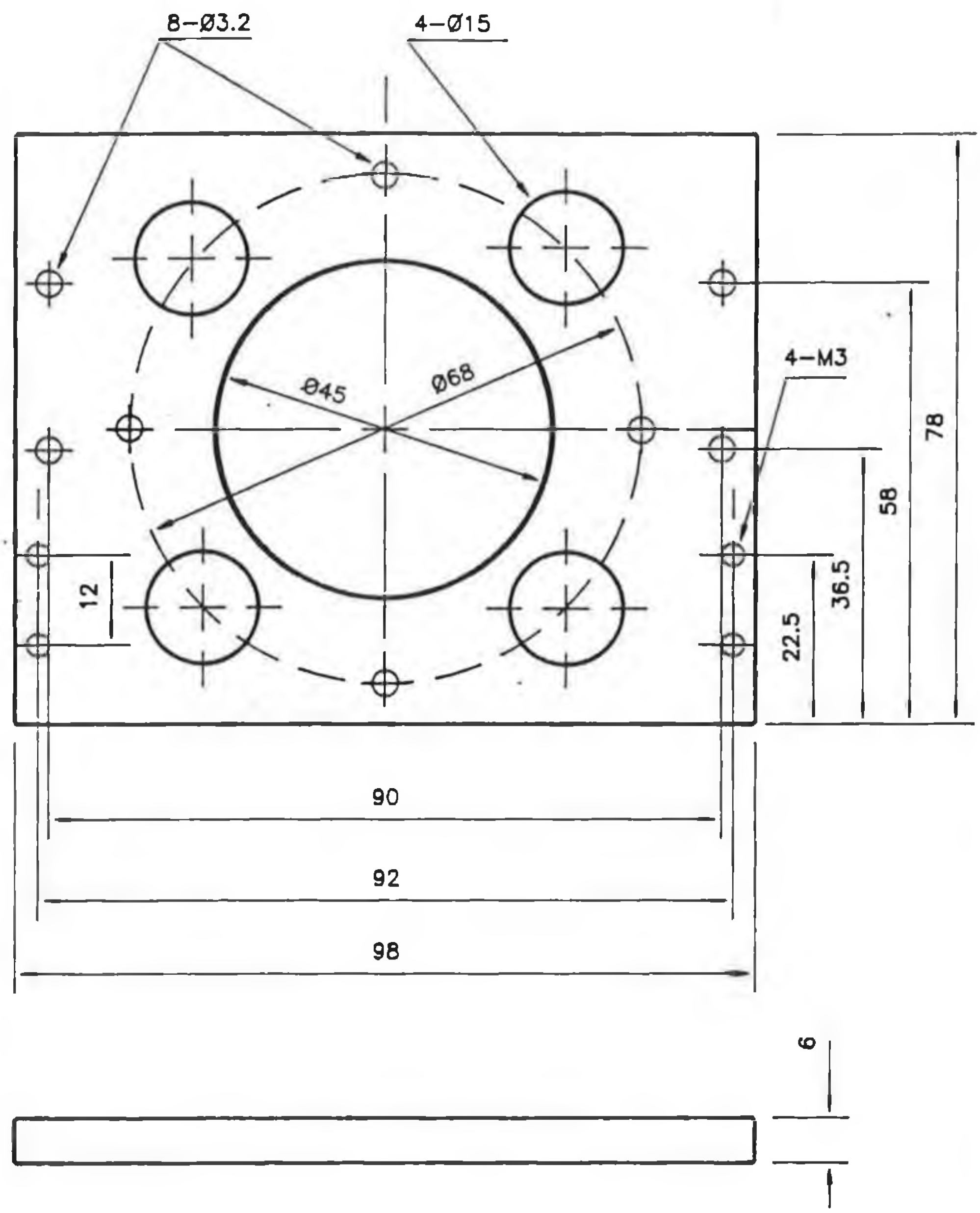
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: FINGER BASE
		MATERIAL: ALUMINIUM
		DRG. No. H1-02-02



Designed by: J. Yan		Dublin City University	
Unit: mm	Scale: 1:1	Title: ENFORCEMENT WALL A	
		MATERIAL:	PLASTIC
		DRG. No.	H1–02–03



Designed by: J. Yan		Dublin City University
Unit: .mm	Scale: 1:1	Title: ENFORCEMENT WALL B
		MATERIAL: STEEL
		DRG. No. H1−02−04



Designed by: J. Yan

Dublin City University

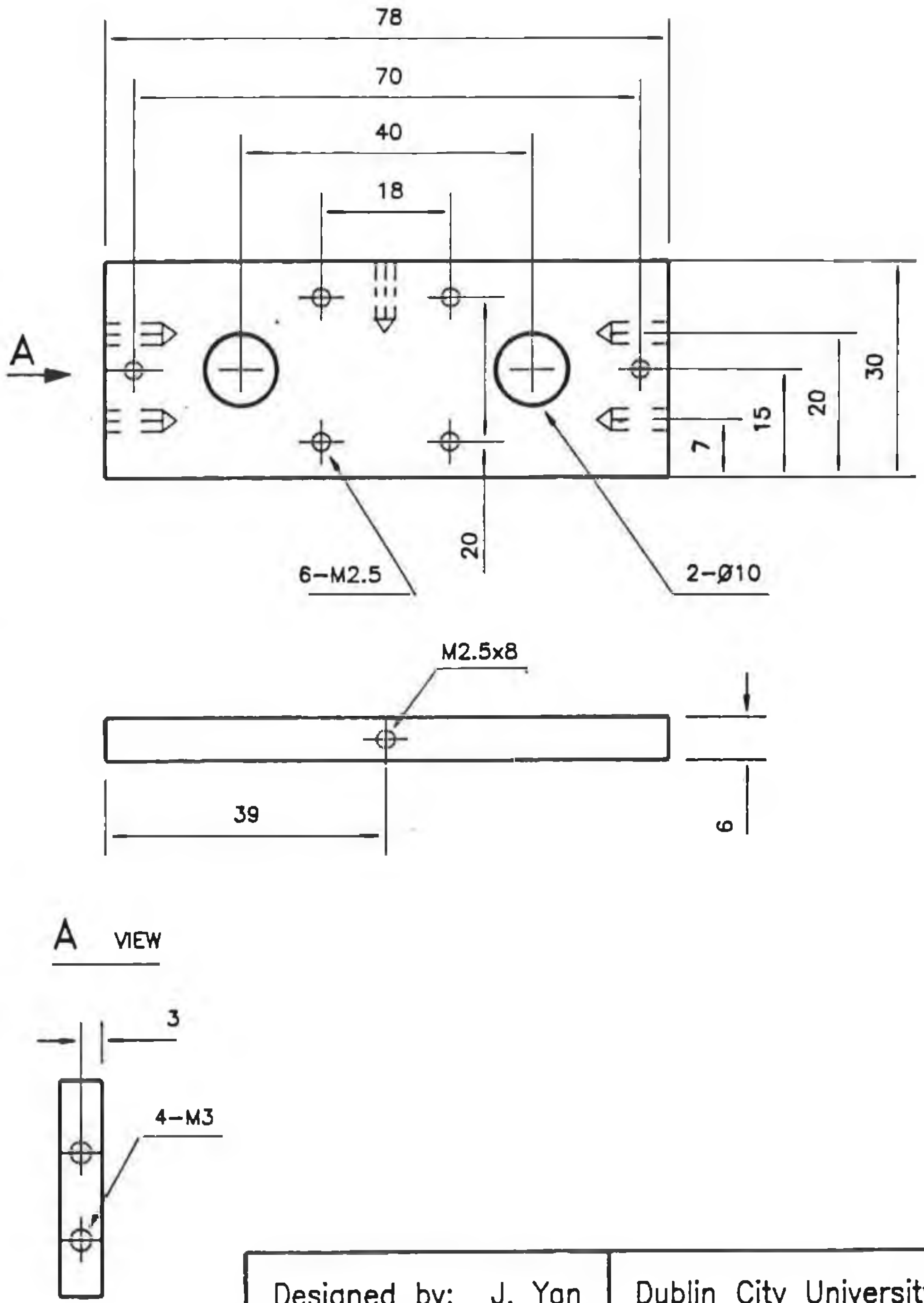
Unit: mm

Scale: 1:1

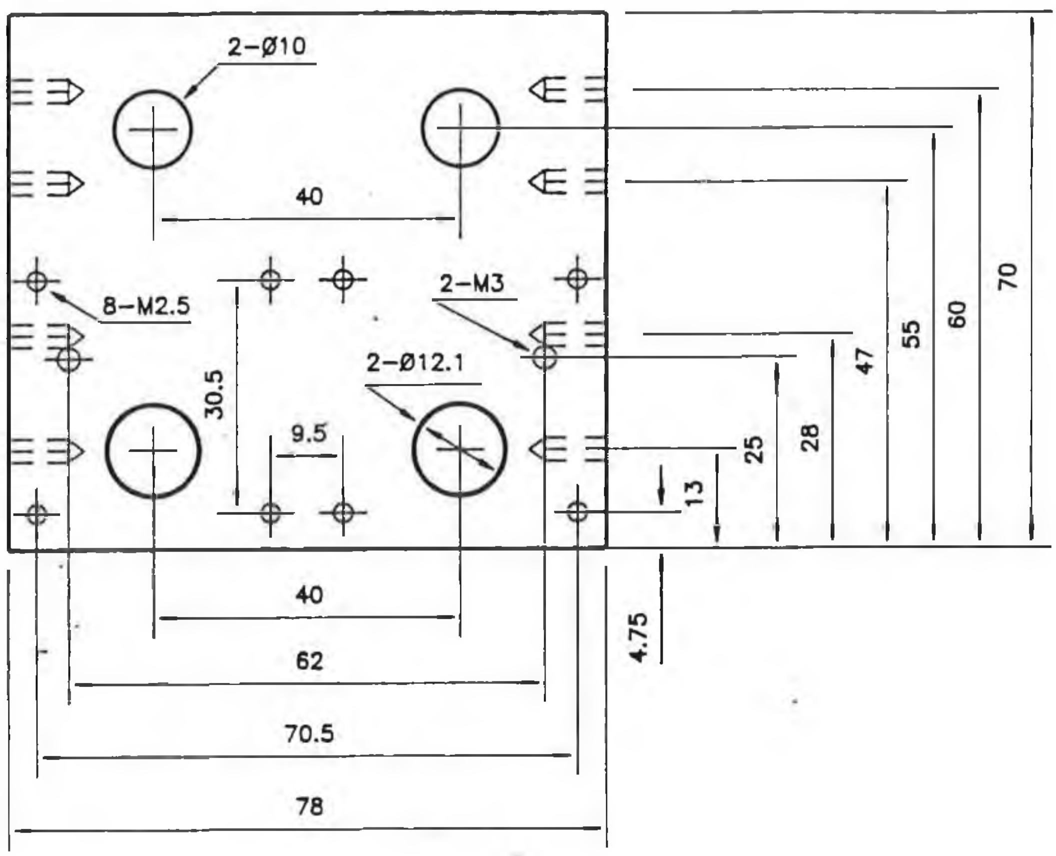
Title: HAND BASE

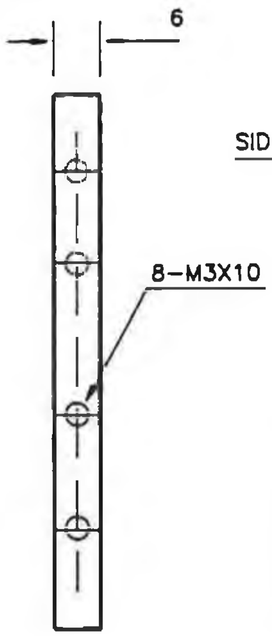
MATERIAL: ALUMINIUM

DRG. No. H1–02–05



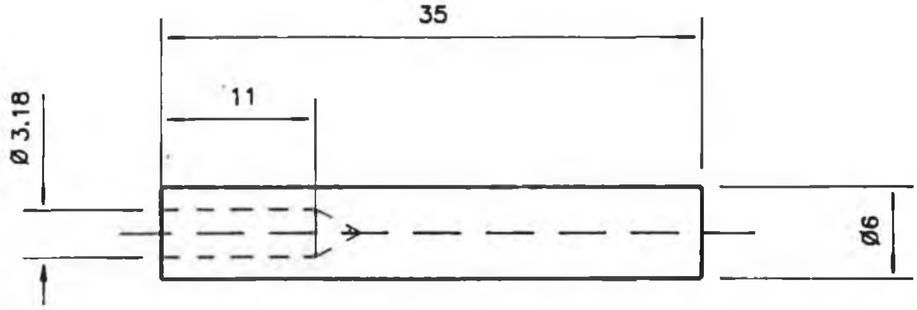
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: SUPPORT WALL A
		MATERIAL: ALUMINIUM
		DRG. No. H1–02–06



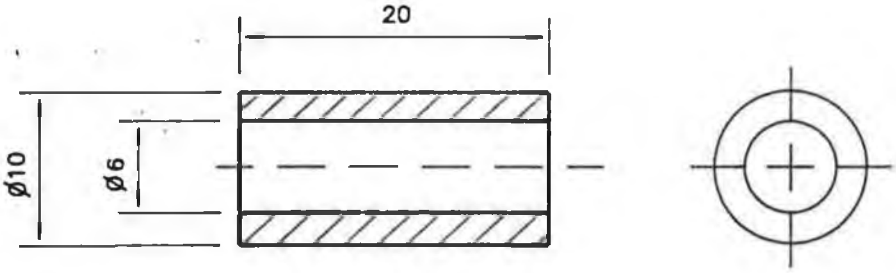


SIDE VIEW

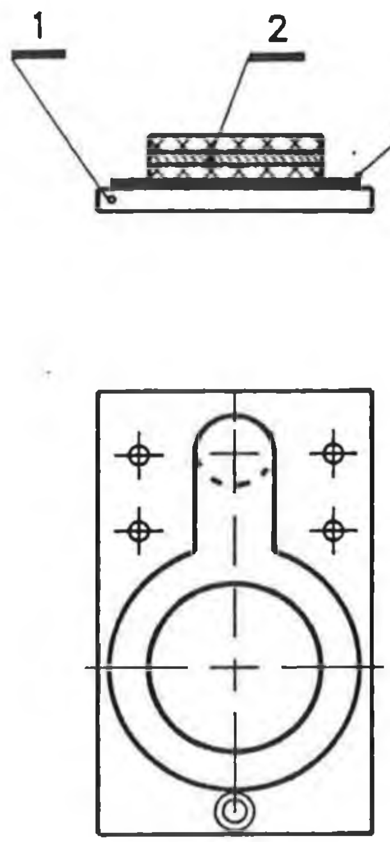
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: SUPPORT WALL B
		MATERIAL: ALUMINIUM
		DRG. No. H1-02-07



Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 2:1	Title: FINGER SHAFT
	MATERIAL: STEEL
	DRG. No. H1-02-08

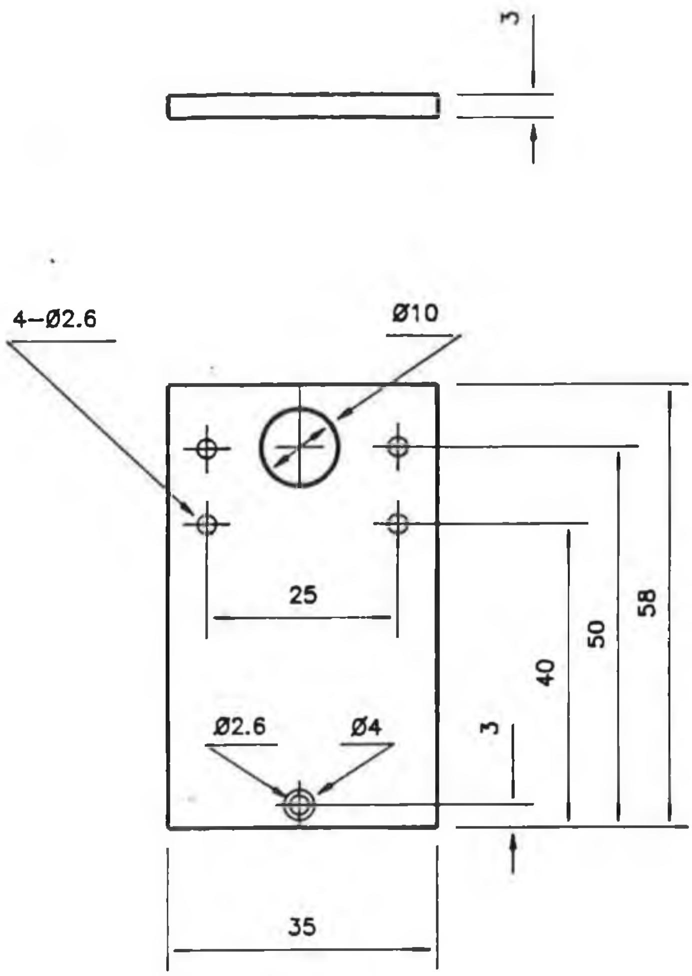


Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 2:1	Title: BUSH
		MATERIAL: BRASS
		DRG. No. H1-02-09

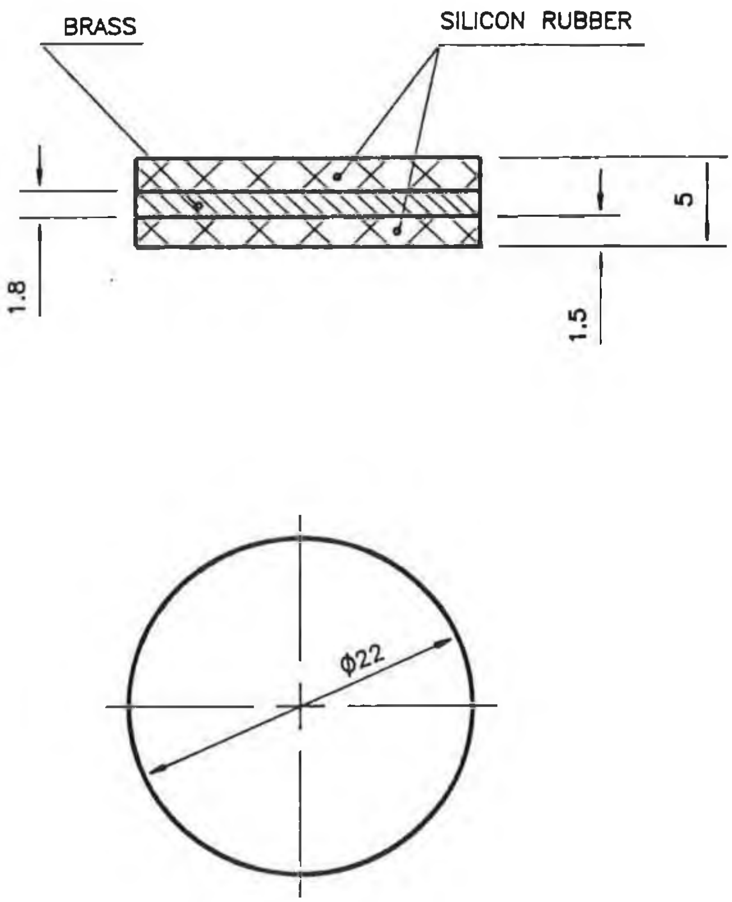


3	1	FSR SENSOR	
2	1	CONTACT PLATE	H1-03-02
1	1	PALM BODY	H1-03-01
No.	QNTY.	DESCRIPTION	DRG. No.

Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: ROBOTIC PALM
		DRG. NO. H1-03



Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: PALM BODY
		MATERIAL: ALUMINIUM
		DRG. No. H1–03–01

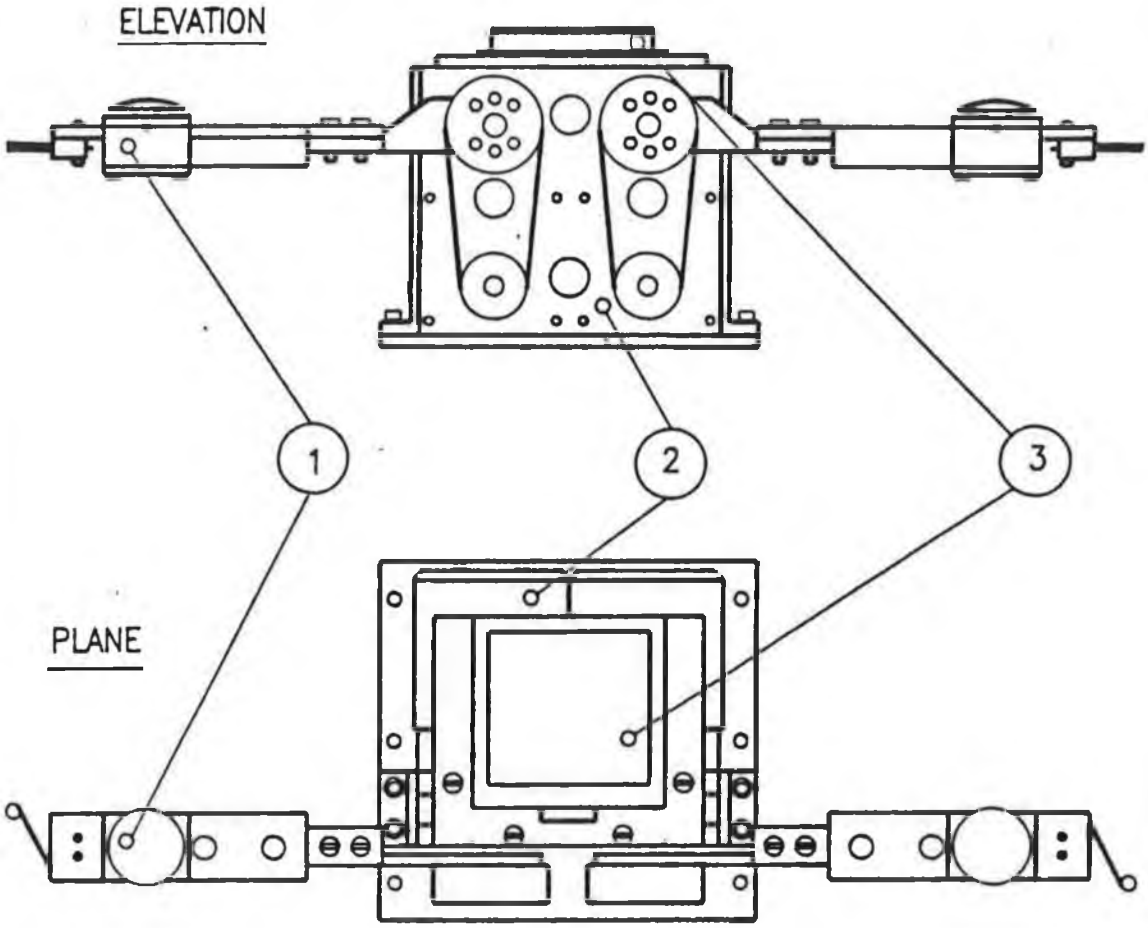


Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 2:1	Title: CONTACT PLATE
	MATERIAL: BRASS, RUBBER
	DRG. No. H1-03-02

APPNEDIX C-2 Mechanical drawings for HAND-II

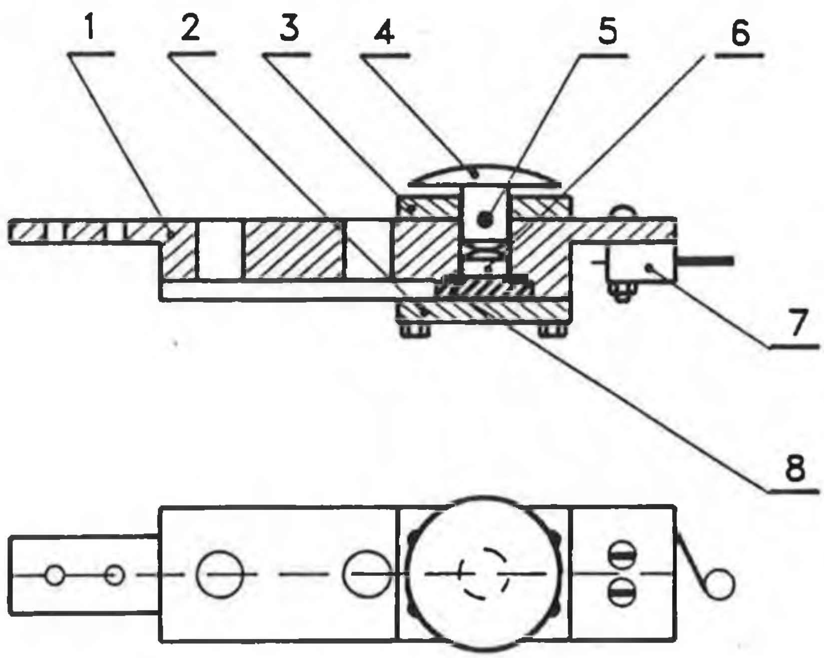
INDEX TO DESIGN DRAWINGS for HAND-II

Drawing No.	Title
H2-00	ROBOTIC HAND-II
H2-01	ROBOTIC FINGER
H2-01-01	FINGER BODY
H2-01-02	FINGER BOTTOM
H2-01-03	FINGER TOP
H2-01-04	TOUCH CAP
H2-01-05	STOP BAR
H2-01-06	CONTACT CYLINDER
H2-02	ROBOTIC HAND BODY
H2-02-01	FINGER BASE
H2-02-02	ENFORCEMENT WALL
H2-02-03	HAND BASE
H2-02-04	SUPPORT WALL A
H2-02-05	SUPPORT WALL B
H2-02-06	FINGER SHAFT
H2-02-07	BUSH
H2-03	ROBOTIC PALM
H2-03-01	PALM BODY
H2-03-02	CONTACT PLATE
H2-04	MOUNTING INTERFACE



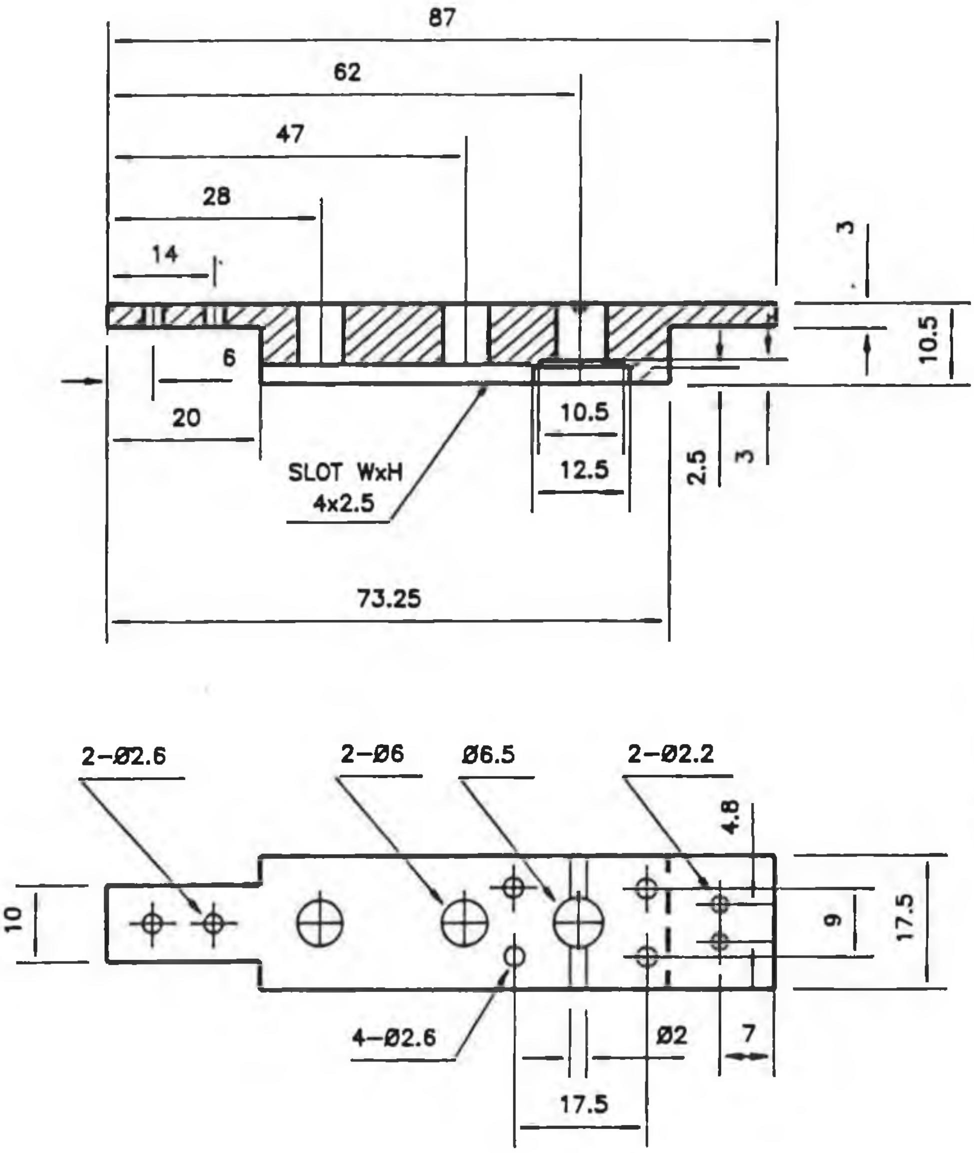
No.	QNTY.	DESCRIPTION	DRG. No.
3	1	ROBOTIC PALM	H2-03
2	1	ROBOTIC HAND BODY	H2-02
1	2	ROBOTIC FINGER	H2-01

Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:2	Title: ROBOTIC HAND (HAND-II)
		DRG. NO. H2-00

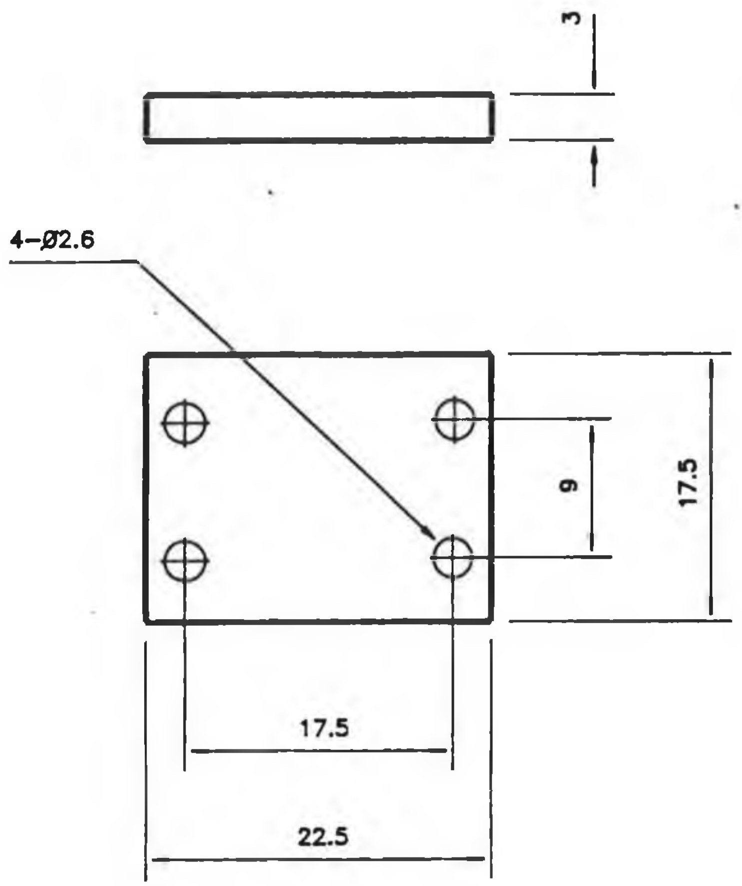


8	1	LOAD CELL	
7	1	MICROSWITCH	
6	1	CONTACT CYLINDER	H2-01-06
5	1	STOP BAR	H2-01-05
4	1	TOUCH CAP	H2-01-04
3	1	FINGER TOP	H2-01-03
2	1	FINGER BOTTOM	H2-01-02
1	1	FINGER BODY	H2-01-01
No.	QNTY.	DESCRIPTION	DRG. No.

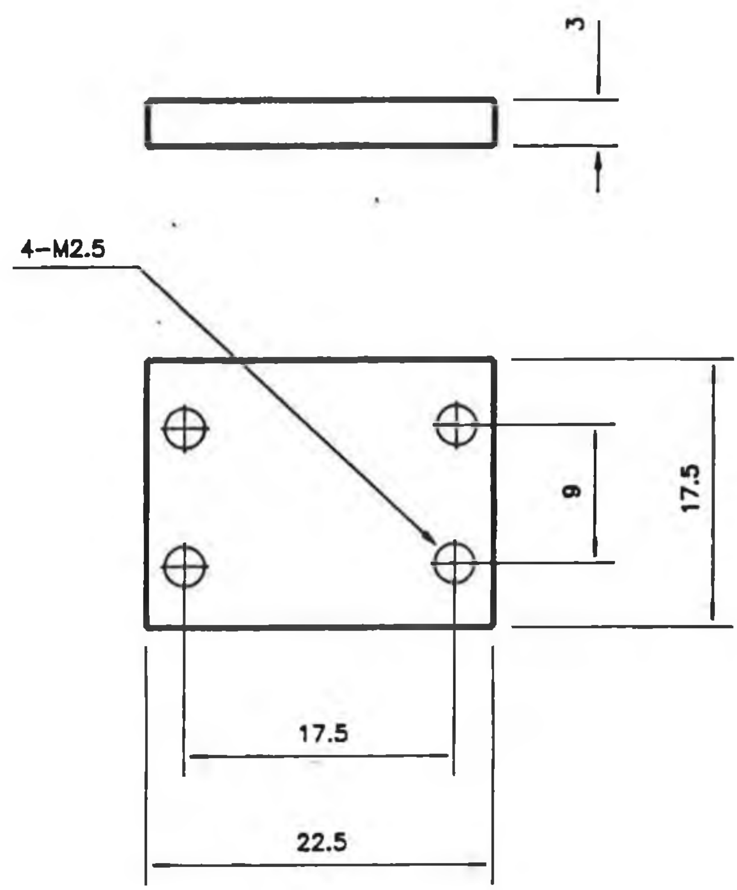
Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:1	Title: ROBOTIC FINGER
	DRG. NO. H2-01



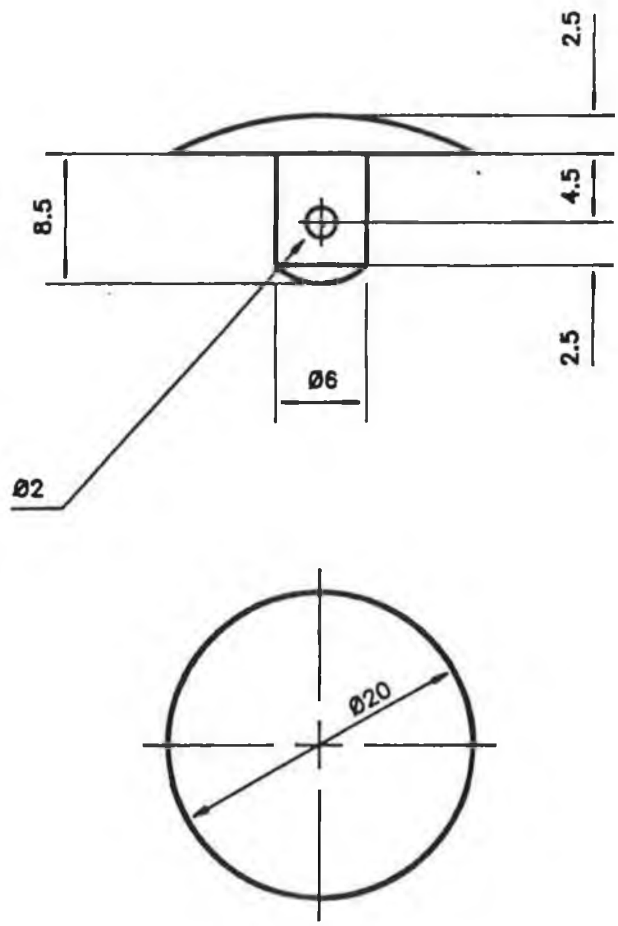
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: FINGER BODY
		MATERIAL: ALUMINIUM
		DRG. No. H2-01-01



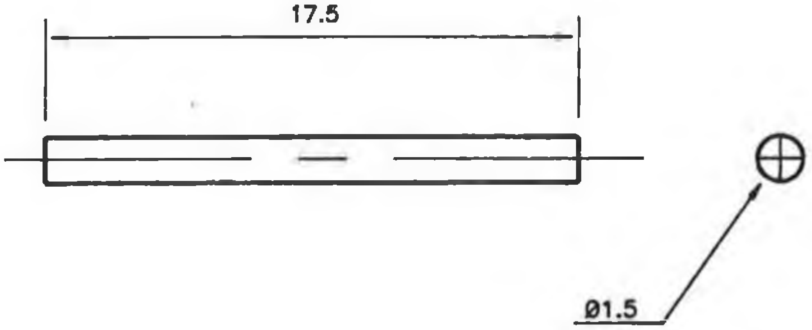
Designed by: J. Yan	Dublin City University
Unit: mm — Scale: 2:1	Title: FINGER BOTTOM
	MATERIAL: ALUMINIUM
	DRG. No. H2-01-02



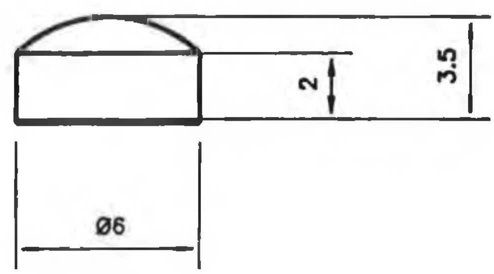
Designed by: J. Yan		Dublin City University	
Unit: mm	Scale: 2:1	Title:	FINGER TOP
		MATERIAL:	ALUMINIUM
		DRG. No.	H2-02-03



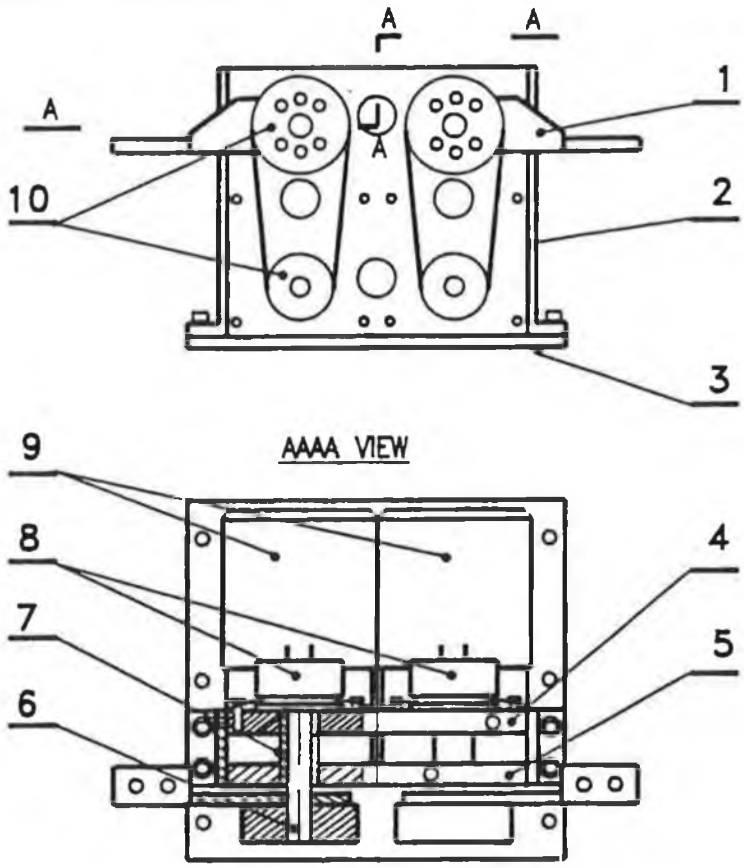
Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 2:1	Title: TOUCH CAP
	MATERIAL: STEEL
	DRG. No. H2-01-04



Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 4:1	Title: STOP BAR
		MATERIAL: STEEL
		DRG. No. H2-01-05

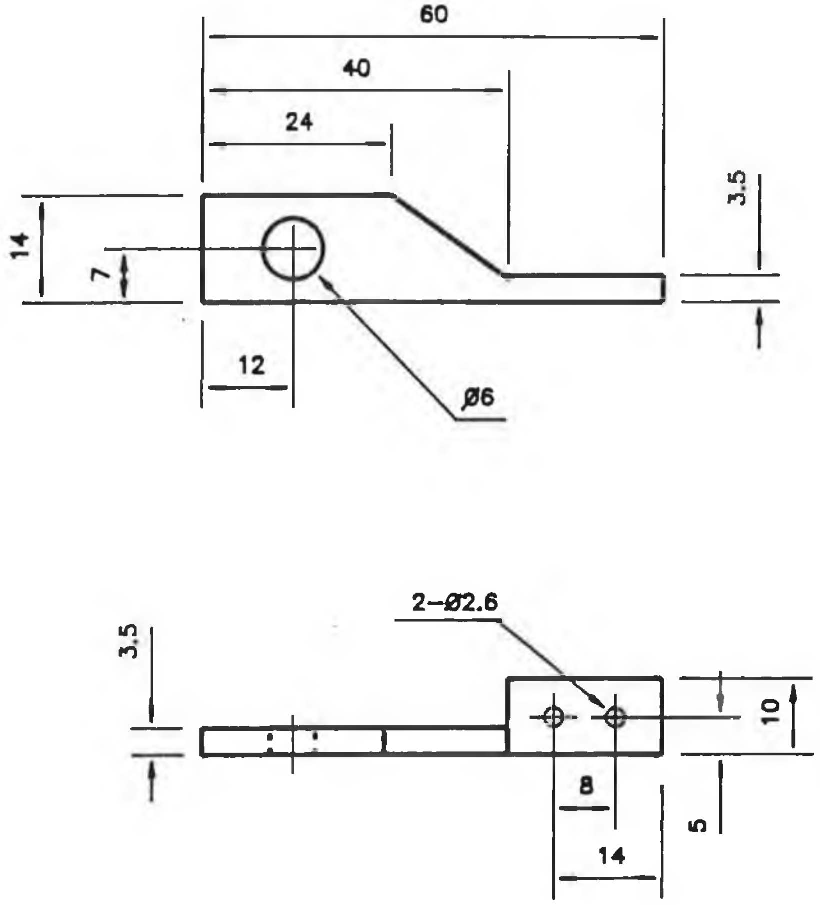


Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 4:1	Title: CONTACT CYLINDER
	MATERIAL: STEEL
	DRG. No. H2-01-06

