PB90-244419

Approach to Telerobot Computing Architecture

(U.S.) National Inst. of Standards and Technology (NEL)
Gaithersburg, MD

18 Jun 90

U.S. Department of Commerce
National Technical Information Service
NTIS

PB90-244419

NISTIR 4357

AN APPROACH TO TELEROBOT COMPUTING ARCHITECTURE

John Fiala
Ronald Lumia

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Robot Systems Division
Intelligent Controls Group
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Dr. John W. Lyons, Director

NIST

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL
INFORMATION SERVICE
SPRINGFIELD, VA 22161

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
PB90-244419/AS

2. PERFORMING ORGANIZATION REPORT NUMBER
NISTIR 4357

3. PUBLICATION DATE
JUNE 1990

4. TITLE AND SUBTITLE

An Approach to Telerobot Computing Architecture

5. AUTHOR(S)

John C. Fiala; Dr. Ron Lumia

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

NIST Category No.
NIST- 135

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

In response to a Goddard Space Flight Center effort to look at the requirements for a computing architecture design appropriate for a telerobot, this document describes the Intelligent Controls Group approach to telerobot computing architectures. It is shown how a seven microprocessor control system can achieve teleoperation with an around-the-loop time of 10 ms. The document focuses on low-level, real-time robotics and does not discuss some issues relevant to space systems such as space-qualification and thermal design.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

control system architecture; microprocessor; robot control; teleoperation; telerobotics

13. AVAILABILITY

[XX] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
25

15. PRICE
A03

ELECTRONIC FORM

NISTIR 4357

AN APPROACH TO TELEROBOT COMPUTING ARCHITECTURE

John Fiala
Ronald Lumia

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards
and Technology
Robot Systems Division
Intelligent Controls Group
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

June 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Dr. John W. Lyons, Director

An Approach to Telerobot Computing Architecture

Intelligent Controls Group
Robot Systems Division
National Institute of Standards and Technology

Principle Authors: John Fiala
Ron Lumia

Date: June 18, 1990

Document number: ICG-# 26
Document version: 1.0

Scope of the Document

In response to a Goddard Space Flight Center effort to look at the requirements for a computing architecture design appropriate for a telerobot, this document describes the Intelligent Controls Group approach to telerobot computing architectures. It is shown how a seven microprocessor control system can achieve teleoperation with an around-the-loop time of 10 ms. The document focuses on low-level, real-time robotics and does not discuss some issues relevant to space systems such as space-qualification and thermal design.

Keywords: telerobotics, teleoperation, control system architecture, robot control, microprocessor

1. Introduction

This document presents a basic design for a telerobot. The complete robot design is not presented here, however, detailed designs of principal subsystems are presented. The emphasis is on the approaches taken toward software and hardware design of the computing architecture. Example designs are taken from the Servo Level. The motivation for the control system design is completely documented in the references at the end. The bibliography is a complete list of publications from the Robot Systems Division, National Institute of Standards and Technology (NIST), related to the design of telerobot control systems.

2. Software

To facilitate the discussion and simplify the figures, the following discussion on the design of software will concentrate on the Servo Levels of two main subsystems of the telerobot, the hand-controller subsystem of the operator interface and the manipulator subsystem of the robot side. These two subsystems are the main emphasis for projects such as Martin Marietta's FTS and Goddard's Multiple Algorithm Robot Control System (MARCS). Other levels and subsystems will have similar structure, so the example can be applied to other parts of the system as well.

The control system forms two main hierarchies, one for the operator interface and one for the robot. In the operator interface hierarchy, a hand-controller subsystem consists of a Primitive and Servo Level under the E-move Level. Likewise, as shown in Figure 1, a manipulator controller

