A Universal Six-Joint Robot Controller

D.G. Bihn
Hewlett-Packard
Cupertino, CA 95104

T.C. Hsia
University of California, Davis
Davis, CA 95616

1. Abstract

A general purpose six-axis robotic manipulator controller was designed and implemented to serve as a research tool for the investigation of the practical and theoretical aspects of various control strategies in robotics. A 80286-based Intel System 310 running the Xenix operating servo software as well as the higher level software (e.g. kinematics and path planning). A Multibus compatible interface board was designed and constructed to handle I/O signals from the robot manipulator's joint motors.

From the design point of view, the universal controller is capable of driving robot manipulators equipped with D.C. joint motors and position optical encoders. To test its functionality, the controller is connected to the joint motor D.C. power amplifier of a PUMA 560 arm bypassing completely the manufacturer-supplied Unimation controller. A controller algorithm consisting of local PD control laws was written and installed into the Xenix operating system. Additional software drivers were implemented to allow application programs access to the interface board. All software was written in the C language.

2. Introduction

Robots are becoming increasingly prevalent in the industrial workplace, as well as creating an industry of their own. This new industry is both driving and being driven by new technologies. New materials, improved mechanical designs, and faster controller electronics are running into the limitations of traditional control techniques. Thus, theoretical work to overcome these limitations is urgently needed. Much of the theoretical work is being carried out in academic research institutions. However, there is often a significant gap between the results of theoretical studies based on simulations and the verification based on actual implementation. Industry is often reticent to try untested theoretical results, preferring the time-tested, sub-optimal control techniques of the past, possibly sacrificing substantial performance improvements. A credible testing ground for new control techniques is needed to bridge the gap between theory and application.

The Robotics Research Laboratory at the University of California, Davis, has a Unimation PUMA 560 robot arm representative of a large and popular class of modern industrial manipulators. The PUMA arm is controlled through the sophisticated robot language, VAL-II. The user only has access to the arm through high level 'move-type' commands. He therefore has little control of the actual arm trajectory and no control over the low level motor servo loops. In typical industrial applications, the inability to alter low level functions of the controller does not represent a functional limitation. To the contrary, it actually affords both the arm and the operator a fair degree of protection and safety. The academic researcher, however, is prevented from using the arm to test and demonstrate new control strategies and is forced to rely on computer simulation.

3. Objective

The objective of this project was to design and implement a computer based robotic controller which allows the researcher to write programs and implement algorithms which control the robot arm from the lowest level of the closed-loop servo system to the higher levels of kinematics, dynamics, path planning and robot language [11]. The use of a familiar software environment was chosen with the intent of making the user interface as clean and simple as possible.

The scope of this project is limited to the design and implementation of a controller consisting of (1) the Joint Interface Board electronics, and (2) the operating system interface to this hardware. A simple low level 6-joint P.I.D. (Proportional Integral Derivative) controller is implemented and presented to serve as both a functional test of the system and as an application example. The topics of joint kinematics and other high level application software are beyond the scope of this project as is the advanced control law design.

4. The Controller System

The controller presented here is designed around an Intel 310, 80286-based, microcomputer [2] running the UNIX-like operating system, XENIX [3]. A signal interface board was designed and constructed to provide the interface between the microcomputer and the joint motors of the arm. The Unimation controller, supplied as part of the PUMA 560, was modified to serve two low level functions: as a convenient access point for the joint feedback signals from the arm and as a multi-channel power amplifier drive the joint motors. All other electronics in the Unimation multi-channel power amplifier to drive the joint motors. All other electronics in the Unimation controller are by-passed; closed-loop control is done in the Intel-based controller described here. The controller system is depicted by the block diagram shown in Figure 1.

A single 80286 CPU running at 6 MHz is used to execute both high level (e.g. kinematics) and low level (e.g. joint servo loops) control software. At a typical sampling rate of 100 Hz, about 30% of the CPU time is required to execute the six P.I.D. controllers implemented in the design example. The remaining CPU time is available for application programs and the operating system. The interface board itself is useful in systems with sampling rates over 2 KHz. However, to utilize this speed, additional CPU power is required.

5. System Design Requirements

Two basic elements constitute the controller system designed and implemented in this project: a digital computer and special purpose interface hardware. The digital computer performs all the control functions, from the joint motor servo control law to the higher levels of coordinated joint motion. The interface hardware function is to provide the basic link between the computer and the physical signals required to control the robot arm.

