1985

Genetic strategies for interfacing vision systems to robotic controllers /

Chang Pei-Chann
Lehigh University

GENERIC STRATEGIES FOR INTERFACING VISION SYSTEMS TO ROBOTIC CONTROLLERS

By

Pei-Chann, Chang

A Research Report
Presented to the Graduate Committee
of Lehigh University
for the Degree of
Master of Science
in
Industrial Engineering
Lehigh University
May 1985

CERTIFICATE OF APPROVAL

This reserach thesis is accepted and approved in partial fulifillment of the requirements for the degree of Master of Science.

9/20/85
(Date)

Nicholas G. Odrey

Professor in Charge
Dr. Nicholas G. Odrey

Geo E. Kane

Chairman of Department
Professor George E. Kane

8

TABLE OF CONTENTS

CHAPTERS:

ABSTRACT

Applying a robot with a vision system involves an interface between the robot and the vision system. Depending on limitations within the robot, this interface could be a mere start or stop control, or it could be a data communications channel with an external device or higher level computer.

For example, in the PUMA/VS-100 Robot Vision System [1], Brian and Terry try to integrate the system by slaving the vision system to the VAL robot control language. As a consequence, a language, besides a complete computer communication interface has been developed to allow remote control of the VS-100 from a higher level or process-control computer.

The hardware portion of the interface consists of a standard DEC 16-bit bidirectional TTL parallel interface module. The software communication protocol was especially developed to provide reliable high-speed inter-computer communication.

Different schemes for interfacing a vision system to a robot controller have been explored for various applications. In order to get quick algorithm, and efficient methodologies for their implementation, significant intrfacing procedures was studied. The approach taken in this thesis is to fully define the interface between a robot controller and a vision system and develop a list of the types of problems in defing such a interface. A strategy is developed for a generic vision-to-robot interface. The final objective is to come out with a plug

compatible vision system and robot interface which would give greater flexiblity and cost benifits in installing vision-robot based interfacing systems.

CHAPTER 1 INTRODUCTION

Robots have become an integral part of modern industry with thousands of robots in use in United States industry today. Robots have the capability of becoming more useful with the addition of sensors (e.g. vision and force sensors) which allow the robots to make decisions dependent on the feedback of these sensors. With increased sensor capability, robots can be applied to more tasks.

With the advent of robots and increased sensor capability, industries would like to upgrade or integrate their existing manufacturing systems. One requirement for adding a new robot, sensor or machine tool is that the addition must be as easy as adding a printer or hard disk to a microprocessor. Currently, such compability does not exit. It is difficult or nonjustifiable for a company to develop their own interfacing system. Of particular interest are robotic vision system which should have standardized interfaces to robots. Plug-compatible robotic vision systems should be developed such that they can be upgrated in much the same way as a microcomputer is today.

Interfacing vision systems to robots involves the communication between the vision controller and robot controller. The boundary that is shared between these two systems is called the interface (Fig. 1.1). Interfacing may be defined as accomplishing the data transfer across this shared boundary. A vision system uses image processing software to process the image and output the data to the robot through an interface. In order to accomplish a task, a robot must wait for the input from the vision system. Such input data typically consists of information on the position and orientation (pose) of an object. The robot will then feedback a signal to the vision system. This interchange is termed the handshaking procedure. The vision system continues to process the image and extract data for the next cycle application.

```
I-----------I       I------------I       I-------------I
| Vision    |       | Interface  |       |  Robot      |
|Controller |<------>|            |<----->| Controller  |
I-----------I       I------------I       I-------------I
```

Fig. 1.1 General Requirements for Vision
to Robot Communication [27].

In such two way communication, the designer has to develop a detailed understanding of the protocols used and produce both software and hardware to control the data link and perform error checking and recovery functions. The principal feature of communication is that bytes, words, and groups of bytes and words are disassembled and transmitted. Received bits are reassembled into bytes, words, and groups of bytes and words. Both the transmitter and the receiver must agree beforehand on the rules to be used in exchanging information. These rules are generally called protocols.

While attempting to create such a interfacing system, engineers are confronted with many difficult problems such as timing and signal interpretations. An important aspect of the interfacing problem between the robot and the vision system is the planning and preparing of the exchange of signals between the robot and the vision system. Different schemes for interfacing a vision system to a robot have been explored for various application. Various approaches to the interfacing problem by different investigator will be reviewed later.

This thesis has the objective to define a better and, hopefully, generic interfacing methodology between a vision system and a robot. The advantage of a defined methodology would be to provide capabilities that will allow the user to easily implement complex interfaces without large investments of time and labor.

1.1 Components of a Robot and Vision Application System

Vision systems coupled with robots have proved successful in inspection, welding, and parts guidance and assembly. In these applications, a vision sensor, a vision controller, a robot, and the necessary interfacing devices define the overall system. One example is the OPTIMATION vision system coupled with the GE P-50 robot in welding applications [1]. The imaging system for this robot -vision system consists of a CID (Charge Integrated Device) camera which is mounted on the robot's end effector for welding seam tracking. The vision controller, which contains a high resolution monitor for image display, is computer programmed to process the image data and communicate with the robot. The robot itself contains a controller and is attached with the welding equipment. The robot control interface circuits for the camera and the welding equipment facilitate communication between the robot operating program, the vision controller, and a servo amplifier. Basically, as the robot is executing a program, it comes to the location of the center of the seam to be welded. Deviations of the actual seam beyond present limits from the nominial or anticipated path result in an error signal to the robot servo system. This results in an updated signal several times per second to the robot.

Fig 1.2 is a block diagram of the OPTOMATION TM and GE P-50. The system is primarily controlled by two INTEL 8086 microprocessors. Primary system interfacing is accomplished over a INTEL-Multibus TM while high speed data flow utilizes custom high speed busses. A detailed description of this system will be discussed in chapter 3.

1.2 The State of the Art

There have been many successful systems in robotics vision, especially in the areas of inspection and determination of the positions and orientations of objects. Still, little has been done in the areas that require recognition and visual interpretation of a scene. According to Takeo Kanade [2], most of the limitation in robotics vision systems are:

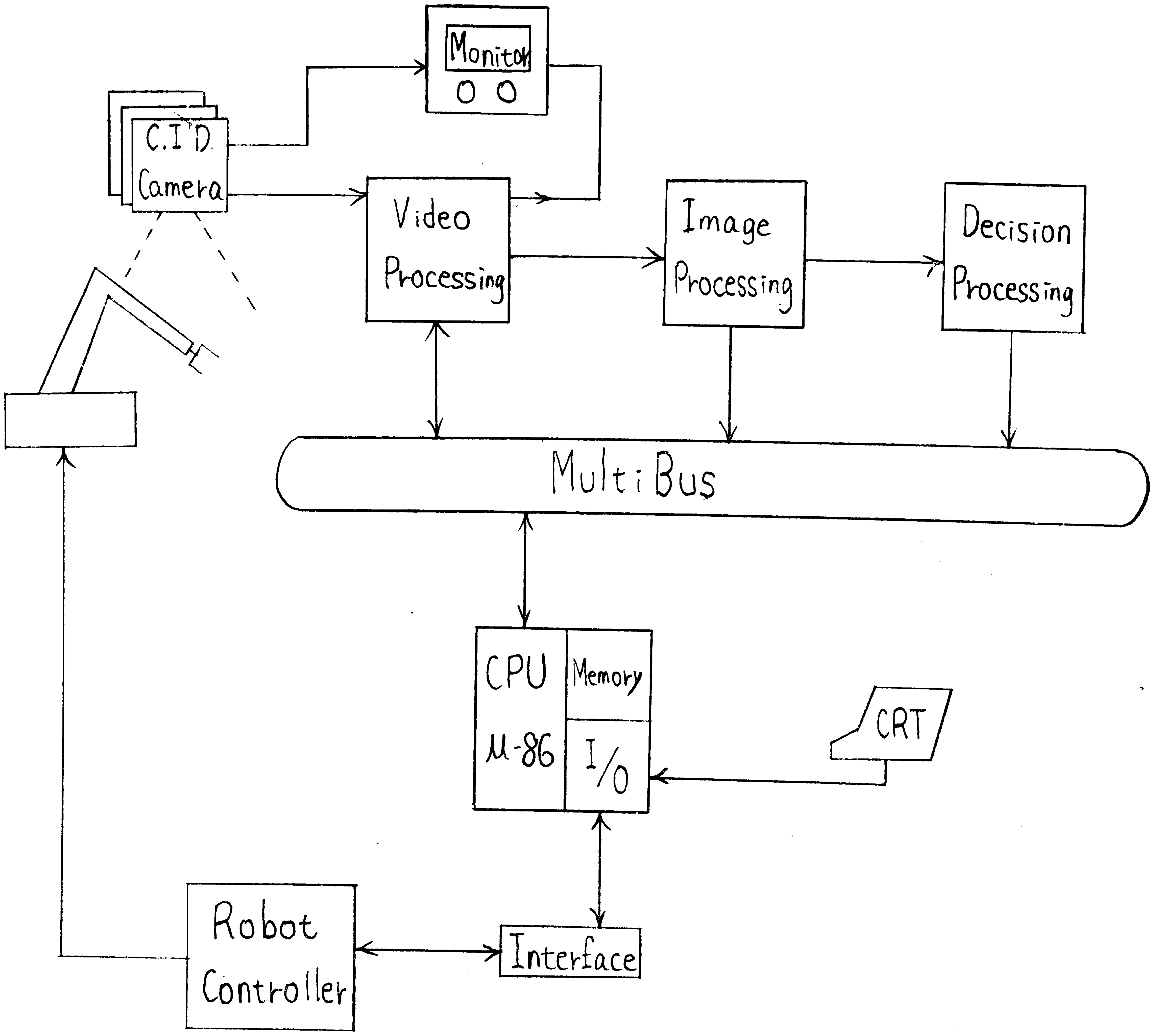


Fig. 1.2 Block Diagram of the OPTOMATION TM Vision System and GE-P50 Robot [1].

1.) 2-D rather than 3-D: Most of the tasks are inherently two dimensional, such as printed circuit board inspection or recognition of 2 1/2-D shape (shape that can be represented by a flat pattern plus a depth coordinate).

2.) Binary vision: Binary image analysis consists of two-dimensional region analysis (topological and geometric properties of a region, such as connectivity analysis, hole identification, and corners). Binary vision was developed to meet not only performance requirements in speed but also cost requirements.

3.) Limited number of rigid objects in the model: Since the shapes of objects are predetermined without variation and articulation, the variations in their appearances are limited.

4.) Engineering setup: Limited and standard view angles (mostly directly overhead views) are used in vision analysis. Careful lighting (such as back-lighting and projected plane of light), ensures high-contrast, shadow-free images.

5.) Special-purpose rather than general-purpose: In current vision applications, the type of scene to be processed and acted upon is usually carefully defined and limited to the capability of the machine.

6.) No standards exist in vision to robot interfacing: Various robot and vision systems commercially available in the market, (re: the vision vendors in Table 1 and robot vendors in Fig 1.3), do not provide an easy and effective means to integrate a robot and a vision system. There is a need for standards to be agreed upon among the robot and vision system manufacturers.

Techniques used in current robotic vision systems are useful and essential for successful industrial vision systems, but a more fundamental understanding of vision processing is necessary to create robots that see and interpret environments. Many research efforts are being devoted to image understanding, the application of artificial intelligence techniques to image understanding, and in general, the development of high-level industrial vision system.

1.3 Objective of the Research

Because the development stage of industrial robots with vision system is still in its infancy, to date there has been no standardization in the interface of vision systems to robots. Designers of such interfaces typically rely on their training and intuition. Constraints in such interface design are the hardware and software supported by a vision system.

The purpose of this research is to investigate the interface of vision systems to robot controller. The following aspects of the interfacing problem are considered in this investigation :

1) Physical interface (hardware)

2) Data transfer protocol (software communication)

3) Calibration Software

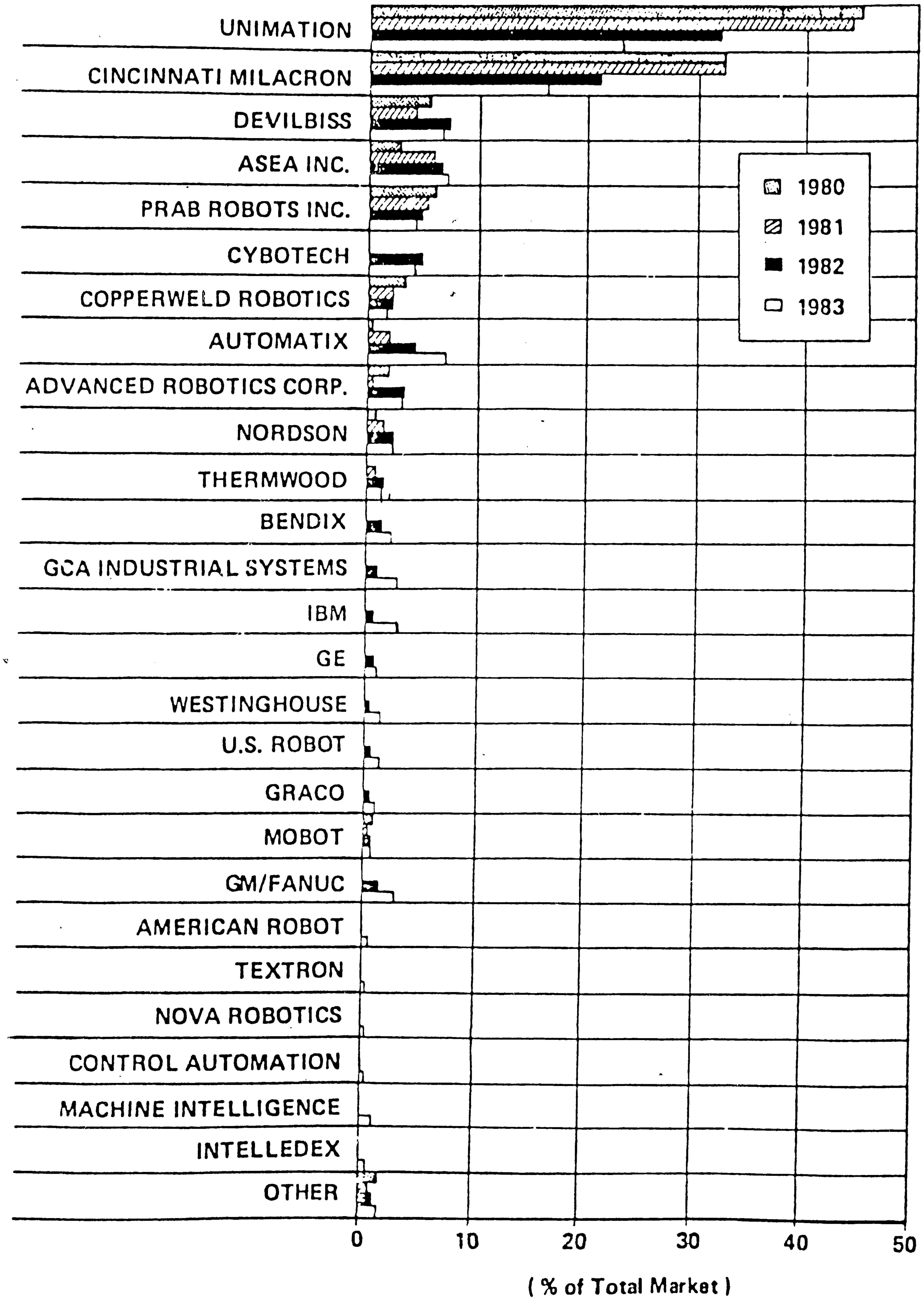
4) Coordinate Transformation

Table 1

REPRESENTATIVE COMMERCIAL ROBOTIC MACHINE VISION SYSTEMS

COMPANY	MODEL	GREY SCALE	PIXELS	SPEED
1. AUTOMATIX	AUTOVISION II	16	256X256	360 PARTS/MIN
	AUTOVISION IV	256	512X256	1800 PARTS/MIN
2. UNIMATION DANBURY,CONN.	UNIVISION I	2	256X256	LESS THAN 500 MILLISEC./FRAME
	UNIVISION II	2	256X240	56 MILLISEC. /FRAME
3. VISION PERIPHERALS ANAHEIM,CALIF.	PIXELCASTER	16	1024	
4. OCTECK,INC. BURLINGTON, MASS.	20/20 VISION DEVELOPMENT SERIES	16	320X240	300PARTS/MIN.
5. GENERAL ELECTRIC SYRACUSE,N.Y.	OPTOMATION II	256	244X248	
6. ELECTRO-OPTIC INTEGRATED SYSTEMS CARLSBAD, CALIF.	ROBOTS			

*COURTESY OF INDUSTRIAL ROBOTS AND ROBOTICS [10].



* Market share percentages are based on mid-points of company and industry estimates.

Fig. 1.3 Robot Vendors in the Market [10]
Source : Bache Halsey Shields, Inc.

CHAPTER 2

LITERATURE REVIEW & STATEMENT OF THE PROBLEM

There are hundreds of robot to vision application systems currently in use in the United States. Major areas of industrial application tasks range from detection of major flaws such as missing parts to detection of small defects, misalignments, size measurement, subtle color changes, and so on. For humans, these tasks tend to be dull, and their repetitiveness leads to decreased performance.

Successful practical applications require fast processing, inexpensive hardware, and high reliability. To simplify the image-analysis problems, lighting may be controlled to give high-contrast images. Major successes have been achieved in the inspection of electronic printed circuit boards (PCBs) and integrated circuits (ICs). A direct approach is to compare the images of the images of the patterns to stored images of defect-free patterns on a pixel by pixel basis. However, difficulties are caused by the variability of the images, alignment errors, changes in size, and so on.

The very nature of a robot system dictates that it must work with other equipment or parts in a manufacturing system. As such , a robot must be interfaced to the other sensors. Different schemes for interfacing vision systems to robots have been explored for various applications.

2.1 Typical Robots to Vision Application Systems Currently in Use

In order to obtain fast algorithms and efficient methodologies for algorithm implementation, significant interfacing procedures must be developed.

For example, in the PUMA/VS-100 Robot Vision System (Fig. 2.1), Brian and Terry [3] attempted to integrate the system by slaving the vision system to the VAL robot control language. As a consequence of their efforts, a number of vision related instructions were added to the VAL language. In addition, a complete computer communication interface was developed to allow remote control of the VS-100 vision system from a higher level or process-control computer.

The hardware portion of the interface consisted of a standard DEC 16-bit bidirectional TTL parallel interface module. The software communication protocol was especially developed to provide reliable high-speed inter-computer communication.

Another example of vision system to robot interface is CO-SIGHT (Fig. 2.2), a practical vision-based robot guidance system developed at GM research laboratory by Mitchel and Lothar [4]. They partitioned the vision system, the robot and the monitor into independent subsystems to obtain a modular organization. Intersystem communication was designed to allow easy substitution of new subsystems with minimum impact. The communication between the robot and vision was through the monitor system PDP 11/34 computer operating under the RSX-11S real time exectuive.

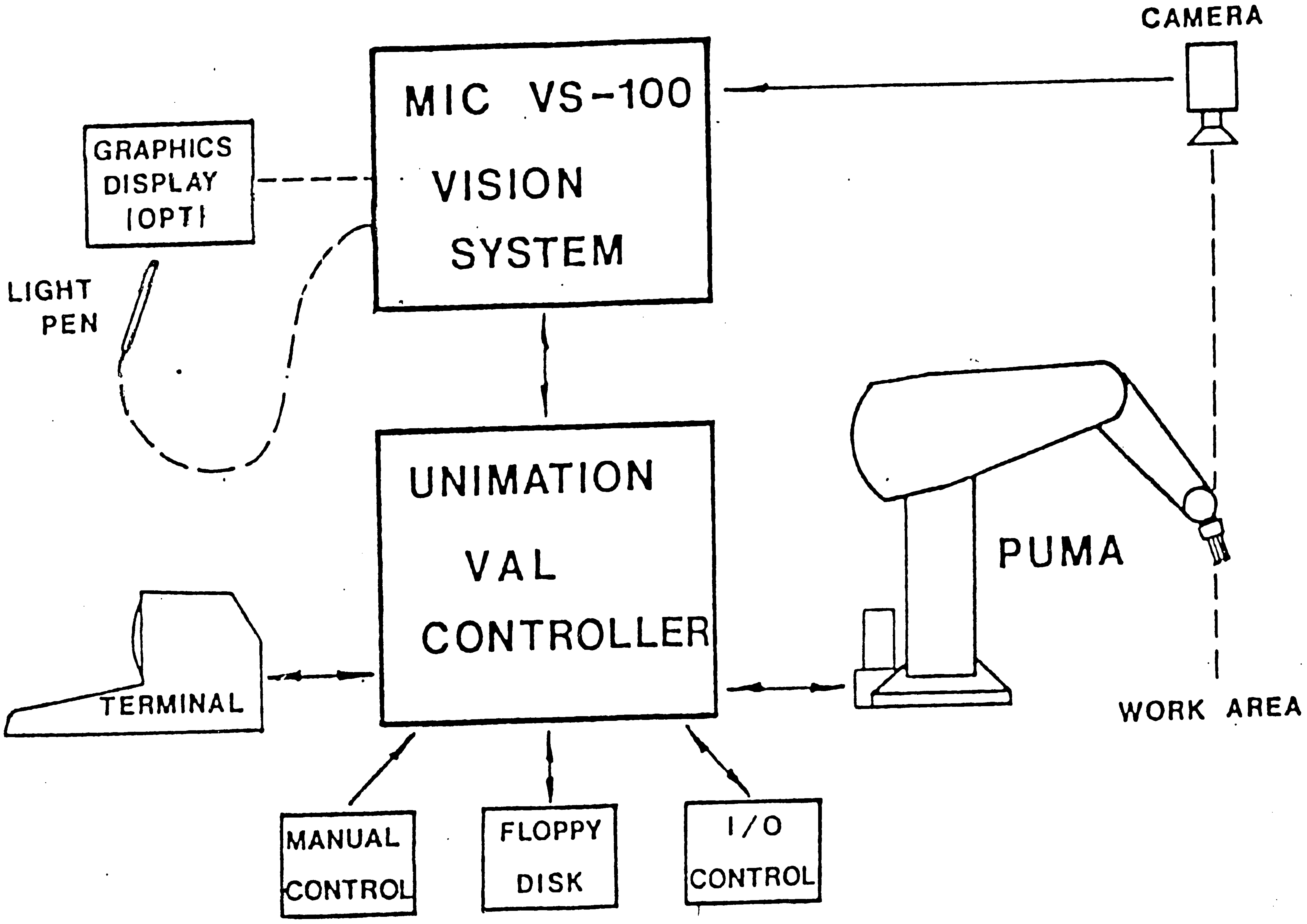
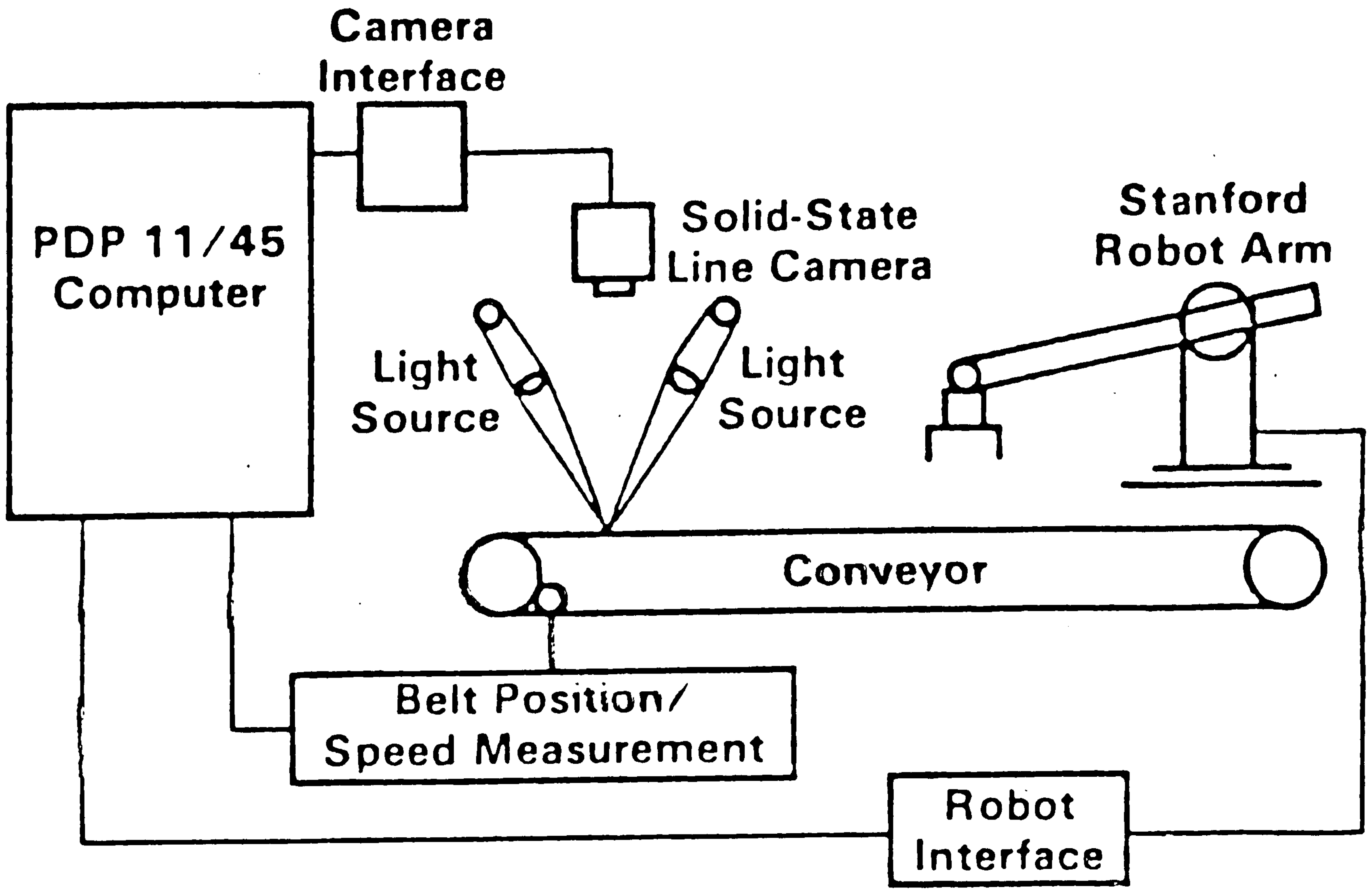
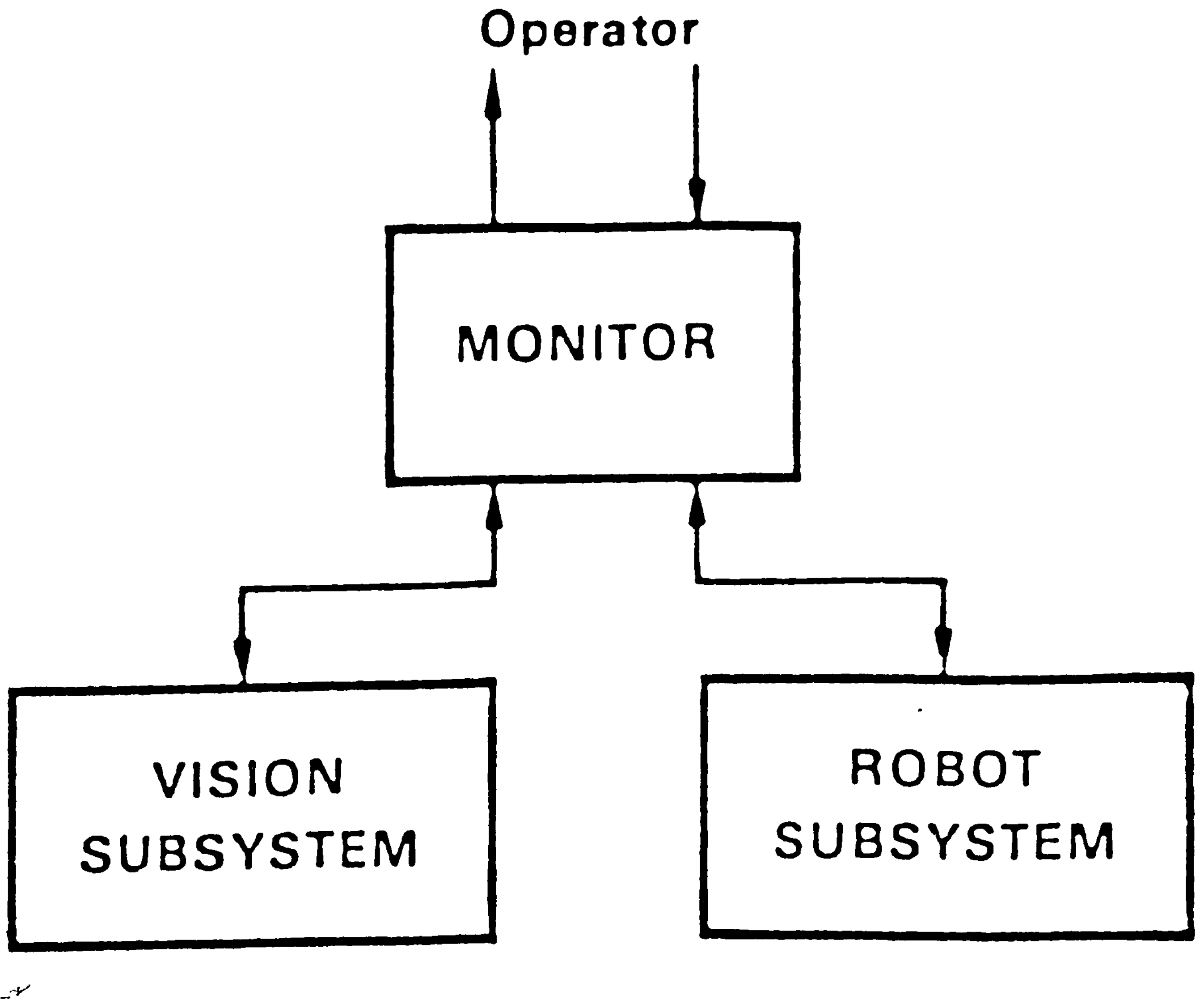


Fig. 2.1 PUMA / VS 100 Vision System [3].