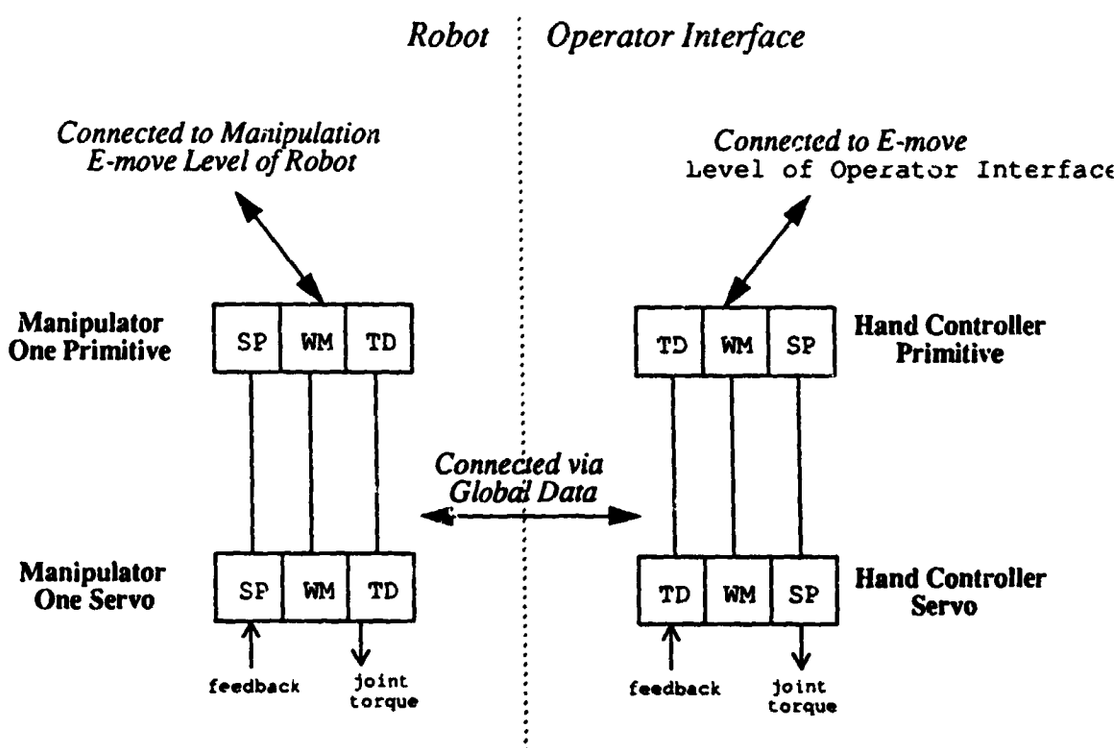


Figure 1. Subsystem Design for Arm and Hand Controller of Telerobot.

consists of a Primitive and Servo Level under the Manipulation E-move Level. On the robot side there are also E-move Levels for the Perception and Positioning subsystems. There is a second manipulator subsystem under the Manipulation E-move Level. This subsystem controls the second manipulator arm in a two arm telerobot. Similarly, there is a second hand-controller subsystem in the operator interface. In the discussion, the control subsystem for only one manipulator/hand-controller pair will be presented, the assumption being that a second manipulator/hand-controller pair would have a duplicate control structure.

The Servo Levels depicted in Figure 1 are directed by the corresponding Primitive Levels. However, in teleoperation mode, the two Servo Levels are linked through the global data system. The manipulator is moved based on sensory data generated in the hand-controller subsystem and the hand-controller is activated by sensory data generated in the manipulator subsystem. This teleoperation mode will be the focus of the discussion.

Figure 2 shows a detailed design for the Servo Levels of Figure 1. Figure 2.a represents the manipulator subsystem Servo Level and Figure 2.b represents the hand-controller Servo Level. Each box in Figure 2 is a cyclicly-executing, concurrent software process of the control system. The ovals are global data buffers. System processes communicate only through these global buffers. The particular configuration of boxes shown in Figure 2 is primarily based on the control architecture of the NIST lab system, but includes elements for the implementation of Martin Marietta's control algorithm. This design is also similar to the MARCS system, which is specifically designed to support algorithms like Martin's. (See Appendix 1 for the detailed MARCS design.) Additional boxes might be added for other algorithms, such as the Goddard teleoperation algorithm with adaptive gains, however the basic arrangement and characteristic of software processes depicted in Figure 2 is designed to support a wide range of algorithms without major modification.

Note that not all possible inputs and outputs of processes are shown in the figure. The figure shows all inputs and outputs necessary for the basic teleoperation mode under consideration. Detailed descriptions of the inputs and outputs to the processes in Figure 2 can be found in the references. Interfaces shown are assumed to be complete interfaces as described in the documents even though only a subset of a given interface might be used for the algorithm being discussed. Also, the output from the Servo Level is the desired torque for the joints. It is assumed that either this value is directly proportional to the required motor current, as in the direct-drive case, or that torque loops are provided subsequent to the output.

The data buffers used for communication between subsystems are shown shaded in Figure 2. The two buffers are repeated in both figures for clarity. The buffer f_m contains the force feedback from the robot force sensor related to the Cartesian coordinates of the hand-controller. The buffer z_0 contains the Cartesian hand-controller command related to the Cartesian coordinates used for the robot control. The z_0 data is the final hand-controller command which includes indexing and scaling.

Each process at the Servo Level in Figure 2 has associated with it a number appearing in the upper right-hand corner of the box. Table 1 gives the execution times for each process executing the appropriate component of the Martin Marietta algorithm. Note that the Martin algorithm is completely implemented by the numbered processes. In fact, some additional functionality (such as gravity compensation) is provided which is not available in Martin's implementation. The execution times are based on Martin Marietta published data. Martin Marietta obtained these times by

