S25-37
78
185107

High Performance Architecture for Robot Control

E. Byler
Grumman Space Systems
Bethpage, NY 11714

G7180425

J. Peterson
Grumman Data Systems
Woodbury, NY 11797

1. Abstract

~~This paper discusses~~ practical aspects of the design and implementation of a modular, high performance, parallel computer control system for telerobots. Topics of consideration include system architecture, operator interface, and control execution. In a laboratory environment, a telerobotics test control configuration is used to obtain measurements on communications and control loop timing for use in an effective full scale operational system design. The feasibility of the selected architectural approach has been successfully demonstrated. The modularity of the software and hardware enables ease of transport for use in the operational system. A ~~major portion of this paper is devoted to the~~ distributed partioning of the control algorithms and the performance measurements acquired during control system implementation are discussed

2. Introduction

Telerobots are manipulators designed to be controlled both directly by a computer and remotely by a human. This combination method, called supervisory control, allows the operator to apply his intelligence to the task without having to maintain continuous control. Computational requirements for robot control either limit the complexity of the control system, or require prohibitively large computers to obtain the required cycle rate. Remote operation and human control requirements create additional environmental constraints. The design of a hierarchical, parallel computer system is described. It provides the required speed within the prescribed design limits at a minimum weight and size. A prototype system was built and measurements were made to verify several performance issues.

3. System Description

The space-based telerobot for which this computer system was designed (Grumman [1]) has several aspects that drive the design of the computer system.

The first major design requirement is imposed by the robot itself (Fig. 1). There are three arms, each with seven joints and an end-effector, a torso, a vision system, and tools and test equipment. All this equipment is under computer control. Each arm must operate under control of several different types of control algorithms to provide flexibility to enhance operator productivity. Two arms must be capable of coordinated action. The manipulation requirements impose definite computation requirements.

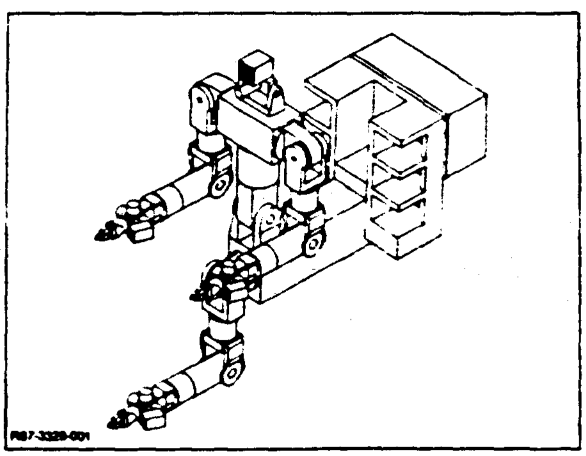

Fig. 1 Telerobot

A second major design driver results from the control devices used to operate in the various control modes. In order to maximize productivity, a variety of control inputs and information displays are needed. Input control devices include a replica master, 6DOF stick controllers, voice control, a lightpen, a keyboard, and a touchbezel. Output devices include graphics displays, monitors, and force reflection (Fig. 2). Each of these require some data processing.

