CONCEPTS FOR A REAL-TIME SENSORY-INTERACTIVE CONTROL SYSTEM ARCHITECTURE

Anthony J. Barbera, M. L. Fitzgerald, and J. S. Albus

Industrial Systems Division, National Bureau of Standards
Washington, D. C. 20234

Abstract

This paper describes concepts used in defining an architecture for a real-time sensory-interactive control system. These concepts were arrived at from testing and evaluating different control system strategies at the National Bureau of Standards. A hierarchical task decomposition architecture has been used to structure the complex information processing for real-time sensory interactive robot control in a manageable form. This structure consists of a number of generic control levels. The task of a generic control level is to sample its input state and generate an appropriate response output state which results in a partial decompostion of its task command. Sensory feedback is provided by a processing structure of modules that are coupled with the appropriate control levels. The requirement that the system must be designed for ease of human comprehension has lead to an implementation using a state-table processing structure. Real-time response results from a multiple processor implementation using synchronized communications through a common memory.

Introduction

A control system must decide on the actions necessary to accomplish a goal according to rules which have been previously programmed. In addition, there must be a way of measuring the environment so that the actions can be modified to ensure that the goal is accomplished even though the environment is changing. Background information of control systems may be obtained in (1). (Figure 1)

Real-time sensory-interactive control must decide the output actions based on both the command goal and the sensory data that measures the state of the environment. In addition, the results must be output in a short enough time to ensure an effective and stable response.

Due to the complexity of real-time sensory-interactive control, the system must be structured in its design and implementation to keep it comprehensible. The structure must take into consideration human limitations in the management of information (2). These limitations relate to the apparent inability of people to easily manage more than seven pieces of information at any time (3). To deal with this limitation, the system is structured into well organized interconnected groups of component parts, each processing a small set of information and each having clearly defined interfaces. This modularization of the system gives the designer a clear understanding of the system at any level of detail. This clarity enhances the designer's ability to efficiently implement a system that is correct, accomplishes the designer's intent, and is easy to maintain and extend. The applicability of this technique is evidenced in the widespread use of structured programming (4,5,6).

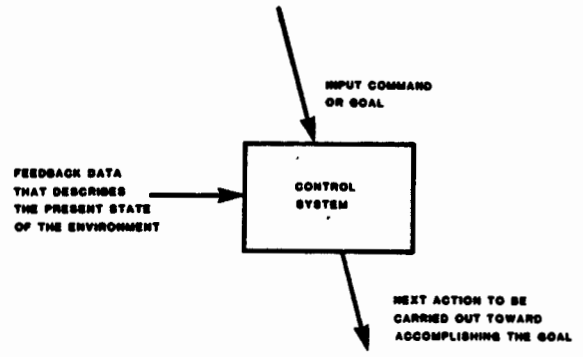
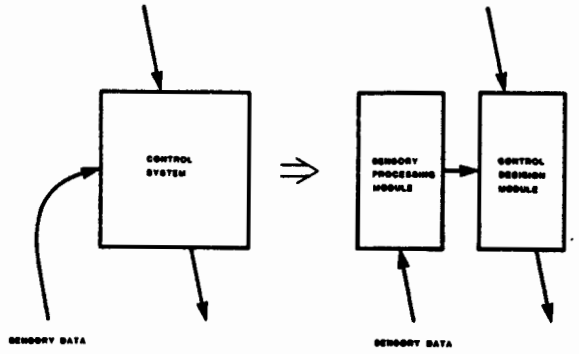


Figure 1: CONTROL SYSTEM INPUTS AND OUTPUTS

A Control Structure

To accomplish complete control for a complex system through one level of processing results in a degree of complexity that is very difficult for people to understand. The first step in the simplification of the processing is to partition the system into two major component modules - sensory processing and control decision. This section describes the control decision module while the following section will show how the sensory processing module can be linked with the control decision module. (Figure 2)

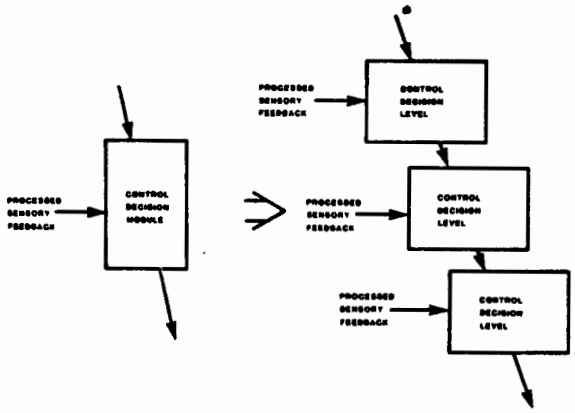
Reprinted from PROCEEDINGS OF THE FOURTEENTH SOUTHEASTERN SYMPOSIUM ON SYSTEM THEORY, April 1982



THE CONTROL SYSTEM IS PARTITIONED INTO SENSORY PROCESSING AND CONTROL DECISION MODULES.