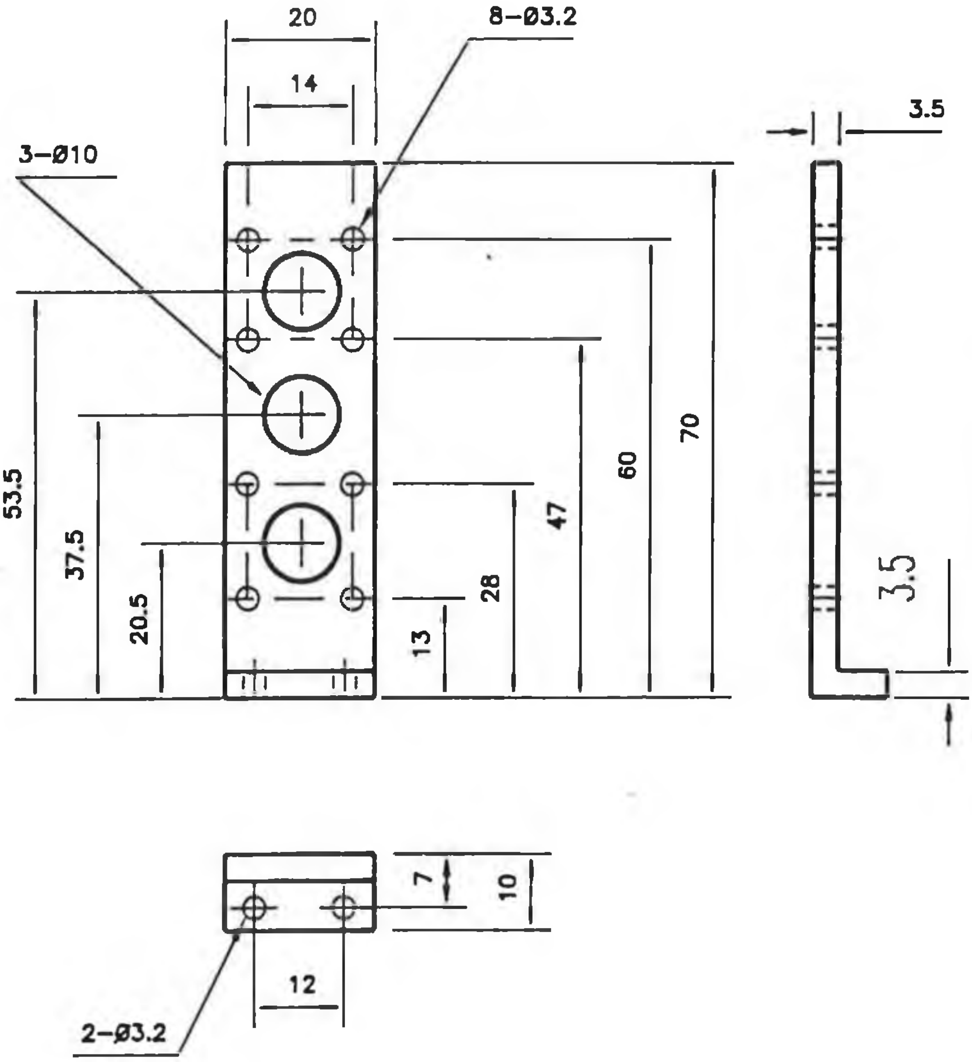


No.	QNTY.	DESCRIPTION	DRG. No.
10	2	TIMING-BELT & PULLEY	
9	2	DC – MOTOR	
8	2	POTENTIOMETER	
7	2	BUSH	H2-02-07
6	2	FINGER SHAFT	H2-02-06
5	1	SUPPORT WALL B	H2-02-05
4	1	SUPPORT WALL A	H2-02-04
3	1	HAND BASE	H2-02-03
2	2	ENFORCEMENT WALL	H2-02-02
1	2	FINGER BASE	H2-02-01

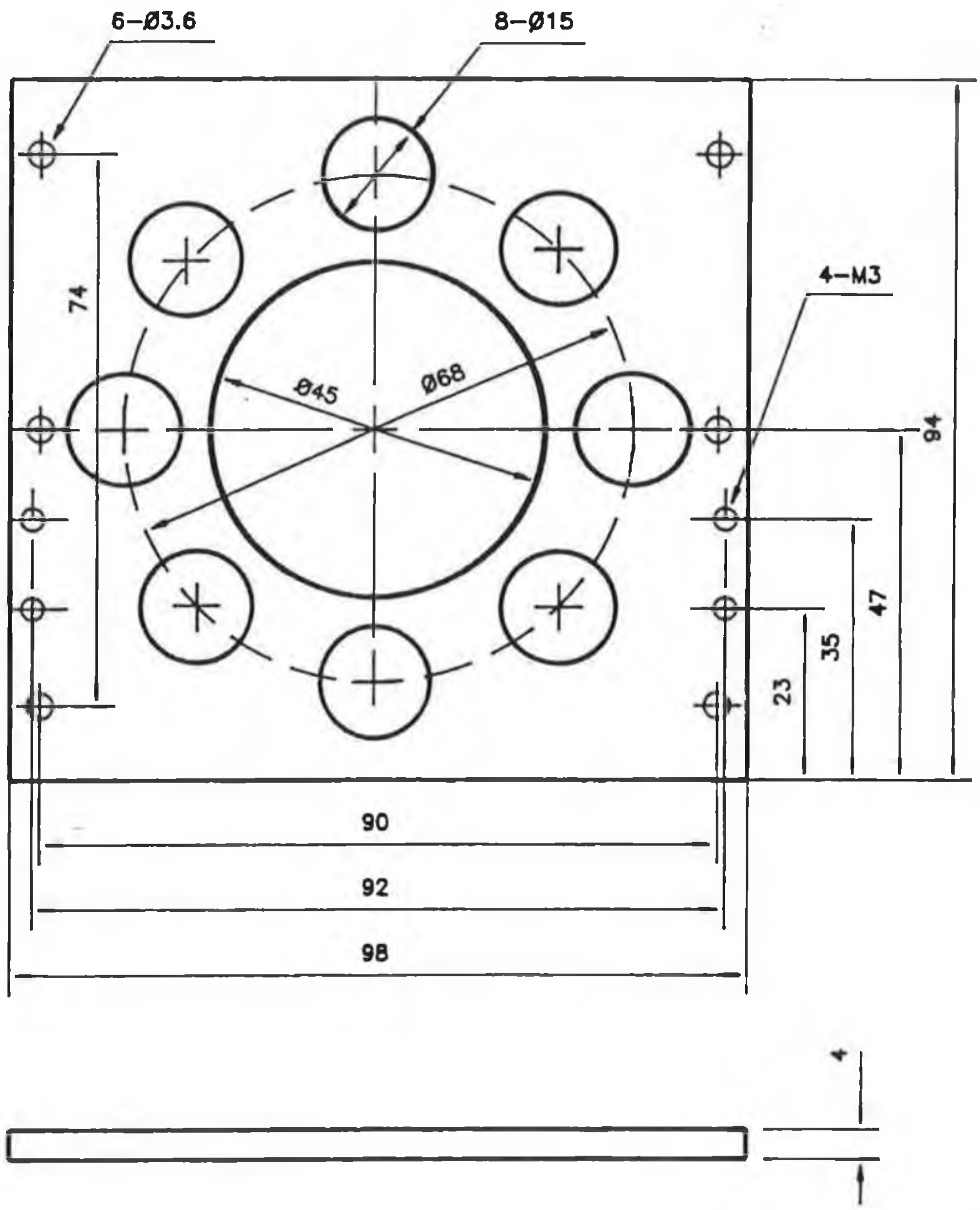
Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:2	Title: ROBOTIC HAND BODY
	DRG. NO. H2-02



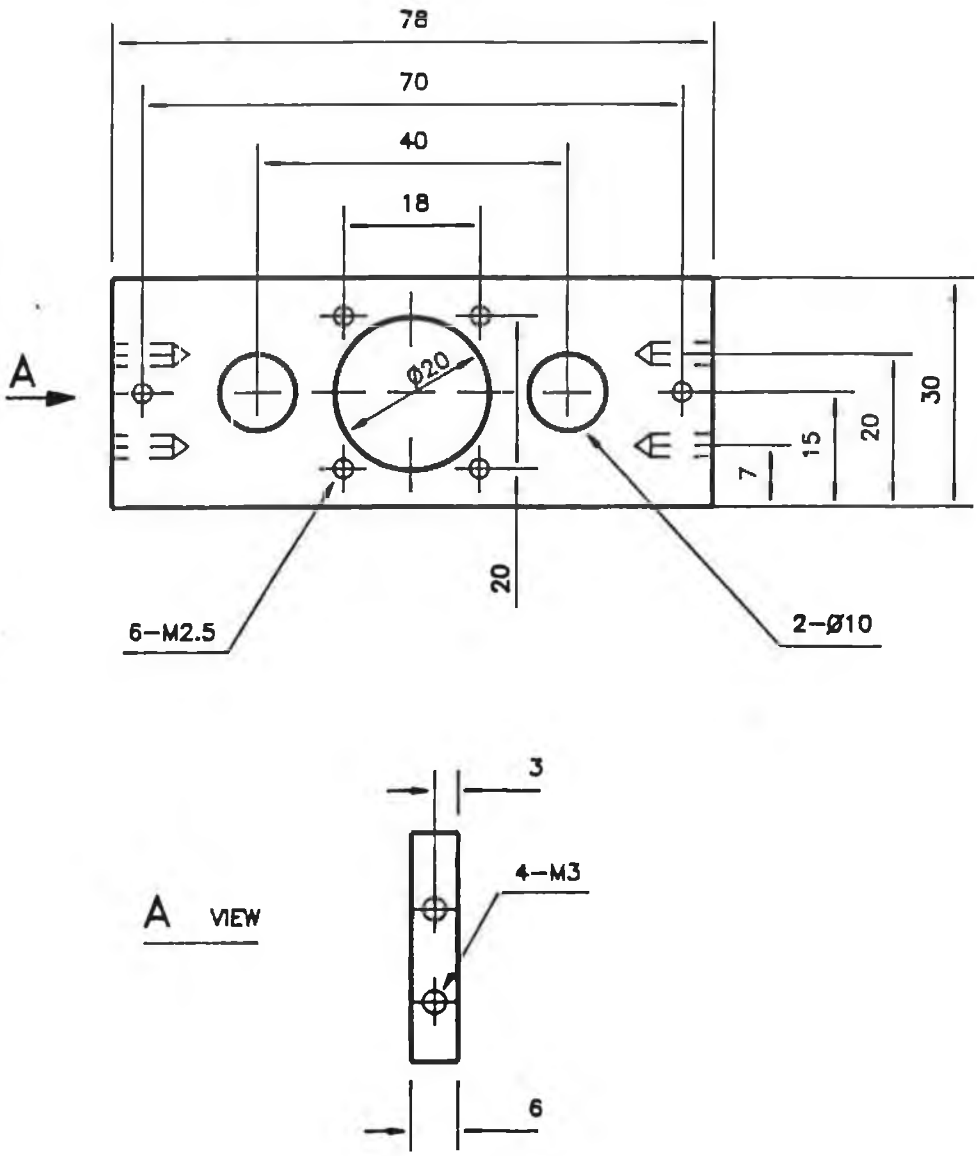
Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: FINGER BASE
		MATERIAL: ALUMINIUM
		DRG. No. H2-02-01



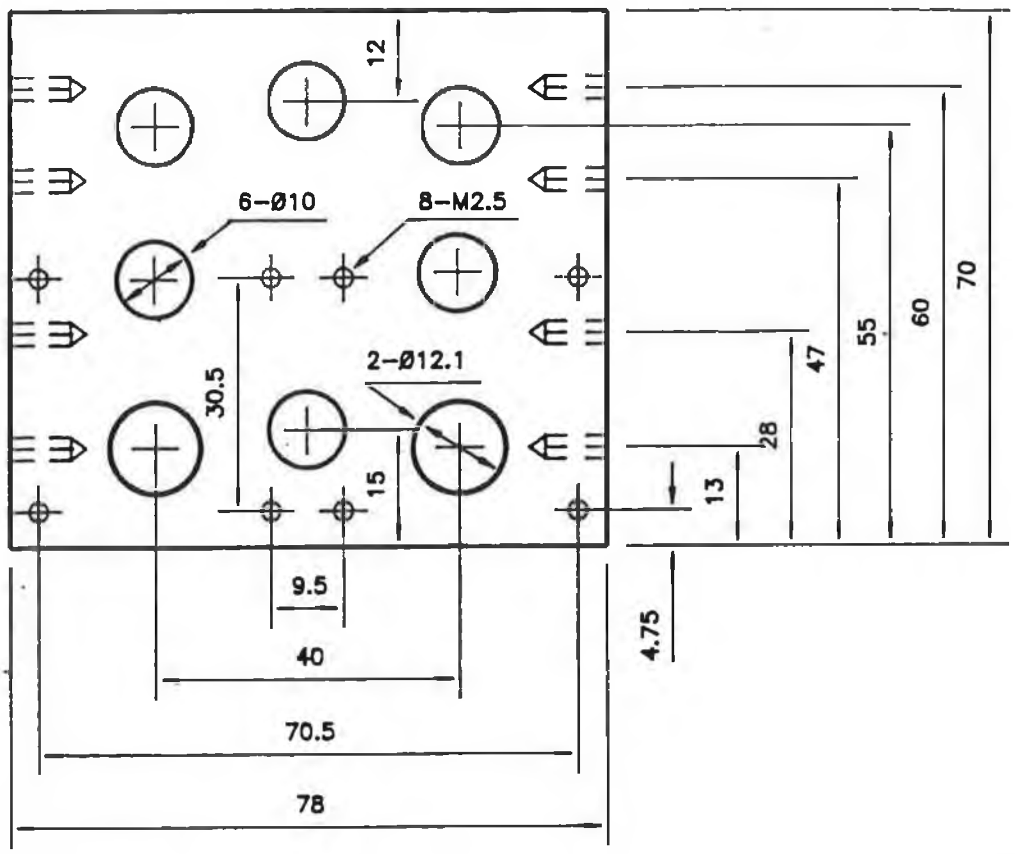
Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:1	Title: ENFORCEMENT WALL
	MATERIAL: ALUMINIUM
	DRG. No. H2-02-02

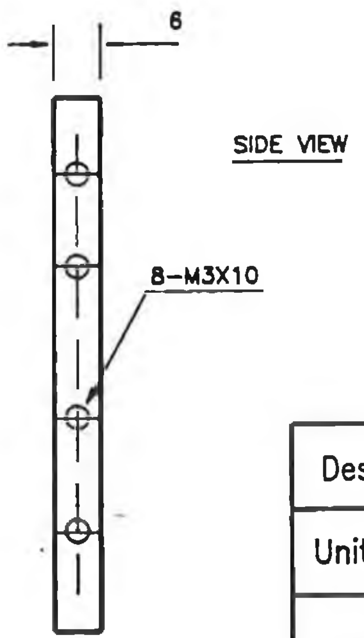


Designed by: J. Yan	Dublin City University
Unit: mm Scale: 1:1	Title: HAND BASE
	MATERIAL: ALUMINIUM
	DRG. No. H2-02-03

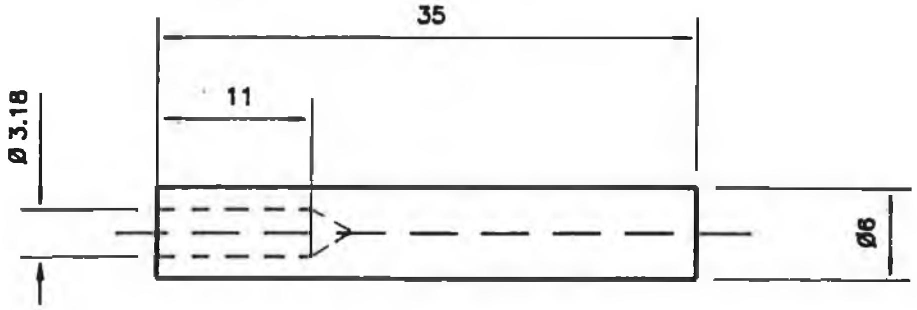


Designed by: J. Yan	Dublin City University
Unit: mm Scale: 1:1	Title: SUPPORT WALL A
	MATERIAL: ALUMINIUM
	DRG. No. H2-02-04

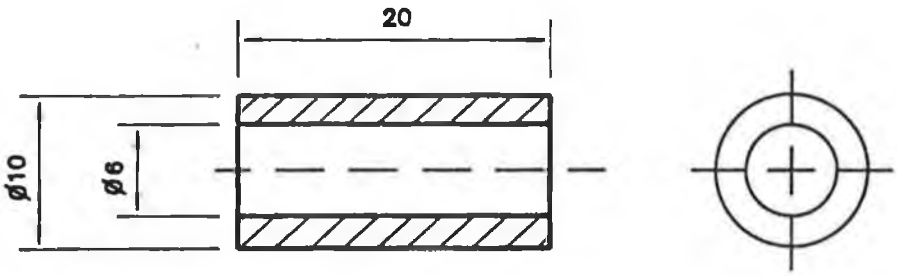




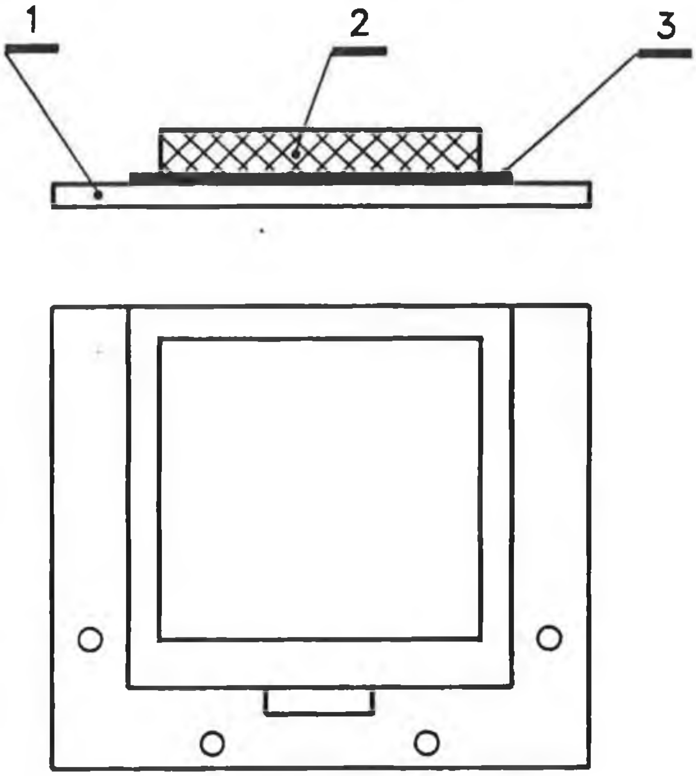
Designed by: J. Yan	Dublin City University
Unit: mm Scale: 1:1	Title: SUPPORT WALL B
	MATERIAL: ALUMINIUM
	DRG. No. H2-02-05



Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 2:1	Title: FINGER SHAFT
		MATERIAL: STEEL
		DRG. No. H2-02-06

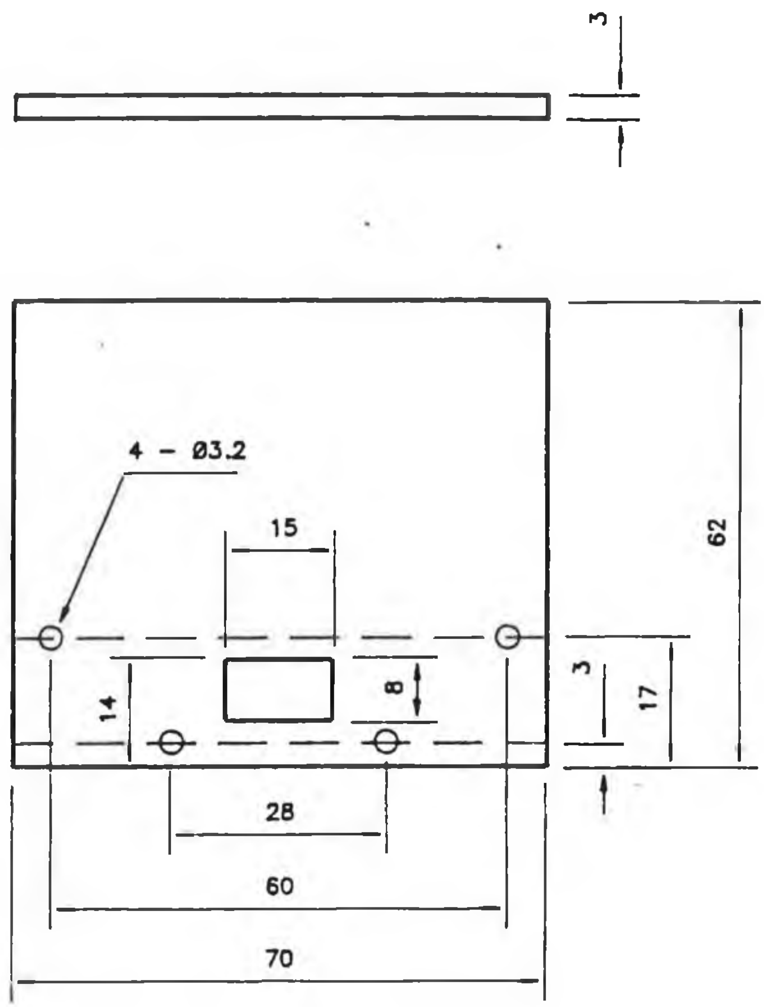


Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 2:1	Title: BUSH
	MATERIAL: BRASS
	DRG. No. H2-02-07

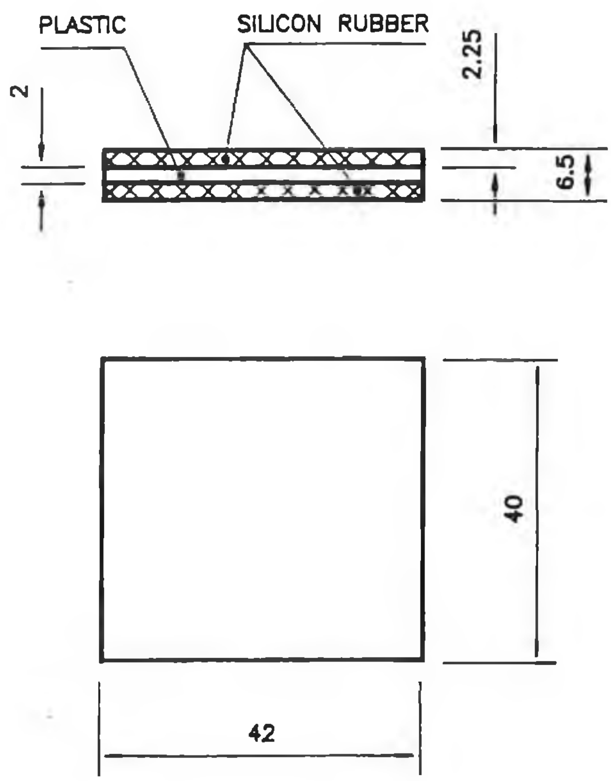


No.	QNTY.	DESCRIPTION	DRG. No.
3	1	FSR SENSOR	
2	1	CONTACT PLATE	H2-03-02
1	1	PALM BODY	H2-03-01

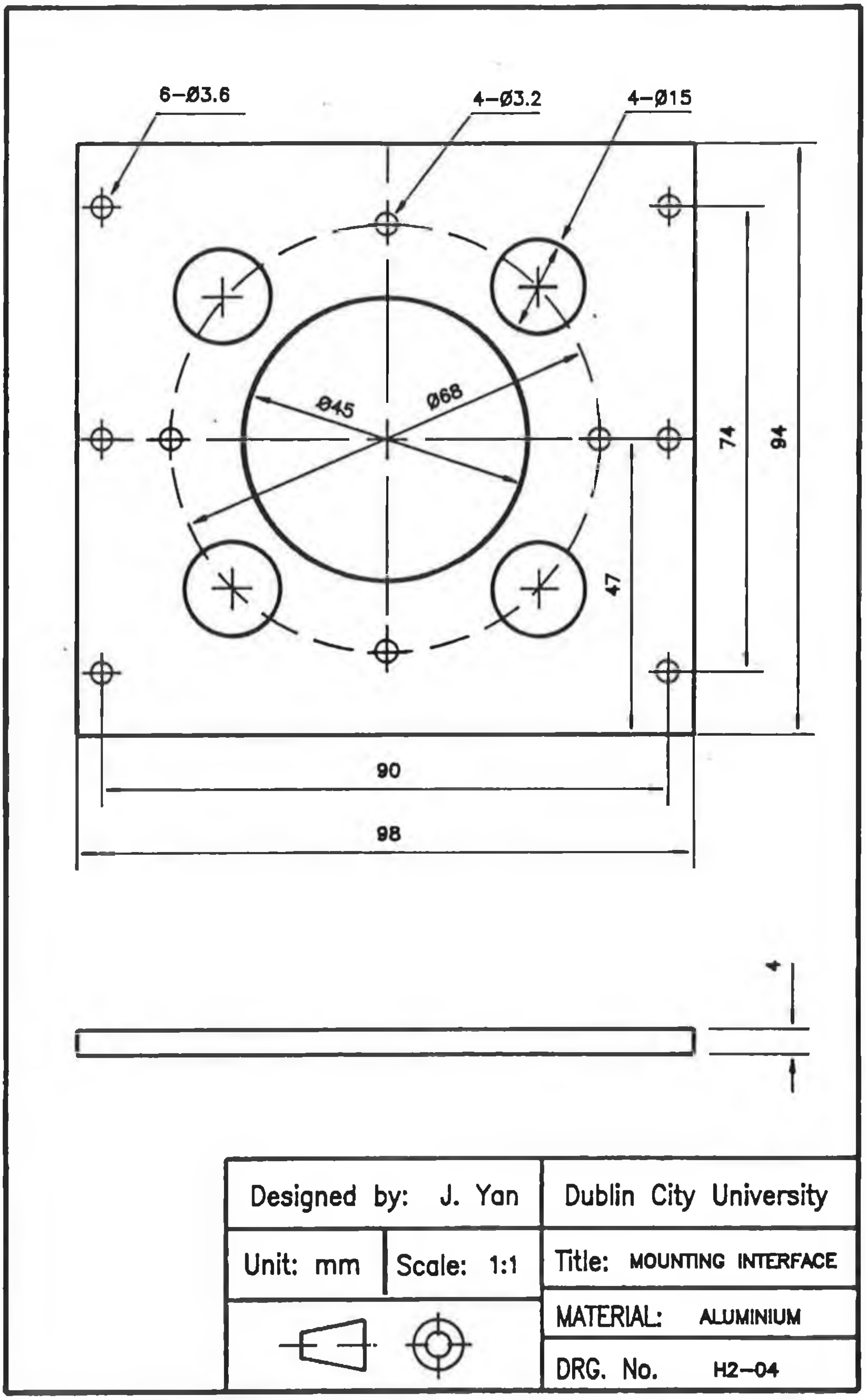
Designed by: J. Yan	Dublin City University
Unit: mm Scale: 1:1	Title: ROBOTIC PALM
	DRG. NO. H2-03



Designed by: J. Yan		Dublin City University
Unit: mm	Scale: 1:1	Title: PALM BODY
		MATERIAL: ALUMINIUM
		DRG. No. H2-03-01



Designed by: J. Yan	Dublin City University
Unit: mm / Scale: 1:1	Title: CONTACT PLATE
	MATERIAL: PLASTIC, RUBBER
	DRG. No. H2-03-02


6-Ø3.6
4-Ø3.2
4-Ø15
Ø45
Ø68
74
94
47
90
98
4
Designed by: J. Yan
Dublin City University
Unit: mm
Scale: 1:1
Title: MOUNTING INTERFACE
MATERIAL: ALUMINIUM
DRG. No. H2-04

APPENDIX D-1 Electronic connections for the PCB of the sensor amplifiers

D-1-1 Pin assignment of PCB for sensors in HAND-I

Pin No. of PCB	signals	Descriptions
1	X	
2	${}^{2}V_{FSR}$	FSR output. Not used.
3	X	
4	X	
5	${}^{2}FSR$	FSR input. Not used.
6	X	
7	${}^{4}V_{FSR}$	FSR #1 output. Linked to A/D CH2
8	${}^{4}FSR$	FSR input. Linked to FSR #1
9	${}^{6}V_{FSR}$	FSR output. Not used.
10	${}^{6}FSR$	FSR input. Not used.
11	${}^{8}V_{FSR}$	FSR output. Not used.
12	X	
13	${}^{8}FSR$	FSR input. Not used.
14	GND	Ground linked to A/D L.L.GND
15	X	

16	${}^{2}V_{\theta}$	Potentiometer #2 output to A/D CH1
17	$V_{\theta 2}$	From potentiometer #2
18	${}^{1}V_{\theta}$	Potentiometer #1 output to A/D CH0
19	$V_{\theta 1}$	From potentiometer #1
20	V_{CC}^{+}	Power supply (+5V)
21	V_{CC}^{-}	Power supply (-5V)
22	V_{CC}^{+}	Power supply (+5V)
23	${}^{7}FSR$	FSR input. Not used.
24	GND	GND of power supply
25	${}^{7}V_{FSR}$	FSR output. Not used.
26	${}^{5}FSR$	FSR input. Not used.
27	${}^{5}V_{FSR}$	FSR output. Not used.
28	${}^{3}FSR$	FSR #2 input. Linked to FSR #2.
29	X	
30	${}^{3}V_{FSR}$	FSR #2 output. Linked to A/D CH3.
31	${}^{1}FSR$	FSR #3 input. Linked to FSR #3.
32	${}^{1}V_{FSR}$	FSR #3 output. Linked to A/D CH4.

Where

X -- not connected

D-1-2 Pin assignment of PCB for sensors in HAND-II

Pin No. of PCB	Sensors	Descriptions
1	${}^{1}V_{\theta}$	Potentiometer #1 output to A/D CH0
2	$V_{\theta 1}$	From potentiometer #1
3	X	
4	V_{θ}	From Potentiometer #2
5	${}^{2}V_{\theta}$	Potentiometer #2 output to A/D CH1
6	X	
7	${}^{2}V_{out}^{+}$	Force #2 output to A/D CH2
8	X	
9	${}^{2}V_{out}^{-}$	Force #2 GND
10	${}^{2}OUT^{-}$	Load cell #2 output GND
11	X	
12	${}^{2}OUT^{+}$	Load cell #2 output to amplifier
13	X	
14	+5V	Power supply
15	X	
16	X	

17	X	
18	-5V	Power supply
19	X	
20	$^{1}OUT^{+}$	Load cell #1 output to amplifier
21	X	
22	$^{1}OUT^{-}$	Load cell #1 GND
23	$^{1}V_{out}^{-}$	Force #1 output GND
24	$^{1}V_{out}^{+}$	Force #1 output to A/D CH3
25	X	
26	$^{2}V_{FSR}$	FSR output. Linked to A/D CH5
27	^{2}FSR	FSR input. Not used.
28	GND	Ground linked to A/D L.L.GND
29	X	
30	^{1}FSR	FSR #1 input. Linked to FSR #1
31	$^{1}V_{FSR}$	FSR (palm) #1 output to A/D CH4
32	X	

Where

X -- Not connected.

APPENDIX D-2 Electronic connections for the PCB of the motor drive circuit for HAND-I & HAND-II

Pin No. of PCB	signals	Descriptions
1	m_1^+	Linked to motor #1 terminal
2	$^2V_{in}$	Control voltage from D/A CH5
3	$^1V_{in}$	Control voltage from D/A CH4
4	m_2^+	Linked to motor #2 terminal
5	m_1^-	Linked to motor #1 terminal
6	m_2^-	Linked to motor #2 terminal
7	+10V	Power supply
8	-10V	Power supply
9	+5V	Power supply
10	X	
11	X	
12	SW5	Linkable to microswitch
13	SW6	Linkable to microswitch
14	SW8	Linkable to microswitch
15	SW7	Linkable to microswitch

16	SW1	Linked to microswitch
17	SW2	Linked to microswitch
18	SW3	Linkable to microswitch
19	SW4	Linkable to microswitch
20	GND	Linked to GND of +5V power supply
21	DIG.COM	Linked to DIG.COM in D/A (I/O)
22	A8	Linked to PA7 in I/O
23	A7	Linked to PA6 in I/O
24	A6	Linked to PA5 in I/O
25	A5	Linked to PA4 in I/O
26	A4	Linked to PA3 in I/O
27	A3	Linked to PA2 in I/O
28	A2	Linked to PA1 in I/O
29	A1	Linked to PA0 in I/O
30	X	
31	GND	Linked to L.L.GND in D/A
32	GND	GND for ±10 V power supply

Where

X -- not connected

APPENDIX D-3 Electronic connections for the PCB of the interfacing in HAND-II

Where

X -- not connected

D-3-1 Robotic hand connector

Pin No.	Signals	Descriptions
1	m_1^+	Linked to motor #1 terminal through current limiter
2	X	
3	m_2^-	To motor #2 terminal
4	SW2	From switch
5	SW4	From switch
6	$V_{\theta 1}$	From potentiometer #1
7	^2FSR	From FSR #2
8	X	
9	$^1OUT^+$	From Load cell #1
10	$^1OUT^-$	From Load cell #1

11	+5V	Power supply for potentiometers
12	GND	Ground for potentiometers
13	$-^{1}IN$	Power supply to load cell #1 (-7.5V)
14	m_2^+	Linked to motor #2 terminal through current limiter
15	m_1^-	Linked to motor #1 terminal
16	SW1	From microswitch
17	SW3	From microswitch
18	$+^{2}IN$	Power supply to load cell #2 (+7.5V)
19	$V_{\theta 2}$	From potentiometer #2
20	^{1}FSR	From FSR #1
21	$^{2}OUT^+$	From load cell #2
22	$^{2}OUT^-$	From load cell #2
23	$-^{2}IN$	Power supply to load cell #2 (-7.5V)
24	GND	Common ground
25	$-^{1}IN$	Power supply to load cell #1 (-7.5V)

D-3-2 Power supply connector

Pin No.	Signals	Descriptions
1	+10V	For motors
2	X	
3	+5V	For potentiometers and Op.Am
4	GND	Ground
5	-7.5V	For load cell
6	-10V	For motors
7	-5V	For Op.Am
8	GND	For microswitches
9	+7.5V	For load cell

D-3-3 D/A connector

Pin No.	Signals	Descriptions
1	A4	To PA4 of I/O
2	A2	To PA2 of I/O
3	DIG.COM	To PCB of microswitches
4	L.L.GND	To PCB of motor drive circuit
5	${}^{1}V_{in}$	Control volts from D/A CH4
6	A3	To PA3 of I/O
7	A1	To PA1 of I/O
8	L.L.GND	To PCB of motor drive circuit
9	${}^{2}V_{in}$	Control volts from D/A CH5

D-3-4 A/D connector

Pin No.	Signals	Descriptions
1	${}^{1}V_{FSR}$	Palm FSR linked to A/D CH4
2	${}^{2}V_{out}^{-}$	Force #2 GND to L.L.GND (CH3)
3	${}^{2}V_{out}^{+}$	Force #2 linked to A/D CH3
4	${}^{2}V_{\theta}$	Potentiometer #2 to A/D CH1
5	L.L.GND	To A/D L.L.GND
6	${}^{2}V_{FSR}$	FSR linked to A/D CH5 (Not used)
7	${}^{1}V_{out}^{+}$	Force #1 linked to A/D CH2
8	${}^{1}V_{out}^{-}$	Force #1 GND to L.L.GND (CH2)
9	${}^{1}V_{\theta}$	Potentiometer #1 to A/D CH0

D-3-5 Measurement connector

Pin No.	Signals	Descriptions
1	${}^{1}V_{\theta}$	Finger #1 position
2	X	
3	${}^{1}V_{out}$	Force on fingertip #1
4	${}^{2}V_{FSR}$	Force on FSR
5	GND	Ground
6	${}^{2}V_{\theta}$	Finger #2 position
7	${}^{2}V_{out}$	Force on fingertip #2
8	X	
9	${}^{1}V_{FSR}$	Palm force

APPENDIX E-1 Discretized universes of massaging speed, robot arm speed and force retention time.

E-1-1 Universe of massaging speed

Fuzzy terms	Discrete universe of massaging speed 0	1	2	3	4	5	6	7	8
LWE	1	0.5	0	0	0	0	0	0	0
LW	0	0.5	1	0.5	0	0	0	0	0
ME	0	0	0	0.5	1	0.5	0	0	0
HG	0	0	0	0	0	0.5	1	0.5	0
HGE	0	0	0	0	0	0	0	0.5	1

E-1-2 Universe of robotic arm speed

Fuzzy terms	Universe of robot arm speed 0	1	2	3	4	5	6	7	8
LWE	1	0.5	0	0	0	0	0	0	0
LW	0	0.5	1	0.5	0	0	0	0	0
ME	0	0	0	0.5	1	0.5	0	0	0
HG	0	0	0	0	0	0.5	1	0.5	0
HGE	0	0	0	0	0	0	0	0.5	1

E-1-3 Universe of force retention time

Fuzzy terms	Universe of force retention time								
	0	1	2	3	4	5	6	7	8
SHE	1	0.5	0	0	0	0	0	0	0
SH	0	0.5	1	0.5	0	0	0	0	0
ME	0	0	0	0.5	1	0.5	0	0	0
LN	0	0	0	0	0	0.5	1	0.5	0
LNE	0	0	0	0	0	0	0	0.5	1

APPENDIX E-2 Fuzzy relations for Rule base 2, Rule base 3 and Rule base 4.

