Proceedings of the Second International Symposium on Robotics and Manufacturing Research, Education and Applications, Albuquerque, NM , November 16-18, 1988.

CONTROL SYSTEM ARCHITECTURE FOR THE TEAM PROGRAM

Sandor Szabo, Harry A. Scott, Roger D. Kilmer
National Bureau of Standards, Robot Systems Division

ABSTRACT

The U.S. Army Laboratory Command is developing a testbed for cooperative, real-time control of multiple land vehicles. The system requires the development and integration of many elements which allow the vehicles to perform autonomously and under supervisory control. The National Bureau of Standards is supporting the program by developing a control architecture based on experience gained with hierarchical control systems in robotics and automated manufacturing. The paper starts with a high level presentation of the program and the background of the hierarchical control concepts. A review of the design methodology, including an example task decomposition follows.

INTRODUCTION

The U.S. Army Laboratory Command is undertaking a program designed to demonstrate cooperative, real-time control of multiple land vehicles. This program is titled TEAM, an acronym for Tech-base Enhancement for Autonomous Machines. The title TEAM is appropriate in describing both the cooperative nature of operations of this system and the organizational structure of the program. There are eight U.S. Army organizations, two national laboratories and currently, three contractors involved in this three year program. They are:

Human Engineering Laboratory [HEL]
Harry Diamond Laboratory [HDL]
Ballistic Research Laboratory [BRL]
Material Technology Laboratory [MTL]
Engineering Technology and Devices Laboratory [ETDL]
U.S. Armor School
Tank Automotive Command [TACOM]
Armament R&D Command

National Bureau of Standards [NBS]
Oak Ridge National Laboratory [ORNL]

Kaman Scientific
Digital Signal Corporation
Honeywell Advanced Systems Center

TEAM involves the teleoperated control of two land vehicles, called Robotic Combat Vehicles (RCV), from a remote command center (RCC). The function of this system is to provide remote operation of military vehicles equipped with any of a variety of reconnaissance or related mission packages. The program is focused on

This work was funded by the U.S. Army Human Engineering Laboratory. This paper was prepared by U.S. Government employees and is not subject to copyright.

several key technical areas including communications between the RCC and the vehicles, video feedback requirements and limitations for teleoperated driving, semi-autonomous retro-traverse navigation of a learned-path, automatic target detection and tracking, and multi-vehicle command and control.

The RCC will contain the operator driving controls and displays, mission package controls and displays, communications equipment and a high-level control system for coordinating all of this equipment. The RCC will exist in two versions: an interim version, the IRCC, that will be used for early system development, integration and testing; and a final version, the RCC, being developed under contract for the U.S. Army Tank Automotive Command. The TEAM vehicles are being designed for operation from both the IRCC and the RCC.

The vehicle chosen for the program is the High Mobility Multi-purpose Wheeled Vehicle, or HMMWV (usually pronounced hum-vee). The two HMMWVs will be equipped with remote driving controls (actuators and status monitoring sensors), a mission package which includes automatic target detection and tracking, communications equipment and a high-level control system for coordinating all subsystems.

NBS's role in TEAM is to apply it's hierarchically-structured real-time sensor-based control system architecture, originally developed for control of industrial robots, to the design of the supervisory control systems in the two vehicles and the RCC. HEL chose this control system approach because of its potential to become a standard control system architecture for all Army robotics. This parallels a decision made by NASA to adopt this architecture for the Space Station Flight Tele-robotic Servicer control system [1].

This paper presents a brief description of the NBS Real-time Control System (RCS) architecture and discusses its role in the TEAM program. The next section presents a high-level description of the general RCS architecture. This is followed by a description of the design process followed for TEAM, including a description of how the RCS architecture is applied. Then, the communications issues arising from the TEAM mission performance functions and from the decision to use the RCS approach are presented. Described last is a brief summary and status report on the design effort.