5.1 The D.C. Servo Motor Position Measurement

The control of the robot arm is equivalant to the control of the joint motors. In this controller, D.C. servo motors are assumed to be equipped with potentiometer and/or incremental encoder position feedback devices. It is also assumed that the D.C. motor can be driven by an analog (voltage) signal buffered by an appropriate external power amplifier (servo motor amplifier). The Unimation PUMA 560 arm has six geared D.C. servo motors with both encoder and potentiomenter position feedback elements and it is considered to be prototypical of the class of manipulators considered in this project.

Each motor, in general, does not directly drive a manipulator joint, but is typically connected through a gear train requiring a multiple number of motor revolutions to drive the joint through its operating range as shown in Figure 2. In the configuration assumed in this project, feedback elements are directly attached to the motor, not the actual joint member. Joint position is inferred from the motor position. This requires that absolute motor position must be measured over multiple revolutions. In the PUMA arm, both a geared (i.e. multi-turn) potentiometer and an incremental shaft encoder are connected to the motor shaft to collectively supply this data. The incremental encoder is used to accurately measure both the relative motor position over an arbitrary number of rotations and the absolute motor position modulo one rotation. The geared potentiometer is used to measure the approximate absolute motor angle over the several revolutions needed to drive the joint through its range. Once the absolute motor angle has been determined, only the relative data supplied by the encoder is needed.

The incremental encoder, which is directly attached to the motor, generates two types of data: (1) high resolution quadrature signals which are decoded into relative (incremental) angular displacement information and (2) an index pulse which is produced once per revolution and can be used to accurately define the absolute angular position of the motor modulo 360° (Figure 2).

The geared potentiometer supplies indirect, low resolution absolute joint position data. The gear ratio of the potentiometer is designed so that when the joint is driven between its mechanical limits, the pot wiper rotates within its mechanical limits (less than 360°). Logically, this pot could have been attached directly to the robot joint. For manufacturability considerations, the pot has been included in the motor assembly.

Once the absolute motor position has been determined, it is continuously updated (incremented or decremented) by the data from the incremental shaft encoders. As long as the electronics are not interrupted (e.g. power-down) the data from both the geared pot and the encoder's index pulse are not used. The difficulty is the initial determination of the absolute motor position is rather involved and will now be discussed.

When the absolute motor position is unknown, the potentiometer wiper voltage can be measured and the absolute motor position estimated. Once estimated, further position measurement can be made by monitoring the relative position data from the incremental encoders. While the incremental data is very accurate, the absolute position can only be as good as the initial estimate. The standard technique to obtain an accurate measurement of the initial absolute position is as follows. First, the motor is driven until the encoder's index pulse is found. At this point the absolute position is known to be an exact multiple of 360°. Next, the potentiometer voltage is measured to give the approximate absolute position. Combining the approximate absolute position with the certain knowledge that the position is an exact multiple of 360°, the exact absolute can then be derived.

The above explanation serves to demonstrate the basic idea and what sort of precision is required. For analysis, the actual parameters of the PUMA 560 joint motors are used to determine the system specifications and Joint Interface Board requirements.

5.2 A Typical D.C. Servo Motor

The PUMA 560 servo motors are integral packages which contain four basic components: (1) a D.C. motor; (2) an electric brake; (3) an optical incremental encoder; and (4) a geared-down potentiometer. The currents activating the motor and the electric brake are the inputs while the encoder and the potentiometer signals are the outputs. The basic functions needed to operate the motor system are described below.

5.2.1 Reading the Incremental Encoder

The incremental encoder has three output signals: channels A, B, and the index pulse. Channels A and B are used to determine both the amount and direction of rotation in discrete steps. The index pulse produces a single short pulse each motor revolution which can be used by the system to determine the absolute angle of the motor and, with the addition of the potentiometer data, can be used to determine absolute position (described above).

The output states of channels A and B are used to detect relative motion (rotation) of the motor shaft and in turn, the joint itself. How this is done is well-known and is not described here.

5.2.2 Counting the State Changes

The incremental encoders on the PUMA 560's motors produces 1000 state transitions per revolution, except for the shoulder joint (#2) which produces 800 transitions. The motor (with the encoder directly attached) rotates from 40 to 60 times during full joint travel (depending on the joint), corresponding to 40,000 to 60,000 state changes per complete joint motion. It is convenient if the hardware keeps count of the total joint range. This way the total joint motion may be read directly from the hardware counters. 16-bit counters have a maximum count of 65,532 and are sufficient to keep track of the joint motors of the PUMA arm. However, it is not essential for the hardware to count the entire joint range. In a sampled data system, the software can keep track of total joint motion, using the hardware only to count the relative motion which has occurred between samples. If the hardware count is used in this manner, the absolute motion is limited only by software and the incremental motion between samples is limited to motion of ± 32K pulses.