CONSIGHT-I hardware schematic.



CONSIGHT-I software organization.

Fig. 2.2 COSIGHT-1 A Vision-Controlled Robot System [5].

Baird describe a system developed at General Motors for orienting IC chips correctly before they are bonded [5]. The orientation is basically determined by a histogram of edges detected in the image. Other systems for orienting IC chips are described by Yachida , Tsuji and Horn in reference [6,7]. Some of these systems are reported to be in large scale production use.

Agin has described laboratory experiments at detection for flaws in castings such as missing or incorrectly dimensioned holes. and inspection of industrial objects [8]. The orientation of the parts is constrained so that perspective variations are avoided.

Another important area of applications in industrial robotics vision system is in materials handling. Here, the parts are usually stacked in a heap or in a bin with other similar or different parts. A part needs to be identified and grasped by a mechanical manipulator at the appropriate pick-up points. Partial success has been obtained in such applications if the possible orientation of the parts are restricted and unoccluded or else occluded in small areas only [9].

2.2 General System Architecture of Vision system and Robot controller

A.) The Vision System

The general architecture of a universal image processing system [10] can be schematically drawn as given in Fig. 2.3. The system depicted must perform several operations and processes such as :

1. converting a physical image into an electrical video-signal using an imaging device such as a vidicon camera;
2. quantizing this video-signal into a digital form;
3. storing the digitized picture in memory;
4. communicating with the computer system for image analysis
5. outputing data manipulation results for use by the robot.

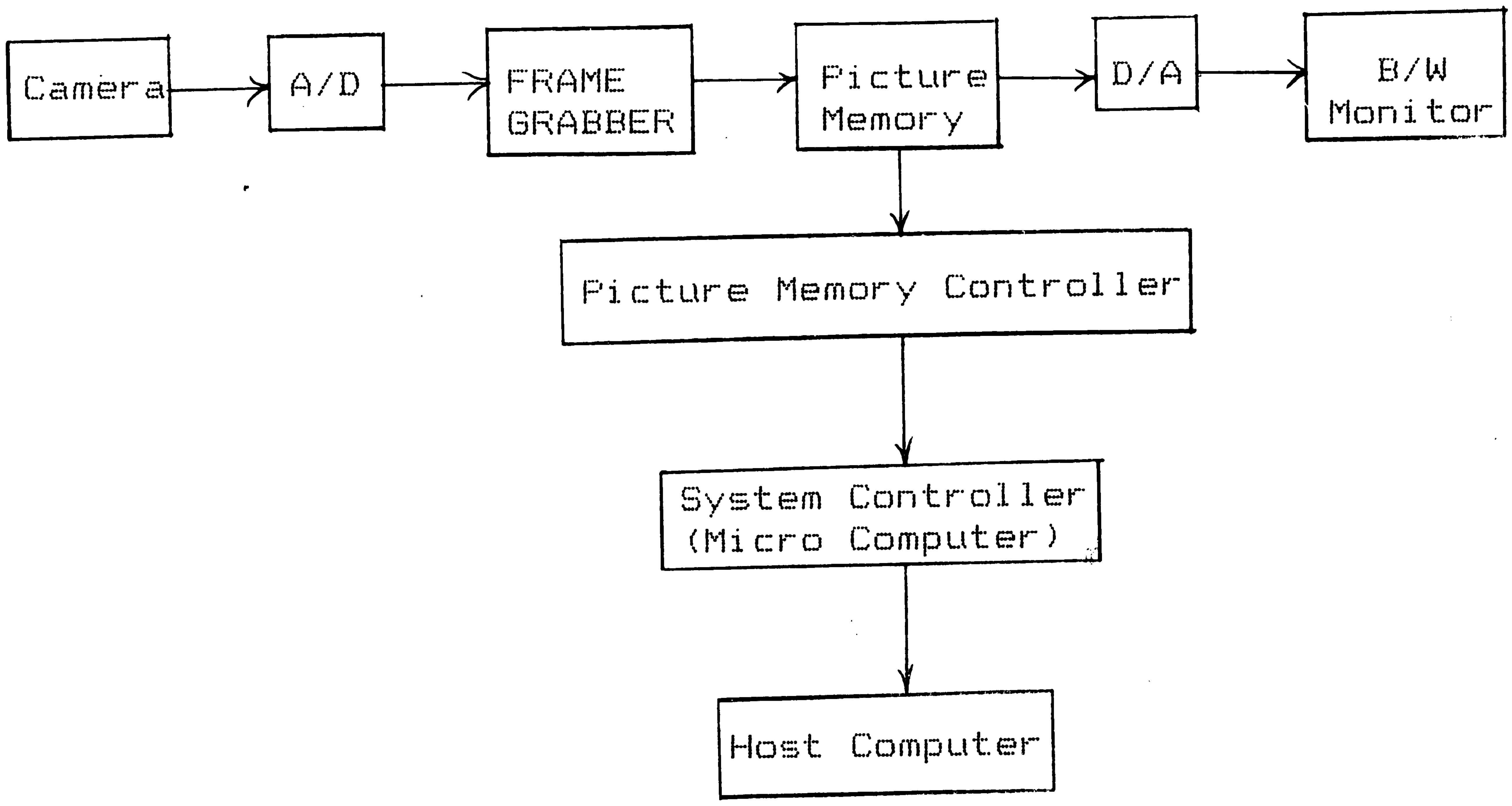


Fig. 2.3 General Architecture of a Vision System [24].

The central components of the architecture are the computer in which all processing is done, and the image device. Computer peripherals enable one to input/output the necessary data. Image memories generated are partially contained in the processor and partially used as picture input/output buffers.

Depending on the type of system, the vision controller provides serial I/O ports to communicate with other peripherals or controllers. In the Optical Recognition System ibot-1 controller, the rear panel includes the line power cord, the video input and auxiliary monitor output connectors. The RS 232 C connectors handle communications among the system CRT terminal, the robot interface, 2 auxiliary ports for accessory data manipulation devices, printers, and the optional disc operating system chassis.

b.) The Robot System

A representative industrial robot system (Fig 2.4) is made of three main parts : the control system, a measuring and servo system, and a manipulator [11]. The control system consists of the computer, memories, inputs and outputs to control the robot, interlocks from peripherial equipment, and functions to control the servo system of the robot.

The measuring and servo systems include servo amplifiers and DC motors with tachogenerator feedback. Position regulation is by means of a cyclic resolver system, consisting of a resolver with associated supply and decoding circuits, and a position

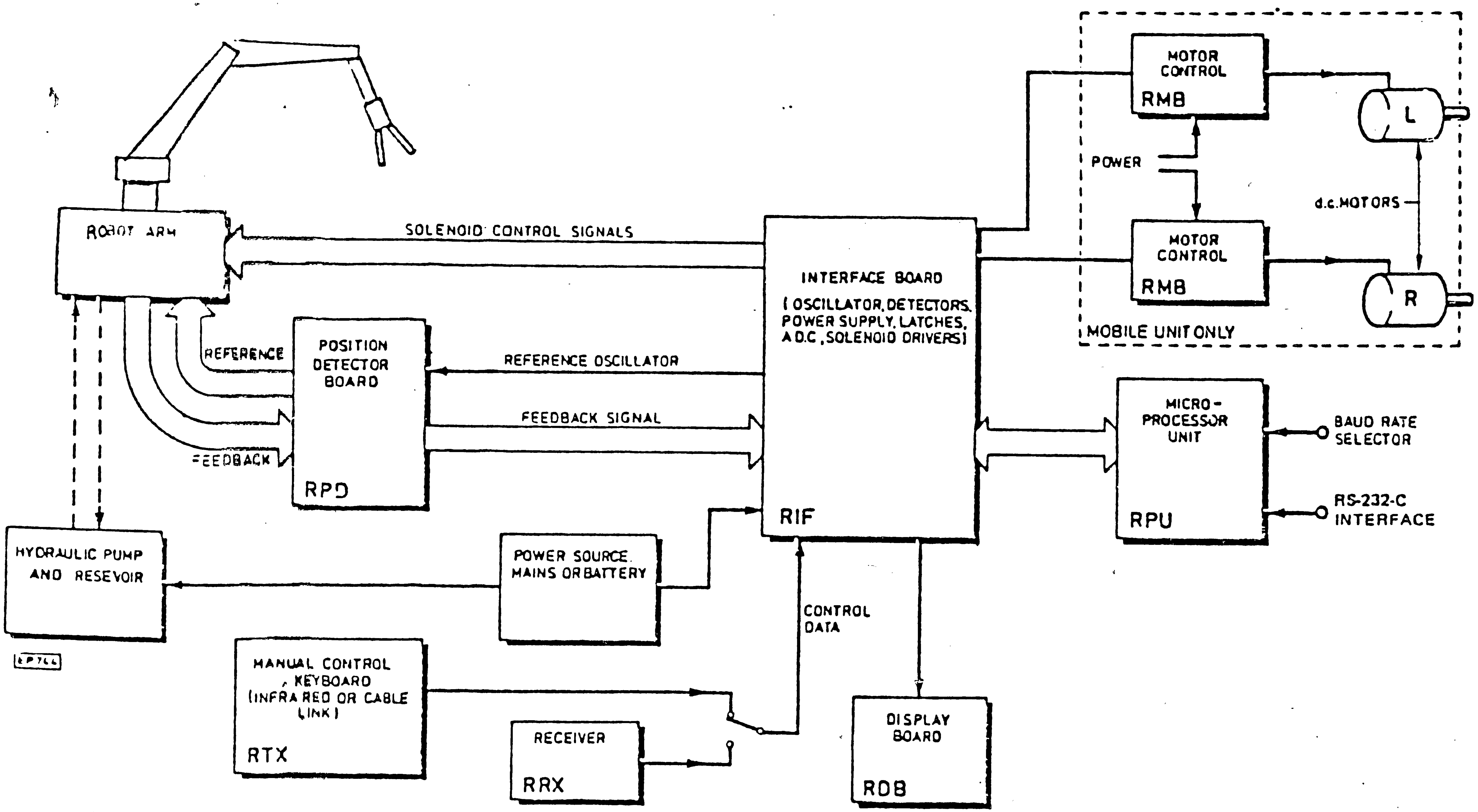


Fig. 2.4 The Representive Architecture of a Robot Control System
Courtesy of Feedback Inc., Berkeley Heights, N.J. [10].

regulator. The mechanical system includes the robot and the transmission that converts the rotation of the motors into the required motion. Robot manufacturers use microprocessor-based systems in the control system.

The heart of the control system is the microprocesso' unit, which performs the task of overseeing the complete system and issuing control signals in response to arm positional data, programmed-in commands, commands from the control box (teach pendant), or information from an external computer. The microprocessor memory stores the positions of the arm, and by using servo techniques ,instructs the robot to repeat the sequence of programmed tasks.

For example, the PUMA 560 controller [12] consists of the DEC LSI-11 computer with DRV11 and DLV11-J interface boards and RAM and EPROM boards. The LSI-11 system is a standard DEC unit that contains a processor (LSI-11/2), memory, and communication boards. System software (VAL) resides in erasable, programmable read-only memory (EPROM). User program information is stored in random-access memory (RAM). Communication between the processor and other components is accomplished as follows:

1. DLV11-J is a four port asynchronous serial I/O board. It is the link between the processor and terminal, teach pendant, and floppy disk.
2. DRV11 is a unit that provides parallel-line communications to and from the digital servo boards and links the processor to the interface board.

2.3 Vision to Robot Interfacing Modules

The complexity of interfacing vision systems to robots has led to various interfacing methods [1, 3, 4, 13, 14, 19]. These methods can be classified as follows :

1. Robot Dependent Method (Vision - Master, Robot - Slave)
2. Vision Dependent Method (Robot - Master, Vision - Slave)
3. Integrated System

2.3.1 Robot Dependent Method (Vision - Master, Robot - Slave)

The architecture of a robot dependent system is given in Fig. 2.5. The characteristics of this type of system include :

a. Vision-Oriented

Most of the operation features are menu driven and under the control of the vision controller. The controller provides the options of a set-up mode, a training mode, and a calibration mode or working mode.

b. Terminal Emulation

When direct control of the robot is required, the vision controller can emulate a terminal and command the robot to download or upload programs.

c. Commands Transfered

In the vision system task execution mode, control for the robot can be achieved through the output of an ASCII code string from the vision controller to the robot controller.

This procedure continues interactively until the task requirements have been satisfied. The robot , based on information feedbacked the vision system, executes the program to satisfy the task requirements.

The advantage of the Robot Dependent Method is that for each different application, the vision processing software can be modified very easily.

The disadvantage of the Robot Dependent Method is that the communication relationship between a vision system and a robot is fixed. If a different type of robot is applied, then the transformation and calibration softwares must be changed. This method is very inflexible and robot dependent. In an automated manufacturing system, this interfacing method may prove to be inconvenient and financially inadvisable in attempts to interface different vision systems to different robots or vice versa.

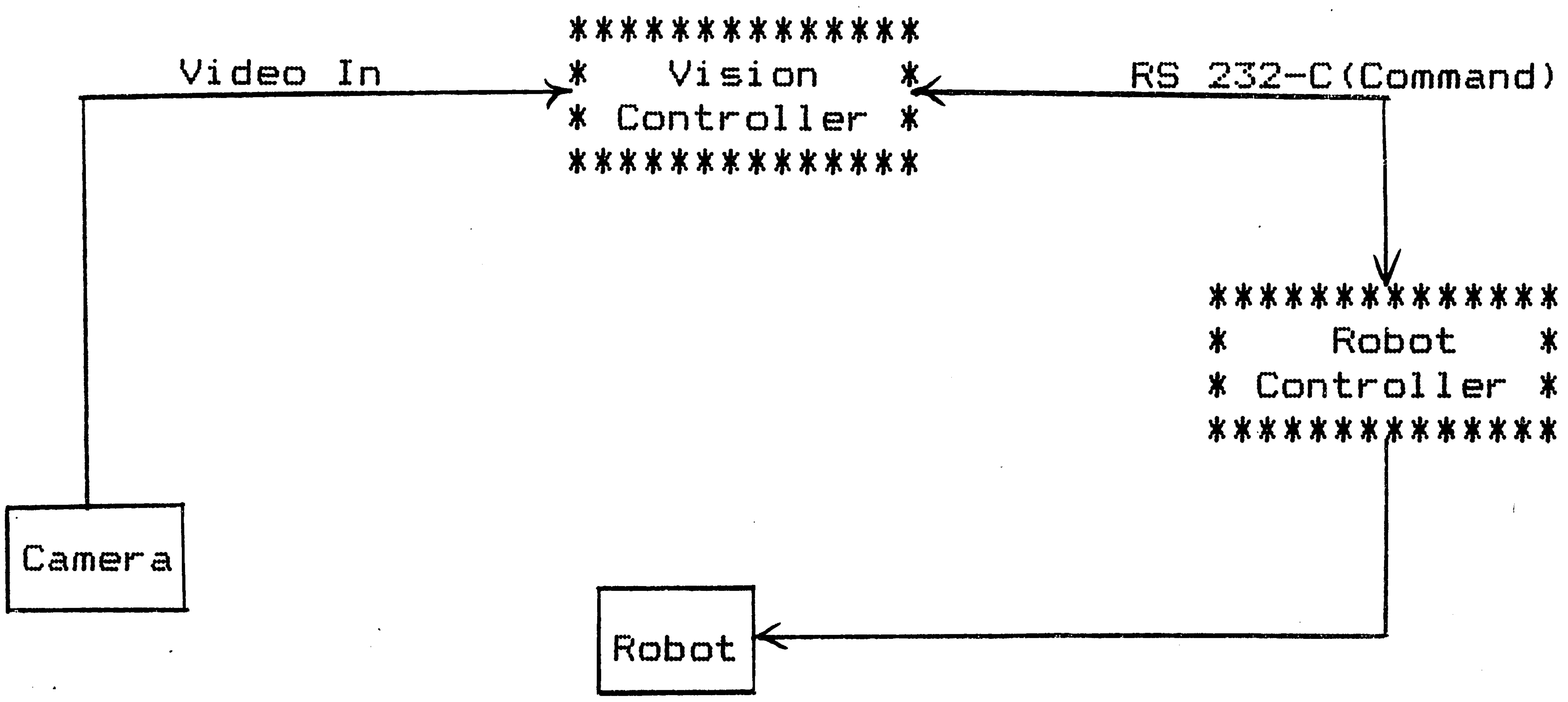


Fig. 2.5 The Architecture of a Robot Dependent System

2.3.2 Vision Dependent System (Robot - Master, Vision - Slave)

The architecture of the vision dependent system is depicted in Fig. 2.6.

The characteristics of this type of system include :

a. Robot-Oriented

The vision system is trained and operated through the robot controller by a number of vision related instructions. These instructions permit the user to calibrate the system, train vision prototypes, identify and locate workpieces, and store or load vision data from a floppy disk unit. A Vision Dependent System is also menu driven and includes the camera set-up, calibration, object training and working modes.

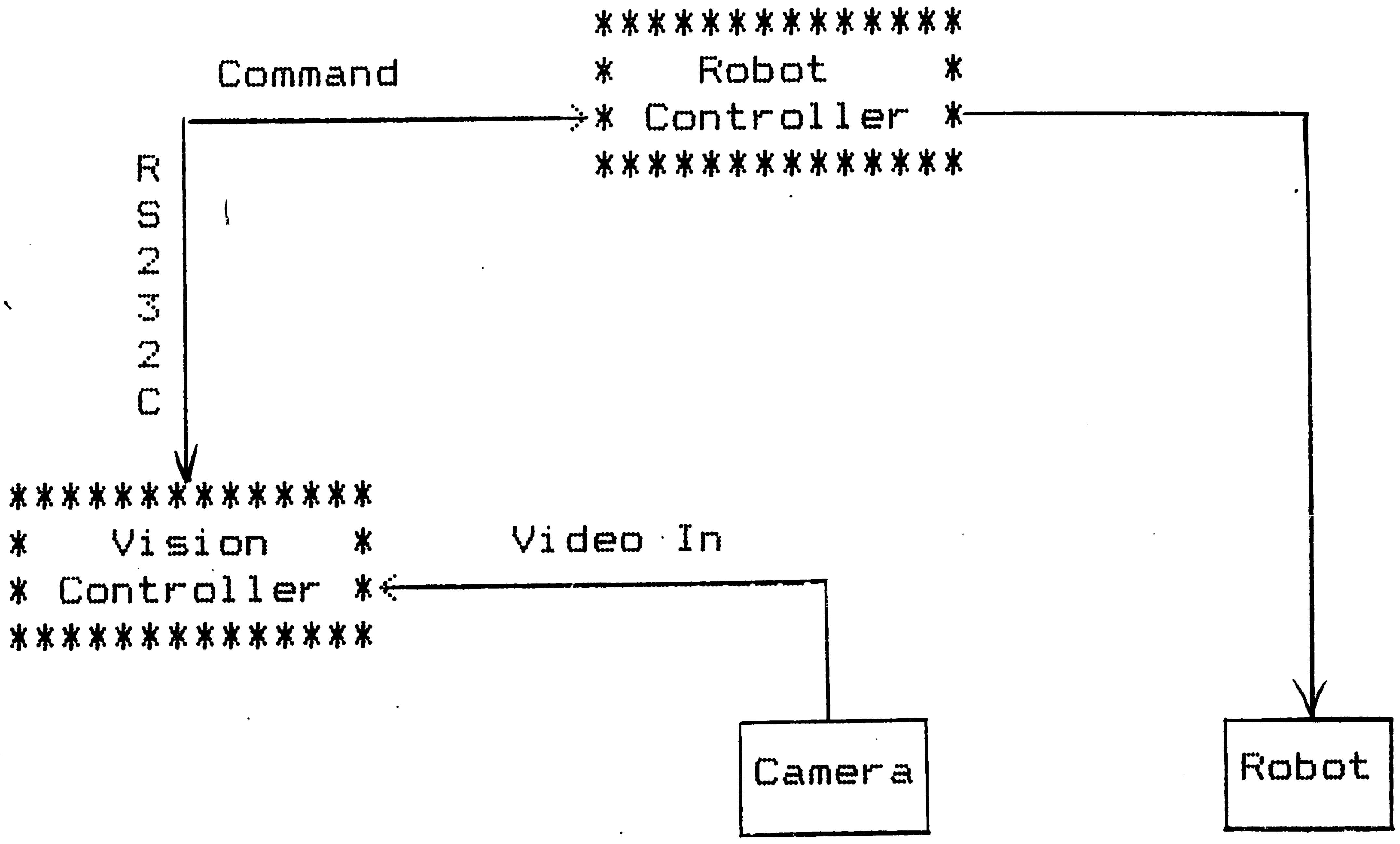


Fig. 2.6 The Architecture of a Vision Dependent System

b. Robot Control Language

The control language of the robot in a Vision Dependent System must be designed to operate the robot and to readily communicate with other computer-based system such as vision and force sensors. Not all the robot control languages have the capability to command vision and force sensors. Fig. 2.7 is a listing of the functions of the various robot control languages. Only those robot control languages which are capable of communicating with other sensors can be applied in the Vision Dependent System.

c. Robot-Vision Interface Instruction

The instructions for vision interface control must be included in the robot language. Typical commands in VAL are PICTURE, LOCATE, FINDHEAP ... etc.

The advantages of a Vision Dependent System are as follows :

1.) The communication between the vision system and the robot in a Vision Dependent System is much easier to implement than in a Robot Dependent System. There is no need for any special software to handle the transfer of robot commands between the robot controller and the vision controller.

2.) From the standpoint of programming, a Vision Dependent System is much easy to program and to modify than a Robot Dependent System. This results simply from the fact that one can build up the vision commands inside the task execution programs.

The disadvantage of a Vision Dependent System is that once the vision system is changed, there is no guarantee that the new system can understand the PUMA's VAL commands, such as PICTURE, LOCATE or FINDHEAP ...etc.
Furthermore, the data transfer protocols and calibration software must be modified to match up the communication procedure between the robot and vision system.

2.3.3 Integrated Robotics Vision System

The architecture of an Integrated Robotics Vision System is given in Fig. 2.7. The system employs a distributed hierarchical control to separate the functions of data gathering and data processing.

In the case of the vision system, there are two levels of control - a system controller in the upper level, and the vision processor in the lower level. The system controller is responsible for the supervision of the total system, the processing of the data obtained through the vision system, and coordinate transformation and communication with the robot controller. The lower level controller, vision processor itself, performs data gathering, image analysis, part identification and orientation.

The function of the system controller in an Integrated System is to command the vision system to gather data and then to record and make decisions based on this data.

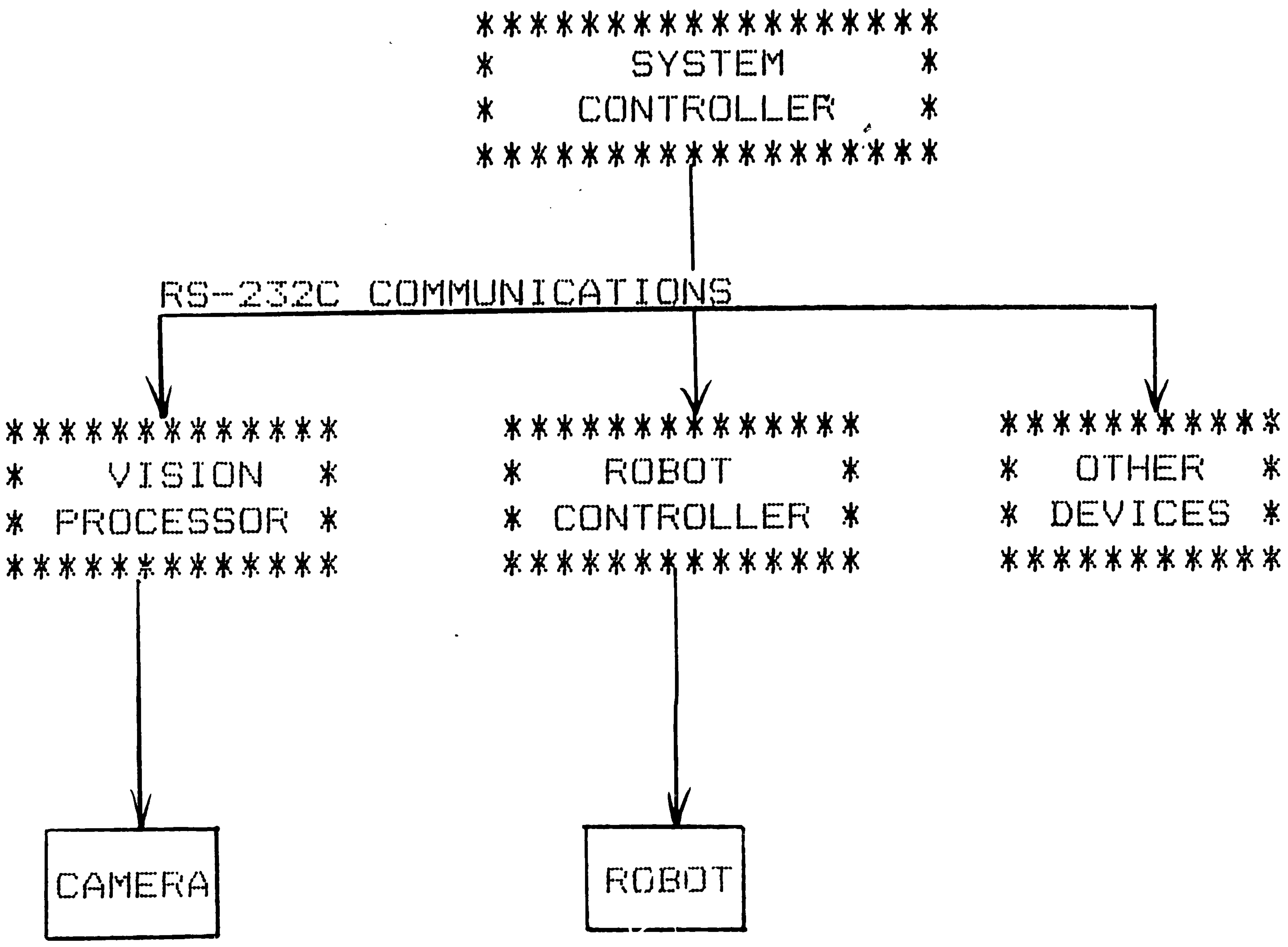


Fig. 2.7 The Architecture of an Integrated System

The vision system is configured as a peripheral in the operational sense, and makes no decisions or value judgements itself. The vision system operats strictly as a data gatherer in response to commands from the system controller which does process control operations such as bin-picking ,welding or assembly.

