CONF-910223--2

CONF-910223--2

DE91 002880

# THE NASA LABORATORY TELEROBOTIC MANIPULATOR* CONTROL SYSTEM ARCHITECTURE

J. C. Rowe, P. L. Butler, R. L. Glassell, and J. N. Herndon
Oak Ridge National Laboratory†
Robotics and Process Systems Division
P.O. Box 2008
Oak Ridge, Tennessee 37831–6304



Paper 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 National Aeronautics and Space Administration, Langley Research Center.

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

# THE NASA LABORATORY TELEROBOTIC MANIPULATOR CONTROL SYSTEM ARCHITECTURE*

**J. C. ROWE, P. L. BUTLER, R. L. GLASSELL, and J. N. HERNDON**

*Oak Ridge National Laboratory*†
*Robotics and Process Systems Division*
*P.O. Box 2008*
*Oak Ridge, Tennessee 37831-6304*

## ABSTRACT

In support of the National Aeronautics and Space Administration (NASA) goals to increase the utilization of dexterous robotic systems in space, the Oak Ridge National Laboratory (ORNL) has developed the Laboratory Telerobotic Manipulator (LTM) system. It is a dexterous, dual-arm, force-reflecting teleoperator system with robotic features for NASA ground-based research. This paper describes the overall control system architecture, including both the hardware and software. The control system is a distributed, modular, and hierarchical design with flexible expansion capabilities for future enhancements of both the hardware and software.

## INTRODUCTION

The LTM system (see Fig. 1), sponsored by the Automation Technology Branch at NASA's Langley Research Center (LaRC), was developed at ORNL to provide a dexterous, dual-arm, force-reflecting teleoperator system with robotic features for NASA ground-based research. Because of the need for significant increases in extravehicular activity and the potential increase in hazards associated with space programs, emphasis is being heightened on telerobotic systems research and development. Telerobotics represents the first stage in the future development of a remotely maintainable orbiting space station utilizing teleoperated robots and eventually fully autonomous robots to free the astronaut of hazardous and repetitive extravehicular tasks. The LTM incorporates traction drives, modularity, redundant kinematics, and state-of-the-art hierarchical control techniques to form a basis for merging teleoperations and autonomous robotic operation into common hardware to further NASA's research.

---

* 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.

**FIG. 1. The LTM system.**

## CONTROL SYSTEM HARDWARE

The LTM control system features a distributed, modular, and hierarchical design with flexible expansion capabilities for future enhancements of both the hardware and software. A top-level block diagram illustrating the overall organization of the system is shown in Fig. 2. The system is composed of two VMEbus computer systems, one master and one slave, connected by a high-speed serial communication link to allow significant separation between the master and slave arms. Each VME rack controls a pair of the LTM arms utilizing data acquired from sensors in the individual arm joints. The sensor data acquisition [1] is provided by three joint processors (JPs) in each arm, one JP for each pitch/yaw joint (wrist roll and grip are handled by the wrist JP), and transferred in packets over high-speed serial links to the respective VME computer system. Each JP communicates with the VME computer system over a separate, full-duplex fiber optic link. The human-machine interface (HMI) consists of a Macintosh II computer tied directly to the master VME computer system and provides a graphics-based interface for operation of the system. A text-based interface is also provided through a VT-100 terminal attached to the system control processor (CP) debug port.

Figure 3 shows a board-level block diagram of the master VME computer system. The slave system is identical except for the disk controller card. There are three 68020 single-board computers in each system. The CP provides overall coordination and also services the human-machine interface and mass storage systems. The other two 68020 boards, or arm processors (APs), operating concurrently, complete the control algorithm calculations for each arm. Only two