5.2.3 Reading the Potentiometers

The potentiometers incorporated into the PUMA 560 joint motors are connected between +5 volts and ground. Rotating the pot through 360° produces a proportional voltage output from 0 to 5 volts (e.g., 90° produces 1.25 volts, 180° produces 2.5 volts, etc.). These pots have been geared so that they rotate somewhat less than 360° for a complete joint movement; depending on the joint, full joint travel may produce as little as 200° of potentiometer motion. This restricted travel corresponds to a change in pot voltage of about 2.78 volts. If the joint produces 60 index pulses (i.e. 60 motor rotations) per full joint motion, the pot voltage must be measured to an absolute accuracy of $1/60^{th}$ of 2.78 volts (0.046 volts) in order to determine the motor shaft angle to within one revolution.

An Analog to Digital Converter (ADC) is used to measure the pot voltages. It must be able to measure a voltage which spans a 0 to 5 volt range, and must have a resolution and accuracy of better than 0.046 volts over this range. This corresponds to a full-scale resolution of 0.92%. A seven-bit ADC has a resolution of 0.78% and is sufficient for this voltage measurement.

Since the potentiometers are not part of the dynamic control scheme presented here, there is no constraint on the conversion speed. For the PUMA arm, both speed and resolution requirements of the ADC are easy to meet. However, to make the system more flexible, other possible applications should be considered. It is often the case where a symmetrical voltage signal (say -5 to +5 volts) needs to be measured and fast conversion time can make dynamic control systems with analog feedback elements possible. Furthermore, since fast (30 microsecond) 12-bit ADCs with input range of ±5 volts are conveniently available and at reasonable cost, this higher performance device was chosen.

5.2.4 Driving the Motor

The drive current and voltage needed by a D.C. motor depends on the size and type of motor used; no solution is appropriate for all motors. It is therefore considered impractical to include the power amplifier as part of the design. The important requirement is how to drive these power amplifiers.

In general, two standard techniques for supplying the current needed for driving D.C. servo motors are commonly used: linear amplifiers and pulse width modulated (PWM) amplifiers. Each have advantages but the important fact to consider is that they both are controlled by a simple analog voltage.

In the particular case of the PUMA 560 arm, the Unimation PUMA controller's power amplifiers can be conveniently used because they have been designed explicitly to drive the PUMA 560 joint motors. Using this controller also makes the external connection to the arm joint motors simple and straightforward. Additionally, the Unimation amplifier has several useful safeguards which automatically shut the amplifier off to prevent damage to the arm.

Power amplifiers are controlled by analog voltages, and to generate these voltage outputs from a digital controller a Digital to Analog Converter (DAC) must be used. Three basic specifications must be considered: (1) voltage swing; (2) resolution (number of bits); and (3) the accuracy. Commercially available power amplifiers typically require a voltage input of -10 volts to +10 volts. This also corresponds to typical DAC device output characteristics, and the input specifications of the Unimation controller's power amplifier.

Selection of resolution and accuracy is more difficult. 8-bit corresponds to a resolution of one part in 256 (0.39%), 10-bit corresponds to one part in 1024 (0.098%), and 12-bit corresponds to one part in 4096 (0.024%). A 10-bit unit was chosen and considered to be a reasonable compromise between price and performance.

5.2.5 Releasing the Brake

The brake is used to lock each joint in position when the servo motor is turned off and is necessary to keep the arm from collapsing. The brake is much like a D.C. relay. When D.C. current passes through the coil (an electromagnet), the brake plate is retracted from the friction plate allowing the motor to rotate. When no current is flowing in the brake coil, the brake plate is forced into contact with the friction plate by compression springs and the motor cannot rotate. On the Unimate controller, the brake release current is supplied whenever the 'arm power' is on. This is a fail-safe system. When the servo power (to the motors) is turned on, the brakes are released. When the arm power is switched off, the brakes are automatically applied, holding the joint in place.

5.2.6 Other I/O Requirements

The Joint Interface Board must not only accommodate the joint motor signals but must also provide the host computer with additional functions to allow all subsystems to be integrated into a complete controller. Included in the design are (1) a digital timer and associated interrupt circuitry, and (2) 24 bits of general purpose I/O lines.

5.3 Host Computer Requirements

The selection of a suitable host computer is very important. The machine must not only be capable of meeting basic execution speed and I/O requirements, but should also be able to support the software tools needed to implement a controller. In this section both the host computer hardware and software are discussed.

5.3.1 The Computer: Intel System 310

The Intel System 310 microcomputer was used because it satisfies the above criteria. It is based on the Intel 80286 16-bit microprocessor [4]; the system also comes equipped with an 80287 floating point math co-processor [5]. It is a Multibus based system [6], a bus standard which is particularly popular in the area of industrial automation. A wide variety of interface board products, including memory, I/O, and blank prototype boards, are available from Intel and third party vendors. A standard Multibus board is comparatively large which allows complex circuits to fit onto a single board, allowing the use of a single bus interface circuit. All the hardware for the Joint Interface Board was able to fit on a single board.