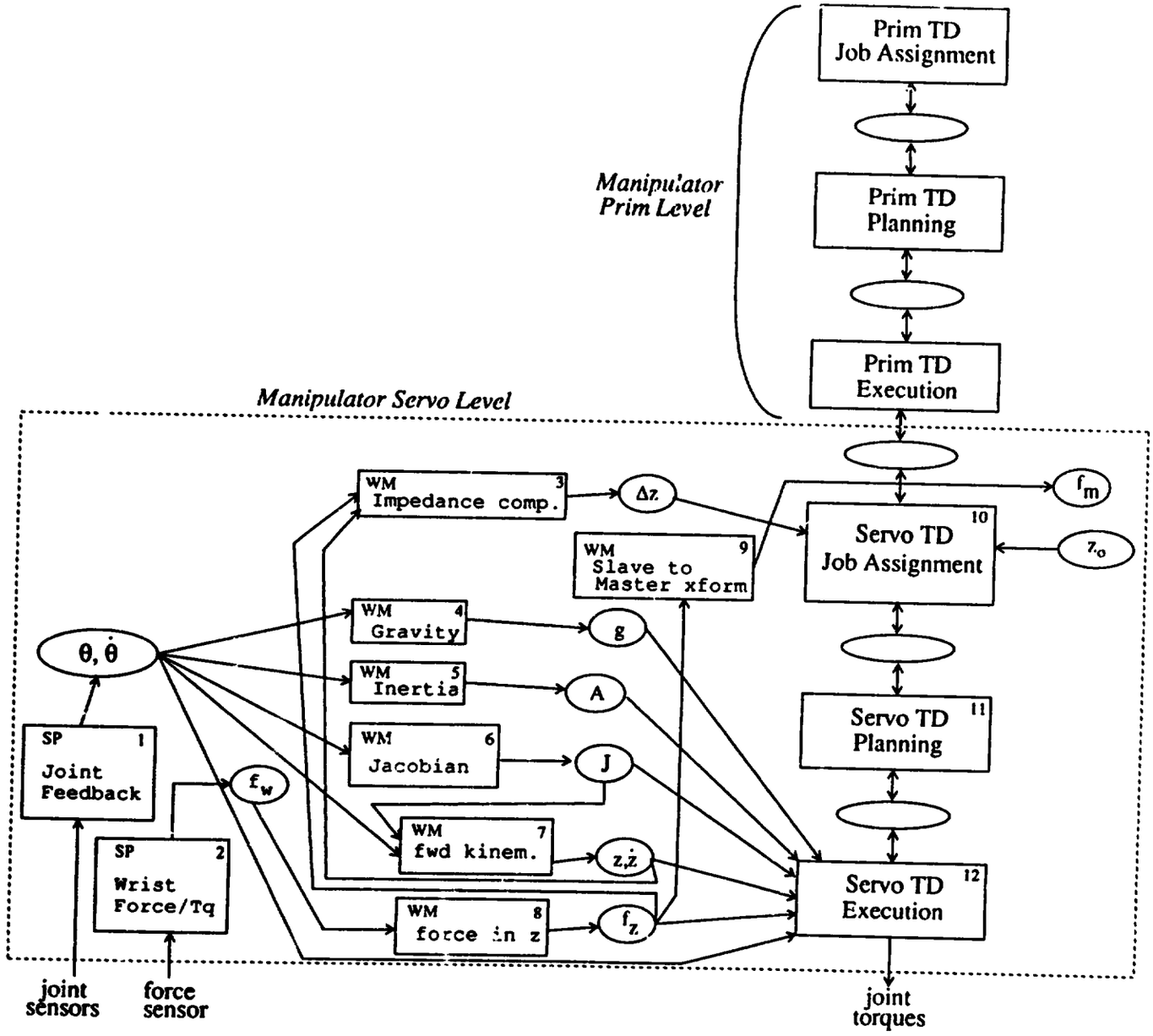


Figure 2. a. Manipulator Subsystem Detailed Design.