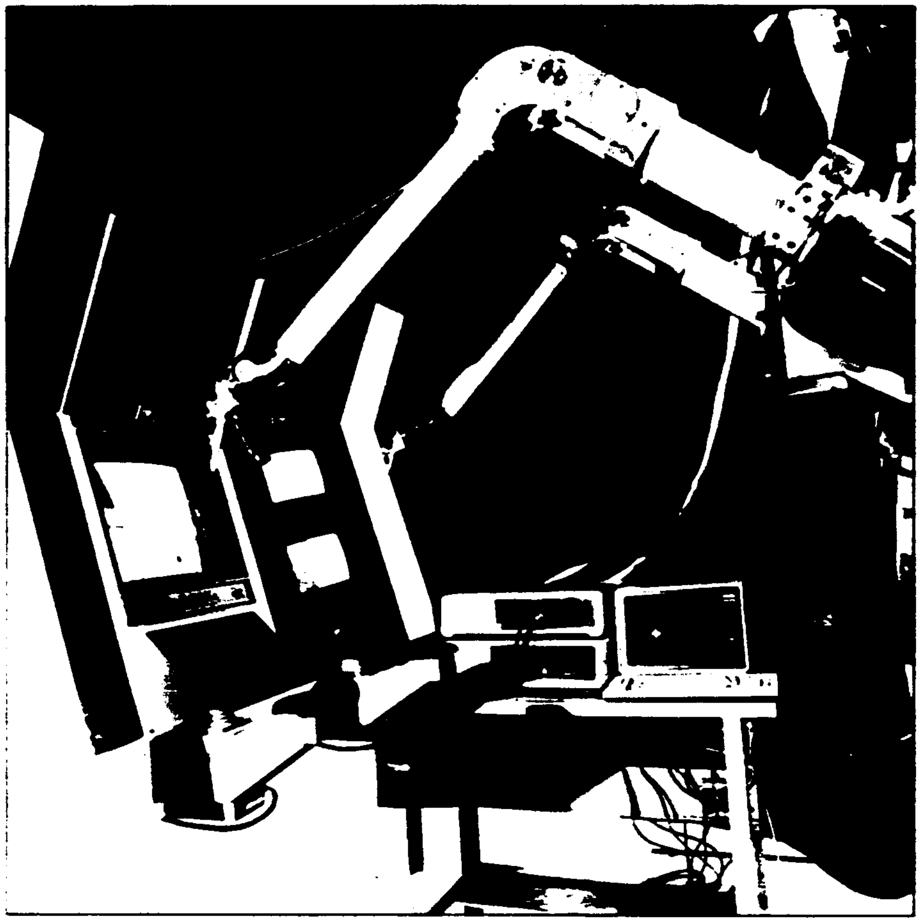

Fig. 2 Telerobot Control Station Testbed

The third major design driver is that the robot is remote from the control center. This creates a requirement for a communication link. Any communication link imposes restrictions and trade-offs on total data bandwidth.

A final design driver is the set of different control algorithms required for human control and robotic control. These impose requirements for different data and control paths. Different control algorithms have different loop rates for stability, within which all computations must be performed.

4. System Specification

The telerobot is required to run three different control algorithms: Bi-lateral Force Reflection (BFR), Resolved Rate (RR), and Active Force Control (AFC) (Fig. 3). Both BFR and RR must have human control. AFC uses computer control only. BFR has large communication requirements in that a complete set of sensor signals must be passed both from the masters to the slaves, and from the slaves to the masters. Good performance is obtained with updates of about 66 Hz. AFC has a required update for its command vector of about 400 Hz in order to maintain stability when the manipulator is in contact with elements of its environment.

Distributed processing was considered to overcome the computational bottleneck inherent in robot control. First, each algorithm was broken into a block diagram by considering which elements were on different levels, which could be run asynchronously, and which were computationally intensive, or could be split into separate variable types (Fig. 4). Then, the different algorithms were compared to determine which elements were common. Data requirements, computation times, and output for each element were tabulated.