E-2-1 Relations of arm speed in Base Rule 2

R_2		Arm speed universe								
		0	1	2	3	4	5	6	7	8
Universe of massaging speed	0	1	0.5	0	0	0	0	0	0	0
	1	0.5	0.5	0.5	0.5	0	0	0	0	0
	2	0	0.5	1	0.5	0	0	0	0	0
	3	0	0.5	0.5	0.5	0.5	0.5	0	0	0
	4	0	0	0	0.5	1	0.5	0	0	0
	5	0	0	0	0.5	0.5	0.5	0.5	0.5	0
	6	0	0	0	0	0	0.5	1	0.5	0
	7	0	0	0	0	0	0.5	0.5	0.5	0.5
	8	0	0	0	0	0	0	0	0.5	1

E-2-2 Relations of force retention time in Rule base 2

R_2		Force retention time universe								
		0	1	2	3	4	5	6	7	8
Universe of massaging speed	0	1	0.5	0	0	0	0	0	0	0
	1	0.5	0.5	0.5	0.5	0	0	0	0	0
	2	0	0.5	1	0.5	0	0	0	0	0
	3	0	0	0.5	0.5	0.5	0.5	0.5	0	0
	4	0	0	0	0.5	1	0.5	0	0	0
	5	0	0	0	0.5	0.5	0.5	0.5	0.5	0
	6	0	0	0	0	0	0.5	1	0.5	0
	7	0	0	0	0	0	0.5	0.5	0.5	0.5
	8	0	0	0	0	0	0	0	0.5	1

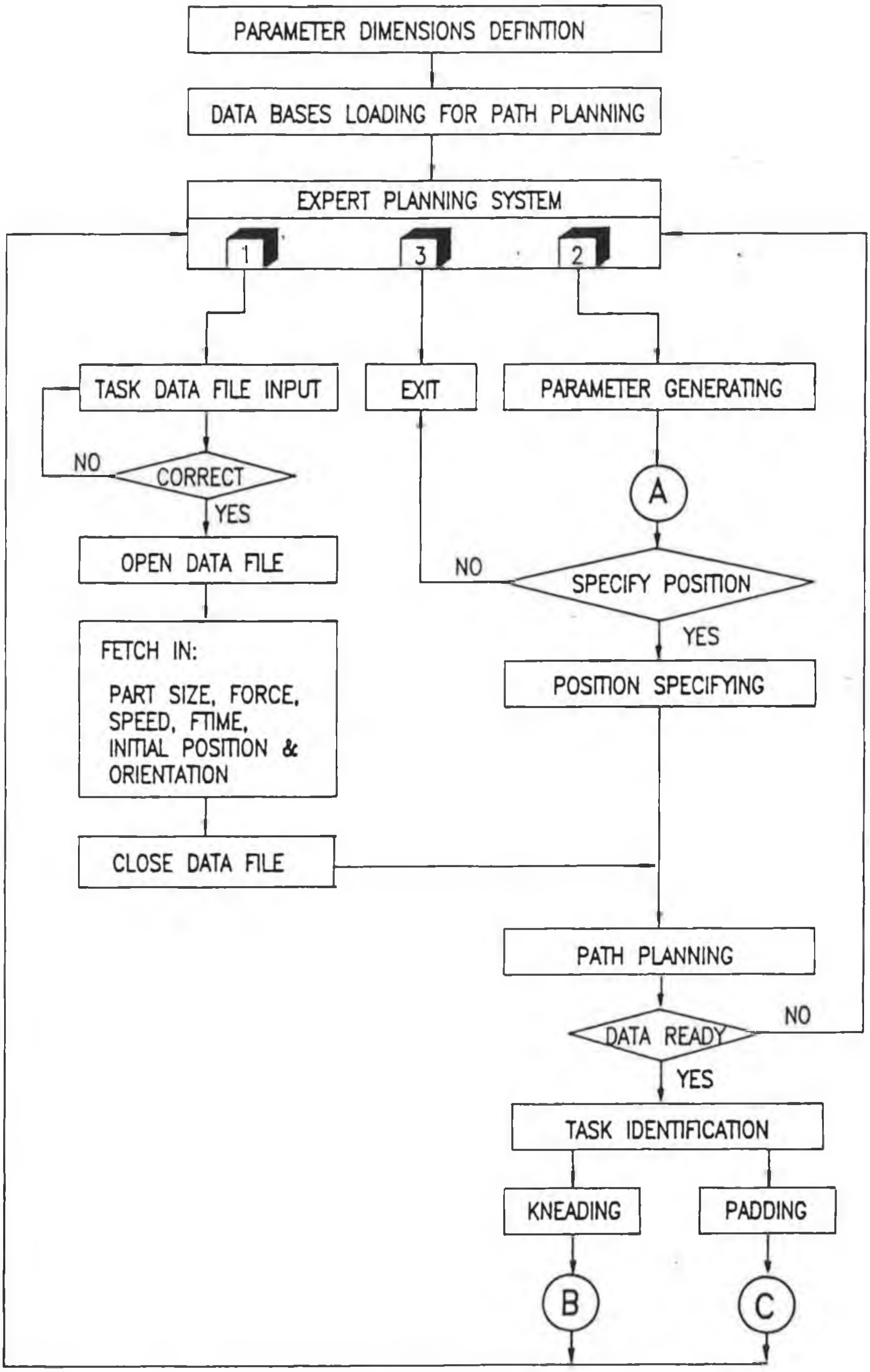
E-2-3 Relations of part size in Rule base 3

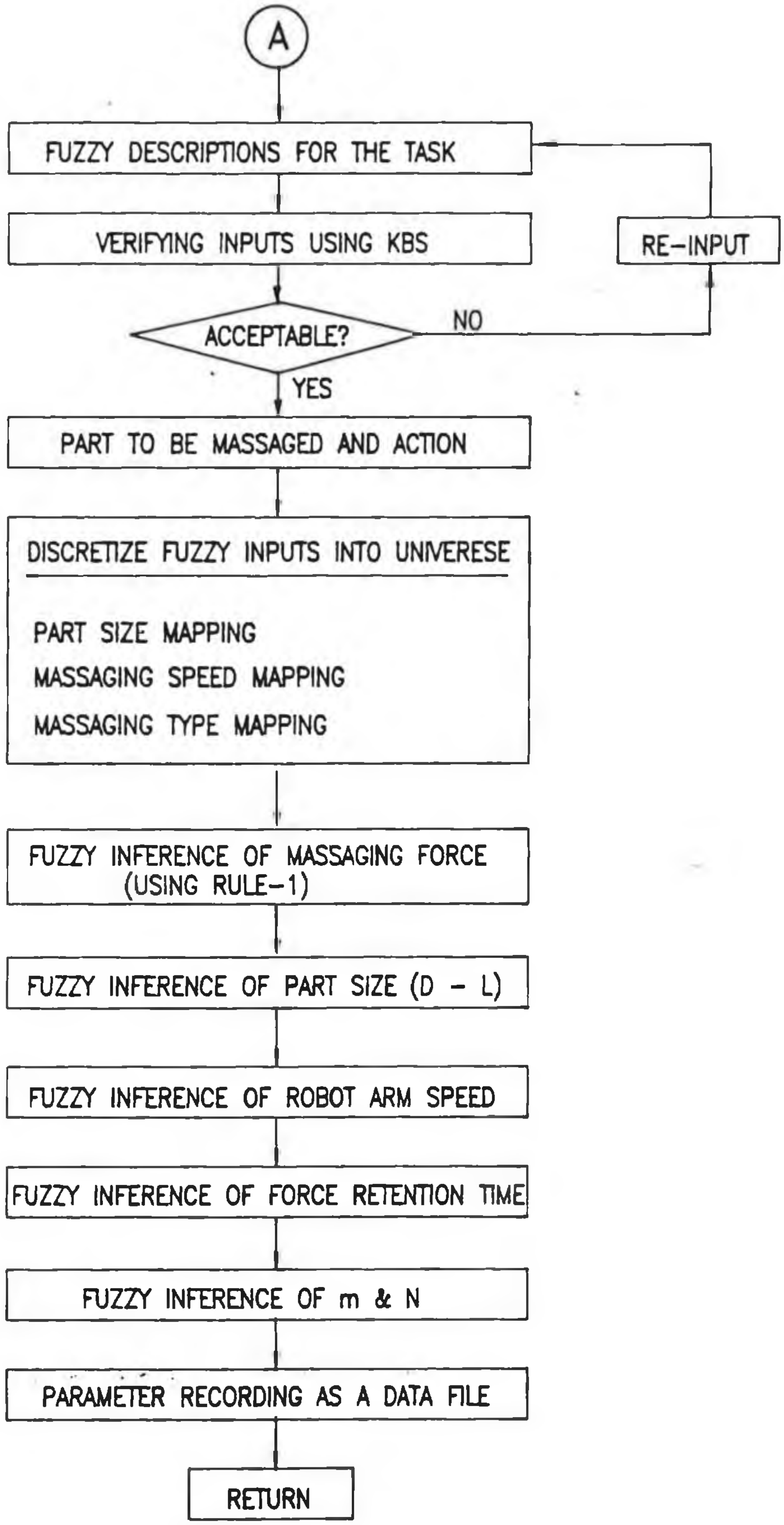
R_3		Part diameter & length universe								
		0	1	2	3	4	5	6	7	8
Part size universe	0	1	0.5	0	0	0	0	0	0	0
	1	0.5	0.5	0.5	0.5	0	0	0	0	0
	2	0	0.5	1	0.5	0	0	0	0	0
	3	0	0.5	0.5	0.5	0.5	0.5	0	0	0
	4	0	0	0	0.5	1	0.5	0	0	0
	5	0	0	0	0.5	0.5	0.5	0.5	0.5	0
	6	0	0	0	0	0	0.5	1	0.5	0
	7	0	0	0	0	0	0.5	0.5	0.5	0.5
	8	0	0	0	0	0	0	0	0.5	1

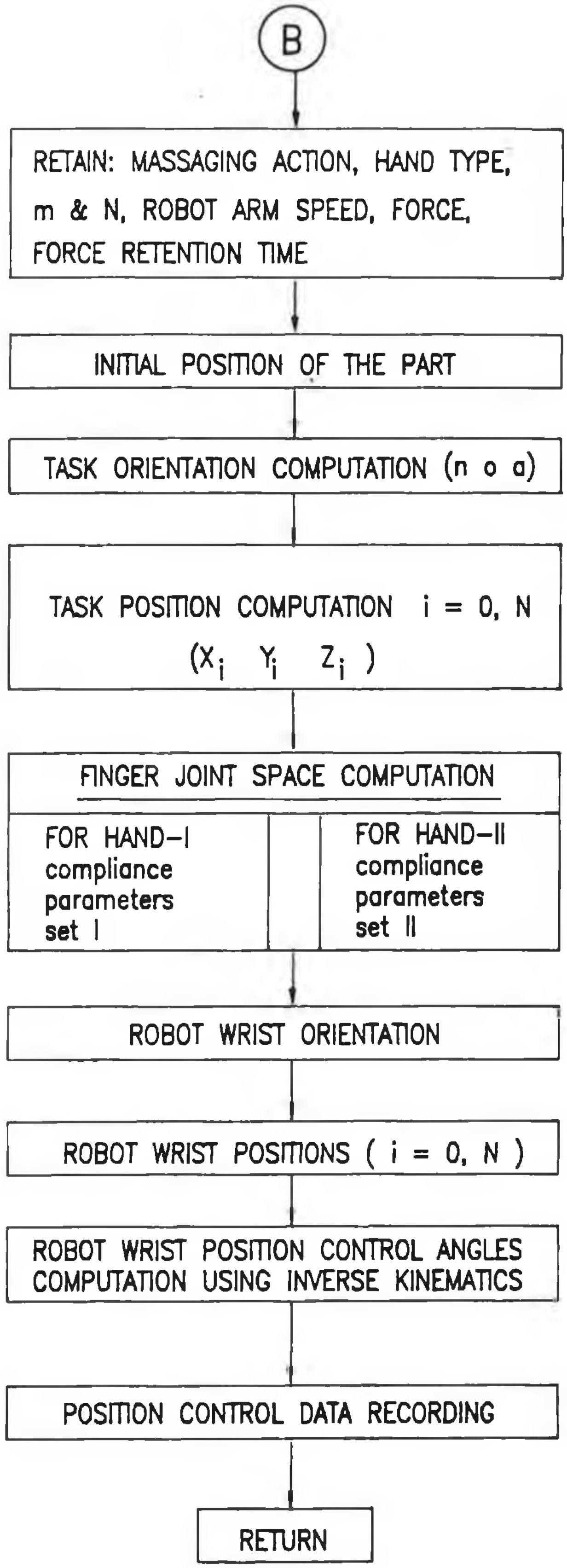
E-2-4 Relations of path number & massaging points in Rule base 4

R_4		Universe for path number & massaging points				
		0	1	2	3	4
Universe of massaging type	0	1	0.5	0	0	0
	1	0.5	0.5	0.5	0.5	0
	2	0	0.5	1	0.5	0
	3	0	0.5	0.5	0.5	0.5
	4	0	0	0	0.5	1

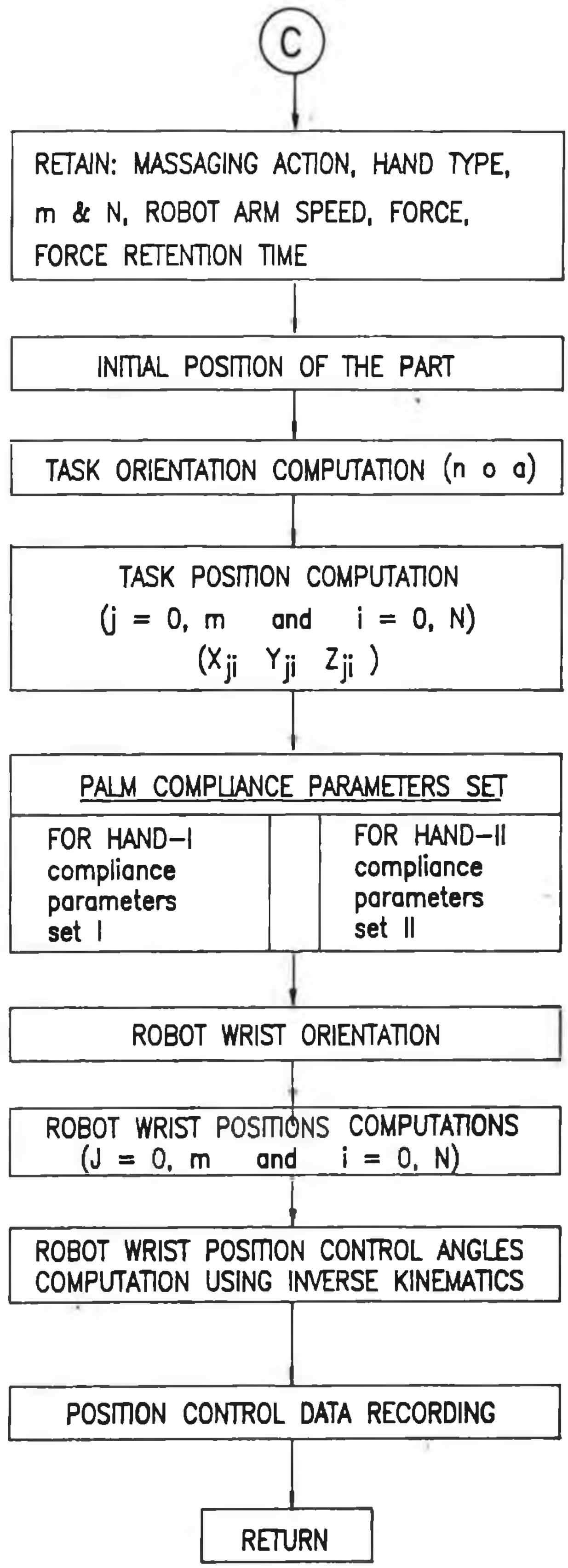
APPENDIX F-1 Parameter generating and path planning software -- EXPERTP.BAS




A
FUZZY DESCRIPTIONS FOR THE TASK
VERIFYING INPUTS USING KBS
RE-INPUT
ACCEPTABLE?
NO
YES
PART TO BE MASSAGED AND ACTION
DISCRETIZE FUZZY INPUTS INTO UNIVERESE
PART SIZE MAPPING
MASSAGING SPEED MAPPING
MASSAGING TYPE MAPPING
FUZZY INFERENCE OF MASSAGING FORCE (USING RULE-1)
FUZZY INFERENCE OF PART SIZE (D - L)
FUZZY INFERENCE OF ROBOT ARM SPEED
FUZZY INFERENCE OF FORCE RETENTION TIME
FUZZY INFERENCE OF m & N
PARAMETER RECORDING AS A DATA FILE
RETURN



B
RETAIN: MASSAGING ACTION, HAND TYPE, m & N, ROBOT ARM SPEED, FORCE, FORCE RETENTION TIME
INITIAL POSITION OF THE PART
TASK ORIENTATION COMPUTATION (n o a)
TASK POSITION COMPUTATION i = 0, N
(X_i Y_i Z_i)
FINGER JOINT SPACE COMPUTATION
FOR HAND-I compliance parameters set I
FOR HAND-II compliance parameters set II
ROBOT WRIST ORIENTATION
ROBOT WRIST POSITIONS (i = 0, N)
ROBOT WRIST POSITION CONTROL ANGLES COMPUTATION USING INVERSE KINEMATICS
POSITION CONTROL DATA RECORDING
RETURN