Figure 2: TWO MAJOR CONTROL SYSTEM MODULES

The control decision module can be decomposed into a number of simpler control levels. Each level represents a well-defined clearly bounded control function with a small number of inputs and a limited set of outputs. As a result of this modularization, the designer only deals with subsets of the total information processing at each stage in the implementation. The amounts of information that are handled are always kept within the limits of comprehension and thus the chance of programming errors is reduced. (Figure 3)

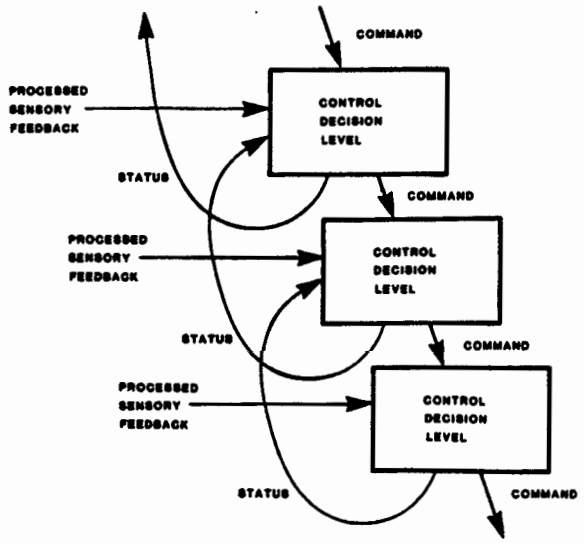


THE CONTROL DECISION MODULE IS PARTITIONED INTO A NUMBER OF SIMPLER CONTROL LEVELS.

Figure 3: CONTROL-DECISION MODULARIZATION

The concurrent operation of these control levels provider a stepwise decomposition of high level tasks into successively simpler and simpler component subtasks. By only requiring each level to decompose the task a little further into the next lower set of subtasks, it is relatively simple to comprehend and manage the control function of each level.

The use of a multi-level control system requires an additional type of feedback to each level. This is status information from the level below reporting how well the command to that lower level is being carried out. Thus, each level will send status to the level above and receive status from the level below. (Figure 4)

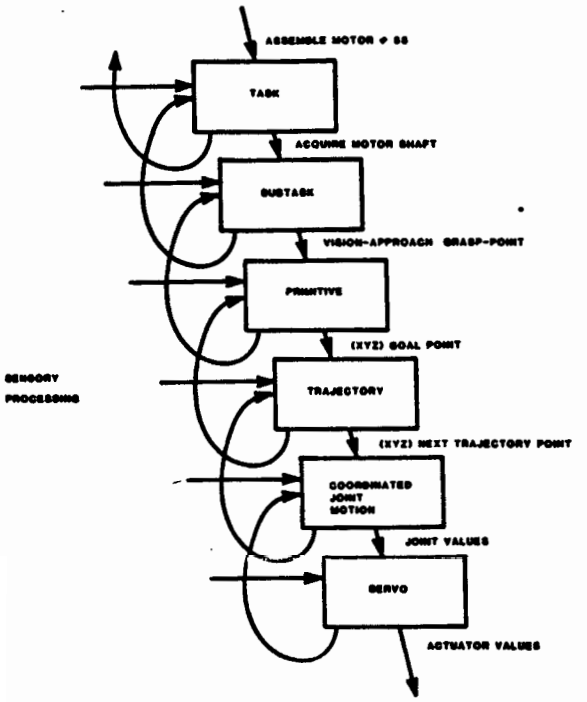


EACH LEVEL REQUIRES FEEDBACK STATUS FROM THE LEVEL BELOW REPORTING ON THE PROGRESS OF THE COMMAND SENT TO THAT LEVEL.