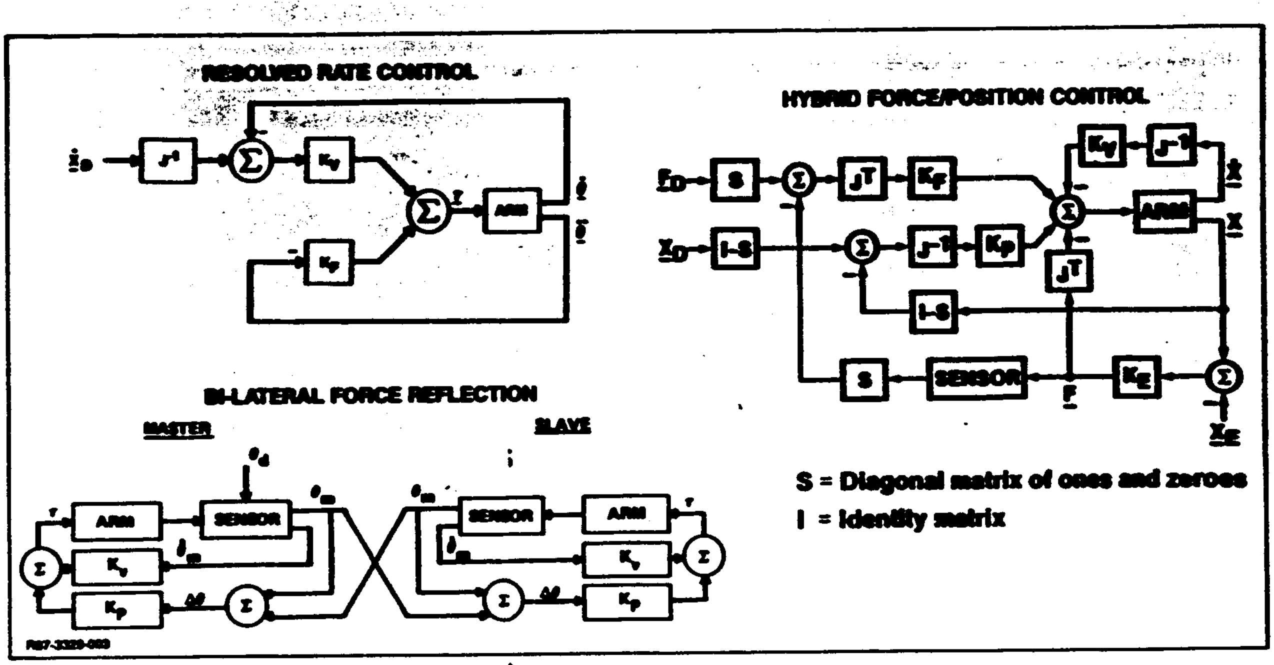

Fig. 3 Telerobotic Modes of Control

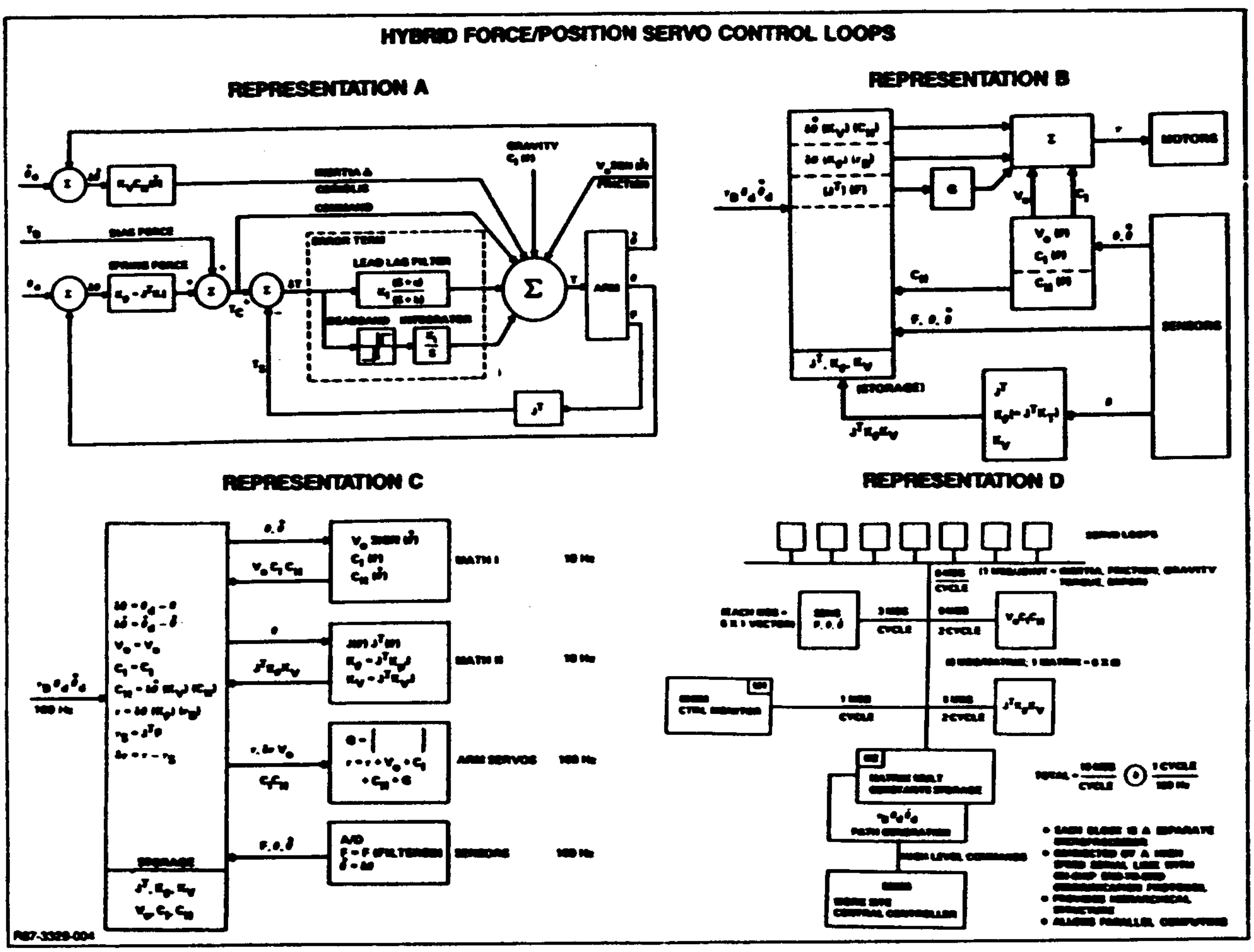

Fig. 4 Evolution of Force Control Algorithm Partitionment

Finally, these separated elements were considered in conjunction with the information flow and general system placement requirements to arrive at a coherent computer architecture (Fig. 5). Processor/algorithm granularity was driven by available commercial products and by accuracy requirements.

This architecture matches the NASA/NBS standard reference model telerobot control system architecture (NBS [2]) for levels one, two, and part of three. Servo control and parameter calculation take advantage of parallel processing techniques. Task planning and path generation are split into hierarchical processes.