HIGH LEVEL REVIEW OF RCS

Underlying development of an RCS is a collection of analysis, design, and implementation methodologies for the development of real-time control systems. The RCS analysis and design techniques discussed here are used to produce a model based upon a standard reference architecture for intelligent machines. The architecture is described in more detail in reference [2]. A summary of the architecture is presented in the remaining portion of this section to provide sufficient background to the reader.

Standard System Architecture for Intelligent Machines

An intelligent machine utilizes computers to control a collection of mechanical devices which allow it to physically interact with the environment. The control system uses sensory information to guide the machine in the execution of complex tasks. In the RCS architecture, complex tasks are viewed hierarchically with motor skill functions performed at the lowest levels and coordinated interaction between machines performed at the highest levels. Such an architecture provides the organization and structure required to effectively integrate the machine's components.

Figure 1 illustrates the RCS system architecture. An important attribute of the architecture, which is evident throughout, is the drive to standardize the controller components. For example, each of the horizontal levels in an RCS consists of H, M, and G modules corresponding respectively to Task Decomposition, World Modeling and Sensor Processing classes of functions. Each of the levels' H modules decompose input tasks (commands from a higher level) into a temporal set of output tasks (commands to lower levels) which allow the machine to exhibit the desired behavior.

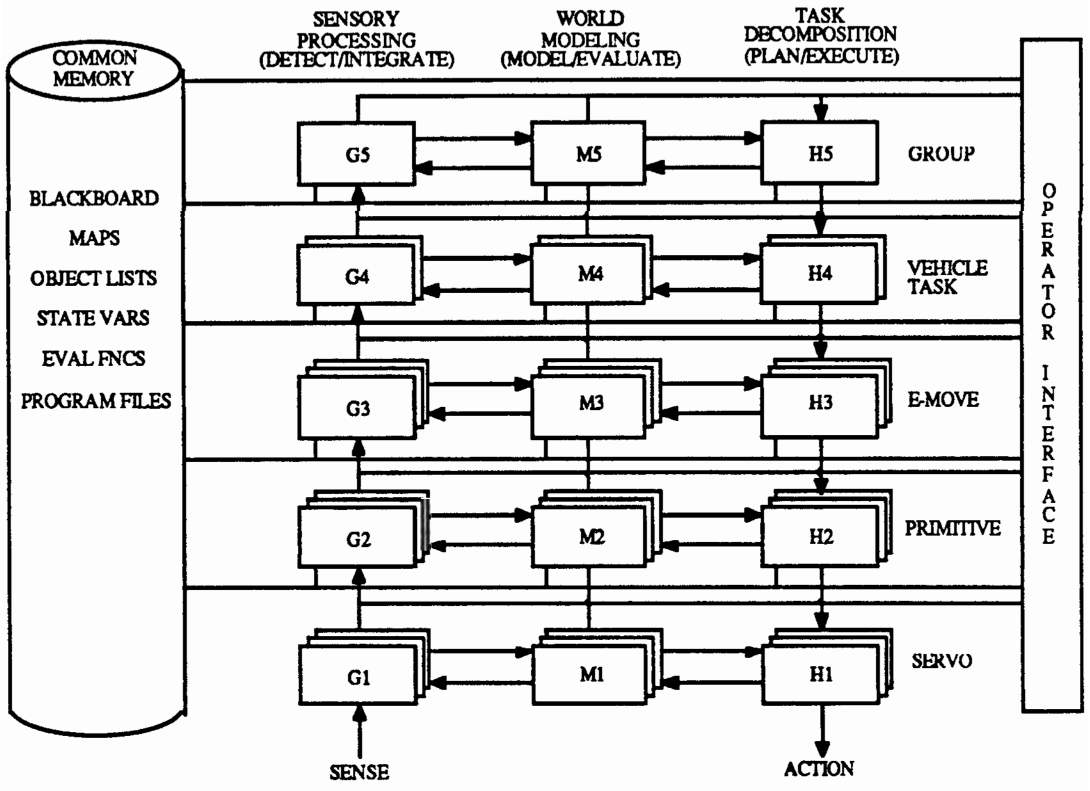

FIG. 1 RCS Systems Architecture for TEAM Program.