5.3.2 The Operating System Choice: XENIX 286

The Intel 310 can run several operating systems: the ubiquitous IBM PC's MS-DOS, the UNIX-like XENIX system O.S., and the real-time, multi-tasking systems RMX-86 and RMX-286.

XENIX was chosen to be the operating system of this project's implementation. A substantial learning effort is required to become proficient with an unfamiliar operating system and new software tools. XENIX minimizes this obstacle; many researchers are familiar with UNIX and need little time to master XENIX. Those unacquainted with UNIX can be motivated to learn XENIX since this knowledge is useful on many other systems. This is a very important consideration on short term projects where learning a new operating system may require more time than the experiment itself.

The XENIX operating system is Microsoft's licensed version of UNIX III with some of the Berkeley Software Distribution (BSD) enhancements (e.g. 'vi' and the C-shell), and several of their own enhancements. It is a multi-user system. UNIX is a very powerful environment for developing software and is widely used in the academic and research communities. The disadvantage is that it was not designed for real-time applications. Details of the techniques used to construct a real-time controller for our purpose are given later.

6. The Design

This section details the design and implementation of the above specifications. The discussion is divided into three sections: (1) the hardware design of the Joint Interface Board (JIB); (2) the connection between the J.I.B. and the Unimate PUMA 560 controller; and (3) the software interface between the XENIX operating system and the JIB.

6.1 Joint Interface Board Design

The block diagram which outlines the J.I.B. hardware is shown in Figure 3. As seen from the computer side of the bus interface, the JIB is a small collection of I/O devices: six 16-bit encoder counters; an encoder reset circuit; two PIO (parallel input/output) devices; timer and the interrupts reset logic. One of the PIOs is used exclusively to interface to the ADC and DAC subsystem, and the other PIO is used for off-board digital expansion.

6.1.1 The Analog-Digital Subsystem

Communication and control signals for all seven DACs, the ADC and the analog multiplexer are [illegible] to one of the PIOs [illegible] PIO) is in the Multibus address space and control of these devices must be made through the [illegible] primary reason for designing the system this way was bus speed considerations. The PIO, an 8255, can operate at the full 80286 bus speed while the ADCs and DACs are about twice as slow (450 ns vs 180 ns). Rather than slow the bus down on this board and degrade performance of the other onboard devices (e.g. the encoders), the ADCs and DACs are given their own private slow bus.

6.1.2 Analog Output: The DACs

Six analog voltage outputs are necessary to drive the basic joint servo motors. An additional analog voltage output was included to permit future expansion, possibly the control of a more sophisticated gripper. To produce these outputs, seven independent DACs (digital to analog converters) were used. The independent DAC approach offers the advantage of a very straightforward interface, improved accuracy and simpler circuit design.

As described in the Analysis section above, the analog outputs must be capable of delivering a voltage from -10 volts to +10 volts at a resolution of 10 bits (1 part in 1024) to properly drive the inputs of the servo motor power amplifier.

6.1.3 Analog Input: The ADCs

As described in the analysis section, each of the PUMA 560 joint motors has a potentiometer which produces an output from 0 to 5 volts and, to be useful in absolute position determination, these signals must be resolved to an 8-bit accuracy. Fast, high resolution analog to digital converters can be obtained at reasonable prices which exceed the basic specification but give the Joint Interface Board more power. Analog Devices' AD574 [7] is a popular example. It has a 12-bit resolution, a conversion speed of less than 30 micro-seconds, selectable input ranges of 0 to +10, 0 to +20, -5 to +5, and -10 to +10 volts, and a cost of less than $35. At this speed of conversion, one device is fast enough to convert all six joints' pot data in less than 0.2 milliseconds, a speed fast enough to allow the pots alone to be used as the primary feedback element in situations where it may be useful.

To use one ADC to convert several analog input signals requires the use of an analog selector or multiplexer. A typical analog multiplexer, the LF1308 has eight voltage inputs which are selected to one output. This output can then be converted by the ADC, one at a time. Like the DAC outputs, ADC outputs must also be latched. However, since the ADC output is digital, it may be easily stored inside the computer using software without using any special hardware.