Figure 4: STATUS FEEDBACK

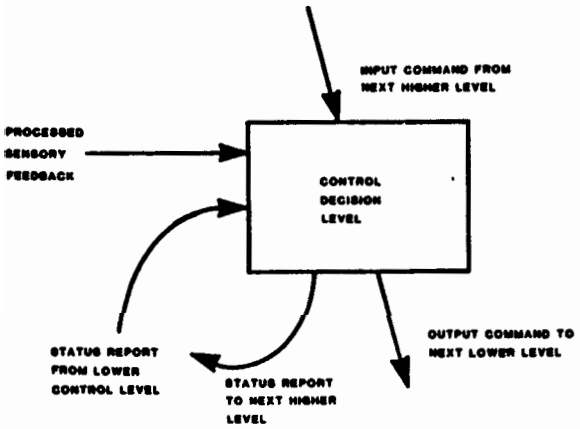
The designer determines the function that each level of the control system should perform by studying a logical flow of control and information processing through a number of example task decompositions. This procedure results in the specification of both the functions of the control levels and the information, structure and data format of their interfaces. (Figure 5)

A control decision level can be represented by a generic control structure. This structure performs a partial decomposition of an input command into a set of simpler commands. At each instant, the particular output (command and status) will be determined by the input command and the feedback data processed to abstract data about the state of the environment relevant to this level. The output command becomes the input command to the next lower level. Each of these levels of control receives an input command and uses real-time feedback data, both sensory and status to generate an output command and status that is a function of this input data. (Figure 6)



THE ARCHITECTURE FOR THE ILLUSTRATED SIX LEVEL CONTROL STRUCTURE WAS DESIGNED BY CONSIDERING PARTICULAR REQUIREMENTS OF THE HARDWARE FOR LOWER LEVELS AND EXAMPLE TASK DECOMPOSITIONS FOR THE HIGHER LEVELS.

Figure 5: EXAMPLE CONTROL LEVEL FUNCTIONS



EACH CONTROL DECISION LEVEL CAN BE REPRESENTED BY THE ABOVE ILLUSTRATED GENERIC CONTROL LEVEL.

Figure 6: GENERIC CONTROL LEVEL

To summarize, the control decision module has been structured as a number of simple control decision levels with well defined inputs and outputs. These levels decompose a task into simpler and simpler subtasks. In the case of robot control, the outputs at the bottom level are the drive signals to the actuators of the robot. Each level has a narrowly defined control capability that results in a clear identification of the type of sensory processing required at each level, the type of status feedback necessary, and the kind of output commands and status reporting that should be generated.

Coupling Sensory Data with the Control Structure

Because of the decomposition performed by the multi-level control structure described above, each level has different requirements as to the type and level of processed sensory data that is needed for its particular decision making process. The control structure, therefore, helps to determine what information must be supplied by the sensory processing structure for each control level to decide its next output.

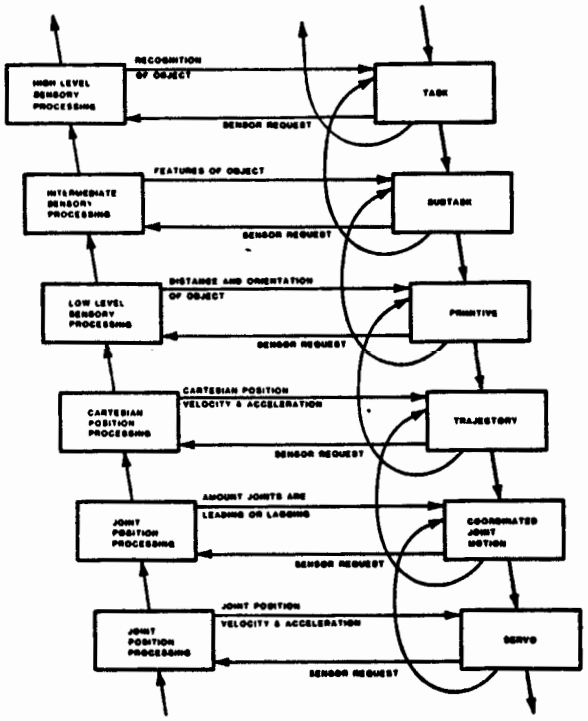
A structure that associates a corresponding level in a sensory processing hierarchy with each level in the control structure will enable the system to provide data relevant to the decision making process at each of those levels. Raw sensory data goes through a processing system that abstracts the required information to be used in the decisions that are made at each level of the control decision module. In general, the higher levels of control require more highly processed sensory data.