A major disadvantage of the Integrated Robotics Vision System is the cost, since another computer is needed for system controlling. However, the system controller in an Integrated System can much more easily and flexibly control the input signals from the external sensors than other systems. Among some compatible systems, the robot and the vision system can be exchanged freely. Under such condition, users do not need to develop their communication and transformation softwares. Such a system could be further developed to be a robot and vision independent standard modular system. In chapter 5, such a proposed standard modular system will be presented in detail.

2.4 Statement of the Problem

It is necessary to integrate vision and other sensors to robots if we wish to use robots in an unstructured environment. The concentration in this thesis is to investigate the vision to robot integration. Most commercially available robot manufacturers have not provided an easy and effective means to integrate vision systems with robots. This has resulted in various interfacing methods such as the Robot Dependent System, the Vision Dependent System and the Integrated Robotics Vision System previously mentioned. Characteristics of the interfacing techniques currently used are as follows :

1. Data transfer between a vision system and a robot controller is carried out by a serial asynchronous communication procedure. Most real-time control applications can be accomplished with a baud rate 9,600 bits per second.
2. Master-Slave interfacing method : Owing to the development of data communication and local area networks, different types of data transfer protocols have been designed. Designers in interfacing Vision systems to robots can utilize these

protocols to implement a system very easily. The master-slave method (hierarchical control algorithm),(regardless if the robot or the vision system is the master), provides convenient communication protocols.

3. Since no standards for interfacing vision systems to robots exist both at a high-level programming language or in a robot control language, the information exchange between these two systems place a tremendous burden on the designer. Image processing softwares can be written in Pascal, C or Fortran, while the robot control languages can be in AML, VAL, RAIL ... etc. The data transfer protocols between these two systems are solved by sending the robot commands back and forth through the RS-232C communication line.

4. Bi-directional data communication : The interface for vision systems to robots is a two-way communication which includes control signals, serial or parallel data, and addressing information. For example, in vision applications the robot controller signals the vision system when a robot needs to know the orientation of the part from the vision controller. This procedure continues interactively until the task has been finished.

With the increased requirements placed upon robot vision application systems, there is a need to further develop the technology for interfacing and controlling these systems.

The objective of this thesis is to define the vision to robot interface problem and attempt to develop a systematic method for integration of external sensors with commercially available robots. This should allow more efficient development of automated systems with a performace capability which extends beyond that of a robot alone.

CHAPTER 3 EQUIPMENT

This chapter describes equipment used for the research in this thesis. A brief description of the function and architecture of each system is presented in the following sections.

3.1 ORS i-bot-1 Vision System

Application Description

"i-bot", a grey-scale vision system developed by Optical Recognition System Inc. of Princeton, N.J. [14], can be used to enable industrial robots to grasp a randomly oriented part from a bin, orient the part, and transfer the part to a fixture or receiving workstation.

This vision system is very flexible with a wide variety of applications. A 64 level grey scale frame grabber enables good accuracy and the capability of image processing. The system can operate at speeds approaching a video frame rate of 30 frames per second.

The i-bot bin picking vision system will operate with a wide variety of objects having a degree of cylindrical and spherical symmetry. When used in conjunction with standard grippers, objects ranging in size from 1" to 8" in length and 1/4" to 1 1/2" in diameter can be acquired from a bin of up to 30" x 30" x 12" deep, when the camera is mounted 6' above the bin.

System Description

The i-bot 1 Vision System consists of a video imaging system, a vision controller, a gripper or end effector controller, and a robot control interface, as shown in Fig. 3.1.

1). Video Imaging System

The i-bot 1 video imaging system uses a NEC(National Electrical Company), model VCI B106, standard, solid state, charge-coupled device, matrix array camera, with an industry standard EIA RS-170 electrical output signal. The solid state matrix array is 280 (vertical) by 340 (horizontal) pixel resolution a separate photodiode.

2). Vision Controller

The vision controller contains an Intel Multibus card cage,power supplies, video monitor, front panel indicator drivers, and a cooling fan. The card cage contains Multibus type plug-in circuit cards necessary for the system to function. The heart of the system is an ORS modified Intel 8086 microprocessor with an attached hardware Math Module. Also included are the Intel Input/Output expansion board, and a 256 K random access memory board. For firmware based systems, an EPROM board is also provided. The frame grabber, the optional hardware histogram and the system board are located on separate boards. The system board contains 2K of battery backup CMOS RAM (used to boot the microprocessor and store calibration and training data).

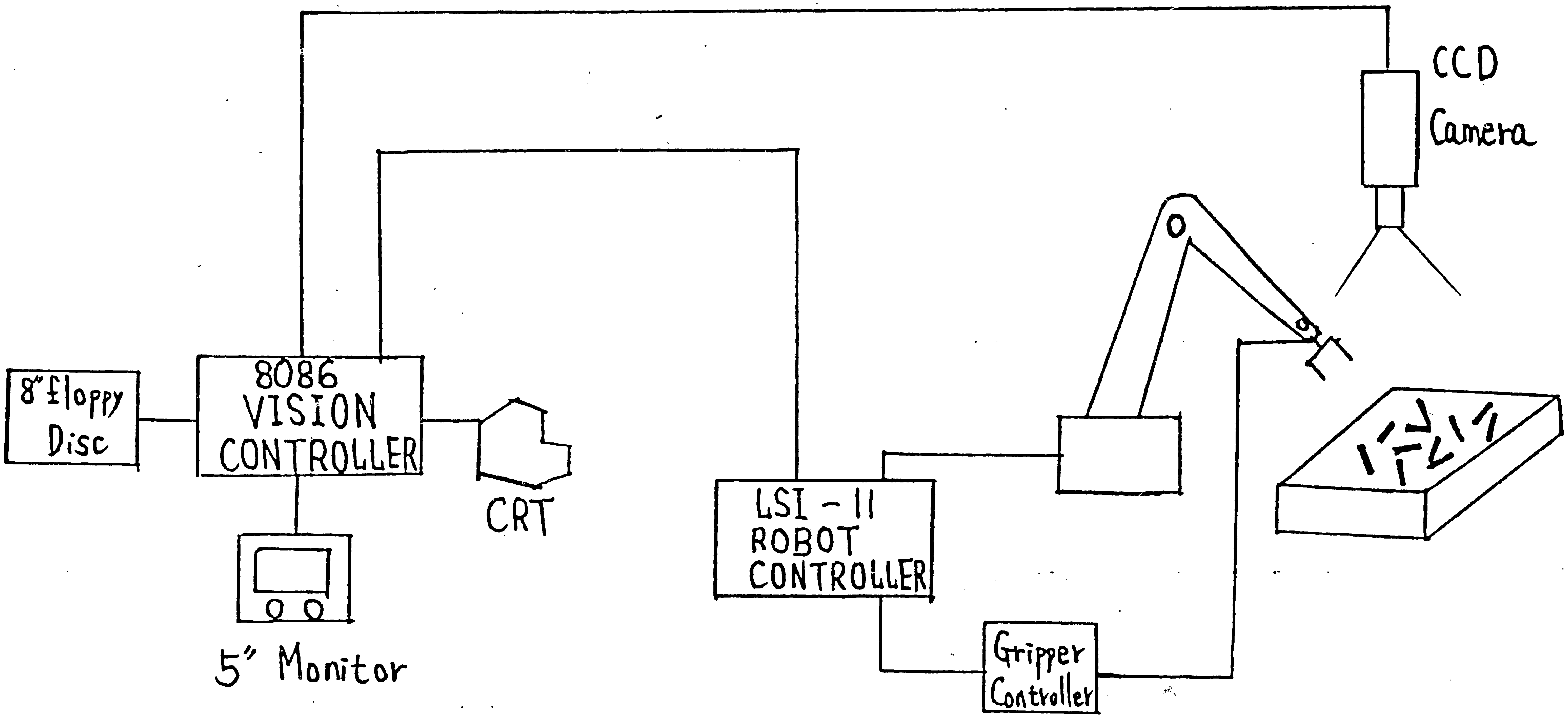


Fig. 3.1 System Diagram of I-bot 1 Vision System

3). Gripper or End Effector Controller

The CCD camera used with the i-bot vision system provides two-dimensional (planar) information of an image. The gripper provides a feedback signal to the robot to allow it to sense the third dimension, depth. The vision controller directs the robot to the proper location above the part in question, and the robot begins to move down towards the part. When the gripper jaws surround the part (as sensed by a light-emitting diode (LED) imbedded on its fingertips) , its controller directs the robot to stop and closes the jaws. The pneumatic gripper requires mechanical adjustment for each object size to be acquired.

4). Robot Control Interface

The robot control interface or input/output module is supplied by the robot manufacturer and requires that the i-bot/vision system generate proper handshaking signals. This interfacing capability is built into both the i-bot i vision controller and the pneumatic and electronic gripper controllers. The i-bot vision controller includes a serial port which operates at a maximum rate of 19.2 K baud.

3.2 Unimation PUMA 560

The Unimate PUMA 550 and 560 Series Robots (Fig. 3.2) are computer controlled robot arm systems manufactured by Unimation Inc., Danbury, Connecticut. The PUMA robot system is designed to adapt to a wide range of applications. The basic units are the teach pendant, software, controller, peripherals, and robot arm.

The system software that controls the robot arm is called VAL. The VAL software is stored in the computer memory located in the controller, which also houses the operating controls for the system.

To teach the robot arm, either one of two procedures can be used. The teach pendant may be used to manually direct the movements of the robot arm through each step of the task. The second method is to write a program using VAL instructions.

In either case, the controller transmits the instructions from the computer memory to the arm. Position data obtained from incremental encoders and potentiometers in the robot arm are transmitted back to the controller/computer to provide closed-loop control of the arm motions.

The basic units of the PUMA System are described as follow :

1). Software: The PUMA System operates on a high level language called VAL. In addition to being a sophisticated programming language developed for assembly, VAL is a complete robot control system.

2). Controller: The main internal components of the controller are listed as follows;

- Unimation interface board
- Digital servo board
- Clock/Termination board
- Input/Output interface board

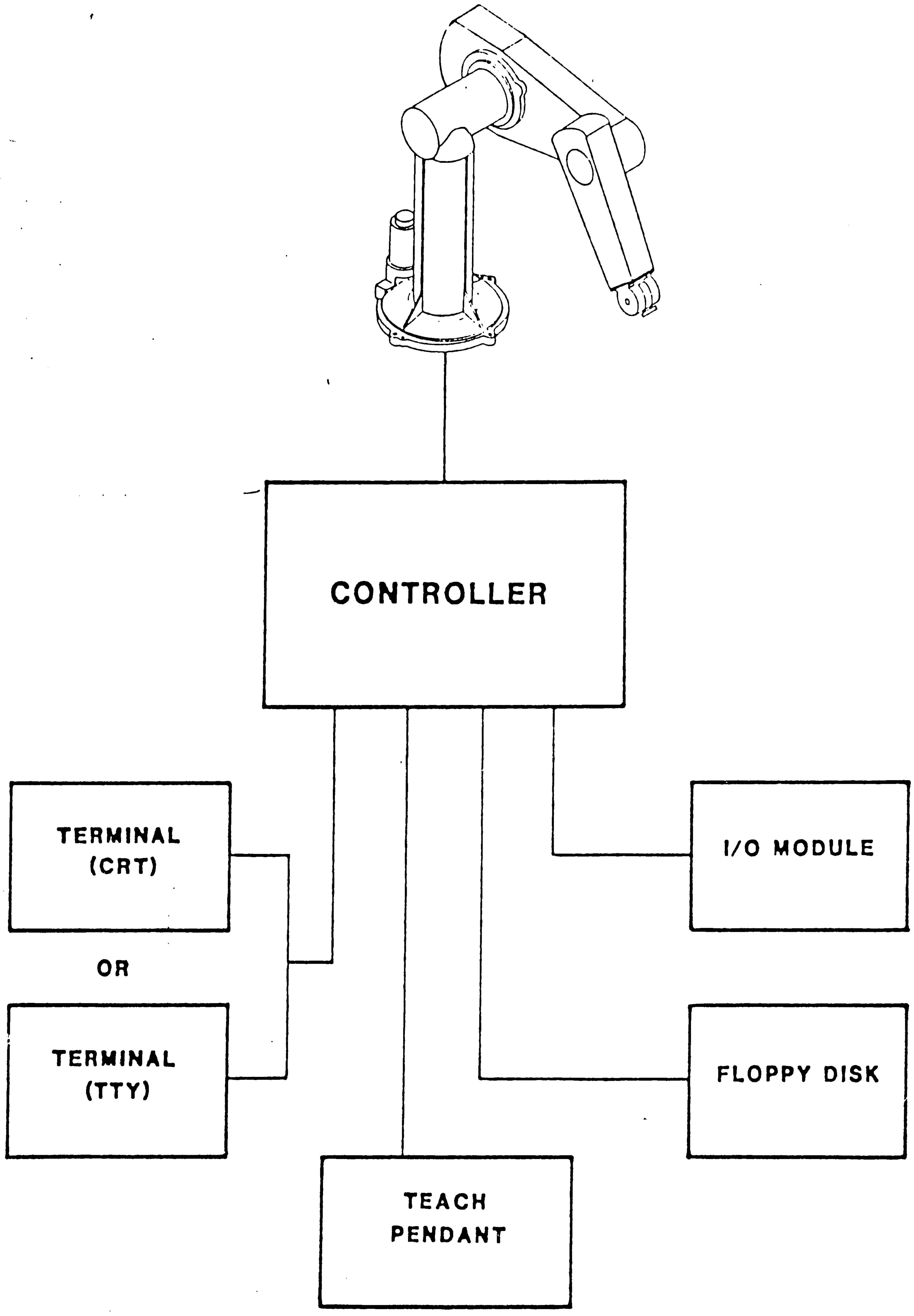


Fig. 3.2 PUMA 560 System Block Diagram [12].

. Power supplies

. Power amplifier assemblies

. High power discharge board

. Arm cable board

3). Peripherals: The peripheral components for the PUMA system are used to input or receive information and consist of a terminal, a teach pendant, floppy disk drive unit, additional memory, and an I/O module.

4). Robot Arm: The robot arm is the mechanical component of the system incorporating 6 degrees of freedom, each controled by a DC servomotor. It is sufficiently flexible to be taught a wide variety of tasks. Each member of the robot arm is connected to another member at a joint, much like a human arm and torso. The members of the robot arm are shown in Fig. 3.3.

3.3 GE P-50

The P50 Process Robot is designed for welding, parts handling, and various other manfacturing tasks[15].

There are three principal parts to the system:

1). Robot Body : The Robot Body is essentially a floor-mounted five-axis arm. (Fig 3.4)

PAGE(S)

MISSING

PAGE(S) 35

The arm is made up of a base that swings in the horizontal plane, an upper arm that attaches to the base, a forearm that attaches to the upper arm, and a wrist with bend and twist axes.

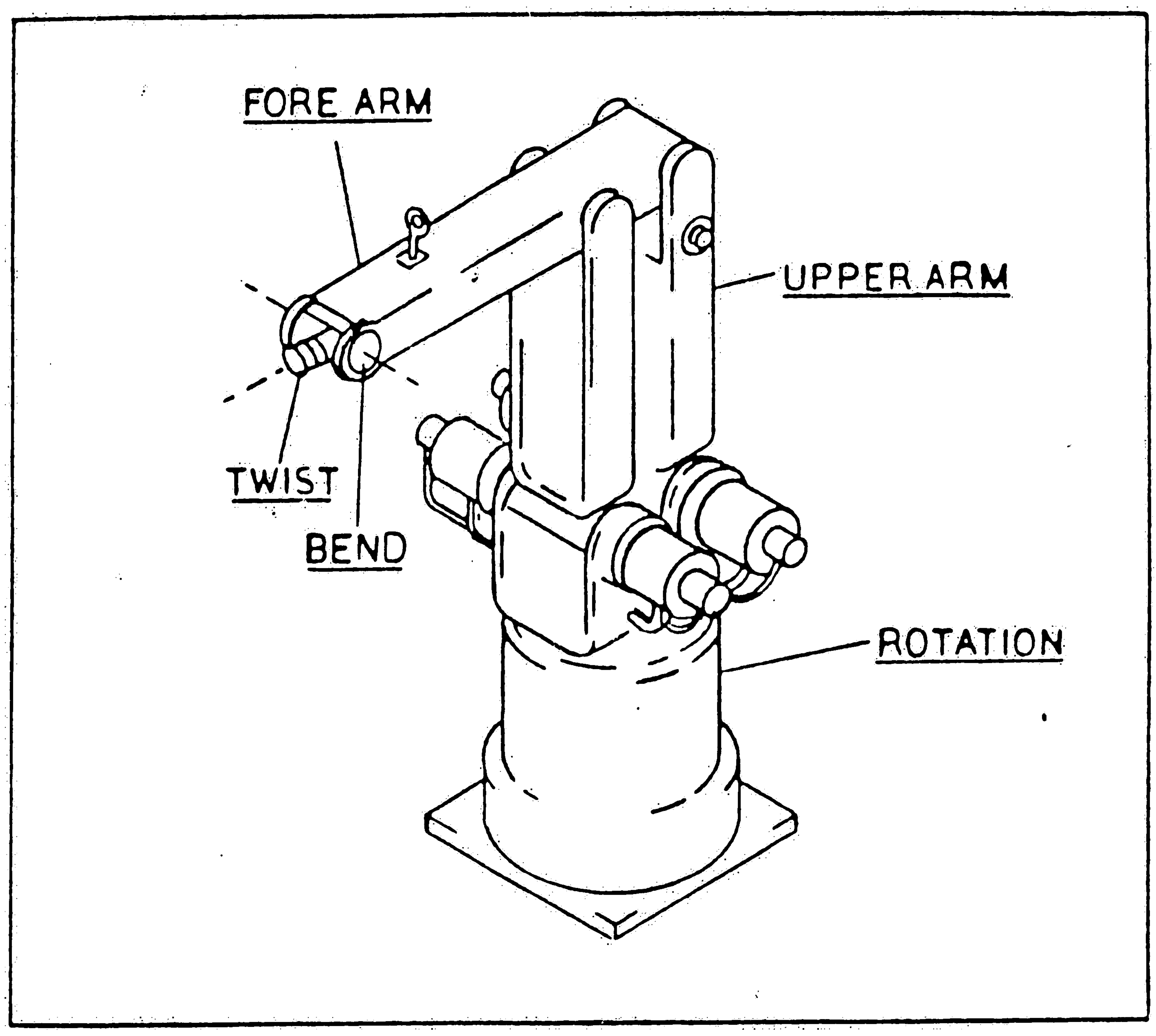


Fig. 3.4 P 50 Robot Body [15].

2). Robot controller

The Robot Controller contains the intelligence of the system, a 9" cathode ray tube display, as well as the power supplies, the control panel, the servo amplifers and the various input and output interface devices, and a receptacle for a cassette recorder. An isolation transformer protects the controller from outside electrical transients.

3). Teach Box

The TEACH BOX is a portable pendant with a sealed membrane switch assembly used to enter the motion instructions into the controller using individual-function keys.

3.4 IBM Personal Computer XT System[16]

The IBM Personal Computer XT in this research was used for developing the software in interfacing the i-bot vision system to the GE P-50 robot.

The system unit is the center of IBM Personal Computer XT system. The system unit contains the system board, which features eight expansion slots, the 8088 microprocessor, 40K of ROM (include BASIC), 128K of base R/W memory, and an audio speaker. A power supply is located in the system unit to supply dc voltages to the system board and internal drives.

The system board consists of five functional areas: the processor subsystem and its support elements, the read-only memory (ROM) subsystem, the read/write (R/W) memory subsystem, integrated I/O adapters, and the I/O channel. (Fig. 3.5)

The heart of the system board is the Intel 8088 microprocessor. This processor is an 8-bit external bus version of Intel's 16-bit 8086 processor, and is software-compatible with the 8086. Thus, the 8088 supports 16-bit operations, including multiply and divide, and supports 20 bits of addressing (1 megabyte of storage). It also operates in maximum mode, so a co-processor can be added as a feature. The processor operates at 4.77 MHz. This frequency, which is derived from a 14.31818-MHz crystal, is

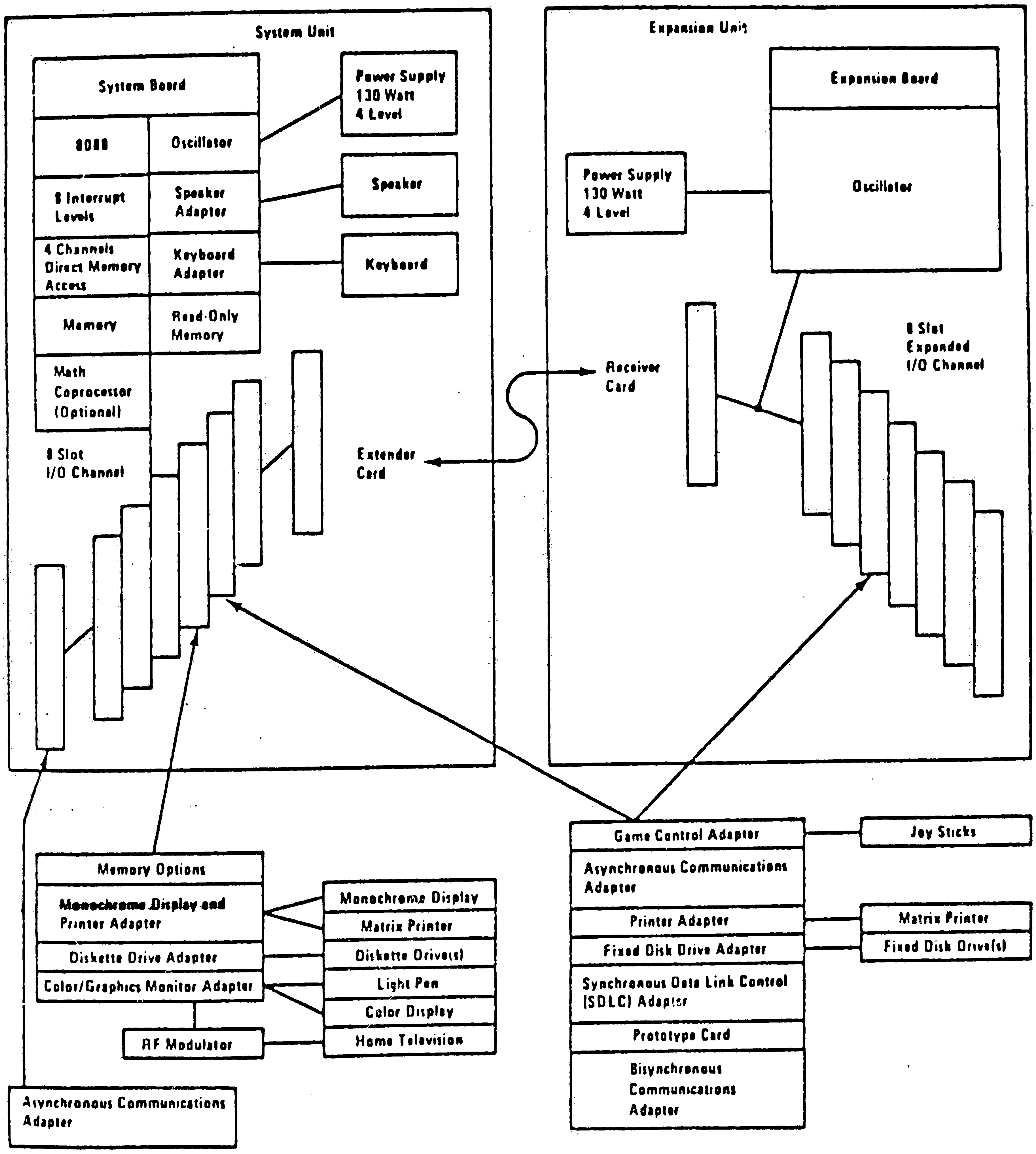


Figure 3.5 IBM PC/XT System Block Diagram [16].

divided by 3 for the processor clock.

Three of the four DMA(Dynamic Memory Access) channels are available on the I/O bus and support high-speed data transfers between I/O devices and memory without processor intervention. The fourth DMA channel is programmed to refresh the system dynamic memory. This is done by programming a channel of the timer-counter device to periodically request a dummy DMA transfer. This action creates a memory-read cycle, which is available to refresh dynamic storage both on the system board and in the system expansion slots.

Of the eight prioritzed levels of interrupt as listed in Table 2 , six are bussed to the system expansion slots for use by feature cards. Two levels are used on the system board. Level 0, the highest priority, is attached to Channel 0 of the timer/counter and provides a periodic interrupt for the time-of-day clock. Level 1 is attached to the keyboard adapter circuits and receives an interrupt for each scan code sent by the keyboard. The non-maskable interrupt (NMI) of the 8088 is used to report memory parity errors.

In this research, the IBM XT is treated as the system controller of the standard modular vision system. The data communication among vision system, the robot and the system controller can be accomplished through the RS 232C and bus interrupted techniques.

TABLE 2

8088 Hardware Interrupt Listing [16].

NUMBER	USAGE
NMI	Parity
0	Timer
1	Keyboard
2	Reserved
3	Asynchronous Communications (Secondary) SDLC Communications BSC (Secondary)
4	Asynchronous Communications (Primary) SDLC Communications BSC (Primary)
5	Fixed Disk
6	Diskette
7	Printer

CHAPTER 4

Study & Development of Robot to Vision Interface

The literature review discussed the Robot Dependent method, the Vision Dependent method and an Integrated method for communications between a robot and vision system. The approach taken in this chapter is to fully define the interface between a robot and vision system and develop a list of the types of problems in defining such an interface. In the succeeding chapter a strategy is developed for a generic vision-to-robot interface. Two robots are used in conjunction with the ORS i-bot vision system to aid in developing such strategies : (1) the PUMA 560, and (2), the GE P-50. The objective is to develop a plug compatible vision system and robot interface which would give greater flexibility and cost benifits in installing vision-robot based interfacing systems. The remainder of this chapter is devoted to detailing the hardware and software aspects of vision to robot communication.

4.1 Hardware Communication Standards

Circuitry for an interface is needed to transfer information from the microcomputer to external peripherals or computers[13]. The microcomputer with its internal bus and interface circuitry is shown schematically in Fig. 4.1 . The interface circuitry is usually on a printed circuit board (card) that plug into connectors on the microcomputer's bus. These cards have standard connectors that are hooked up to external peripherals via simple cables.

Several standard interfaces are used in the microcomputer industry today. The most common are listed in Table 2. [27].

