CONF-910223--6

DE91 004151

A DISTRIBUTED DATA ACQUISITION SOFTWARE SCHEME FOR THE LABORATORY TELEROBOTIC MANIPULATOR*

P. L. Butler, R. L. Glassell, and J. C. Rowe
Oak Ridge National Laboratory†
Robotics and Process Systems Division
P. O. Box 2008
Oak Ridge, Tennessee 37831

To be presented at the
AMERICAN NUCLEAR SOCIETY
FOURTH TOPICAL MEETING ON ROBOTICS AND REMOTE SYSTEMS
Albuquerque, New Mexico
February 24-28, 1991

*Research sponsored by the U. S. Department of Energy and the National Aeronautics and Space Administration.

†Managed by Martin Marietta Energy Systems, Inc., for the U. S. Department of Energy under Contract No. DE-AC05-84OR21400.

A DISTRIBUTED DATA ACQUISITION SOFTWARE SCHEME FOR THE LABORATORY TELEROBOTIC MANIPULATOR*

P. L. Butler, R. L. Glassell, and J. C. Rowe
Oak Ridge National Laboratory †
Robotics and Process Systems Division
P. O. Box 2008
Oak Ridge, Tennessee 37831

ABSTRACT

A custom software architecture was developed for use in the Laboratory Telerobotic Manipulator (LTM) to provide support for the distributed data acquisition electronics. This architecture was designed to provide a comprehensive development environment that proved to be useful for both hardware and software debugging. This paper describes the development environment and the operational characteristics of the real-time data acquisition software.

INTRODUCTION

With the increased emphasis on space exploration and habitation, the need for robotic systems that can extend human capabilities in the hazardous environment of outer space is greatly apparent. To help meet the goal of increased use of robotics in space, Oak Ridge National Laboratory (ORNL) has designed and developed a dual-arm, 7 degree-of-freedom (DOF) master/slave manipulator for ground-based research at NASA's Langley Research Center (LaRC). This manipulator, the Laboratory Telerobotic Manipulator LTM [1], provides a two-fold test-bed for space manipulator research. First, new design and fabrication techniques were developed to ascertain their feasibility for eventual use in a highly reliable space manipulator. Second, the LTM provides valuable performance information as to what is feasible for a telerobotic system.

Many techniques of manipulator design were brought together in the design of LTM. First, LTM had to be a good force-reflecting teleoperator. This normally requires low reflected joint friction to reduce operator fatigue and high joint back drivability for good force-reflection. Second, LTM also had to be a good robot. This requires high joint stiffness for increased end-effector positioning accuracy. These otherwise conflicting constraints were solved with a modular pitch-yaw differential joint design and is complimented with control compensation software [2,3].

---

* Research sponsored by the National Aeronautics and Space Administration, Langley Research Center.

† Managed by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy under contract DE-AC05-84OR21400.

LTM provides NASA LaRC with a valuable research tool for performance testing. One aspect of this performance testing is task segmentation and evaluation. Another aspect is evaluation of training techniques for operators. These two diverse issues may eventually lead to a semiautonomous system that can take advantage of past operation successes and failures.

ORNL also provided a similar system, called the MicroGravity Manipulator (μGM), to NASA's Lewis Research Center (LeRC) for microgravity research. This system consists of a single-arm, 4-DOF manipulator used in a robotics mode. NASA LeRC uses the μGM to explore the forces and torques that a robotic arm places on its support as it executes a typical motion.

LTM/μGM CONTROL ARCHITECTURE

The computer architecture for the LTM system is arranged in a hierarchical, modular fashion [4]. The computer system consists of two identical VME racks, a master and a slave, as shown in Fig. 1. The master is also connected, via a 9600-baud serial link, to a Macintosh II computer, which serves as a human-machine interface. There are three commercially available Motorola 68020 CPU cards on each rack. One processor, the control processor (CP), serves as a control and communications processor and the other two 68020 CPU cards, called arm processors (APs), run the control loops for the right and left arms. Communication between the racks is accomplished using a 10-Mbaud Proteon link, which provides the necessary data swapping for the servo-level control loops. The data swapping is controlled by the master rack and swaps 232 bytes from one rack to another at 250 Hz.