C
RETAIN: MASSAGING ACTION, HAND TYPE, m & N, ROBOT ARM SPEED, FORCE, FORCE RETENTION TIME
INITIAL POSITION OF THE PART
TASK ORIENTATION COMPUTATION (n o a)
TASK POSITION COMPUTATION
(j = 0, m and i = 0, N)
(X_{ji} Y_{ji} Z_{ji})
PALM COMPLIANCE PARAMETERS SET
FOR HAND-I compliance parameters set I
FOR HAND-II compliance parameters set II
ROBOT WRIST ORIENTATION
ROBOT WRIST POSITIONS COMPUTATIONS
(J = 0, m and i = 0, N)
ROBOT WRIST POSITION CONTROL ANGLES COMPUTATION USING INVERSE KINEMATICS
POSITION CONTROL DATA RECORDING
RETURN

```
1400    ' ***************************************************
1410    ' *                                                 *
1420    ' *      EXPERT SYSTEM FOR PHYSIOTHERAPIC ROBOT     *
1430    ' *                                                 *
1435    ' *       PATH PLANNING & PARAMETER ORGANIZING      *
1440    ' *                                                 *
1450    ' *       %%  FOR KNEAD AND PAD OPERATIONS  %%      *
1460    ' *                                                 *
1465    ' *       a. TASK DESCRIPTIONS                      *
1470    ' *       b. PATH PLANNING & ORGANIZING MODULE      *
1475    ' *       c. OFF-LINE KBS FOR PLANNING MODULE       *
1480    ' *       d. FUZZY LOGIC FOR PARAMETER GENERATING   *
1485    ' *                                                 *
1490    ' *-------------------------------------------------*
1495    ' *                                                 *
1500    ' *            FILE NAME  -->  EXPERTP.BAS          *
1505    ' *                                                 *
1510    ' *                EDITED BY J. YAN                 *
1515    ' *                                                 *
1520    ' *            DUBLIN CITY UNIVERSITY               *
1525    ' *                                                 *
1530    ' ***************************************************
1540    '
1600    '
1610    '  %*******  DIMENSION SECTION  ********%
1620    '
1630    '  **  Comman buffer  **
1640    '
1650    '  OO(3,3)      -- Robot arm orientation
1660    '  PP(3)        -- Robot arm position
1665    '  QQ(6)        -- Robot arm joint angles
1670    '
1680    DIM OO(3,3),PP(3),QQ(6)
1685    '
1690    '
1695    '  **  Robot finger space  **
1700    '
1710    '  QF(30)       -- Finger openning angles
1715    '  CPZ(30)      -- Compliance grasping distance
1720    '
1730    DIM QF(30),CPZ(30)
1735    '
1740    '
1745    '  **  Kneading space  **
1750    '
1755    '  XKT(30)      -- Task position along X axis
1760    '  YKT(30)      -- Task position along Y axis
1762    '  ZKT(30)      -- Task position along Z axis
1765    '  XKP(30)      -- Robot arm X control position
1767    '  YKP(30)      -- Robot arm Y control position
1770    '  ZKP(30)      -- Robot arm Z control position
1775    '  QKP(30)      -- Robot pitch control angle
1780    '  QKR(30)      -- Robot  roll control angle
1782    '
```

```
1785    DIM XKT(30),YKT(30),ZKT(30)
1790    DIM XKP(30),YKP(30),ZKP(30),QKP(30),QKR(30)
1792    '
1800    '  **  Padding space  **
1810    '
1815    '  XPT(10,30)   -- Task position along X axis
1820    '  YPT(10,30)   -- Task position along Y axis
1825    '  ZPT(10,30)   -- Task position along Z axis
1830    '  XPP(10,30)   -- Robot arm X control position
1832    '  YPP(10,30)   -- Robot arm Y control position
1835    '  ZPP(10,30)   -- Robot arm Z control position
1840    '  QPP(10,30)   -- Robot pitch control angle
1845    '  QPR(10,30)   -- Robot  roll control angle
1850    '
1855    DIM XPT(10,30),YPT(10,30),ZPT(10,30)
1860    DIM XPP(10,30),YPP(10,30),ZPP(10,30)
1865    DIM QPP(10,30),QPR(10,30)
1870    '
1900    '  ** Fuzzy inference process **
1905    '
1910    '  FM(9,9)      -- Fuzzy membership
1915    '  PSIZE(6,9)   -- Part size for length & diameter
1920    '  FORCE(3,9)   -- Massaging force
1924    '  SPDA(9)      -- Arm speed
1928    '  TFR(9)       -- Force retention time
1930    '  PTYPE(2,5)   -- Path number & point number
1934    '
1938    DIM FM(9,9),PSIZE(6,9),FORCE(3,9)
1940    DIM SPDA(9),TFR(9),PTYPE(2,5)
1946    '
1990    '
2000    ' ***********************************************
2010    ' *                                             *
2020    ' *     DATA BASE LOADING FOR PATH PLANNING     *
2030    ' *                                             *
2040    ' *              * PART SIZE                    *
2050    ' *              * MASSAGING FORCE              *
2060    ' *              * ROBOT ARM SPEED              *
2070    ' *              * MASSAGING TYPE               *
2080    ' *              * FUZZY MEMBERSHIP             *
2090    ' *                                             *
2100    ' ***********************************************
2120    '
2130    GOSUB 30000
2140    '
2160    '
3000    ' ***********************************************
3010    ' *                                             *
3020    ' *      EXPERT PLANNING SYSTEM MAIN MENU       *
3030    ' *                                             *
3040    ' *     * DATA FILE INPUT & PATH PLANNING       *
3050    ' *     * PARAMETER GENERATING & PATH PLANNING  *
3070    ' *     * RETURN TO DOS                         *
3080    ' *                                             *
3090    ' ***********************************************
```

```
3100    '
3105    '
3110    COLOR 7,1:CLS
3120    LOCATE 4,20:COLOR 2,4
3130    PRINT"** MAIN MENU FOR EXPERT PLANNING SYSTEM **"
3135    '
3140    '
3145    COLOR 1,7
3150    LOCATE 7,20
3155    PRINT"<1> - TASKS DATA FILE INPUT & PATH PLANNING"
3160    LOCATE 8,20
3165    PRINT"<2> - PARAMETERS GENERATING & PATH PLANNING"
3170    LOCATE 9,20
3175    PRINT"<3> - RETURN TO DOS      "
3180    '
3190    COLOR 4,2
3200    LOCATE 11,20
3210    INPUT"Please input your choice [1-3] ";MAIN£
3220    '
3230    IF VAL(MAIN£)=1 THEN 4000         ' TASK DATA FILE
3240    IF VAL(MAIN£)=2 THEN 6000         ' MAN-MACHINE
3250    IF VAL(MAIN£)=3 THEN 3500         ' RETURN TO DOS
3260    GOTO 3200
3270    '
3280    '
3300    '
3500    ' !!!!! RETURN TO DOS !!!!!
3510    '
3520    GOSUB 9000                ' PROMPT BOX FRAME
3530    LOCATE 21,24:COLOR 2,4
3540    PRINT".. QUIT FROM EXPERT PLANNING SYSTEM .."
3550    LOCATE 24,1
3560    END
3600    '
3610    '
3620    '
4000    ' ***********************************************
4005    ' *                                             *
4010    ' *   PATH PLANNING BASED ON TASKS DATA INPUT   *
4015    ' *                                             *
4020    ' *          *  TASK DATA FILE INPUT            *
4024    ' *          *  PATH PLANNING                   *
4028    ' *          *  PATH DATA SAVING                *
4030    ' *                                             *
4035    ' ***********************************************
4040    '
4050    '
4060    COLOR 7,1:CLS
4062    LOCATE 4,20:COLOR 4,2
4066    PRINT"TASK DATA FILE INPUT FOR PATH PLANNING"
4070    LOCATE 8,20:COLOR 2,4
4080    INPUT"PLEASE INPUT THE TASKS DATA FILE .. ";DTASK£
4100    '
4110    LOCATE 10,20:COLOR 7,1
4120    PRINT"IS ";
```

```
4130     COLOR 4,2:PRINT DTASK£;
4140     COLOR 7,1:PRINT " THE RIGHT TASKS DATA FILE (Y/N)?"
4150     '
4160     A£=INKEY£
4170     IF A£="Y" OR A£="y" THEN 4200
4180     IF A£="N" OR A£="n" THEN 4060
4190     GOTO 4160
4195     '
4200     '
4205     GOSUB 20000      ' DATA FETCH
4210     '
4220     GOSUB 9000       ' PROMPT
4230     LOCATE 21,26:COLOR 4,2
4240     PRINT"..TASK DATA FILE HAS BEEN LOADED.."
4250     LOCATE 10,20:COLOR 20,2
4260     PRINT".PRESS ANY KEY TO  START  PATH PLANNING."
4270     '
4280     IF INKEY£="" THEN 4280
4300     '
4330     '
4340     GOSUB 14500      ' PATH PLANNING & RECORDING
4350     '
4360     GOSUB 9000
4370     LOCATE 21,26:COLOR 4,2
4380     PRINT".. PATH HAS BEEN PLANNED & STORED .."
4390     GOTO 3120
4400     '
4500     '
6000     ' ***********************************************
6005     ' *                                             *
6010     ' *    PARAMETER GENERATING & PATH PLANNING     *
6020     ' *                                             *
6030     ' *     * FUZZY INPUTS & FUZZY INFERENCE        *
6040     ' *     * PART LOCATION SPECIFYING              *
6050     ' *     * PATH PLANNING                         *
6060     ' *     * PATH DATA SAVING                      *
6070     ' *                                             *
6080     ' ***********************************************
6090     '
6095     '
6100     GOSUB 10000
6110     '
6120     GOSUB 9000
6130     LOCATE 21,26:COLOR 4,2
6140     PRINT".. PATH HAS BEEN PLANNED & STORED .."
6150     GOTO 3120
6160     '
6300     '
9000     ' !!!!!--  PROMPT BOX  --!!!!! (SUBROUTINE)
9010     '
9020     COLOR 7,1:CLS
9030     LOCATE 20,15:COLOR 1,7
9035     PRINT"**********!!!!!      ";
9040     COLOR 20,7:PRINT"PROMPT BOX";
9045     COLOR 1,7:PRINT"      !!!!!***********"
```

```
9100    LOCATE 22,15:COLOR 1,7
9110    PRINT"*********************************";
9115    PRINT"********************"
9120    COLOR 7,1
9150    RETURN
9400    '
9500    '
10000   ' **********************************************
10001   ' *                                            *
10002   ' *      TASK DESCRIPTION & PATH PLANNING      *
10004   ' *                                            *
10006   ' **********************************************
10008   '
10010   '
10012   ' aa. TASK DESCRIPTION <1> -- fuzzy variables  .aa
10014   '
10015   '|-------------   Fuzzy variables used  ----------|
10016   '| The following fuzzy variables have been used to
10017   '| describe the task and environment:
10018   '| SME -- (smaller)  SM  -- (small)  ME -- (medium)|
10019   '| LG  -- (large)    LGE -- (larger)
10020   '| LNE -- (longer)   LN  -- (long)   SH -- (short)
10022   '| SHE -- (shorter)  LWE -- (lower)  LW -- (low)
10023   '| FIN -- (fine)     HGE -- (higher) HG -- (high)
10024   '| CRS -- (coarse)   STD -- (standard)
10025   '|------------------------------------------------|
10026   '
10030   COLOR 7,1:CLS
10032   LOCATE 4,15:COLOR 4,2
10035   PRINT"** TASK DESCRIPTIONS USING FUZZY CONCEPTS **"
10038   '
10040   COLOR 1,7
10045   LOCATE 6,20
10050   INPUT"PART TO BE MASSAGED (arm, neck, back) ";PART£
10055   LOCATE 7,20
10060   INPUT"THE PART SIZE  (SME, SM, ME, LG, LGE) ";SIZE£
10065   LOCATE 8,20
10070   INPUT"THE MASSAGING TYPE  ( CRS, STD, FIN ) ";TYPE£
10072   LOCATE 9,20
10076   INPUT"THE MASSAGING SPEED (LWE,LW,ME,HG,HGE)";SPDM£
10078   '
10080   LOCATE 11,20
10082   INPUT"ROBOT HAND USED ( HANDOLD or HANDNEW )";HAND£
10084   '
10086   LOCATE 13,15
10090   PRINT"THE ";
10092   COLOR 2,4:PRINT PART£;
10094   COLOR 1,7:PRINT" WITH ";
10096   COLOR 2,4:PRINT SIZE£;
10100   COLOR 1,7
10101   PRINT" SIZE HAS BEEN SPECIFIED TO BE MASSAGED "
10102   LOCATE 14,15
10105   PRINT"IN ";
10108   COLOR 2,4:PRINT SPDM£;
10110   COLOR 1,7
```

```
10112    PRINT" SPEED! AND THE MASSAGING TYPE IS ";
10116    COLOR 2,4:PRINT TYPE£
10118    '
10120    LOCATE 16,20:COLOR 2,4
10125    PRINT"ARE THE TASK DESCRIPTIONS RIGHT (Y/N) ?"
10130    A£=INKEY£
10135    IF A£="Y" OR A£="y" THEN 10200
10140    IF A£="N" OR A£="n" THEN 10030
10150    GOTO 10120
10160    '
10165    '
10170    ' bb. PARAMETER CREATING 1. -- for fuzzy inputs .bb
10175    '
10180    ' |---------------------------------------------------|
10182    ' |  The fuzzy decription and parameter generating    |
10184    ' |  KBs have been constructed here to assist:        |
10186    ' |      * verify the fuzzy inputs                    |
10188    ' |      * generate the task parameters               |
10190    ' |-------------------- < 1 > ------------------------|
10194    '
10200    COLOR 7,1:CLS
10210    LOCATE 4,20:COLOR 2,4
10220    PRINT"Task parameters are inferred as follows .."
10240    '
10245    '
10248    ' bb * STATEMENT * DIMENSION STATEMENT
10250    '
10251    ' |----------  DIMENSION STATEMENT  -----------------|
10252    ' |  FM(i,j)     - fuzzy membership                  |
10253    ' |  PSIZE(i,j)  - part size for length & diameter   |
10254    ' |                i = 0-1 for  ARM SIZE             |
10256    ' |                i = 2-3 for NECK SIZE             |
10257    ' |                i = 4-5 for BACK SIZE             |
10258    ' |  FORCE(i,j)  - massaging force                   |
10259    ' |                i = 0  for  ARM force             |
10260    ' |                i = 1  for NECK force             |
10262    ' |                i = 2  for BACK force             |
10265    ' |  SPDA(i)     - arm speed                         |
10267    ' |  TFR(i)      - force retention time (s)          |
10268    ' |  PTYPE(i,j)  - path number & point number        |
10269    ' |                i = 0  for massaging points       |
10270    ' |                i = 1  for massaging paths        |
10272    ' |--------------------------------------------------|
10275    '
10300    ' bb * KB-1 * DECIDE THE MASSAGING ACTION
10302    '
10304    ' ACT£ -- massaging action
10306    ' PART -- part number for control
10307    '
10310    IF PART£="ARM"  OR PART£="arm"  THEN 10370
10320    IF PART£="NECK" OR PART£="neck" THEN 10375
10330    IF PART£="BACK" OR PART£="back" THEN 10380
10332    '
10335    ' GO BACK TO FUZZY INPUT IF NO MATCHING
10338    '
```

```
10340   GOSUB 9000        ' PROMPT BOX
10345   LOCATE 21,18:COLOR 2,4
10350   PRINT"THE INPUT IS NOT CORRECT FOR THE ";
10355   PRINT"PART TO BE MASSAGED"
10360   '
10365   GOTO 10032        ' RE-INPUT
10368   '
10370   ACT£="KNEAD":PART=0:GOTO 11000   ' ARM
10375   ACT£="KNEAD":PART=1:GOTO 11000   ' NECK
10380   ACT£="PAD":  PART=2:GOTO 11000   ' BACK
10500   '
11000   ' bb * KB-2 *  FUZZIFY INPUTS INTO THE UNIVERSE
11005   '
11006   ' ** PART SIZE MAPPING
11008   '
11009   ' SIZE£ -- part fuzzy input
11010   ' SIZE  -- universe number of the part input
11015   '
11020   IF SIZE£="SME"    OR SIZE£="sme"    THEN 11070
11025   IF SIZE£="SME-SM" OR SIZE£="sme-sm" THEN 11072
11030   IF SIZE£="SM"     OR SIZE£="sm"     THEN 11074
11032   IF SIZE£="SM-ME"  OR SIZE£="sm-me"  THEN 11076
11035   IF SIZE£="ME"     OR SIZE£="me"     THEN 11078
11037   IF SIZE£="ME-LG"  OR SIZE£="me-lg"  THEN 11080
11040   IF SIZE£="LG"     OR SIZE£="lg"     THEN 11082
11042   IF SIZE£="LG-LGE" OR SIZE£="lg-lge" THEN 11084
11045   IF SIZE£="LGE"    OR SIZE£="lge"    THEN 11086
11050   '
11052   ' GO BACK TO FUZZY INPUT IF NO MATCHING
11055   '
11057   GOSUB 9000
11060   LOCATE 21,25:COLOR 2,4
11062   PRINT".. PART SIZE INPUT IS NOT CORRET .."
11065   GOTO 10032        ' RE-INPUT
11068   '
11070   SIZE=0:GOTO 11090
11072   SIZE=1:GOTO 11090
11074   SIZE=2:GOTO 11090
11076   SIZE=3:GOTO 11090
11078   SIZE=4:GOTO 11090
11080   SIZE=5:GOTO 11090
11082   SIZE=6:GOTO 11090
11084   SIZE=7:GOTO 11090
11086   SIZE=8:GOTO 11090
11088   '
11090   '
11100   ' ** MASSAGING SPEED MAPPING
11105   '
11110   ' SPDM£ -- massaging speed fuzzy input
11112   ' SPDM -- universe number of massaging speed input
11115   '
11120   IF SPDM£="LWE"    OR SPDM£="lwe"    THEN 11180
11125   IF SPDM£="LWE-LW" OR SPDM£="lwe-lw" THEN 11182
11130   IF SPDM£="LW"     OR SPDM£="lw"     THEN 11184
11135   IF SPDM£="LW-ME"  OR SPDM£="lw-me"  THEN 11186
```

```
11140   IF SPDM£="ME"      OR SPDM£="me"      THEN 11188
11145   IF SPDM£="ME-HG"   OR SPDM£="me-hg"   THEN 11190
11150   IF SPDM£="HG"      OR SPDM£="hg"      THEN 11192
11155   IF SPDM£="HG-HGE"  OR SPDM£="hg-hge"  THEN 11194
11160   IF SPDM£="HGE"     OR SPDM£="hge"     THEN 11196
11162   '
11165   ' GO BACK TO FUZZY INPUT IF NO MATCHING
11168   '
11170   GOSUB 9000
11172   LOCATE 21,27:COLOR 2,4
11174   PRINT".. SPEED INPUT IS NOT CORRECT .."
11176   GOTO 10032       ' RE-INPUT
11178   '
11180   SPDM=0:GOTO 11200
11182   SPDM=1:GOTO 11200
11184   SPDM=2:GOTO 11200
11186   SPDM=3:GOTO 11200
11188   SPDM=4:GOTO 11200
11190   SPDM=5:GOTO 11200
11192   SPDM=6:GOTO 11200
11194   SPDM=7:GOTO 11200
11196   SPDM=8:GOTO 11200
11198   '
11199   '
11200   ' ** MASSAGING TYPE MAPPING
11202   '
11204   ' TYPE£ -- massaging type fuzzy input
11206   ' TYPE  -- universe number of the massaging type
11208   '
11210   IF TYPE£="CRS"     OR TYPE£="crs"      THEN 11260
11215   IF TYPE£="CRS-STD" OR TYPE£="crs-std"  THEN 11262
11220   IF TYPE£="STD"     OR TYPE£="std"      THEN 11264
11225   IF TYPE£="STD-FIN" OR TYPE£="std-fin"  THEN 11266
11230   IF TYPE£="FIN"     OR TYPE£="fin"      THEN 11268
11235   '
11240   ' GO BACK TO FUZZY INPUT IF NO MATCHING
11245   '
11250   GOSUB 9000
11252   LOCATE 21,23:COLOR 2,4
11254   PRINT".. MASSAGING TYPE INPUT IS NOT CORRECT .."
11256   GOTO 10032       ' RE-INPUT
11258   '
11260   TYPE=0:GOTO 11300
11262   TYPE=1:GOTO 11300
11264   TYPE=2:GOTO 11300
11266   TYPE=3:GOTO 11300
11268   TYPE=4:GOTO 11300
11270   '
11300   ' bb * KB-3 * FUZZY INFERENCE PROCESS
11304   '
11310   ' ** INFER MASSAGING FORCE USING RULE-1
11312   '
11315   ' FM( SIZE,j )      --  fuzzy membership
11320   ' FORCE( PART,j )   --  force data
11322   ' FFORCE            --  inferred force
```

```
11410    FUZZYM=0
11420    FUZZYF=0
11430    FOR J=0 TO 8
11435    FUZZYM=FUZZYM+FM(SIZE,J)    ' SUM OF MEMBERSHIP
11440    FUZZYF=FUZZYF+FM(SIZE,J)*FORCE(PART,J)
11450    NEXT J
11455    '
11460    FFORCE=FUZZYF/FUZZYM
11465    '
11470    '
11500    ' ** INFER THE PART SIZE ( D and L )
11502    '
11510    ' FM( SIZE, j)        -- fuzzy membership
11512    ' PSIZE(PART*2, j)    -- data of part diameter
11514    ' PSIZE(PART*2+1,j)   -- data of part length
11516    ' DPART               -- inferred part diameter
11518    ' LPART               -- inferred part length
11519    '(for BACK, diameter is width & length is height)
11520    '
11530    '
11600    '
11610    FUZZYM=0
11615    FUZZYD=0
11620    FUZZYL=0
11625    '
11630    FOR J=0 TO 8
11635    FUZZYM=FUZZYM+FM(SIZE,J)    ' SUM OF MEMBERSHIP
11640    FUZZYD=FUZZYD+FM(SIZE,J)*PSIZE(PART*2,J)
11645    FUZZYL=FUZZYL+FM(SIZE,J)*PSIZE(PART*2+1,J)
11650    NEXT J
11655    '
11660    DPART=FUZZYD/FUZZYM       ' PART DIAMETER
11665    LPART=FUZZYL/FUZZYM       ' PART LENGTH
11670    '
11680    '
11685    ' ** INFER THE ROBOT ARM SPEED
11688    '
11690    ' PM(SPDM,j)    -- fuzzy membership
11692    ' SPDA(j)       -- data of robot arm speed
11695    ' SPEEDA        -- inferred robot arm speed
11700    '
11710    '
11780    FUZZYM=0
11782    FUZZYS=0
11785    FOR J=0 TO 8
11790    FUZZYM=FUZZYM+FM(SPDM,J)
11792    FUZZYS=FUZZYS+FM(SPDM,J)*SPDA(J)
11794    NEXT J
11795    '
11796    SPEEDA=FUZZYS/FUZZYM      ' Arm speed
11798    SPEEDA=CINT(SPEEDA)
11799    '
11800    '
11810    ' ** INFER THE FORCE RETENTION TIME
11812    '
```

```
11815   ' FM(SPDM,j)     -- fuzzy membership
11817   ' FTR(j)         -- data of force retention time
11819   ' FTIME          -- inferred force retention time
11820   '
11830   '
11890   FUZZYM=0
11900   FUZZYT=0
11910   FOR J=0 TO 8
11912   FUZZYM=FUZZYM+FM(SPDM,J)    ' SUM OF MEMBERSHIP
11920   FUZZYT=FUZZYT+FM(SPDM,J)*TFR(J)
11925   NEXT J
11930   '
11935   FTIME=FUZZYT/FUZZYM       ' Force retention time
11938   '
11940   '
11950   ' ** INFER THE MASSAGING PATH & POINT
11952   '
11954   ' FM(TYPE,j)     -- fuzzy membership
11956   ' PTYPE(0,j)     -- data of massaging point (m)
11958   ' PTYPE(1,j)     -- data of massaging path  (N)
11960   ' PMM            -- inferred massaging point
11962   ' PNN            -- inferred massaging path
11980   '
12000   '
12010   FUZZYM=0
12020   FUZZY0=0
12030   FUZZY1=0
12040   FOR J=0 TO 4
12050   FUZZYM=FUZZYM+FM(TYPE,J)     ' SUM OF MEMBERSHIP
12060   FUZZY0=FUZZY0+FM(TYPE,J)*PTYPE(0,J)
12070   FUZZY1=FUZZY1+FM(TYPE,J)*PTYPE(1,J)
12100   NEXT J
12110   '
12120   PMM=FUZZY0/FUZZYM    ' NUMBER OF MASSAGING POINT
12125   PNN=FUZZY1/FUZZYM    ' NUMBER OF MASSAGING PATH
12130   PMM=CINT(PMM)+1
12135   PNN=CINT(PNN)+1
12240   '
12300   '
12310   '    **** BREAK POINT FOR CHECKING ****
12312   '
12314   COLOR 4,2
12316   LOCATE 8,25
12318   PRINT"   ACT = ";ACT£
12320   LOCATE 9,25
12322   PRINT"  HAND = ";HAND£
12326   LOCATE 10,25
12330   PRINT"   PNN = ";USING"####";PNN
12334   LOCATE 11,25
12338   PRINT"   PMM = ";USING"####";PMM
12340   LOCATE 12,25
12342   PRINT" SPEED = ";USING"####";SPEEDA
12344   LOCATE 13,25
12346   PRINT" FTIME = ";USING"+###.##";FTIME
12350   LOCATE 14,25
```

```
12352    PRINT" FORCE = ";USING"+###.##";FFORCE
12354    LOCATE 15,25
12358    PRINT" DPART = ";USING"+###.##";DPART
12360    LOCATE 16,25
12364    PRINT" LPART = ";USING"+###.##";LPART
12370    '
12380    '
12400    LOCATE 18,20:COLOR 2,4
12410    PRINT"DO YOU WANT TO SAVE THE ABOVE DATA (Y/N) ? "
12420    A£=INKEY£
12430    IF A£="Y" OR A£="y" THEN 12500
12440    IF A£="N" OR A£="n" THEN 12550
12450    GOTO 12420
12470    '
12500    GOSUB 25000      ' DATA RECORDING
12520    '
12530    '
12540    LOCATE 20,20:COLOR 20,2
12550    PRINT"Press any key to continue path planning"
12560    IF INKEY£="" THEN 12560
12570    '
12580    '
13000    '
13010    ' cc. PARAMETER CREATING 2. -- for part location .cc
13015    '
13020    ' |-----------------------------------------------|
13025    ' | The location of the part can be generated in  |
13026    ' | two ways:                                     |
13028    ' |   a. use the defined locations in KB, such as |
13032    ' |      Locp i ( the ith parallel location)      |
13036    ' |      Locv i ( the ith vertical position)      |
13040    ' |   b. specify the locations by users, such as  |
13048    ' |      (XXX, YYY, ZZZ) -- the initial position  |
13050    ' |                 of the part to be massaged    |
13060    ' |      (ALF, BTA)      -- the direction of the  |
13065    ' |                 part in the robotic space     |
13068    ' |-----------------------------------------------|
13070    ' |                                               |
13072    ' | The location specified can be understood as:  |
13074    ' |                                               |
13076    ' |   a. For arm & neck, the location refers to   |
13078    ' |      the center line of the part              |
13080    ' |   b. For back, the location refers to the     |
13082    ' |      center line of the back surface along    |
13084    ' |      the length direction. ALF angle refers   |
13086    ' |      to the angle between the plane and the   |
13088    ' |      XOY plane                                |
13090    ' |                                               |
13092    ' |-----------------  < 2 >  ---------------------|
13100    '
13110    '
13160    ' cc * Path location specifying * cc
13165    '
13170    '
```

```
13190    COLOR 7,1:CLS
13200    LOCATE 5,15:COLOR 2,4
13220    PRINT"*** Part LOCATION specify & input menu ***"
13222    '
13230    COLOR 1,7
13235    LOCATE 8,18
13240    PRINT"< 1 > - KB  assists to generate part location"
13250    LOCATE 9,18
13255    PRINT"        (using the defined positions in KB) "
13262    LOCATE 11,18
13265    PRINT"< 2 > - User assists to specify part location"
13270    LOCATE 12,18
13275    PRINT"        (specifying the positions directly) "
13280    '
13285    COLOR 4,2
13290    LOCATE 14,15
13295    INPUT"Which way to specify part position ";WAY£
13298    IF VAL(WAY£)=1 THEN 13500         ' KB  assistance
13300    IF VAL(WAY£)=2 THEN 14000         ' USER specify
13310    GOTO 13290
13320    '
13400    '
13500    ' cc * KB HELP * USING KB TO GENERATE PART LOCATION
13510    '
13515    COLOR 7,1:CLS
13520    LOCATE 2,15:COLOR 4,2
13525    PRINT"Input part LOCATION using the data in KB";
13530    '
13535    LOCATE 4,18:COLOR 1,7
13540    PRINT"Positions for the part parallel to XOY plane"
13550    LOCATE 5,23:COLOR 2,4
13555    PRINT"( LOCP1,LOCP2,LOCP3,LOCP4,LOCP5 )
13560    '
13565    LOCATE 7,18:COLOR 1,7
13570    PRINT"Positions for the part vertical to XOY plane"
13575    LOCATE 8,23:COLOR 2,4
13580    PRINT"( LOCV1,LOCV2,LOCV3,LOCV4,LOCV5 )
13585    '
13590    LOCATE 10,15:COLOR 1,7
13592    INPUT"Input your choice (LOCPi or LOCVi) ";LYC£
13595    '
13600    '
13620    ' |----------- PARAMETERS DEFINED IN KB --------|
13625    ' |  XXX  --  position along X axis             |
13630    ' |  YYY  --  position along Y axis             |
13635    ' |  ZZZ  --  position along Z axis             |
13640    ' |  ALF  --  angle of part with respect of XOY |
13650    ' |  BTA  --  angle of part with respect of XOZ |
13660    ' |---------------------------------------------|
13670    '
13675    '
13680    '
13700    GOSUB 19000
13705    '
13710    IF REINPUT=0 THEN 13800
```

```
13715   '
13720   GOSUB 9000
13725   LOCATE 21,23:COLOR 4,2
13730   PRINT".. PART LOCATION INPUT IS INCORRECT .."
13735   '
13740   GOTO 13200
13745   '
13750   '
13760   '
13800   XXX=X
13810   YYY=Y
13820   ZZZ=Z
13830   ALF=AF
13840   BTA=BA
13850   '
13860   GOTO 14500        ' GO TO PATH PLANNING
13870   '
13880   '
13900   '
14000   '  cc * USER ASSIST * USER SPECIFY PART LOCATION
14005   '
14010   COLOR 7,1:CLS
14015   LOCATE 2,20:COLOR 4,2
14020   PRINT".. USER SPECIFYING THE PART LOCATION .."
14025   '
14030   LOCATE 5,15:COLOR 4,2
14035   PRINT"PLEASE INPUT THE FOLLOWING PARAMETERS..."
14040   '
14045   COLOR 1,7
14050   LOCATE 7,20
14055   INPUT"INITIAL POSITION ALONG X AXIS (mm) ";XXX
14060   LOCATE 8,20
14065   INPUT"INITIAL POSITION ALONG Y AXIS (mm) ";YYY
14070   LOCATE 9,20
14075   INPUT"INITIAL POSITION ALONG Z AXIS (mm) ";ZZZ
14080   LOCATE 10,20
14085   INPUT"ANGLE ALF FOR PART WITH  XOY (Deg) ";ALF
14090   LOCATE 11,20
14095   INPUT"ANGLE BTA FOR PART WITH  XOZ (Deg) ";BTA
14100   '
14105   LOCATE 13,20:COLOR 4,2
14110   PRINT"ARE THE INPUTS CORRECT (Y/N) ? "
14120   A£=INKEY£
14130   IF A£="Y" OR A£="y" THEN 14500
14140   IF A£="N" OR A£="n" THEN 14005
14150   GOTO 14120
14160   '
14170   '
14180   '
14190   '
14400   ' *********************************************
14401   ' *                                           *
14402   ' *          PATH PLANNING SECTION            *
14403   ' *                                           *
14404   ' *********************************************
```

```
14406   '
14410   '
14415   ' For arm & neck, the robot hand moves along a
14417   ' line which is defined by initial position & the
14418   ' angles ALF and BTA in Cartesian space:
14420   ' XX0 -- initial position specified along X axis
14425   ' YY0 -- initial position specified along Y axis
14428   ' ZZ0 -- initial position specified along Z axis
14430   ' BTA -- angle between the line and XOZ plane
14435   ' ALF -- angle between the line and XOY plane
14440   '
14445   ' For back, the robot hand moves in a flat surface
14448   ' which consists of the points in the back surface
14450   ' coordinates [Xback Yback Zback]:
14454   ' Xback -> [ -DPART/2, +DPART/2 ]
14458   ' Yback -> [ 0, LPART ]
14460   ' Zback -> [ 0, 0 ]
14462   '
14464   ' The back surface in Cartesian space is given by
14468   ' a plane attached to a line. The plane is a flat
14470   ' surface.  The line is the center line of the
14472   ' plane along its axial direction.
14474   ' XX0 -- initial position along X axis
14476   ' YY0 -- initial position along Y axis
14480   ' ZZ0 -- initial position along Z axis
14482   ' BTA -- angle between the line and XOZ plane
14484   ' ALF -- angle between back surface and XOY plane
14486   '
14490   '
14500   ' aa. PLANNING STRATEGY & HAND PARAMETERS .aa
14505   '
14510   COLOR 7,1:CLS
14515   LOCATE 10,20:COLOR 20,2
14520   PRINT"PATH PLANNING IS GOING ON, PLEASE WAIT !"
14525   '
14530   CC=3.141596/180
14535   ALF=ALF*CC
14540   BTA=BTA*CC
14545   '
14560   '
14570   '  ** Strategies -- Padding or Kneading **
14580   '
14590   '
14600   IF ACT£="PAD"   THEN 16000  ' Planning for padding
14610   '
14620   ' OTHERWISE PLANNING FOR KNEADING
14630   '
14640   '
15000   '  bb.  PLANNING FOR KNEADING ACTIONS  .bb
15010   '
15015   '
15020   ' bb * ARRAY * Array used for keading
15022   '
```

```
15025   ' XKT(30)    -- TASK POSITION ALONG X AXIS
15027   ' YKT(30)    -- TASK POSITION ALONG Y AXIS
15030   ' ZKT(30)    -- TASK POSITION ALONG Z AXIS
15032   ' OO(3,3)    -- TASK ORIENTATION
15034   ' XKP(30)    -- ROBOT ARM X CONTROL POSITION
15036   ' YKP(30)    -- ROBOT ARM Y CONTROL POSITION
15038   ' ZKP(30)    -- ROBOT ARM Z CONTROL POSITION
15040   ' QKR(30)    -- ROBOT  ROLL AGNLE
15044   ' QKP(30)    -- ROBOT PITCH ANGLE
15048   ' PP(3)      -- ROBOT ARM POSITIONS
15050   ' QQ(5)      -- ROBOT JOINT ANGLE
15052   ' QF(30)     -- ROBOT FINGER OPENNING ANGLES
15054   ' CPZ(30)    -- COMPLIANCE GRASPING DISTANCE
15060   '
15062   '
15065   ' bb * Parameters Retaining *
15068   '
15070   ACTK£=ACT£
15075   HANDK£=HAND£
15078   PNNK=PNN-1
15080   PMMK=PMM-1
15082   SPEEDAK=SPEEDA
15084   FTIMEK=FTIME
15086   FFORCEK=FFORCE
15088   '
15090   '
15110   ' bb * Task matrix *  Position & orientation
15115   '
15117   ' ** Initial positions of the part
15119   '
15120   XX0=XXX
15125   YY0=YYY
15130   ZZ0=ZZZ
15140   '
15160   '
15165   ' ** Task Orientation (Attached to grasp center)
15168   '
15170   TNX=-COS(BTA)*COS(ALF)               ' nx
15172   TNY=-SIN(BTA)*COS(ALF)               ' ny
15174   TNZ=-SIN(ALF)                        ' nz
15176   '
15178   TOX=-SIN(BTA)                        ' ox
15180   TOY=COS(BTA)                         ' oy
15182   TOZ=0                                ' oz
15184   '
15186   TAX=+COS(BTA)*SIN(ALF)               ' ax
15188   TAY=+SIN(BTA)*SIN(ALF)               ' ay
15190   TAZ=-COS(ALF)                        ' az
15192   '
15195   '
15200   ' ** Task Position (Attached to grasp center)
15205   '
15210   DLPART=LPART/PNNK        ' part segment
15215   '
```

```
15220   FOR I=0 TO PNNK
15225   XKT(I)=XX0+I*DLPART*COS(ALF)*COS(BTA)    ' px
15230   YKT(I)=YY0+I*DLPART*COS(ALF)*SIN(BTA)    ' py
15235   ZKT(I)=ZZ0+I*DLPART*SIN(ALF)             ' pz
15240   NEXT I
15245   '
15248   '
15250   ' bb * Finger joint *  Finger joint space
15252   '
15254   ' ** Initial compliance parameters
15256   '
15258   IF HANDK£="HANDOLD" OR HANDK£="handold" THEN 15280
15260   '
15262   LFING=90          ' HAND -II parameters
15264   DFING=40
15266   ZH=65
15270   CPX=-37:CPY=0
15272   '
15274   GOTO 15300
15276   '
15280   LFING=115         ' HAND -I parameters
15282   DFING=40
15284   ZH=65
15286   CPX=-34:CPY=0
15288   '
15290   ' ** Compliance grasping distance & finger angles
15292   '
15294   ' DDDN -- the end diameter of the part
15296   ' DDD0 -- the initial diameter of the part
15298   ' DDDI -- the ith diameter of the part
15300   '
15302   DDD0=DPART
15304   DDDN=DPART
15308   '
15310   FOR I=0 TO PNNK
15312   DDDI=DDD0+(DDDN-DDD0)*I/PNNK
15314   XR=(DDDI-DFING)/2
15318   XQ=ABS(LFING*LFING-XR*XR)
15320   DGRASP=SQR(XQ)              ' GRASP DISTANCE
15321   '
15322   ' ** 1st. compliance distance along Z axis
15323   '
15325   CPZ(I)=ZH+DGRASP
15327   '
15330   ' ** 2nd. finger joint angle
15332   '
15336   YY=DGRASP:XX=XR
15340   GOSUB 17800       ' using KB
15345   QF(I)=Q           ' FINGER JOINT ANGLE
15350   NEXT I
15352   '
15360   '
15370   ' bb * ARM MATRIX *  AFTER COMPLIANCE
15374   '
```

```
15376   ' ** 1st. The wrist orientation is the same
15378   '         as that of task orientation
15380   '
15382   ' ** 2nd. The wrist position after compliance
15385   '
15388   FOR I=0 TO PNNK
15390   XKP(I)=XKT(I)-(CPX*TNX+CPY*TOX+CPZ(I)*TAX)
15392   YKP(I)=YKT(I)-(CPX*TNY+CPY*TOY+CPZ(I)*TAY)
15395   ZKP(I)=ZKT(I)-(CPX*TNZ+CPY*TOZ+CPZ(I)*TAZ)
15398   NEXT I
15399   '
15400   '
15405   ' bb * Inverse * pitch & roll angles
15410   '
15415   FOR I=0 TO PNNK
15418   '
15420   OO(0,0)=TNX:OO(0,1)=TOX:OO(0,2)=TAX
15425   OO(1,0)=TNY:OO(1,1)=TOY:OO(1,2)=TAY
15430   OO(2,0)=TNZ:OO(2,1)=TOZ:OO(2,2)=TAZ
15432   '
15434   PP(0)=XKP(I):PP(1)=YKP(I):PP(2)=ZKP(I)
15438   BTA=BTA
15440   ALF=ALF
15450   '
15455   ' ** Inverse solution using KB
15460   '
15465   GOSUB 17000
15470   '
15472   IF YERR=1 THEN COLOR 7,1:END
15474   '
15478   QKR(I)=QQ(4):QKP(I)=QQ(6)
15480   '
15484   NEXT I
15488   '
15490   '
15500   ' bb * Path data save *
15560   '
15565   ' |----------------------------------------|
15570   ' |  Record the path planned for Kneading  |
15575   ' |----------------------------------------|
15580   '
15600   ' bb * KB -1 *  Path planned recording
15605   '
15610   ' ** 1st.  Data file name input
15615   '
15620   COLOR 7,1:CLS
15625   LOCATE 5,20:COLOR 2,4
15630   PRINT"SAVE THE PLANNED PATH AS DATA FILES"
15634   '
15638   COLOR 1,7
15640   LOCATE 7,15
15642   INPUT"PLEASE INPUT FILE NAME FOR .DOC ";FDOC$
15650   LOCATE 8,15
15655   INPUT"PLEASE INPUT FILE NAME FOR .DAT ";FDAT$
15658   '
```

```
15660   LOCATE 10,15:COLOR 2,4
15665   PRINT"ARE THE INPUTS CORRECT (Y/N) ? "
15668   A£=INKEY£
15670   IF A£="Y" OR A£="y" THEN 15700
15675   IF A£="N" OR A£="n" THEN 15620
15680   GOTO 15668
15685   '
15690   '
15700   ' ** 2nd. Data processing
15705   '
15710   CD=180/3.141596
15714   FOR I=0 TO PNNK
15716   QKR(I)=QKR(I)*CD:QKP(I)=QKP(I)*CD
15718   QF(I)=QF(I)*CD
15720   NEXT I
15722   '
15724   '
15726   ' ** 3rd. data saving section
15728   '
15730   ' ** .DOC FILE **
15731   '
15732   OPEN FDOC£ FOR OUTPUT AS #1
15734   PRINT #1," FOR THE PART & PART LOCATION INPUTS"
15736   PRINT #1,"                                     "
15738   PRINT #1,"  DDD= ";USING"+###.###";DPART
15740   PRINT #1,"  LLL= ";USING"+###.###";LPART
15742   PRINT #1,"  XX0= ";USING"+###.###";XX0
15744   PRINT #1,"  YY0= ";USING"+###.###";YY0
15746   PRINT #1,"  ZZ0= ";USING"+###.###";ZZ0
15748   PRINT #1,"  BTA= ";USING"+###.###";BTA*CD
15750   PRINT #1,"  ALF= ";USING"+###.###";ALF*CD
15752   PRINT #1,"                                     "
15754   PRINT #1," AND THE TASK INPUTS"
15756   PRINT #1,"  ACT= ";ACTK£
15757   PRINT #1," HAND= ";HANDK£
15758   PRINT #1,"  PNN= ";USING"####";PNNK
15760   PRINT #1,"  PMM= ";USING"####";PMMK
15762   PRINT #1,"SPEED= ";USING"####";SPEEDAK
15764   PRINT #1,"FTIME= ";USING"##.##";FTIMEK
15766   PRINT #1,"FORCE= ";USING"##.##";FFORCEK
15768   PRINT #1,"     ":PRINT #1," "
15770   '
15774   PRINT #1," PATH DATA FOR ROBOT MOTION CONTROL"
15778   PRINT #1," "
15780   '
15782   PRINT #1,"   NX= ";USING"+##.##";TNX
15784   PRINT #1,"   NY= ";USING"+##.##";TNY
15786   PRINT #1,"   NZ= ";USING"+##.##";TNZ
15790   PRINT #1,"   OX= ";USING"+##.##";TOX
15792   PRINT #1,"   OY= ";USING"+##.##";TOY
15794   PRINT #1,"   OZ= ";USING"+##.##";TOZ
15798   PRINT #1,"   AX= ";USING"+##.##";TAX
15800   PRINT #1,"   AY= ";USING"+##.##";TAY
15804   PRINT #1,"   AZ= ";USING"+##.##";TAZ
15806   '
```

```
15810   FOR I=0 TO PNNK
15812   PRINT #1,"POSITION =";USING"###";I
15814   PRINT #1,"  PX= ";USING"+###.###";XKP(I)
15816   PRINT #1,"  PY= ";USING"+###.###";YKP(I)
15818   PRINT #1,"  PZ= ";USING"+###.###";ZKP(I)
15820   PRINT #1,"  QP= ";USING"+###.###";QKP(I)
15822   PRINT #1,"  QR= ";USING"+###.###";QKR(I)
15824   PRINT #1,"  QF= ";USING"+###.###";QF(I)
15828   PRINT #1," "
15830   NEXT I
15834   CLOSE #1
15840   '
15842   '
15845   ' ** .DAT FILE **
15850   '
15852   '
15854   OPEN FDAT£ FOR OUTPUT AS #1
15858   '
15860   WRITE #1,ACTK£
15864   WRITE #1,HANDK£
15868   WRITE #1,PNNK
15870   WRITE #1,PMMK
15876   WRITE #1,SPEEDAK
15878   WRITE #1,FTIMEK
15880   WRITE #1,FFORCEK
15882   '
15885   WRITE #1,TNX,TNY,TNZ
15890   WRITE #1,TOX,TOY,TOZ
15895   WRITE #1,TAX,TAY,TAZ
15898   '
15900   FOR I=0 TO PNNK
15905   WRITE #1,XKP(I),YKP(I),ZKP(I)
15910   WRITE #1,QKP(I),QKR(I),QF(I)
15915   NEXT I
15920   CLOSE #1
15930   '
15940   RETURN
15950   '
15970   '
15990   '
16000   '    cc.  PLANNING FOR PADDING ACTION .cc
16015   '
16020   '   cc * ARRAY * Array used for padding
16025   '
16030   ' XPT(10,30)   -- TASK POSITION ALONG X AXIS
16032   ' YPT(10,30)   -- TASK POSITION ALONG Y AXIS
16034   ' ZPT(10,30)   -- TASK POSITION ALONG Z AXIS
16036   ' OO(3,3)      -- TASK ORIENTATION
16038   ' XPP(10,30)   -- ROBOT ARM X CONTROL POSITION
16040   ' YPP(10,30)   -- ROBOT ARM Y CONTROL POSITION
16042   ' ZPP(10,30)   -- ROBOT ARM Z CONTROL POSITION
16044   ' QPR(10,30)   -- ROBOT  ROLL ANGLE
16046   ' QPP(10,30)   -- ROBOT PITCH ANGLE
16048   ' QQ(5)        -- ROBOT JOINT ANGLES
16050   ' PP(3)        -- ROBOT ARM POSITIONS (X Y Z)
```

```
16060   ' cc * Parameter retain *
16062   '
16065   ACTP£=ACT£
16068   HANDP£=HAND£
16070   PNNP=PNN-1
16074   PMMP=PMM-1
16078   SPEEDAP=SPEEDA
16080   FTIMEP=FTIME
16082   FFORCEP=FFORCE
16085   '
16090   '
16100   ' cc * Task matrix * Position & orientation
16105   '
16110   ' ** Initial position of the part
16112   '
16116   XX0=XXX
16118   YY0=YYY
16120   ZZ0=ZZZ
16125   '
16128   '
16130   ' ** Task Orientation (Attached to robotic palm)
16135   '
16140   TNX=-COS(BTA)*COS(ALF)             ' nx
16145   TNY=-SIN(BTA)*COS(ALF)             ' ny
16148   TNZ=-SIN(ALF)                      ' nz
16150   '
16152   TOX=-SIN(BTA)                      ' ox
16155   TOY=+COS(BTA)                      ' oy
16158   TOZ=0                              ' oz
16160   '
16162   TAX=+COS(BTA)*SIN(ALF)             ' ax
16164   TAY=+SIN(BTA)*SIN(ALF)             ' ay
16168   TAZ=-COS(ALF)                      ' az
16170   '
16172   '
16174   ' ** Task position (Attached to the robotic palm)
16176   '
16178   DLPART=LPART/PNNP      ' part segment along axial
16180   DDPART=DPART/PMMP      ' part segment along radial
16182   '
16186   FOR I=0 TO PMMP
16188   FOR J=0 TO PNNP
16190   '
16195   ' POSITION IN BACK PLANE
16200   '
16210   XBK=-DPART/2+I*DDPART
16215   YBK=J*DLPART
16218   ZBK=10
16220   '
16222   ' POSITION IN CARTESIAN SPACE
16228   '
16230   SBT=SIN(BTA):CBT=COS(BTA)
16232   SAT=SIN(ALF):CAT=COS(ALF)
16238   XPT(I,J)= SBT*XBK + CBT*CAT*YBK - CBT*SAT*ZBK+XX0
16240   YPT(I,J)=-CBT*XBK + SBT*CAT*YBK - SBT*SAT*ZBK+YY0
```

```
16245   ZPT(I,J)=                   SAT*YBK +     CAT*ZBK+ZZ0
16248   '
16250   NEXT J
16260   NEXT I
16280   '
16290   '
16300   ' cc * Hand space * Compliance distance & angle
16305   '
16310   ' ** Compliance distance
16312   '
16314   IF HANDP£="HANDOLD" OR HANDP£="handold" THEN 16340
16316   '
16320   ZH=65             ' HAND -II Parameters
16322   CPX=-5
16324   CPY=0
16326   DPALM=30
16330   '
16332   GOTO 16360
16334   '
16340   ZH=65             ' HAND -I Parameters
16342   CPX=-15
16346   CPY=0
16348   DPALM=30
16350   '
16352   '
16360   CPZ=ZH+DPALM
16364   '
16368   ' ** Finger angle
16370   '
16375   QF=0
16380   '
16385   '
16400   ' cc * ARM MATRIX *  AFTER COMPLIANCE
16402   '
16404   ' ** 1st. The palm orientation is the same as
16408   '           that of task orientation
16410   '
16412   ' ** 2nd. The robot arm position after compliance
16415   '
16418   FOR I=0 TO PMMP
16420   FOR J=0 TO PNNP
16422   XPP(I,J)= XPT(I,J) - (CPX*TNX+CPY*TOX+CPZ*TAX)
16426   YPP(I,J)= YPT(I,J) - (CPX*TNY+CPY*TOY+CPZ*TAY)
16430   ZPP(I,J)= ZPT(I,J) - (CPX*TNZ+CPY*TOZ+CPZ*TAZ)
16438   NEXT J
16440   NEXT I
16444   '
16448   '
16450   ' cc * Inverse * pitch & roll angles
16455   '
16460   FOR I=0 TO PMMP
16465   FOR J=0 TO PNNP
16470   '
16475   OO(0,0)=TNX:OO(0,1)=TOX:OO(0,2)=TAX
16480   OO(1,0)=TNY:OO(1,1)=TOY:OO(1,2)=TAY
```

```
16485   OO(2,0)=TNZ:OO(2,1)=TOZ:OO(2,2)=TAZ
16488   '
16490   PP(0)=XPP(I,J)
16495   PP(1)=YPP(I,J)
16500   PP(2)=ZPP(I,J)
16505   '
16510   BTA=BTA:ALF=ALF
16525   '
16528   ' ** Inverse solution using KB
16530   '
16532   GOSUB 17000
16534   '
16536   IF YERR=1 THEN COLOR 7,1:END
16538   '
16540   QPR(I,J)=QQ(4)           ' Roll  angle
16542   QPP(I,J)=QQ(6)           ' Pitch angle
16545   '
16550   NEXT J
16555   NEXT I
16560   '
16562   '
16565   '
16570   '  cc * Path data save *
16580   '
16600   ' |---------------------------------------|
16602   ' |  Record the path planned for Padding  |
16606   ' |---------------------------------------|
16610   '
16612   '
16616   ' cc * KB -1 * Path planned recording
16618   '
16620   ' ** 1st. Data file name input
16622   '
16624   COLOR 7,1:CLS
16626   LOCATE 5,20:COLOR 2,4
16628   PRINT"SAVE THE PLANNED PATH AS DATA FILES"
16630   '
16632   COLOR 1,7
16635   LOCATE 7,15
16638   INPUT"PLEASE INPUT FILE NAME FOR .DOC ";FDOC£
16640   LOCATE 8,15
16645   INPUT"PLEASE INPUT FILE NAME FOR .DAT ";FDAT£
16650   '
16655   LOCATE 10,15:COLOR 2,4
16660   PRINT"ARE THE INPUTS CORRECT (Y/N) ? "
16665   A£=INKEY£
16670   IF A£="Y" OR A£="y" THEN 16700
16675   IF A£="N" OR A£="n" THEN 16624
16680   GOTO 16665
16690   '
16695   '
16700   ' ** 2nd. Data processing
16705   '
16710   CD=180/3.141596
16712   FOR I=0 TO PMMP
```

```
16716   FOR J=0 TO PNNP
16718   QPR(I,J)=QPR(I,J)*CD
16720   QPP(I,J)=QPP(I,J)*CD
16722   NEXT J
16726   NEXT I
16728   '
16730   '
16732   ' ** 3rd. Data saving section
16734   '
16736   ' ** .DOC FILE **
16738   '
16740   OPEN FDOC£ FOR OUTPUT AS #1
16742   PRINT #1," FOR THE PART & PART LOCATION INPUTS"
16745   PRINT #1,"                           "
16748   PRINT #1,"  WWW= ";USING"###.###";LPART
16750   PRINT #1,"  HHH= ";USING"###.###";DPART
16752   PRINT #1,"  XX0= ";USING"+###.###";XX0
16756   PRINT #1,"  YY0= ";USING"+###.###";YY0
16758   PRINT #1,"  ZZ0= ";USING"+###.###";ZZ0
16760   PRINT #1,"  BTA= ";USING"+###.###";BTA*CD
16762   PRINT #1,"  ALF= ";USING"+###.###";ALF*CD
16768   PRINT #1,"                           "
16770   PRINT #1," AND THE TASK INPUTS"
16772   PRINT #1,"  ACT= ";ACTP£
16775   PRINT #1," HAND= ";HANDP£
16778   PRINT #1,"  PNN= ";USING"####";PNNP
16780   PRINT #1,"  PMM= ";USING"####";PMMP
16782   PRINT #1,"SPEED= ";USING"####";SPEEDAP
16785   PRINT #1,"FTIME= ";USING"##.##";FTIMEP
16788   PRINT #1,"FORCE= ";USING"##.##";FFORCEP
16790   PRINT #1,"    ":PRINT #1,"  "
16795   '
16800   PRINT #1," PATH DATA FOR ROBOT MOTION CONTROL"
16805   PRINT #1,"    "
16810   PRINT #1,"   NX= ";USING"+##.##";TNX
16815   PRINT #1,"   NY= ";USING"+##.##";TNY
16818   PRINT #1,"   NZ= ";USING"+##.##";TNZ
16820   PRINT #1,"   OX= ";USING"+##.##";TOX
16822   PRINT #1,"   OY= ";USING"+##.##";TOY
16826   PRINT #1,"   OZ= ";USING"+##.##";TOZ
16828   PRINT #1,"   AX= ";USING"+##.##";TAX
16830   PRINT #1,"   AY= ";USING"+##.##";TAY
16832   PRINT #1,"   AZ= ";USING"+##.##";TAZ
16836   '
16838   PRINT #1,"      "
16840   '
16842   FOR I=0 TO PMMP
16845   FOR J=0 TO PNNP
16848   PRINT #1,"POSITION = ( ";USING"##";I;
16850   PRINT #1," , ";USING"##";J;
16851   PRINT #1," )"
16852   PRINT #1,"  PX= ";USING"+####.###";XPP(I,J)
16854   PRINT #1,"  PY= ";USING"+####.###";YPP(I,J)
16858   PRINT #1,"  PZ= ";USING"+####.###";ZPP(I,J)
16860   PRINT #1,"  QP= ";USING"+####.###";QPP(I,J)
```

```
16862   PRINT #1,"  QR= ";USING"+####.###";QPR(I,J)
16864   PRINT #1,"    "
16868   NEXT J
16870   NEXT I
16874   CLOSE #1
16880   '
16890   ' ** .DAT FILE **
16895   '
16900   OPEN FDAT£ FOR OUTPUT AS #1
16902   WRITE #1,ACTP£
16906   WRITE #1,HANDP£
16908   WRITE #1,PNNP
16910   WRITE #1,PMMP
16912   WRITE #1,SPEEDAP
16914   WRITE #1,FTIMEP
16918   WRITE #1,FFORCEP
16920   '
16922   WRITE #1,TNX,TNY,TNZ
16924   WRITE #1,TOX,TOY,TOZ
16926   WRITE #1,TAX,TAY,TAZ
16928   '
16930   FOR I=0 TO PMMP
16936   FOR J=0 TO PNNP
16938   WRITE #1,XPP(I,J),YPP(I,J),ZPP(I,J)
16940   WRITE #1,QPP(I,J),QPR(I,J)
16942   NEXT J
16945   NEXT I
16948   CLOSE #1
16950   '
16960   '
16970   RETURN
16980   '
16990   '
17000   ' **********************************************
17005   ' *                                            *
17010   ' *        Inverse -- I   (Joint space)        *
17015   ' *                                            *
17020   ' *    **  Inverse computation for robot arm   *
17025   ' *    **  Direct computing to verify          *
17030   ' *    **  Intelligent boundary checking,etc.  *
17035   ' *                                            *
17040   ' **********************************************
17045   '
17050   '
17060   '  %%** KB -1. Inverse computation (joint space)
17070   '
17100   '  ** Initial parameter setting for robot arm
17110   '
17115   D1=300          ' Robot shoulder height
17120   A2=250          ' Robot upper arm length
17125   A3=160          ' Robot lower arm length
17130   D5=72           ' Robot wrist length
17135   '
17140   '
17150   CC=3.141596/180
```

```
17160   '
17170   '
17180   '  ** I.1. ** -- Q1 -- (-60,240)
17190   '
17200   YY=PP(1):XX=PP(0)
17205   GOSUB 17800
17210   Q1=Q
17215   Q1L=-60*CC:Q1H=240*CC
17220   IF (Q1<Q1L OR Q1>Q1H) THEN GOTO 17600
17221   QQ(0)=Q1
17224   '
17226   '  ** I.2. ** -- Q234 -- (-230,190)
17230   '
17234   QP=-(3.141596/2-ALF)
17240   QPL=-230*CC:QPH=190*CC
17242   IF (QP<QPL OR QP>QPH) THEN 17600
17244   Q234=QP:QQ(6)=QP
17245   '
17246   '  ** I.3. ** -- Q5 -- (-180,180)
17248   '
17250   PP=COS(Q234):PJ=ABS(PP)
17252   '
17254   IF PJ<0.05 THEN 17290
17256   '
17260   '  -- && FOR COS(Q234) <> 0
17262   '
17264   YY=-OO(2,1):XX=-OO(2,0)
17270   '
17274   GOSUB 17800
17276   QQ(4)=Q:Q5=Q
17280   '
17284   GOTO 17350
17286   '
17290   '  -- && FOR COS(Q234) = 0
17294   '
17300   PP=SIN(Q234):PJ=SGN(PP)
17302   IF PJ=-1 THEN 17326
17306   '
17310   YY=-OO(0,1):XX=OO(0,0)
17312   GOSUB 17800
17314   QQ(4)=Q1-Q:Q5=QQ(4)
17320   GOTO 17350
17324   '
17326   YY=-OO(0,1):XX=-OO(0,0)
17330   GOSUB 17800
17332   QQ(4)=-Q1+Q:Q5=QQ(4)
17334   '
17338   '
17340   '  ** I.4. ** -- Q3 -- (-110,0)
17342   '
17350   ALFA=PP(2)-D1-D5*SIN(Q234)
17352   BETA=PP(0)*COS(Q1)+PP(1)*SIN(Q1)-D5*COS(Q234)
17354   PPC1=ALFA*ALFA+BETA*BETA-A2*A2-A3*A3
17356   PPC2=2*A2*A3
17358   PPC=PPC1/PPC2
```

```
17362   IF ABS(PPC)>1 THEN 17600         'NO SOLUTION
17363   '
17365   PPS=1-PPC*PPC
17366   PPS=SQR(PPS):PJ=ABS(PPC)
17368   IF PJ<0.00051 THEN Q3=-3.141596/2:GOTO 17380
17370   PAA=ABS(PPS/PPC)
17372   Q3A=ATN(PAA)
17373   '
17374   IF PPC>=0 THEN Q3=-Q3A:GOTO 17380
17375   IF PPC<0 AND Q3A<70*CC THEN Q3=-Q3A:GOTO 17380
17376   Q3=-(3.141596-Q3A)
17380   Q3L=-110*CC:Q3H=0
17382   IF (Q3<Q3L OR Q3>Q3H) THEN 17600  'OUT OF WORKRANGE
17384   '
17386   QQ(2)=Q3
17388   '
17389   '
17390   '  ** I.5. ** -- Q2 -- (-30,100)
17392   '
17400   PL1=A3*COS(Q3)+A2:PL2=A3*SIN(Q3)
17402   PP1=ALFA*PL1-BETA*PL2
17406   PP2=BETA*PL1+ALFA*PL2
17408   '
17410   IF (PP1=0 AND PP2=0) THEN 17600  'NO SOULTION
17412   '
17414   YY=PP1:XX=PP2
17416   GOSUB 17800
17418   Q2=Q
17420   Q2L=-30*CC:Q2H=100*CC
17422   IF (Q2<Q2L OR Q2>Q2H) THEN 17600  'OUT OF WORKRANGE
17424   '
17426   QQ(1)=Q2
17428   '
17430   '
17432   '  ** I.6. ** -- Q4 -- (-90,90)
17434   '
17440   Q4=Q234-Q3-Q2
17442   '
17446   Q4L=-3.141596/2:Q4H=3.141596/2
17448   IF (Q4<Q4L OR Q4>Q4H) THEN 17600  'OUT OF WORKRANGE
17450   '
17452   QQ(3)=Q4
17454   '
17460   '
17500   '   %%** KB -2 . Verification
17510   '
17520   Q23=Q2+Q3
17522   P1=SIN(Q234):P2=COS(Q234)
17524   P3=SIN(Q23):P4=COS(Q23)
17530   '
17534   NXX=P1*COS(Q1)*COS(Q5)+SIN(Q1)*SIN(Q5)
17536   NYY=P1*SIN(Q1)*COS(Q5)-COS(Q1)*SIN(Q5)
17538   NZZ=-P2*COS(Q5)
17540   OXX=P1*COS(Q1)*SIN(Q5)-SIN(Q1)*COS(Q5)
17542   OYY=P1*SIN(Q1)*SIN(Q5)+COS(Q1)*COS(Q5)
```

```
17544   OZZ=-P2*SIN(Q5)
17546   AXX=P2*COS(Q1)
17548   AYY=P2*SIN(Q1)
17550   AZZ=P1
17552   '
17554   PXX=(A2*COS(Q2)+A3*P4+D5*P2)*COS(Q1)
17556   PYY=(A2*COS(Q2)+A3*P4+D5*P2)*SIN(Q1)
17558   PZZ=D1+A2*SIN(Q2)+A3*P3+D5*P1
17560   '
17562   DNX=ABS(NXX-OO(0,0)):DOX=ABS(OXX-OO(0,1))
17563   DAX=ABS(AXX-OO(0,2))
17564   DPX=ABS(PXX-PP(0)):DPY=ABS(PYY-PP(1))
17565   DPZ=ABS(PZZ-PP(2))
17566   IF (DNX>0.1 OR DOX>0.1 OR DAX>0.1) THEN 17700
17568   IF (DPX>1.5 OR DPY>1.5 OR DPZ>1.5) THEN 17700
17570   '
17574   '  !!!! If the results are reasonale !!!!
17576   YERR=0
17580   RETURN
17590   '
17595   '
17600   '  ?? If the results are not reasonable ??
17610   '
17640   '
17650   GOSUB 9000
17660   LOCATE 21,18:COLOR 4,2
17670   PRINT"ERROR OCCURRED DURING PATH PLANNING,";
17675   PRINT" PLEASE ADJUST"
17680   YERR=1
17685   RETURN      ' RETURN TO MAIN MENU
17690   '
17695   '
17700   GOSUB 9000
17710   LOCATE 21,23:COLOR 4,2
17720   PRINT"THE POSITION CANNOT BE ADJUSTED, SORRY!"
17735   YERR=1
17740   RETURN      ' RETURN TO MAIN MENU
17750   '
17760   '
17800   ' ***********************************************
17802   ' *                                             *
17804   ' *       ANGLE COMPUTATION OF ATN(YY/XX)       *
17808   ' *                                             *
17810   ' ***********************************************
17815   '
17820   '
17830   IF XX=0 THEN 17850
17832   IF YY=0 THEN 17860
17833   AA=ABS(YY/XX):QA=ATN(AA)
17835   '
17838   IF (YY>0 AND XX>0) THEN Q=QA:RETURN
17840   IF (YY>0 AND XX<0) THEN Q=3.141596-QA:RETURN
17842   IF (YY<0 AND XX>0) THEN Q=-QA:RETURN
17844   IF (YY<0 AND XX<0) THEN Q=3.141596+QA:RETURN
17848   '
```

```
17850   IF (YY>0) THEN Q=3.141596/2:RETURN
17852   IF (YY<0) THEN Q=-3.141596/2:RETURN
17856   '
17860   IF (XX>0) THEN Q=0:RETURN
17861   IF (XX<0) THEN Q=3.141596:RETURN
17870   '
17880   '
17890   '
18000   ' ************************************************
18002   ' *                                              *
18004   ' *       Inverse -- II  (Cartesian space)       *
18006   ' *                                              *
18008   ' ************************************************
18010   '
18012   '
18014   CC=3.14159/180
18016   '
18020   '  ** Q1 **
18024   '
18028   YY=PP(1):XX=PP(0)
18030   GOSUB 17800
18034   Q1=Q
18038   Q1L=-60*CC-0.1:Q1H=240*CC+0.1
18040   IF (Q1<Q1L OR Q1>Q1H) THEN 18050
18044   QQ(0)=Q1
18048   '
18050   '  ** Q234 **
18054   '
18058   QP=-(3.14159/2-ALF)
18060   Q234=QP
18064   '
18068   '
18070   '  ** Q5 **
18072   '
18074   '   NOTE:  Q5=(BTA+90)-(Q1+90)
18076   '
18078   Q5=BTA-Q1
18080   '
18084   QQ(4)=Q5
18088   '
18090   RETURN
18094   '
18098   '
19000   ' ************************************************
19002   ' *                                              *
19004   ' *       THE DEFINED POSITIONS IN THE KB        *
19006   ' *                                              *
19008   ' ************************************************
19010   '
19014   '
19016   '
19018   REINPUT=0
19020   IF LYC£="LOCP1" OR LYC£="locp1" THEN 19120
19025   IF LYC£="LOCP2" OR LYC£="locp2" THEN 19155
19030   IF LYC£="LOCP3" OR LYC£="locp3" THEN 19190
```

```
19035   IF LYC£="LOCP4" OR LYC£="locp4" THEN 19225
19040   IF LYC£="LOCP5" OR LYC£="locp5" THEN 19260
19045   '
19050   IF LYC£="LOCV1" OR LYC£="locv1" THEN 19300
19055   IF LYC£="LOCV2" OR LYC£="locv2" THEN 19335
19060   IF LYC£="LOCV3" OR LYC£="locv3" THEN 19370
19065   IF LYC£="LOCV4" OR LYC£="locv4" THEN 19405
19070   IF LYC£="LOCV5" OR LYC£="locv5" THEN 19440
19090   '
19100   REINPUT=1
19105   RETURN
19110   '
19115   '
19120   X=-300              ' LOCP #1
19125   Y=0
19130   Z=200
19135   AF=0
19140   BA=180
19145   RETURN
19150   '
19155   X=-250              ' LOCP #2
19160   Y=0
19165   Z=250
19170   AF=0
19175   BA=180
19180   RETURN
19185   '
19190   X=0                 ' LOCP #3
19195   Y=300
19200   Z=200
19205   AF=0
19210   BA=90
19215   RETURN
19220   '
19225   X=0                 ' LOCP #4
19230   Y=300
19235   Z=250
19240   AF=0
19245   BA=180
19250   RETURN
19255   '
19260   X=-300              ' LOCP #5
19265   Y=50
19270   Z=200
19275   AF=0
19280   BA=210
19285   RETURN
19290   '
19295   '
19300   X=-480              ' LOCV #1
19305   Y=0
19310   Z=200
19315   AF=90
19320   BA=180
19325   RETURN
```

```
19330   '
19335   X=-500              ' LOCV #2
19340   Y=0
19345   Z=200
19350   AF=90
19355   BA=180
19360   RETURN
19365   '
19370   X=-450              ' LOCV #3
19375   Y=0
19380   Z=150
19385   AF=90
19390   BA=180
19395   RETURN
19400   '
19405   X=0                 ' LOCV #4
19410   Y=500
19415   Z=200
19420   AF=90
19425   BA=90
19430   RETURN
19435   '
19440   X=-100              ' LOCV #5
19445   Y=200
19450   Z=170
19455   AF=90
19460   BA=90
19465   RETURN
19470   '
19500   '
19600   '
19700   '
20000   ' ***********************************************
20010   ' *                                             *
20020   ' *          TASKS DATA FILE LOADING            *
20030   ' *                                             *
20040   ' ***********************************************
20050   '
20060   '
20100   OPEN DTASK£ FOR INPUT AS #1
20110   '
20120   INPUT #1,ACT£
20130   INPUT #1,HAND£
20140   INPUT #1,PNN
20150   INPUT #1,PMM
20160   INPUT #1,SPEEDA
20170   INPUT #1,FTIME
20180   INPUT #1,FFORCE
20190   INPUT #1,DPART
20195   INPUT #1,LPART
20200   '
```

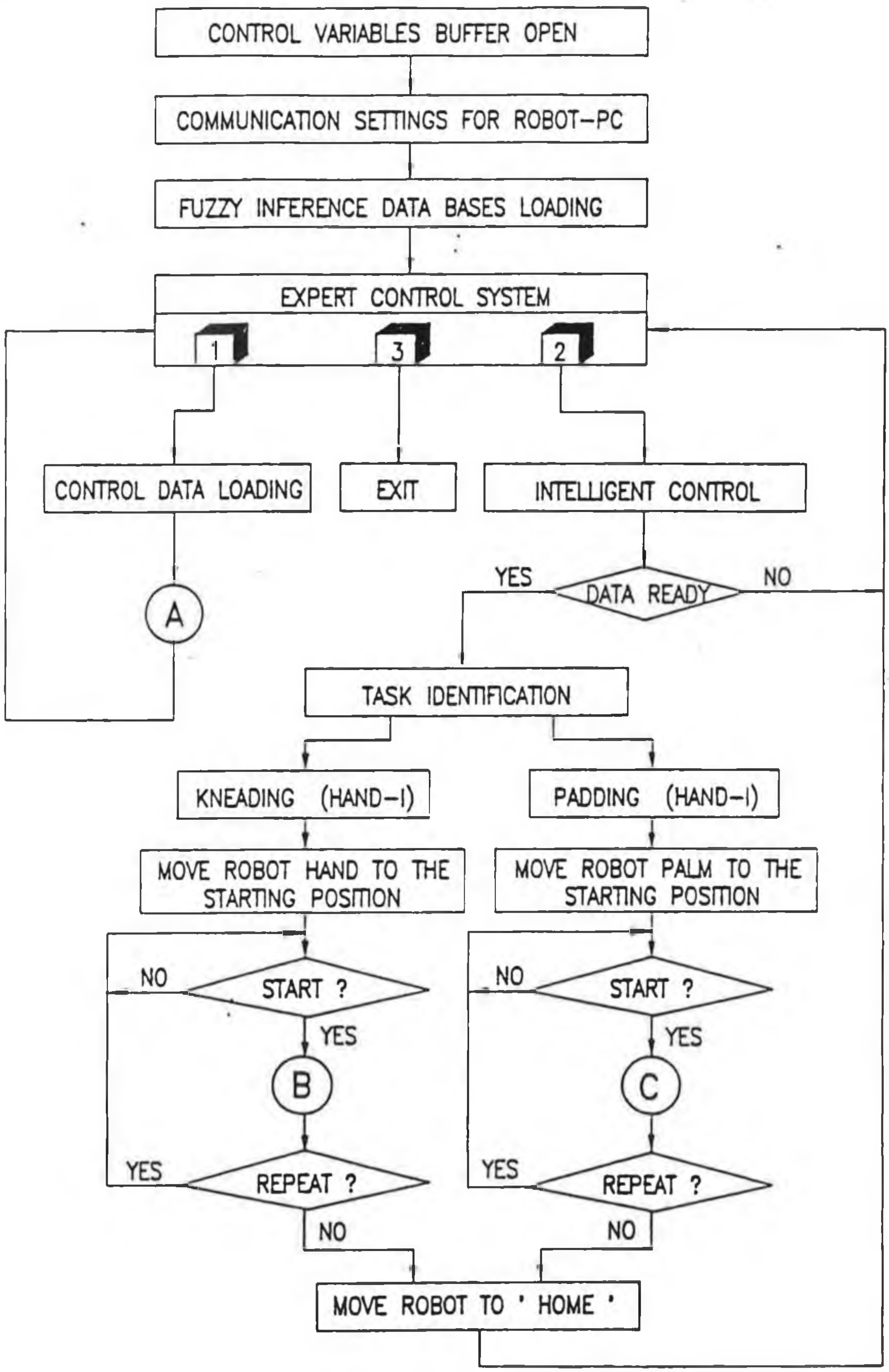
```
20210   INPUT #1,XXX
20220   INPUT #1,YYY
20230   INPUT #1,ZZZ
20240   INPUT #1,ALF
20250   INPUT #1,BTA
20260   '
20270   CLOSE #1
20280   '
20290   RETURN
20300   '
20320   '
20350   '
25000   ' ***********************************************
25010   ' *                                             *
25020   ' *       INFERRED TASK DATA RECORDING          *
25030   ' *                                             *
25040   ' ***********************************************
25050   '
25060   '
25100   COLOR 7,1:CLS
25110   LOCATE 10,20:COLOR 4,2
25120   INPUT"PLEASE INPUT THE TASK DATA NAME ";FTASK£
25130   '
25135   LOCATE 12,20:COLOR 2,4
25140   PRINT"IS ";FTASK£;" CORRECT (Y/N) ? "
25150   '
25160   A£=INKEY£
25170   IF A£="Y" OR A£="y" THEN 25210
25180   IF A£="N" OR A£="n" THEN 25100
25190   GOTO 25160
25200   '
25210   OPEN FTASK£ FOR OUTPUT AS #1
25220   '
25230   WRITE #1,ACT£
25240   WRITE #1,HAND£
25250   WRITE #1,PNN
25260   WRITE #1,PMM
25270   WRITE #1,SPEEDA
25280   WRITE #1,FTIME
25290   WRITE #1,FFORCE
25300   WRITE #1,DPART
25310   WRITE #1,LPART
25320   '
25330   CLOSE #1
25340   '
25350   '
25360   RETURN
25370   '
25380   '
25390   '
```