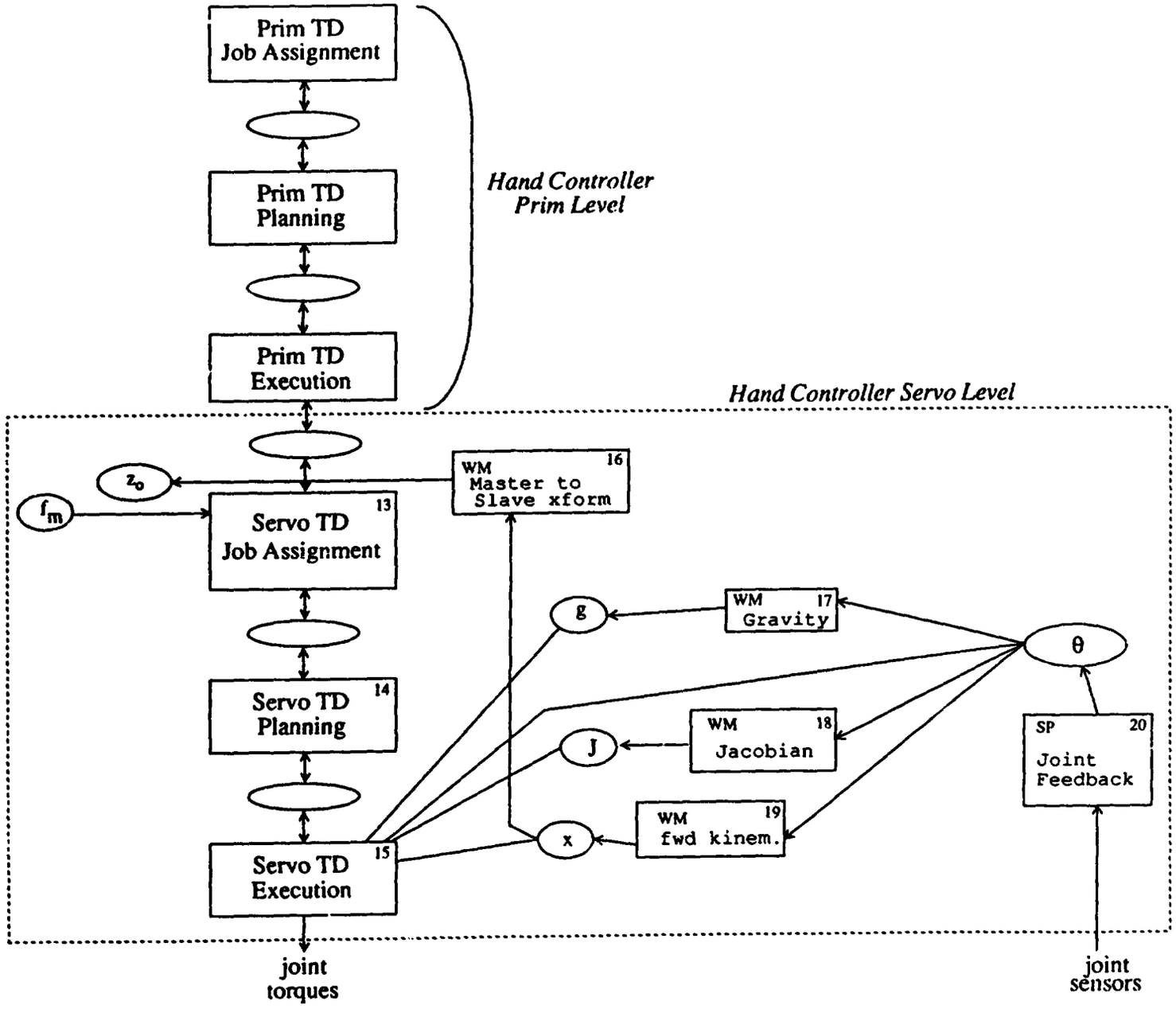


Figure 2. b. Hand Controller Subsystem Detailed Design.

Table 1. Execution/Communication Times for Figure 2 Processes in Martin Algorithm

Process	Process Function in Martin Algortihm	Execution Time
1	Acquire joint feedback	0.34
2	Acquire force/torque sensor data	0.17
3	Calc. position mod. based on impedance	1.50
4	N/A	2.10
5	Calc. inertia decoupling matrix	17 00
6	N/A	4.70
7	Forward kinematics on joint position	1.43
8	Trans. force sensor data to control coord.	0.38
9	Translate manip. to hand cntllr coord.	0.39
10	Combine imp. and cmd/ inverse kinematics	1.47
11	Joint rate limiting	0.27
12	Joint PD control with inertia decoupling	0.90
13	N/A	0.25
14	Force feedback limiting	0.27
15	Jacobian transpose multiplication	0.57
16	Translate hand cntrllr to manip. coord.	0.78
17	N/A	1.80
18	Calc. hand-controller Jacobian	1.32
19	Forward kinematics on joint position	1.43
20	Acquire joint feedback	0.34

implementing and timing actual Ada code on the 80386 microprocessor. In calculating the times for Table 1 for each process, Martin Marietta execution times for functions performed in a process were summed and the resulting value was multiplied by 1.2 to allow for a communication overhead of 20%. The result is rounded up to the nearest hundredth of a millisecond. For processes for which execution times are not available, times obtained on Ada code implemented on the 68020 for a seven degree-of-freedom manipulator control system were used as comparable values. The execution times of Table 1 are used in the next section to show how processes can be distributed to processors for real-time performance when the correct hardware architecture is chosen.

3. Hardware

The problem of telerobot control requires a powerful multiprocessing architecture. This multiprocessing architecture must involve a large number of tightly-coupled processors to perform the

highly centralized aspects of coordinated control. This fact will be demonstrated for the example subsystems of Section 2. First, consider the following desirable features of the centralized part of the computer architecture design for a telerobot.

3.1. Processor coupling via a 32-bit data bus.

The rate that data is transferred between processors in the control system requires that the communication bus support a high communication bandwidth. Since many processors will share the same data bus, megaword-per-second data rates are necessary. Also, the word size for values used in robot applications is mostly 32 bits, since robot control computations are done using 32-bit floating point representations. Current floating point hardware technology is capable of computations with no less than 32 bits. Any smaller representation requires expensive conversions to a larger format. In addition, 32-bit data is the state-of-the-art in microprocessors and multi-microprocessor backplanes.

3.2. Multislot (>20) backplane.

To accommodate the large number of processors and allow for easy system expansion, a multislot backplane is required. Processor and other hardware elements connected by an address and data bus configured as a multislot backplane provides for tight coupling with the added benefit that elements can easily be added or removed. Even a 20-slot backplane may not be big enough depending on the complexity of the system and its degree of autonomy. Multiple multislot backplanes may be desirable.

3.3. Subsystem bus capability.

In addition to a high-bandwidth main bus connecting processors, it is useful to have subsystems buses that allow subgroups of processors to communicate separately from the main bus. For example, the processors performing centralized control of an arm need to communicate more frequently with each other than they do with the processors controlling the other arm, in general. This communication is local to the arm processors and traffic on the main bus can be reduced by giving the arm processors a separate subsystem bus. The arm processors can still use the main bus, but use the subsystem bus for communication within the subsystem. Since the values transferred over the subsystem bus are for the most part 32-bit, the subsystem bus should also have a 32-bit data path.