Table 2

Communication Standards

Name	Type	Common usage
EIA RS-232-C	Serial	Video terminals, printers
20 mA current-loop	Serial	Terminals
Centronics parallel	Parallel	Printers, plotters
IEEE 488 (HP IB)	Parallel	Test equipment, printers

Microcomputer interfaces fall into two basic categories : Serial and Parallel. Serial interfaces transfer data one bit at a time, while parallel interfaces transfer one byte (eight bits) at a time. A detailed description of these two interfaces are as follows :

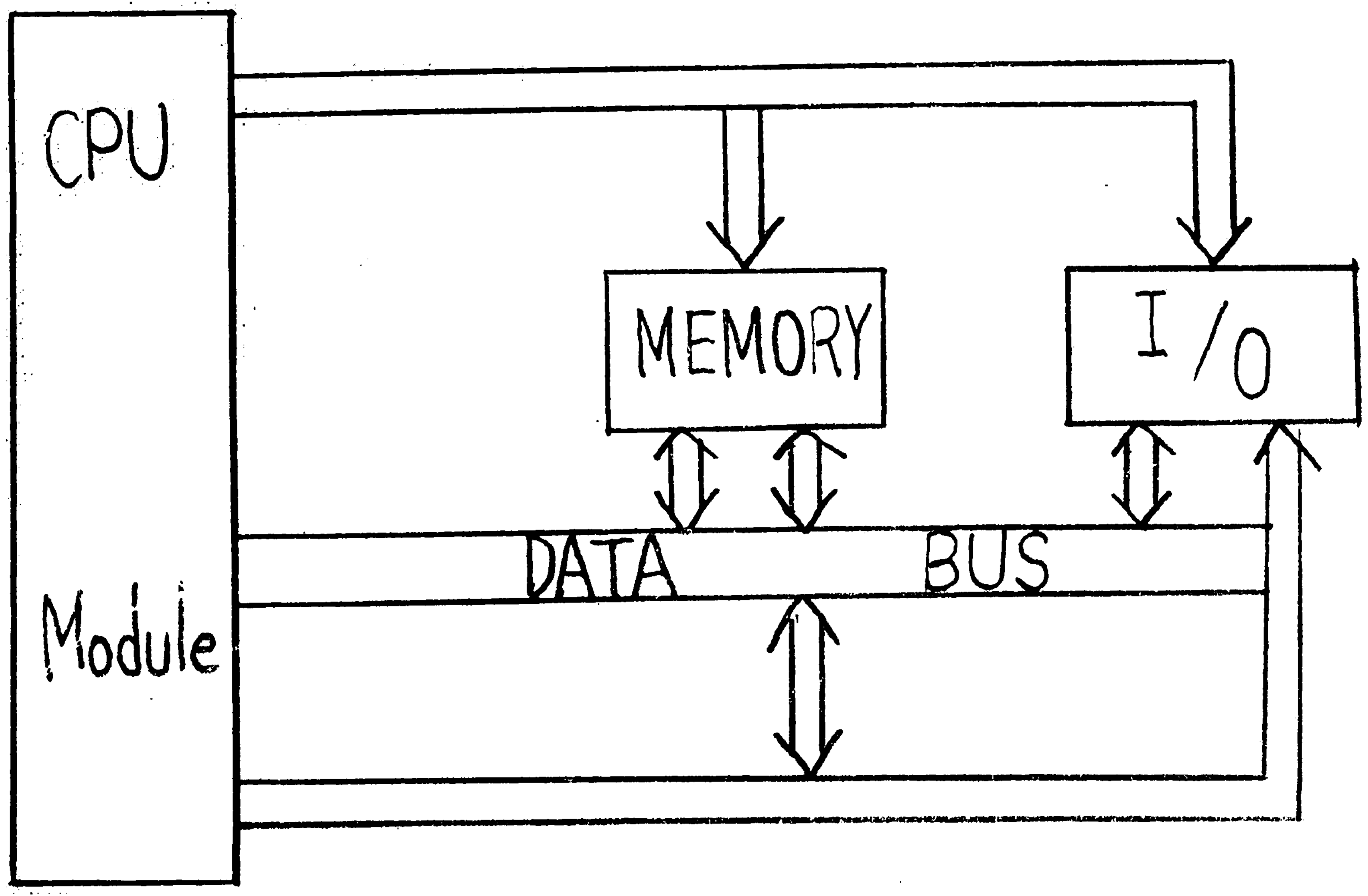


Fig. 4.1 Interface Circuitry

1). Serial I/O and Communication : Usually every general-purpose microcomputer includes a serial interface based on a Universal Asynchronous Receiver/Transmitter (UART). The UARTs contain serial-to-parallel and parallel-to-serial shift registers and timing and control logic. These form two essentially independent data channels, one for output from the computer and one for input to it. They are used to support communication interfaces among computers and for interface to interactive terminals. Figure 4.2 is a block diagram of the internal organization of a typical UART.

In serial communication, data are transmitted and received one bit at a time. The UARTs accept bytes to be transmitted from the microcomputer, surround them with the required formatting and error-detecting bits and transmit the resulting character serially. Similarly, received serial data are accumulated by the UART so that the data byte may be read by the microcomputer in parallel. Formatting bits are removed from the received data and limited error checking is performed by the UART. Figure 4.3 diagrams the transmission of a character using the UART method.

2). Parallel I/O usually includes control logic and support circuitry for two or more parallel I/O ports of eight bits each. One of the most important characteristicss of a parallel I/O interface is that it is programmable. Program capability means that the directionality, formats, and functions of the

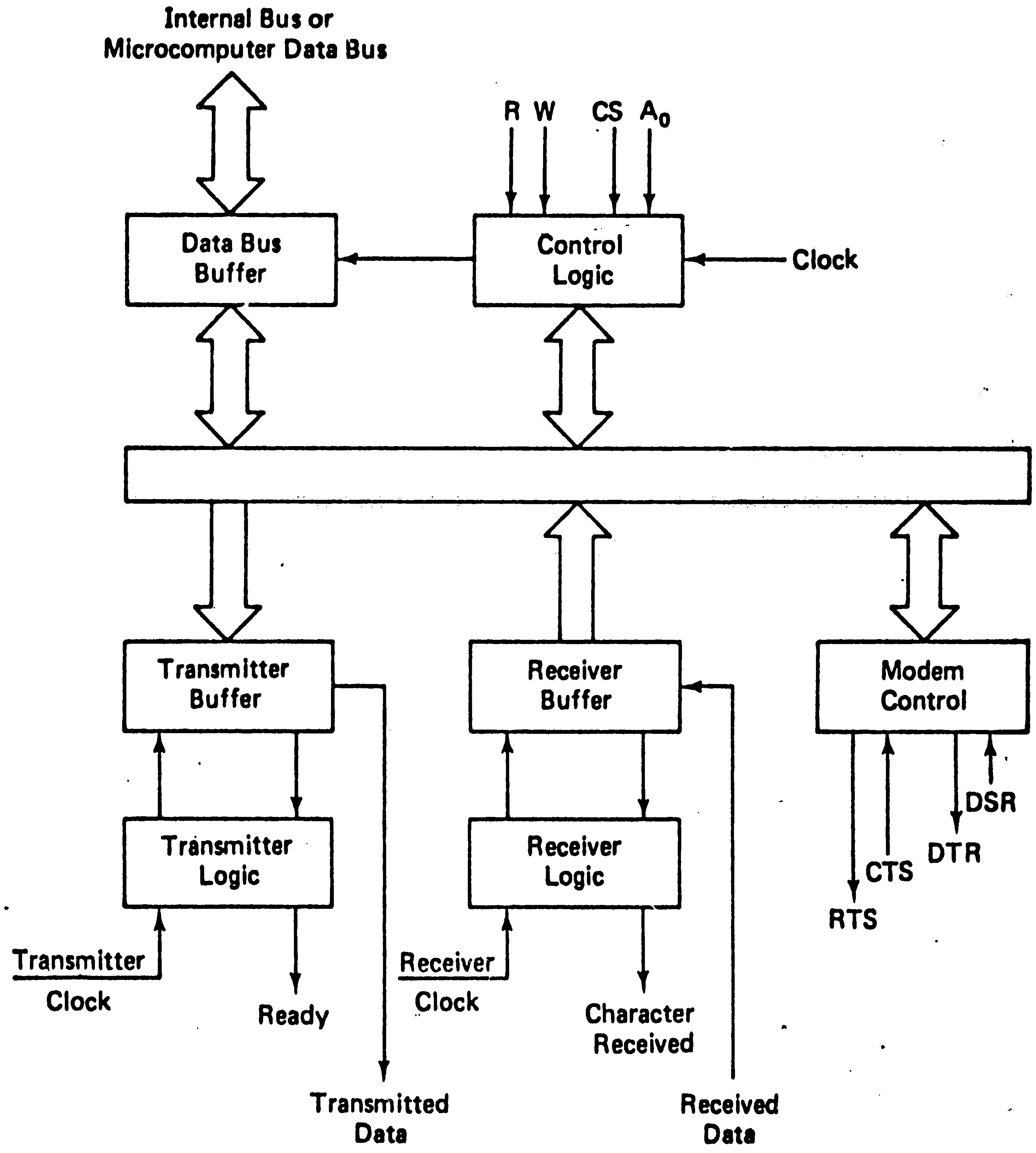


Figure 4.2 A Internal Organization of a Typical UART [17].

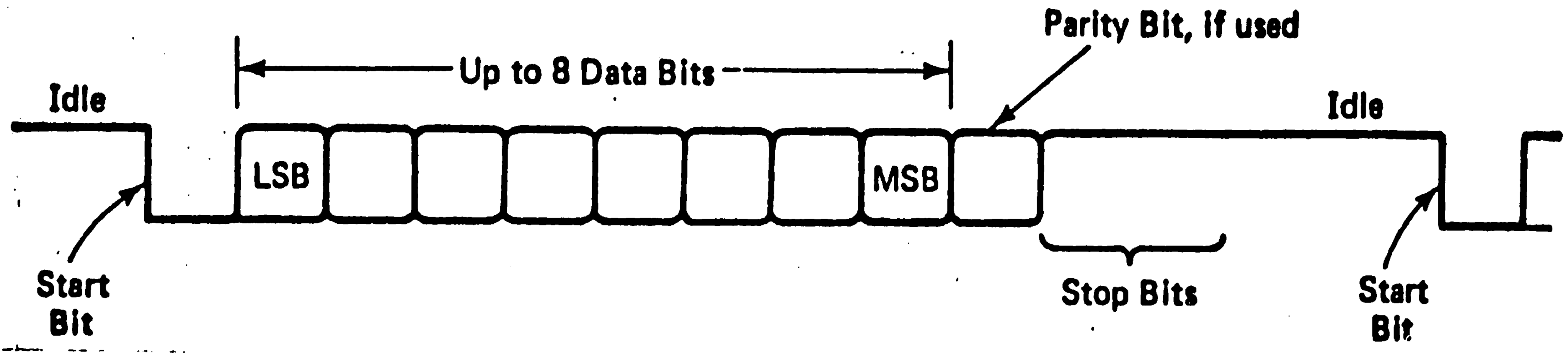


FIGURE 4.3 Asynchronous character transmission. [i7].

I/O ports can be changed by means of mode control output bytes transferred from the microcomputer with which they are used. Since each manufacturer has defined his own connections between the microprocessor and peripheral devices, there is no accepted standard for peripheral chips. The Motorola 6821 Peripheral Interface Adapter (PIA) and the Intel 8255 Programmable Peripheral Interface (PPI) are in widespread use such that they may establish de facto standards for peripheral interface chips. The block diagrams of the 6821 and 8255 chips are given in Figure 4.4.

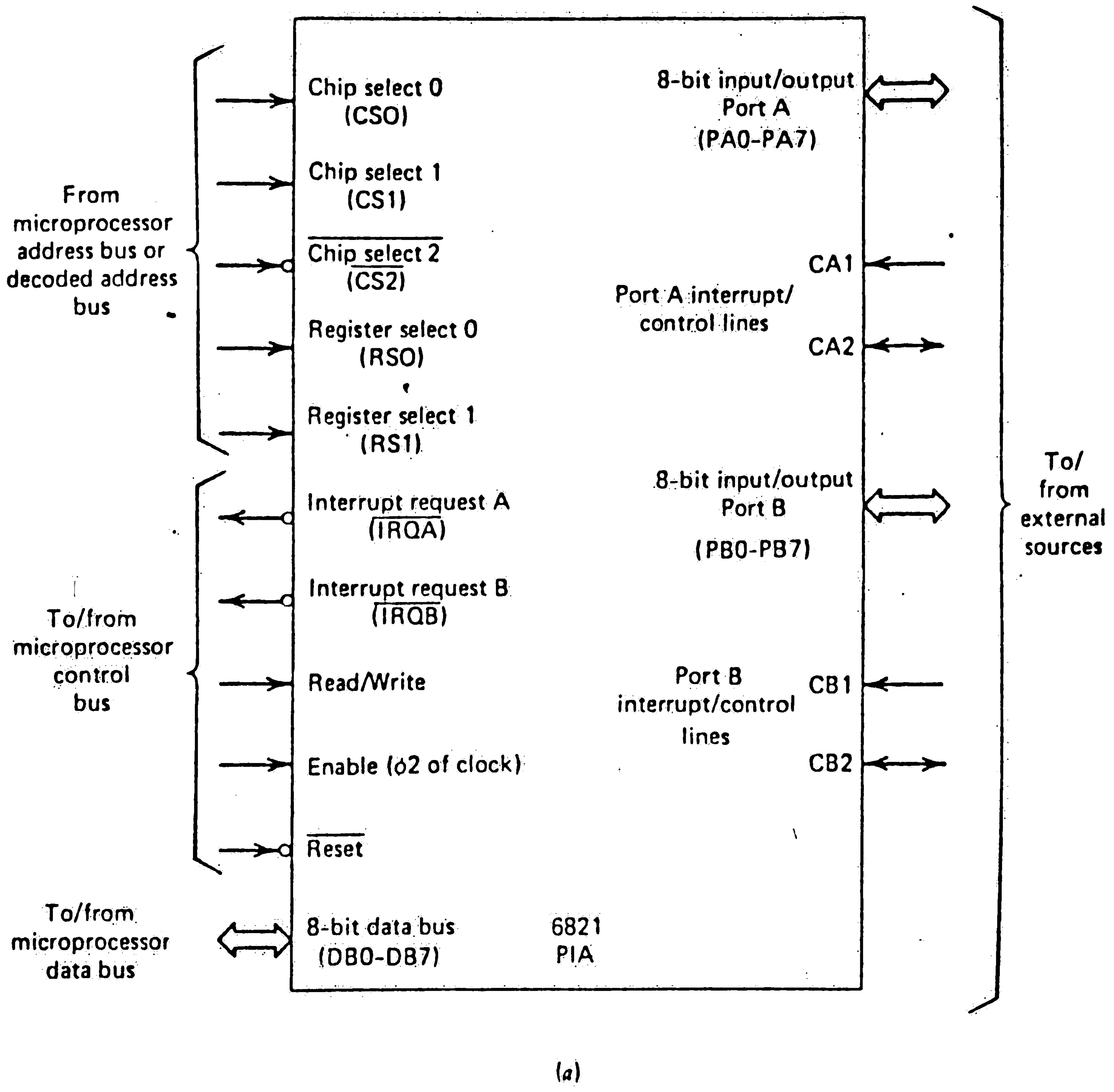


FIGURE 4.4 Block diagrams of 6821 and 8255 interface chips.
(a) Motorola 6821 Peripheral Interface Adapter (PIA) [17].

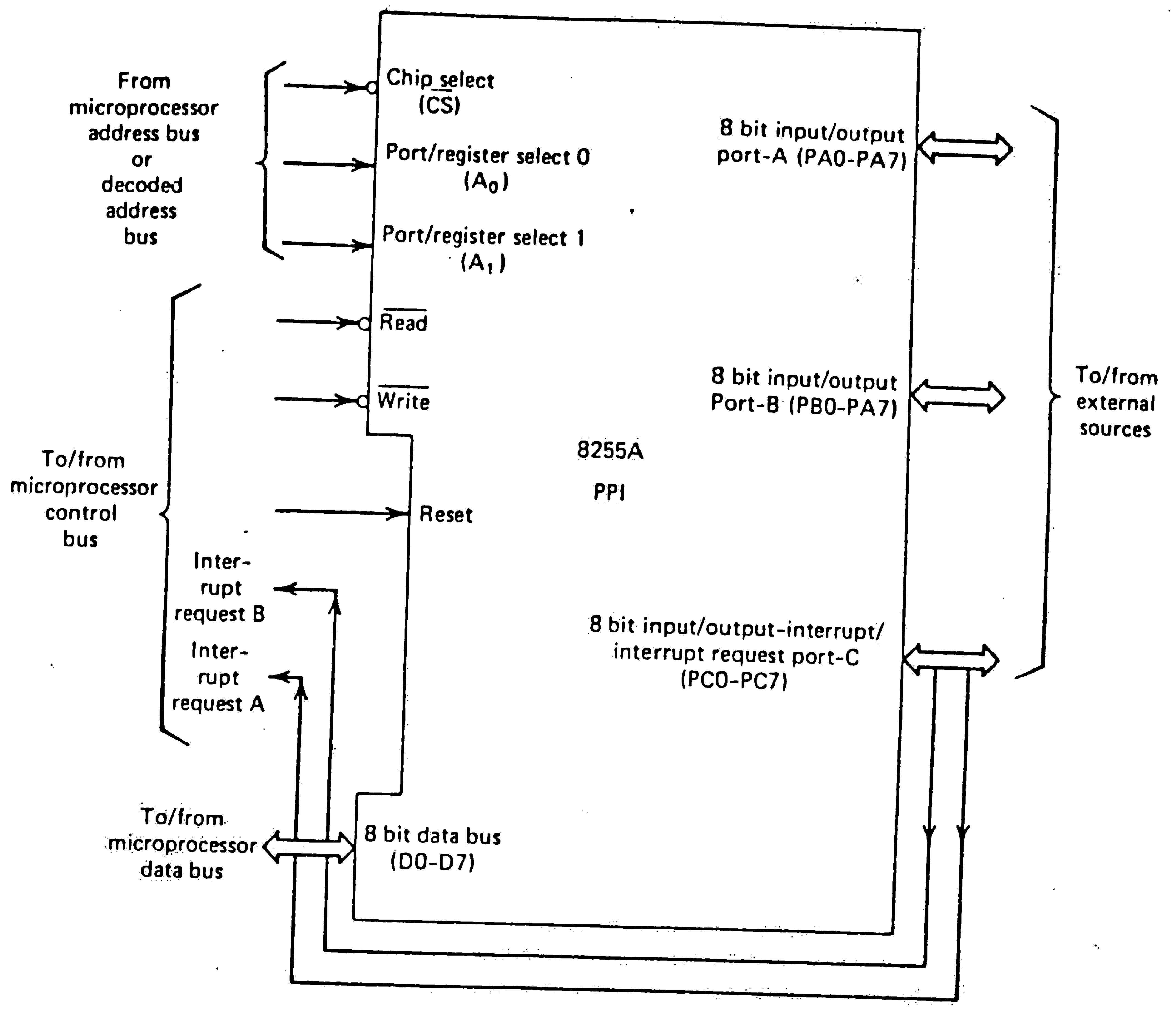

From microprocessor address bus or decoded address bus
Chip select (CS)
Port/register select 0 (A0)
Port/register select 1 (A1)
Read
Write
To/from microprocessor control bus
Reset
Inter-rupt request B
Inter-rupt request A
8255A
PPI
8 bit input/output port-A (PA0-PA7)
8 bit input/output Port-B (PB0-PA7)
8 bit input/output-interrupt/interrupt request port-C (PC0-PC7)
To/from external sources
To/from microprocessor data bus
8 bit data bus (D0-D7)

FIGURE 4.4 (b) Intel 8255 Programmable Peripheral Interface [17].

Each port in a parallel I/O can be programmed to be an input or an output port. For some devices, a port can be programmed to be bidirectional. This is useful for interfacing with microcomputer peripheral devices that have adopted bidirectional interface formats.

When programmed for output, the flip-flop will latch the output data. When a device programmed for input is addressed, data are gated through the devices and onto the data bus interfacing it with the microprocessor. Data thus transferred are in the state they were in during the transfer. Some programmable parallel I/O support chips, such as the Intel 8255 chip, add input latching so that data can be written into the chip from an external device and read by the microprocessor later.

The RS-232-C is the most common serial standard. It is used for video terminals and many printers and plotters. The two most popular parallel standards are the IEEE 488 and the Centronics parallel. The IEEE 488, sometimes called the HP IB (Hewlett-Packard Interface Bus), is used primarily by electronic instrument manufacturers for communicating with their test equipment.

4.2 Examples of a Robot to Vision Interface

There are two main categories of electronics interfaces : digital and analog. The microcomputer may be considered as the ultimate sink of incoming information and source of outgoing information and is essentially digital. A robot and vision controller are typically both microprocessor-based systems. The interface between the vision system to the robot is digital. Interfaces between digital elements generally confine their reforming of information. To aid in defining the interface between a vision system and robot controller two commercially available vision and robot systems were considered. These system were developed for specific applications. They differ in that the Westinghouse system is a robot dependent system wheras the PUMA / VS 100 is a Vision Dependent system.

The first example is the Westinghouse robotics vision system [20]. The system considered was used for for assembly of an electromechanical relay under vision control. The structure of the vision - robot system is shown in Fig.4.5.

A PUMA 250 industrial robot was used to assemble a four piece electromechical relay. All work pieces were located in the field of view of a Westinghouse TV camera and were placed such that there was no overlapping of work parts and a space existed between the work pieces. Vision-robot communications were realized through an RS-232 serial link. A PUMA assembly program and a vision-robot communications protocol used in the study are illustratedly by Fig. 4.6. The program "Assembly" used for this study was written in the VAL robot programming language and the PUMA robot communicated with the image postprocessor through its controller's CRT port.

In the Westinghouse robotics vision system of this example, Makhlin and Tinsd noted that "... the vision system commands the robot to start the "Assembly" program (EXECUTE ASSEMBLY). The program stops at step 2, and , at this point, the robot is asked to display a point A (an arbitrary location of a work piece within the field of view). The vision system, then modifies the location A by sending to the robot a value of its position (X,Y,Z) and orientation (O,A,T), calculated on the basis of visual information. Consequently, the robot moves into the newly defined location A, picks up a workpiece and brings it to the part predefined assembly point B. The program returns them to

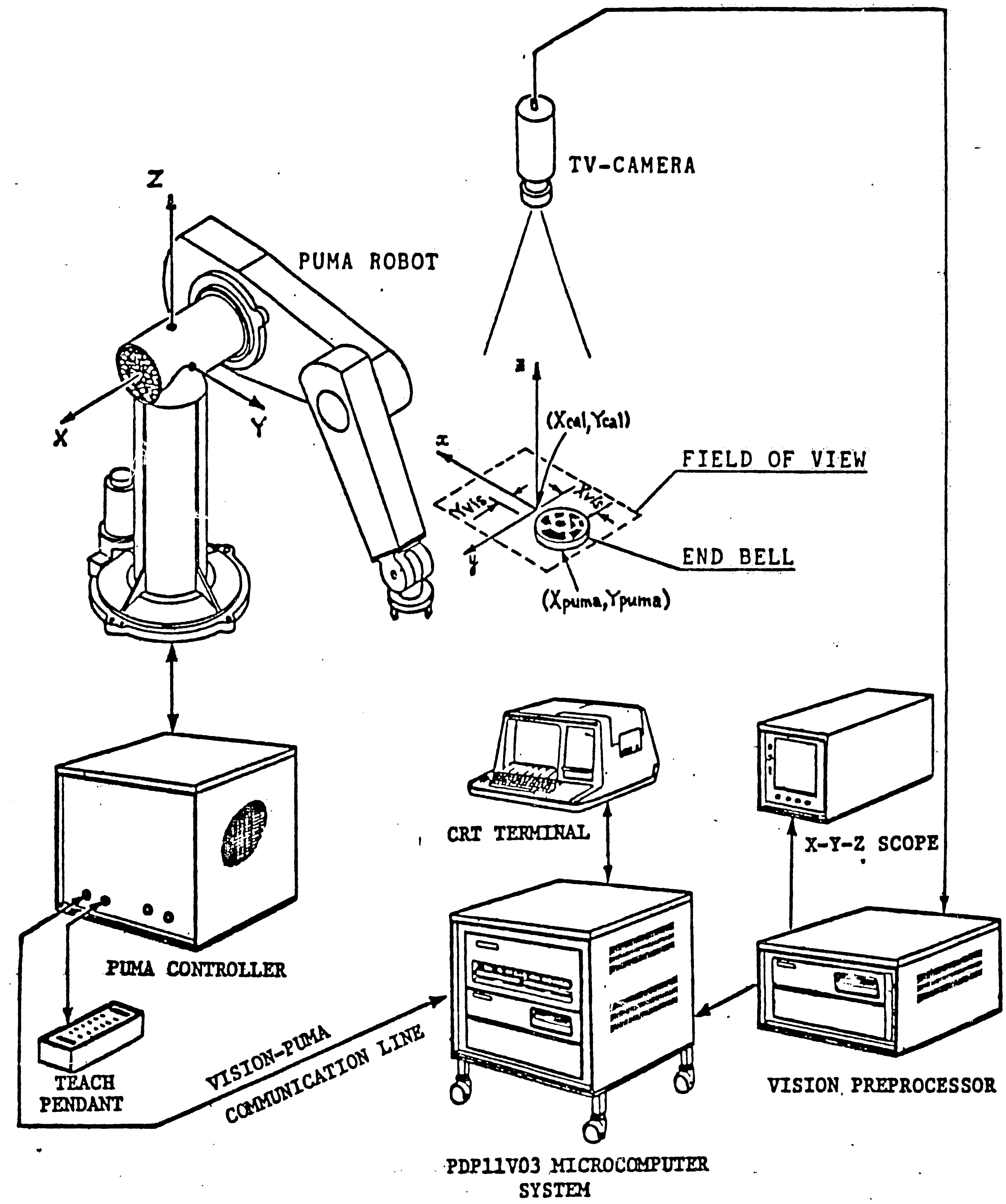
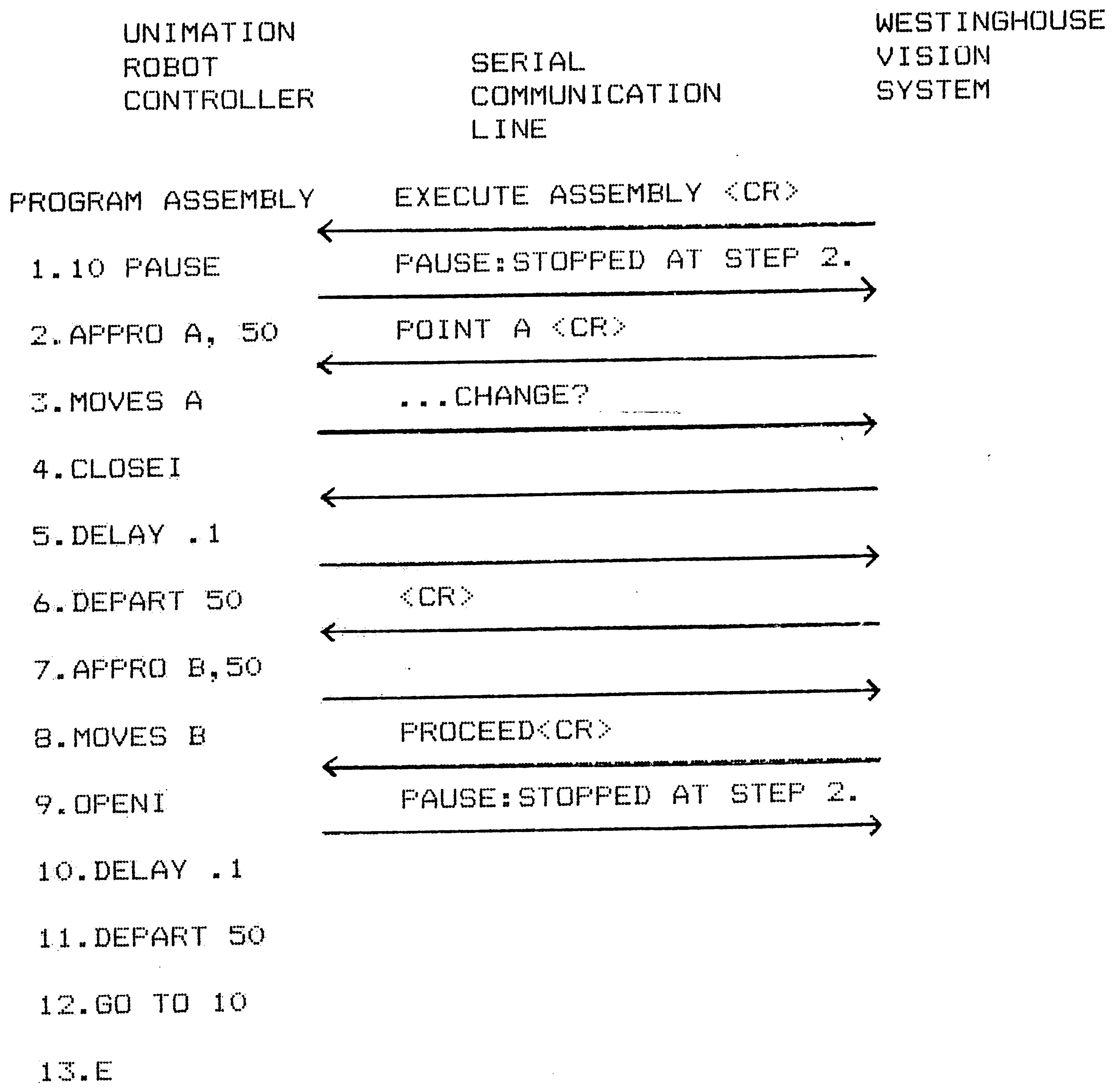


FIGURE 4.5 Structure of the Westinghouse Vision - Robot system [19].



```
PROGRAM ASSEMBLY
 1.10 PAUSE
 2.APPRO A, 50
 3.MOVES A
 4.CLOSEI
 5.DELAY .1
 6.DEPART 50
 7.APPRO B,50
 8.MOVES B
 9.OPENI
 10.DELAY .1
 11.DEPART 50
 12.GO TO 10
 13.E
```

FIG. 4.6 Vision-Robot communications protocol [19].

the step 2 and the robot receives the next location A etc..."