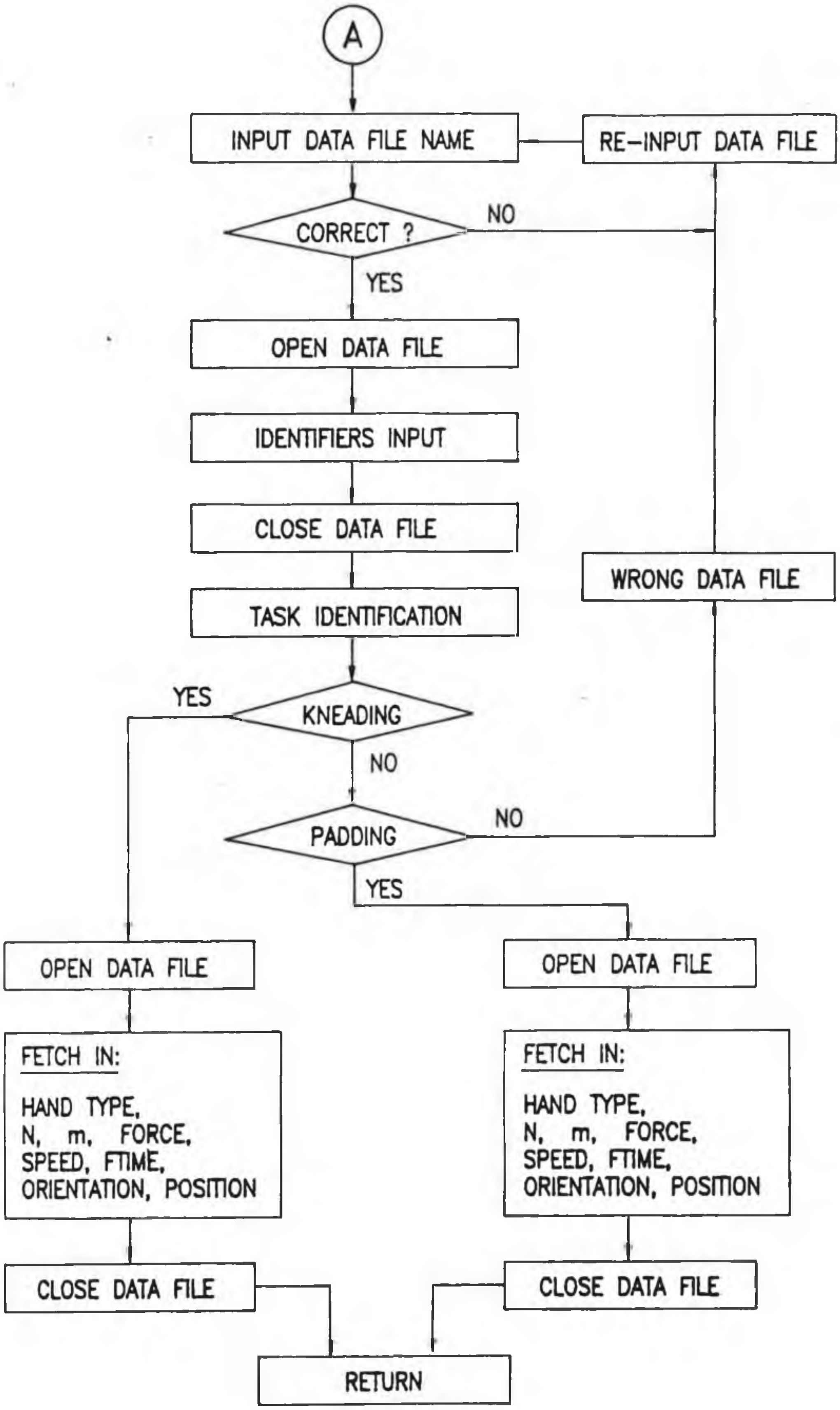
```
30000  ' **************************************
30002  ' *                                    *
30004  ' *   DATA BASE LOADING FOR PATH PLANNING  *
30006  ' *                                    *
30008  ' **************************************
30010  '
30016  '  ** LOADING PART SIZE DATA BASE **
30020  '
30030  FOR I=0 TO 5
30040  FOR J=0 TO 8
30050  READ PSIZE(I,J)
30060  NEXT J
30070  NEXT I
30080  '
30090  '
30100  '  ** LOADING FORCE DATA BASE **
30110  '
30120  FOR I=0 TO 2
30130  FOR J=0 TO 8
30140  READ FORCE(I,J)
30150  NEXT J
30160  NEXT I
30170  '
30180  '
30200  '  ** LOADING ROBOT ARM SPEED DATA BASE **
30210  '
30220  FOR I=0 TO 8
30230  READ SPDA(I)
30240  NEXT I
30250  '
30260  '
30300  '  ** LOADING FORCE RETENTION TIME DATA BASE **
30310  '
30320  FOR I=0 TO 8
30330  READ TFR(I)
30340  NEXT I
30350  '
30360  '
30400  '  ** LOADING MASSAGING PATH & POINT DATA BASE **
30410  '
30420  FOR I=0 TO 1
30430  FOR J=0 TO 4
30440  READ PTYPE(I,J)
30450  NEXT J
30460  NEXT I
30470  '
30480  '
30500  '  ** LOADING FUZZY MEMBERSHIP FOR INFERENCE **
30510  '
30520  FOR I=0 TO 8
30530  FOR J=0 TO 8
30540  READ FM(I,J)
30550  NEXT J
30560  NEXT I
30580  RETURN
```

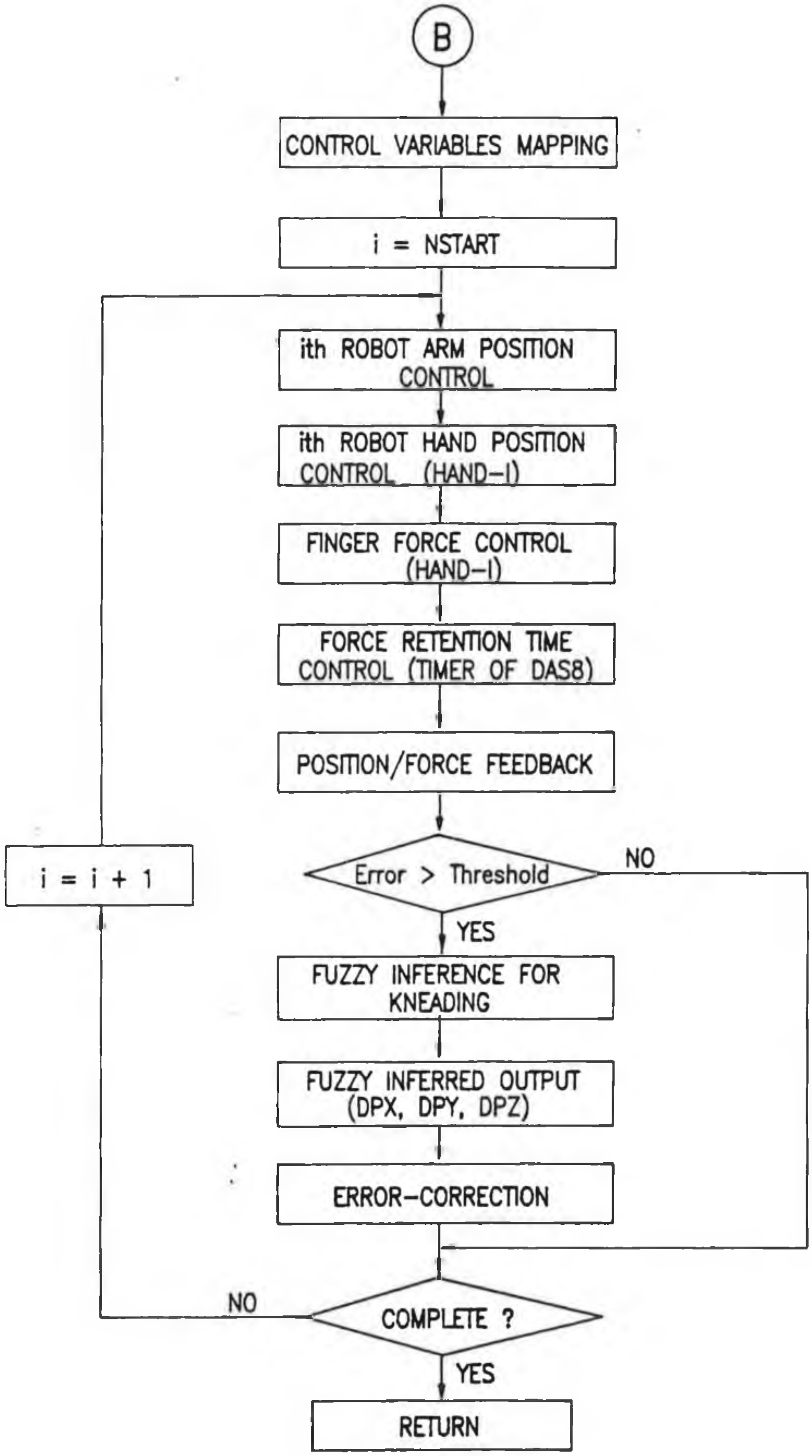
```
40000   ' ***********************************************
40010   ' *                                             *
40020   ' *      EXPERT DATA BASE & FUZZY MEMBERSHIP    *
40030   ' *                                             *
40040   ' ***********************************************
40050   '
40060   '
40100   '  ** PART SIZE DATA BASE **
40110   '
40115   '
40120   '   a. DATA BASE FOR ARM SIZE ( D - L )
40125   '
40130   DATA 60, 70,  80,  90, 100, 110, 120, 130, 140
40135   DATA 80, 90, 100, 110, 120, 130, 140, 150, 160
40140   '
40145   '   b. DATA BASE FOR NECK SIZE ( D - L )
40150   '
40155   DATA 80, 85, 90, 95, 100, 105, 110, 115, 120
40160   DATA 30, 35, 40, 45,  50,  55,  60,  65,  70
40165   '
40170   '   c. DATA BASE FOR BACK SIZE ( W - L )
40175   '
40180   DATA 80, 90, 100, 110, 120, 130, 140, 150, 160
40185   DATA 80, 90, 100, 110, 120, 130, 140, 150, 160
40190   '
40195   '
40200   '  ** MASSAGING FORCE DATA BASE **
40205   '
40210   '
40215   '  a. DATA BASE FOR ARM FORCE (N)
40220   '
40225   DATA 1.50,2.00,2.50,3.00,3.50,4.00,4.50,5.00,5.50
40230   '
40235   '  b. DATA BASE FOR NECK FORCE (N)
40240   '
40245   DATA 1.50,2.00,2.50,3.00,3.50,4.00,4.50,5.00,5.50
40250   '
40255   '  c. DATA BASE FOR BACK FORCE (N)
40260   '
40265   DATA 2.00,2.50,3.00,3.50,4.00,4.50,5.00,5.50,6.00
40270   '
40275   '
40300   '  ** DATA BASE OF ROBOT ARM SPEED (LEVEL) **
40310   '
40320   DATA 1,2,3,4,5,6,7,8,9
40330   '
40340   '
40400   '  ** DATA BASE OF FORCE RETENTION TIME **
40410   '
40420   DATA 2.00,1.50,1.50,1.00,1.00,1.00,0.75,0.75,0.50
40430   '
40440   '
```

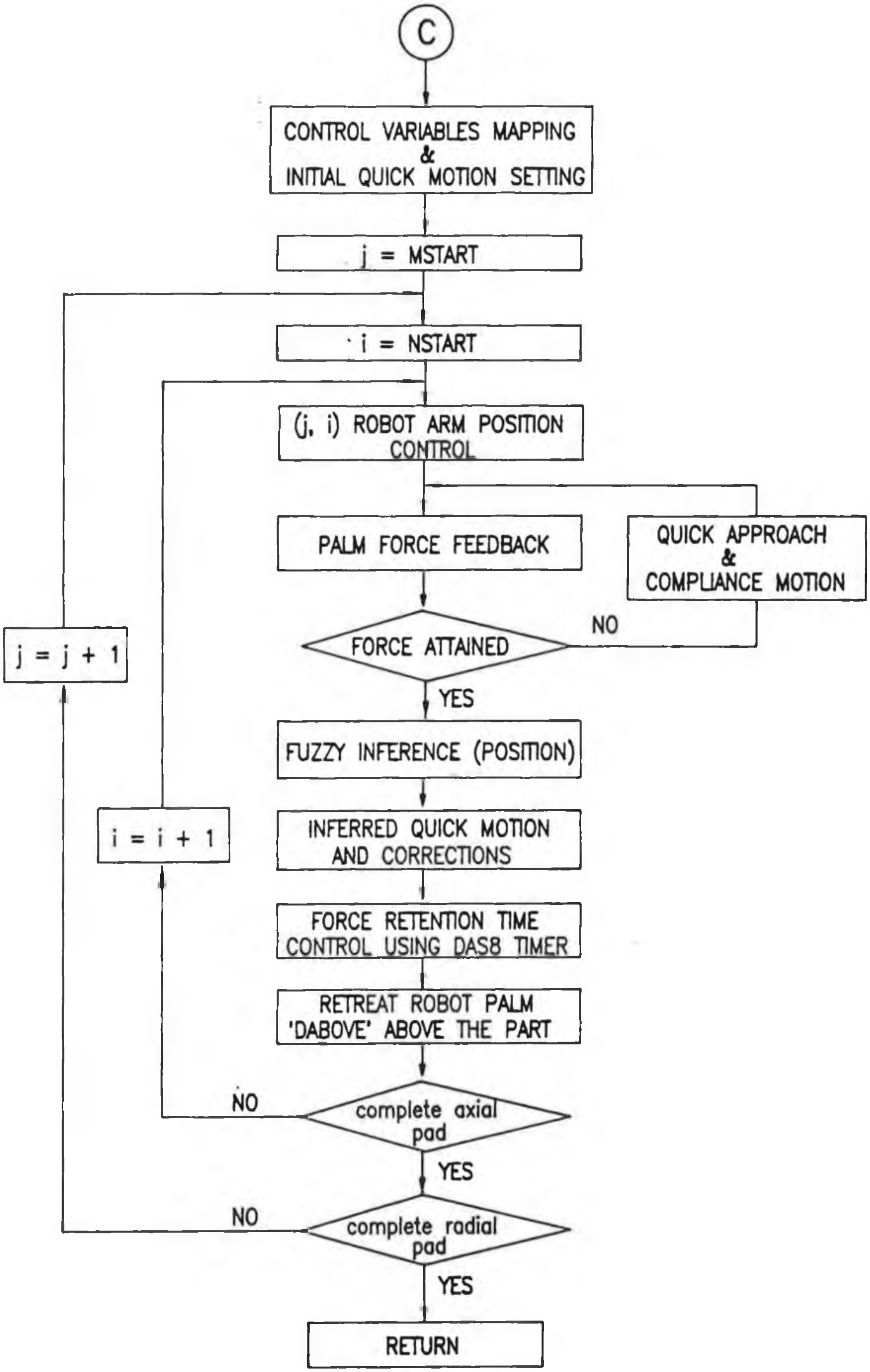
```
40500   '  ** DATA BASE OF PATH NUMBER & POINT **
40510   '
40520   '
40530   '  a. DATA BASE OF MASSAGING POINTS
40540   '
40550   DATA 2,3,4,5,6
40560   '
40570   '
40575   '  b. DATA BASE OF MASSAGING PATH NUMBER
40580   '
40585   DATA 4,6,8,10,12
40590   '
40595   '
40600   '  ** FUZZY MEMBERSHIP FOR ALL FUZZY RELATIONS **
40605   '
40610   DATA  1.0, 0.5,   0,   0,   0,   0,   0,   0,   0
40615   DATA  0.5, 0.5, 0.5, 0.5,   0,   0,   0,   0,   0
40620   DATA    0, 0.5, 1.0, 0.5,   0,   0,   0,   0,   0
40625   DATA    0, 0.5, 0.5, 0.5, 0.5, 0.5,   0,   0,   0
40630   DATA    0,   0,   0, 0.5, 1.0, 0.5,   0,   0,   0
40635   DATA    0,   0,   0, 0.5, 0.5, 0.5, 0.5, 0.5,   0
40640   DATA    0,   0,   0,   0,   0, 0.5, 1.0, 0.5,   0
40645   DATA    0,   0,   0,   0,   0, 0.5, 0.5, 0.5, 0.5
40650   DATA    0,   0,   0,   0,   0,   0,   0, 0.5, 1.0
40655   '
41000   '
42000   END
```

APPENDIX F-2 Intelligent control software for robot system using HAND-I -- EXPERTO.BAS




A
INPUT DATA FILE NAME
RE-INPUT DATA FILE
CORRECT ?
NO
YES
OPEN DATA FILE
IDENTIFIERS INPUT
CLOSE DATA FILE
WRONG DATA FILE
TASK IDENTIFICATION
YES
KNEADING
NO
PADDING
NO
YES
OPEN DATA FILE
OPEN DATA FILE
FETCH IN:
HAND TYPE,
N, m, FORCE,
SPEED, FTIME,
ORIENTATION, POSITION
FETCH IN:
HAND TYPE,
N, m, FORCE,
SPEED, FTIME,
ORIENTATION, POSITION
CLOSE DATA FILE
CLOSE DATA FILE
RETURN


B
CONTROL VARIABLES MAPPING
i = NSTART
ith ROBOT ARM POSITION CONTROL
ith ROBOT HAND POSITION CONTROL (HAND-I)
FINGER FORCE CONTROL (HAND-I)
FORCE RETENTION TIME CONTROL (TIMER OF DAS8)
POSITION/FORCE FEEDBACK
Error > Threshold
NO
YES
FUZZY INFERENCE FOR KNEADING
FUZZY INFERRED OUTPUT (DPX, DPY, DPZ)
ERROR-CORRECTION
i = i + 1
NO
COMPLETE ?
YES
RETURN