Another example of RCS standardization is seen in the structuring of the control loop at each level, i.e., the organization of each loop, when possible, is identical. Each loop functions as a state machine where first all inputs to the level are examined. Then, the current state of the system is analyzed and the appropriate transformation function is applied. Finally, all outputs are transmitted. Each of the modules (H, G, M) contribute to this input-process-output control cycle for each horizontal level in the system.

A significant attribute of the system architecture is that it enforces a clear partitioning of the classes of control functions. The partitioning is handled through interface definition between system components and formal specification of internal functionality. The benefits of such modularism include: improved understandability, reuseability of code, eased system integration, the support of upgrades and modifications, and efficient distribution of programming efforts.

An important consideration in an RCS design is the integration of the role of humans in control. While the technology of machine intelligence is in its infancy, human dexterity, knowledge and expertise can greatly increase the usefulness of most machines. The smooth integration of human control and the transparency with which control is exchanged are critical to the success of systems using such shared control. The interface for an operator is shown in figure 1 along the right hand side. Modeling of the operator interface itself using the RCS methodology is being explored.

The final component of the RCS architecture is Common Memory (left side of figure 1). Common memory serves as a blackboard where information is posted and readily available to any process in the system. The data structures, such as maps, plans, object models also reside in common memory. This form of memory is essential whenever information is shared between processes.

Levels of the RCS Control Hierarchy

The RCS architecture is intended as a structure for the implementation of autonomous control functions. For any such function, there exists a spot in the architecture where it logically resides. When low level functions are complete, they become available for combination into higher level functions [3]. The modularity of the functions makes them reusable, increasing the overall flexibility. The classes of functions performed at each level are standardized, from the bottom up, as follows:

Servo Level - Performs the lowest level servo control functions of devices such as a robot joint actuator. Low level servos provide control updates quite often (typically in milliseconds) and because of the timing constraints of software, a large portion of the control may be implemented in hardware. Additional responsibilities at the Servo Level include transformations between cartesian and joint space, joint interpolated trajectories, and limit checking for commanded joint positions, velocities and accelerations.

Primitive Level - Performs the next level of control functions above the Servo Level. In general, the Primitive Level class of functions deals with tasks requiring fine, controlled motion of a machine through cartesian space. Such motion, also called a trajectory, is defined as the position or velocity of a machine as it travels through a series of points in space within a time interval. Typical trajectories include straight-line motion, but more complex ones can be generated dynamically at the Primitive Level as a result of sensed forces or loads. An example of the commanded inputs to the Primitive Level are the cartesian goal and the type of trajectory to be used for a particular motion.

Elemental Level - The cartesian goals for the Primitive Level, also called key frames, are generated by the Elemental Level (a.k.a. E-move Level). A robust level will make use of sensory information provided by cameras, laser scanners, acoustic transducers or other sensors to automatically generate the cartesian goals and to modify them when more information becomes available. The goals should be clear of obstacles and provide optimum traversability to simplify Primitive Level trajectory generation. Such complex processing uses a much longer planning horizon then is evident at the lower levels. This leads to the requirement for asynchronous processing between control levels, a function supported by the RCS.

Task Level - Incorporates the highest level of control for a single machine often involving the planning and coordination of several Elemental Level subsystems. At this level begins the type of processing associated with AI concepts such as expert systems. Complex data structures, including terrain maps, plan trees, and object models, are required to support this processing. The level also serves to limit the space the Elemental Level must search to generate key frames.

Group Level - Performs processing similar to the Task Level, but the Group Level is concerned with the planning and coordination of multiple intelligent machines. This level determines what tasks must be performed, when they are to be scheduled, how machines interact and how global knowledge from different machines is merged.