6.1.4 Timer Subsystem

Generating a constant sampling interval requires an external clock source to interrupt the CPU and cause the control software to execute. The Multibus provides a 10 MHz clock requiring an onboard frequency divider logic. To allow convenient changing of the sampling rate, a divide-by-ten prescaler followed by a microprocessor compatible programmable timer was selected. An Intel I8254 triple 16-bit timer I.C. [8] was used. It features extensive programmability, high resolution (one part in 65K). In the divide-by-n mode it can be programmed to generate a square wave with a period from 2 microseconds to 65 milliseconds in 1-microsecond steps. This corresponds to a rate from 22 Hz to 500 KHz (though rates above 200 Hz are not usable in the present system). Timer #0 is used as the interrupt clock, leaving timers #1 and #2 available for future applications.

6.1.5 Digital I/O

To make the Joint Interface Board a more flexible and general purpose interface, an additional parallel input/output (PIO) I.C. was included in the design. All the 24 outputs from this device go directly off-board via the connector J12 and are not used by any of the onboard electronics.

6.1.6 The Encoder Subsystem

The JIB accepts six sets of incremental encoder signals. Each input set is used to control its own 16-bit counter, instructing it to count UP, count DOWN, do nothing, or RESET to zero. The encoder subsystem can be divided into three parts: (1) the basic up-down counter; (2) the decode logic; and (3) the reset logic.

A. The Counters

The 16-bit up-down counter is a straightforward cascading of four 4-bit up-down synchronous counter with three control inputs: clock enable (CE), up-down select (UD), and reset (R). The system clock is running continuously at 1 MKz.

To directly implement a state decoder, six decoders would have to be constructed. This would probably require six 16-pin DIP packages. These would probably have to be either bipolar PROM (programmable read only memory) or some type of PLA (programmable logic array). If the PROM approach is used, a 16 x 2 = 32-bit PROM would be sufficient. The total number of bits required by all six units in this scheme is 196 bits.

This new 12-bit vector has 4096 possible states, each of which must be decoded to generate a 6-bit output vector, O, with the proper CE and UD signals for three counters.

For six counters, a total of 4096 x 6 x 2 = 48 K-bits is required. This is two orders of magnitude greater than the scheme where each counter has its own state decoder. The advantage of this bit wasteful approach is that all this decoding can be done using just two 8 K-byte EPROMs packaged in 28-pin DIPs. These memory I.C.s are inexpensive and EPROM programmers are typically found in microprocessor development laboratories.

B. The Encoder Reset System

An index pulse signal is generated every incremental encoder (servo motor) rotation. This signal is used to supply quasi absolute position information about the motor so that the motor revolutions (e.g. 0°, 360°, 720°, etc.) can be distinguished from one another. Typically these index signals are only used during initialization of the hardware and software after system power up. Once the system has been initialized, incremental information alone is sufficient to determine absolute position (provided no encoder state changes are missed).

The basic scheme of the reset/calibrate routine is to rotate each motor until the index pulse is found and then this position is defined to be the position zero. Conceivably, this could be done in software by continuously reading the index signal until it is detected. This would require the software to sample the signal fast enough so that the pulse is not missed when the motor is moving at some speed.

To overcome this limitation, a hardware scheme was devised which allows calibration of the system with the motor to be running at any speed within its operating limits. Each counter has a synchronous reset input. The index signal from the encoder could be connected to this input causing the particular counter to reset to zero whenever the index pulse occurs. However, since the motor typically rotates tens of times during the joint travel, some form of selectively gating the index signal on and off was required.

This circuit is asynchronously set or 'armed' via the ARM RESET signal. Once armed, the next occurrence of the index pulse generates a single reset pulse for the associated counter circuit. Once the reset pulse is issued, the circuit disarms itself so that further occurrences of the index pulse will not reset the counters. The software can monitor these signals to check if the reset circuit is armed or not and can thereby determine if the index pulse has occurred.

6.1.7 The Multi-Bus Interface

Up to this point, all the subsystems described here have been computer independent (except for the general requirement of a 16-bit bus). This allows easy conversion to many other 16-bit computers such as the IBM-AT. At this point the design becomes specific to the hardware of the host machine. The Intel 310 system is based on the Multibus. The Multibus supports direct addressability up to one megabyte through a 20-bit address and 8- and 16-bit data transfers at a rate of five million transfers per second (10 MB/sec). The Joint Interface Board has been designed as a simple slave and never controls the Multibus. The JIB only decodes the address lines and acts upon the command signals from the bus master.

6.2 The Unimation Interface

The following sections describe how the Joint Interface Board and the XENIX software interface running on the Intel 310 were connected to the Unimate PUMA 560 arm. Position feedback signals from the arm servo motors are sent to the JIB, and the JIB sends analog voltage outputs to the power amplifier, which in turn drives the servo motor in each joint.