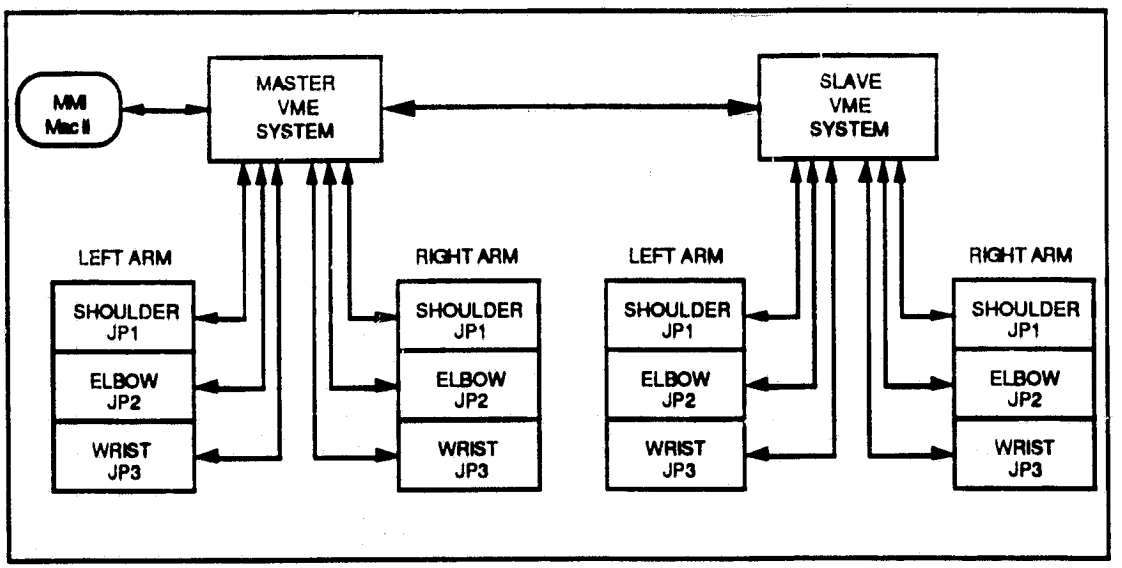


**FIG. 2. LTM block diagram.**

APs are shown, but this is considered a minimal system. Additional APs can be added to the system as computational requirements warrant it. The system controller (SC) provides additional functions to enhance the operation of the system such as a real-time battery-backed clock, full VME arbitration, global interrupts, and memory expansion. The other VME boards perform system I/O: the two link processor (LP) baseboards service the communication links to the JPs located in each arm, the Proteon communication boards provide the communication link to the alternate VME computer system, the D/A converters provide drive signals to the pulse-width modulated (PWM) amplifiers, and the disk controller board interfaces to the mass storage devices.

Custom electronics packages were developed to reduce the number of cables required for each arm [1]. Since the LTM utilizes an embedded cable approach with all of the power, control, and communication cables passing through the pitch/yaw joints, it was necessary to minimize the number of cables required. The custom systems reduce the required cabling by acquiring, processing, and multiplexing the many sensor signals over the serial communication links. The electronics packages consist of four individual systems: a JP logic board (JPl) in each joint, a JP power board (JPp) in each joint, an LP board for each joint to interface with the VMEbus, and the fiber optic communication system. The JPl board and the LP board are both high-density circuit boards utilizing surface-mount technology on both sides of a multilayer board.

The LP baseboard consists of a full-height VME baseboard with the three custom-designed, plug-in link interface modules (LPs) each containing an 80C196 with RAM and ROM memory. The baseboard contains the VME buffers, address generation logic, and the interrupt handlers for the individual link interface modules. The baseboard can be configured for standard memory mapped, extended memory mapped, or short I/O mapped addressing. It is configured in the LTM system as a standard 24-bit memory mapped address occupying 64 Kbytes of VME address space. The baseboard serves as an intelligent slave on the VME backplane. There are four memory mapped windows on the baseboard associated with each of the LPs. The interface modules each have 16 Kbytes of EPROM and 16 Kbytes of SRAM memory. In addition, each link interface module has 4 Kbytes of dual-port SRAM. The dual-port RAM is interfaced to the VME backplane through the