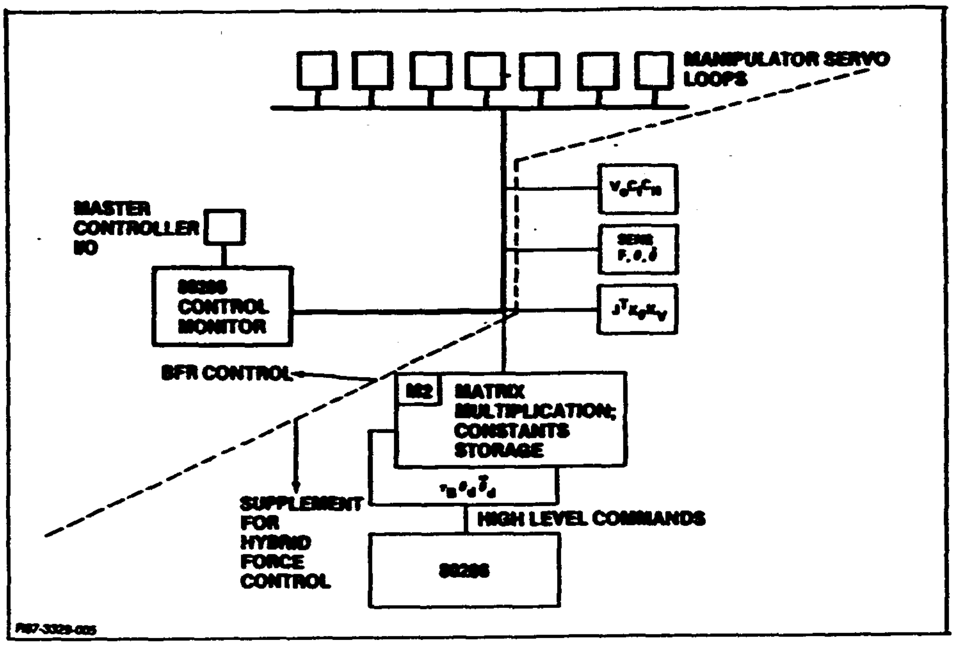

Fig. 5 Selected Computer Architecture

5. Performance Issues

Several issues were examined to verify the performance of this parallel computing system. First, central to the distributed control architecture is the correct partitionment of alogrithms among the parallel processors to properly distribute the computing loads. Second, communications bandwidth requirements of algorithm components must be considered to partition the algorithms effectively without creating bottlenecks. Third, CPU consumption by algorithm components must be analyzed. Finally, use of operating system and communication system services must also be examined to eliminate unnecessary overhead.

6. Implementation

The development environment for the digital robot control makes use of Intel microprocessor and microcontroller chip, board, and bus technology. The 80286 based System 286/380 is used to run high-level robot path planning code. The system is presently used as both a software development and target machine for implementing and running controlsite software. The 8085-based Personal Development System (PDS) is used for developing 8044 microcontroller code, and contains the necessary text editor, cross assembler, linkage editor, and monitor tools for firmware development. An EMV44 emulator provides a window to the 8044 microcontroller for debug.

The Distributed Control Module (DCM) package is the set of hierarchical, microcontroller nodes and complimentary communications software that forms the backbone of the prototype robot control system. It includes the iRCB 44/10 Remote Controller Board, the iSBX 344 Multimodule Board, the DCM Controller, and the BITBUS Interconnect. A distributed control node may be either an iRCB 44/10 or an iSBX 344 board. The DCM controller is actually an 8044 microcontroller chip with built-in SDLC communications firmware and is part of all boards. The BITBUS Interconnect is the high-speed serial communications link for the network of nodes.

The development configuration was designed specifically to be able to measure system performance. It groups six microcontrollers into one distributed network of nodes, linked together by a high speed serial communications bus (Fig. 6). The communications link implements a SDLC master slave protocol on chip and performs all message handling and error checking automatically without help from the CPU. Mastership of the network is claimed by either the Personal Development System 344 microcontroller node, or the System 286 344 node and is a function of the particular control algorithm currently executing. The 44/10 microcontroller nodes plug into a common cardframe which supplies the required power and communications wiring (Fig 7).