C
CONTROL VARIABLES MAPPING & INITIAL QUICK MOTION SETTING
j = MSTART
i = NSTART
(j, i) ROBOT ARM POSITION CONTROL
PALM FORCE FEEDBACK
QUICK APPROACH & COMPLIANCE MOTION
FORCE ATTAINED
NO
YES
j = j + 1
FUZZY INFERENCE (POSITION)
i = i + 1
INFERRED QUICK MOTION AND CORRECTIONS
FORCE RETENTION TIME CONTROL USING DAS8 TIMER
RETREAT ROBOT PALM 'DABOVE' ABOVE THE PART
NO
complete axial pad
YES
NO
complete radial pad
YES
RETURN

```
1000    '
1500    ' *************************************************
1510    ' *                                               *
1520    ' *     EXPERT SYSTEM FOR PHYSIOTHERAPIC ROBOT    *
1530    ' *                                               *
1535    ' *      ** Intelligent control software **       *
1540    ' *       %%  FOR ROBOT USING HAND-I  %%          *
1545    ' *                                               *
1550    ' *      a. PARAMETER ORAGNIZING & DATA LOAD      *
1554    ' *      b. TASK EXECUTION WITH INTELLIGENCE      *
1555    ' *      c. ON-LINE KB FOR INTELLIGENT CONTROL    *
1560    ' *      d. FUZZY LOGIC FOR ERROR-CORRECTING      *
1565    ' *                                               *
1568    ' *-----------------------------------------------*
1569    ' *                                               *
1570    ' *          FILE NAME  -->  EXPERTO.BAS          *
1572    ' *                                               *
1575    ' *               EDITED BY J. YAN                *
1576    ' *                                               *
1580    ' *            DUBLIN CITY UNIVERSITY             *
1582    ' *                                               *
1585    ' *************************************************
1590    '
1600    '
1610    '  %*******  DIMENSION SECTION  ********%
1620    '
1630    '  **  Comman buffer  **
1640    '
1650    '  OO(3,3)       -- Robot arm orientation
1660    '  PP(3)         -- Robot arm position
1665    '  QQ(5)         -- Robot arm joint angles
1670    '
1680    DIM OO(3,3),PP(3),QQ(5)
1685    '
1690    '
1695    '  **  Robot finger space  **
1700    '
1710    '  QF(30)        -- Finger openning angles
1715    '  CPZ(30)       -- Compliance grasping distance
1720    '
1730    DIM QF(30),CPZ(30)
1735    '
1740    '
1745    '  **  Kneading space  **
1750    '
1755    '  XKT(30)       -- Task position along X axis
1760    '  YKT(30)       -- Task position along Y axis
1762    '  ZKT(30)       -- Task position along Z axis
1765    '  XKP(30)       -- Robot arm X control position
1767    '  YKP(30)       -- Robot arm Y control position
1770    '  ZKP(30)       -- Robot arm Z control position
1775    '  QKP(30)       -- Robot pitch control angle
1780    '  QKR(30)       -- Robot  roll control angle
1782    '
```

```
1785    DIM XKT(30),YKT(30),ZKT(30)
1790    DIM XKP(30),YKP(30),ZKP(30),QKP(30),QKR(30)
1792    '
1795    '
1800    '   **  Padding space  **
1810    '
1815    '   XPT(10,30)    -- Task position along X axis
1820    '   YPT(10,30)    -- Task position along Y axis
1825    '   ZPT(10,30)    -- Task position along Z axis
1830    '   XPP(10,30)    -- Robot arm X control position
1832    '   YPP(10,30)    -- Robot arm Y control position
1835    '   ZPP(10,30)    -- Robot arm Z control position
1840    '   QPP(10,30)    -- Robot pitch control angle
1845    '   QPR(10,30)    -- Robot  roll control angle
1850    '
1855    DIM XPT(10,30),YPT(10,30),ZPT(10,30)
1860    DIM XPP(10,30),YPP(10,30),ZPP(10,30)
1865    DIM QPP(10,30),QPR(10,30)
1870    '
1900    '   ** Fuzzy inference process **
1905    '
1910    '   SFIRE(7,7)  -- Fire strength for rules
1915    '   YYYK(8,3)   -- Kneading correction output
1920    '   YYYP(8,3)   -- Padding correction output
1924    '   WW(8)       -- Truth value for rule base
1928    '   RULEQ1(8)   -- Truth value in order for EQ1
1930    '   RULEQ2(8)   -- Truth value in order for EQ2
1934    '
1938    DIM SFIRE(7,7),YYYK(8,3),YYYP(8,3),WW(8)
1940    DIM RULEQ1(8),RULEQ2(8)
1942    '
1946    '
1950    '   ** Servo loop dimension **
1952    '
1954    '   DIO%(10)       -- Input/output for DAS8
1958    '   PARRAY%(30)    -- Position sampling array
1960    '   FARRAY%(30)    -- Force sampling array
1968    '   AA(20)         -- Used in feedback of ARM
1970    '   VV(20)         -- Used in feedback of ARM
1974    '
1978    DIM DIO%(10),PARRAY%(30),FARRAY%(30)
1980    DIM AA(20),VV(20)
2000    '
3000    ' ***********************************************
3005    ' *                                             *
3010    ' *         PC - ROBOT COMMUNICATION SETTING    *
3020    ' *                                             *
3030    ' ***********************************************
3040    '
3050    COLOR 7,1:CLS
3060    LOCATE 10,20:COLOR 20,2
3070    PRINT". PLEASE SWITCH ON THE ROBOT DRIVE UNIT ."
3080    LOCATE 11,20:COLOR 2,4
3090    PRINT"  Set the robot under control of the PC  "
3095    '
```

```
3100    LOCATE 15,20:COLOR 7,1
3106    PRINT" Press any key when robot is switched on "
3110    '
3120    IF INKEY£="" THEN 3120
3130    '
3135    '
3140    '  ** LOADING A/D BOARD ADDRESS **
3145    '
3150    OPEN "DAS8.ADR" FOR INPUT AS #1
3155    INPUT #1,BADR%
3160    CLOSE #1
3165    '
3170    DAS8=0
3175    MD%=0
3180    FLAG%=0
3185    CALL DAS8(MD%,BADR%,FLAG%)
3200    '
3210    '
3220    '  ** SETTING D/A BOARD (PORT A AS INPUT) **
3225    '
3235    OUT &H31F,&H9B
3240    '
3245    '
3255    '  ** RELEASE ROBOTIC HAND MOTORS **
3260    '
3265    IL1=0:IL2=0
3270    GOSUB 40400      ' MOTOR #1
3275    GOSUB 40500      ' MOTOR #2
3280    '
3285    '
3290    '
3300    '  ** OPEN COMMUNICATION BUFFER FOR ROBOT **
3310    '
3320    OPEN "COM1:9600,E,7,2,DS60000" AS #2
3330    PRINT #2, "TL 0"          ' TOOL LENGTH
3340    PRINT #2, "NT"            ' GO TO HOME
3350    '
3400    '
3450    '
3500    ' **********************************************
3510    ' *                                            *
3520    ' *          FUZZY TRUTH TABLE LOADING         *
3530    ' *                                            *
3540    ' *         *  FIRE STRENGTH MATRIX            *
3550    ' *         *  KNEADING OUTPUT MATRIX          *
3560    ' *         *  PADDING OUTPUT MATRIX           *
3570    ' *                                            *
3580    ' **********************************************
3590    '
3600    '
3610    GOSUB 37000
3620    '
3630    '
3650    '
```

```
4000    ' ************************************************
4010    ' *                                              *
4020    ' *           EXPERT SYSTEM MAIN MENU            *
4030    ' *                                              *
4040    ' *       *. DATA LOADING    ( KNEAD & PAD )     *
4050    ' *       *  TASK EXECUTION ( KNEAD & PAD )      *
4060    ' *                                              *
4070    ' ************************************************
4080    '
4090    '
4100    COLOR 7,1:CLS
5000    LOCATE 5,20:COLOR 2,4
5020    PRINT"***  MAIN MENU FOR ROBOTIC EXPERT SYSTEM  ***"
5030    '
5040    COLOR 1,7
5050    LOCATE 6,25
5060    PRINT"< 1 > -- DATA  &  PARAMETERS LOADING "
5080    LOCATE 7,25
5090    PRINT"< 2 > -- TASKS EXECUTION USING ROBOT "
5100    LOCATE 8,25
5110    PRINT"< 3 > -- RETURN TO DOS        "
5120    '
5130    COLOR 2,4
5140    LOCATE 10,20
5150    INPUT"Please input your choice [ 1 - 3 ] ";CHY2£
5160    '
5170    IF VAL(CHY2£)=1 THEN 20000     ' DATA LOADING
5180    IF VAL(CHY2£)=2 THEN 25000     ' INTELLIGENT CONTROL
5200    IF VAL(CHY2£)=3 THEN 6000      ' RETURN TO MAIN MENU
5300    '
5400    GOTO 5150
5500    '
5600    '
6000    ' !!!!! RETURN TO DOS WITH PROMPT !!!!!
6010    '
6020    GOSUB 9000                  ' PROMPT BOX FRAME
6040    LOCATE 21,25:COLOR 4,2
6050    PRINT"..EXIT FROM TASK EXECUTION MODULE .."
6080    LOCATE 24,1
6090    END
7000    '
8000    '
9000    ' !!!!!--  PROMPT BOX  --!!!!! (SUBROUTINE)
9010    '
9020    COLOR 7,1:CLS
9030    LOCATE 20,15:COLOR 1,7
9035    PRINT"**********!!!!!      ";
9040    COLOR 20,7:PRINT"PROMPT BOX";
9045    COLOR 1,7:PRINT"      !!!!!***********"
9050    '
9100    LOCATE 22,15:COLOR 1,7
9110    PRINT"********************************";
9115    PRINT"*********************"
9120    COLOR 7,1
9150    RETURN
```

```
10000    '
20000    ' **********************************************
20005    ' *                                            *
20010    ' *        DATA LOADING FOR TASK EXECUTION     *
20020    ' *                                            *
20030    ' **********************************************
20040    '
20050    '
20100    '  aa.   Data file name input    .aa
20105    '
20110    COLOR 7,1:CLS
20115    LOCATE 5,20:COLOR 20,2
20120    PRINT".. DATA LOADING FOR ROBOT CONTROL .."
20125    '
20130    LOCATE 10,15:COLOR 1,7
20135    INPUT"PLEASE INPUT RIGHT DATA FILE NAME ";DFILE£
20140    LOCATE 12,15:COLOR 2,4
20145    PRINT"IS < ";DFILE£;" > THE CORRECT NAME (Y/N) ?"
20150    '
20160    A£=INKEY£
20165    IF A£="Y" OR A£="y" THEN 20200
20170    IF A£="N" OR A£="n" THEN 20110
20180    GOTO 20160
20185    '
20190    '
20200    '  bb. Data file structure judgment .bb
20210    '
20220    OPEN DFILE£ FOR INPUT AS #1
20230    INPUT #1,ACT£
20240    INPUT #1,HAND£
20250    CLOSE #1
20255    '
20260    IF ACT£="KNEAD" OR ACT£="knead" THEN 21000
20270    IF ACT£="PAD"   OR ACT£="pad"   THEN 22000
20275    '
20280    GOSUB 9000
20285    LOCATE 21,23:COLOR 2,4
20290    PRINT".. THE INPUT DATA FILE IS NOT CORRECT .."
20295    GOTO 20100
20300    '
20400    '
20500    '
21000    ' cc. Data loading for Kneading operation .cc
21010    '
21030    '
21050    OPEN DFILE£ FOR INPUT AS #1
21055    '
21060    INPUT #1,ACTK£
21062    INPUT #1,HANDK£
21064    INPUT #1,PNNK
21066    INPUT #1,PMMK
21070    INPUT #1,SPEEDA
21074    INPUT #1,FTIME
21078    INPUT #1,FFORCE
21080    '
```

```
21082    INPUT #1,TNX,TNY,YNZ
21084    INPUT #1,TOX,TOY,TOZ
21090    INPUT #1,TAX,TAY,TAZ
21095    '
21100    '
21105    FOR I=0 TO PNNK
21110    INPUT #1,XKP(I),YKP(I),ZKP(I)
21120    INPUT #1,QKP(I),QKR(I),QF(I)
21130    NEXT I
21135    '
21140    CLOSE #1
21150    '
21160    '
21200    GOSUB 9000
21210    LOCATE 21,22:COLOR 2,4
21220    PRINT"..THE DATA HAVE BEEN LOADED FOR KNEADING.."
21230    '
21240    GOTO 5000         ' GO BACK TO MAIN-MENU
21250    '
21300    '
21400    '
22000    '  dd. Data loading for padding operation .dd
22005    '
22010    '
22020    OPEN DFILE£ FOR INPUT AS #1
22030    '
22040    INPUT #1,ACTP£
22045    INPUT #1,HANDP£
22050    INPUT #1,PNNP
22055    INPUT #1,PMMP
22060    INPUT #1,SPEEDA
22062    INPUT #1,FTIME
22064    INPUT #1,FFORCE
22070    '
22074    INPUT #1,TNX,TNY,TNZ
22076    INPUT #1,TOX,TOY,TOZ
22080    INPUT #1,TAX,TAY,TAZ
22085    '
22090    FOR I=0 TO PMMP
22095    FOR J=0 TO PNNP
22100    INPUT #1,XPP(I,J),YPP(I,J),ZPP(I,J)
22110    INPUT #1,QPP(I,J),QPR(I,J)
22120    NEXT J
22130    NEXT I
22140    '
22150    CLOSE #1
22160    '
22170    '
22200    GOSUB 9000
22210    LOCATE 21,22:COLOR 4,2
22220    PRINT"..THE DATA HAVE BEEN LOADED FOR PADDING.."
22230    '
22240    GOTO 5000         ' GO BACK TO MAIN-MENU
23000    '
23500    '
```

```
24000   '
24500   '
25000   ' ************************************************
25005   ' *                                              *
25010   ' *        TASK EXECUTION & ROBOT CONTROL        *
25015   ' *                                              *
25020   ' *      * INTELLIGENT  PADDING MODULE           *
25025   ' *      * INTELLIGENT KNEADING MODULE           *
25030   ' *      * FUZZY LOGIC INFERENCE                 *
25035   ' *      * INTELLIGENT SENSING FEEDBACK          *
25040   ' *                                              *
25045   ' ************************************************
25050   '
25060   '
25100   '  aa.  TASK TYPE DETECTION FROM DATA FILE  .aa
25110   '
25120   '
25130   IF ACTK£="KNEAD" THEN 26000      ' KNEADING
25140   IF ACTP£="PAD"   THEN 28000      ' PADDING
25150   '
25160   GOSUB 9000
25170   LOCATE 21,23:COLOR 2,4
25180   PRINT"NO DATA IS FOUND, PLEASE INPUT DATA FIRST"
25190   GOTO 5000
25200   '
25210   '
25300   '
26000   ' ************************************************
26002   ' *                                              *
26004   ' *               KNEADING OPERATION             *
26006   ' *                                              *
26010   ' ************************************************
26012   '
26014   '
26020   '  ** Decision-making **
26030   '
26040   COLOR 7,1:CLS
26050   LOCATE 5,20:COLOR 4,2
26055   PRINT"CARRY OUT THE KNEADING OPERATION (Y/N) ? "
26060   A£=INKEY£
26070   IF A£="Y" OR A£="y" THEN 26100
26075   IF A£="N" OR A£="n" THEN 4100   'BACK TO MAIN MENU
26080   GOTO 26060
26100   '
26102   '  ***********************************************
26104   '  *                                             *
26106   '  *  Massaging + direction is referred as the  *
26110   '  *  original specified direction along which  *
26112   '  *  the Robot moves in the beginning.          *
26114   '  *                                             *
26116   '  *  Massaging - direction is referred as the  *
26120   '  *  negitive direction along which the Robot  *
26122   '  *  retreats back to the starting position.   *
26124   '  *                                             *
26126   '  ***********************************************
```

```
26130   '
26132   '
26134   '   **  MOVE ROBOT TO THE WORK POSITION   **
26136   '
26140   '
26142   '    a.  MOTOR TORQUE RELEASE   .a
26144   '
26146   IL1=0:IL2=0
26150   GOSUB 40400
26152   GOSUB 40500
26154   '
26156   '
26160   '    b.  SPEED SETTING FOR ROBOT ARM   .b
26162   '
26164   '
26166   SPD£=STR£(SPEEDA)
26170   PRINT #2,"SP"+SPD£
26172   '
26174   '
26180   '
26200   '    c.  MOVE ROBOT ARM TO WORK POSITION   .c
26202   '
26204   '
26210   XP=CINT(XKP(0))
26212   YP=CINT(YKP(0))
26214   ZP=CINT(ZKP(0))
26216   QP=CINT(QKP(0))
26220   QR=CINT(QKR(0))
26222   '
26224   X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
26230   P£=STR£(QP):R£=STR£(QR)
26232   '
26234   '
26240   GOSUB 40050
26242   '
26244   '
26250   COLOR 7,1:CLS
26252   LOCATE 10,15:COLOR 4,2
26254   PRINT"PRESS ANY KEY TO START THE KNEADING"
26260   IF INKEY£="" THEN 26260
26262   '
26264   '
26270   '
26300   '   ** KNEADING ALONG + MASSAGING DIRECTION **
26310   '
26320   '
26330   '    a.  ROBOT SYSTEM (ARM & HAND) MOTION   .a
26340   '
26350   NSTART=0          ' PATH PARAMETERS
26354   NSTOP=PNNK
26358   NSTEP=1
26360   MSTART=0          ' POINT PARAMETERS
26364   MSTOP=PMMK
26368   MSTEP=1
26370   '
```

```
26375   CMARK=0            ' CONTROL MARK
26378   '
26380   DDX=0:DDY=0:DDZ=0
26385   EQF1=0:EQF2=0
26390   '
26400   COLOR 7,1:CLS
26405   FOR NI=NSTART TO NSTOP STEP NSTEP
26410   '
26420   XP=CINT(XKP(NI)+DDX)
26422   YP=CINT(YKP(NI)+DDY)
26424   ZP=CINT(ZKP(NI)+DDZ)
26426   QP=CINT(QKP(NI))
26428   QR=CINT(QKR(NI))
26430   '
26432   X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
26434   P£=STR£(QP):R£=STR£(QR)
26436   '
26438   '  b.  ROBOT ARM MOTION   .b
26440   '
26450   GOSUB 40050        ' ROBOT ARM MOTION
26455   GOSUB 40200        ' FEEDBACK ARM POSITION
26460   '
26465   '
26470   '  c.  ROBOT FINGER HYBRID CONTROL   .c
26475   '
26480   '
26500   FTIME=FTIME        ' FORCE RETAIN TIME
26505   FD1=FFORCE         ' FORCE #1
26510   FD2=FD1            ' FORCE #2
26515   QD1=QF(NI)-3       ' ANGLE #1
26520   QD2=QD1            ' ANGLE #2
26535   '
26540   '
26545   '  d. KNEADING POINTS REPEAT .d
26550   '
26560   ' FOR JM=MSTART TO MSTOP STEP MSTEP
26570   '
26600   GOSUB 43000        ' FINGER POSITION CONTROL
26605   GOSUB 46000        ' TIME DELAY
26610   GOSUB 44000        ' FINGER FORCE
26615   GOSUB 46000        ' TIME DELAY
26620   GOSUB 41500        ' POSITION INITIALIZE
26630   GOSUB 40600        ' FINGER POSITION FEEDBACK
26640   '
26650   EQF1=QD1-FQF1:EQF2=QD2-FQF2
26655   '
26660   '  e.  RESTORE FINGER POSITION   .e
26665   '
26670   '
26672   QD1=QF(NI)-3:QD2=QD1
26676   GOSUB 43000        ' FINGER POSITION
26680   GOSUB 45000        ' TIME DELAY
26685   '
26690   '  f. ERROR CORRECTION USING FUZZY LOGIC   .f
26700   '
```

```
26710   GOSUB 30000              ' FUZZY INFERENCE
26720   '
26730   DDX=DDX+DPX
26735   DDY=DDY+DPY
26740   DDZ=DDZ+DPZ
26745   '
26750   GOSUB 39000              ' DISPLAY
26755   ' NEXT JM
26760   NEXT NI
26770   '
26780   '
26790   '
26800   '    **. KNEADING ALONG - MASSAGING DIRECTION .**
26810   '
26820   '
26830   '    a.  DECISION-MAKING FOR REPEAT  .a
26835   '
26840   '
26845   COLOR 7,1:CLS
26850   LOCATE 10,20:COLOR 4,2
26855   PRINT"REPEAT THE KNEADING OPERATION (Y/N) ?"
26860   '
26865   A£=INKEY£
26870   IF A£="Y" OR A£="y" THEN 27000
26875   IF A£="N" OR A£="n" THEN 26900
26880   GOTO 26865
26885   '
26890   '
26900   COLOR 7,1:CLS
26905   LOCATE 10,20:COLOR 2,4
26910   PRINT"LET ROBOT GO BACK TO HOME POSITION (Y/N) ?"
26915   A£=INKEY£
26920   IF A£="Y" OR A£="y" THEN 26945
26925   IF A£="N" OR A£="n" THEN 26960
26930   GOTO 26915
26935   '
26940   '
26945   PRINT #2,"NT"
26950   '
26955   '
26960   GOSUB 9000
26965   LOCATE 21,24:COLOR 4,2
26970   PRINT"..KNEADING OPERATION HAS BEEN COMPLETED.."
26975   GOTO 5000
26980   '
26990   '
27000   '  b.  REPEAT KNEADING OPERATION  .b
27010   '
27015   '
27020   IF CMARK=1 THEN 27100
27025   '
27030   NSTART=PNNK
27035   NSTOP=0
27040   NSTEP=-1
27045   MSTART=0
```

```
27050   MSTOP=PMMK
27055   MSTEP=1
27060   '
27065   CMARK=1
27070   '
27080   GOTO 27200
27090   '
27100   '
27110   NSTART=0
27120   NSTOP=PNNK
27130   NSTEP=1
27140   MSTART=0
27150   MSTOP=PMMK
27160   MSTEP=1
27170   '
27180   CMARK=0
27190   '
27200   '
27210   '    c.  INITIATE THE OPERATION   .c
27215   '
27220   COLOR 7,1:CLS
27230   LOCATE 15,20:COLOR 4,2
27240   PRINT"PRESS ANY KEY TO REPEAT KNEADING OPERATION"
27250   '
27260   IF INKEY£="" THEN 27260
27270   '
27280   GOTO  26400
27290   '
27300   '
27310   '
28000   ' ***********************************************
28002   ' *                                             *
28004   ' *                 PADDING OPERATION           *
28006   ' *                                             *
28010   ' ***********************************************
28012   '
28014   '
28016   '    **.  DECISION-MAKING FOR PADDING  .**
28020   '
28022   COLOR 7,1:CLS
28024   LOCATE 5,20:COLOR 4,2
28026   PRINT"CARRY OUT THE PADDING OPERATION (Y/N) ? "
28030   A£=INKEY£
28032   IF A£="Y" OR A£="y" THEN 28050
28034   IF A£="N" OR A£="n" THEN 4100   ' BACK TO MAIN MENU
28036   GOTO 28030
28040   '
28042   '
28044   '   **  MOVE ROBOT TO THE WORK POSITION **
28046   '
28050   '
28052   '    a. INITIALIZE DAS8 FOR PALM FORCE .a
28054   '
28056   GOSUB 41700
28058   '
```

```
28060    '
28062    '    b. SPEED SETTING FOR ROBOT ARM .b
28064    '
28066    SPD£=STR£(SPEEDA)
28070    PRINT #2,"SP 7,H"
28072    '
28074    '
28076    '    c. MOVE ROBOT ARM TO THE WORK POSITION .c
28080    '
28082    '
28084    XP=CINT(XPP(0,0))
28086    YP=CINT(YPP(0,0))
28090    ZP=CINT(ZPP(0,0))
28092    QP=CINT(QPP(0,0))
28094    QR=CINT(QPR(0,0))
28096    '
28100    X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
28102    P£=STR£(QP):R£=STR£(QR)
28104    '
28110    GOSUB 40050
28112    '
28114    COLOR 7,1:CLS
28120    LOCATE 10,15:COLOR 4,2
28122    PRINT"PRESS ANY KEY TO START THE PADDING"
28130    IF INKEY£="" THEN 28130
28135    '
28140    '
28150    '   **.  PADDING ALONG + MASSAGING DIRECTION  .**
28154    '
28160    DDX=0:DDY=0:DDZ=0
28162    '
28164    DABOVE=20          ' PALM ABOVE PART SURFACE
28166    DGRADE=10          ' PALM INITIAL MOTION GRADE
28170    DFUZZC=0           ' PALM QUICK MOTION DISTANCE
28172    '
28174    '
28178    NSTART=0
28180    NSTOP=PNNP
28182    NSTEP=1
28184    MSTART=0
28186    MSTOP=PMMP
28190    MSTEP=1
28192    '
28194    ZZC=0:FDZ=0
28196    '
28200    COLOR 7,1:CLS
28202    FOR JM=MSTART TO MSTOP STEP MSTEP
28206    FOR NI=NSTART TO NSTOP STEP NSTEP
28210    '
28220    EZC=0
28225    XP=CINT(XPP(JM,NI)+DDX)
28230    YP=CINT(YPP(JM,NI)+DDY)
28235    ZP=CINT(ZPP(JM,NI)+DDZ)
28240    QP=CINT(QPP(JM,NI))
28242    QR=CINT(QPR(JM,NI))
```

```
28250   '
28252   '
28255   X£=STR£(XP)
28260   Y£=STR£(YP)
28265   Z£=STR£(ZP)
28270   P£=STR£(QP)
28275   R£=STR£(QR)
28280   '
28290   '
28300   GOSUB 40050      ' ROBOT ARM MOTION EXECUTION
28310   GOSUB 40200      ' FEEDBACK ARM POSITION
28320   '
28325   GOSUB 40900      ' FORCE FEEDBACK
28330   GOSUB 41400      ' FORCE COMPUTING
28334   GOSUB 38000      ' DISPLAY SENSED INFORMATION
28336   '
28340   FLIMIT=0.2
28342   JFDDP=ABS(FDDP-FFORCE)
28344   IF FDDP<FLIMIT  THEN PZD=DGRADE:GOTO 28356
28346   IF FDDP>=FFORCE THEN PZD=-DGRADE/2:GOTO 28400
28348   IF JFDDP<=0.3   THEN PZD=0:GOTO 28400
28350   PZD=DGRADE/2
28354   '
28356   EZC=EZC+PZD                 ' FINE MOTION CONTROL
28360   ZZC=EZC+DFUZZC              ' QUICK APPROACH
28364   '
28366   DXC=ZZC*TAX                 ' MOTION COORDINATING
28370   DYC=ZZC*TAY
28374   DZC=ZZC*TAZ
28378   '
28380   XC=CINT(XP+DXC)
28384   YC=CINT(YP+DYC)
28386   ZC=CINT(ZP+DZC)
28388   X£=STR£(XC)
28390   Y£=STR£(YC)
28392   Z£=STR£(ZC)
28394   '
28396   GOTO 28300                  ' FINE MOTION REPEAT
28398   '
28400   DGRADE=3
28410   EZC=ZZC
28415   '
28420   IF ABS(EZC)>=29 THEN 28460
28425   '
28430   GOSUB 32000                 ' FUZZY INFERENCE
28432   FDZ1=DABOVE*SGN(FDZ)
28435   FDZ=FDZ-FDZ1                ' FUZZY CORRECTION
28440   DFUZZC=(EZC-FDZ)*0.75       ' QUICK APPROACH
28445   GOTO 28480
28450   '
28455   '
28460   FDZ=EZC-DABOVE              ' NON-FUZZY CORRECTION
28465   DFUZZC=DABOVE*0.75          ' QUICK APPROACH
28470   '
28475   '
```

```
28480   DDX=DDX+FDZ*TAX          ' MOTION COORDINATING
28485   DDY=DDY+FDZ*TAY
28490   DDZ=DDZ+FDZ*TAZ
28500   '
28520   MBACK=-DABOVE
28530   XC=CINT(XC+MBACK*TAX)
28535   YC=CINT(YC+MBACK*TAY)
28540   ZC=CINT(ZC+MBACK*TAZ)
28545   '
28550   X£=STR£(XC)
28555   Y£=STR£(YC)
28560   Z£=STR£(ZC)
28565   '
28570   GOSUB 40050
28575   '
28580   NEXT NI
28585   NEXT JM
28590   '
28595   '
28600   '  **.  PADDING ALONG - MASSAGING DIRECTION  .**
28605   '
28610   '
28615   COLOR 7,1:CLS
28620   LOCATE 10,20:COLOR 4,2
28625   PRINT"REPEAT THE PADDING OPERATION (Y/N) ? "
28630   '
28635   A£=INKEY£
28640   IF A£="Y" OR A£="y" THEN 28800
28645   IF A£="N" OR A£="n" THEN 28665
28650   GOTO 28635
28655   '
28660   '
28665   COLOR 7,1:CLS
28670   LOCATE 10,20:COLOR 4,2
28675   PRINT"LET ROBOT GO BACK TO HOME POSITION (Y/N) ?"
28680   '
28685   A£=INKEY£
28690   IF A£="Y" OR A£="y" THEN 28705
28694   IF A£="N" OR A£="n" THEN 28710
28698   GOTO 28685
28700   '
28705   PRINT #2,"NT"
28708   '
28710   GOSUB 9000
28715   LOCATE 21,22:COLOR 4,2
28720   PRINT".. PADDING OPERATION HAS BEEN COMPLETED .."
28725   GOTO 5000
28730   '
28740   '   ** REPEAT PADDING **
28750   '
28760   '
28800   COLOR 7,1:CLS
28810   LOCATE 10,20:COLOR 4,2
28820   PRINT"PRESS ANY KEY TO REPEAT PADDING OPERATION"
28830   '
```

```
28840   IF INKEY£="" THEN 28840
28850   '
28860   GOTO  28200                 ' REPEAT
28870   '
28900   '
28920   '
28930   '
30000   ' %%** On-line error correction for kneading **%%
30005   '
30010   ' ****************** Statement *****************
30011   ' *                                            *
30012   ' *     Subroutine to infer the corrections    *
30014   ' *                                            *
30016   ' *   a. Fuzzification of the error input      *
30018   ' *   b. Fuzzy inference                       *
30020   ' *   c. Defuzzification of inferred output    *
30022   ' *   d. Computation of correction distance    *
30026   ' *                                            *
30028   ' **********************************************
30030   '
30040   '
30050   '  aa. Judge if correction is required .aa
30055   '
30060   '
30065   IF ABS(EQF1)>9 OR ABS(EQF2)>9 THEN 30200
30070   '
30075   '  ** NO CORRECTION REQUIRED **
30078   '
30080   DPX=0:DPY=0:DPZ=0
30090   RETURN
30095   '
30100   '
30200   '  bb. Fuzzfication of error input .bb
30205   '
30210   '  ** Fuzzy scaler **
30215   '
30220   FKKQ=3            ' Degree
30225   '
30330   '  ** FUZZIFYING INPUTS **
30332   '
30334   ' FOR FINGER #1
30338   '
30340   FXX=EQF1
30344   SCALE=FKKQ
30346   GOSUB 36000
30350   QFU1£=FUZ£                 ' FIRED TERM
30352   NFIRE1=FFIRE               ' FIRE STRENGTH TERM
30355   '
30360   ' FOR FINGER #2
30365   '
30370   FXX=EQF2
30375   SCALE=FKKQ
30380   GOSUB 36000
30385   QFU2£=FUZ£                 ' FIRED TERM
30390   NFIRE2=FFIRE               ' FIRE STRENGTH TERM
```

```
30400   '
30450   '
30500   '  cc. Truth for control rules (Knead) .cc
30505   '
30510   '  WW(8) -- Truth value for kneading rule base
30520   '
30530   '   For EQ1 input, the fire strength vector is:
30535   '   SFIRE(NFIRE1,J)  where J=0 to 6
30542   '
30544   '   For EQ2 input, the fire strength vector is:
30546   '   SFIRE(NFIRE2,J)  where J=0 to 6
30550   '
30554   '   The order of the fire strength in the control
30558   '   rule base should be organized as follows:
30564   '
30566   '          | SFIRE(NFIRE1,1), SFIRE(NFIRE2,5) |
30568   '          | SFIRE(NFIRE1,0), SFIRE(NFIRE2,6) |
30570   '          | SFIRE(NFIRE1,5), SFIRE(NFIRE2,1) |
30572   '          | SFIRE(NFIRE1,6), SFIRE(NFIRE2,0) |
30576   '   min    | SFIRE(NFIRE1,1), SFIRE(NFIRE2,1) |
30578   '          | SFIRE(NFIRE1,0), SFIRE(NFIRE2,0) |
30580   '          | SFIRE(NFIRE1,5), SFIRE(NFIRE2,5) |
30582   '          | SFIRE(NFIRE1,6), SFIRE(NFIRE2,6) |
30584   '
30590   '
30600   RULEQ1(0)=SFIRE(NFIRE1,1):RULEQ2(0)=SFIRE(NFIRE2,5)
30605   RULEQ1(1)=SFIRE(NFIRE1,0):RULEQ2(1)=SFIRE(NFIRE2,6)
30610   RULEQ1(2)=SFIRE(NFIRE1,5):RULEQ2(2)=SFIRE(NFIRE2,1)
30615   RULEQ1(3)=SFIRE(NFIRE1,6):RULEQ2(3)=SFIRE(NFIRE2,0)
30620   RULEQ1(4)=SFIRE(NFIRE1,1):RULEQ2(4)=SFIRE(NFIRE2,1)
30625   RULEQ1(5)=SFIRE(NFIRE1,0):RULEQ2(5)=SFIRE(NFIRE2,0)
30630   RULEQ1(6)=SFIRE(NFIRE1,5):RULEQ2(6)=SFIRE(NFIRE2,5)
30635   RULEQ1(7)=SFIRE(NFIRE1,6):RULEQ2(7)=SFIRE(NFIRE2,6)
30640   '
30645   '
30650   FOR I=0 TO 7
30655   IF RULEQ1(I)<RULEQ2(I) THEN 30670
30660   WW(I)=RULEQ2(I)
30665   GOTO 30675
30670   WW(I)=RULEQ1(I)
30675   NEXT I
30680   '
30685   '
30690   '
30700   ' dd. fuzzy inference process .dd
30705   '
30710   '
30712   '  ** Defuzzy scaler **
30714   '
30718   FKK0=4            ' mm
30720   '
30722   '  ** Fuzzy inference based on fuzzy logic **
30724   '
30725   EPX=0:EPY=0:EPZ=0
30728   WWD=0
```

```
30730    '
30740    FOR I=0 TO 7
30745    WWD=WWD+WW(I)
30750    EPX=EPX+WW(I)*YYYK(I,0)
30755    EPY=EPY+WW(I)*YYYK(I,1)
30760    EPZ=EPZ+WW(I)*YYYK(I,2)
30765    NEXT I
30770    '
30772    IF WWD=0 THEN  30790
30776    EPX=EPX/WWD*FKK0
30780    EPY=EPY/WWD*FKK0
30782    EPZ=EPZ/WWD*FKK0
30784    GOTO  30820
30786    '
30790    EPX=0:EPY=0:EPZ=0
30792    '
30794    '
30800    '  ee. Path modifying .ee
30810    '
30815    '
30820    DPX=EPX*TNX+EPY*TOX+EPZ*TAX
30830    DPY=EPX*TNY+EPY*TOY+EPZ*TAY
30835    DPZ=EPX*TNZ+EPY*TOZ+EPZ*TAZ
30850    '
30900    RETURN
31000    '
31500    '
32000    '  %%** On-line error correction for padding **%%
32010    '
32015    '  ****************  Statement  *****************
32020    '  *                                             *
32025    '  *    Subroutine to infer the corrections      *
32030    '  *                                             *
32032    '  *    a. Fuzzification of the error input      *
32034    '  *    b. Fuzzy inference                       *
32040    '  *    d. Defuzzification of inferred output    *
32044    '  *    e. Computation of correction distance    *
32048    '  *                                             *
32050    '  ***********************************************
32060    '
32105
32110    '  bb. Fuzzification of error input .bb
32115    '
32120    '  **  Fuzzy scaler  **
32125    '
32130    FKKD=5           ' mm
32135    '
32140    '  ** FUZZIFYING INPUTS **
32145    '
32150    FXX=EZC
32155    SCALE=FKKD
32160    GOSUB 36000
32165    DFU£=FUZ£                ' FIRED TERM
32170    NFIRE=FFIRE              ' FIRE STRENGTH
32175    '
```

```
32180   '
32185   '  cc.  Truth for control rules (pad)  .cc
32190   '
32195   '  WW(6)          -- Truth value for control rule
32200   '  SFIRE(NFIRE,J) -- Fire strength vector
32204   '
32206   WW(0)=SFIRE(NFIRE,4)
32210   WW(1)=SFIRE(NFIRE,5)
32215   WW(2)=SFIRE(NIFRE,6)
32218   WW(3)=SFIRE(NFIRE,2)
32220   WW(4)=SFIRE(NFIRE,1)
32225   WW(5)=SFIRE(NFIRE,0)
32230   '
32235   '
32240   '  dd. Fuzzy inference process .dd
32245   '
32250   '  **  Defuzzy scaler  **
32255   '
32260   FKK0=5                    ' mm
32265   '
32270   '  ** Fuzzy inference based ob fuzzy logic **
32275   '
32280   FDZ=0
32285   WWD=0
32290   '
32295   FOR I=0 TO 3
32300   WWD=WWD+WW(I)
32325   FDZ=FDZ+WW(I)*YYYP(I,2)
32330   NEXT I
32335   '
32350   FDZ=FDZ/WWD*FKK0
32360   '
32370   RETURN
32380   '
32400   '
35000   '
36000   ' %%** FUZZIFACATION MODULE FOR KNEADING **%%
36005   '
36010   '
36020   '  ** FUZZIFICATION FOR KNEADING **
36030   '
36035   '  FXX    -- CRISP INPUTS
36037   '  SCALE  -- FUZZIFICATION SCALER
36040   '  FUZ£   -- FUZZY LABELS
36045   '
36050   FXX=FXX/SCALE
36060   IF FXX<(-5)                      THEN 36100
36065   IF FXX>=(-5) AND FXX<(-3)        THEN 36110
36070   IF FXX>=(-3) AND FXX<(-1)        THEN 36120
36075   IF FXX>=(-1) AND FXX<=1          THEN 36130
36080   IF FXX>1     AND FXX<=3          THEN 36140
36085   IF FXX>3     AND FXX<=5          THEN 36150
36090   IF FXX>5                         THEN 36160
36095   '
36100   FUZ£="NB":FFIRE=0:RETURN
```

```
36110   FUZ£="NM":FFIRE=1:RETURN
36120   FUZ£="NS":FFIRE=2:RETURN
36130   FUZ£="ZE":FFIRE=3:RETURN
36140   FUZ£="PS":FFIRE=4:RETURN
36150   FUZ£="PM":FFIRE=5:RETURN
36160   FUZ£="PB":FFIRE=6:RETURN
36170   '
36180   '
36800   '
37000   ' %%** DATA LOADING FOR ON-LINE CONTROL **%%
37004   '
37008   '  a. * FIRE STRENGTH TABLE LOADING *
37010   '
37020   '  SFIRE(I,J)  --  FIRE STRENGTH TABLE
37025   '  I   --  No. of fuzzy input terms for QF1 & QF2
37030   '  J   --  No. of fuzzy terms in the Rule base
37035   '
37040   FOR I=0 TO 6
37042   FOR J=0 TO 6
37048   READ SFIRE(I,J)
37050   NEXT J
37052   NEXT I
37055   '
37060   DATA  1.0,  0.3,    0,    0,    0,    0,    0
37062   DATA  0.3,  1.0,  0.3,    0,    0,    0,    0
37068   DATA    0,  0.3,  1.0,  0.3,    0,    0,    0
37070   DATA    0,    0,  0.3,  1.0,  0.3,    0,    0
37075   DATA    0,    0,    0,  0.3,  1.0,  0.3,    0
37080   DATA    0,    0,    0,    0,  0.3,  1.0,  0.3
37085   DATA    0,    0,    0,    0,    0,  0.3,  1.0
37090   '
37095   '
37100   '
37105   '  b. * OUTPUT LOADING FOR KEADING RULE BASE *
37110   '
37112   ' YYYK(8,3) -- OUTPUT TABLE FOR KNEADING
37114   ' YYYK(I,1) -- Ex output
37116   ' YYYK(I,2) -- Ey output
37118   ' YYYK(I,3) -- Ez output
37120   '
37130   FOR I=0 TO 7
37132   FOR J=0 TO 2
37135   READ YYYK(I,J)
37138   NEXT J
37140   NEXT I
37145   '
37148   '
37150   DATA  0, -5,  0
37152   DATA  0, -6,  0
37154   DATA  0,  5,  0
37156   DATA  0,  6,  0
37158   DATA  0,  0,  5
37160   DATA  0,  0,  6
37162   DATA  0,  0, -5
37164   DATA  0,  0, -6
```

```
37170    '
37180    '
37200    '  c. * OUTPUR LOADING FOR PADDING RULE BASE *
37210    '
37215    '  YYYP(6,3)  --  OUTPUT TABLE FOR PADDING
37220    '  YYYP(I,0)  --  EX
37225    '  YYYP(I,1)  --  EY
37230    '  YYYP(I,2)  --  EZ
37235    '
37240    '
37245    FOR I=0 TO 5
37250    FOR J=0 TO 2
37255    READ YYYP(I,J)
37260    NEXT J
37270    NEXT I
37300    '
37310    DATA  0,  0,  3
37320    DATA  0,  0,  4
37330    DATA  0,  0,  5
37335    DATA  0,  0, -3
37340    DATA  0,  0, -4
37350    DATA  0,  0, -5
37355    '
37360    '
37370    RETURN
37380    '
37390    '
37500    '
37600    ' %%** INFORMATION DISPLAY FOR PADDING **%%
37700    '
37800    '
38000    ' ** COMMANDED POSITION & FORCE DISPLAY **
38005    '
38010    '
38020    LOCATE 2,10:COLOR 2,4
38030    PRINT"***** COMMAND  POSITION *****"
38040    COLOR 4,2
38050    LOCATE 4,20: PRINT"PX = ";USING"+####.##";XPP(JM,NI)
38060    LOCATE 5,20: PRINT"PY = ";USING"+####.##";YPP(JM,NI)
38070    LOCATE 6,20: PRINT"PZ = ";USING"+####.##";ZPP(JM,NI)
38080    LOCATE 7,20: PRINT"QP = ";USING"+####.##";QPP(JM,NI)
38090    LOCATE 8,20: PRINT"QR = ";USING"+####.##";QPR(JM,NI)
38100    '
38110    LOCATE 10,10:COLOR 2,4
38120    PRINT"***** COMMAND PAD FORCE *****"
38130    COLOR 4,2
38140    LOCATE 12,20:PRINT"FORCE = ";USING"+##.#";FFORCE
38150    '
38170    '
38180    '
38200    '
38300    ' ** SENSED POSITION & FORCE **
38305    '
38310    '
38320    LOCATE 2,40:COLOR 2,4
```

```
38330   PRINT"***** SENSED  POSITION *****"
38340   COLOR 4,2
38350   LOCATE 4,50:PRINT"PX = ";USING"+####.##";VV(1)
38360   LOCATE 5,50:PRINT"PY = ";USING"+####.##";VV(2)
38370   LOCATE 6,50:PRINT"PZ = ";USING"+####.##";VV(3)
38380   LOCATE 7,50:PRINT"QP = ";USING"+####.##";VV(4)
38390   LOCATE 8,50:PRINT"QR = ";USING"+####.##";VV(5)
38400   '
38410   LOCATE 10,40:COLOR 2,4
38420   PRINT"***** SENSED PAD FORCE  *****"
38430   COLOR 4,2
38440   LOCATE 12,50:PRINT"FORCE = ";USING"+##.#";FDDP
38450   '
38500   '
38550   '
38600   ' ** FUZZY INFERENCE RESULTS **
38605   '
38610   LOCATE 16,10:COLOR 2,4
38620   PRINT"*****  FUZZY INFERENCE  *****"
38630   COLOR 4,2
38640   LOCATE 18,15
38650   PRINT"FUZZY  INPUT = ";USING"+####.##";ZZC-DABOVE
38660   LOCATE 19,15
38670   PRINT"FUZZY OUTPUT = ";USING"+####.##";FDZ
38680   LOCATE 20,15
38690   PRINT"QUICK MOTION = ";USING"+####.##";DFUZZC
38700   '
38710   LOCATE 16,40:COLOR 2,4
38720   PRINT"***** TOTAL CORRECTIONS *****"
38730   COLOR 4,2
38740   LOCATE 18,50:PRINT"DDX = ";USING"+####.##";DDX
38750   LOCATE 19,50:PRINT"DDY = ";USING"+####.##";DDY
38760   LOCATE 20,50:PRINT"DDZ = ";USING"+####.##";DDZ
38770   '
38780   RETURN
38800   '
38850   '
38900   '
38950   ' %%**  INFORMATION DISPLAY FOR KNEADING **%%
38980   '
39000   ' ** COMMAND POSITION **
39005   '
39010   LOCATE 2,10:COLOR 2,4
39020   PRINT"***** COMMAND  POSITION *****"
39025   COLOR 4,2
39030   LOCATE 4,20:PRINT"PX = ";USING"+####.##";XKP(NI)
39040   LOCATE 5,20:PRINT"PY = ";USING"+####.##";YKP(NI)
39050   LOCATE 6,20:PRINT"PZ = ";USING"+####.##";ZKP(NI)
39060   LOCATE 7,20:PRINT"QP = ";USING"+####.##";QKP(NI)
39070   LOCATE 8,20:PRINT"QR = ";USING"+####.##";QKR(NI)
39080   LOCATE 9,20:PRINT"Q1 = ";USING"+####.##";QF(NI)
39090   LOCATE 10,20:PRINT"Q2 = ";USING"+####.##";QF(NI)
39100   '
39105   '
```

```
39110   ' ** SENSED POSITION **
39120   '
39125   LOCATE 2,40:COLOR 2,4
39130   PRINT"*****  SENSED POSITION  *****"
39135   COLOR 4,2
39140   LOCATE 4,50:PRINT"PX = ";USING"+####.##";VV(1)
39145   LOCATE 5,50:PRINT"PY = ";USING"+####.##";VV(2)
39150   LOCATE 6,50:PRINT"PZ = ";USING"+####.##";VV(3)
39160   LOCATE 7,50:PRINT"QP = ";USING"+####.##";VV(4)
39170   LOCATE 8,50:PRINT"QR = ";USING"+####.##";VV(5)
39180   LOCATE 9,50:PRINT"Q1 = ";USING"+####.##";FQF1
39190   LOCATE 10,50:PRINT"Q2 = ";USING"+####.##";FQF2
39200   '
39210   '
39215   ' ** FUZZY INFERENCE **
39220   '
39230   LOCATE 14,10:COLOR 2,4
39240   PRINT"*****   FUZZY  INPUTS   *****"
39250   COLOR 4,2
39260   LOCATE 16,20:PRINT"EQ1 = ";USING"+####.##";EQF1
39270   LOCATE 18,20:PRINT"EQ2 = ";USING"+####.##";EQF2
39280   '
39300   '
39310   LOCATE 14,40:COLOR 2,4
39320   PRINT"*****  FUZZY   OUTPUTS   *****"
39330   COLOR 4,2
39340   LOCATE 16,50:PRINT"EPX = ";USING"+####.##";EPX
39350   LOCATE 17,50:PRINT"EPY = ";USING"+####.##";EPY
39360   LOCATE 18,50:PRINT"EPZ = ";USING"+####.##";EPZ
39370   '
39380   RETURN
39400   '
39420   '
40000   ' %%**  ROBOT ARM POSITION MOTION **%%
40010   '
40050   PRINT #2,"MP"+X£+","+Y£+","+Z£+","+P£+","+R£
40055   RETURN
40060   '
40080   '
40200   ' %%** FEEDBACK OF THE ROBOT ARM POSITION **%%
40208   '
40210   PRINT #2,"WH"
40215   LINE INPUT #2,A£
40220   D£=A£
40224   K=1
40226   FOR I1=1 TO 5
40228   IF I1=5 THEN 40232
40230   AA(I1)=INSTR(K,D£,","):GOTO 40236
40232   AA(I1)=LEN(D£)+1
40236   VV(I1)=VAL(MID£(D£,K,AA(I1)-1))
40238   K=AA(I1)+1
40240   NEXT I1
40250   '
40260   RETURN
40280   '
```

```
40290   '
40300   ' %%**  MICROSWITH DETECTION FETCH **%%
40310   '
40320   '
40330   PIOA%=INP(&H31C)
40340   RETURN
40350   '
40390   '
40400   ' %%** POWER SUPPLY FOR MOTOR #1 (D/A #4) **%%
40405   '
40408   '
40410   VIN1=20.1*IL1/1000
40415   '
40420   DD=2047+INT(204.8*VIN1)          'VOLTS
40425   '
40430   DH%=INT(DD/256)
40440   DL%=DD-DH%*256
40445   OUT &H318,DL%
40450   OUT &H319,DH%
40455   '
40460   RETURN
40465   '
40470   '
40500   ' %%** POWER SUPPLY FOR MOTOR #2 (D/A #5) **%%
40505   '
40510   VIN2=20.1*IL2/1000
40515   '
40520   DD=2047+INT(204.8*VIN2)          'VOLTS
40525   '
40530   DH%=INT(DD/256)
40535   DL%=DD-DH%*256
40540   OUT &H31A,DL%
40545   OUT &H31B,DH%
40550   '
40555   RETURN
40560   '
40570   '
40600   ' %%**  HAND-I POSITION SERVO SAMPLING  **%%
40602   '
40605   MD%=5                             ' MODE 5
40610   DIO%(0)=VARPTR(PARRAY%(0))
40615   DIO%(1)=8
40618   FLAG%=0
40620   CALL DAS8(MD%,DIO%(0),FLAG%)
40625   '
40630   POSIT1=0:POSIT2=0
40635   FOR JJ%=0 TO 3
40640   POSIT1=POSIT1+PARRAY%(2*JJ%)
40645   POSIT2=POSIT2+PARRAY%(2*JJ%+1)
40650   NEXT JJ%
40655   '
40660   POSIT1=POSIT1/4:POSIT2=POSIT2/4
40665   '
40668   KVD=2048/5
40670   FQF1=200/3*(POSIT1/KVD-1.8)
```

```
40675   FQF2=200/3*(POSIT2/KVD-1.8)
40680   '
40685   RETURN
40690   '
40790   '
40800   ' %%** FORCE SAMPLING FOR FINGER SERVO LOOP **%%
40805   '
40808   '
40810   MD%=5                        ' MODE 5
40815   DIO%(0)=VARPTR(FARRAY%(0))
40820   DIO%(1)=16
40825   FLAG%=0
40830   CALL DAS8(MD%,DIO%(0),FLAG%)
40835   '
40840   FORCE1=0:FORCE2=0
40844   FOR JJ%=0 TO 7
40848   FORCE1=FORCE1+FARRAY%(2*JJ%)
40850   FORCE2=FORCE2+FARRAY%(2*JJ%+1)
40855   NEXT JJ%
40860   '
40865   FORCE1=FORCE1/8:FORCE2=FORCE2/8
40870   RETURN
40880   '
40890   '
40900   ' %%** FORCE SAMPLING FOR PALM SERVO LOOP **%%
40905   '
40910   '
40915   MD%=5
40920   DIO%(0)=VARPTR(FARRAY%(0))
40924   DIO%(1)=16
40928   FLAG%=0
40930   CALL DAS8(MD%,DIO%(0),FLAG%)
40934   '
40936   FORCEP=0
40940   '
40944   FOR JJ%=0 TO 15
40946   FORCEP=FORCEP+FARRAY%(JJ%)
40950   NEXT JJ%
40954   '
40956   FORCEP=FORCEP/16
40960   RETURN
40964   '
40965   '
41000   ' %%** CALIBRATION FOR HAND-I F-SENSOR #1 **%%
41005   '      ( FINGER #1 IN HAND-I -- FORCE )
41007   '
41010   KD1=FORCE1*5/2048
41015   NCCN=0.00981
41020   IF KD1>2.3 THEN 41030
41025   FDD1=NCCN*(224+224*(KD1-2.3)/2.27):RETURN
41030   IF KD1>3.41 THEN 41040
41035   FDD1=NCCN*(624+400*(KD1-3.41)/1.11):RETURN
41040   IF KD1>3.74 THEN FDD1=8.1:RETURN
41045   FDD1=NCCN*(824+200*(KD1-3.74)/0.33;
41050   RETURN
```

```
41060   '
41070   '
41100   ' %%** CALIBRATION FOR HAND-I F-SENSOR #2 **%%
41105   '       ( FINGER #2 IN HAND-I -- FORCE )
41110   '
41120   KD2=FORCE2*5/2048
41125   NCCN=0.00981
41130   IF KD2>2.5 THEN 41140
41135   FDD2=NCCN*(224+224*(KD2-2.5)/2.49):RETURN
41140   IF KD2>3.65 THEN 41150
41145   FDD2=NCCN*(524+300*(KD2-3.65)/1.15):RETURN
41150   IF KD2>4.16 THEN FDD2=8.1:RETURN
41155   FDD2=NCCN*(824+300*(KD2-4.16)/0.51)
41160   RETURN
41170   '
41180   '
41400   ' %%** CALIBRATION FOR PALM-I F-SENSOR **%%
41402   '       ( PALM-I IN HAND-I  --  FORCE )
41408   '
41410   KDP=FORCEP*5/2048
41412   NCCN=0.00981
41418   IF KDP>0.63 THEN 41424
41420   FDDP=0:RETURN
41424   IF KDP>2.45 THEN 41430
41428   FDDP=NCCN*(200+200*(KDP-2.45)/1.82):RETURN
41430   IF KDP>3.64 THEN 41436
41432   FDDP=NCCN*(500+300*(KDP-3.64)/1.19):RETURN
41436   IF KDP>4.22 THEN FDDP=8:RETURN
41440   FDDP=NCCN*(800+300*(KDP-4.22)/0.58)
41442   RETURN
41444   '
41446   '
41500   '  %%** INITIALIZE DAS8 FOR POSITION A/D **%%
41505   '        ( FINGER ANGLE DETECTION )
41508   '
41510   MD%=10
41514   DIO%(0)=2
41518   DIO%(1)=3
41520   FLAG%=0
41525   CALL DAS8(MD%,DIO%(0),FLAG%)
41528   '
41530   FREQ=2000                     ' FREQUENCY=Samples/sec
41534   NC2=CINT(6000/FREQ*1000) ' SYSTEM CLOCK = 12 MHZ
41536   '
41538   MD%=11
41540   DIO%(0)=2
41544   DIO%(1)=NC2
41546   FLAG%=0
41548   CALL DAS8(MD%,DIO%(0),FLAG%)
41550   '
41552   MD%=1
41554   DIO%(0)=0
41558   DIO%(1)=1
41560   FLAG%=0
41564   CALL DAS8(MD%,DIO%(0),FLAG%)
```

```
41566   '
41568   MD%=2
41570   CH%=0
41572   FLAG%=0
41574   CALL DAS8(MD%,CH%,FLAG%)
41576   '
41580   RETURN
41585   '
41590   '
41600   ' %%** INITIALIZE DAS8 FOR FORCE A/D **%%
41604   '        ( FINGER FORCE DETECTION )
41608   '
41610   MD%=10
41612   DIO%(0)=2
41616   DIO%(1)=3
41618   FLAG%=0
41620   CALL DAS8(MD%,DIO%(0),FLAG%)
41622   '
41626   FREQ=2000                    ' FREQENCY=Samples/sec
41628   NC2=CINT(6000/FREQ*1000)     ' system clock = 12 MHZ
41630   '
41634   MD%=11
41636   DIO%(0)=2
41638   DIO%(1)=NC2
41640   FLAG%=0
41645   CALL DAS8(MD%,DIO%(0),FLAG%)
41648   '
41650   '
41652   MD%=1
41654   DIO%(0)=2
41656   DIO%(1)=3
41660   FLAG%=0
41664   CALL DAS8(MD%,DIO%(0),FLAG%)
41670   '
41672   MD%=2
41674   CH%=2
41676   FLAG%=0
41678   CALL DAS8(MD%,CH%,FLAG%)
41680   '
41685   RETURN
41690   '
41695   '
41700   ' %%** INITIALIZE DAS8 FOR FORCE A/D **%%
41705   '          ( PALM FORCE DETECTION )
41710   '
41720   MD%=10
41722   DIO%(0)=2
41724   DIO%(1)=3
41728   FALG%=0
41730   CALL DAS8(MD%,DIO%(0),FLAG%)
41732   '
41734   FREQ=1000
41736   NC2=CINT(6000/FREQ*1000)
41740   '
```

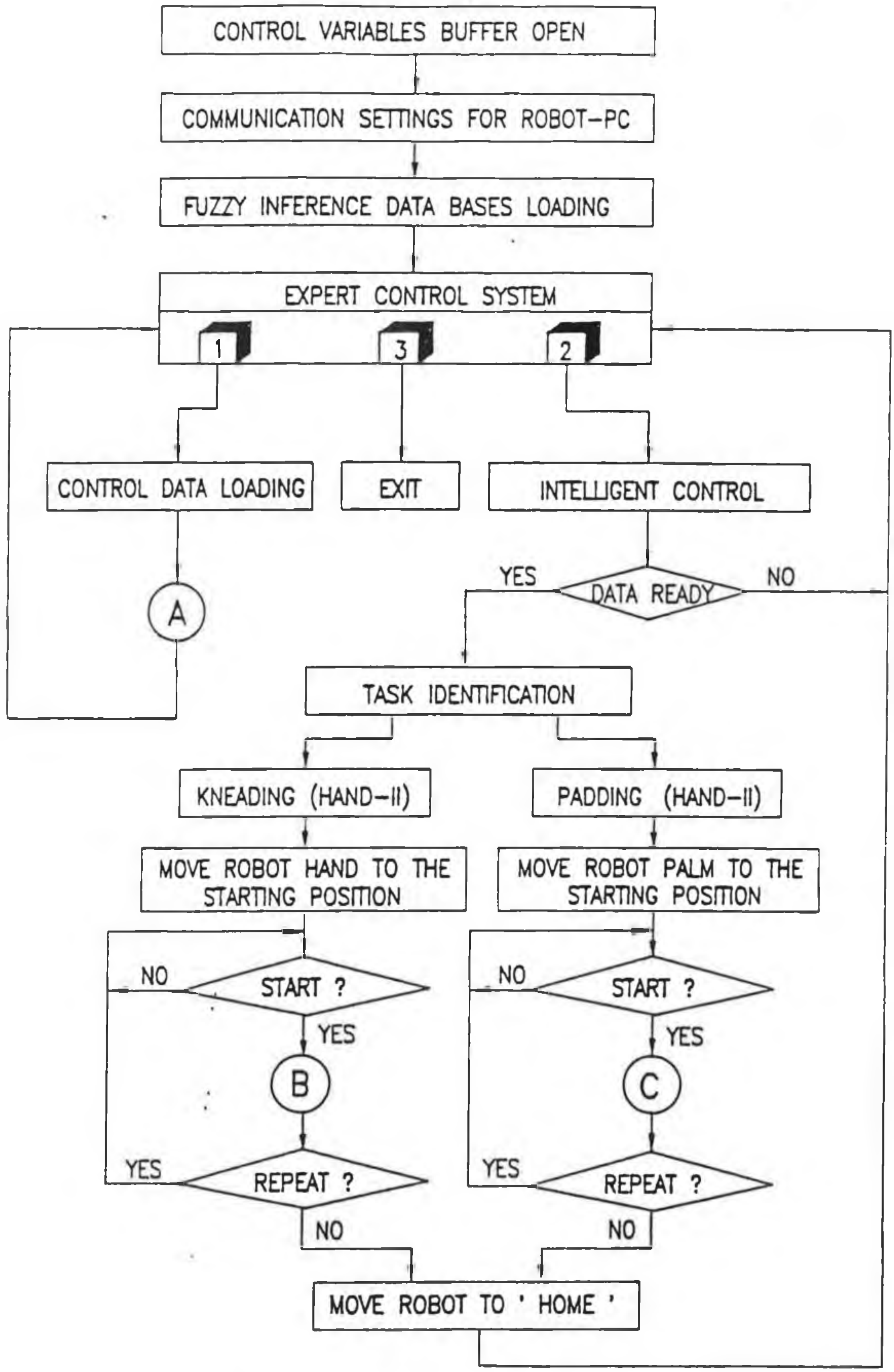
```
41742   MD%=11
41744   DIO%(0)=2
41748   IF NC2<32767 THEN DIO%(1)=NC2:GOTO 41752
41750   DIO%(1)=NC2-65536!
41752   FLAG%=0
41754   CALL DAS8(MD%,DIO%(0),FLAG%)
41758   '
41760   MD%=1
41762   DIO%(0)=4
41764   DIO%(1)=4
41768   FLAG%=0
41770   CALL DAS8(MD%,DIO%(0),FLAG%)
41772   '
41775   MD%=2
41778   CH%=4
41780   FLAG%=0
41782   CALL DAS8(MD%,CH%,FLAG%)
41785   '
41790   RETURN
41800   '
41900   '
42000   ' %%** TIME DELAY USING ROBOT TIMER **%%
42010   '
42020   '
42030   STIME=FTIME/3
42040   STIME=STIME*10
42050   STIME=CINT(STIME)
42060   '
42070   ST£=STR£(STIME)
42080   PRINT #2,"TI"+ST£
42100   '
42110   IL1=8:IL2=8
42120   GOSUB 40400
42130   GOSUB 40500
42140   '
42150   RETURN
42160   '
42170   '
42500   '  %%** TIME DELAY USING PC TIMER **%%
42505   '
42510   FTIME=FTIME
42515   TIME1£=TIME£
42520   TS1=VAL(MID£(TIME1£,7,2))
42525   TM1=VAL(MID£(TIME1£,4,2))
42530   '
42535   TIME2£=TIME£
42540   TS2=VAL(MID£(TIME2£,7,2))
42545   TM2=VAL(MID£(TIME2£,4,2))
42550   '
42555   DTIME=(TS2-TS1)+(TM2-TM1)*60
42560   IF DTIME<FTIME THEN 42510
42570   '
42575   IL1=2:IL2=2
42580   GOSUB 40400
42585   GOSUB 40500
```

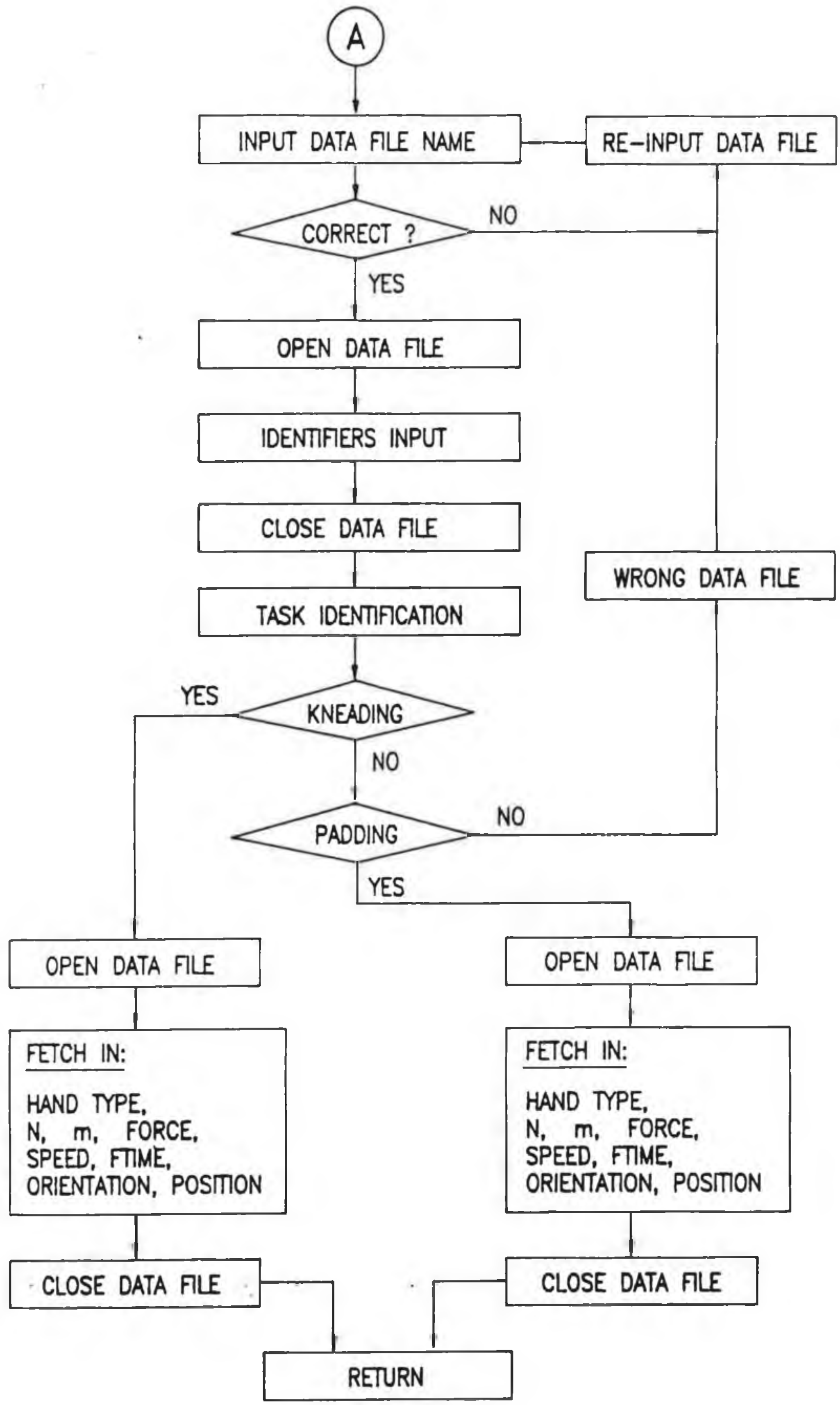
```
42590   '
42600   RETURN
42610   '
42620   '
42630   '
43000   '  %%**  HAND-I POSITION SERVO CONTROL  **%%
43005   '
43010   '
43015   '  ** INITIALIZE DAS8 FOR POSITION SAMPLING
43020   '
43030   GOSUB 41500
43035   '
43040   '  ** COFFICIENCE
43045   '
43050   QLIM=1.0
43055   V01=25/50:V02=25/50
43060   K10=25/10/50
43062   K05=6/5/50
43066   '
43070   '
43075   '  ** FRICTION COMPENSATION
43080   '
43085   GOSUB 40600
43100   '
43110   IF (QD1>30 AND FQF1>30) THEN VF1=2/50:GOTO 43200
43130   VF1=10/50
43150   '
43200   IF (QD2>30 AND FQF2>30)  THEN VF2=2/50:GOTO 43300
43210   VF2=10/50
43220   '
43225   '
43230   '  **  SERVO LOOP CONTROL
43280   '
43290   '
43300   NN=0
43310   '
43320   GOSUB 40600      'POSITION FEEDBACK
43325   '
43330   EQ1=QD1-FQF1:EQ2=QD2-FQF2
43335   S1=SGN(EQ1):S2=SGN(EQ2)
43340   AE1=ABS(EQ1):AE2=ABS(EQ2)
43345   '
43350   '
43400   IF AE1<=QLIM THEN VA1=0:GOTO 43460
43410   IF AE1>20    THEN VA1=VF1+60/50:GOTO 43460
43420   IF AE1>10    THEN VA1=VF1+38/50+K10*AE1:GOTO 43460
43430   IF AE1>5     THEN VA1=VF1+V01+K05*AE1:GOTO 43460
43440   IF AE1>QLIM  THEN VA1=V01
43450   '
43460   VIN1=S1*VA1
43470   GOSUB 40420
43480   '
43490   '
43500   IF AE2<=QLIM THEN VA2=0:GOTO 43560
43510   IF AE2>20    THEN VA2=VF2+60/50:GOTO 43560
```