The second example uses the PUMA/VS-100 vision system. In order for the robot to acquire parts located by the vision system, there must exist a common cartesian frame of references. This is accomplished during system calibration. First, the vision system is calibrated to establish an accurate two dimensional coordinate system in the field of view. Next, the camera coordinate system is related to the coordinate system of the robot.

Carlisle, Gonzalez and Mcghie noted that "... Vision System Calibration is accomplished by determining a camera-to-robot coordinate transformation. The transformation relating the field of view to the robot can be calculated in the following procedure. Using a pointer and disk, four robot locations and their corresponding vision locations are determined. The disk is placed at one corner of the field of view. The robot is moved to place the pointer in the center hole of the disc. The robot location is recorded by typing HERE R1 on the terminal. The V1 position is recorded by the vision system. The procedure is repeated until R2, V2, R3, V3 and R4, V4 are all defined as shown in figure 4.7, then the commands are entered.

```
DO FRAME TOB = R1, R2, R3, R4
DO FRAME TCB = V1, V2, V3, V4
DO INVERSE TBC = TCB
DO SET TO.CAMERA = TOB : TBC
```

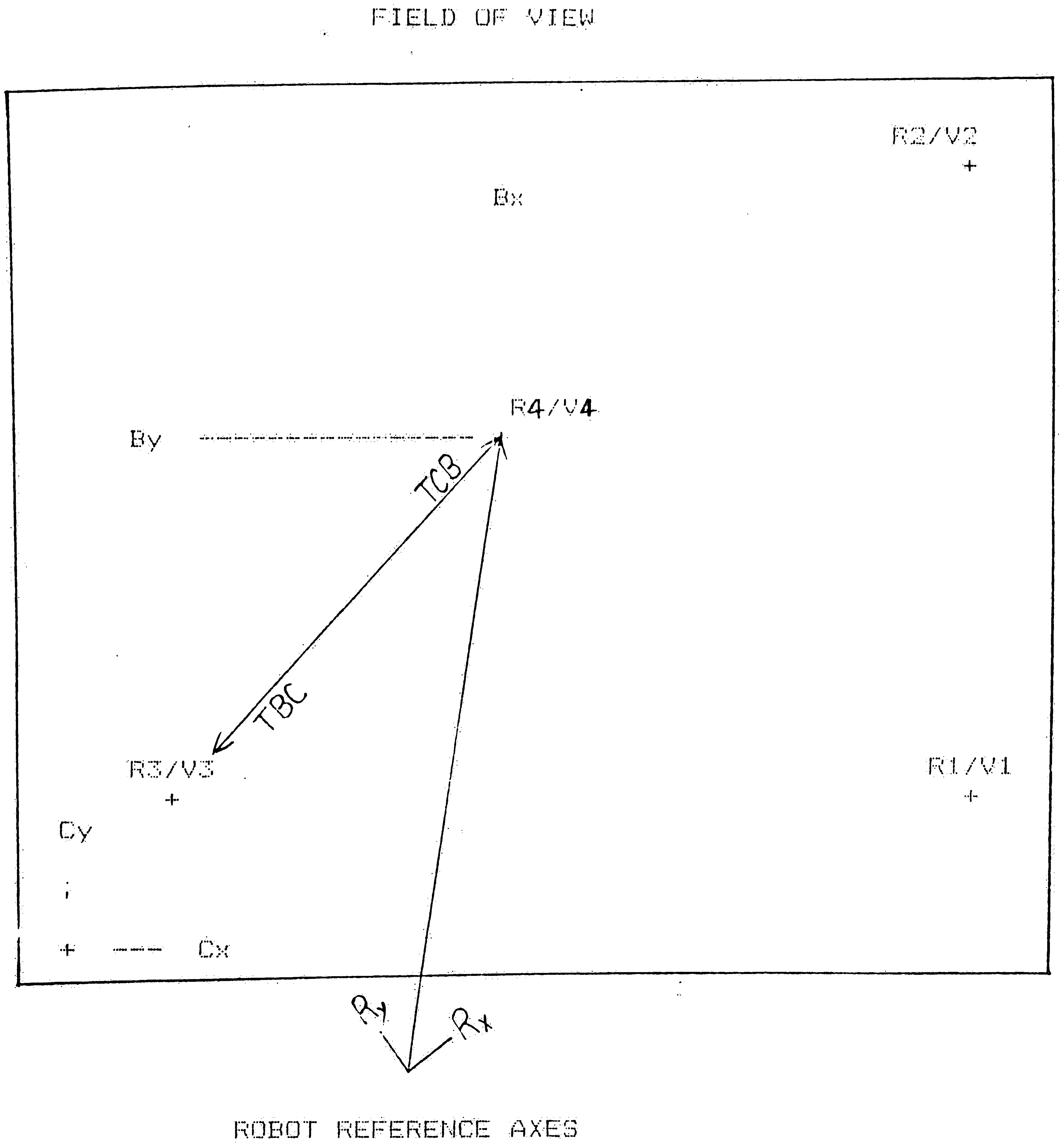


Figure 4.7 Robot/Camera transformation [3]

TO.CAMERA is the transformation which takes the robot to the origin of the camera coordinate system. If the command DO MOVE TO.CAMERA is entered the robot would move so that the pointer is at the origin of the camera coordinate system etc..."

For A different robot using the same vision system, the command should be changed since different robots have different control languages, but the transformation procedure can be achieved in the same way.

4.3 Proposed Procedures for Robot-Vision Interfacing

From the preceeding examples, the interface between the vision system to the robot can be seen to encompass the physical interface, the data transfer protocol, the calibration software and coordinate transformation software. The Westinghouse and Puma systems of these examples typify the general structure of the interfaces for vision system and robot system communication. Based on these examples and the literature review the following definitions and procedural descriptions are presented for the above four aspects pertaining to the vision-robot systems interfaces.

(1) Physical interface: A RS 232C asynchronous communication line is proposed for the physical interface between the vision controller and the robot controller. As noted by Artwick [27] the EIA RS 232C interface has been derived from the International Telephone and Telegraph Consultative Committee (C.C.I.T.T.) as a telecommunication standard and has been accepted as an industry standard. Furthermore, there are no standardized parallel interface formats (such as the IEEE 488 bus) which have been accepted by industry even as a de facto standard. In

addition, the speed requirement (data transfer rate) for typical vision-to-robot systems in an industrial setting is a 9600 baud rate which is easily available with a serial RS 232C line. Further considerations indicate a lower cost and long distance remote capability (minimal transmission error) when choosing a serial communication line over a parallel line.

The typical vision controller has on-board switches used to set parity,stop bit count, data bit count, and baud rate. An additional justification for choosing the RS 232C interface is that it has memory or port locations assigned to it for communication with the CPU. The typical RS-232 interface has at least four of such ports: two for transmitting and receiving data and two for controlling and monitoring various operating parameters.

The receive and transmit locations, or registers on the RS 232C interface, can only receive or transmit data. The receive register cannot be "written into" by the CPU while the transmit register cannot be "readed" by the CPU. The control registers are often designed to be bidirectional - similar to regular computer memory.

All of the data transferred byte-by-byte between the vision system and the robot controller are passed through the RS 232C interface line.

(2) Data transfer protocol : Software needs for data communication are classified as pertaining to either low-level data communication protocol or high-level data communication

protocol.

A. low-level data communication protocol

The lowest level of robot control is the software routines which define how we send data to and from the robot system. The vision controller is physical connected with the robot controller through an RS-232 asynchronous serial data link. Software consists of two subroutines. (1) robot-to-vision: getting a byte of data from the robot controller, and (2) vision-to-robot: sending a byte to the robot controller.

B. high-level data communication protocol :

A high-level software routine reads and writes complete programs (as opposed to defining data protocol) beween the vision system and the robot controller. Whereas low-level protocols are concerned with the transfer of data bytes under a specified format (protocol), the high-level protocol is concerned with the transfer of ASCII data. For example, transfer of a program command such as EXE TEST involves the high-level protocol which transfer the command in ASCII code character-by-character for the complete command in which is imbedded the low-level protocol for translating ASCII code into a binary bit stream.

(3). Calibration software

Calibration software is required to calculate the transformation scaling factor from coordinates relative to a vision system (pixel by pixel) to a robot coordinate system and , in the case of two-dimension (2-D) vision systems, account for the three-dimension (3-D) dispersion angle (i.e. depth information)

Scaling routine : These software routines are specifically developed for two-dimensional (2-D) vision systems and are written to transform the data from a vision system reference frame to a robot reference frame. The software routines consist of a top-plane-x and top-plane-y for a predetermined top plan view in the vision system and a bottom-plane-x and bottom-plane-y for a predetermined bottom plan view (typically a surface upon which parts rest or the background of an image). At either the top or bottom plane, the data that is transformed is the x,y positional data and the θ orientation datum. Only 3 values at each level are needed for these software calibration routines.

(4). Coordinate transformation procedure

During the vision to robot communication phase the following task must be carried out in the vision controller:

-calculation of the center of gravity O (X_o , Y_o)

-identification of the workpiece

-calculation of the orientation of the workpiece

-transformation of the coordinates of the center position to the coordinates of the robot system

The typical two-dimensional (2-D) image processing vision system has three data values: consisting of the "X" and "Y" coordinates relative to the pixel coordinate system and the orientation of a part in radians. Before data is sent to the robot controller, a scaling procedure which transforms X and Y coordinates into a Robot's reference frame must be done.

Coordinate transformation software : These two routines accept 3 points from the vision system, and the information is coded so as to be recognizable by the robot controller. This coded information is then passed to the robot system.

4.4 ORS i-bot 1 to PUMA Interfacing Example

The system configuration diagram for the current ORS i-bot 1 vision system to a PUMA robot is shown in Fig. 4.8. According to the above definition in interfacing vision system to the robot, the i-bot 1 is a Robot Dependent system. The following section details the defined procedures of the previous section to the current vision-robot interfacing system :

1. Physical Interface

The current vision controller with a 8086 processor is connected to the Puma through an RS 232 C cable. The communication between the vision controller and the robot controller is handled by a DLV11-J board in the robot and an Intel 8251 chip in the vision system. As previously mention, the robot controller contains a LSI-11 processor with a DLV11-J board which is a four-port asynchronous serial I/O board. One of the

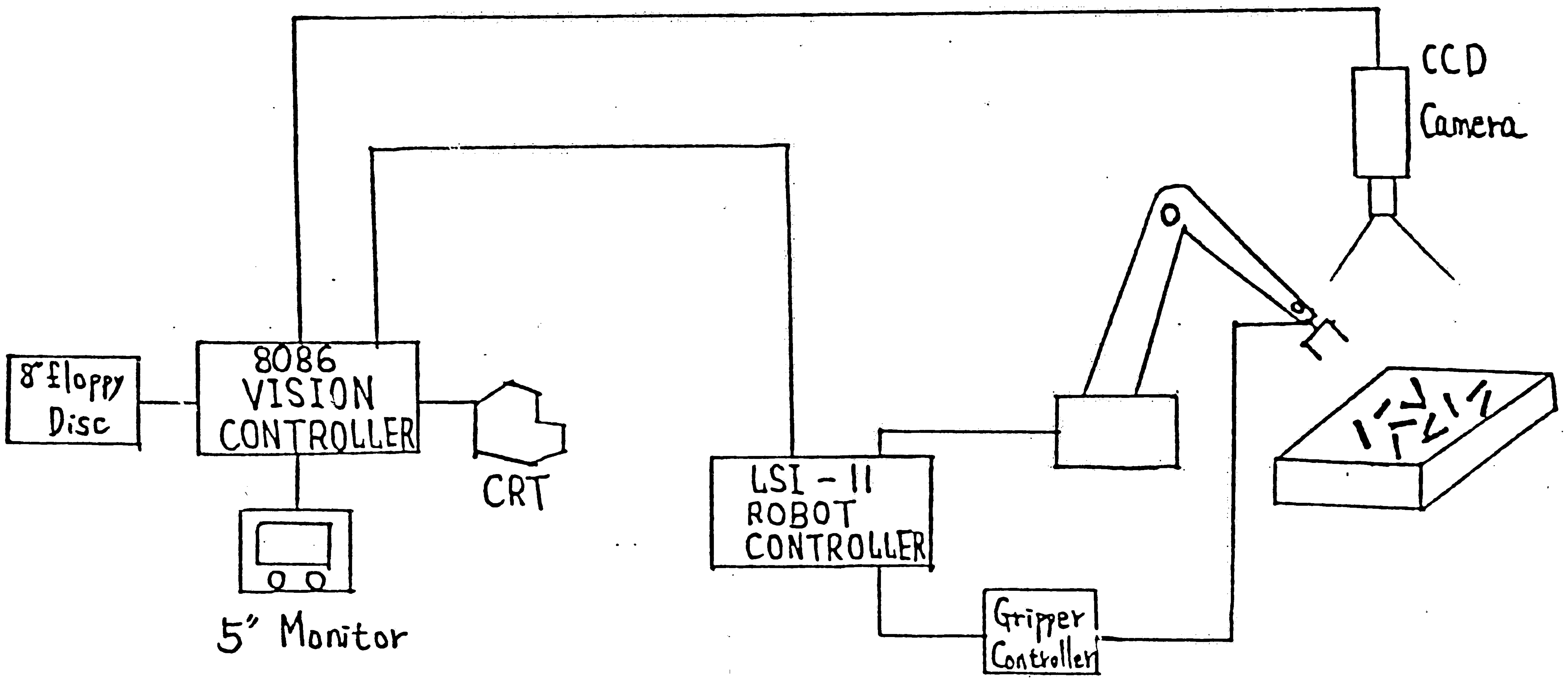


Fig. 4.8 i-bot 1 to PUMA System Interfacing

RETAKE

The Operator has Determined that the Previous Frame is Unacceptable and Has Refilmed the Page in the Next Frame.

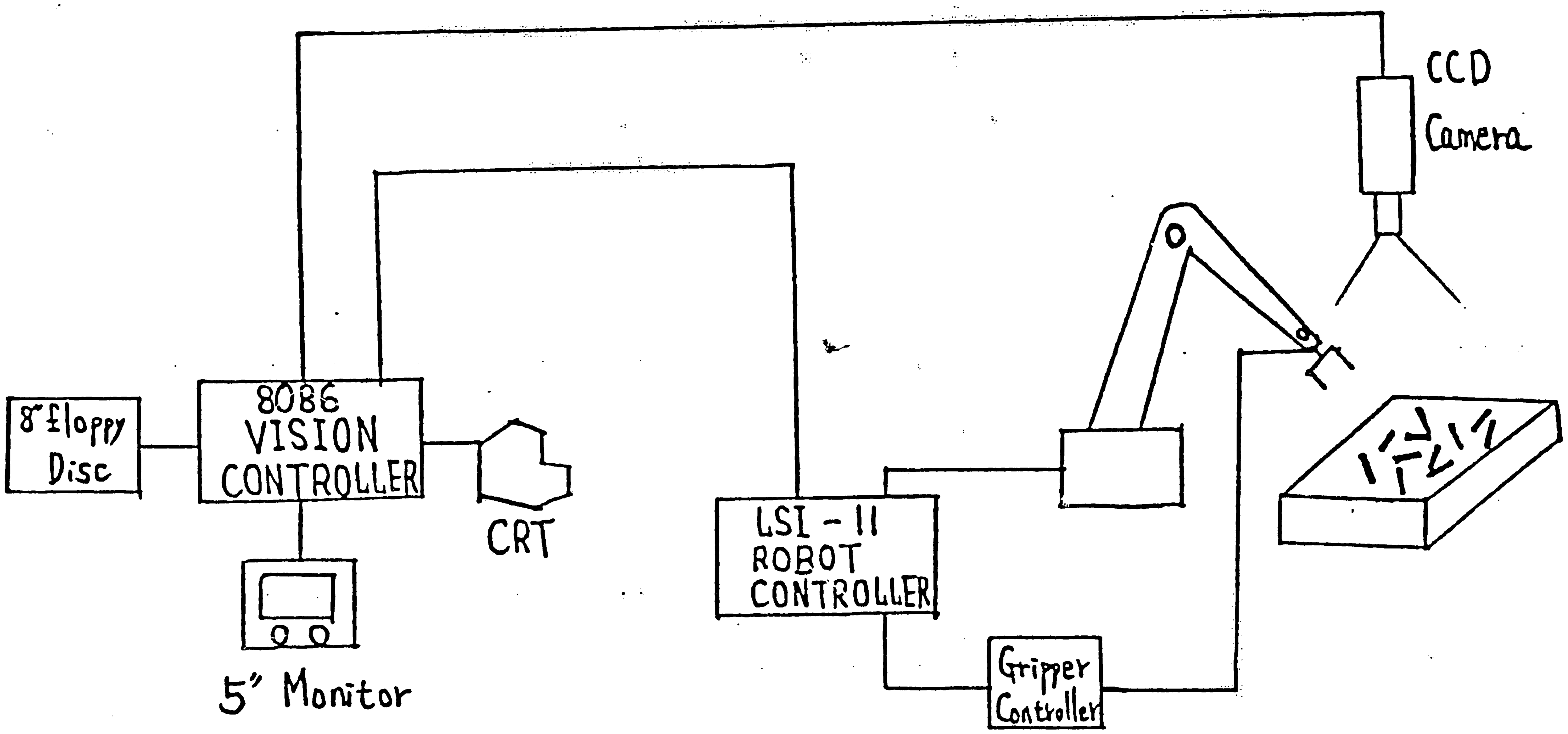


Fig. 4.8 i-bot 1 to PUMA System Interfacing

ports is used for a terminal connection. The other ports are used for a teach pendant, disc controller, and any accessory expansion boards. In the ORS Vision Controller, there is a 8251 programmable communication interface for handling the I/O task.

The 8251 is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use. The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or wherever it has received a character for the CPU. The data format for the communication between the ibot and the PUMA is shown in table 3.

TABLE 3

PUMA and IBOT Communication Data Format

Mode	Full Duplex
Baud Rate	9600
Parity	No
Data Bit	8
Stop Bit	1

Based on our procedural definition developed as a proposed standard, the physical interface between the i-bot vision system and the PUMA robot is commensurate with an RS 232C serial interface port.

2. Data Transfer Protocol

The communication between the ibot and Puma necessiates the use of software to handle the byte transmission and data acknowledgement. Protocols are needed to calibrate the robot and to down-load or up-load a program from off-line disks. The data transfer protocol should be flexible enough to handle all such tasks. Considering the software design between the ibot and PUMA system, the data transfer protocols can be described as follows:

a. Robot-in and Robot-out

The lowest level of robot control has protocol software which sends data to and from the robot system. The current state of robot control assumes an RS 232-C asynchronous serial data link between the i-bot system and the robot controller. The code consists of two subroutines, one for getting a byte of data from the robot controller and one for sending a byte to the robot controller. (as shown in appendix 1 "remote.asm")

b. Robot Mode (Talk to Robot)

Robot Mode is a terminal emulator program. It consists of two subroutines : (1) (Robot md) which is the over-all control of Robot Mode and the (2) (Robot comm) is the terminal emulator. (as shown in appendix 2)

The Robot-in and Robot-out routines described above correspond to the low-level communication protocol of the proposed standard, whereas the Robot mode routine corresponds to the high level communication protocol of the proposed standard.

3. Calibration Software

The objective of the data transformation is to effectively calculate scaling factors to be used to map video pixel element (pel) locations to robot reference frame spatial locations.

a. line-of-sight problem

In order to accurately define a 3 dimensional point in space, the two dimensional data provided by the imaging plane is used in conjunction with information provided by the photosensor beam in the tips of the gripper mechanism attached to the robot. Any spacial reference point can be viewed as lying on a vector between the point itself and the imaging camera lens. This vector is referred to as the "line-of-sight" vector (Fig. 4.9). The robot moves to a point on this vector which intersects the Z1 plane. The path will be along the "line-of-sight" vector towards the ZO plane. When the photosensor beam is interrupted by the part, the gripper jaws are closed and the part is retrieved.

b. Scaling Problem

There are 4 calibration points taken from the ZO plane and 4 more taken from the Z1 plane. The points taken are as shown in Fig. 4.10. When the points are defined, the distances calculated (pixels by the vision system (referred to as dVX) and millimeters by the robot system (referred to as dRX)) are used as a ratio to compute the X, Y scaling. The ORS i-bot 1 vision system handles the scaling factorsin the way described above. Different designers may come out with different methodologies. The scaling factors derived from the 4 calibration points will transform the X, Y, O data in a vision system reference frame into a robot's reference frame. For example, in the top plane the X, Y, O data which correspond to a vision reference frame will be multiplied by scaling factors SX and SY. Thus the X * SX and Y * SY data will correspond to the point which is relative to the robot's reference frame. This would then correspond to the proposed standard.

4. Coordinate Transformation Procedure

Before data is sent to the robot controller, a check is made using the routine "robot-diagnostic" to see if the robot system is ready to accept data. Move Robot scales the coordinate data and passes the new coordinates to the robot cotroller. Move Robot is a software routine which uses "top-set-robot" subroutine to send the top plane coordinates which are scaled by top plane scaling factors SX and SY, and "bot-set-robot" subroutine to send the bottom plane coordinates which is scaled by bottom scaling factors SX and SY. Control is then passed to "robot-acquire" which starts the robot controller based acquisition program. (as shown in appendix 3)

These routines in the i-bot 1 vision system correspond to the previously defined procedure protocols. They transform the data in the i-bot 1 vision reference frame into a robot's reference frame and then transfer the data to the robot's controller.

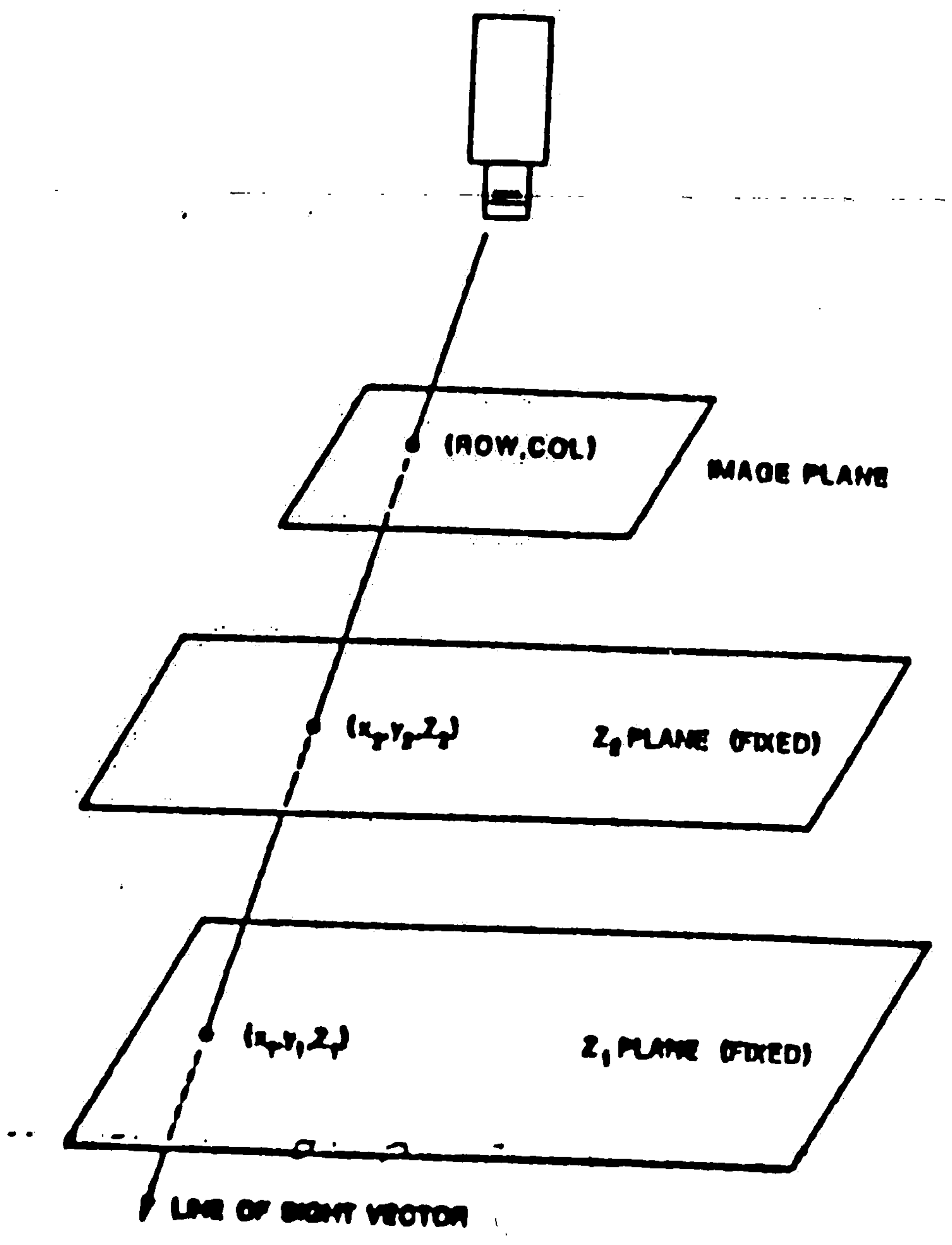
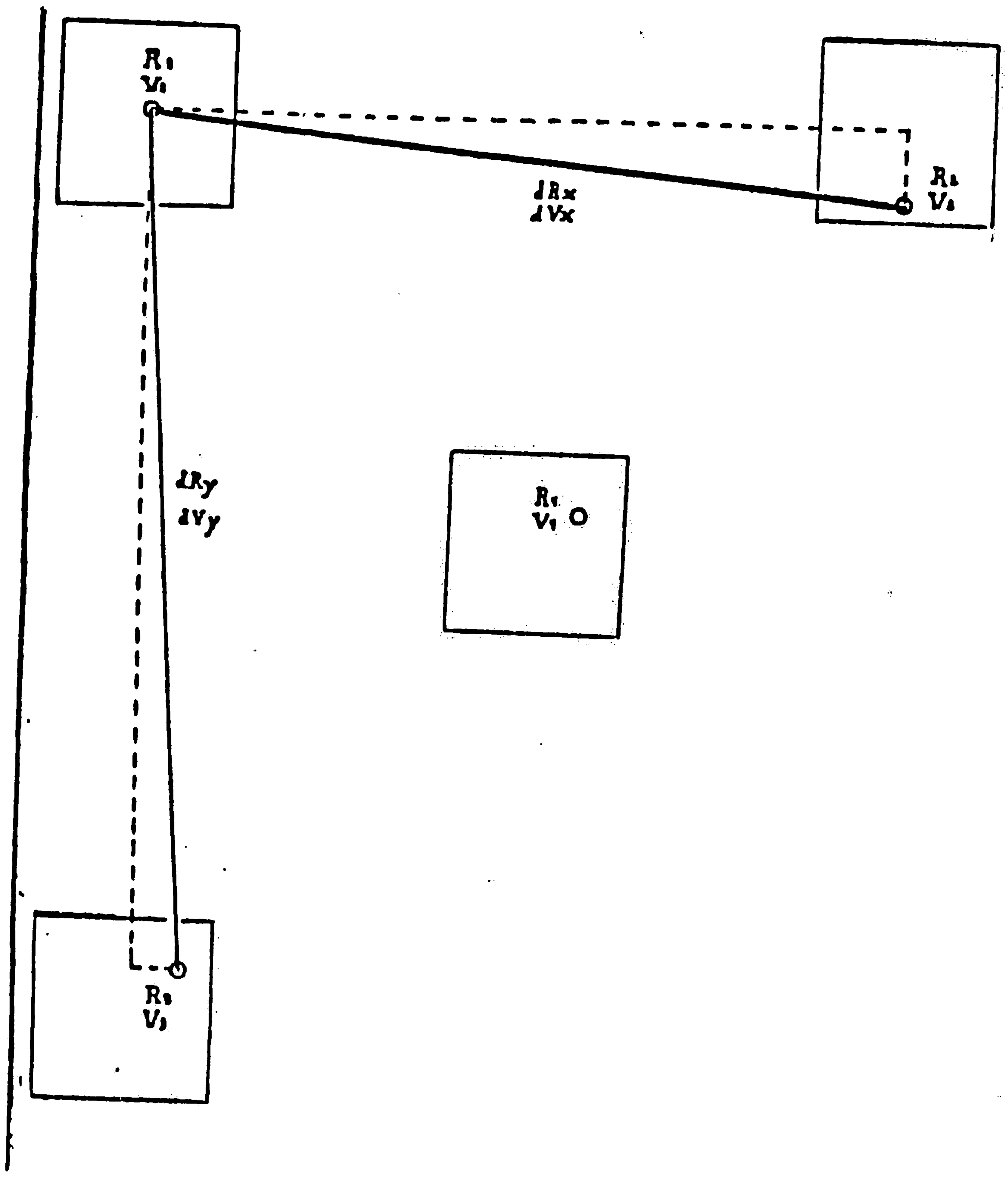


Fig. 4.9 Line of Sight Vector [14]



$$SX = \frac{dRX}{dVX} \qquad SY = \frac{dRY}{dVY}$$

Fig. 4.10 Relative Measurements [14]

4.5 I-BOT to GE-P50 Interfacing Approach

The ORS i-bot vision system which was specifically designed to be used with a PUMA robot is a Robot Dependent system. If a robot by a different manufacturer is used, problems arise in command structure (I/O) and signal controls such that a new interface must be designed. Such problems are typical and are a disadvantage of Robot Dependent system. Based on the previously defined procedure and definitions, the approach taken in this thesis is to interface the i-bot vision system with the GE P50 robot. The GE P-50 robot available for this study did not have an RS 232C interface port. The approach taken was to design a simplified interface with the goal to send X, Y positional data and orientation data to the GE P50 robot. This interface was designed such that one can program the robot to move to these coordinates.