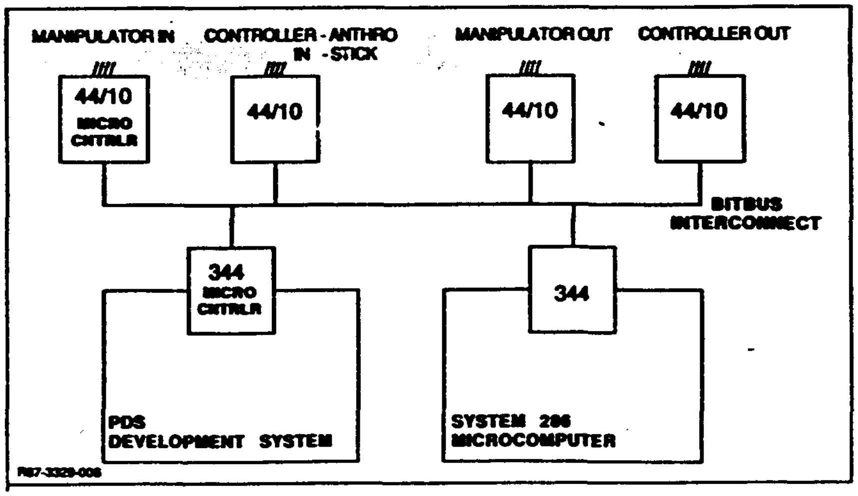

Fig. 6 Development Configuration Block Diagram

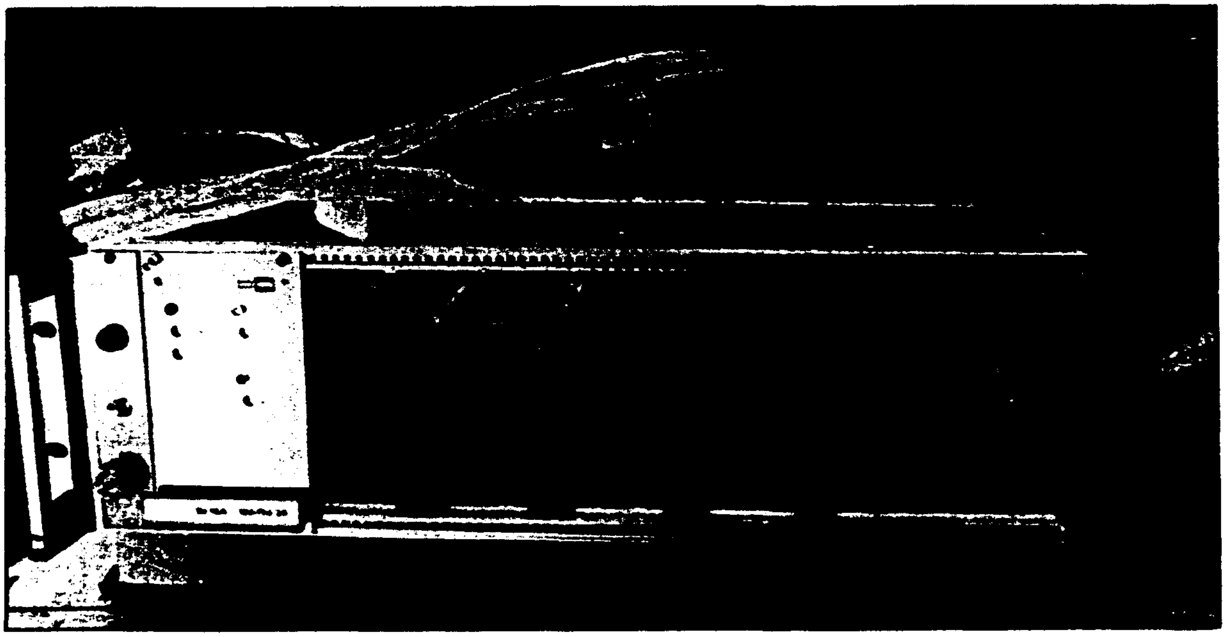

Fig. 7 Prototype Parallel Computer System for Telerobot Control

7. Measurements

Measuring the system performance is accomplished by attaching an oscilloscope to the pins on one of the dedicated parallel I/O ports (Fig. 8). The separate signal lines are toggled by the development firmware to provide a mechanism for viewing the algorithms as they run (Fig. 9). Each bit of the 8-bit port is dedicated to provide the timing for different components of the algorithms running on each processor.

Measurements were taken in three areas to examine system performance and algorithm partitionment. The three areas examined were communications, CPU utilization, and operating system.