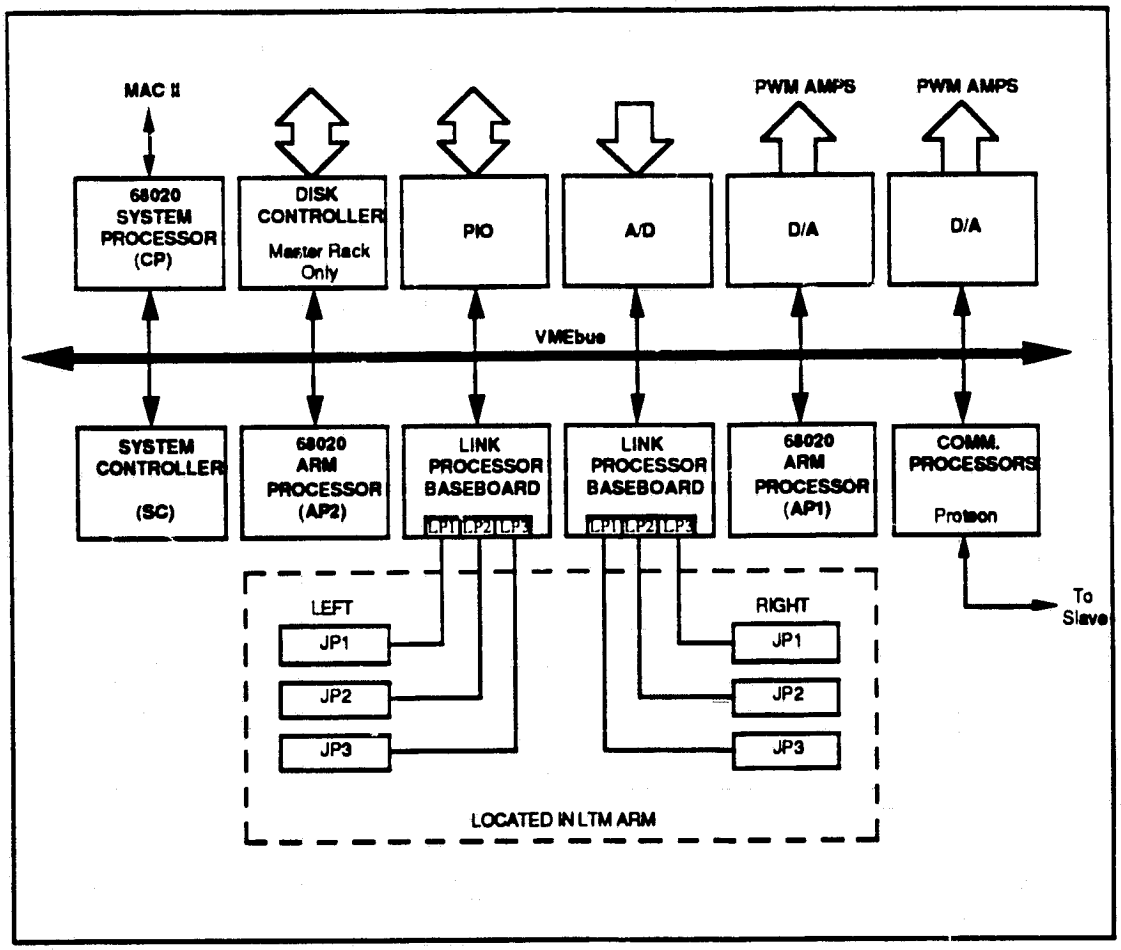


**FIG. 3. LTM board-level block diagram.**

baseboard and is utilized for system global memory as well as communication with the LPs and JPs.

The JPs are responsible for sensor data acquisition within the individual LTM joint modules [2]. Each JP communicates with the VME computer system by using a single, full-duplex fiber optic serial link with a maximum data rate of 2 Mbaud. Each JP module has several sensors from which it must gather data. The arm module and contained JP have two drive trains with motors, sensors, and brakes. Each motor/reducer has a position, velocity, temperature, and torque sensor. There are two other position encoders (resolvers) located on the traction drive differential output axes. The encoders are used to obtain position and velocity for both the pitch and yaw motions. Additional temperature sensors are used to monitor the JP temperature. Each motor also has a permanent magnet fail-safe brake over which the JP has individual control. In addition, the wrist JP handles the wrist roll sensors and motor as well as the grip controls. The main function of the JPs is to acquire the sensor data and transmit that data to the VME system. The data acquisition rate is asynchronous to the VME control system but is synchronized to 620 Hz by an internal LP interrupt clock.