1.) GE P-50 ROBOT I/O CAPABILITIES

Signals can be exchanged between the GE P-50 robot control unit and a peripheral device through the manufacturer supplied single cable (CNEX) as shown in the Figure 4.11. An I/O pulse is defined for the CNEX cable as 40 ms. Arbitrary pulse lengths can be created using timers.

There are 16 available input signals for the GE P-50 robot. Input signals 0 through 7 can be programmed via the teach box using the IN key. Input signals are received from external units by using the ENTER key on the operation panel. Enter 0 through

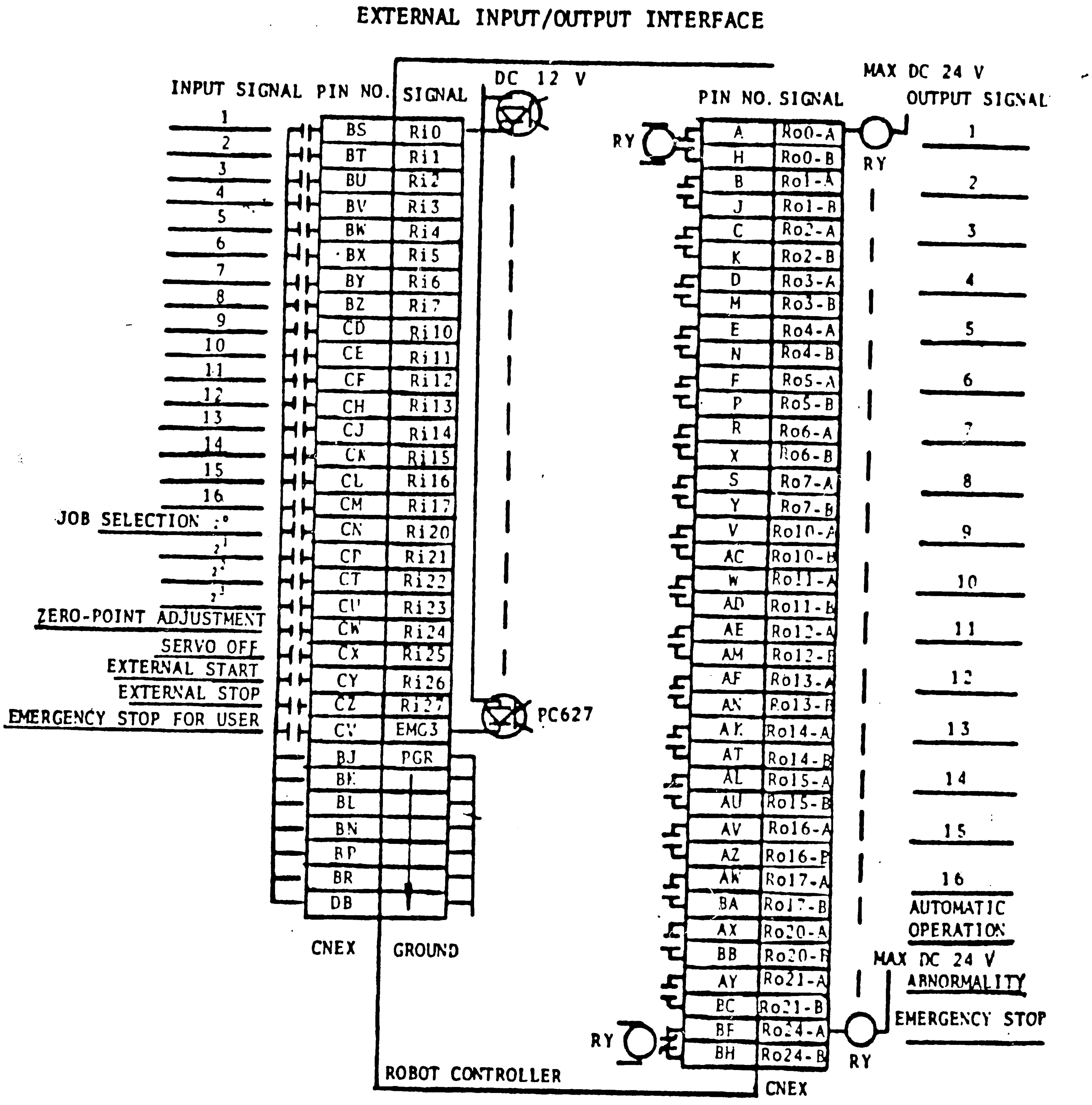


Figure 4.11 Input/Output Cable Connection
for the GE P-50 robot [15]

7, or ENTER 192 through ENTER 199 are used as the first eight input commands in a job. The next eight available signals are ENTER 200 through ENTER 207. Fig. 4.12 indicates the input circuit connection from the CNEX cable to the robot controller. The input circuit connector enables the P-50 robot to interact with the outside environment.

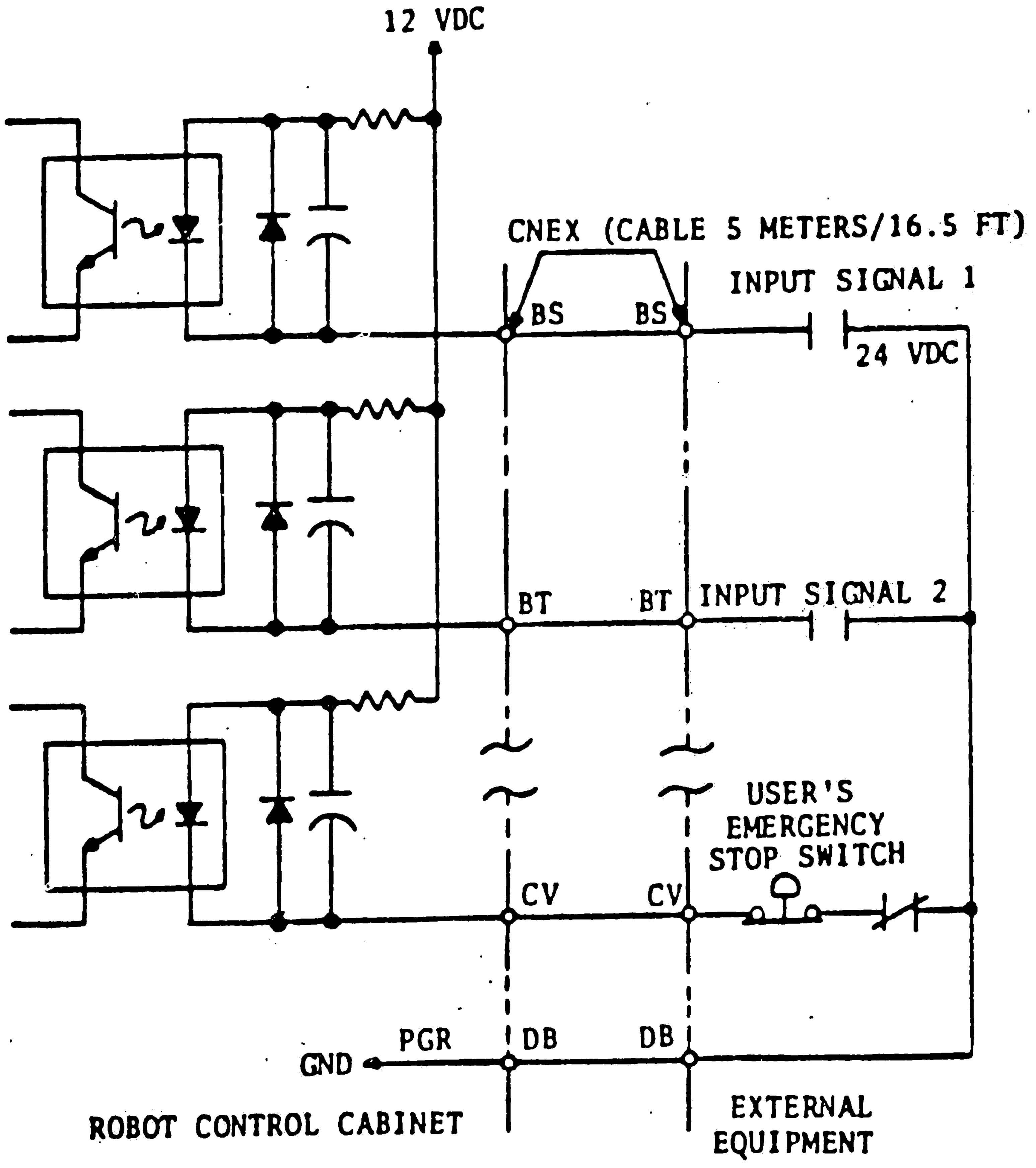


Fig. 4.12 Input Circuit Connection [15]

In addition, there are 16 available output signals for the GE

P-50 robot. Output signals are sent to external units by use of the OUT key. Fig. 4.13 details the output circuit connection. Thus, a total of 32 available I/O bits can be accessed through the use of the ENTER 192 through ENTER 207 and OUT 16 through OUT 31 commands. Appendix 4 gives a detail description of the I/O Assignments and Functions as supplied by General Electric for the P-50 robot.

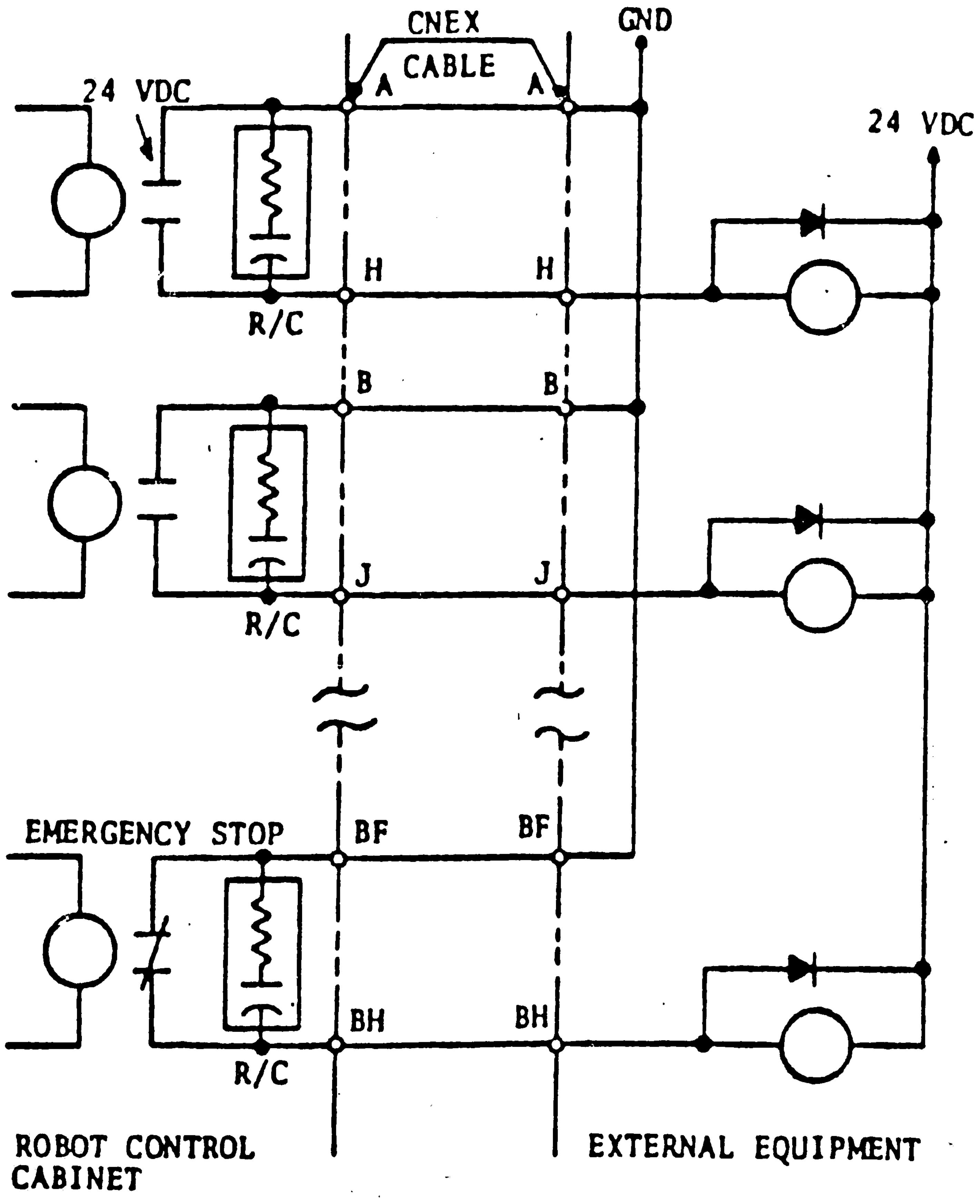


Fig. 4.13 Output Circuit Configuration [15]

The output contact rating of the CNEX cable is 1 amp at 24 VDC (resistive). Although each contact is provided with surge suppression, a diode should be used with inductive loads (e.q. relay) to insure sufficient protection against large inductive switching transients.

Other input signals include the job select bits. These form a 4 bit register to indicate a job to be performed. The command CALL X will execute the job indicated in the job select register. These are inputs 208-211. Input 212 is for external initiation of the zero point adjustment procedure. This also serves as a servo on signal. Servo off, enternal start and stop and emergency stop inputs are also available. Output bits indicating automatic operation and emergency stop exist also.

2.) INTERFACE DESCRIPTION

To develop an interface between the GE P-50 robot and the ORS vision system involved a design based on the I/O capabilities described in the preceeding section. Although these I/O bits are primarily for discrete process control applications, there use in the designed interface is necessary since this model of the P-50 robot does not provide for any other input. (Later models of the P-50 robot include an RS-232 port, simplifying this process.)

a.) Physical Interface

During the interface design procedure, inputs 200-207 were set up to handle an 8 bit input word. Writing a P-50 robot program would then transfer data for these inputs to internal shift registers. There are 16 of these registers each with an X,Y and Z component. The purpose of these registers is to offset a programmed robot move. Thus, if the robot is programmed to move to point (X_O, Y_O, Z_O), the shift register values will cause it to move to $(X_{O+X}, Y_{O+Y}, Z_{O+Z})$. Since the shift is relative, it is possible to program the point of origin. The serial to parallel conversion, timing, signal coordination and overall control is performed by a IBM XT microcomputer.

The IBM PC/XT system was selected for both its ease of interfacing through an INTEL standard 8255 prallel I/O port, and its speed of operation. The programming was done in Turbo Pascal due to the I/O structure of the program when used with the XT. This structure is a logical device structure using the PC-DOS operating system. In the PC-DOS operating system, all devices attached to the machine are "configured" into the operating system as logical devices, therefore making reading from and writing to these devices quite simple. Based on the previously defined procedure, the physical interface should be through an RS 232C cable. Since the GE P-50 of this study did not have an RS 232C interface (Later models include an RS 232C cable), the physical interface as described above was utilized.

b.) Data Transfer Protocol

1. Low-level Communication protocols

A. P-50 Software

The purpose of the P-50 programs is to transfer the data from the previously mentioned input word (discrete bits 200-207) to the appropriate shift registers. This is accomplished by 'polling' the bits of the input word and adding the appropriate value (2**N) to the appropriate shift register. There are six jobs, each one for +X, -X, +Y, -Y, +θ, -θ. The job to be executed is determined by the programmable controller and indicated by the job select register.

Upon the occurance of an external start signal, the P-50 will be in Mode X, and Job 000 will be executed. The following listing details the program if JOB 000 which inputs the X, Y, O data from the i-bot vision system to the GE P-50 robot controller.

JOB 000

```
 Program                       Description

  Do 3                          * Once for X, Y and θ
   Begin
    Out 24                      * Send the Data Request bit
    If ENTER 192                * If data ready
     Then
      If ENTER 193              * If data not zero
       Call   X                 * Execute job in job select
    End                           register.
   Pfm   001                    * Programmed  move   to
   Job End                        Origin.
```

This program loops three times to input data for X, Y and θ respectively. The command Call X imbedded within the program serves to call Jobs 1 through 6 which will perform the tasks of manipulating the positive and negative X, Y and θ data.

Programs for JOBS 001 and 002 (for +X and -X data transfer) are listed below :

Job 001   (+X)

```
Program                          Description

SEL   MREG 001                   * Select shift register one
                                   (there are 16)
UNIT MREG 008                    * Unit is 8 X .125 mm =1 mm
                                   (.125 mm is minimum unit)
CLR   MREG  001    01            * Set  X  component  of
                                   register one to zero

If ENTER 200                     * Poll first bit.
 Then
  Begin
   Const 001                     * Constant = 2**0 units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.
 End

If ENTER 201                     * Poll second bit.
 Then
  Begin
   Const 002                     * Constant = 2**1 units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.
 End

If ENTER 202                     * Poll third bit.
 Then
  Begin
   Const 004                     * Constant = 2**2 units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.
 End
```

```
Program                        Description

 If ENTER 203                      * Poll fourth bit.
  Then
   Begin
    Const 008                      * Constant = 2**3 units.
    Add   MREG   001   01          * Add  constant  to  X
                                     component of first shift
                                     register.
  End

 If ENTER 204                      * Poll fifth bit.
  Then
   Begin
    Const 016                      * Constant = 2**4 units.
    Add   MREG   001   01          * Add  constant  to  X
                                     component of first shift
                                     register.
  End


 If ENTER 205                      * Poll sixth bit.
  Then
   Begin
    Const 032                      * Constant = 2**5 units.
    Add   MREG   001   01          * Add  constant  to  X
                                     component of first shift
                                     register.
  End



 If ENTER 206                      * Poll seventh bit.
  Then
   Begin
    Const 064                      * Constant = 2**6 units.
    Add   MREG   001   01          * Add  constant  to  X
                                     component of first shift
                                     register.
  End

 If ENTER 207                      * Poll eighth bit.
  Then
   Begin
    Const 128                      * Constant = 2**7 units.
    Add   MREG   001   01          * Add  constant  to  X
                                     component of first shift
                                     register.
  End
```

JOB 002 (-X)

```
Program                          Description

SEL   MREG 001                   * Select shift register one
                                   (there are 16)
UNIT MREG 008                    * Unit is 8 X .125 mm =1 mm
                                   (.125 mm is minimum unit)
CLR   MREG  001    01            * Set  X  component  of
                                   register one to zero

If ENTER 200                     * Poll first bit.
 Then
  Begin
   Const -001                    * Constant = -(2**0) units.

   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.

 End
If ENTER 201                     * Poll second bit.
 Then
  Begin
   Const -002                    * Constant = -(2**1) units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.

 End
If ENTER 202                     * Poll third bit.
 Then
  Begin
   Const -004                    * Constant = -(2**2) units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.

 End

                                   register one to zero

If ENTER 203                     * Poll fourth bit.
 Then
  Begin
   Const -008                    * Constant = -(2**3) units.
   Add   MREG   001   01         * Add  constant  to  X
                                   component of first shift
                                   register.

 End
```

```
Program                         Description

If ENTER 204                       * Poll fifth bit.
 Then
  Begin
   Const -016                       * Constant = -(2**4) units.
   Add  MREG  001  01              * Add  constant  to  X
                                     component of first shift
                                     register.
 End

If ENTER 205                       * Poll sixth bit.
 Then
  Begin
   Const -032                      * Constant = -(2**5) units.
   Add  MREG  001  01              * Add  constant to  X
                                     component of first shift
                                     register.
 End

If ENTER 206                       * Poll seventh bit.
 Then
  Begin
   Const -064                      * Constant = -(2**6) units.
   Add  MREG  001  01              * Add  constant  to  X
                                     component of first shift
                                     register.
 End

If ENTER 207                       * Poll eighth bit.
 Then
  Begin
   Const -128                      * Constant = -(2**7) units.
   Add  MREG  001  01              * Add  constant  to  X
                                     component of first shift
                                     register.
 End
```

The corresponding Y and θ operations will be identical programs except that the appropriate offset registers for the Y and Z will be substituted for in the following table where applicable.

AXIS	MREG
X	001 01
Y	001 02
Z	001 03

B. PC XT <=> P50 COMMUNICATIONS

The GE P-50 robot receives positional data from the microcomputer as a positive binary number. This positional data, the job select data, and the other control bits are transferred from the micro to the robot via a pair of parallel ports on the micro. These ports require some hardware to generate the appropriate voltages for the robots input. Such hardware includes a set of line drivers to transform the 5 volt which is the output from the micro to the 24 volt signal which is required in the robot's circuit . The P-50 also generates a data request bit when data is needed, and the micro replies. A zero flag is also needed if the X, Y or Z coordinate data is zero.

2. High-level Communication Protocols

IBM XT PROGRAMMING

A program was developed to read ASCII data through the RS-232C port on the IBM XT from the ibot vision system. The function of the program was to convert the input data to the offset notation necessary for the GE P-50 robot. The program generates the offsets, appropriate job select number, zero flag, and start bits and then output them to the robot via a pair of parallel ports through the CNEX cable. One port was dedicated for the offset value, and other port was used for the necessary bits listed above.

(i.e. job select number, zero flag ...etc.) With minor modifications this software could be adapted to other sources (e.q., other vision system) outputting positional data. The inputting device of this study (i-bot) has been configured to have the data available as a binary word on the RS-232 port of the IBM XT. Other possibilities (ASCII coded data is common) could easily be handled by the micro computer program. (subtract 40H and multiply by the appropriate power of 10 for ASCII digits) This type of programming was chosen because of the flexibility of handling different types of input.

Listed below is a program that takes the output from the RS-232 device, properly formats it for the robot controllers input, and transfers the formatted input to the robot. In this case, the input device is the i-bot vision system. The program is written in Turbo-Pascal.

```
                        program robot;

   Program                                   Description

var
  offset,  control:byte;                   * Offset  and control are
                                             defined as a byte.
  job,  inword,  count:integer;            * Job, inword and count
                                             are integers.

procedure readbyte;

  begin

    read(usr,inword);                       * Input the data from i-
                                              bot
  end;
```

```
Program                                        Description

procedure make_offset_output_byte;

  begin
    if  inword < 0 then                        * Check if the data  is
                                                 positive or negative
        begin                                  * If negative,  select
                                                 the right job number
          job:=job+1;
          inword:= abs(inword);
        end;
  end;



  begin                                        * Input the X, Y and O
    while 1<>2 do                                from the i-bot
      begin
        for count:=1 to 3 do
          begin
            readbyte;
            job:=(count*2)-1;                  * Select the job number
            make_offset_output_byte;
```

Comment : If any unit conversions are necessary, they are to be done repeatedly, i.e., above program is repeated (while statement to make-offset-output-byte), control is then passed back to the main program as follows :

```
offset:=inword;
```

Comment : According to the data input from the i-bot vision system, a control word is assigned for job selection, the purpose of which is to branch the job for +X, +Y, or +O manipulation. The control word consists of 8 bits, each of which is assigned by the programmer. The bit pattern is as follows :

The control word looks like this:

bit	purpose
0	used for job select
1	"
2	"
3	"
4	0 flag
5	start
6	unused
7	unused

Comment: The following subprogram assign the control word :

```
Program                               Description

    control:=  0;                     * Set the control word
    control:=control + job;           * Set the job number

    if inword:= 0 then

      control:=  control  +  16;      * Set  0  bit  if
                                        appropriate

    control:=control + 32;            * Set the start bit

    write(lst,offset);                * Output the data to
    write(aux,control);                 GE P-50

  end;

 end;

end.
```

According to the previously defined procedure and definitions, the program developed is compatible to the low-level and high-level data communication protocols for the transfer of ASCII data.

c. Calibration Software and Coordinate Transformation Procedure

Having developed the physical interface and the low-level and high-level communication protocols, the calibration software and coordinate transformation procedure would involve much programming detail in the ORS i-bot vision system. Due to the program command structure in the GE P-50 robot, the send-out command from the i-bot calibration software routines must be changed into the offset distance data. For example, the send-out command "move X1", will be changed into the offset data "offset-X, offset-Y" for the P-50 robot to move to the defined position X1. The scaling factors then can be achieved through this modified calibration software. Based on the previously defined procedure, the X, Y and θ data sent out from the i-bot vision system can be transformed into the robot's reference frame by multiplying the X, Y data by the scaling factors. Such a software routine would be compatible to previously defined procedure as a coordinate transformation procedure.