HIGH LEVEL REVIEW OF TEAM DESIGN

At this point in the program the design effort has concentrated on identifying the key functions to be demonstrated, developing a scenario, and filling in the architecture with an initial breakdown of the functions. This section briefly reviews the outputs from each step. Discussion of the design process provides insight into how an RCS is created for a novel application.

Mission Performance Functions

The first step in the development of the TEAM concept was to identify the functions that would enhance the state of autonomous machines. The functions fall into three categories: fully automatic, remotely supervised and remotely controlled. Automatic functions can be performed without operator intervention. An example of an automatic function envisioned for TEAM is the ability to acquire and track tank-like targets. An example of a remote supervision function, which requires operator control but makes use of certain autonomous functions, is the ability to follow a previously traversed path. This function, called Retro-traverse, requires the operator to drive the RCV via teleoperation while the onboard controller records the path. Once recorded, the RCV can automatically traverse the path back to a previous point. With the Retro-traverse function, the RCV has the ability to acquire targets and reposition without assistance from the operator. The remote functions address one-to-one operations between operator controls and remote device such as the master steering controls used in the RCC to drive an RCV.

Key to the success of the teleoperation driving functions are work in video compression and the overall reduction of required communications bandwidth. Progress in these areas addresses the problems of the anticipated limited bandwidth available in the battle environment and the problems of non-line-of-sight communications paths resulting from foliage and terrain characteristics.

Mission Scenario

One possible mission scenario is shown in figure 2. Two RCVs (A, B) are teleoperated by drivers in the RCC. A mission is planned by the commander to set up surveillance along the expected route of enemy targets. Vehicle A travels along the path marked {a1, a2, a3, a4}. Due to loss of line-of-site, path segments from a1 to a3 may require low data rate communication. At location a3, a vantage point is recorded as a fall back position. The driver then drives the RCV to location a4 (length of path segment not to exceed 150 meters), while the onboard RCV controller records the path. At location a4, the RCV is placed in a surveillance mode until targets are acquired. In one scenario, the target information is displayed to the operator. The operator can process the targets and request the RCV to retro-traverse back to location a3. The RCV autonomously follows the prerecorded path (speed not to exceed 5 km/hr) back to location a3 where additional targets can be acquired.