3.4. Industry standards.

Any computer architecture design should try to use industry standards where possible. Compliance to industry standards reduces development costs, improves reliability, and increases the compatibility with current and future products. This feature, along with items 3.1-3.3, indicates that a good choice for the coupling processors would be an industry standard 32-bit address and data bus, such as Multibus-II or VME bus, configured as a multislot backplane. This type of architecture has been in use for many years and is well tested.

With respect to the distributed part of the control system, i.e., the part that resides in the manipulator arms, it is our understanding that there are two major problems. The first problem is ca-

bling. It is desirable to minimize the number of cables passing through a manipulator. The second problem relates to thermal control within a manipulator. The amount of heat generated within an arm should be kept to a minimum. With these points in mind, here are some features that may be desirable for the distributed part of the hardware architecture.

3.5. Only local control at the joints.

The nature of the control problem for a serial mechanism such as a robot manipulator requires a centralization of control for coordination. An example of this is the inertia decoupling compensation required for FTS. This decoupling can only be done for the manipulator as a whole, not by each joint independently. Any attempt at joint-local decoupling will result in high communication requirements between joints, defeating the whole purpose of the local joint control. Thus, it is important that any control distributed to joint controllers be truly local. For the FTS this means that only joint torque control loops should be done in joint-mounted electronics.

3.6. Digital communication bus in arm.

The large number of sensors (current, temperature, etc.) and the desire to reduce cabling leads to the conclusion that sensory data should be digitally encoded and transmitted over a common link. A simple hardware link, such as a twisted-shielded pair, may be best for this purpose since the cable will need to be quite flexible to minimize cable friction. This need, along with item 3.4, may indicate that 1553 is an appropriate choice. The bandwidth required for this link should not be that great provided the link only transmits sensor feedback and reference signals for <u>local</u> control at the joints. If additional bandwidth is needed, an additional digital bus of the same type can be added to the manipulator.

3.7. Programmable torque loops.

The widely varying thermal conditions in which the FTS will operate, and the inability to test it fully in real space environments, make it likely that the torque loop parameters will need to be modified after the FTS is constructed. In addition, if the FTS is to operate for a number of years on the space station, wear and other aging factors will also require that control parameters change. Thus, the torque loops operating on the joints should be programmable in that all the control parameters could be modified. The most flexible way to do this is to implement digital torque loops on a general microprocessor, although it may be just as easy to implement the torque loops using a microcontroller or programmable gain amplifiers. Not using general microprocessors may be the better approach in minimizing size and heat generated at the joints.

3.8. Single motor power bus.

To minimize cabling complexity the motor power for the various joints in a manipulator should all be taken off of a single power bus. This implies that the PWM electronics for providing motor currents should be distributed in the manipulator. The motor command for the PWM's will be taken from the output of the torque loops.

Considering all of the above points, the telerobot hardware architecture for a manipulator/hand-controller Servo Level pair is obtained as depicted in Figure 3. (For further validation of this basic