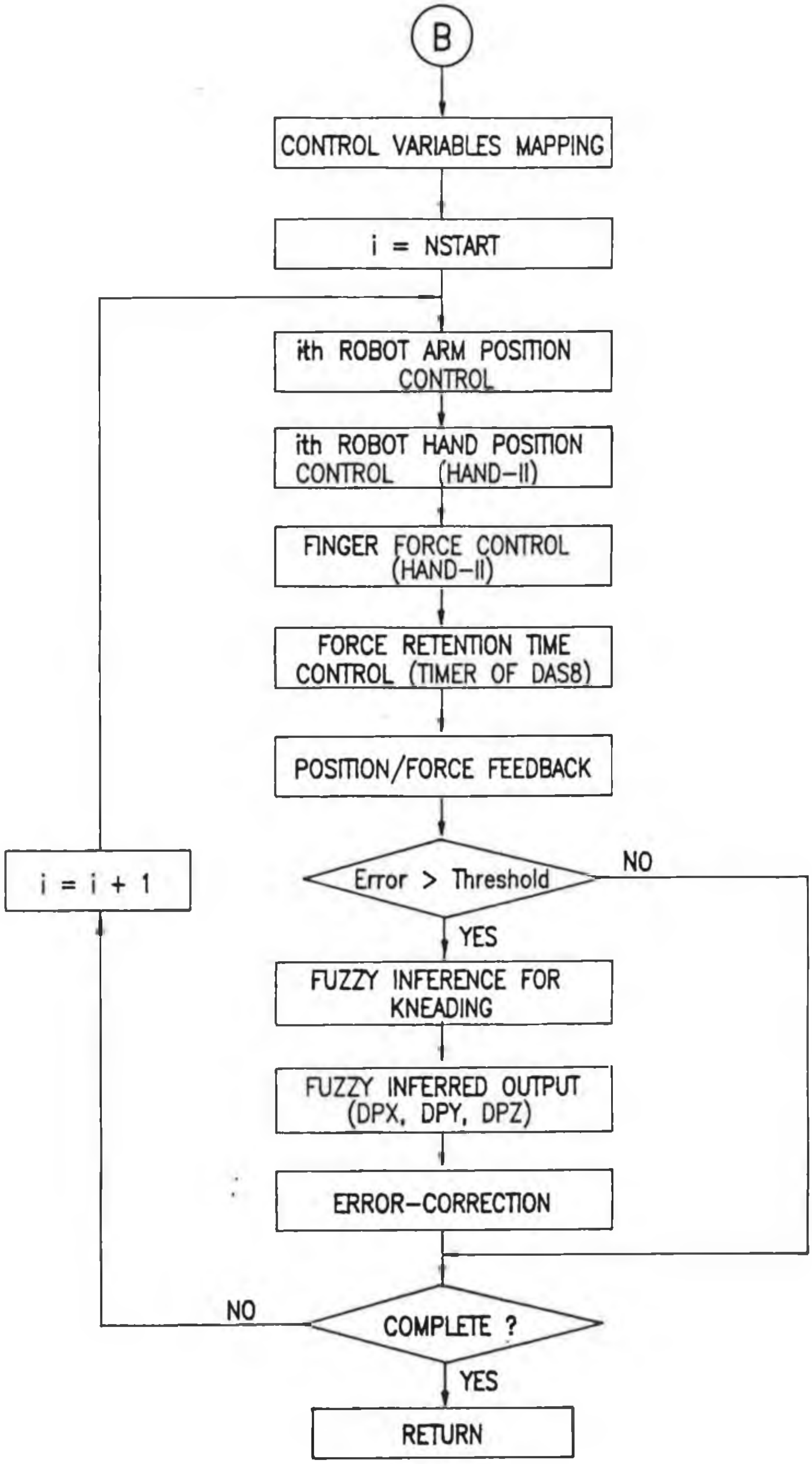
```
43520   IF AE2>10     THEN VA2=VF2+38/50+K10*AE2:GOTO 43560
43530   IF AE2>5      THEN VA2=VF2+V02+K05*AE2:GOTO 43560
43540   IF AE2>QLIM   THEN VA2=V02
43550   '
43560   VIN2=S2*VA2
43570   GOSUB 40520
43580   '
43590   '
43600   IF AE1<=QLIM AND AE2<=QLIM THEN 43650
43605   '
43610   NN=NN+1
43620   IF NN>300 THEN 43650
43630   GOTO 43320
43640   '
43650   VIN1=0:VIN2=0
43660   GOSUB 40420
43670   GOSUB 40520
43680   '
43690   RETURN
43700   '
43710   '
44000   '  %%**  HAND-I FORCE SERVO CONTROL   **%%
44005   '
44010   '
44020   '  ** INITIALIZE DAS8 FOR FORCE SAMPLING
44025   '
44030   '
44035   GOSUB 41600
44040   '
44045   '  ** REQUIRED CURRENT FOR MOTORS
44050   '
44055   LL0=0.115
44060   KVIN0=LL0/0.18144
44065   VF1=KVIN0*FD1
44070   VF2=KVIN0*FD2
44075   '
44080   '  ** INITIAL TOUCH DETECT
44085   '
44090   DIC=0
44100   VIN1=(30+DIC)/50:VIN2=(30+DIC)/50
44105   IF (VIN1>=4 OR VIN2>=4) THEN 44200
44110   GOSUB 40420
44115   GOSUB 40520
44120   '
44125   GOSUB 40800               ' FEEDBACK OF FORCE
44130   GOSUB 41000               ' FORCE #1
44140   GOSUB 41100               ' FORCE #2
44145   '
44150   IF (FDD1>1.5 AND FDD2>1.5) THEN 44200
44155   '
44160   DIC=DIC+4
44170   GOTO 44100
44175   '
44200   '  ** TIME DELAY
44210   '
```

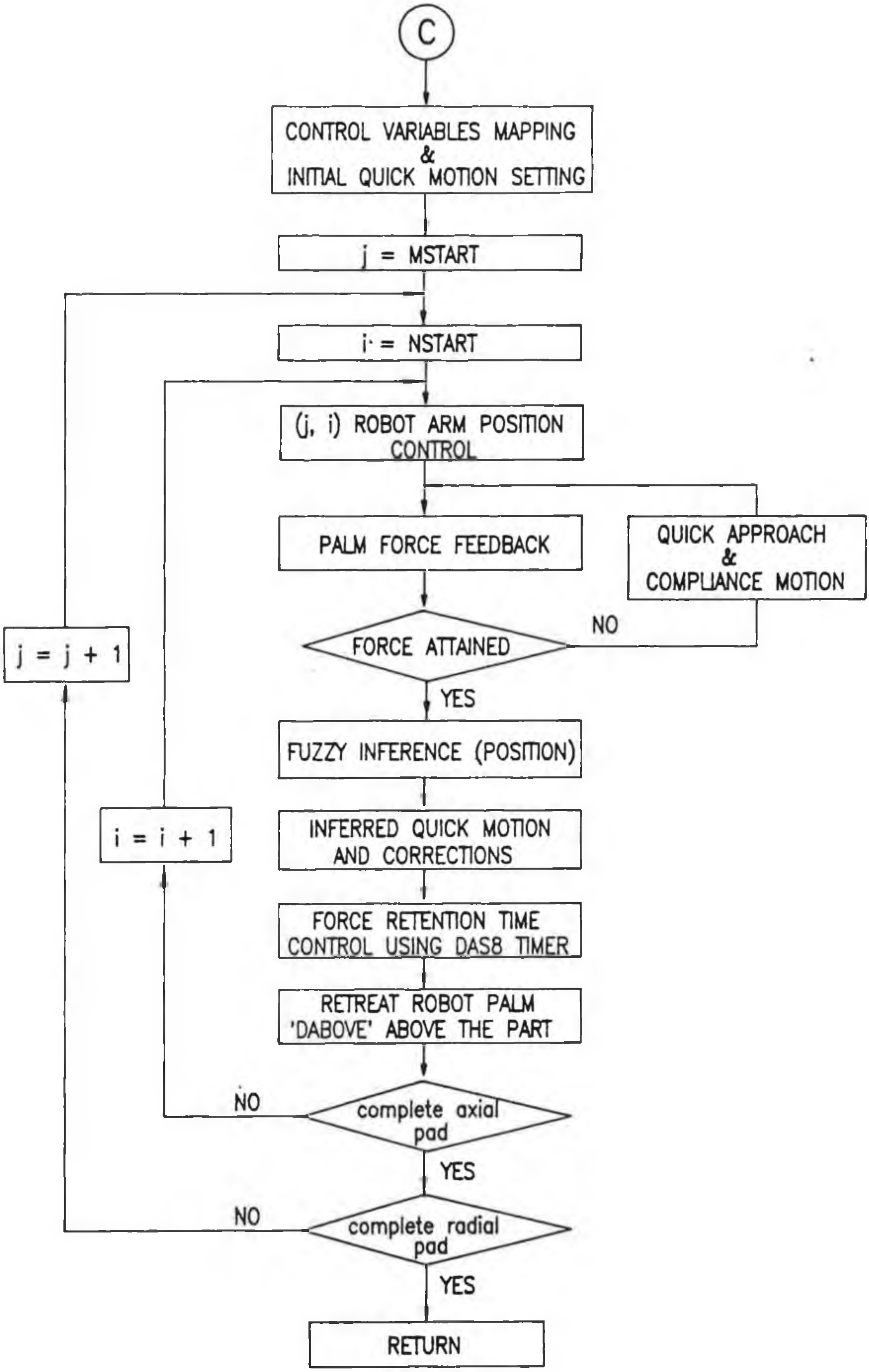
```
44220   GOSUB 45000
44230   '
44235   '
44240   '   ** FORCE CONTROL
44245   '
44250   VIN1=VF1:VIN2=VF2
44255   GOSUB 40420
44260   GOSUB 40520
44280   '
44290   RETURN
44300   '
44310   '
44320   '
45000   '    %%** TIME DELAY FOR FORCE CONTROL **%%
45010   '
45020   '
45030   FOR KW%=0 TO 100
45040   NEXT KW%
45050   RETURN
45100   '
45200   '
46000   '    %%** TIME DELAY FOR HYBRID CONTROL **%%
46010   '
46020   FOR KW%=0 TO 500
46030   NEXT KW%
46035   '
46040   IL1=0:IL2=0
46050   GOSUB 40400
46060   GOSUB 40500
46070   '
46080   RETURN
46100   '
46110   '
50000   ' *************  END OF THE FILE  **************
```

APPENDIX F-3 Intelligent control software for robot system using HAND-II -- EXPERTN.BAS




A
INPUT DATA FILE NAME
RE-INPUT DATA FILE
CORRECT ?
NO
YES
OPEN DATA FILE
IDENTIFIERS INPUT
CLOSE DATA FILE
WRONG DATA FILE
TASK IDENTIFICATION
YES
KNEADING
NO
PADDING
NO
YES
OPEN DATA FILE
OPEN DATA FILE
FETCH IN:
HAND TYPE,
N, m, FORCE,
SPEED, FTIME,
ORIENTATION, POSITION
FETCH IN:
HAND TYPE,
N, m, FORCE,
SPEED, FTIME,
ORIENTATION, POSITION
CLOSE DATA FILE
CLOSE DATA FILE
RETURN


B
CONTROL VARIABLES MAPPING
i = NSTART
ith ROBOT ARM POSITION CONTROL
ith ROBOT HAND POSITION CONTROL (HAND-II)
FINGER FORCE CONTROL (HAND-II)
FORCE RETENTION TIME CONTROL (TIMER OF DAS8)
POSITION/FORCE FEEDBACK
Error > Threshold
NO
YES
i = i + 1
FUZZY INFERENCE FOR KNEADING
FUZZY INFERRED OUTPUT (DPX, DPY, DPZ)
ERROR-CORRECTION
NO
COMPLETE ?
YES
RETURN