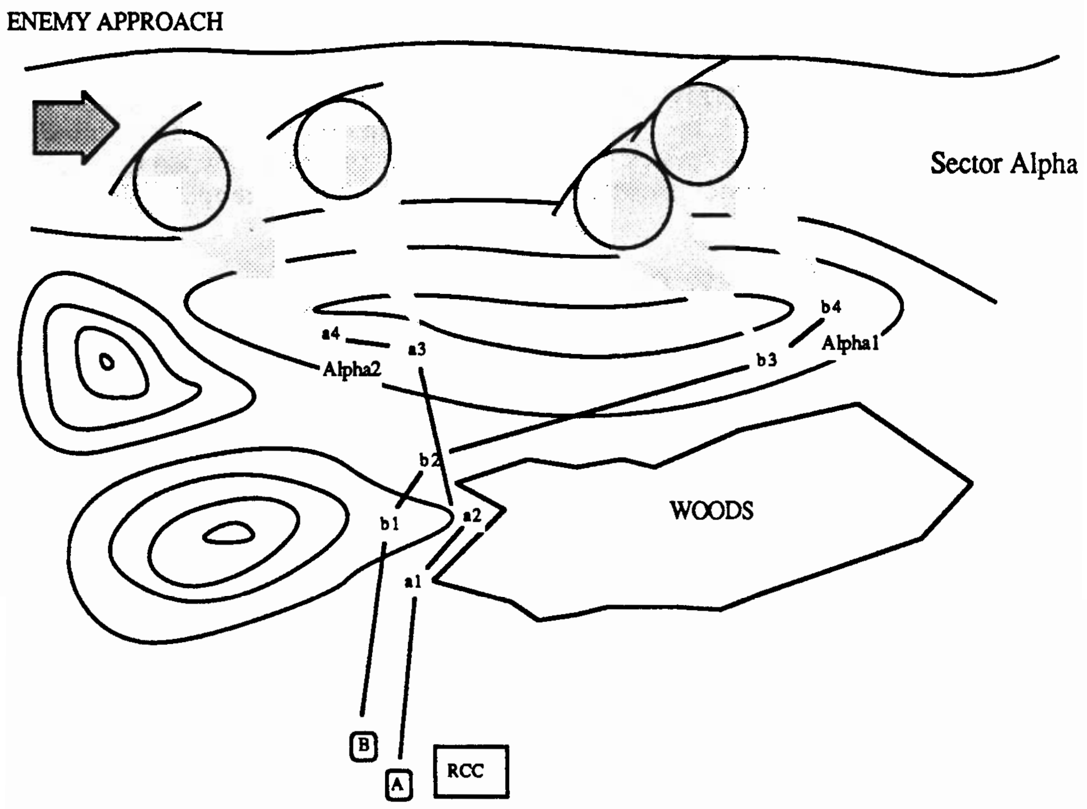

Figure 2. Proposed Mission Scenario for Two RCV's A and B.

Other aspects of the scenario require the RCV to act cooperatively in the search for targets. Areas of interest are established to define the desired actions to be taken on acquired targets. A mode of cooperative action allows targets acquired by one vehicle in an area-of-interest to be handed off to another vehicle

Functional Flow Diagrams

Functional flow diagrams are used to document each of the scenario steps. This analysis helps to identify all functions that are to be developed. As the analysis becomes more extensive, the diagrams can be simplified by leveling into a hierarchy. Figure 3 shows the top two levels for the TEAM RCC and RCV. Level One shows that the phases of a TEAM mission are primarily composed of preparation, deployment and aquisition. In Level Two, the preparation phase is broken down into powerup, initialization and planning for the mission. The deployment phase consists of driving the vehicles to various locations. The deployment can require several different modes of driving, but at this level they are all grouped together. The aquisition phase consists of automatic target acquisition (ATA) and tracking of targets. The partitioning of these functions provides flexibility to handle different scenarios. The hierarchical nature of the analysis also allows it to naturally map into the RCS architecture.

Task Decomposition

The functions derived from the functional flow analysis define actions that the operator and the TEAM vehicles must perform. These actions become tasks for the RCV and RCC control systems. It is essential to model these tasks to ensure that the behavior of the controller reflects the desired actions. The RCS methodology initially supports this modeling by defining how tasks logically fit into the control hierarchy. This stage marks the transition to an RCS implementation. The designers take advantage of the implementation characteristics of the RCS in developing the tasks at each level. In this manner, tasks between vehicles are implemented at the Group Level, tasks requiring coordination of subsystems within a vehicle are implemented at the Task Level, generation of "key frames" takes place at the Elemental Level, smooth trajectory generation is performed by the Primitive Level and low level motor skills are implemented at the Servo Level. The example in the next section illustrates the task decomposition for an RCV deployment.