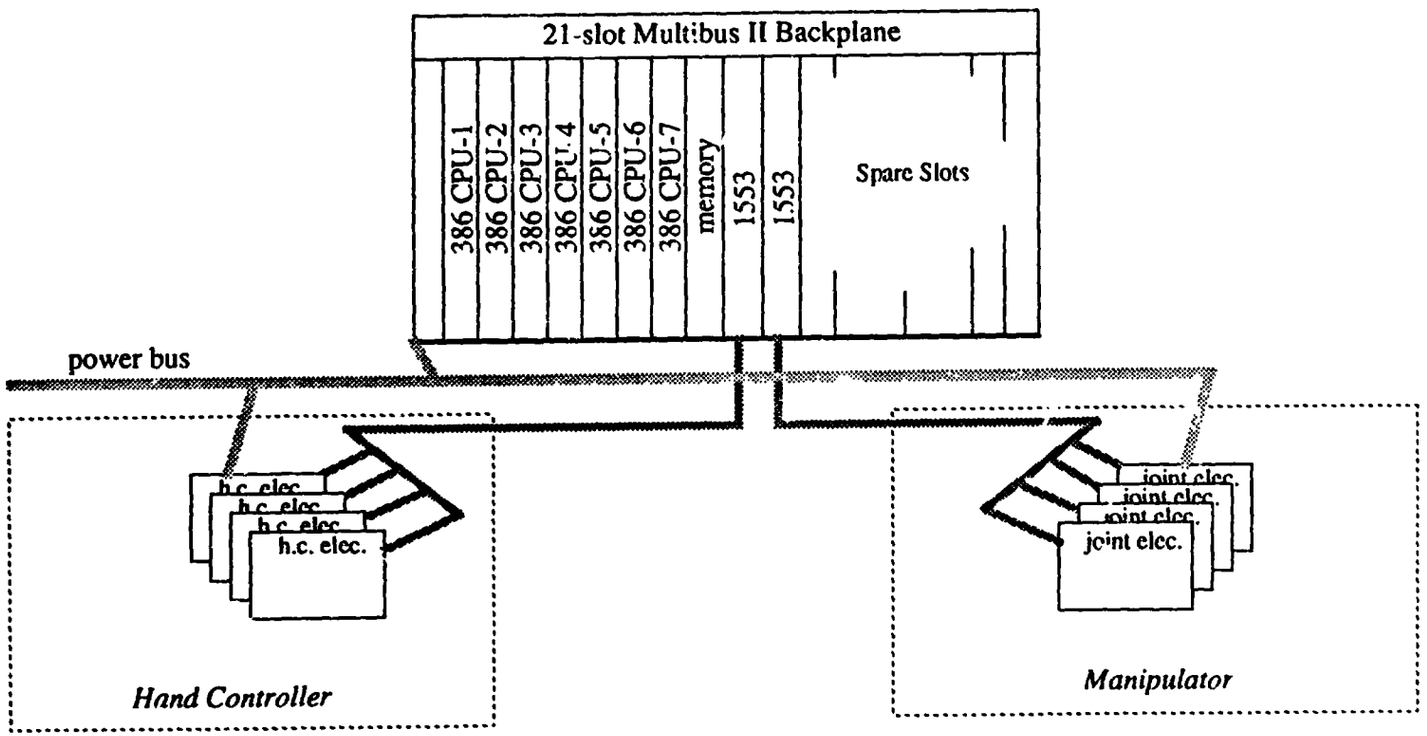


Figure 3. Hardware Architecture for Manipulator/Hand Controller Pair.

approach, consider the hardware architecture developed independently for MARCS in Appendix 1.) In this design, all of the general processors reside in a common backplane. These processors communicate with each other over Multibus II using common memory residing on the shared bus. The number of general 386 processors is seven, which is chosen to correspond with the number of microprocessors used by Martin Merietta for a manipulator/hand-controller pair. The processors communicate with device-mounted electronics over the 1553 serial data lines.

The device-mounted electronics include all the electronics necessary to receive the digital torque commands from the 1553 and produce the corresponding actuator motions. In addition, these electronics interface the sensors to the digital bus for sensor feedback. For the manipulator, the device-mounted electronics could be physically located within the links of the manipulator arm. The electronics may include, in addition to the 1553 interface, power electronics, commutation electronics, PWM electronics, torque control loops, and analog i/o hardware. It is assumed that general microprocessors are not used in the device-mounted electronics.

Referring back to Figure 2, all the processes depicted in the figure reside on the seven general-purpose processors in the multislot backplane. Thus, the communication at the "bottom" of the Servo Levels, (joint torques and feedback in Figure 2,) is over the 1553 bus. By restricting 1553 data to be this low-level data only, the ability of the 1553 bus to meet the communication requirements is assured. For example, if seven 16-bit joint torques are sent, seven 32-bit joint positions and six 16-bit force/torque values are received, this requires 27 16-bit words. With the 1553, each 16-bit word is transferred at a cost of 20 bits, for a total of 540 bits. Add a few protocol words and the total transfer is about 600 bits. If this data is transferred every millisecond then the required bus rate would be 600Kbits/s. This rate is well within the capabilities of the 1553 which has a specification for 1 Mbit/s. If an actual 1553 bus only achieved 80% of this (800Kbit/s), the transfer of 600 bits would still take only 0.75 ms.

Now consider the execution times presented in Table 1, and note the following. Processes 4, 5, 6, 17, and 18 can run at a fairly slow rate with respect to the rest of the system and not effect control system performance. These processes need not update their outputs more than about 20 times a second. There are numerous ways to distribute the remaining processes to processors. The system is completely flexible in this respect since all processors are tightly coupled. Processes can be distributed to minimize the around-the-loop time of the teleoperation control, maximize the processing margin, minimize the time between new updates of joint torques, or optimize a number of other factors.

Suppose, as an example, a minimum around-the-loop time for the force-feedback teleoperation control is desired. Note that the around-the-loop time is defined by the process sequence {2, 8, 9, 13, 14, 15, 20, 19, 16, 10, 11, 12}. By grouping processes 4,5,6,17, and 18 on one processor, each of these processes would compute a new output at around 37 Hz. The remaining processes can be distributed to processors as follows.

CPU-1 : 4,5,6,17,18

CPU-2 : 1,2,8

CPU-3 : 9,13,14

CPU-4 : 15,20

CPU-5: 19,16,10,11

CPU-6: 12

CPU-7: 7,3

Here, each processor executes the processes in the order listed. Processors 2, 3, 4 and 6 can each execute their processes in less than a millisecond. The reader may verify this by adding the execution times from Table 1. Processor 5 can execute its processes in less than 4 ms, while processor 7 takes less than 3 ms. Thus, all the processors can be synchronized on a one-millisecond boundary so that outputs of processes 1, 2, 8, 9,13,14,15, 20, and 12 get updated every millisecond, outputs of processes 7 and 3 get updated every 3 ms, and the outputs of processes 19, 16, 10, and 11 get updated every 4 ms. The around-the-loop time is determined by the processor sequence {CPU-2, CPU-3, CPU-4, CPU-5, CPU-6}. Adding up the times for this sequence (1+1+1+4+1=8), and making the mild assumption that the additional time-delay for the 1553 and the device-mounted electronics is less than 1 ms per combined torque/feedback transaction, the around-loop-time is 10 ms. This loop has pipelined updates coming every millisecond. In addition, the example achieves a 333 Hz local joint control rate, and a 100 Hz impedance loop rate.