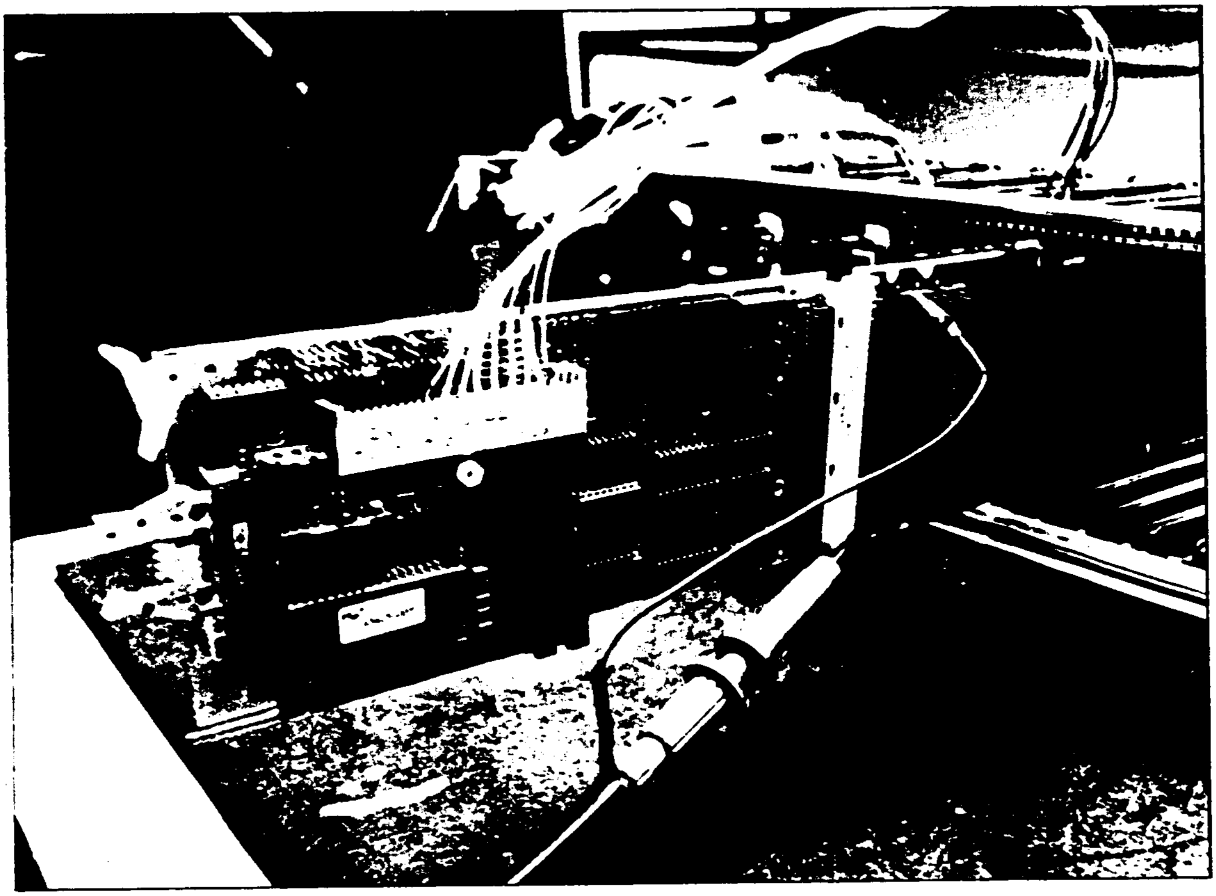

Fig. 8 Oscilloscope Attachment for Performance Measurements

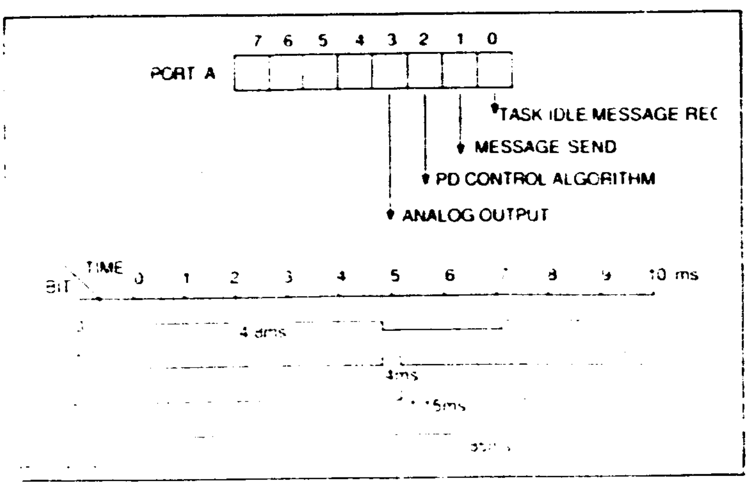

Fig. 9 Measurement Process

A variety of communications strategies have been implemented and measured for their performance. The effects of preloading the Serial Interface Unit (SIU) controller and the effects of serial and parallel message passing techniques were measured and compared. In addition, the effects of variations in asynchronous and synchronous communications bandwidth on overall timing were examined.