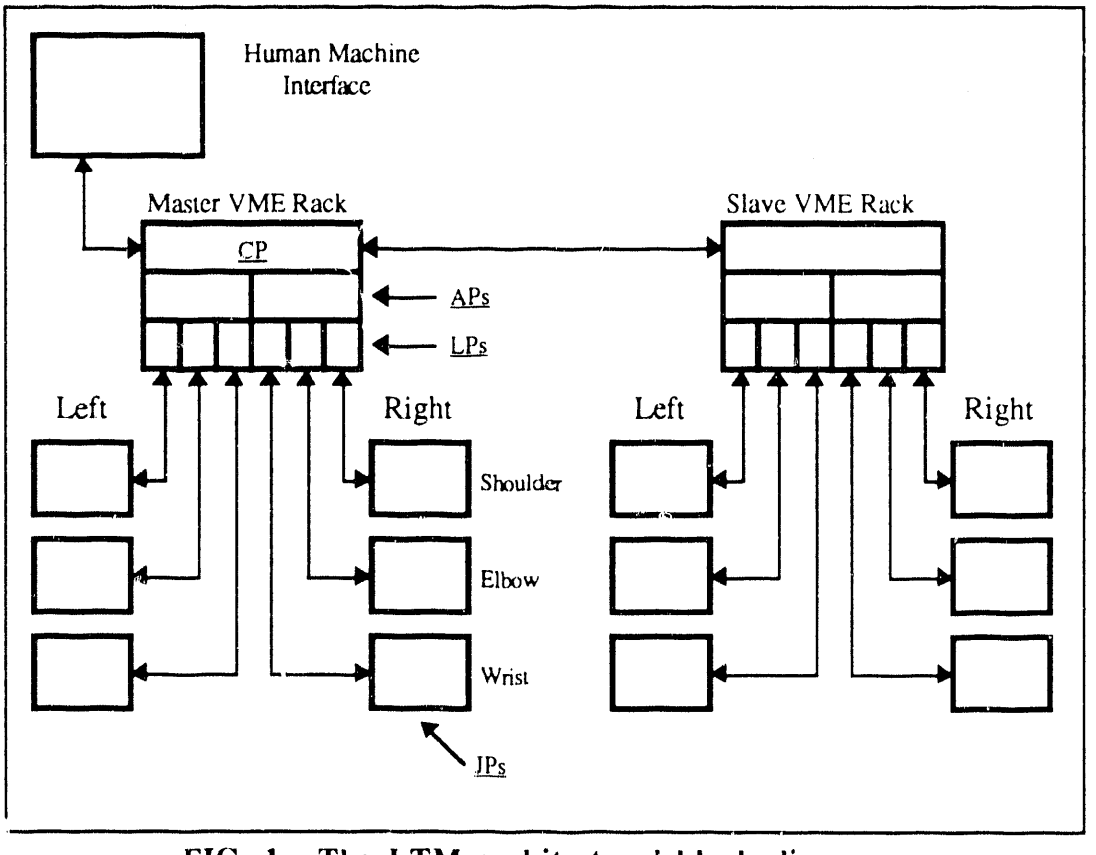

FIG. 1. The LTM architectural block diagram.

Custom data acquisition processors [5], based on the Intel 80C196 microcontroller, reside within each pitch/yaw joint. This arrangement has the advantage of minimizing the wiring from dozens of analog signals to a single bidirectional, fiber optic line for each joint. A multi-drop arrangement using a single fiber optic line for each arm was considered but not implemented because of the higher communications bandwidth that would have been required. Since the joint processors (JPs) are directly within the joint, signal noise is also greatly reduced. An Intel 82588 local-area-network (LAN) controller, with a data rate of 2 Mbaud, is used to communicate with the link processors (LPs) within the VME racks. LPs handle the limit checking and place the data acquired by JPs directly into dual-port RAM (DPRAM) for access by the arm processors. The functional organization of a single LP/JP pair is shown in Fig. 2.