The important point is that the hardware architecture provides many options in terms of allocating computing resources. Other processing resource allocations can easily be made without changing either the hardware or software architectures. For example, the distribution

CPU-1 : 4,5,6,17,18

CPU-2 : 2,8,9,13,14,15,20

CPU-3 : 19,16

CPU-4 : 10,11

CPU-5: 1,12

CPU-6: 7,3

with processors 2, 3, and 4 repeating every 2.5 ms, processor 5 repeating every 1.25 ms, and processor 6 repeating every 3.75 ms, results in an implementation with a 14 ms teleoperation around-the-loop time and a 400 Hz local position loop on the manipulator. And this is with a spare processor left over. It is possible to implement the whole system on just five processors and still achieve good performance by making the distribution

CPU-1 : 4,5,6,17,18

CPU-2 : 1,2,8,9,13,14,15,20

CPU-3 : 19,16

CPU-4 : 10,11,12

CPU-5: 7,3.

The examples show the tradeoffs in performance for the implementation of the Martin Marietta algorithm that can be made for the computing architecture approach of Figures 2 and 3. It should be noted, however, that the Martin algorithm has not been shown to be superior in any way to other approaches. In fact, Martin's use of explicit inverse kinematics in the control loop ensures that the algorithm can not be used for arms with more than six degrees of freedom, or even with six degree-of-freedom arms without simple kinematics. This has been the conclusion of both the NIST and

Goddard labs. Thus, it is important that the control system design for a telerobot be able to run more than just the Martin Marietta algorithm. The system should be easily extensible to run other algorithms and additional levels of the telerobot functional architecture.

4. Extensibility

As discussed in Section 2, the complete telerobot architecture involves many more components than the Servo Levels analyzed in Sections 2 and 3. Some of these components are depicted in the partial telerobot architecture of Figure 4.

Primitive Levels, as depicted in Figure 2, connect the manipulator and hand-controller Servo Levels to the upper levels of the telerobot. These Prim Levels generate trajectories and autonomous reference commands for the manipulator and hand-controller. There must be a Prim and an E-move Level for autonomous manipulator operations such as automatic end effector exchange. Thus, the processes which realize these upper levels must be given computing resources within the system. Note that, in general, resources cannot be taken away from the Servo Level processes since many of the processes must remain in operation when the upper levels are active. (The E-move, Prim, and Servo Levels form a pipeline for hierarchical control.)

With the hardware design of Figure 3, extra slots are available to add processors for the upper levels. Some processors from the seven original ones may also be available to implement the upper levels depending on how processes are allocated to processors at the Servo Level. Extra processors

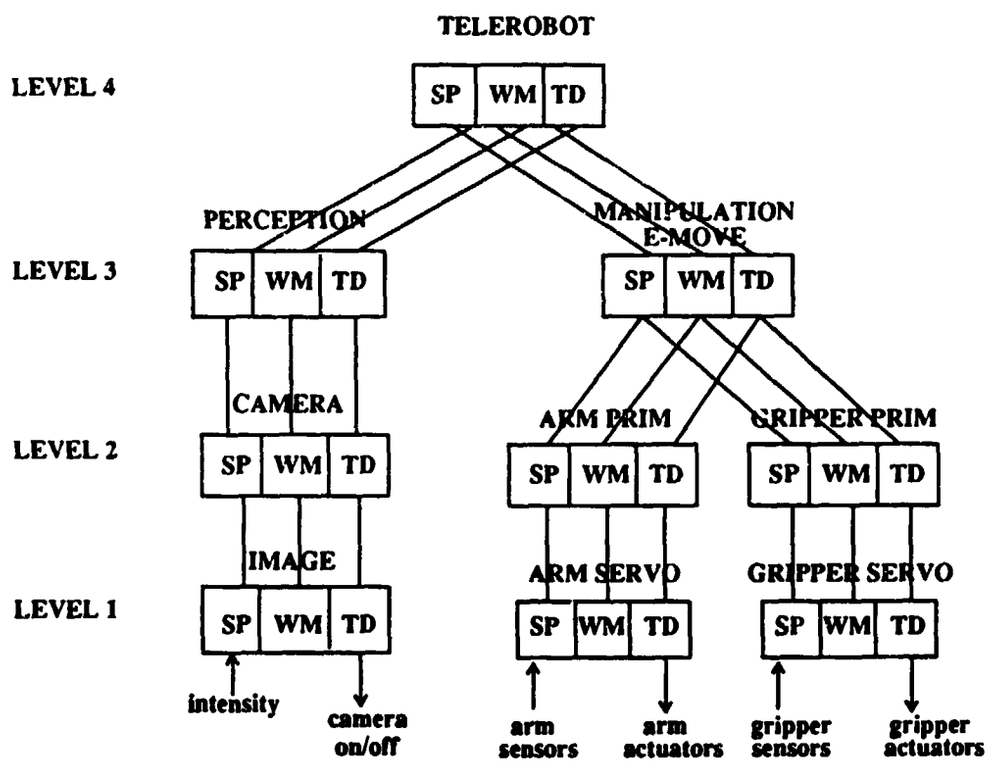


Figure 4. Partial Telerobot Control System.

can also be made available to execute other subsystems of the telerobot such as Safety, Positioning and Perception.