This sensory data can also include information from sensor systems that might detect error or emergency situations. Because of the well delineated levels of control, it is easy for the designer to determine at which level that information should be handled to provide the proper corrective action within the critical response time.

This sensory-interactive control architecture, therefore, consists of linked control decision and sensory processing structures (7). In the control decision structure, a set of control decision levels perform a hierarchical task decomposition based on feedback from the sensory processing structure. This sensory processing structure is a corresponding set of levels that interact with the control levels to provide the degree of processed sensory data necessary for that particular control level to make the decisions required. (Figure 7)

In summary, the basic control decision level generates three types of outputs as a function of three types of inputs. The inputs are: 1) an input command task, 2) sensory processing data from the corresponding sensory processing level, and 3) status information from the control decision level below. As a result of these inputs, the control decision level will generate the following

outputs: 1) a control command to the level below, 2) a status report to the next level above, and 3) a request to the corresponding sensory processing level to indicate what type of sensory information is required at this time.



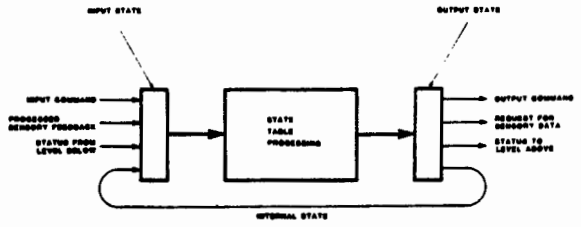
ILLUSTRATES THE CROSS-COUPLING BETWEEN THE SENSORY PROCESSING AND CONTROL DECISION MODULES. A SENSORY PROCESSING LEVEL RESPONDS WITH THE FEEDBACK INFORMATION REQUESTED BY THE CORRESPONDING CONTROL DECISION LEVEL.

Figure 7: SENSORY-CONTROL INTERACTION

State-Table Implementation

Sensory-interactive control must generate an output that is a function of its entire input state. An important aspect of this concept is that the entire input state, not just some subset of the input state, must be evaluated for each output. In order to implement a system that accomplishes this while remaining comprehensible, the system has been structured so that each control level is a state-table process where all of the inputs are sampled each time an output is to be generated.

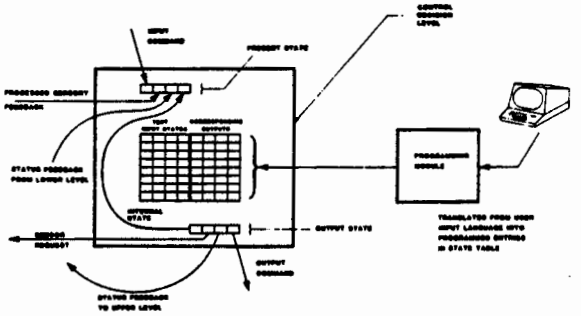
Each level must also include an internal state variable that effectively encodes the history of how the process arrived at the present condition. It is the internal memory of the process and stores the contextual information necessary to allow the system to step through a sequence of actions. (Figure 8)



EACH CONTROL LEVEL PERFORMS STATE-TABLE PROCESSING. AT EACH CONTROL CYCLE, ALL OF THE INPUTS ARE SAMPLED AND THE CONTROL DECISION FUNCTION GENERATES THE PROPER SET OF OUTPUT VALUES FOR THAT INPUT STATE.

Figure 8: STATE-TABLE INPUS AND OUTPUTS

The implementation of this hierarchical task decomposition control system configures each of the control decision levels as a state-table function generator. The input state is defined by the data that encode the input command, the processed sensory information, the status from lower levels and the internal state values. These input values are compared with preprogrammed sets of possible input conditions. If a match is made, then the corresponding output procedures are executed. The output values generated are the output command to the lower level, the request for sensory processing, the status report to the next higher level and the next internal state value. (Figure 9)



EACH LINE OF THE TABLE REPRESENTS A PREPROGRAMMED POSSIBLE INPUT STATE CONDITION AND A CORRESPONDING OUTPUT IF THAT STATE OCCURS. THE CONTROL LEVEL COMPARES THE PRESENT INPUT STATE AGAINST THE TEST STATE IN EACH LINE. WHEN A MATCH IS FOUND, THE LEVEL ASSUME THE OUTPUT STATE THAT LINE.