The Unimate PUMA controller consists of an LSI-11/73, six 6503-based joint controller boards, several low level interface boards, and a six channel-high current power amplifier. The controller presented in this project makes use of only the power amplifiers and one of the feedback signal conditioning circuits. The LSI-11 and the six microprocessor joint controllers are completely bypassed.

To close the loop around the joint motors, the feedback signals from the PUMA 560 have to be connected to the Intel system and the output command voltages must be returned from the Intel to the power amplifiers in the Unimate controller. It was considered desirable to make the necessary modification to the Unimate controller in such a way that switching between the Intel controller and the internal Unimate controller systems is as simple and safe as possible.

Connecting the feedback signals from the Unimate Controller to the Joint Interface Board is accomplished by inserting a proto-typing card (from here on called the Unimate Interface Board) into one of the several available empty, unwired slots of the joint controller portion of the Unimate card cage [9]. This technique was selected for several reasons. All of the PUMA arm feedback signals enter the controller through connector J-30 and are hard-wired directly to the ARM CABLE CARD in the card cage. Here some basic signal conditioning is performed, power is supplied to the joint pots and encoders, and the encoder outputs are then buffered to produce clean logic levels. Since these functions are required and would have to be duplicated if this subsystem was not used, it was convenient to use the external hardware and obtain these signals after conditioning.

The only place these feedback signals are found is on the backplane of the PUMA joint controller's card-cage. One of the available slots was chosen and all the necessary connections were made only by adding wires to the backplane, bringing all the feedback signals to the selected slot. This has the attractive feature of not having to break or cut any Unimate connections, leaving the controller intact. When the Unimation Interface Card is removed from its slot, the system is electrically and logically in its original condition. The card which is inserted into this slot also contains an inverting line driver to buffer the encoder signal to drive the wires connecting it to the Intel/JIB system.

While the feedback signals can be sensed without breaking any connections in the Unimate controller, this is not possible for the motor current command signals. In general, either the Unimate or the Intel/JIB joint motor current command signals (the DAC outputs) can be used, since only one controller can be selected to drive any motor at any one time. Current command signals enter the power amplifier through connectors P73 and P74. These connectors are located on the top of the POWER AMP CONTROL card and are readily detached. This is the only point where these signals can be 'intercepted' and the Intel signals injected without permanently modifying the circuit (e.g. cutting wires). A small interface panel with the appropriate connectors was fabricated to allow the JIB and Unimate motor current command signals to be selectively sent to the Unimate's power amplifier on a joint-by-joint basis (which is very useful during system debugging).

6.3 The XENIX to the Joint Controller Board Interface

The Joint Interface Board is installed in the I/O space of the Intel 310 (distinct from the memory space) and like all other system hardware in XENIX, the user can only access it through system device drivers. Drivers for all the JIB devices have been written and installed into XENIX (see software listing in Appendix H [1]). Application programs access these devices through symbolic names (e.g. "dac_1", "adc_4", "timer_1", etc.). The device driver controls the details of the data format and of physically addressing the hardware transparent to the application program [10].

Properly written drivers protect the system from the application programs and make the user interface clean and simple. A motor controller can be implemented entirely at the application level, individually accessing the incremental encoders, DACs and the ADC through their respective device drivers. While this will work, much of the CPU time is consumed in operating system overhead. Each I/O request (e.g. read and write) takes substantially longer to execute than if the software is able to directly address the hardware (not permitted in XENIX). An alternative to implementing the controller at the application level is to place it in the XENIX kernel as a single logical device (e.g. rather than "dac_1" and "encd_1" devices, a single "pid_1" can be considered as the basic I/O unit). Code written at the kernel level has direct access to the I/O space and may read and write to the JIB without going through the operating system. This reduction of overhead can reduce execution time by about 50%.

6.4 Real-Time Issues

XENIX is not a real-time operating system; it does not guarantee when a particular application program will get executed. It is often said that XENIX (vis-a-vis UNIX) does not guarantee when an interrupt is serviced. This only refers to the application level, not the lowest level of interrupt handling. In the common application of a terminal handler, an interrupt is issued from the serial interface hardware (a UART) each time a new character is received from the terminal. The interrupt handling software then services the hardware, taking the new character and putting it into the terminal handler's buffer. This software only competes with other interrupt routines (e.g. other terminals) for CPU time. Non-interrupt level operating system software which processes the characters in the terminal handler's buffer must compete with the entire system (including other application programs) for CPU time and it is here where XENIX cannot guarantee response time. This issue is important in designing a real-time controller.