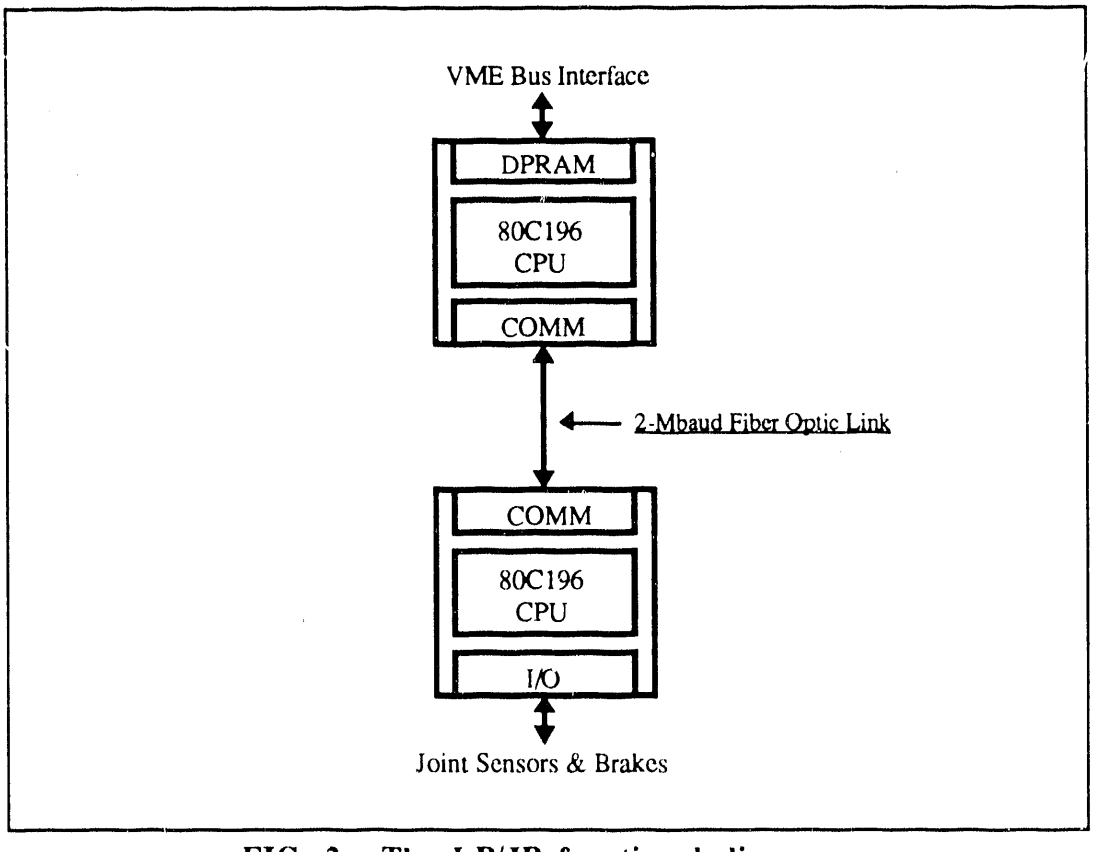

FIG. 2. The LP/JP functional diagram.

The OS-9 operating system from Microware is used for the main control algorithms, providing a real-time, multitasking development environment. Most of the operational code is written in the C language, but some Forth is used for utilities and debug support. All of the LP/JP code, however, was written in the Forth language. This provides an interesting challenge – interfacing between the C language and the Forth language. This problem is solved by having the LP/JP data acquisition system place data in common memory to conform to a predefined C structure. After initialization, the control algorithms only need to reference this C structure for the required data.

DATA ACQUISITION REQUIREMENTS

The LP/JP data acquisition software provides fast, reliable acquisition of all analog and digital data points within each pitch/yaw joint. These data points are acquired by the JP and transmitted to the LP using one of the independent 2-Mbaud, bidirectional fiber optic links. The LP places the data into the dual-port RAM for use by the arm processors and communication to the other VME rack. The data acquisition software also provides error checking and recovery from occasional missed or corrupted packets. Since the LP is more directly controllable by the CP and the APs, it is actually in control of the LP/JP synchronization. The LP receives a periodic interrupt to communicate command information to the JP. The JP receives the command packet, starts data acquisition, and sends a return packet back to the LP with the newly acquired data. After the LP receives this packet, it performs certain safety limit checks and places the data into the dual-port RAM for use by the APs.