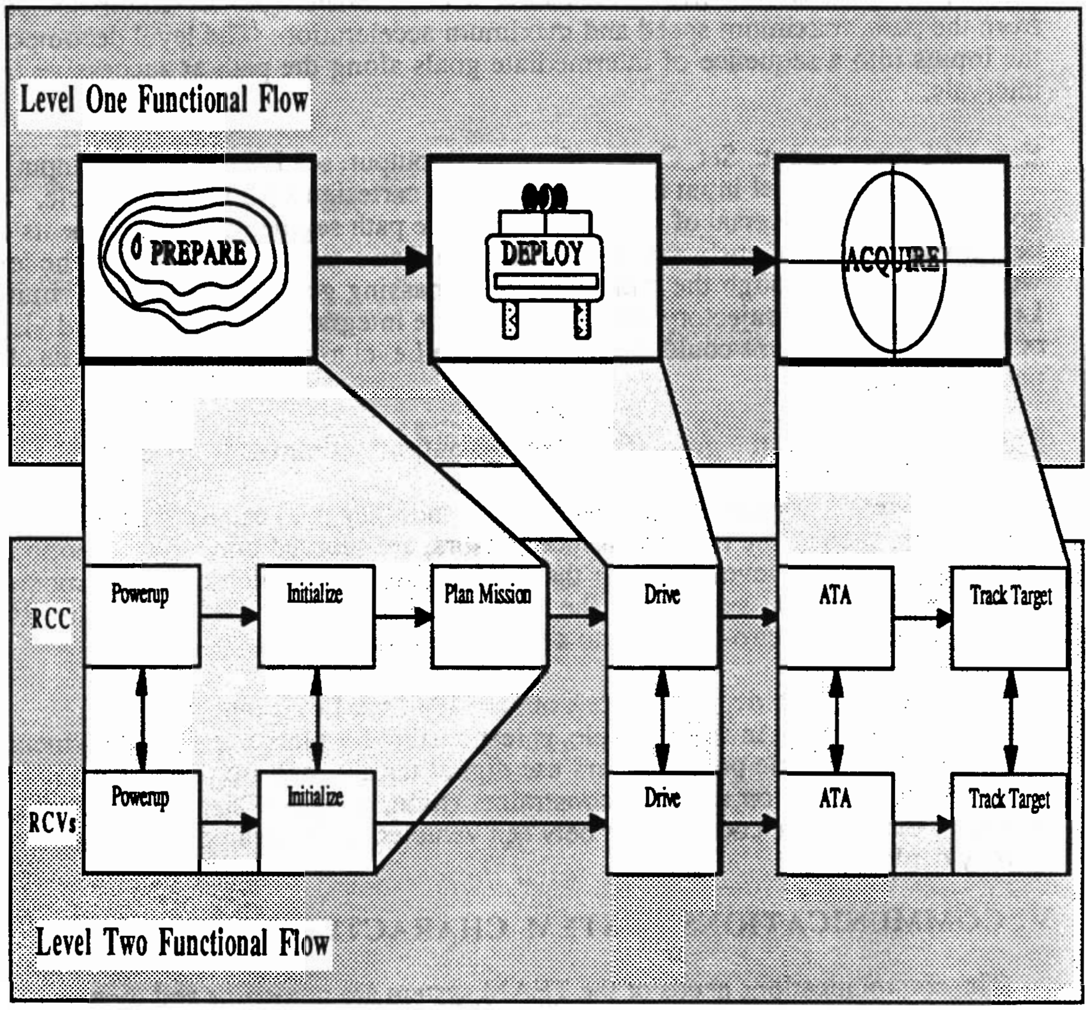

Figure 3. Level One and Two of Functional Flow Diagrams.

RCS HIEARCHICAL TASK DECOMPOSITION

A TEAM mobility function for RCV's involves several modes of driving. A characteristic of the driving modes is that they share similar functions. The RCS methodology, through the use of task decomposition, effectively takes advantage of this characteristic. The following example shows how the RCS architecture can be applied to implement the driving function for the TEAM vehicle. It is limited to a description of the command interfaces between the levels of control and a brief synopsis of the internal decomposition.

Servo Level - [input: set_velocity(vector); output: actuator voltages]
The Servo Level input is the desired velocity of the vehicle during a discreet time interval. The velocity is decomposed into a value representing the desired steering angle and throttle position for the next time interval and output to the drivers.

Primitive Level - [input: Set_Heading(goal, trajectory); output: see Servo Level input]
The Primitive Level input is the desired cartesian goal for the vehicle and the trajectory it should follow. The goal consists of the position and orientation (i.e., pose) of the vehicle at the termination of the trajectory. The trajectory is a straight-line path, ideally free of obstacles, with parameters for maximum deviation from the path, maximum speed and maximum acceleration. The level decomposes the inputs into a sequence of intermediate goals along the path at successive time intervals.