Figure 9: STATE-TABLE IMPLEMENTATION

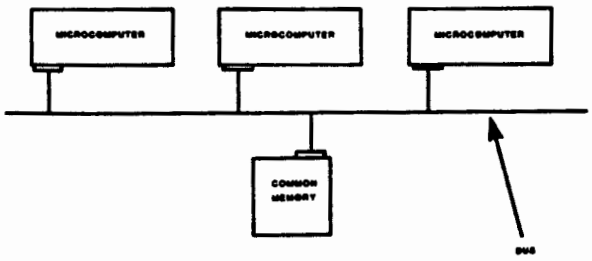
The table-like structure illustrated becomes the format into which a programming system can translate a robot task description (3,8). Each line of the table essentially represents a production rule of the type IF "this input condition" THEN "generate this output." It is a straight forward conversion into this table format from a representation of a task description written in an English-like procedural programming language.

This table format also provides the additional benefit of simplfying the modification or extension of the programming at each level. Each line of the table is a description of an input state and its corresponding response. Thus, each line in the table acts like an individual function or subroutine. Adding new routines to handle new conditions is accomplished by inserting new lines into the table and does not alter the operation of the previously entered lines.

Communication Mechanisms

The real-time aspect of control is based on the concept of producing a response to changes in input data (input state) in a appropriate time period so that this response is effective. If a control cycle (in which an input state is sampled and an output response generated) is repeated at a sufficiently fast rate, the system will provide continuous control in real-time.

In real-time robot control, the amount of computation, in general, is greater that can be completed on a single computer within the necessary time period for effective responsive behavior. For this reason, a multi-processor implementation is used to provide additional computational capacity (9). The multi-level control system lends itself readily to implementation on a multi-processor system. Each of the levels of the control system can execute independently on a separate processor. The inputs to a level - the command, status, and feedback data - can be read in from common memory buffers. The processor then computes the function of that level and the corresponding outputs can be written to common memory buffers that are available to the other processors (levels). An initial configuration has been designed that consists of several microcomputers connected through a common bus structure to a common memory. (Figure 10)

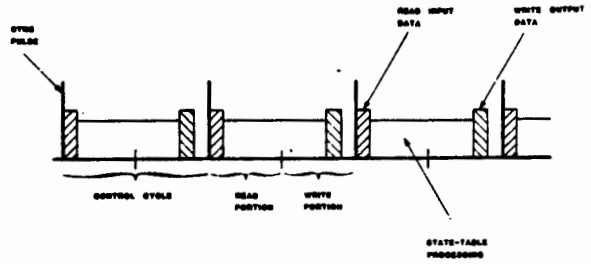


SEVERAL MICROCOMPUTERS CONNECTED THROUGH A COMMON BUS STRUCTURE TO A COMMON MEMORY AREA.

Figure 10: MULTIPLE-PROCESSOR STRUCTURE

To simplify the complexities of multi-processor interactions, a common memory buffer communication structure has been used. All data is written into and read from the common memory buffers. The addition or deletion of processes is also simplified. This indirect common memory communications link between processors allows the development of each process in isolation by supplying appropriate test values to the proper input common buffers for that particular process. Integration of the tested processes is accomplished by assigning the appropriate common buffers to the processes.

Real-time behavior requires the continuous repetition of the control cycle that samples inputs and generates outputs. This cyclic processing is realized through the use of a periodic synchronization pulse that defines the start of the control cycle. Within this cycle, each process will sample its input state and generate the appropriate output response. Each of these cycles is divided into a read portion and a write portion. During the first half of a cycle, each processor reads data from the common memory buffers. After computations and during the write portion of the cycle, the results can be written to common memory. This ensures that no data will be overwritten while another processor is reading that buffer. Only one processor is programmed to update any particular data buffer in common memory. If a process takes longer than one communication cycle, it waits until the write half period of the next cycle before writing its results to common memory. (Figure 11)



A PERIODIC SYNCHRONIZATION PULSE STARTS A CONTROL CYCLE. DURING THE FIRST HALF OF THE CYCLE, A PROCESSOR READS ITS DATA FROM COMMON MEMORY. AFTER PROCESSING, IT WRITES THE RESULTS TO COMMON MEMORY DURING THE WRITE PORTION OF THE CYCLE. THIS TIMING STRUCTURE DIRECTLY SUPPORTS THE STATE-TABLE PROCESSING CONCEPT.