The preloading technique involves setting up the SIU of a slave node for a reply prior to receiving the order from the master. A time savings equivalent to the required computation time of the slave node for one control loop cycle, is observed when the SIU is preloaded.

Serial and parallel message passing techniques differ in the sequence by which the master node communicates with the slave nodes. Serial message passing techniques cause the master node to wait for replies after every message sent, which may cause other nodes to wait for their turn if the reply requires some computation. Parallel message passing techniques allow the master to have several outstanding messages, allowing slave nodes more chance to process in parallel and reduce control loop timing (Fig. 10).

The communications bandwidth has significant effect on overall timing. With the nodes jumpered for synchronous serial communications on the BITBUS interconnect, a 2.4 Mbps transmission rate is achieved as compared with the slower asynchronous rate (Fig. 11). The time savings is readily observed when the master uses serial message passing.

	SERIAL	PARALLEL
PRELOADED 4/4 NODES	7.2ms	7.2ms
HALF PRELOADED 2/4 NODES	12.0ms	8.8ms
NO PRELOADED 0/4 NODES	14.2ms	9.8ms

R87-3329-010

Fig. 10 Communications Schemes

SYNCHRONOUS 2.4M CRYSTAL OSC. ALL BOARDS	5.0ms
SYNCHRONOUS 2.4M CRYSTAL OSC. MASTER ONLY	6.1ms
SYNCHRONOUS .92M ALE CLOCK	7.2ms
ASYNCHRONOUS 375K	9.2ms

R87-3329-011

Fig. 11 Clock Rates

CPU utilization measurements verify how effectively the algorithms are partitioned. For example, the slave input nodes measured a 60% idle time. This reflects a capacity to handle more of the computational burden of the system. Computation measurements for different parts of the algorithm equations enable efficient redistribution of portions of the algorithm to fine tune the design. For example, the PD loop timing was approximately 0.8 ms. This measurement helped to redistribute the gain computation of the servo loop to a more appropriate node to optimize the cycle time of the loop.

Operating system calls have a significant effect on algorithm timing. For example, the preloading of a message for transmission by a slave node takes approximately 0.4 ms. Internal RAM buffer space (system pool space) must be allocated for a message dynamically, via a system call, if it is to be sent by a task on one node to a task on a different node. Since there is a limited amount of buffer space (approx. 4 20-byte buffers) available for a message send, buffer space will be unavailable until the buffers associated with previous sends are released after successful transmission. A method of reducing the allocation calls at the master node is to use the buffer allocated to hold the previously received reply message for the next order message since this buffer is automatically allocated by the system.

Use of the standard RAC communication services may also impose penalties. Passing a message through the RAC services automatically causes task context switching. This is important if the node is on the critical path either in terms of calculations or buffer allocations. The time penalties can be avoided by taking direct control of the receive buffers. The applications firmware must then provide the communication error handling capability.

8. Conclusions

Distributed processing of control algorithms is a feasible solution to improve robot control computation speed. Measurement of system elements is important in order to optimize system CPU usage and avoid bottlenecks. Application programs must be careful to avoid unnecessary operating system use.

This architecture, as implemented, covers the National Bureau of Standards Hierarchical Control Specification levels one, two, and part of three. It is independent of path generation and task description techniques and runs bilateral force reflection as easily as resolved rate. Hardware weight and power consumption are very low and are appropriate for space flight hardware.

9. References

[1] "Telepresence Work System, System Definition Study," Grumman Aerospace Corporation, October 1985.

[2] "NASA/NBS Standard Reference Model Telerobot Control System Architecture Preliminary Draft," National Bureau of Standards, September 1986.