E-move Level - [input: Set_Course (final goal); output: see Primitive Level input]
The E-move Level input is the desired final cartesian goal for the vehicle. The goal may require traversal of several straight-line path segments. The segments can be derived from previously saved paths (part of Retro-traverse function). The level would then step through the paths segments, passing goals poses to the Primitive Level for automatic trajectory generation. Future integration of sophisticated vision or laser scanner systems could allow the E-move Level to dynamically calculate clear path segments.

Task Level - [input: goto (final goal); output: commands to each E-move subsystem]
The current Task Level responsibility for mobility is to ensure that each of the subsystems, such as the reconnaissance sensors, are secured prior to any motion of the vehicle. Higher level tasks for the level includes the combination of mobility with other functions, such as in the scenario described earlier where the RCV engages targets and retro-traverses to a fall back position.

Group Level - [input: deploy (sector); output: see Task Level input]
The Group Level, in this example, is responsible for planning the deployment of the RCV's. Automatic path planners use digital terrain maps to automatically plan optimal paths based on relevant constraints (time, distance, stealth, etc.). Some strategies may require coordination between vehicles, as for example, execution of a convoy deployment.

V. COMMUNICATIONS SYSTEM CHARACTERISTICS

The communications needs of the TEAM system are numerous and diverse. The particular mechanisms to be used must address such requirements as compatibility with the RCS architecture, use of particular frequency bands for RCC-RCV communications to minimize line-of-sight considerations and the ability to support low data rate compressed video transmissions for operator feedback.

Communication System Responsibilities

The various modules of the RCS system architecture (see figure 1) have clearly defined interfaces among which information flows. The vertical interconnecting lines represent paths over which commands flow down the hierarchy and command status information flows up. The horizontal lines represent information flowing within a level and typically require more bandwidth than the vertical paths. Horizontal lines also represent links to external information sources such as the blackboard and user interfaces. Implementation of all of the communications paths in the TEAM control system requires several different communications mechanisms.

At the highest level, communication must be supported between two physically separate sites, the RCC and the RCV. Within the RCC and RCV, various computer subsystems must communicate. These subsystems are to be implemented on independent computer systems with separate backplanes. Within a particular subsystem, communications are needed between processors sharing the backplane. Finally, process-to-process communications is required for processes executing on the same processor. Physically, communications are required across backplanes, through networks, and via point-to point links.

The communications system must, in addition to meeting overall speed requirements, be able to provide certain communications paths with predictable performance. Such a time critical link will be essential for teleoperation driving and possibly for other functions.

Another communications system responsibility is to handle changes in the communications path configuration during system execution. An example of this is illustrated by the use of a different command path for teleoperation than is used in execution of a high level command such as Retro-traverse. In the more autonomous case of Retro-traverse, the Servo Level of the Mobility Module of the RCV receives commands from the Primitive Level immediately above it in the RCS hierarchy. The command's path is simply Primitive to Servo. In the teleoperation mode, the command does not come from the Primitive Level, but from a joystick in the RCC, clearly a different communications path.

VI SUMMARY

The TEAM project requires a systems architecture which will fulfill its objective as a testbed for autonomous machines. The RCS architecture described in this paper provides a uniform structure for integration of system components and development of system functions. To date, the initial components and functions have been identified. A scenario that demonstrates the performance of the TEAM system has been developed. A first step of the design, the examination of the functions via functional flow and task decomposition has also been completed.

REFERENCES

[1] J.S. Albus, H.G. McCain, and R. Lumia, "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NBS Technical Report 1235, July 1986.
[2] A.J. Barbera, J.S. Albus, M.L. Fitzgerald, and L.S. Haynes, "RCS: The NBS Real-Time Control System", Robots 8 Conference and Exposition, Detroit, MI., June 1984.
[3] Harry G. McCain, Roger D.Kilmer, Sandor Szabo, Azizollah Abrishamian, "A Hierarchically Controlled Autonomous Robot for Heavy Payload Military Field Applications", Proceedings of the Intelligent Autonomous Systems Conference, Amsterdam, The Netherlands, December 8-11, 1986.