The JP also has control over the individual joint brakes. Therefore, it must accept commands from the LP to lock or unlock the brakes. This places additional safety constraints on the design of the software while providing a redundant safety system. In the event of loss of LP/JP communications, the JP automatically resets and forces the brakes to lock. Only with valid communications can the brakes be unlocked, activating the joint. As an example of the redundant safety system, the AP can also disable brake power on each arm, thus locking the brakes.

A development mode was required for direct debugging of both hardware and software. It was decided that the development mode should be as close as possible in nature to the acquisition mode. This minimized the effort that would have normally been required to implement two totally separate modes. In fact, both modes can be active at the same time. In other words, development can be in progress while data are actually being acquired. Due to the number of LP/JP pairs in the system, this development environment was implemented to allow direct multiprocessor debugging on any LP or JP in the system from a single terminal.

FORTH LANGUAGE

Forth was chosen as the development language for the data acquisition system because of its highly interactive nature. Forth allows a minimal system to have powerful debug capabilities, important because of the limited RAM and ROM on the link and joint processors (16 Kbytes of each). In addition, the Forth system is open and can be modified for a specific application. For a development environment, Forth requires a terminal and disk interface. However, the user is not bound by conventional terminal and disk interfaces. For example, the DPRAM interface provides the terminal and disk interface to the LPs and the 2-Mbaud link provides the same interface from the LPs to the JPs. A server program on the VME system connects these logical interfaces to the actual RS-232 VT-100 terminal and the OS-9 file system.

ROM CODE FOR LINK AND JOINT PROCESSORS

The ROM code holds the bootstrap mechanism for the data acquisition system. The ROM code contains the polyFORTH kernel [6], communication controller code, and virtual terminal and disk support. Because of the similar nature of the LP and

JP (processor type and address maps), it was decided that one ROM set should plug into both the LP and JP. This reduced code management by a factor of two during the ROM development. It did, however, require some "manipulations" to get around the few differences between the LP and JP. For example, the 82588 communication controller chip interrupt line was connected to different 80C196 interrupt lines on the two processor boards. This required a table lookup scheme to allow two different interrupt behaviors while not compromising speed. On power-up, the ROM code can determine which board it is running on by trying to write and read from DPRAM. If it sees DPRAM, the board is a LP. Otherwise, it is a JP.

The ROM code for the LP waits for a command from the VME system and then initiates communications with the JP. Once started, communication with the JP proceeds at approximately a 620-Hz rate. The original design called for a 1-Khz loop-rate, which was initially attained. However, as the system evolved, some of the error checking was off-loaded from the arm processors and placed on the link processors. Therefore, the fastest rate that the loop could run was determined to be 620-Hz.

The JP has to perform additional tasks at power-up. For safety reasons, the JP locks the joint brakes on power-up. Only when valid communication is established with the LP can the VME system command the JP to unlock the brakes. The Intel 80C196 processor also provides support for a watchdog timer that will reset the processor if a certain control port is not loaded with the proper pattern on a periodic basis. This watchdog support is tied to the communication link on the JP so that if communications were ever lost, the JP would automatically reset, therefore locking the brakes. The LP does not utilize the watchdog capabilities of the 80C196, but does update a heartbeat counter that the APs check to ensure the acquisition system is functioning properly.

The LP communication is through the DPRAM interface. A server program on the CP polls a chosen LP for I/O requests. Character I/O on the LP is a simple matter of reading or writing characters to DPRAM. A semaphore handshake flag is used to tell both the LP and VME system when valid characters are in DPRAM. Disk I/O is similar to character I/O, except that the LP ROM tells the VME system where in DPRAM to read or write a 1024-byte block of data.