A XENIX based real-time controller may be constructed in two fundamentally different ways. Both methods require that an external timer interrupt the CPU at fixed intervals and that a kernel level device driver be installed in the XENIX system to process this interrupt. In the first method, the interrupt handler of the driver responds to the timer interrupt by only setting a flag in the driver's memory. When the 'device' is read by the application software, the read part of the driver tests this memory flag. If the flag is set, it returns back to the application program. If the flag has not been set (i.e. the timer has not yet interrupted the CPU), the read routine keeps testing the flag until it is set by the timer interrupt. This technique allows application programs to synchronize themselves to the external clock and produce a constant sampling rate for a digital controller written at this level. However, XENIX does not guarantee when the application program will be allowed to execute, and this may lead to occasional missed sampling intervals. As long as XENIX is run in the single user mode and the timer interrupt is not faster than 100 Hz, a useful system can be implemented.

In the second method, the one used in this project, the entire control system software is installed at the kernel level of XENIX and is executed as part of the interrupt service routine of the driver itself. Since the interrupt service routine does not have to compete with the non-interrupt portion of XENIX (including all application programs), this technique is guaranteed to be executed on each timer interrupt, producing a reliable sampling interval.

This is an effective method of implementing a real-time controller in XENIX. There are disadvantages to having the controller at the device driver level, however. First is software development time. Drivers must be physically linked to the XENIX kernel. This takes about 15 minutes and substantially increases the development time for the controller code. Secondly, since device drivers have full access to the system, programming errors may destroy the software system (requiring XENIX to be reloaded from diskette). In spite of these problems, this still seems to be the most practical way of building a controller in XENIX.

7. Application and Conclusion

7.1 Application

Once the Joint Interface Board was constructed and debugged, the basic I/O drivers were installed into the XENIX system and tested. After the basic system became operational, a simple but complete example of a controller was designed and tested.

The objective of this project was to design and construct the hardware and interface software to implement a robot controller. To perform a functional test on the entire system, a simple six-axis P.D. digital controller was implemented. In addition to testing the integrated system, it also served as a documented application guide for use of the JIB and the XENIX interface.

Figure 4 shows the basic controller system. The controller is divided into five distinct subsystems: (1) the application software which issues high level joint motion commands (kinematics, path planning, etc.) and runs in the normal application environment of XENIX; (2) kernel level driver software which interprets the read and write commands from the application programs; (3) interrupt level driver software which "services" the timer interrupt by executing the control structure software, reading and writing directly to the JIB hardware; (4) the Joint Interface Board which interfaces the computer joint motor signals; and (5) the robot arm itself, including power amplifiers, joint motors and feedback elements.

The simple P.D. controller implemented in this project was able to satisfactorily control all six PUMA 560 joint motors simultaneously. The P.D. coefficients were experimentally determined by trial and error. This was done one joint at a time while the other joints were locked. When all joints were operated together, the strong coupling between joints 2 and 3 (shoulder and elbow) caused strong oscillations. The gains of these joints were reduced to produce a more stable system. This is an area where more sophisticated control techniques should produce improved results.

7.2 Conclusion

The basic objective of designing and constructing a general purpose robotic controller was completed successfully. The system has been used to control the PUMA 560 robot arms, demonstrating the functionality and flexibility of the design. The Joint Interface Board has served its overall design objective well.

Using the XENIX operating system was done with mixed results. High level software is easily developed (at least for UNIX users). Whereas the method of low-level servo-loop software programming was somewhat less than desirable in that routines on this level must be directly linked (using the 'ld' linked) to the XENIX kernel. Therefore, it involves a fairly time-consuming task. XENIX also prohibits writing C-code in the kernel level which uses the floating point coprocessor (via an undocumented c-compiler flag). This was disappointing, but there are ways around this problem. This last issue is an area where more time and effort would be useful.

8. References

[1] D. G. Bihn, A Universal Six Joint Robot Controller, M.S. Thesis, Department of Electrical and Computer Engineering, University of California, Davis, 1986.

[2] Intel Corporation, System 310 Installation an Operation Guide, O.N. 173211-002, October 1983.

[3] Intel Corporation, XENIX 286 Reference Manual, O.N. 174390-008, 1984.

[4] Intel Corporation, Introduction to the iAPX 286, O.N. 210308-001, 1985.

[5] Intel Corporation, Introduction to the iAPX 287, 1985.