## SOFTWARE

The LTM control software also supports a modular hierarchical design with expansion capabilities for future enhancements. It is based on past ORNL experiences in complex hierarchical manipulator systems [3] and the need to be consistent with the overall space station NASA/NBS Standard Reference Model (NASREM) control approach [4]. Therefore, a global or common memory concept is utilized for sharing of data between applications and processors in the system. Any process, current or future, can access control data directly through common memory thereby avoiding the direct linkage of code modules.

The OS-9 operating system was chosen to meet the design requirements for LTM. It is a modular, multiprogramming, multilingual, and multitasking operating system for the Motorola 68000 family of microprocessors. It has a hierarchical file structure similar to UNIX, but with an assembly language kernel for speed in real-time applications and for efficient memory usage. OS-9 provides a modular software development environment structured in a layered hierarchy of named modules that can be shared between applications. For completed applications, the operating system is ROMable, permitting operation without mass storage if desired.

All of the major application modules are written in the C language, including the control algorithms [5,6] in both master and slave VME computer systems, the global memory implementation, and other high-level routines. The only exceptions are the LP code, the JP code, and the application that interfaces to the LP/JP system, all of which are implemented in the FORTH language. FORTH was chosen for the embedded modules because of its small size, flexibility, and the extensive built-in debug capabilities.

The overall organization of software applications is illustrated in Fig. 4. At this level, the diagram is essentially a processor model with each bubble representing a processor in the system. As indicated on the bubbles, some represent multiple identical processors in order to keep the diagram readable and to represent the organization more clearly.

The Macintosh Application (1.1) runs on the Mac II processor and provides a graphics-based operator interface to the LTM control system. It is written entirely in Think C. The only inputs and outputs are the serial link to the VME system and the local monitor and input devices.

The highest level of control in the overall control hierarchy is provided by the System Control Processes (1.2) running on the first CPU in the master VME rack. In addition to overall coordination of lower levels in the system, the high-priority communication routines that provide data, commands, file transfer, and synchronization between the master and slave racks execute from a periodic clock interrupt on this processor. Some low-priority utilities also execute on this CPU. The system control processes interact with the arm control processes on other CPUs through common memory and through pipes to each processor. Low-priority functions, such as starting new tasks and copying files, are handled through the system pipes, but changing modes or gains on an arm processor are handled through the common memory. Through the common memory interface, a mode or even the full control algorithm can be changed from the control processor within one loop of the control algorithm. With the system pipes, the timing is much less deterministic. Interaction with the LP/JPs is also through common memory on the LP cards.

The primary system coordination task executes on the system control processor. It communicates commands and data to lower level rack control tasks that execute on