The 2-Mbaud communication link between the LP and JP is synchronized by the LP, as shown in Fig. 3, so that it can be synchronized by the VME control system. A 620-Hz timer interrupt on the LP starts a packet transmission from the LP to the JP. After the JP receives this packet, it transmits a data packet back to the LP. If, at any time, communications is lost, the LP can notify the VME system of this condition. This communication handshake was determined to be what the actual data acquisition code required, so the Forth support was designed to conform to this scheme. To do this, a packet header is at the start of every packet with Forth terminal and disk handshake information. To keep the interrupt time constant, special consideration had to be given to disk I/O transfers of 1024 bytes. Disk I/O during normal development uses 32-byte chunks between the LP and JP. However, during data acquisition, disk I/O is limited to 2 bytes for every time slice. This slows disk transfers to slightly over 0.5 s while the JPs are acquiring data but was considered acceptable, since a programmer would probably not be doing code development during an actual run.

LINK PROCESSOR FUNCTIONS

The run-time code for the LP is primarily responsible for placing the JP data into the dual-port memory. The LP will also perform certain data conversions before placing the JP data into the dual-port memory. For example, zero offsets are subtracted from the raw data for use by the control algorithms. In addition, the LP performs safety limit checks for all critical data points. The main rationale is to offload and parallelize the processing from the arm processors required to perform limit checks. The arm processors only have to read a single status byte from each pitch/yaw joint to tell if an error occurred. The LP is also sensitive to corrupt data packets. If a communication error occurs, the LP keeps the old data instead of placing corrupt data into the dual-port memory common block. However, if a large number of sequential bad data packets arrive, then the LP will notify the VME control system, initiating a system shutdown.

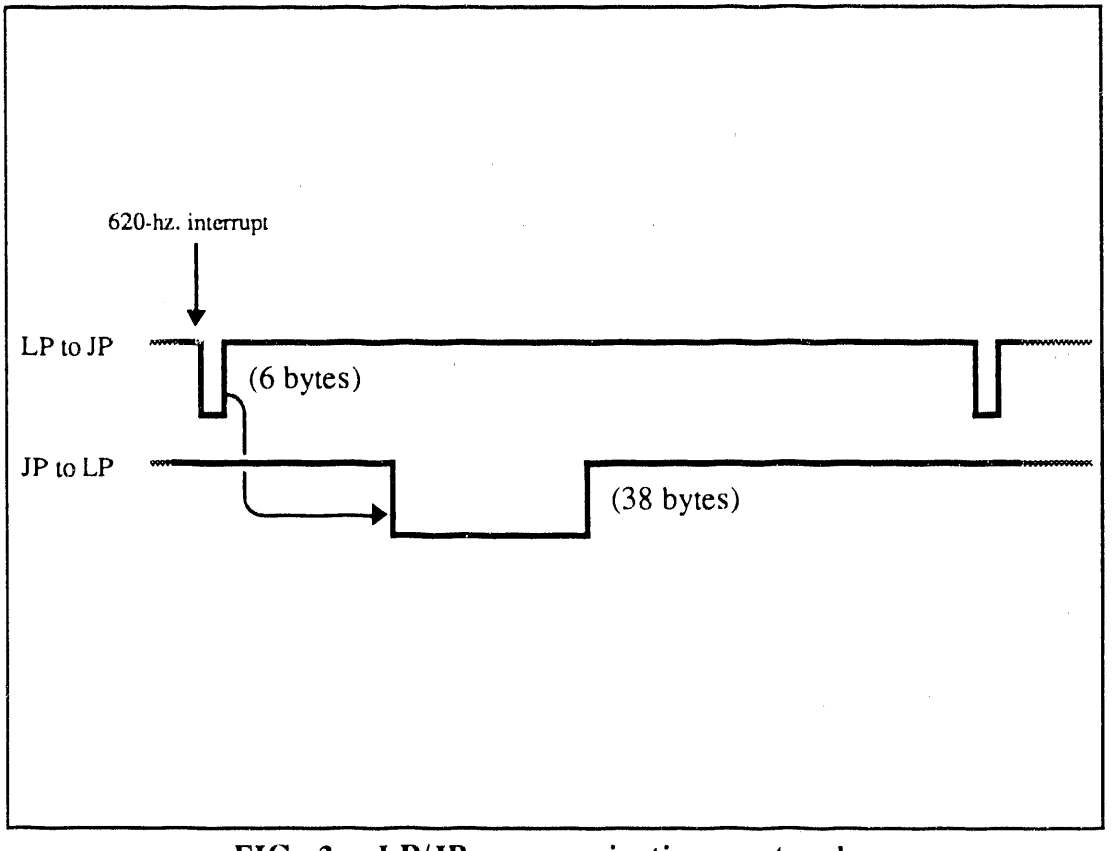

FIG. 3. LP/JP communication protocol.