4.5.1 Analysis of Results

The approach in interfacing the i-bot vision system to GE P-50 robot controller presented the following problems :

1. No physical interface standards : An RS 232-C communication port was not available in the GE P-50 robot of this study. An interface was developed to accommodate signal commands between the vision system and an IBM XT with subsequent data transfer to the robot controller.

2. Data transfer protocol : Due to the program command structure in the GE P-50 robot, the high-level communication software in i-bot vision system was modified to be able to output the GE P50 commands instead of the VAL commands. An IBM PC-XT is proposed as a host computer for communicating the information between the i-bot vision system and the GE P-50 robot.

3. Calibration Software : The calibration software would involve much programming detail. Since the thrust of this research was to develop grneric stratefies for robot-vision interfacing, it was not considered necessary to fully develop such software routines. Rather, it was identified that it was feasible to develop the software to transform X, Y and O data in a vision system reference frame to the GE P-50 reference frame. Based on the previously defined procedure, one can write down the calibration software (code) satisfy the line-of-sight and scaling requirements.

4. Coordinate transformation procedure : Owing to the different application purpose of the P-50 robot, the I-BOT software must be modified to be able to send out seam tracking data to the GE P-50 for welding feedback control. The basic concept would be to send out X, Y and Orientation data from the i-bot vision system to the GE P-50 robot. Once such data is transferred, it can be modified according to the application purpose. The transformation procedure would then be the same as in the bin-picking application.

The above problems list indicates that the robot dependent or the vision dependent methodologies do not have the flexibility for free exchange of information between different vision systems and different robot controller. A modular system is proposed in this paper. The modular system provides a method of connecting controllers in distributed control applications. By means of standard interfaces and specification of useful hardware and software interface, the modular controller can link together many intelligent parts of industrial work-cells, distributed motors, or device controllers.

CHAPTER 5

GENERIC APPROACH TO ROBOT-VISION INTERFACING

Current manufacturers typically define either a robot dependent system or a vision dependent system when interfacing a vision system to a robot. Such an approach has led to various problems in interfacing between specific vision system and specific robot controllers. For example, one problem that arises is the interface of a "vision dependent" robot to a "robot dependent" vision system. In the ORS i-bot 1 vision/Puma robotics system used in this study, the vision system is robot-dependent, sending VAL commands to the slave, Puma robot. If the Puma is replaced by the GE P-50, all the commands transmitted by the data transfer protocol and transformation software in the i-bot vision system must be changed as discussed in chapter 4. System users are typically not familiar with the data transfer protocol and transformation software required between general vision system and robot controllers. The purpose of this study is to define a vision-robot system which is plug-compatible and easily implemented.

Because of problems with the robot dependent and the vision dependent systems, a modular algorithm is preferable. A modular algorithm is similar to the integrated system briefly described in chapter 4. The main difference between a modular system and an integrated system is that the modular system has modular structured software designed which allows a flexibility in choosing different robots for different vision systems. Integrated systems of today do not provide such flexibility.

5.1 Relationship Between the Vision System and a Robot Controller

The relationship between a robot and a vision system in the modular system is totally different from the vision dependent (master-robot, slave-vision) or robot dependent (master-vision, slave-robot) systems. The concept of the proposed modular system is based on distributed processing and hierarchical control. The vision controller performs data gathering, image analysis, part identification and orientation. The vision controller is driven by an external computer, and the system controller makes decisions based on this information. The system diagram of this proposed system is shown in Fig. 5.1.

In the proposed approach, both the robot controller and vision controller are configured strictly as peripherals. To the controlling computer, the robot and vision controller appear electrically identical to a computer terminal.

5.2 System Architecture in the Proposed Modular System

The proposed system, illustrated in Fig. 5.1 , consist of a microcomputer with a large memory (e.g. hard disk) for software communication and application software development. Communication between a robot controller and a vision controller

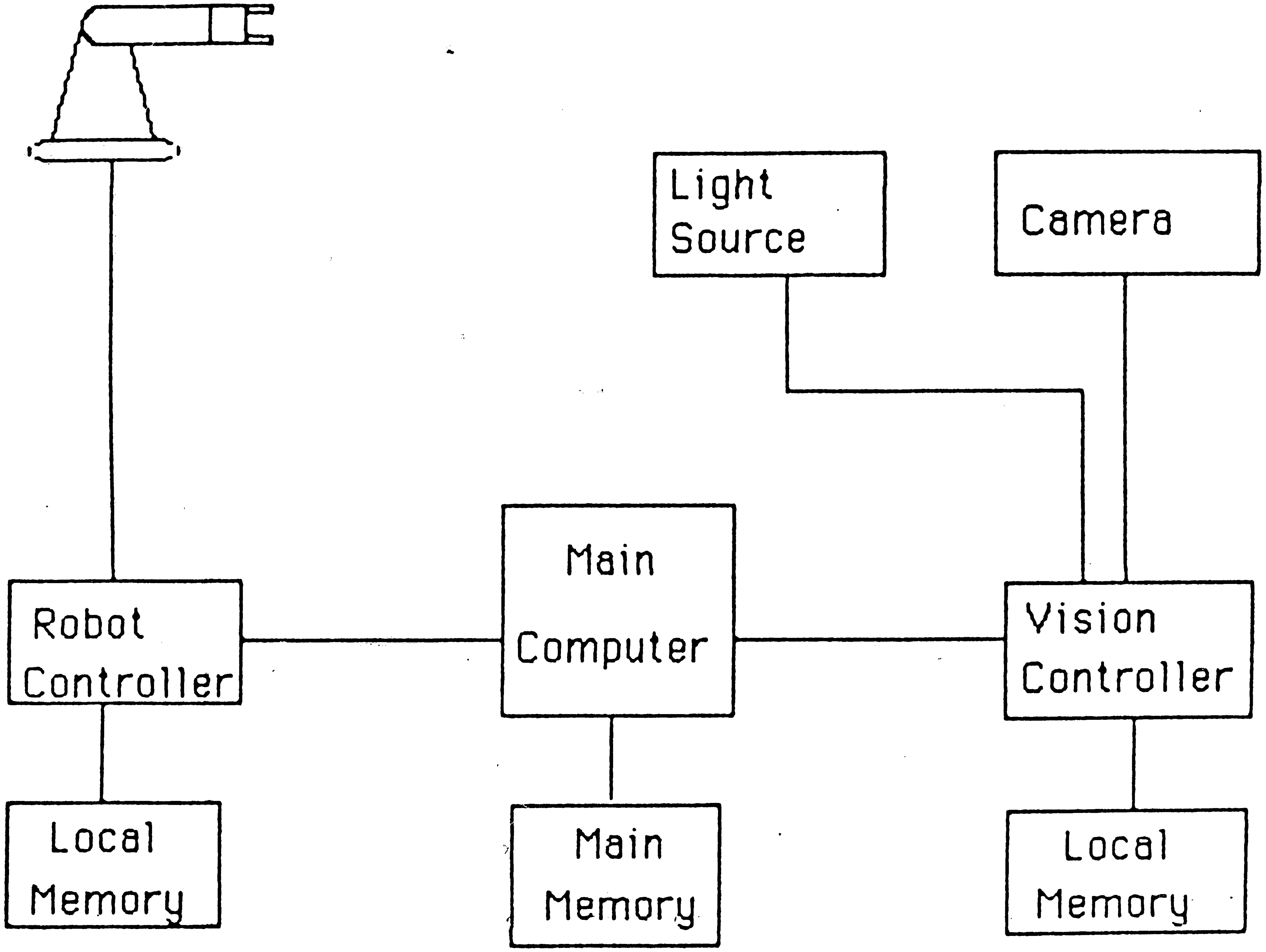


Fig. 5.1 A Modular Interfacing System Diagram

to the computer is through the RS-232C standard serial communication line. This communication protocol was selected because it is, in essence, a de facto interface standard that nearly all computers are equipped with. The time taken to communicate between the controller to the external computer is only a few milli-seconds when running at 9,600 Baud, adding no significant overhead to the overall cycle time.

The IBM PC XT was preferred as the control computer because of its 10 Mega byte hard disk and ease of expansion. Other microcomputers with similar compatibility are just as usable. The expansion unit in the IBM XT contains a power supply, an expansion board, and a receiver card. The IBM asynchronous communications adapter can be used for the communication between the robot and vision controller to the XT microcomputer. The different modes of operation may be selected by programming the 8250 asynchronous communications element. This is done by selecting the I/O address and writing data out to the card.

The communications adapter provides an EIA RS 232-C interface. One 25-pin D-shell, male type connector is provided to attach various peripheral devices. While in the vision and robot system, there must be an RS-232-C I/O port for data communication.

5.3 Software Communication System

According to the previously defined procedure and definitions given in chapter 4, the interface between a vision system and a robot's controller can be implemented as Fig. 5.2. This interface includes the following features :

(1.) Physical Interface

(2.) Data Transfer Protocol

a. Low-level Communication Protocol

b. High-level Communication Protocol

(3.) Calibration Software

(4.) Coordinate Transformation Procedure

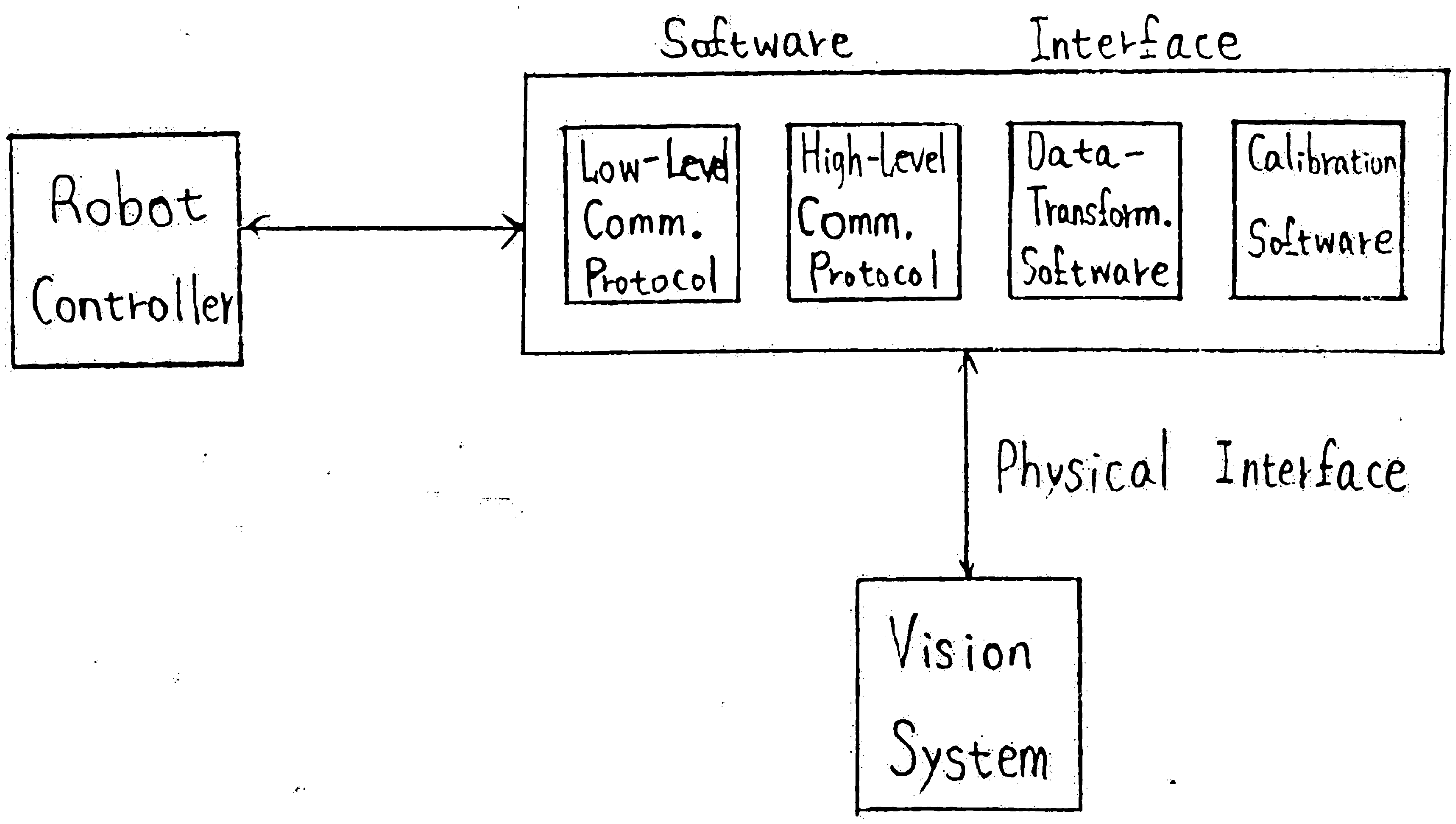


Fig. 5.2 The interface between a vision system and a robot controller

In the proposed modular system, the host computer is in charge of the communication between a vision system and a robot's controller. The data transfer protocol, calibration software and coordinate transformation software will be developed and saved in the hard disk of the host computer.

To provide a compability for implementing a modular interfacing system, software modules can be expanded and designed as illustrated in Fig. 5.3. Each block in such a modular structure can be accessed by the process control program on the master computer (IBM XT). This process control program adaptively controls robot motions based on image processing information.

The control computer contains various levels of modular software, such as Application specific software, coordinate transformation software, low-level communication software, signal processing software, low-level data acquisition software, and Vision calibration software. The detail in each block will be as following :

1. Low-level Communication Software

   This software supports asynchronous communication between two stations (robot/computer, vision/computer). It guarantees transfer of error-free data. It allows the transfer to be initiated by either station.

2. High-Level Communication Software

   The software reads and writes the messages among the host computer, a vision system and a robot's controller. The messages include the following items:

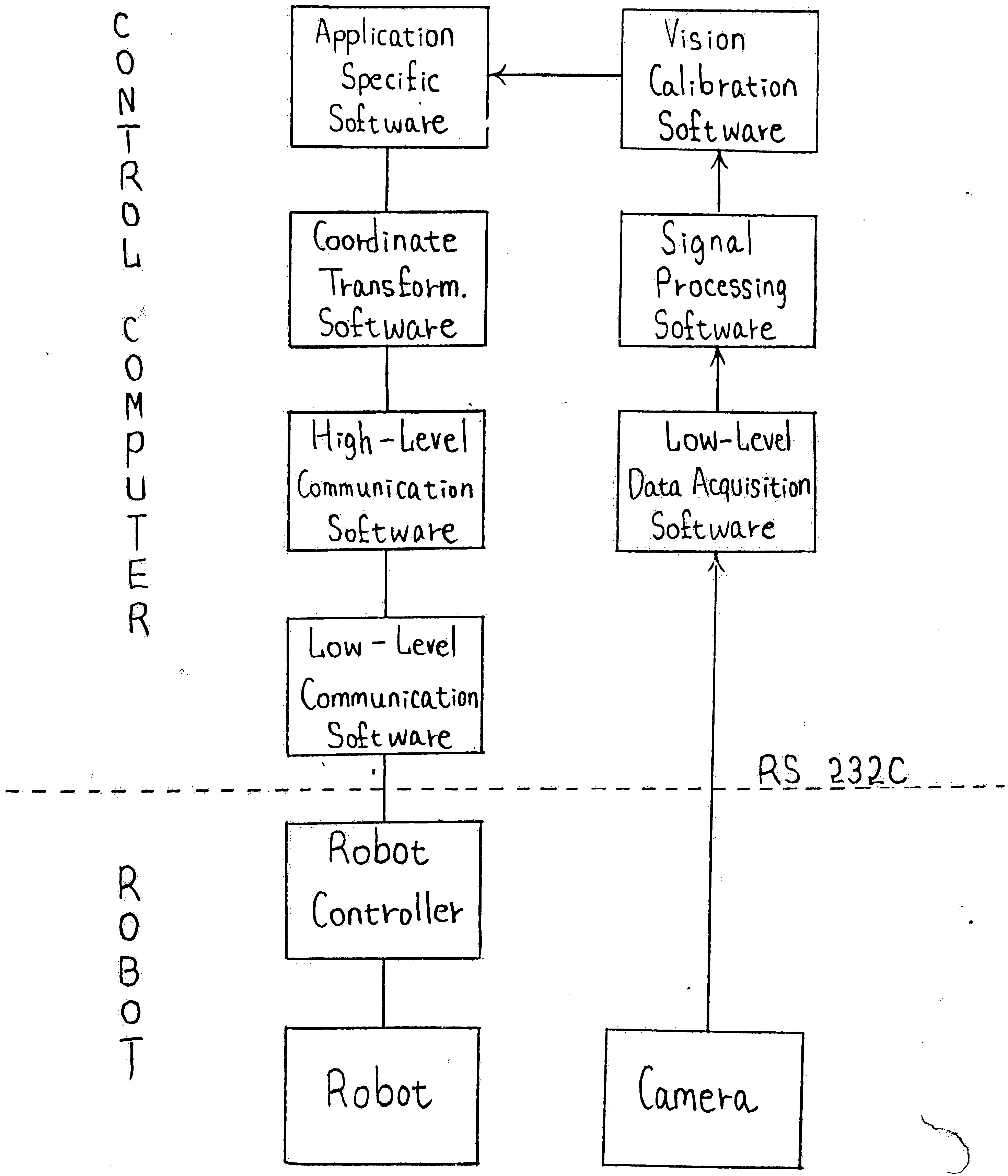

CONTROL COMPUTER
Application Specific Software
Vision Calibration Software
Coordinate Transform. Software
Signal Processing Software
High-Level Communication Software
Low-Level Data Acquisition Software
Low - Level Communication Software
RS 232C
ROBOT
Robot Controller
Robot
Camera

Fig. 5.3 Software Block Diagram of the Modular Interfacing System

a. Send a program to the robot from the host computer.

b. Receive a program from the robot at the host computer.

c. Start the program that currently resides in the robot memory.

3. Coordinate Transformation Software

This software module is to transfer the data captured in vision system into the robot-readable information. The module should provide conversion for different robot application.

4. Low-level Data Acquisition Software

This level of software enables the system's ability to communicate with vision sensor. The software module is specifically written to meet the characteristics of the vision sensor, such as transmission speed, encoding of data, and ensuring data integrity.

5. Signal Processing Software

This module provides processing of raw sensor signals to provide information directly usable at a higher level. For example, this processing is performed in the output of vision sensors to provide recognition or location information.

6. Application Specific Software

This software is process specific, because it controls the sequence of operations to be performed. Typically, it is a set of modules that uses some, or all, of the previously described modules to create a fully integrated program that supports all the functionality required by the system. Tasks that must be performed at this levle are :

- Assembly
- Inspection
- Welding
- Bin-Picking

7. Vision Calibration Software

In order for the robot to a:quire parts located by the vision system, there must exist a common cartesian frame of reference. This is accomplished by the vision calibration software. The procedure is as the following. First, the vision system is calibrated to establish an accurate two dimensional coordinate system in the field of view. Next, the camera coordinate system must be related to the coordinate system of the robot. This is accomplished by determining a camera-to-robot coordinate transformation. A coordinate transformation is an operator which describes transformation an rotation of a coordinate system from one location to another. Thus the camera-to-robot transformation describes the transformation and rotation required to bring the camera coordinate system into coincidence with robot coordinate system.

The proposed modular system includes a main program which is in the application specific software. The main program is an interactive program containing a menu. The menu is a list of tasks that may be executed by the system, such as welding, assembly, inspection and bin-picking.

Once these choices are made, the task program imbedded in the application specific software module is run. The task program is called by the main program. The task program oversees the actual operation of the robot vision system in current use. The task program can be edited and modified according to the job requirements. For example, in the assembly work users can modify the task program according to the application purpose or even rewrite the task program and save it in the hard disk.

The calibration software is written down for solving the calibration problem between the vison system and robot controller. For different application purpose, e.q. inspection or welding, the vision system could be two-dimensional or three-dimensional. There are two calibration routines. One is for a two-dimensional vision system and the other is for a three-dimensional calibration system. The proposed calibration procedure will be the same as that used for the i-bot vision system described in chapter 4. After finishing the calibration procedure, the scaling factors SX, and SY (or SZ in a three-dimensional vision system) will be determined. Once the scaling factors have been determined, the coordinate transformation software can transform the data from the vision system to the robot controller by multiplying the data by scaling factors.

Based on the proposed software model described above, the host computer is equivalent to a task controller. The host computer interfaces with the vision system by way of a request and a feedback buffer for information exchange. Based on the data transferred, the host computer will decompose the task commands into low-level control commands for the robot controller. The more levels that the task is decomposed, the more detailed command structure of the system will become. As proposed by M. L. Fitzgerald and Anthony J. Barbera of the National Bureau of Standards for a real time robot controller [31], the task is decomposed into five levels as showed in Fig. 5.4. These levels consist of (1) task, (2) elemental move, (3) primitive, (4) coordinated joint, (5) servo. After the task is decomposed, the robot controller will transfer the command signals to the robot arm step-by-step. In such a low-level control interface, the proposed modular software system will provide the ability to be able to integrate a different robot with a different vision system. The proposed plug-compatible system then can be implemented.

In an automated manufacturing system, the current trend id for the architecture of the control system to be arranged hierarchically. Each controller takes commands from the next higher-level in the system. The tasks can be entered from the highest level and are decomposed down into subtasks. The subtasks are executed at that level,and, if necessary, are further decomposed into commands as output for the next lower level. The proposed modular system, which includes a robot controller and a vision system, can be integrated as part of such a hierarchically structured automated manufacturing system. The proposed control computer for a vision system and a robot controller will be able to respond in two hierarchical directions

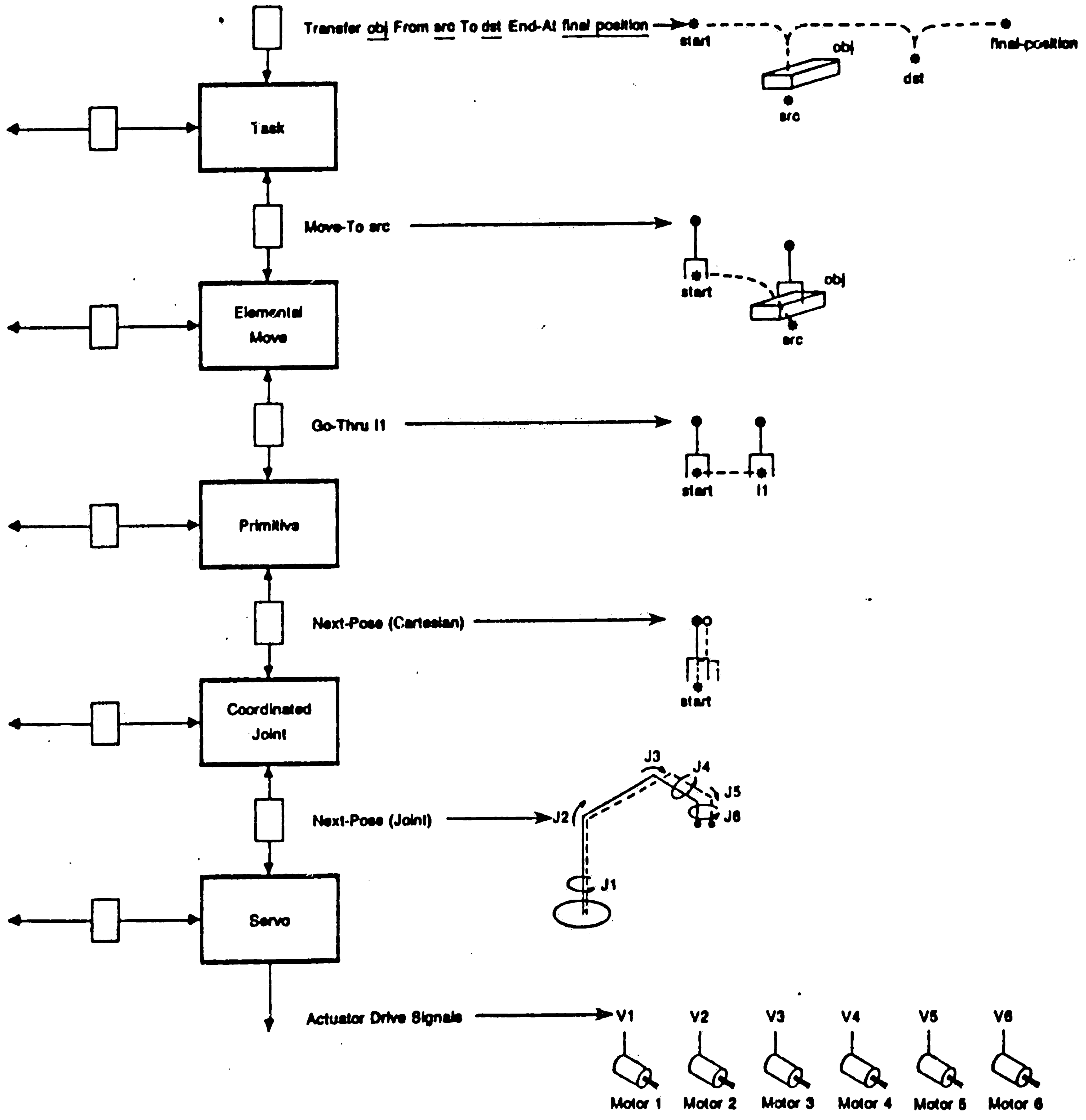


Fig. 5.4 The five generic control levels between the Robot Task Controller and the Robot Joint Controller. [31]

: 1) down the hierarchy to a vision system or a robot controller and 2) up the hierarchy to a higher-level computer.

By means of the proposed modular software interface and low-level control interface, the proposed system can be linked together with other intelligent work-cells. With such interchange ability, the system can become part of a larger system, and can quickly respond to requests for data from the higher level factory computer. Present technology does not provide the flexibility to easily integrate the components of industrial work cells. However, the proposed modular software system and low-level control interface, combined with advances from increasing technology, will assure the implementation of a total network computer control system.

CHAPTER 6

CONCLUSIONS

The results of this research showed that the interfacing of a vision system to a robot controller is directly dependent on the following features :

a. Physical interface

b. Data transfer protocol

c. Calibration software

d. Coordinate transformation procedures

Because of the variation between vision systems and robots, designers interface their systems in different ways. Such methods include Robot Dependent method, Vision Dependent method and Integrated method. There are advantages and disadvantages in each system. Users can choose their interfacing method based on the application purpose.

An procedure was presented to illustrate interfacing of an i-bot vision system with a GE P-50 robot. As per the discussion in chapter 4, the designer will be confronted with data communication and signal controlling problems. A common disadvantage exists in these systems (Robot Dependent, Vision Dependent, and Integrated Method), in that there is no plug compatibility in interfacing different robot controllers with different vision systems.

A modular system for a generic interface between any vision system to any robot controller has been proposed in this research. The architecture of the modular system is that there is a control computer which communicates back and forth with the robot and vision system. The control computer functions as a system coordinator. The robot and vision controller is configured strictly as a peripheral to the control computer.

The software to generically interface a robot and vision system is proposed to be of a modular structure. In each module, the information exchange and signal control are interchangeable. The proposed software modules are as follows :

1. Low-level Communication Software
2. High-level Communication Software
3. Coordinate Transformation Software
4. Low-level Data Acquisition Software
5. Signal Processing Software
6. Application Specific Software
7. Vision Calibration Software

In the modular interfacing system proposed, designers can interchange robot and vision systems. The flexibility of the proposed generic interfacing method for robot and vision systems should lead to compatibility in an automated manufacturing facility.