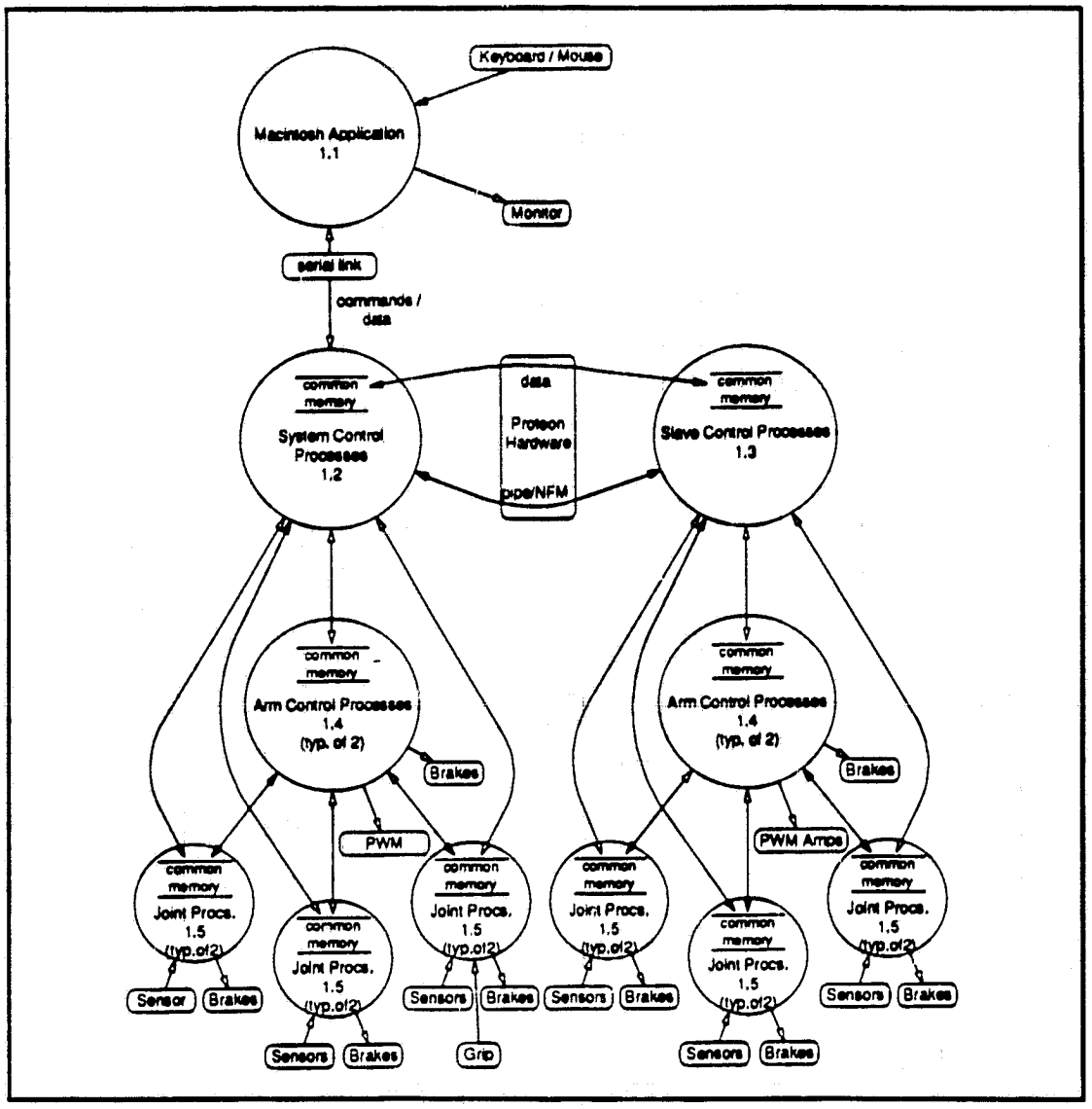


**FIG. 4. LTM software architecture.**

the system control processor in the master control rack and on the slave control processor in the slave control rack. This communication is through system pipes, allowing the interface to the two rack control tasks to be identical even though one resides in a separate rack. The rack control tasks on the master and slave racks are essentially identical and coordinate the arm control tasks executing on the balance of the CPU cards in each rack. The appropriate commands or functions are written to the AP's common memory, depending on the task to be completed. The use of common memory for direct control of the arm processor functions, in addition to the usual data transfer, leads to great flexibility in coordination of the control functions and will be described in more detail in the discussion of the arm control processes below.

The Slave Control Processes (1.3) are essentially that subset of the System Control Processes which control only the local (in this case, slave) system, and communication with the master rack. As described above, the subset is identical to that in the master rack, except for the path taken by the data on input and output to higher layers in the the control hierarchy. It is possible to make them identical

because the system pipes implemented through the Proteon communication link essentially make the master-slave communication link transparent to the software. There are some necessary differences between the lowest-level communications software on the slave and that on the master but they are handled through conditional compilation of the same source code.