JOINT PROCESSOR FUNCTIONS

The data acquisition code is downloaded into RAM upon system initialization by the VME system. This allows quick modifications to be made on the data acquisition code. The actual data acquisition routines are machine coded for speed and also for easy incorporation into the communication link interrupt routines. The Forth language was used for the debugging of these routines and can be used interactively during a run to check on their progress. One interesting point of the data acquisition code is that lower priority data are read after the main data block has been transmitted. This keeps the main acquisition speed up and allows convenient

control over an 8 to 1 analog multiplexer. One channel of this multiplexer is read after the main data packet has been sent back to the LP. Therefore, lower speed signals such as motor temperatures are updated every 8 time-slices.

DEBUG SUPPORT

A program called SERVE on the VME system allows extensive capabilities for software development and system debugging. The multiprocessor development environment uses the same philosophy that was developed for OPSNET [7,8]. SERVE allows any LP or JP in a particular rack to be brought up in a debug mode with an interactive Forth prompt. Several debug words allow for quick checks on the integrity of the data being read by the JP. Fig. 4 shows the main SERVE status screen. From this screen, an operator can select which LP or JP to "attach" a terminal to, check status and error codes, and determine brake status. Fig. 5 shows the same terminal after it has been switched to one of the LPs. Notice that the name of the attached processor is always displayed on the bottom of the screen. In this case, a program is running, called SHOW, that displays all important information that the LP is receiving from the JP. SHOW also keeps track of signal minimums (second data column) and maximums (third data column) for use by system developers.

```
Laboratory Telerobotic Manipulator System                    01/26/90
8096 processor server system (01/23/90)                      Master Rack
                                                             Beep On
          Processor          Stat Err  MotorA    MotorB      Roll
Online    Right Shoulder LP  C0   00   Locked    Locked
Online                   JP  C0
Online    Right Elbow    LP  C0   00   Locked    Locked
Online                   JP  C0
Online    Right Wrist    LP  C0   00   Locked    Locked      Locked
Online                   JP  C0
Online    Left  Shoulder LP  C0   00   Locked    Locked
Online                   JP  C0
Online    Left  Elbow    LP  C0   00   Locked    Locked
Online                   JP  C0
Online    Left  Wrist    LP  C0   00   Locked    Locked      Locked
Online                   JP  C0

Q  - Quit to OS-9                          T  - Toggle beep mode
U  - Move up in processor status           C  - Clear error status
D  - Move down in processor status         ^B - Load all processors
sp - Loop through processor status         ^D - Load selected processor
cr - Serve selected processor              ^R - Reset selected processor
```

FIG. 4. – SERVE debug screen.

CONCLUSION

The LP/JP data acquisition software of the LTM project has demonstrated proper design of a software system to minimize the difficulties normally associated with multiprocessor system configuration. One important factor is to create a modeless communication system that is useful both in development and run-time operation. Another factor is to create a multiprocessor development environment that allows software to be debugged with a minimum of effort.