Figure 11: PROCESSING TIMING STRUCTURE

All buffers are tagged with a number that indicates the cycle in which they were written so that any process determines how current its input data is by checking this number against the current cycle number. This technique allows processors to verify that they are executing with current data and also provides a check of the system. If a buffer has not been updated within a specified number of control cycles, the processor reading that buffer can report this information, and appropriate action can then be taken. (Figure 12)

This multiprocessor communications structure directly supports the state-table processing by providing an input-compute-output repetitive cycling with a short enough time period to generate

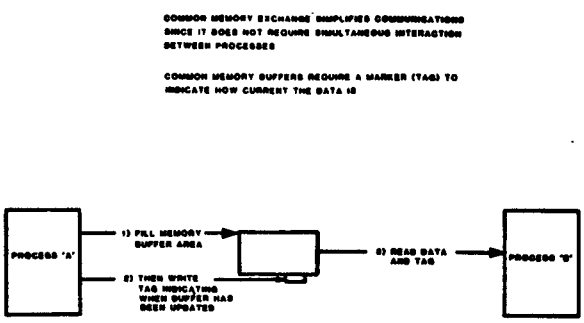


Figure 12: COMMON MEMORY COMMUNICAION

effective responses. This structure also provides many powerful debugging capabilities by the fact that the synchronized communication maintains in common memory a map of all the important variables at each instant. These variables are available to independent diagnostic processes that can monitor the performance of the system in real-time. Capabilities for real-time trace and breakpoint mechanisms are possible as well as the ability to examine common memory while single stepping the execution of the parallel processes by controlling the synchronization pulse.

Summary

The above discussion has treated the design of an architecture for a real-time sensory-interactive control system. Concepts from the areas of real-time control and complex system design were brought together to shape and guide this design.

Complex System Design Concepts

1) All systems must be designed to meet the human limitation of not being able to handle more than seven pieces of information at a time.
2) Systems are structured to account for this limitation by modularization into well-bounded, functionally independent components, each of which processes not more than a few pieces of information.
3) Further reductions in complexity are obtained by the use of generic processing structures.

Real-Time Control Concepts

1) A control system produces output actions that are a function of both its input command and feedback.
2) A sampled control system must generate an output within a task dependent time period after the occurrence of an event in order to provide an effective response.
3) Each response of the control system must be a function of the entire input state.

The application of complex system design concepts to those for real-time sensory-interactive control lead to the following:

1) Modularization of the system into independent components which clearly delineates the function and responsibility of each component.
2) Definition of generic control structures (generic control levels.)
3) Use of multiple processor architecture to provide sufficient processing for time critical responses.
4) Use of synchronized, common memory communications for continuous real-time response and clarity of understanding of multiple processor interactions.
5) Use of state-table processing for ease of control function specification, clarity and programmability.

References

1. Second Interim Report, Robotics Support Project for the Air Force ICAM Program, Management Summary of Levels of Control, April, 1979.

2. G. M. Schneider, S. W. Weingart and D. M. Perlman, "An Introduction to Programming and Problem Solving with Pascal", John Wiley & Sons, New York, 1978.

3. G. A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", Psychology Review, American Psychological Association, Inc. Vol 63., No. 2, March, 1956.

4. S. Alagic and M. A. Arbib, "The Design of Well-structured and Correct Programs", Springer-Verlag, New York, 1978.

5. B. W. Kernighan and P. J. Plauger, "The Elements of Programming Style", McGraw-Hill Publishing Company, New York, 1974.

6. B. W. Kernighan and P. J. Plauger, "Software Tools", Addison-Wesley Publishing Company, Reading, Massachusetts, 1976.

7. A. J. Barbera and M. L. Fitzgerald, "Recommended Procedures for the Design and Implementation of Real-time Control Software", NBS/U. S. Air Force ICAM Report, March 1982.

8. J. S. Albus, A. J. Barbera and M. L. Fitzgerald, "Programming a Hierarchical Robot Control System", Proceeding of the 12th Internation Symposium on Industrial Robots, June 9-11, 1982.

9. A. J. Barbera, J. S. Albus and M. L. Fitzgerald, "Hierarchical Control of Robots Using Microcomputers", Proceedings of the Ninth International Symposium on Industrial Robots, 1979, pp. 405-422.