The Arm Control Processes (1.4) execute on CPUs 2 and 3 in both master and slave VME racks, a total of four processors in all. The arm control calculations are completed on these processors with inputs from common memory and outputs to the PWM amplifiers. The Arm Control Processes interface to the local LP/JPs through the common memory on the individual LP cards and the interaction between the Arm Control Processes and the LP/JPs is completely asynchronous. All of the Arm Control Processes are written in the C Language.

The APs serve as function execution engines for the high-priority algorithms and hardware control code on the LTM system. The code is completely modular at the functional level (i.e., functions in memory on the processor can be dynamically inserted or removed from the control loop while it is executing in order to alter system operation). This is done through insertion or removal of function vectors in an execution table in the common block, thus providing great flexibility in the configuration of the control loops. For example, the data input function could be replaced with a function that reads data from a simulation rather than from the actual joints without altering the rest of the control calculations. Functions can be inserted in the table on a joint by joint basis (i.e., the control algorithm for one joint can be changed independently of those for other joints and the functions can be changed at the loop rate if necessary). For functions that are not required to complete every loop, such as electronic counterbalance calculations, the algorithms can be split into several sequential functions that spread the required calculations over a number of loops thus requiring minimal time within a single loop–each function completes a portion of the calculation, then enters the address of the next function in the required sequence into the execution table.

The Joint Processes (1.5) execute on the LP and JP pairs associated with each joint–three for each arm, for a total of twelve in all. Because of limitations in the VME baseboard into which the LPs are installed, the LP processes cannot access VME memory. Therefore, the only interface to the rest of the system is through the portion of common memory residing on the individual LPs. All of the software for the LPs and JPs is written in the FORTH language.

All except the very highest level (lowest speed) communication for system coordination, development-related communications, and file transfer occur through common memory. Common memory is not in a single block, but is distributed throughout the VME system. It is common in that any VME control processor has access to the data over the bus even though it may not reside locally on that processor. LPs can only access their own local memory, but they are implemented as data acquisition systems that run asynchronously to the rest of the LTM system so there is no need for them to access the other common memory. They update the data in their memory as fast as possible and the rest of the system reads it whenever required. Allocation of common memory was based on keeping bus traffic to a minimum and keeping access speed high where it was critical. Therefore, portions of common memory that are critical to a process are kept local to the processor where that process executes. All of the critical joint data are maintained in the common memory of both the master and slave racks, with the transfer occurring at the highest available loop rate, but data not needing to be updated at the loop rate are not maintained in both racks. Status and errors are returned to the master rack and HMI through system pipes at lower speeds.

## CONCLUSION

In summary, the LTM control architecture is a modular and flexible architecture in which a list of control functions can simply be written to a table for execution at the loop rate. The list of control functions can vary from loop to loop as necessary. It is a hierarchical architecture in which APs receive commands and data from the higher level coordination processes and data from the lower level processes on individual joints. The LTM is currently at NASA's Langley Research Center where it is being used for testing to resolve teleoperation and robotics issues for space.

## REFERENCES


1. R. L. Glassell, et al., "Custom Electronic Subsystems for the Laboratory Telerobotic Manipulator", Proc. of the ANS Fourth Topical Meeting on Robotics and Remote Systems, Albuquerque, New Mexico, February 24–28, 1991.

2. P. L. Butler, R. L. Glassell, and J. C. Rowe, "A Distributed Data Acquisition Software Scheme for the Laboratory Telerobotic Manipulator", Proc. of the ANS Fourth Topical Meeting on Robotics and Remote Systems, Albuquerque, New Mexico, February 24–28, 1991.

3. J. C. Rowe, R. F. Spille, and S. D. Zimmermann, "Integrated Digital Control and Man-Machine Interface for Complex Remote Handling Systems", Proc. of the International Topical Meeting on Remote Systems and Robotics in Hostile Environments, Pasco, Washington, March 29–April 2, 1987.

4. R. Lumia, J. Fiala, and A. Wavering, "NASREM: Robot Control System and Testbed", Robotics and Manufacturing (ASME Press, 1988).

5. J. Jansen and J. N. Herndon, "Design of a Telerobotic Controller with Joint Torque Sensors", IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, 1990.

6. J. 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.