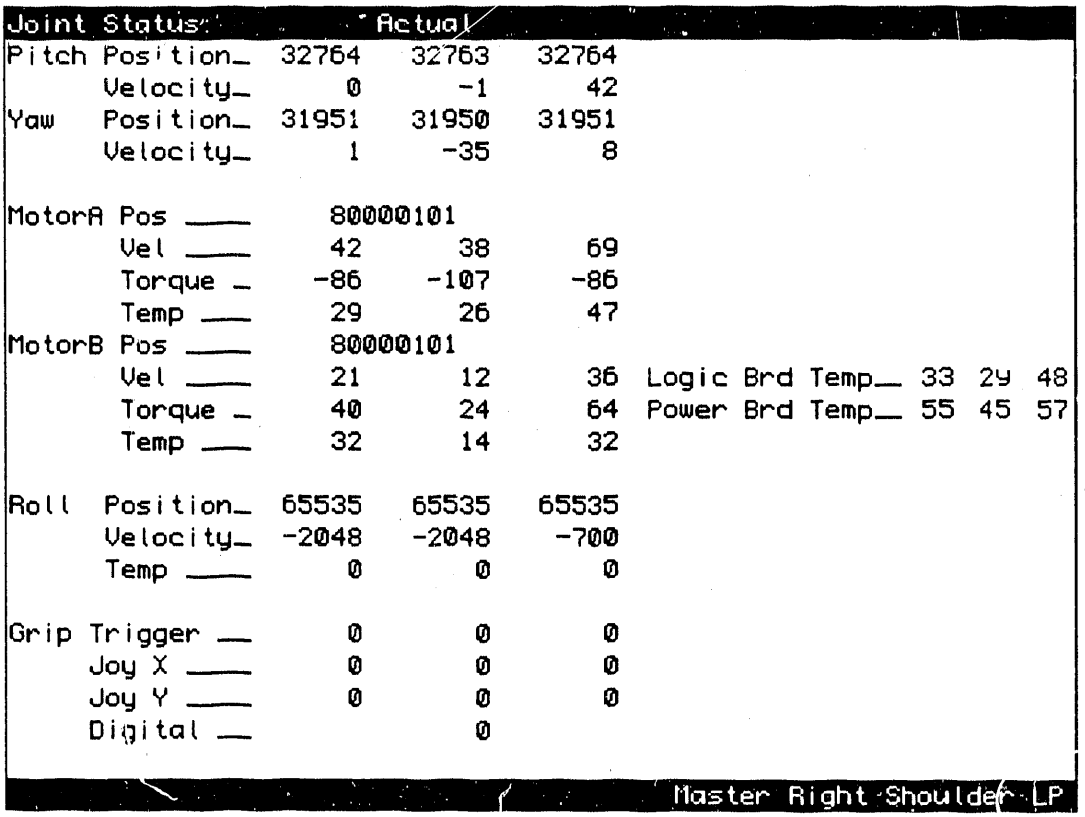

FIG. 5. The joint status display.

REFERENCES

1. J. N. Herndon et al., "The Laboratory Telerobotic Manipulator Program", Proceedings for the NASA Conference on Space Telerobotics Volume IV, Pasadena, California, January 31–February 2, 1989.
2. J. F. Jansen and J. N. Herndon, "Design of a Telerobotic Controller with Joint Torque Sensors", IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, 1990.
3. J. F. Jansen and J. N. Herndon, "Design of a Telerobotic Controller with Joint Torque Sensors Using 2-port Network Theory", Third International Symposium on Robotics and Manufacturing, Vancouver, British Columbia, Canada, July 18–20, 1990.
4. J. C. Rowe, P. L. Butler, R. L.Glassell, and J. N. Herndon, "The NASA Laboratory Telerobotic Manipulator Control System Architecture", Proceedings of the ANS Fourth Topical Meeting on Robotics and Remote Systems, Albuquerque, New Mexico, February 24–28, 1991.
5. R. L. Glassell, P. L. Butler, J. C. Rowe, and S. D. Zimmermann, "Custom Electronic Subsystems for the Laboratory Telerobotic Manipulator", Proceedings of the ANS Fourth Topical Meeting on Robotics and Remote Systems, Albuquerque, New Mexico, February 24–28, 1991.
6. polyFORTH II Reference Manual, FORTH Inc., Hermosa Beach, California
7. P. L. Butler, J. D. Allen, and D. W. Bouldin, "Design and Implementation of a Parallel Computer for Expert System Applications", SPIE Vol. 937 Applications of Artificial Intelligence VI, Orlando, Florida, April 4–8, 1988.
8. P. L. Butler, J. D. Allen, and D. W. Bouldin, "Parallel Architecture for OPS5", Proceedings of the Fifteenth International Symposium on Computer Architecture, Honolulu, Hawaii, May 30–June 2, 1988.

END

DATE FILMED

01/08/91