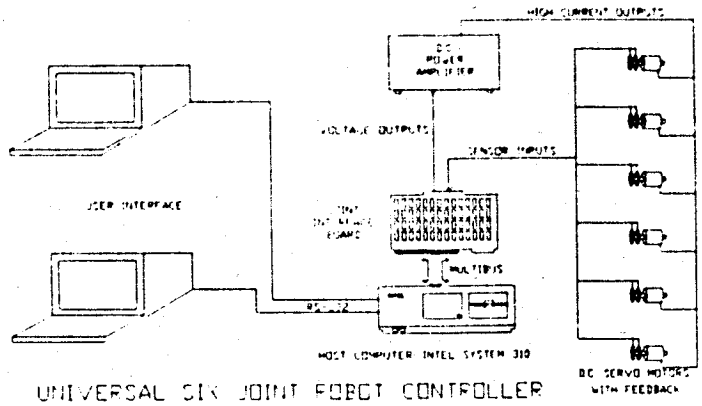
[6] Intel Corporation, Intel Multibus Specification, O.N. 9800683, 1978.

[7] Analog Devices Inc., "Fast, Complete 12-bit A/D Converter with Microprocessor Interface," Data-Acquisition Databook 1984, Volume I Integrated Circuits, 10-55, 1984.

[8] Intel Corporation, "8254 Programmable Interval Timer," Microsystem Components Handbook, Volume II, 5-240.

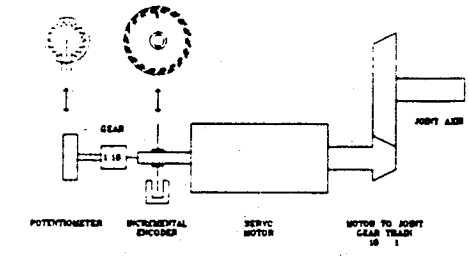
[9] Unimation, "500 Series Electrical Drawing Set for VAL II and VAL Plus Operating Systems," Unimate PUMA Mark II Robot, 394AC1, July 1985.

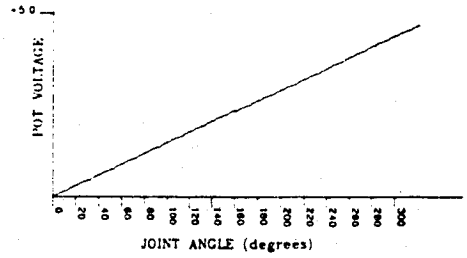
[10] Intel Corporation, XENIX Device Driver Guide, O.N. 174393-001, 1985.

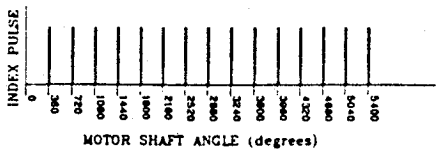


UNIVERSAL SIX JOINT ROBOT CONTROLLER

Figure 1

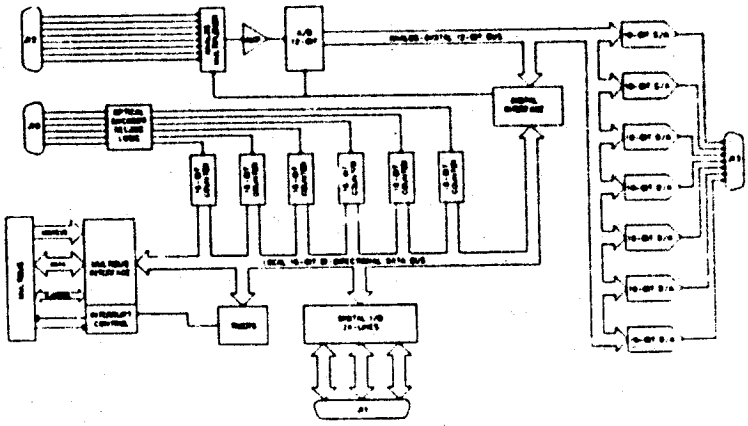






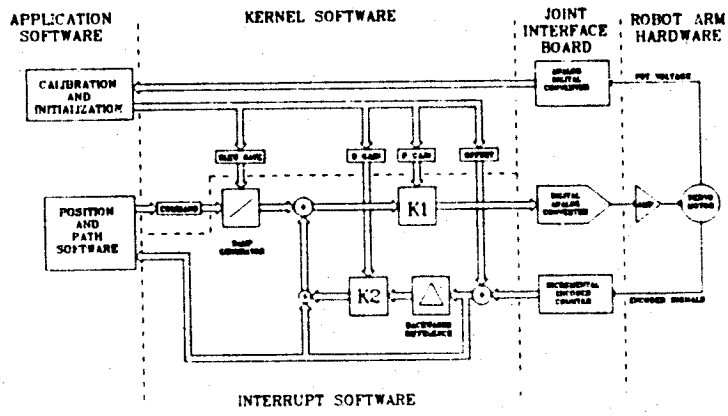
TYPICAL JOINT MOTOR CONFIGURATION

Figure 2



JOINT INTERFACE BOARD BLOCK DIAGRAM

Figure 3



CONTROLLER LOGICAL DIAGRAM

Figure 4