C
CONTROL VARIABLES MAPPING & INITIAL QUICK MOTION SETTING
j = MSTART
i = NSTART
(j, i) ROBOT ARM POSITION CONTROL
PALM FORCE FEEDBACK
QUICK APPROACH & COMPLIANCE MOTION
FORCE ATTAINED
NO
YES
j = j + 1
FUZZY INFERENCE (POSITION)
i = i + 1
INFERRED QUICK MOTION AND CORRECTIONS
FORCE RETENTION TIME CONTROL USING DAS8 TIMER
RETREAT ROBOT PALM 'DABOVE' ABOVE THE PART
NO
complete axial pad
YES
NO
complete radial pad
YES
RETURN

```
1500    ' ***************************************************************
1510    ' *                                                             *
1520    ' *      EXPERT SYSTEM FOR PHYSIOTHERAPIC ROBOT                 *
1530    ' *                                                             *
1535    ' *       ** INTELLIGENT CONTROL SOFTWARE **                    *
1540    ' *        %%  FOR ROBOT USING HAND-II  %%                      *
1545    ' *                                                             *
1550    ' *       a. PARAMETER ORAGNIZING & DATA LOAD                   *
1554    ' *       b. TASK EXECUTION WITH INTELLIGENCE                   *
1555    ' *       c. ON-LINE KB FOR INTELLIGENT CONTROL                 *
1560    ' *       d. FUZZY LOGIC FOR ERROR-CORRECTING                   *
1565    ' *                                                             *
1568    ' *-------------------------------------------------------------*
1569    ' *                                                             *
1570    ' *            FILE NAME  -->  EXPERTN.BAS                      *
1572    ' *                                                             *
1575    ' *                 EDITED BY J. YAN                            *
1576    ' *                                                             *
1580    ' *               DUBLIN CITY UNIVERSITY                        *
1582    ' *                                                             *
1585    ' ***************************************************************
1590    '
1600    '
1610    '  %*******  DIMENSION SECTION  ********%
1620    '
1630    '  **  Comman buffer  **
1640    '
1650    '  OO(3,3)       -- Robot arm orientation
1660    '  PP(3)         -- Robot arm position
1665    '  QQ(5)         -- Robot arm joint angles
1670    '
1680    DIM OO(3,3),PP(3),QQ(5)
1685    '
1690    '
1695    '  **  Robot finger space  **
1700    '
1710    '  QF(30)        -- Finger openning angles
1715    '  CPZ(30)       -- Compliance grasping distance
1720    '
1730    DIM QF(30),CPZ(30)
1735    '
1740    '
1745    '  **  Kneading space  **
1750    '
1755    '  XKT(30)       -- Task position along X axis
1760    '  YKT(30)       -- Task position along Y axis
1762    '  ZKT(30)       -- Task position along Z axis
1765    '  XKP(30)       -- Robot arm X control position
1767    '  YKP(30)       -- Robot arm Y control position
1770    '  ZKP(30)       -- Robot arm Z control position
1775    '  QKP(30)       -- Robot pitch control angle
1780    '  QKR(30)       -- Robot  roll control angle
1782    '
```

```
1784    DIM XKT(30),YKT(30),ZKT(30)
1786    DIM XKP(30),YKP(30),ZKP(30),QKP(30),QKR(30)
1788    DIM XKPREC(30),YKPREC(30),ZKPREC(30),QKPREC(30)
1790    DIM QKRREC(30),FKXREC(30),FKYREC(30),FKZREC(30)
1792    DIM EQ1REC(30),EQ2REC(30),QFREC(30)
1794    '
1796    '
1800    '  **  Padding space  **
1810    '
1815    '  XPT(10,30)   -- Task position along X axis
1820    '  YPT(10,30)   -- Task position along Y axis
1825    '  ZPT(10,30)   -- Task position along Z axis
1830    '  XPP(10,30)   -- Robot arm X control position
1832    '  YPP(10,30)   -- Robot arm Y control position
1835    '  ZPP(10,30)   -- Robot arm Z control position
1840    '  QPP(10,30)   -- Robot pitch control angle
1845    '  QPR(10,30)   -- Robot  roll control angle
1850    '
1855    DIM XPT(10,30),YPT(10,30),ZPT(10,30)
1860    DIM XPP(10,30),YPP(10,30),ZPP(10,30)
1865    DIM QPP(10,30),QPR(10,30)
1870    DIM XPPREC(10,30),YPPREC(10,30),ZPPREC(10,30)
1875    DIM QPPREC(10,30),QPRREC(10,30),DZREC(10,30)
1880    DIM FPXREC(10,30),FPYREC(10,30),FPZREC(10,30)
1885    '
1890    '
1900    '  ** Fuzzy inference process **
1905    '
1910    '  SFIRE(7,7)   -- Fire strength for rules
1915    '  YYYK(8,3)    -- Kneading correction output
1920    '  YYYP(8,3)    -- Padding correction output
1924    '  WW(8)        -- Truth value for rule base
1928    '  RULEQ1(8)    -- Truth value in order for EQ1
1930    '  RULEQ2(8)    -- Truth value in order for EQ2
1934    '
1938    DIM SFIRE(7,7),YYYK(8,3),YYYP(8,3),WW(8)
1940    DIM RULEQ1(8),RULEQ2(8)
1942    '
1946    '
1950    '  ** Servo loop dimension **
1952    '
1954    '  DIO%(10)      -- Input/output for DAS8
1958    '  PARRAY%(30)   -- Position sampling array
1960    '  FARRAY%(30)   -- Force sampling array
1962    '  FDELAY%(i)    -- Force generated during delay
1968    '  AA(20)        -- Used in feedback of ARM
1970    '  VV(20)        -- Used in feedback of ARM
1974    '
1980    DIM DIO%(10),PARRAY%(30),FARRAY%(30)
1985    DIM FDELAY%(1100)
1990    DIM AA(20),VV(20)
1995    '
2000    '
2500    '
```

```
3000    ' *********************************************
3005    ' *                                           *
3010    ' *       PC - ROBOT COMMUNICATION SETTING    *
3020    ' *                                           *
3030    ' *********************************************
3040    '
3050    COLOR 7,1:CLS
3060    LOCATE 10,20:COLOR 20,2
3070    PRINT". PLEASE SWITCH ON THE ROBOT DRIVE UNIT ."
3080    LOCATE 11,20:COLOR 2,4
3090    PRINT"  Set the robot under control of the PC  "
3095    '
3100    LOCATE 15,20:COLOR 7,1
3106    PRINT" Press any key when robot is switched on "
3110    '
3120    IF INKEY£="" THEN 3120
3130    '
3135    '
3140    '  ** LOADING A/D BOARD ADDRESS **
3145    '
3150    OPEN "DAS8.ADR" FOR INPUT AS #1
3155    INPUT #1,BADR%
3160    CLOSE #1
3165    '
3170    DAS8=0
3175    MD%=0
3180    FLAG%=0
3185    CALL DAS8(MD%,BADR%,FLAG%)
3200    '
3210    '
3220    '  ** SETTING D/A BOARD (PORT A AS INPUT) **
3225    '
3235    OUT &H31F,&H9B
3240    '
3245    '
3255    '  ** RELEASE ROBOTIC HAND MOTORS **
3260    '
3265    IL1=0:IL2=0
3270    GOSUB 40400       ' MOTOR #1
3275    GOSUB 40500       ' MOTOR #2
3280    '
3285    '
3290    '
3300    '  ** OPEN COMMUNICATION BUFFER FOR ROBOT **
3310    '
3320    OPEN "COM1:9600,E,7,2,DS60000" AS #2
3330    PRINT #2, "TL 0"          ' TOOL LENGTH
3340    PRINT #2, "NT"            ' GO TO HOME
3350    '
3400    '
3450    '
```

```
3500    ' *********************************************
3510    ' *                                           *
3520    ' *          FUZZY TRUTH TABLE LOADING        *
3530    ' *                                           *
3540    ' *         *  FIRE STRENGTH MATRIX           *
3550    ' *         *  KNEADING OUTPUT MATRIX         *
3560    ' *         *  PADDING OUTPUT MATRIX          *
3570    ' *                                           *
3580    ' *********************************************
3590    '
3600    '
3610    GOSUB 37000
3620    '
3630    '
3650    '
4000    ' *********************************************
4010    ' *                                           *
4020    ' *          EXPERT SYSTEM MAIN MENU          *
4030    ' *                                           *
4040    ' *      *. DATA LOADING   ( KNEAD & PAD )    *
4050    ' *      *  TASK EXECUTION ( KNEAD & PAD )    *
4060    ' *                                           *
4070    ' *********************************************
4080    '
4090    '
4100    COLOR 7,1:CLS
5000    LOCATE 5,20:COLOR 2,4
5020    PRINT"***  MAIN MENU FOR ROBOTIC EXPERT SYSTEM  ***"
5030    '
5040    COLOR 1,7
5050    LOCATE 6,25
5060    PRINT"< 1 > -- DATA  &  PARAMETERS LOADING "
5080    LOCATE 7,25
5090    PRINT"< 2 > -- TASKS EXECUTION USING ROBOT "
5100    LOCATE 8,25
5110    PRINT"< 3 > -- RETURN TO DOS        "
5120    '
5130    COLOR 2,4
5140    LOCATE 10,20
5150    INPUT"Please input your choice [ 1 - 3 ] ";CHY2£
5160    '
5170    IF VAL(CHY2£)=1 THEN 20000    ' DATA LOADING
5180    IF VAL(CHY2£)=2 THEN 25000    ' INTELLIGENT CONTROL
5200    IF VAL(CHY2£)=3 THEN 6000     ' RETURN TO MAIN MENU
5300    '
5400    GOTO 5150
5500    '
5600    '
6000    ' !!!!! RETURN TO DOS WITH PROMPT !!!!!
6010    '
6020    GOSUB 9000                ' PROMPT BOX FRAME
6040    LOCATE 21,25:COLOR 4,2
6050    PRINT"..EXIT FROM TASK EXECUTION MODULE .."
6080    LOCATE 24,1
6090    END
```

```
7000     '
8000     '
9000     ' !!!!!--  PROMPT BOX  --!!!!! (SUBROUTINE)
9010     '
9020     COLOR 7,1:CLS
9030     LOCATE 20,15:COLOR 1,7
9035     PRINT"***********!!!!!      ";
9040     COLOR 20,7:PRINT"PROMPT BOX";
9045     COLOR 1,7:PRINT"      !!!!!************"
9050     '
9100     LOCATE 22,15:COLOR 1,7
9110     PRINT"*********************************";
9115     PRINT"**********************"
9120     COLOR 7,1
9150     RETURN
9300     '
9400     '
9500     '
10000    '
20000    ' ***********************************************
20005    ' *                                             *
20010    ' *        DATA LOADING FOR TASK EXECUTION      *
20020    ' *                                             *
20030    ' ***********************************************
20040    '
20050    '
20100    '  aa.   Data file name input   .aa
20105    '
20110    COLOR 7,1:CLS
20115    LOCATE 5,20:COLOR 20,2
20120    PRINT".. DATA LOADING FOR ROBOT CONTROL .."
20125    '
20130    LOCATE 10,15:COLOR 1,7
20135    INPUT"PLEASE INPUT RIGHT DATA FILE NAME ";DFILE£
20140    LOCATE 12,15:COLOR 2,4
20145    PRINT"IS < ";DFILE£;" > THE CORRECT NAME (Y/N) ?"
20150    '
20160    A£=INKEY£
20165    IF A£="Y" OR A£="y" THEN 20200
20170    IF A£="N" OR A£="n" THEN 20110
20180    GOTO 20160
20185    '
20190    '
20200    '  bb. Data file structure judgment .bb
20210    '
20220    OPEN DFILE£ FOR INPUT AS #1
20230    INPUT #1,ACT£
20240    INPUT #1,HAND£
20250    CLOSE #1
20255    '
20260    IF ACT£="KNEAD" OR ACT£="knead" THEN 21000
20270    IF ACT£="PAD"   OR ACT£="pad"   THEN 22000
20275    '
20280    GOSUB 9000
20285    LOCATE 21,23:COLOR 2,4
```

```
20290   PRINT".. THE INPUT DATA FILE IS NOT CORRECT .."
20295   GOTO 20100
20300   '
20400   '
20500   '
21000   ' cc. Data loading for Kneading operation .cc
21010   '
21030   '
21050   OPEN DFILE£ FOR INPUT AS #1
21055   '
21060   INPUT #1,ACTK£
21062   INPUT #1,HANDK£
21064   INPUT #1,PNNK
21066   INPUT #1,PMMK
21070   INPUT #1,SPEEDA
21074   INPUT #1,FTIME
21078   INPUT #1,FFORCE
21080   '
21082   INPUT #1,TNX,TNY,YNZ
21084   INPUT #1,TOX,TOY,TOZ
21090   INPUT #1,TAX,TAY,TAZ
21095   '
21100   '
21105   FOR I=0 TO PNNK
21110   INPUT #1,XKP(I),YKP(I),ZKP(I)
21120   INPUT #1,QKP(I),QKR(I),QF(I)
21130   NEXT I
21135   '
21140   CLOSE #1
21150   '
21160   '
21200   GOSUB 9000
21210   LOCATE 21,22:COLOR 2,4
21220   PRINT"..THE DATA HAVE BEEN LOADED FOR KNEADING.."
21230   '
21240   GOTO 5000        ' GO BACK TO MAIN-MENU
21250   '
21300   '
21400   '
22000   '  dd. Data loading for padding operation .dd
22005   '
22010   '
22020   OPEN DFILE£ FOR INPUT AS #1
22030   '
22040   INPUT #1,ACTP£
22045   INPUT #1,HANDP£
22050   INPUT #1,PNNP
22055   INPUT #1,PMMP
22060   INPUT #1,SPEEDA
22062   INPUT #1,FTIME
22064   INPUT #1,FFORCE
22070   '
22074   INPUT #1,TNX,TNY,TNZ
22076   INPUT #1,TOX,TOY,TOZ
22080   INPUT #1,TAX,TAY,TAZ
```

```
22085  '
22090  FOR I=0 TO PMMP
22095  FOR J=0 TO PNNP
22100  INPUT #1,XPP(I,J),YPP(I,J),ZPP(I,J)
22110  INPUT #1,QPP(I,J),QPR(I,J)
22120  NEXT J
22130  NEXT I
22140  '
22150  CLOSE #1
22160  '
22170  '
22200  GOSUB 9000
22210  LOCATE 21,22:COLOR 4,2
22220  PRINT"..THE DATA HAVE BEEN LOADED FOR PADDING.."
22230  '
22240  GOTO 5000        ' GO BACK TO MAIN-MENU
23000  '
23500  '
24000  '
24500  '
25000  ' ********************************************
25005  ' *                                          *
25010  ' *       TASK EXECUTION & ROBOT CONTROL     *
25015  ' *                                          *
25020  ' *     * INTELLIGENT  PADDING MODULE        *
25025  ' *     * INTELLIGENT KNEADING MODULE        *
25030  ' *     * FUZZY LOGIC INFERENCE              *
25035  ' *     * INTELLIGENT SENSING FEEDBACK       *
25040  ' *                                          *
25045  ' ********************************************
25050  '
25060  '
25100  '  **  INITIAL SETTING  **
25105  '
25110  '  aa.  MOTOR TORQUE RELEASE  .aa
25115  '
25120  IL1=0:IL2=0
25125  GOSUB 40400
25130  GOSUB 40500
25135  '
25140  '
25145  '  bb. SPEED SETTING FOR ROBOT ARM .bb
25150  '
25160  SPD£=STR£(SPEEDA)
25170  PRINT #2,"SP"+SPD£
25180  '
25190  '
25200  '  cc. TASK TYPE DETECTION FROM INPUT DATA .cc
25210  '
25220  '
25230  IF ACTK£="KNEAD" THEN 26000      ' KNEADING
25240  IF ACTP£="PAD"   THEN 28000      ' PADDING
25250  '
25260  GOSUB 9000
25270  LOCATE 21,23:COLOR 2,4
```

```
25280    PRINT"NO DATA IS FOUND, PLEASE INPUT DATA FIRST"
25290    GOTO 5000
25300    '
25400    '
25500    '
26000    ' ********************************************
26002    ' *                                          *
26004    ' *             KNEADING OPERATION           *
26006    ' *                                          *
26008    ' ********************************************
26010    '
26015    '
26020    '
26022    '  **  DECISION-MAKING FOR KNEADING  **
26024    '
26030    COLOR 7,1:CLS
26035    LOCATE 5,20:COLOR 4,2
26040    PRINT"MOVE ROBOT TO KNEADING START POSITION (Y/N)?"
26045    A£=INKEY£
26050    IF A£="Y" OR A£="y" THEN 26100
26055    IF A£="N" OR A£="n" THEN 4100   'BACK TO MAIN MENU
26060    GOTO 26045
26065    '
26070    '
26100    '  ** MOVE ROBOT TO KNEADING START POSITION **
26105    '
26110    '
26115    XP=CINT(XKP(0))
26120    YP=CINT(YKP(0))
26125    ZP=CINT(ZKP(0))
26130    QP=CINT(QKP(0))
26135    QR=CINT(QKR(0))
26140    '
26145    X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
26150    P£=STR£(QP):R£=STR£(QR)
26155    '
26160    GOSUB 40050
26165    '
26170    COLOR 7,1:CLS
26175    LOCATE 15,20:COLOR 2,4
26180    PRINT"PRESS ANY KEY TO START KNEADING OPERATION"
26185    '
26190    IF INKEY£="" THEN 26190
26210    '
26215    '  ******************************************
26220    '  *                                        *
26225    '  *  Massaging + direction is referred as the  *
26230    '  *  original specified direction along which  *
26245    '  *  the Robot moves in the beginning.     *
26250    '  *                                        *
26255    '  *  Massaging - direction is referred as the  *
26260    '  *  negitive direction along which the Robot  *
26265    '  *  retreats back to the starting position.   *
26270    '  *                                        *
26275    '  ******************************************
```

```
26295   '
26300   '   ** KNEADING ALONG + MASSAGING DIRECTION **
26310   '
26320   '
26330   NSTART=0          ' PATH PARAMETERS
26335   NSTOP=PNNK
26340   NSTEP=1
26345   MSTART=0          ' POINT PARAMETERS
26350   MSTOP=PMMK
26355   MSTEP=1
26360   '
26365   CMARK=0           ' CONTROL MARK
26370   '
26380   DDX=0:DDY=0:DDZ=0
26385   EQF1=0:EQF2=0
26390   '
26395   '
26400   COLOR 7,1:CLS
26405   FOR NI=NSTART TO NSTOP STEP NSTEP
26410   '
26420   XP=CINT(XKP(NI)+DDX):XKPREC(NI)=XP
26422   YP=CINT(YKP(NI)+DDY):YKPREC(NI)=YP
26424   ZP=CINT(ZKP(NI)+DDZ):ZKPREC(NI)=ZP
26426   QP=CINT(QKP(NI)):QKPREC(NI)=QP
26428   QR=CINT(QKR(NI)):QKRREC(NI)=QR
26430   '
26432   X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
26434   P£=STR£(QP):R£=STR£(QR)
26436   '
26438   '  *.  ROBOT ARM MOTION   .*
26440   '
26450   GOSUB 40050       ' ROBOT ARM MOTION
26455   GOSUB 40200       ' FEEDBACK ARM POSITION
26460   '
26470   '  *.  ROBOT FINGER HYBRID CONTROL  .*
26480   '
26490   FTIME=FTIME
26500   FD1=FFORCE        ' FORCE #1
26510   FD2=FD1           ' FORCE #2
26515   QD1=QF(NI)        ' ANGLE #1
26520   QD2=QD1           ' ANGLE #2
26525   '
26530   QFREC(NI)=QD1
26535   '
26540   '
26545   '  *. KNEADING POINTS REPEAT .*
26550   '
26560   ' FOR JM=MSTART TO MSTOP STEP MSTEP
26570   '
26600   GOSUB 43500       ' FINGER POSITION CONTROL
26605   GOSUB 46000       ' TIME DELAY
26610   GOSUB 44500       ' FINGER FORCE
26615   GOSUB 42000       ' FORCE RETENTION TIME
26620   GOSUB 41500       ' POSITION INITIALIZE
26630   GOSUB 40700       ' FINGER POSITION FEEDBACK
```

```
26640    '
26650    EQF1=QD1-FQF1:EQF2=QD2-FQF2
26655    EQ1REC(NI)=EQF1:EQ2REC(NI)=EQF2
26658    '
26660    '  *.  RESTORE FINGER POSITION  .*
26665    '
26670    '
26672    QD1=QF(NI):QD2=QD1
26676    GOSUB 43500      ' FINGER POSITION
26680    GOSUB 46000      ' TIME DELAY
26685    '
26690    '  *. ERROR CORRECTION USING FUZZY LOGIC  .*
26700    '
26710    GOSUB 30000                 ' FUZZY INFERENCE
26720    '
26722    FKXREC(NI)=DPX
26726    FKYREC(NI)=DPY
26730    FKZREC(NI)=DPZ
26736    '
26738    DDX=DDX+DPX
26740    DDY=DDY+DPY
26744    DDZ=DDZ+DPZ
26748    '
26750    GOSUB 39000                 ' DISPLAY
26755    ' NEXT JM
26760    NEXT NI
26770    '
26780    '
26790    '
26800    '  ** RESULTS RECORDING **
26805    '
26810    GOSUB 47000
26820    '
26825    '   **. KNEADING ALONG - MASSAGING DIRECTION .**
26830    '
26835    '   *.  DECISION-MAKING FOR REPEAT  .*
26840    '
26845    COLOR 7,1:CLS
26850    LOCATE 10,20:COLOR 4,2
26855    PRINT"REPEAT THE KNEADING OPERATION (Y/N) ?"
26860    '
26865    A£=INKEY£
26870    IF A£="Y" OR A£="y" THEN 27000
26875    IF A£="N" OR A£="n" THEN 26900
26880    GOTO 26865
26885    '
26890    '
26900    COLOR 7,1:CLS
26905    LOCATE 10,20:COLOR 2,4
26910    PRINT"LET ROBOT GO BACK TO HOME POSITION (Y/N) ?"
26915    A£=INKEY£
26920    IF A£="Y" OR A£="y" THEN 26945
26925    IF A£="N" OR A£="n" THEN 26960
26930    GOTO 26915
26935    '
```

```
26940   '
26945   PRINT #2,"NT"
26950   '
26955   '
26960   GOSUB 9000
26965   LOCATE 21,24:COLOR 4,2
26970   PRINT"..KNEADING OPERATION HAS BEEN COMPLETED.."
26975   GOTO 5000
26980   '
26990   '
27000   '  **.  REPEAT KNEADING OPERATION  .**
27010   '
27015   '
27020   IF CMARK=1 THEN 27100
27025   '
27030   NSTART=PNNK
27035   NSTOP=0
27040   NSTEP=-1
27045   MSTART=0
27050   MSTOP=PMMK
27055   MSTEP=1
27060   '
27065   CMARK=1
27070   '
27080   GOTO 27200
27090   '
27100   '
27110   NSTART=0
27120   NSTOP=PNNK
27130   NSTEP=1
27140   MSTART=0
27150   MSTOP=PMMK
27160   MSTEP=1
27170   '
27180   CMARK=0
27190   '
27200   '
27220   COLOR 7,1:CLS
27230   LOCATE 15,20:COLOR 4,2
27240   PRINT"PRESS ANY KEY TO REPEAT KNEADING OPERATION"
27250   '
27260   IF INKEY£="" THEN 27260
27270   '
27280   GOTO  26400
27290   '
27300   '
27310   '
28000   ' ***********************************************
28002   ' *                                             *
28004   ' *              PADDING OPERATION              *
28006   ' *                                             *
28008   ' ***********************************************
28010   '
28012   '
28015   '   **.  DECISION-MAKING FOR PADDING  .**
```

```
28020   '
28022   COLOR 7,1:CLS
28024   LOCATE 5,20:COLOR 4,2
28026   PRINT"MOVE ROBOT TO THE PADDING POSITION (Y/N)? "
28030   A£=INKEY£
28032   IF A£="Y" OR A£="y" THEN 28045
28034   IF A£="N" OR A£="n" THEN 4100    ' BACK TO MAIN MENU
28036   GOTO 28030
28038   '
28040   '
28045   '   ** INITIALIZE DAS8 FOR PALM FORCE **
28050   '
28052   GOSUB 41700
28054   '
28060   '
28065   '   ** MOVE ROBOT TO START POSITION **
28070   '
28072   XP=CINT(XPP(0,0))
28074   YP=CINT(YPP(0,0))
28076   ZP=CINT(ZPP(0,0))
28078   QP=CINT(QPP(0,0))
28080   QR=CINT(QPR(0,0))
28082   '
28084   X£=STR£(XP):Y£=STR£(YP):Z£=STR£(ZP)
28086   P£=STR£(QP):R£=STR£(QR)
28090   '
28095   GOSUB 40050
28100   '
28102   COLOR 7,1:CLS
28104   LOCATE 15,20:COLOR 2,4
28106   PRINT"PRESS ANY KEY TO START THE PADDING"
28110   '
28112   IF INKEY£="" THEN 28112
28114   '
28116   '
28120   '
28122   '    **.  PADDING ALONG + MASSAGING DIRECTION  .**
28124   '
28125   '
28126   FTIME=FTIME      ' FORCE RETENTION TIME
28130   DDX=0:DDY=0:DDZ=0
28132   '
28134   DABOVE=20        ' PALM ABOVE PART SURFACE
28136   DGRADE=11        ' PALM INITIAL MOTION GRADE
28140   DFUZZC=0         ' PALM QUICK MOTION DISTANCE
28142   '
28144   '
28150   NSTART=0
28152   NSTOP=PNNP
28155   NSTEP=1
28160   MSTART=0
28165   MSTOP=PMMP
28170   MSTEP=1
28175   '
28180   ZZC=0:FDZ=0
```

```
28190   '
28200   COLOR 7,1:CLS
28202   FOR JM=MSTART TO MSTOP STEP MSTEP
28206   FOR NI=NSTART TO NSTOP STEP NSTEP
28210   '
28215   GOSUB 41700       ' DAS8 INITIALIZE FOR PALM FORCE
28218   '
28220   EZC=0
28225   XP=CINT(XPP(JM,NI)+DDX):XPPREC(JM,NI)=XP
28230   YP=CINT(YPP(JM,NI)+DDY):YPPREC(JM,NI)=YP
28235   ZP=CINT(ZPP(JM,NI)+DDZ):ZPPREC(JM,NI)=ZP
28240   QP=CINT(QPP(JM,NI)):QPPREC(JM,NI)=QP
28242   QR=CINT(QPR(JM,NI)):QPRREC(JM,NI)=QR
28250   '
28252   '
28255   X£=STR£(XP)
28260   Y£=STR£(YP)
28265   Z£=STR£(ZP)
28270   P£=STR£(QP)
28275   R£=STR£(QR)
28280   '
28290   '
28300   GOSUB 40050       ' ROBOT ARM MOTION EXECUTION
28310   GOSUB 40200       ' FEEDBACK ARM POSITION
28320   '
28325   GOSUB 40900       ' FORCE FEEDBACK
28330   GOSUB 41400       ' FORCE COMPUTING
28334   GOSUB 38000       ' DISPLAY SENSED INFORMATION
28336   '
28340   FLIMIT=0.2
28342   JFDDP=ABS(FDDP-FFORCE)
28344   IF FDDP<=FLIMIT  THEN PZD=DGRADE:GOTO 28356
28346   IF JFDDP<=0.4    THEN PZD=0:GOTO 28400
28348   IF FDDP>=FFORCE  THEN PZD=-DGRADE/2:GOTO 28400
28350   PZD=0.5
28354   '
28356   EZC=EZC+PZD                ' FINE MOTION CONTROL
28360   ZZC=EZC+DFUZZC             ' QUICK APPROACH
28364   '
28366   DXC=ZZC*TAX                ' MOTION COORDINATING
28370   DYC=ZZC*TAY
28374   DZC=ZZC*TAZ
28378   '
28380   XC=CINT(XP+DXC)
28384   YC=CINT(YP+DYC)
28386   ZC=CINT(ZP+DZC)
28388   X£=STR£(XC)
28390   Y£=STR£(YC)
28392   Z£=STR£(ZC)
28394   '
28396   GOTO 28300                 ' FINE MOTION REPEAT
28398   '
28400   DGRADE=1.5
28410   EZC=ZZC
28415   '
```

```
28420   IF ABS(EZC)>=29 THEN 28460
28425   '
28430   GOSUB 32000                 ' FUZZY INFERENCE
28432   FDZ1=DABOVE*SGN(FDZ)
28435   FDZ=FDZ-FDZ1                ' FUZZY CORRECTION
28440   DFUZZC=(EZC-FDZ)*0.75       ' QUICK APPROACH
28445   GOTO 28480
28450   '
28455   '
28460   FDZ=EZC-DABOVE              ' NON-FUZZY CORRECTION
28465   DFUZZC=DABOVE*0.75          ' QUICK APPROACH
28470   '
28475   '
28480   DDX=DDX+FDZ*TAX             ' MOTION COORDINATING
28485   DDY=DDY+FDZ*TAY
28490   DDZ=DDZ+FDZ*TAZ
28495   '
28500   FPXREC(JM,NI)=FDZ*TAX
28502   FPYREC(JM,NI)=FDZ*TAY
28504   FPZREC(JM,NI)=FDZ*TAZ
28508   DZREC(JM,NI)=EZC
28510   GOSUB 42000                 ' FORCE RETENTION TIME
28515   '
28520   MBACK=-DABOVE
28530   XC=CINT(XC+MBACK*TAX)
28535   YC=CINT(YC+MBACK*TAY)
28540   ZC=CINT(ZC+MBACK*TAZ)
28545   '
28550   X£=STR£(XC)
28555   Y£=STR£(YC)
28560   Z£=STR£(ZC)
28565   '
28570   GOSUB 40050
28575   '
28580   NEXT NI
28585   NEXT JM
28590   '
28594   '   ** RESULTS RECORDING **
28598   '
28600   GOSUB 47500
28602   '
28605   '
28608   '   **.  PADDING ALONG - MASSAGING DIRECTION  .**
28610   '
28615   COLOR 7,1:CLS
28620   LOCATE 10,20:COLOR 4,2
28625   PRINT"REPEAT THE PADDING OPERATION (Y/N) ? "
28630   '
28635   A£=INKEY£
28640   IF A£="Y" OR A£="y" THEN 28800
28645   IF A£="N" OR A£="n" THEN 28665
28650   GOTO 28635
28655   '
28660   '
28665   COLOR 7,1:CLS
```

```
28670   LOCATE 10,20:COLOR 4,2
28675   PRINT"LET ROBOT GO BACK TO HOME POSITION (Y/N) ?"
28680   '
28685   A£=INKEY£
28690   IF A£="Y" OR A£="y" THEN 28705
28694   IF A£="N" OR A£="n" THEN 28710
28698   GOTO 28685
28700   '
28705   PRINT #2,"NT"
28708   '
28710   GOSUB 9000
28715   LOCATE 21,22:COLOR 4,2
28720   PRINT".. PADDING OPERATION HAS BEEN COMPLETED .."
28725   GOTO 5000
28730   '
28735   '
28740   '   **.   REPEAT THE PADDING   .**
28745   '
28750   '
28760   '
28800   COLOR 7,1:CLS
28810   LOCATE 10,20:COLOR 4,2
28820   PRINT"PRESS ANY KEY TO REPEAT PADDING OPERATION"
28830   '
28840   IF INKEY£="" THEN 28840
28850   '
28860   GOTO  28200               ' REPEAT
28870   '
28900   '
28920   '
28930   '
30000   ' %%** On-line error correction for kneading **%%
30005   '
30010   ' ****************** Statement *****************
30011   ' *                                              *
30012   ' *     Subroutine to infer the corrections      *
30014   ' *                                              *
30016   ' *   a. Fuzzification of the error input        *
30018   ' *   b. Fuzzy inference                         *
30020   ' *   c. Defuzzification of inferred output      *
30022   ' *   d. Computation of correction distance      *
30026   ' *                                              *
30028   ' ************************************************
30030   '
30040   '
30050   '  aa. Judge if correction is required .aa
30055   '
30060   '
30065   IF ABS(EQF1)>8.0 OR ABS(EQF2)>8.0 THEN 30200
30070   '
30075   '  ** NO CORRECTION REQUIRED **
30078   '
30080   DPX=0:DPY=0:DPZ=0
30090   RETURN
30095   '
```

```
30100   '
30200   '  bb. Fuzzfication of error input .bb
30205   '
30210   '  ** Fuzzy scaler **
30215   '
30220   FKKQ=3            ' Degree
30225   '
30330   '  ** FUZZIFYING INPUTS **
30332   '
30334   ' FOR FINGER #1
30338   '
30340   FXX=EQF1
30344   SCALE=FKKQ
30346   GOSUB 36000
30350   QFU1£=FUZ£                      ' FIRED TERM
30352   NFIRE1=FFIRE                    ' FIRE STRENGTH TERM
30355   '
30360   ' FOR FINGER #2
30365   '
30370   FXX=EQF2
30375   SCALE=FKKQ
30380   GOSUB 36000
30385   QFU2£=FUZ£                      ' FIRED TERM
30390   NFIRE2=FFIRE                    ' FIRE STRENGTH TERM
30395   '
30400   '
30450   '
30500   '  cc. Truth for control rules (Knead) .cc
30505   '
30510   '  WW(8) -- Truth value for kneading rule base
30520   '
30530   '   For EQ1 input, the fire strength vector is:
30535   '   SFIRE(NFIRE1,J)  where J=0 to 6
30542   '
30544   '   For EQ2 input, the fire strength vector is:
30546   '   SFIRE(NFIRE2,J)  where J=0 to 6
30550   '
30554   '   The order of the fire strength in the control
30558   '   rule base should be organized as follows:
30564   '
30566   '          | SFIRE(NFIRE1,1), SFIRE(NFIRE2,5) |
30568   '          | SFIRE(NFIRE1,0), SFIRE(NFIRE2,6) |
30570   '          | SFIRE(NFIRE1,5), SFIRE(NFIRE2,1) |
30572   '          | SFIRE(NFIRE1,6), SFIRE(NFIRE2,0) |
30576   '  min     | SFIRE(NFIRE1,1), SFIRE(NFIRE2,1) |
30578   '          | SFIRE(NFIRE1,0), SFIRE(NFIRE2,0) |
30580   '          | SFIRE(NFIRE1,5), SFIRE(NFIRE2,5) |
30582   '          | SFIRE(NFIRE1,6), SFIRE(NFIRE2,6) |
30584   '
30590   '
30600   RULEQ1(0)=SFIRE(NFIRE1,1):RULEQ2(0)=SFIRE(NFIRE2,5)
30605   RULEQ1(1)=SFIRE(NFIRE1,0):RULEQ2(1)=SFIRE(NFIRE2,6)
30610   RULEQ1(2)=SFIRE(NFIRE1,5):RULEQ2(2)=SFIRE(NFIRE2,1)
30615   RULEQ1(3)=SFIRE(NFIRE1,6):RULEQ2(3)=SFIRE(NFIRE2,0)
30620   RULEQ1(4)=SFIRE(NFIRE1,1):RULEQ2(4)=SFIRE(NFIRE2,1)
```

```
30625   RULEQ1(5)=SFIRE(NFIRE1,0):RULEQ2(5)=SFIRE(NFIRE2,0)
30630   RULEQ1(6)=SFIRE(NFIRE1,5):RULEQ2(6)=SFIRE(NFIRE2,5)
30635   RULEQ1(7)=SFIRE(NFIRE1,6):RULEQ2(7)=SFIRE(NFIRE2,6)
30640   '
30645   '
30650   FOR I=0 TO 7
30655   IF RULEQ1(I)<RULEQ2(I) THEN 30670
30660   WW(I)=RULEQ2(I)
30665   GOTO 30675
30670   WW(I)=RULEQ1(I)
30675   NEXT I
30680   '
30685   '
30690   '
30700   ' dd. fuzzy inference process .dd
30705   '
30710   '
30712   '  ** Defuzzy scaler **
30714   '
30718   FKK0=4            ' mm
30720   '
30722   '  ** Fuzzy inference based on fuzzy logic **
30724   '
30725   EPX=0:EPY=0:EPZ=0
30728   WWD=0
30730   '
30740   FOR I=0 TO 7
30745   WWD=WWD+WW(I)
30750   EPX=EPX+WW(I)*YYYK(I,0)
30755   EPY=EPY+WW(I)*YYYK(I,1)
30760   EPZ=EPZ+WW(I)*YYYK(I,2)
30765   NEXT I
30770   '
30772   IF WWD=0 THEN  30790
30776   EPX=EPX/WWD*FKK0
30780   EPY=EPY/WWD*FKK0
30782   EPZ=EPZ/WWD*FKK0
30784   GOTO  30820
30786   '
30790   EPX=0:EPY=0:EPZ=0
30792   '
30794   '
30800   '  ee. Path modifying .ee
30810   '
30815   '
30820   DPX=EPX*TNX+EPY*TOX+EPZ*TAX
30830   DPY=EPX*TNY+EPY*TOY+EPZ*TAY
30835   DPZ=EPX*TNZ+EPY*TOZ+EPZ*TAZ
30850   '
30900   RETURN
31000   '
31500   '
32000   '  %%** On-line error correction for padding **%%
32010   '
```

```
32015   '  ****************  Statement  ****************
32020   '  *                                              *
32025   '  *    Subroutine to infer the corrections       *
32030   '  *                                              *
32032   '  *    a. Fuzzification of the error input       *
32034   '  *    b. Fuzzy inference                        *
32040   '  *    d. Defuzzification of inferred output     *
32044   '  *    e. Computation of correction distance     *
32048   '  *                                              *
32050   '  ************************************************
32060   '
32105
32110   '  bb. Fuzzification of error input .bb
32115   '
32120   '  **  Fuzzy scaler  **
32125   '
32130   FKKD=5          ' mm
32135   '
32140   '  ** FUZZIFYING INPUTS **
32145   '
32150   FXX=EZC
32155   SCALE=FKKD
32160   GOSUB 36000
32165   DFU£=FUZ£                   ' FIRED TERM
32170   NFIRE=FFIRE                 ' FIRE STRENGTH
32175   '
32180   '
32185   '  cc.  Truth for control rules (pad)  .cc
32190   '
32195   '  WW(6)          -- Truth value for control rule
32200   '  SFIRE(NFIRE,J) -- Fire strength vector
32204   '
32206   WW(0)=SFIRE(NFIRE,4)
32210   WW(1)=SFIRE(NFIRE,5)
32215   WW(2)=SFIRE(NIFRE,6)
32218   WW(3)=SFIRE(NFIRE,2)
32220   WW(4)=SFIRE(NFIRE,1)
32225   WW(5)=SFIRE(NFIRE,0)
32230   '
32235   '
32240   '  dd. Fuzzy inference process .dd
32245   '
32250   '  **  Defuzzy scaler  **
32255   '
32260   FKK0=5                      ' mm
32265   '
32270   '  ** Fuzzy inference based ob fuzzy logic **
32275   '
32280   FDZ=0
32285   WWD=0
32290   '
32295   FOR I=0 TO 3
32300   WWD=WWD+WW(I)
32325   FDZ=FDZ+WW(I)*YYYP(I,2)
32330   NEXT I
```

```
32335   '
32350   FDZ=FDZ/WWD*FKK0
32360   '
32370   RETURN
32380   '
32400   '
35000   '
36000   ' %%** FUZZIFACATION MODULE FOR KNEADING **%%
36005   '
36010   '
36020   '   ** FUZZIFICATION FOR KNEADING **
36030   '
36035   '   FXX     -- CRISP INPUTS
36037   '   SCALE   -- FUZZIFICATION SCALER
36040   '   FUZ£    -- FUZZY LABELS
36045   '
36050   FXX=FXX/SCALE
36060   IF FXX<(-5)                       THEN 36100
36065   IF FXX>=(-5) AND FXX<(-3)         THEN 36110
36070   IF FXX>=(-3) AND FXX<(-1)         THEN 36120
36075   IF FXX>=(-1) AND FXX<=1           THEN 36130
36080   IF FXX>1      AND FXX<=3          THEN 36140
36085   IF FXX>3      AND FXX<=5          THEN 36150
36090   IF FXX>5                          THEN 36160
36095   '
36100   FUZ£="NB":FFIRE=0:RETURN
36110   FUZ£="NM":FFIRE=1:RETURN
36120   FUZ£="NS":FFIRE=2:RETURN
36130   FUZ£="ZE":FFIRE=3:RETURN
36140   FUZ£="PS":FFIRE=4:RETURN
36150   FUZ£="PM":FFIRE=5:RETURN
36160   FUZ£="PB":FFIRE=6:RETURN
36180   '
36800   '
37000   ' %%** DATA LOADING FOR ON-LINE CONTROL **%%
37004   '
37008   '  a. * FIRE STRENGTH TABLE LOADING *
37010   '
37020   '  SFIRE(I,J)   --   FIRE STRENGTH TABLE
37025   '  I   --   No. of fuzzy input terms for QF1 & QF2
37030   '  J   --   No. of fuzzy terms in the Rule base
37035   '
37040   FOR I=0 TO 6
37042   FOR J=0 TO 6
37048   READ SFIRE(I,J)
37050   NEXT J
37052   NEXT I
37055   '
37060   DATA  1.0,  0.3,    0,    0,    0,    0,    0
37062   DATA  0.3,  1.0,  0.3,    0,    0,    0,    0
37068   DATA    0,  0.3,  1.0,  0.3,    0,    0,    0
37070   DATA    0,    0,  0.3,  1.0,  0.3,    0,    0
37075   DATA    0,    0,    0,  0.3,  1.0,  0.3,    0
37080   DATA    0,    0,    0,    0,  0.3,  1.0,  0.3
37085   DATA    0,    0,    0,    0,    0,  0.3,  1.0
```

```
37090   '
37095   '
37100   '
37105   '  b. * OUTPUT LOADING FOR KEADING RULE BASE *
37110   '
37112   ' YYYK(8,3) -- OUTPUT TABLE FOR KNEADING
37114   ' YYYK(I,1) -- Ex output
37116   ' YYYK(I,2) -- Ey output
37118   ' YYYK(I,3) -- Ez output
37120   '
37130   FOR I=0 TO 7
37132   FOR J=0 TO 2
37135   READ YYYK(I,J)
37138   NEXT J
37140   NEXT I
37145   '
37148   '
37150   DATA  0, -5,  0
37152   DATA  0, -6,  0
37154   DATA  0,  5,  0
37156   DATA  0,  6,  0
37158   DATA  0,  0,  5
37160   DATA  0,  0,  6
37162   DATA  0,  0, -5
37164   DATA  0,  0, -6
37170   '
37180   '
37200   '  c. * OUTPUR LOADING FOR PADDING RULE BASE *
37210   '
37215   '  YYYP(6,3)  --  OUTPUT TABLE FOR PADDING
37220   '  YYYP(I,0)  --  EX
37225   '  YYYP(I,1)  --  EY
37230   '  YYYP(I,2)  --  EZ
37235   '
37240   '
37245   FOR I=0 TO 5
37250   FOR J=0 TO 2
37255   READ YYYP(I,J)
37260   NEXT J
37270   NEXT I
37300   '
37310   DATA  0,  0,  3
37320   DATA  0,  0,  4
37330   DATA  0,  0,  5
37335   DATA  0,  0, -3
37340   DATA  0,  0, -4
37350   DATA  0,  0, -5
37355   '
37360   '
37370   RETURN
37380   '
37390   '
37500   '
```

```
37600   ' %%** INFORMATION DISPLAY FOR PADDING **%%
37700   '
37800   '
38000   ' ** COMMANDED POSITION & FORCE DISPLAY **
38005   '
38010   '
38020   LOCATE 2,10:COLOR 2,4
38030   PRINT"***** COMMAND  POSITION *****"
38040   COLOR 4,2
38050   LOCATE 4,20: PRINT"PX = ";USING"+####.##";XPP(JM,NI)
38060   LOCATE 5,20: PRINT"PY = ";USING"+####.##";YPP(JM,NI)
38070   LOCATE 6,20: PRINT"PZ = ";USING"+####.##";ZPP(JM,NI)
38080   LOCATE 7,20: PRINT"QP = ";USING"+####.##";QPP(JM,NI)
38090   LOCATE 8,20: PRINT"QR = ";USING"+####.##";QPR(JM,NI)
38100   '
38110   LOCATE 10,10:COLOR 2,4
38120   PRINT"***** COMMAND PAD FORCE *****"
38130   COLOR 4,2
38140   LOCATE 12,20:PRINT"FORCE = ";USING"+##.#";FFORCE
38150   '
38170   '
38180   '
38200   '
38300   ' ** SENSED POSITION & FORCE **
38305   '
38310   '
38320   LOCATE 2,40:COLOR 2,4
38330   PRINT"***** SENSED  POSITION *****"
38340   COLOR 4,2
38350   LOCATE 4,50:PRINT"PX = ";USING"+####.##";VV(1)
38360   LOCATE 5,50:PRINT"PY = ";USING"+####.##";VV(2)
38370   LOCATE 6,50:PRINT"PZ = ";USING"+####.##";VV(3)
38380   LOCATE 7,50:PRINT"QP = ";USING"+####.##";VV(4)
38390   LOCATE 8,50:PRINT"QR = ";USING"+####.##";VV(5)
38400   '
38410   LOCATE 10,40:COLOR 2,4
38420   PRINT"***** SENSED PAD FORCE  *****"
38430   COLOR 4,2
38440   LOCATE 12,50:PRINT"FORCE = ";USING"+##.#";FDDP
38450   '
38500   '
38550   '
38600   ' ** FUZZY INFERENCE RESULTS **
38605   '
38610   LOCATE 16,10:COLOR 2,4
38620   PRINT"*****  FUZZY INFERENCE  *****"
38630   COLOR 4,2
38640   LOCATE 18,15
38650   PRINT"FUZZY  INPUT = ";USING"+####.##";ZZC-DABOVE
38660   LOCATE 19,15
38670   PRINT"FUZZY OUTPUT = ";USING"+####.##";FDZ
38680   LOCATE 20,15
38690   PRINT"QUICK MOTION = ";USING"+####.##";DFUZZC
38700   '
```

```
38710   LOCATE 16,40:COLOR 2,4
38720   PRINT"***** TOTAL CORRECTIONS *****"
38730   COLOR 4,2
38740   LOCATE 18,50:PRINT"DDX = ";USING"+####.##";DDX
38750   LOCATE 19,50:PRINT"DDY = ";USING"+####.##";DDY
38760   LOCATE 20,50:PRINT"DDZ = ";USING"+####.##";DDZ
38770   '
38780   RETURN
38800   '
38850   '
38900   '
38950   ' %%**  INFORMATION DISPLAY FOR KNEADING **%%
38980   '
39000   ' ** COMMAND POSITION **
39005   '
39010   LOCATE 2,10:COLOR 2,4
39020   PRINT"***** COMMAND  POSITION *****"
39025   COLOR 4,2
39030   LOCATE 4,20:PRINT"PX = ";USING"+####.##";XKP(NI)
39040   LOCATE 5,20:PRINT"PY = ";USING"+####.##";YKP(NI)
39050   LOCATE 6,20:PRINT"PZ = ";USING"+####.##";ZKP(NI)
39060   LOCATE 7,20:PRINT"QP = ";USING"+####.##";QKP(NI)
39070   LOCATE 8,20:PRINT"QR = ";USING"+####.##";QKR(NI)
39080   LOCATE 9,20:PRINT"Q1 = ";USING"+####.##";QF(NI)
39090   LOCATE 10,20:PRINT"Q2 = ";USING"+####.##";QF(NI)
39100   '
39105   '
39110   ' ** SENSED POSITION **
39115   '
39120   '
39125   LOCATE 2,40:COLOR 2,4
39130   PRINT"*****  SENSED POSITION  *****"
39135   COLOR 4,2
39140   LOCATE 4,50:PRINT"PX = ";USING"+####.##";VV(1)
39145   LOCATE 5,50:PRINT"PY = ";USING"+####.##";VV(2)
39150   LOCATE 6,50:PRINT"PZ = ";USING"+####.##";VV(3)
39160   LOCATE 7,50:PRINT"QP = ";USING"+####.##";VV(4)
39170   LOCATE 8,50:PRINT"QR = ";USING"+####.##";VV(5)
39180   LOCATE 9,50:PRINT"Q1 = ";USING"+####.##";FQF1
39190   LOCATE 10,50:PRINT"Q2 = ";USING"+####.##";FQF2
39200   '
39205   '
39210   '
39215   ' ** FUZZY INFERENCE **
39220   '
39225   '
39230   LOCATE 14,10:COLOR 2,4
39240   PRINT"*****   FUZZY  INPUTS   *****"
39250   COLOR 4,2
39260   LOCATE 16,20:PRINT"EQ1 = ";USING"+####.##";EQF1
39270   LOCATE 18,20:PRINT"EQ2 = ";USING"+####.##";EQF2
39280   '
39300   '
```

```
39310   LOCATE 14,40:COLOR 2,4
39320   PRINT"*****  FUZZY  OUTPUTS  *****"
39330   COLOR 4,2
39340   LOCATE 16,50:PRINT"EPX = ";USING"+####.##";EPX
39350   LOCATE 17,50:PRINT"EPY = ";USING"+####.##";EPY
39360   LOCATE 18,50:PRINT"EPZ = ";USING"+####.##";EPZ
39370   '
39380   RETURN
39400   '
39410   '
39420   '
40000   ' %%**  ROBOT ARM POSITION MOTION **%%
40010   '
40050   PRINT #2,"MP"+X£+","+Y£+","+Z£+","+P£+","+R£
40055   RETURN
40060   '
40080   '
40200   ' %%** FEEDBACK OF THE ROBOT ARM POSITION **%%
40205   '
40208   '
40210   PRINT #2,"WH"
40215   LINE INPUT #2,A£
40220   D£=A£
40224   K=1
40226   FOR I1=1 TO 5
40228   IF I1=5 THEN 40232
40230   AA(I1)=INSTR(K,D£,","):GOTO 40236
40232   AA(I1)=LEN(D£)+1
40236   VV(I1)=VAL(MID£(D£,K,AA(I1)-1))
40238   K=AA(I1)+1
40240   NEXT I1
40250   '
40260   RETURN
40280   '
40290   '
40300   ' %%**  MICROSWITH DETECTION FETCH **%%
40310   '
40320   '
40330   PIOA%=INP(&H31C)
40340   RETURN
40350   '
40390   '
40400   ' %%** POWER SUPPLY FOR MOTOR #1 (D/A #4) **%%
40405   '
40408   '
40410   VIN1=20.1*IL1/1000
40415   '
40420   DD=2047+INT(204.8*VIN1)          'VOLTS
40425   '
40430   DH%=INT(DD/256)
40440   DL%=DD-DH%*256
40445   OUT &H318,DL%
40450   OUT &H319,DH%
40455   '
40460   RETURN
```

```
40465   '
40470   '
40500   ' %%** POWER SUPPLY FOR MOTOR #2 (D/A #5) **%%
40505   '
40510   VIN2=20.1*IL2/1000
40515   '
40520   DD=2047+INT(204.8*VIN2)          'VOLTS
40525   '
40530   DH%=INT(DD/256)
40535   DL%=DD-DH%*256
40540   OUT &H31A,DL%
40545   OUT &H31B,DH%
40550   '
40555   RETURN
40560   '
40570   '
40700   ' %%** HAND-II POSITION SERVO SAMPLING **%%
40705   '
40710   '
40720   MD%=5
40725   DIO%(0)=VARPTR(PARRAY%(0))
40732   DIO%(1)=8
40734   FLAG%=0
40738   CALL DAS8(MD%,DIO%(0),FLAG%)
40740   '
40744   POSIT1=0:POSIT2=0
40748   FOR JJ%=0 TO 3
40750   POSIT1=POSIT1+PARRAY%(2*JJ%)
40754   POSIT2=POSIT2+PARRAY%(2*JJ%+1)
40756   NEXT JJ%
40760   '
40764   POSIT1=POSIT1/4:POSIT2=POSIT2/4
40768   FQF1=90/1.21*(POSIT1*5/2048-1.85)        'Degree
40770   FQF2=90/1.21*(POSIT2*5/2048-1.85)
40772   '
40778   RETURN
40780   '
40790   '
40800   ' %%** FORCE SAMPLING FOR FINGER SERVO LOOP **%%
40805   '
40808   '
40810   MD%=5                          ' MODE 5
40815   DIO%(0)=VARPTR(FARRAY%(0))
40820   DIO%(1)=16
40825   FLAG%=0
40830   CALL DAS8(MD%,DIO%(0),FLAG%)
40835   '
40840   FORCE1=0:FORCE2=0
40844   FOR JJ%=0 TO 7
40848   FORCE1=FORCE1+FARRAY%(2*JJ%)
40850   FORCE2=FORCE2+FARRAY%(2*JJ%+1)
40855   NEXT JJ%
40860   '
40865   FORCE1=FORCE1/8:FORCE2=FORCE2/8
40870   RETURN
```

```
40880   '
40890   '
40900   ' %%** FORCE SAMPLING FOR PALM SERVO LOOP **%%
40905   '
40910   '
40915   MD%=5
40920   DIO%(0)=VARPTR(FARRAY%(0))
40924   DIO%(1)=16
40928   FLAG%=0
40930   CALL DAS8(MD%,DIO%(0),FLAG%)
40934   '
40936   FORCEP=0
40940   '
40944   FOR JJ%=0 TO 15
40946   FORCEP=FORCEP+FARRAY%(JJ%)
40950   NEXT JJ%
40954   '
40956   FORCEP=FORCEP/16
40960   RETURN
40964   '
40965   '
41180   '
41200   '  %%** CALIBRATION FOR HAND-II F-SENSOR #1  **%%
41205   '       ( FINGER #1 IN HAND-II -- FORCE )
41210   '
41220   KD1=FORCE1*5/2048
41225   NCCN=0.00981
41227   IF KD1>0.23 THEN 41230
41229   FDD1=0:RETURN
41230   IF KD1>0.92 THEN 41240
41235   FDD1=NCCN*(124+124*(KD1-0.92)/0.74):RETURN
41240   IF KD1>3.89 THEN FDD1=9:RETURN
41245   FDD1=NCCN*(924+800*(KD1-3.89)/2.97)
41250   RETURN
41255   '
41260   '
41270   '
41300   '  %%** CALIBRATION FOR HAND-II F-SENSOR #2 **%%
41305   '       ( FINGER #2 IN HAND #2 -- FORCE )
41310   '
41320   KD2=FORCE2*5/2048
41325   NCCN=0.00981
41327   IF KD2>0.60 THEN 41330
41329   FDD2=0:RETURN
41330   IF KD2>1.08 THEN 41340
41335   FDD2=NCCN*(124+124*(KD2-1.08)/0.88):RETURN
41340   IF KD2>4.03 THEN FDD2=9:RETURN
41345   FDD2=NCCN*(924+800*(KD2-4.03)/2.95)
41350   RETURN
41355   '
41360   '
41380   '
41400   ' %%** CALIBRATION FOR PALM-II F-SENSOR **%%
41410   '      ( PALM-II IN HAND-II  -- FORCE )
41420   '
```

```
41450   KDP=FORCEP*5/2048
41452   NCCN=0.00981
41458   '
41460   IF KDP>0.20 THEN 41464
41462   FDDP=0:RETURN
41464   IF KDP>1.60 THEN 41472
41470   FDDP=NCCN*(124+124*(KDP-1.6)/1.4):RETURN
41472   IF KDP>2.28 THEN 41480
41474   FDDP=NCCN*(324+200*(KDP-2.28)/0.68):RETURN
41480   IF KDP>3.56 THEN FDDP=8:RETURN
41482   FDDP=NCCN*(924+600*(KDP-3.56)/1.28):RETURN
41484   RETURN
41490   '
41492   '
41494   '
41500   '   %%** INITIALIZE DAS8 FOR POSITION A/D **%%
41505   '           ( FINGER ANGLE DETECTION )
41508   '
41510   MD%=10
41514   DIO%(0)=2
41518   DIO%(1)=3
41520   FLAG%=0
41525   CALL DAS8(MD%,DIO%(0),FLAG%)
41528   '
41530   FREQ=2000                    ' FREQUENCY=Samples/sec
41534   NC2=CINT(6000/FREQ*1000)     ' SYSTEM CLOCK = 12 MHZ
41536   '
41538   MD%=11
41540   DIO%(0)=2
41544   DIO%(1)=NC2
41546   FLAG%=0
41548   CALL DAS8(MD%,DIO%(0),FLAG%)
41550   '
41552   MD%=1
41554   DIO%(0)=0
41558   DIO%(1)=1
41560   FLAG%=0
41564   CALL DAS8(MD%,DIO%(0),FLAG%)
41566   '
41568   MD%=2
41570   CH%=0
41572   FLAG%=0
41574   CALL DAS8(MD%,CH%,FLAG%)
41576   '
41580   RETURN
41585   '
41590   '
41600   ' %%** INITIALIZE DAS8 FOR FORCE A/D **%%
41604   '        ( FINGER FORCE DETECTION )
41608   '
41610   MD%=10
41612   DIO%(0)=2
41616   DIO%(1)=3
41618   FLAG%=0
41620   CALL DAS8(MD%,DIO%(0),FLAG%)
```

```
41622   '
41626   FREQ=2000                     ' FREQENCY=Samples/sec
41628   NC2=CINT(6000/FREQ*1000)      ' system clock = 12 MHZ
41630   '
41634   MD%=11
41636   DIO%(0)=2
41638   DIO%(1)=NC2
41640   FLAG%=0
41645   CALL DAS8(MD%,DIO%(0),FLAG%)
41648   '
41650   '
41652   MD%=1
41654   DIO%(0)=2
41656   DIO%(1)=3
41660   FLAG%=0
41664   CALL DAS8(MD%,DIO%(0),FLAG%)
41670   '
41672   MD%=2
41674   CH%=2
41676   FLAG%=0
41678   CALL DAS8(MD%,CH%,FLAG%)
41680   '
41685   RETURN
41690   '
41695   '
41700   ' %%** INITIALIZE DAS8 FOR FORCE A/D **%%
41705   '        ( PALM FORCE DETECTION )
41710   '
41720   MD%=10
41722   DIO%(0)=2
41724   DIO%(1)=3
41728   FALG%=0
41730   CALL DAS8(MD%,DIO%(0),FLAG%)
41732   '
41734   FREQ=1000
41736   NC2=CINT(6000/FREQ*1000)
41740   '
41742   MD%=11
41744   DIO%(0)=2
41748   IF NC2<32767 THEN DIO%(1)=NC2:GOTO 41752
41750   DIO%(1)=NC2-65536!
41752   FLAG%=0
41754   CALL DAS8(MD%,DIO%(0),FLAG%)
41758   '
41760   MD%=1
41762   DIO%(0)=4
41764   DIO%(1)=4
41768   FLAG%=0
41770   CALL DAS8(MD%,DIO%(0),FLAG%)
41772   '
41775   MD%=2
41778   CH%=4
41780   FLAG%=0
41782   CALL DAS8(MD%,CH%,FLAG%)
41785   '
```

```
41790   RETURN
41800   '
41900   '
42000   ' %%** TIME DELAY USING TIMER IN DAS8 **%%
42010   '
42020   '
42030   DSCAN=FTIME       ' FORCE RETENTION TIME
42040   '
42050   FREQ=200          ' A/D FREQENCY
42055   '
42060   '
42065   MD%=10
42070   DIO%(0)=2:DIO%(1)=3
42075   CALL DAS8(MD%,DIO%(0),FLAG%)
42080   '
42085   NC2=CINT(6000/FREQ*1000)          ' SYSTEM CLOCK
42090   '
42100   MD%=11
42110   DIO%(0)=2
42120   IF NC2<32767 THEN DIO%(1)=NC2:GOTO 42140
42130   DIO%(1)=NC2-65536!
42140   CALL DAS8(MD%,DIO%(0),FLAG%)
42150   '
42160   NCCD=CINT(DSCAN*FREQ)
42170   NCCD1=CINT(NCCD/2)
42180   IF (NCCD-2*NCCD1)=0 THEN 42200
42190   NCCD=NCCD-1
42195   '
42200   MD%=5
42205   DIO%(0)=VARPTR(FDELAY%(0))
42210   DIO%(1)=NCCD
42230   FLAG%=0
42240   CALL DAS8(MD%,DIO%(0),FLAG%)
42250   '
42260   '
42270   ' ** RELEASE FORCE HOLDING **
42300   '
42310   '
42320   IL1=2:IL2=2
42330   GOSUB 40400
42340   GOSUB 40500
42350   '
42360   RETURN
42370   '
42380   '
43000   '
43500   '  %%**  HAND-II POSITION SERVO CONTROL  **%%
43505   '
43510   '
43515   '  ** INITIALIZE DAS8 FOR POSITION SAMPLING
43520   '
43525   GOSUB 41500
43530   '
43535   '  ** COFFICIENCE
43540   '
```

```
43545   V01=30/50:V02=50/50
43550   K101=15/10/50:K102=15/10/50
43555   K051=10/10/50:K052=10/5/50
43560   QLIM=1.0
43565   '
43570   '  ** FRICTION COMPENSATION **
43575   '
43580   VF1=-5/50
43584   VF2=10/50
43586   '
43590   '  ** FINGER POSITION SERVO CONTROL
43594   '
43596   NN=0
43600   '
43610   GOSUB 40700
43620   '
43625   EQ1=QD1-FQF1:EQ2=QD2-FQF2
43630   S1=SGN(EQ1):S2=SGN(EQ2)
43635   AE1=ABS(EQ1):AE2=ABS(EQ2)
43640   '
43700   IF AE1<=QLIM THEN VA1=0:GOTO 43720
43702   IF AE1>20    THEN VA1=VF1+60/50:GOTO 43720
43704   IF AE1>10    THEN VA1=VF1+45/50+K101*AE1:GOTO 43720
43706   IF AE1>5     THEN VA1=VF1+V01+K051*AE1:GOTO 43720
43710   IF AE1>QLIM  THEN VA1=V01
43715   '
43720   VIN1=S1*VA1
43725   GOSUB 40420
43728   '
43730   IF AE2<QLIM THEN VA2=0:GOTO 43740
43732   IF AE2>20   THEN VA2=VF2+75/50:GOTO 43740
43734   IF AE2>10   THEN VA2=VF2+60/50+K102*AE2:GOTO 43740
43736   IF AE2>5    THEN VA2=VF2+V02+K052*AE1:GOTO 43740
43738   IF AE2>QLIM THEN VA2=VF2+V02
43740   '
43742   VIN2=S2*VA2
43744   GOSUB 40520
43746   '
43748   '
43750   IF AE1<=QLIM AND AE2<=QLIM THEN 43770
43752   '
43754   NN=NN+1
43756   IF NN>300 THEN 43770
43760   GOTO 43610
43765   '
43770   VIN1=0:VIN2=0
43775   GOSUB 40420
43780   GOSUB 40520
43785   '
43790   RETURN
43795   '
43800   '
43810   '
```

```
44500    '  %%**  HAND-II FORCE SERVO CONTROL  **%%
44505    '
44510    '
44515    '  ** INITIALIZE DAS8 FOR FORCE SAMPLING
44520    '
44525    GOSUB 41600
44530    '
44535    '  ** REQUIRED CURRECT FOR MOTORS
44540    '
44545    LL0=0.090
44550    KVIN0=LL0/0.16495
44555    VF1=KVIN0*FD1
44560    VF2=KVIN0*FD2
44565    '
44570    '
44600    '  ** INITIAL TOUCH DETECT
44605    '
44610    FOR KN=0 TO 100
44615    VIN1=KVIN0*(1.5+(FD1*2/3-1.5)/100*KN)
44620    VIN2=KVIN0*(1.5+(FD2*2/3-1.5)/100*KN)
44625    '
44630    GOSUB 40420
44635    GOSUB 40520
44665    GOSUB 45000
44670    NEXT KN
44680    '
44685    '  ** TIME DELAY
44690    '
44695    GOSUB 45000
44700    '
44705    '  ** FORCE CONTROL
44710    '
44715    FOR KN=1 TO 100
44720    VIN1=VF1*(2/3+1/300*KN)
44730    VIN2=VF2*(2/3+1/300*KN)
44740    '
44750    GOSUB 40420
44755    GOSUB 40520
44760    NEXT KN
44765    '
44780    RETURN
44785    '
44790    '
44895    '
45000    '   %%** TIME DELAY FOR FORCE CONTROL **%%
45010    '
45020    '
45030    FOR KW%=0 TO 1000
45040    NEXT KW%
45050    RETURN
45100    '
45200    '
45300    '
46000    '   %%** TIME DELAY FOR HYBRID CONTROL **%%
46010    '
```

```
46020   FOR KW%=0 TO 300
46030   NEXT KW%
46040   '
46050   IL1=0:IL2=0
46060   GOSUB 40400
46070   GOSUB 40500
46080   '
46090   RETURN
46100   '
46500   '
47000   ' ** KNEADING EXPERIMENTAL RESULTS RECORDING **
47005   '
47010   COLOR 7,1:CLS
47020   LOCATE 5,20:COLOR 4,2
47030   PRINT"RECORD THE KNEADING RESULTS (Y/N)?"
47040   '
47050   A£=INKEY£
47060   IF A£="Y" OR A£="y" THEN 47100
47070   IF A£="N" OR A£="n" THEN RETURN
47080   GOTO 47050
47090   '
47100   COLOR 7,1:CLS:LOCATE 10,15:COLOR 4,2
47105   INPUT"PLEASE INPUT DATA FILE NAME *.DOC ";RDOC£
47110   '
47120   LOCATE 12,15:COLOR 2,4
47125   PRINT"IS THE FILE NAME CORRECT (Y/N)? "
47130   A£=INKEY£
47135   IF A£="Y" OR A£="y" THEN 47170
47140   IF A£="N" OR A£="N" THEN 47100
47150   GOTO 47130
47160   '
47170   OPEN RDOC£ FOR OUTPUT AS #1
47175   '
47180   PRINT #1,"KNEADING EXPERIMENT RESULTS"
47190   PRINT #1," "
47195   '
47200   FOR I=NSTART TO NSTOP STEP NSTEP
47205   PRINT #1,"POSITION NO. = ";USING"###";I
47210   PRINT #1," "
47220   PRINT #1,"CONTROL VARIABLES FOR HAND AND ARM"
47225   '
47230   PRINT #1,"PX_CONTROL = ";USING"+####";XKPREC(I)
47240   PRINT #1,"PY_CONTROL = ";USING"+####";YKPREC(I)
47250   PRINT #1,"PZ_CONTROL = ";USING"+####";ZKPREC(I)
47260   PRINT #1,"QP_CONTROL = ";USING"+####";QKPREC(I)
47270   PRINT #1,"QR_CONTROL = ";USING"+####";QKRREC(I)
47280   PRINT #1,"QF_CONTROL = ";USING"+###.#";QFREC(I)
47285   '
47290   PRINT #1," "
47300   PRINT #1,"INPUTS TO FUZZY INFERENCE MECHANISM"
47302   '
47305   PRINT #1,"EQF1_INPUT = ";USING"+###.#";EQ1REC(I)
47310   PRINT #1,"EQF2_INPUT = ";USING"+###.#";EQ2REC(I)
47312   '
47315   PRINT #1," "
```

```
47320   PRINT #1,"INFERRED FUZZY CORRECTIONS (X Y Z)"
47322   '
47325   PRINT #1,"CX_FUZZY    = ";USING"+###.#";FKXREC(I)
47330   PRINT #1,"CY_FUZZY    = ";USING"+###.#";FKYREC(I)
47340   PRINT #1,"CZ_FUZZY    = ";USING"+###.#";FKZREC(I)
47345   PRINT #1," "
47350   NEXT I
47355   '
47360   CLOSE #1
47370   '
47380   RETURN
47400   '
47410   '
47420   '
47500   '
47510   ' ** PADDING EXPERIMENT RESULTS RECORDING **
47515   '
47520   '
47525   COLOR 7,1:CLS
47530   LOCATE 5,20:COLOR 2,4
47540   PRINT"RECORD THE PADDING RESULTS (Y/N)?"
47550   '
47560   A£=INKEY£
47570   IF A£="Y" OR A£="y" THEN 47610
47580   IF A£="N" OR A£="n" THEN RETURN
47590   GOTO 47560
47600   '
47610   COLOR 7,1:CLS:LOCATE 10,15:COLOR 4,2
47615   INPUT"PLEASE INPUT THE DATA FILE *.DOC ";RDOC£
47620   '
47625   LOCATE 12,15:COLOR 2,4
47630   PRINT"IS THE DATA FILE NAME CORRECT (Y/N)?"
47640   '
47650   A£=INKEY£
47660   IF A£="Y" OR A£="y" THEN 47700
47670   IF A£="N" OR A£="n" THEN 47610
47680   GOTO 47650
47690   '
47700   OPEN RDOC£ FOR OUTPUT AS #1
47705   '
47710   PRINT #1,"PADDING EXPERIMENT RESULTS"
47715   PRINT #1," "
47720   FOR J=MSTART TO MSTOP STEP MSTEP
47725   FOR I=NSTART TO NSTOP STEP NSTEP
47730   PRINT #1,"POSITION NO. = ( ";USING"##";J;
47735   PRINT #1," , ";USING"##";I;
47738   PRINT #1," )"
47740   PRINT #1,"  "
47745   PRINT #1,"CONTROL VARIABLES FOR HAND AND ARM"
47750   '
47755   PRINT #1,"PX_CONTROL = ";USING"+####";XPPREC(J,I)
47760   PRINT #1,"PY_CONTROL = ";USING"+####";YPPREC(J,I)
47770   PRINT #1,"PZ_CONTROL = ";USING"+####";ZPPREC(J,I)
47780   PRINT #1,"QP_CONTROL = ";USING"+####";QPPREC(J,I)
47790   PRINT #1,"QR_CONTROL = ";USING"+####";QPRREC(J,I)
```

```
47800   '
47805   PRINT #1,"  "
47810   PRINT #1,"INPUTS TO FUZZY INFERENCE MECHANISM"
47815   '
47820   PRINT #1,"DZ_INPUT    = ";USING"+###.#";DZREC(J,I)
47830   '
47840   PRINT #1,"  "
47850   PRINT #1,"INFERRED FUZZY CORRECTIONS (X Y Z)"
47860   '
47870   PRINT #1,"CX_FUZZY    = ";USING"+###.#";FPXREC(J,I)
47880   PRINT #1,"CY_FUZZY    = ";USING"+###.#";FPYREC(J,I)
47885   PRINT #1,"CZ_FUZZY    = ";USING"+###.#";FPZREC(J,I)
47890   '
47900   PRINT #1, "  "
47910   '
47920   NEXT I
47930   NEXT J
47940   '
47950   CLOSE #1
47960   '
47970   RETURN
47980   '
47990   '
47995   '
50000   ' *************  END OF THE FILE  **************
```

APPENDIX G Publications

1. J. Yan, M.A. El-Baradie and M.S.J. Hashmi "Fuzzy logic based robotic on-line error correction", to be published on the 1st Int. Conf. on Manufact. Tech., Hongkong, Dec. 1991

2. J. Yan, M.A. El-Baradie and M.S.J. Hashmi "The development of a robotic compliance control system", (accepted) Int. J. Machine Tools & Manufact., Jan. 1991

3. J. Yan, M.A. El-Baradie and M.S.J. Hashmi "AI system for the robotic physiotherapic applications", to be published on the Int. Conf. on CIM (ICCIM'91), Singapore, Oct. 1991

4. J. Yan, J.J. Murphy, M.A. El-Baradie and M.S.J. Hashmi "Path planning and compliance control system for physiotherapic applications", Proc. of the 11th Int. Conf. on production research (ICPR'91), Hefei, China, Aug. 1991, PP601-604

5. J. Yan, M.A. El-Baradie and M.S.J. Hashmi " The development of a robotic compliance system", Proc. of Int. Conf. on FAIM'91 (Factory Automation & Information Management), Limerick, Ireland, Mar. 1991, PP729-742

6. J. Yan, M.A. El-Baradie and M.S.J. Hashmi "Modelling and software development of the trajectory control of a robot's hand", Proc. of IMC-7 conf. on Advanced Manufact. Tech. & Systems, Dublin, Aug. 1990, PP249-266