CHAPTER 7

RECOMMENDATIONS FOR FUTURE RESEARCH

This thesis has considered the design of a generic interface between a general robot and a general vision system, and specifically the communication and data transfer protocols between such systems. In the proposed modular system, the microcomputer is the master controller for the robot controller and vision controller. It is recommended that the host controller of this proposed modular system be developed and expanded by the following means :

1. Adding Intelligence to Control

   The approach here would be to furnish the expertise, intuition, i.e. intelligence, such that the system is adaptable to new industrial procedures and work loads. Future Automated systems software must have the ability to plan and to handle unpredictable events, faults, or crises. It must learn from each experience. This calls for artificial intelligence (AI) - a major step in the evolution of automated manufacturing control systems. From the viewpoint of this thesis, such software would be imbedded in the Application Specific Software.

2. Distributed Control System

   In a distributed control system architecture, there must be a standard interface to the data base and control signal for sharing the information between elements of the architecture. It would be advantageous to extend the work of this thesis to include other elements, e.q. NC machines, process controllers, etc., within a distributed architecture. Because of the multivendor architecture and differences in the requirements for data manipulation, the distributed control system will be a real challenge to the designer. The modular approach proposed in this thesis is one step toward alleviating this problem.

3. Communication System

Standards for communication are desirable for information to be transferred between control processes. The proposed physical interface between a vision system and a robot controller is an RS 232C interface board which has been accepted as a de factl standard. Nevertheless, the IEEE 488 interface board is readily available and provides another option for a physical interface. According to the approach presented in this thesis, such an adaption can be implemented in the physical interface. It is recommended that the IEEE 488 interface be investigated.

4. Parallel Processing Systems

Parallel Processing Systems are the trend of the future. With the parallel processing technique, a vision system can easily and quickly feedback the information to the control computer. For a real time controlling purpose, the proposed modular system can transfer the data down to the low-level control interface. According to this thesis, the parallel processing technique can be implemented in the vision system. Futher research of parallel processing and its impact on the modular interface of robot and vision systems is needed.

5. More work is needed on developing calibration software for robot and vision systems within the constructs of the modules proposed in this thesis.

6. Extensions to database communications between the host computer for robot and vision system and CAD/CAM systems would provide a valuable future technique. Downloading of CAD product data to the host comtroller would aid in applications specific software i.e. inspection, complex assembly, etc.

BIBLIOGRAPHY

1.] General Electric, "PN 2305 OPTOMATION II VPL LANGUAGE REFERENCE MANUAL", Feb. , 1984.

2.] Takeo Kanade,"Visual Sensing and Interpretation : The Image Understanding Point of View", Department of Computer Science Carnegie-Mellon Univ., CIME April, 1983.

3.] B. Carlisle, S. Gonzalez and D. Mcghie,"The PUMA/VS-100 Robot Vision System", 1st Conf. on Robot Vision and Sensory controls, April 1981.

4.] S. W. Holland, L. Rossol and M. R. Ward, General Motors Research Lab., "Consight - 1 : A Vision-Controlled Robot System for Transferring Parts from Belt Conveyors,"Symposium of Computer Vision and Sensor-Based Robot, G. M. L. Sep. 25 and 26, 1978.

5.] M. L. Baird, "SIGHT-1 : A Computer Vision System for Automated IC Chip Manufacture", IEEE Transactions on Systems, Man and Cybernetics, Vol. 8, February 1978, pp. 133-139.

6.] M. Yachida and S. Tsuji, "Industrial Computer Vision in Japan", IEEE Computer, May 1980, pp.50-64.

7.] B. K. P. Horn, "A Problem in Computer Vision : Orienting Silicon Integrated Circuit Chips for Lead Bonding", Computer Graphics and Image Processing, Vol. 4, No. 3, Sep. 1975., pp. 294-303.

8.] G. J. Agin,"Computer Vision System for Industrial Inspection and Assembly", IEEE Computer, May 1980, pp. 11-20.

9.] W. A. Perkins," A Model-Based Vision System for Industrial Parts", IEEE Transactions on Computers, Vol. 27, 1978, pp. 126-143.

10.] James Rehg," Introduction to Robotics : A Systems Approach", Director of Robotics Resource Center, Piedmont Technical College South Carolina, Prentice-Hall, Inc. 1985.

11.] Mikell P. Groover, M. Weiss, Roger N. Nagel and Nicholas G. Odrey," Industrial Robotics : Technology, Programming, and Application", McGraw-Hill, Inc. 1985.

12.] "Unimate Puma Robot 550/560 Series - Equipment manual", Unimation Inc. 398H1A, May 1982.

13.] G. I. Robertson,"Vision System Separate Gathering From Processing", Control Automation Inc., Princeton, Sensor Review, Vol. 2 No. 2 April 1982.

14.] "i-bot 1" manual, Object Recognition System,Inc. May 1984.

15.] J. C. Schwab and H. Podlecki, "General Electric P50 Process Robot Operator's Manual", P50VE080 Feb. 1983.

16. "Personal Computer XT Technical Reference Manual", IBM , April 1983.

17.] Murray Sargent III and Richard L. Shoemaker,"Interfacing Microcomputers to the Real World", Univ. of Arizona, Addison-Wesley Publishing Company, 1981.

18.] Fredrick J. Hill and Gerald R. Peterson," Digital Systems : Hardware Organization and Design", Univ. of Arizona, John Wiley & Sons, 1983.

19.] Arkady G. Makhlin and Glenn E. Tinsd," Westinghouse Grey Scale Vision System for Real-Time Control and Inspection", Westinghouse Electric Corp., 1981.

20.] Riccardo Cassinis," Hierarchical Control of Integrated Manufacturing Systems", Milan Polytechnic Artificial Intelligence Project, 1981.

21.] Mclean C. R., Mitchell M. and Barkmeyer E., "A computer architecture for small-batch manufacturing", IEEE Spectrum 59-64, 1983.

22.] Rafael C. Gonzalez and Raze Safabakhah, "Computer Vision Techniques for Industrial Applications and Robot Control", IEEE Computer, DEC, 1982.

23.] W. Thompson (ed.), " Special Issue on Machine Perception", IEEE Computer, Vol. 13, No. 5, May 1980.

24.] M. J. B. Duff," Languages and Architectures for Image Processing", Academic Press, 1981.

25.] George G. Dodd and Lothar Rossol General Motors Research Lab.," Computer Vision and Sensor-Based Robots", Plenum Press, 1979.

26.] John F. Jaruis, " Research Directions in Industrial Machine Vision : A Workshop Summary", IEEE Computer, DEC. , 1982.

27.] Bruce A. Artwick, " Microcomputer Interfacing", Prentice-Hall, 1981.

28.] David L. Hudson and John E. Trombly Octek Inc. Burlington, Mass.," Developing Industrial Applications for Machine Vision", CIME, April 1983.

29.] Charles A. Rosen and David Nitzan, Stanford Research Institute, "Use of Sensors in Programmable Automation", IEEE Computer, Dec., 1983.

30.] Philippe Villers, President, Automatix Inc., "Present Industrial Use of Vision Sensors for Robot Guidance", Proceedings of the 12th International Symposium on Industrial Robots, Washington, D. C. 1981.

31.] M. L. Fitz Gerald and Anthony J. Barbera, " A Low-level Control Interface for Robot Manipulators ", Robot Standards June 6-7, 1985

32.] Vanderbrug, G. J. ,Ablus, J. S., and Barkmeyer E., " A Vision System for Real Time Control of Robots", National Bureau of Standards, Proceedings of the 9th International Symposium on Industrial Robots, Washington, D. C., SME, March 1979.

33.] William M. Marcy, "Digital Electronics for Microprocessor Applications in Control of Manufacturing Process", AIIE Transactions, 1980.

34.] Peter L. Wolochow, Intel Cor., " Intel's Bitbus Microcontroller Interconnect : A Modern Method of Robot Communication", Robotics Age, June 1984.

35.] R. A. Jarvis, "Application-orientation robotic vision - a review", Department of Computer Science, Australian National University, Robotica Volume 2, pp 3-15. , 1984.

36.] J. F. Jarvis," Visual Inspection Automation", IEEE Computer, May 1980, pp. 32-39.

37.] M. Ejiri, T. Uno, M. Mese and S. Ikeda," A Process for Detecting Defects in Complicated Patterns", Computer Graphics and Image Processing, Vol. 2, 1973, pp. 326-339.

38.] A. Gill," Visual Feedback and Related Problems in Computer Controlled Hand Eye Coordination", Stanford Artificial Intelligence Lab. Memo AIM-178, Oct. 1972.

39.] Gleason, Gerald J. and Agin Gerald J., " A Modular System for sensor-Controlled Manipulation and Inspection", Proceedings of the 9th International Symposium on Industrial Robots, Washington, D. C. SME, March 1979.

ACKNOWLEDGEMENTS

The author would like to thank Professor Nicholas G. Odrey for his encouragement, aid, and many technical discussions which resulted in the completion of this work. Gratitude is also due to Optical Recognition Systems Company for providing an i-bot vision system and technical support. The author would also like to thank undergraduate students Ian C. Mcleod and Stephen D. Cohen for their cooperation in aiding the author to design the physical interface as part of their ECE 151 senior project.

****
* APPENDIX 1 *
****

This appendix includes the lowest level data transfer routines ,i.e. "remote.asm" and "sbyte.asm", which send data between the robot system and vision system.

```
;:ts=8
;
;       Communications with Robot via Remote Communications
;       MANX assembler ( microsoft/intel format ).
;

codeseg segment para public 'code'
        assume  cs:codeseg

CTLPORT         equ     6DH
DATPORT         equ     6CH
TXRDY           equ     1
RXRDY           equ     2
RDELAY          equ     20

                public  ROBOT_I_, ROBOT_IN
ROBOT_I_        proc    near
ROBOT_IN        equ     ROBOT_I_

RRWAIT:
                IN      AL,CTLPORT      ; GET STATUS
                TEST    AL,RXRDY
                JZ      RRWAIT          ; JUMP IF NO CHAR
                                          AVAILABLE
                IN      AL,DATPORT      ; READ PORT DATA
                AND     AX,7FH
                JZ      RRWAIT          ; THROW AWAY NULLS
                RET

                public  TO_ROBO_, TO_ROBOT
TO_ROBO_:
TO_ROBOT:
                PUSH    BP              ; SAVE BASE PAGE
                MOV     BP,SP           ; SET BASE TO STACK
                PUSH    CX
                MOV     CX,4[BP]        ; LOAD PORT ADDRESS
TRWAIT:         IN      AL,CTLPORT      ; GET CONTROL BYTE
                TEST    AL,TXRDY        ; TRANSMITTER READY?
                JZ      TRWAIT          ; Yes, If Non-Zero
                MOV     AX,CX
                OUT     DATPORT,AX      ; SEND BYTE OUT
                MOV     CX,RDELAY
TRDEL:
                LOOP    TRDEL           ; WAIT TILL STABLE
                POP     CX
                POP     BP              ; RESTORE BASE
                RET                     ; EXIT

ROBOT_I_        endp
codeseg         ends
                end
```

```
;:ts=8
;
;       Standard ORS port i/o routines made compatible with
;       MANX assembler ( microsoft/intel format ).
;

codeseg segment para public 'code'
        assume  cs:codeseg


                public  STORE_B_, STORE_BY

STORE_BY        proc    near
STORE_B_:       PUSH    BP              ; SAVE BASE PAGE
                PUSH    BX
                MOV     BP,SP           ; SET BASE TO STACK
                MOV     DX,6[BP]        ; LOAD PORT ADDRESS
                MOV     AX,8[BP]        ; LOAD DATA TO [AX]
                MOV     CX,10[BP]       ; LOAD LOW BYTE OF COUNT
                MOV     BX,12[BP]       ; LOAD HIGH BYTE OF COUNT
                JCXZ    STBY2           ; LOW BYTE == 0 -> SKIP
                                          1ST LOW-BYTE LOOP
STBY1:          OUT     DX,AL           ; SEND BYTE OUT FOR LOW
                                          BYTE OF COUNT
                LOOP    STBY1
STBY2:          OR      BX,BX           ; CHECK FOR NON-ZERO HIGH
                                          BYTE OF COUNT
                JZ      STBY4           ; HIGH BYTE == 0 -> SKIP
                                          HIGH-BYTE LOOP
STBY3:          OUT     DX,AL           ; SEND BYTE OUT 64K TIMES
                                          FOR HIGH BYTE OF COUNT
                LOOP    STBY3           ;
                DEC     BX              ; DECREMENT HIGH BYTE OF COUNT
                JNZ     STBY3           ; HIGH BYTE STILL != 0 ->
                                          64K MORE BYTES OUT
STBY4:          POP     BX              ; RESTORE BASE
                pop     bp
                RET                     ; EXIT


STORE_BY        endp
codeseg         ends
```

****

* APPENDIX 2 *

****

This appendix includes two software routines,i.e. "Robot Mode" and "Robot Comm". Robot Mode is a terminal emulator program. Robot Comm defines the data transfer protocol and keeps a queue of incoming data from the robot.

```
#define QSZ     200
robot_mode()
{
        /* Get Operator Password */
        if (passwd() == FAILURE) return;

        /* check for Alive Robot */
        if (robot_check() == S_FATAL) return;
        onlamp(3);      /* robot */
        erase_screen();
        curs_on();
        robot_comm();
        offlamp(3);     /* robot */
        curs_off();
        cal_move(RINEAR);
}

robot_comm()
{
int tbuff[QSZ+2];
register int *outq;
register int *inq;
int temp;

        inq = outq = tbuff;

        for (;;) {
                /* If Robot Sending a Character, Get it */
#ifdef MANX
                while (inportb(CONTROLPORT) & RCV_READY)
                        *inq++ = inportb(DATAPORT);

                /* If the Robot is Ready to Accept a Char, See */
                /* if there is Something from the Keyboard for */
                /* the robot */
                if (inportb(CONTROLPORT) & XMIT_READY) {
#else
                while (BYTE_IN(CONTROLPORT) & RCV_READY)
                       *inq++ = BYTE_IN(DATAPORT);
                /* If the Robot is Ready to Accept a Char, See */
                /* if there is Something from the Keyboard for */
                /* the robot */
                if (BYTE_IN(CONTROLPORT) & XMIT_READY) {
#endif

                        if (keystruck()) {
                                temp = tty_input();
                                /* If ESCAPE Dequeue and Exit */
                                if (temp == 0x1b)
                                        return;
                                else
#ifdef MANX
                                        outportb(DATAPORT,temp);
                        }
                }
                if    (inq  !=  outq  &&  inportb(TERM_STATUS)   &
                       XMIT_READY) {
                        outportb(TERM_DATA,*outq++);
#else
                                        BYTE_OUT(DATAPORT,temp);
                        }
                }
                if    (inq  !=  outq  &&  BYTE_IN(TERM_STATUS)   &
                       XMIT_READY) {
                        BYTE_OUT(TERM_DATA,*outq++);
#endif
                        if (inq == outq) inq = outq = tbuff;
                        if (inq == tbuff+QSZ) {
                                rm_error("  Robot Buffer Overflow
                                 ");
                                return;
                        }
                }
        }
}
```

****
* APPENDIX 3 *
****

This appendix includes four software routines,i.e. "robot_diagnostic", "top_set_robot", "bot_set_robot", and "robot_acquire". The function of each routine is described in the program comment.

```
/************************************************************/
/*                                                          */
/*           Name:                                          */
/*                   rob_diagnostic                         */
/*           Function:                                      */
/*                   Check to see if we can talk to the     */
/*                   Unimate Controller                     */
/*           Calls:                                         */
/*                   BYTE_IN                                */
/*                   BYTE_OUT                               */
/*           Returns:                                       */
/*                   OK if Talking is allowed               */
/*                   FAILURE if Timeout                     */
/*                                                          */
/************************************************************/

rob_diagnostic()
{
int rd_time_out;
int rd_too_many;

        rd_time_out = 500;
        rd_too_many = 50;
#ifdef MANX
        while     (rd_time_out-- &&     (inportb(CONTROLPORT)     &
                    XMIT_READY) == 0);
        if (rd_time_out == 0) return(FAILURE);
        outportb(DATAPORT,'\r');
        while(rd_too_many--) {
                rd_time_out = 2500;
                while                           (rd_time_out-- &&
                 (inportb(CONTROLPORT)&RCV_READY) == 0);
                if (inportb(DATAPORT) == '.') return(OK);
        }
#else
        while     (rd_time_out-- &&     (BYTE_IN(CONTROLPORT)     &
                   XMIT_READY) == 0);
        if (rd_time_out == 0) return(FAILURE);
        BYTE_OUT(DATAPORT,'\r');
        while(rd_too_many--) {
                rd_time_out = 2500;
                while                           (rd_time_out-- &&
                 (BYTE_IN(CONTROLPORT)&RCV_READY) == 0);
                if (BYTE_IN(DATAPORT) == '.') return(OK);
        }
#endif
        return(FAILURE);
}
```

```
/************************************************************/
/*                                                          */
/*           Name:                                          */
/*                   top_set_robot                          */
/*           Function:                                      */
/*                   Send Pre-Scaled, Relative Coordinates  */
/*                   for Top-of-Bin Z Axis                  */
/*           Calls:                                         */
/*                   talk                                   */
/*                   talk_loc                               */
/*           Returns:                                       */
/*                   nothing                                */
/*                                                          */
/************************************************************/

top_set_robot(x,y,theta)
double x,y,theta;
{
char comd[40];

        talk("DO SETI COUNT = 0");
        talk_loc("PO OBJ");
        sprintf(comd,"%4.2f,%4.2f,,%5.3f",x,y,theta);
        talk_loc(comd);
        talk(" ");
}
```

```
/************************************************************/
/*                                                          */
/*           Name:                                          */
/*                   bot_set_robot                          */
/*           Function:                                      */
/*                   Send Pre-Scaled, Relative Coordinates  */
/*                   for Bottom-of-Bin Z Axis               */
/*           Calls:                                         */
/*                   talk                                   */
/*                   talk_loc                               */
/*           Returns:                                       */
/*                   nothing                                */
/*                                                          */
/************************************************************/

bot_set_robot(x,y,theta)
double x,y,theta;
{
char comd[40];

        talk_loc("PO OBJO");
        sprintf(comd,"%4.2f,%4.2f,,%5.3f",x,y,theta);
        talk_loc(comd);
        talk(" ");
}
```

```
/*************************************************************/
/*                                                           */
/*           Name:                                           */
/*                   Robot_acquire                           */
/*           Function:                                       */
/*                   Tell robot to start acquisition         */
/*                   phase.  Wait for either a dollar        */
/*                   sign (indicating part found, start      */
/*                   parallel processing) or a period        */
/*                   (indicating part not found process      */
/*                   next pick).                             */
/*           Calls:                                          */
/*                   blab                                    */
/*           Returns:                                        */
/*                   OK for part found                       */
/*                   FAILURE for No Part                     */
/*                                                           */
/*************************************************************/

robot_acquire()
{
register int j,i;

        blab("EX ACQUIRE\r");
        for (;;) {
#ifdef MANX
                while ((inportb(CONTROLPORT)&RCV_READY) == 0);
                if ((i = inportb(DATAPORT)) == '.')
#else
                while ((BYTE_IN(CONTROLPORT)&RCV_READY) == 0);
                if ((i = BYTE_IN(DATAPORT)) == '.')
#endif
                        break;
                if (i == '$')
                        return(OK);
        }
        j = 100;
        for (;;) {
                i = 10000;
#ifdef MANX
                while (i-- && (inportb(CONTROLPORT)&RCV_READY) ==
                       0);
                if (inportb(DATAPORT) == ':') break;
#else
                while (i-- && (BYTE_IN(CONTROLPORT)&RCV_READY) ==
                       0);
                if (BYTE_IN(DATAPORT) == ':') break;
#endif
                if (--j == 0) break;
        }
        j = 30;

        for (;;) {
                i = 1000;
#ifdef MANX
                while (i-- && (inportb(CONTROLPORT)&RCV_READY) ==
                       0);
                if (inportb(DATAPORT) == '.') break;
#else
                while (i-- && (BYTE_IN(CONTROLPORT)&RCV_READY) ==
                       0);
                if (BYTE_IN(DATAPORT) == '.') break;
#endif
                if (--j == 0) break;
        }
        return(FAILURE);
}
```

****
* APPENDIX 4 *
****

Detailed description of I/O assignments and functions for GE P-50 robot controller (Taken from P-50 process robot OPERATOR'S MANUAL [15])

Table 1. Input Signal Assignments and Functions

Connector Pin No.	Signal Name	Function	I/O Resister Assignment No.	Circuit (RYIF) Signal Name
BS	Input signal 0	1) If an input signal has been taught, the robot will wait until an external signal appears at the corresponding input contacts before commencing its move. 2) If the OUT switch on the Teach Box (DT) is set to OFF during playback, it is possible to cause the robot, which is still waiting for an input signal, to operate by pressing the START button (J0).	192/000	RI0
BT	" 1		193/001	RI1
BU	" 2		194/002	RI2
BV	" 3		195/003	RI3
BW	" 4		196/004	RI4
BX	" 5		197/005	RI5
BY	" 6		198/006	RI6
BZ	" 7		199/007	RI7
CD	Enter 200	1) These signals are referred to by the job instruction ENTER during job teaching. *Input signals 0 -- 7 may be designated in both program teaching and job teaching. Care should be exercised programming them, however, when assigning external signals.	200	RI10
CE	" 201		201	RI11
CF	" 202		202	RI12
CH	" 203		203	RI13
CJ	" 204		204	RI14
CK	" 205		205	RI15
CL	" 206		206	RI16
CM	" 207		207	RI17
CN	Job select 2^0	1) These are job select external signals (JOB1 - JOB15). 2) Job teaching of the CALL instruction is made. When it is executed, operation of the selected (input) Job No. is performed.	208	RI20
CP	" 2^1		209	RI21
CT	" 2^2		210	RI22
CU	" 2^3		211	RI23
CW	Zero-point adjustment signal (option)	1) This signal is valid only when MODE X is designated. 2) This signal also functions as servo ON external command signal. Zero-point adjustment operation starts when this signal (pulse signal of 200ms or longer) is deactivated after activation, and playback is immediately permitted when zero-point adjustment is completed.	212	RI24

Table 1. Input Signal Assignments and Functions (Cont.)

Connector Pin No.	Signal Name	Function	I/O Resister Assignment No.	Circuit (RY1F) Signal Name
A/H	Output 0 signal	1) If these signals are taught, a signal that corresponds to any one of them is is produced on completion of positioning during playback operation. An external unit is enabled. 2) If playback operation is performed with the OUT ON switch (on the Operation Panel) set in the OFF position, operation of the robot may be checked without producing taught output signals (without causing external units to operate). This function is invalid when job teaching is done.	16/8	Ro0 A/B
B/J	" 1		17/9	Ro1 A/B
C/K	" 2		18/10	Ro2 A/B
D/M	" 3		19/11	Ro3 A/B
E/N	" 4		20/12	Ro4 A/B
F/P	" 5		21/13	Ro5 A/B
R/X	" 6		22/14	Ro6 A/B
S/Y	" 7		23/15	Ro7 A/B
V/AC	Out 24	1) These signals may be used as exterior output signals using the OUT button. If job teaching specifies OUT 24, for instance, an output signal occurs between connector pins V and AC during playback operation. *Job teaching of output signals 0-7 may also be done. If job teaching specifies OUT 16, for instance, an output signal occurs between pins A and H . Therefore, care should be exercised in assigning these signals.	24	Ro10 A/B
W/AD	" 25		25	Ro11 A/B
AE/AM	" 26		26	Ro12 A/B
AF/AN	" 27		27	Ro13 A/B
AK/AT	" 28		28	Ro14 A/B
AL/AU	" 29		29	Ro15 A/B
AV/AZ	" 30		30	Ro16 A/B
AW/BA	" 31		31	Ro17 A/B
AX/BB	In automatic operation	1) This signal is produced while the start indicator lamp on the operation panel is lit.	240	Ro20 A/B
AY/BC	Abnormal welding	1) When a fault signal (arc breakage, torch contact, etc.) is input from the welder, the robot stops in servo ON state. Restart may be made when the cause for the fault is eliminated.	241	Ro21 A/B
BF/BH	Emergency Stop	1) This signal indicates that the robot is in an Emergency Stop state. The contact of this signal is normally closed. 2) Restart cannot be made even when the cause for emergency stop is eliminated.	---	Ro24 A/B

Table 2. Output Signal Assignments and Functions

OUTPUT	INTERNAL NAME	PROGRAM ASSIGNMENT #	JOB USE	CONNECTOR PIN NUMBER
0	Ro0 A/B	0	OUT 16	A/H
1	Ro1 A/B	1	OUT 17	B/J
2	Ro2 A/B	2	OUT 18	C/K
3	Ro3 A/B	3	OUT 19	D/M
4	Ro4 A/B	4	OUT 20	E/N
5	Ro5 A/B	5	OUT 21	F/P
6	Ro6 A/B	6	OUT 22	R/X
7	Ro7 A/B	7	OUT 23	S/Y
8	Ro10 A/B	X	OUT 24	V/AC
9	Ro11 A/B	X	OUT 25	W/AD
10	Ro12 A/B	X	OUT 26	AE/AM
11	Ro13 A/B	X	OUT 27	AF/AN
12	Ro14 A/B	X	OUT 28	AK/AT
13	Ro15 A/B	X	OUT 29	AL/AU
14	Ro16 A/B	X	OUT 30	AV/AZ
15	Ro17 A/B	X	OUT 31	AW/BA

Table 3. I/O Interface Connections

P-50 USER INPUTS

Connector Pin No.	Signal Name	Function	I/O Resister Assignment No.	Circuit (RYIF) Signal Name
CW (cont.)	Zero-point adjustment signal (option)	3) Exercise care incorporating this signal as the zero-point adjustment operation is immediately discontinued when this signal is activated. If zero-point adjustment operation has already been completed, this signal is ignored and no operation is performed.	212	R124
CX	Servo OFF signal	1) This is an external servo OFF command signal. 2) Regardless of the mode, this signal is accepted and the motor main circuit is cut off.	213	R125
CY	External start signal	1) Automatic start signal (external) for the robot which is valid only when Mode X is designated.	214	R126
CZ	External stop signal	1) This is a temporary (external) stop signal to the robot during playback operation. 2) The robot stops in servo ON state. Restart may be made.	215	R127
CV	External (user) Emergency Stop signal	1) Emergency Stop command signal (normally closed contact) from user-supplied external safety interlocks or Emergency Stop buttons. 2) Terminals CV and DB of the attached connector are shorted prior to shipment. Remove the wire used for the shorting, and connect and Emergency Stop switch between these two terminals.	---	EMG3

Table 3. I/O Interface Connections (Cont.)

P-50 USER OUTPUTS

INPUT	INTERNAL NAME	PROGRAM ASSIGNMENT #	JOB USE	CONNECTOR PIN NUMBER
0	Ri0	0	ENTER 0 (or 192)	BS
1	Ri1	1	ENTER 1 (or 193)	BT
2	Ri2	2	ENTER 2 (or 194)	BU
3	Ri3	3	ENTER 3 (or 195)	BV
4	Ri4	4	ENTER 4 (or 196)	BW
5	Ri5	5	ENTER 5 (or 197)	BX
6	Ri6	6	ENTER 6 (or 198)	BY
7	Ri7	7	ENTER 7 (or 199)	BZ
8	Ri10	X	ENTER 200	CD
9	Ri11	X	ENTER 201	CE
10	Ri12	X	ENTER 202	CF
11	Ri13	X	ENTER 203	CH
12	Ri14	X	ENTER 204	CJ
13	Ri15	X	ENTER 205	CK
14	Ri16	X	ENTER 206	CL
15	Ri17	X	ENTER 207	CM

VITA

Pei-Chann Chang was born to Mr. and Mrs. Sen-Hu Chang on September 9, 1957, in Taiwan, R.O.C. He received his secondary education at The Kaohsiung High School, Taiwan. In the fall of 1975, he entered the National Tsing-Hua University and was awarded a Bachelor of Science degree in Industrial Engineering in june, 1979. After fulfilling his military service in the Chinese Marine Corps. as a second lieutenant, he worked in the Tang-Eng Stainless Steel Company as an assistant engineer in 1981. In fall, 1983, he came to the United States to pursue his graduate study in Lehigh University.