With respect to Perception, if the telerobot is to eventually have the capability to analyze images, then it must be possible to add computing resources to the telerobot to perform this function. Generally, specialized hardware is used to perform the computationally expensive function of converting the input intensity image from an iconic (spatially based) image to a symbolic (feature based) image. Additional general support computers process the symbolic image to complete the perception task. A number of specialized hardware systems exist for performing image analysis including PIPE, WARP, PIFEX, and DATACUBE. Such specialized hardware should be easily incorporated into the control system.

With the hardware design of Figure 3, the specialized vision hardware can be given a dedicated processor in multislot backplane. The approach which has been successful in the NIST laboratory has been to use a PIPE interface board along with a general processor to control and configure the PIPE. This allows the processor to use the PIPE like an accelerator to speed up vision computation. The addition of the PIPE has little effect on the performance of the rest of the control system.

5. Conclusion

The design approach presented here represents an accumulation of knowledge from many years of work at NIST. This approach is well-documented in the references which follow. The design presented here supports flexibility in control system performance, a variety of control algorithms, and provides an evolutionary path to additional telerobot functionality.

6. Bibliography

6.1. General

[1] Albus, J. S., Lumia, R., McCain, H. G., "Hierarchical Control of Intelligent Machines Applied to Space Station Telerobots," Proc. IEEE Intl. Symp. Intelligent Control, Philadelphia, PA, January, 1987.

[2] Albus, J. S., Lumia, R., McCain, H. G., "A Control System Architecture for the Space Station Flight Telerobotic Servicer," Space Telerobotics Workshop, Pasadena, CA, January, 1987.

[3] Albus, J. S., Lumia, R., "Software Architecture for Manufacturing and Space Robotics," 2nd AIAA/NASA/USAF Symp. Automation, Robotics, & Adv. Comp. for Nat. Space Program, March, 1987.

[4] Albus, J. S., Lumia, R., "Space Robotics -- From Teleoperation to Autonomy," 3rd Intl. Service Robot Congress and Exposition, Chicago, April, 1987.

[5] Albus, J. S., Lumia, R., "NASREM -- NASA/NBS Standard Reference Model for Telerobot Control," Goddard Conf. on Space Appl. of AI and Robotics, May, 1987.

[6] Albus, J.S., McCain, H.G., Lumia, R., "NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)," NIST Tech. Note 1235, NIST, Gaithersburg, MD, July, 1987.

[7] Lumia, R., Albus, J. S., "An Architecture to Support Teleoperation and Autonomy," IEEE Conf. Sys., Man, Cybern., Alexandria, October, 1987.

[8] Lumia, R., Fiala, J., Wavering, A., Albus, J. S., "Kinematics and Dynamics in a Hierarchically Organized Robot Control System," NATO Workshop on Kinematic and Dynamic Issues in Sensor Based Control, Il Ciocco, Italy, October, 1987.

[9] Wavering, A. J., Fiala, J. C., "The Real-Time Control System of the Horizontal Workstation Robot," NIST Internal Report 88-3692, December, 1987.

[10] Albus, J. S., Lumia, R., "Teleoperation and Autonomy for Space Robotics," Special Issue on Automation in Space, R. Lumia, ed., Robotics, Vol. 4, no. 1, March, 1988.

[11] Leake, S. A., Kilmer, R. D., "The NBS Real-time Control System User's Reference Manual," NIST Tech. Note 1250, August, 1988.

[12] Albus, J. S., Lumia, R., McCain, H. G., "A Control System Architecture for the Space Station Flight Telerobotic Servicer," IEEE Trans. Aerospace and Electronic Systems, Vol. 24, no. 5, September, 1988.

[13] Lumia, R., Albus, J. S., "Space Robotics: Evolution and Applications," Paper #88-1640, Proc. ISA 88 Intl. Conf. & Exhibit, Houston, TX, October, 1988.

[14] Lumia, R., "Space Robotics: Automata in Unstructured Environments," IEEE Conf. Robotics & Automation, Scottsdale, AZ, May, 1989.

[15] Lumia, R., "NASREM as a Functional Architecture for the Design of Robot Control Systems," Workshop on the Integration of AI and Robotics Systems, Scottsdale, AZ, May, 1989.

[16] Lumia, R., Fiala, J., Wavering, A., "The NASREM Robot Control System Standard," Robotics and Computer Integrated Manufacturing, Vol. 6, no. 3, 1989.

[17] Lumia, R., Fiala, J., Wavering, A., "The NASREM Robot Control System and Testbed," Intl. Jour. Robotics & Automation, special issue on Robots in Unstructured Environments.

[18] Lumia, R., "NASREM: A Functional Architecture for Control of the Flight Telerobotic Servicer," 2nd European In-Orbit Operations Tech. Symp., Toulouse, France, September, 1989.

[19] Albus, J.S., Lumia, R., Fiala, J., Wavering, A.,"NASREM: The NASA/NBS Standard Reference Model Telerobot Control System Architecture," Japan, October, 1989.

[20] Albus, J.S., Lumia, R., "NASA/NBS Reference Model," Aerospace America, February, 1990.

[21] Lumia, R., "A Standard Architecture for Space Applications," Eastern Quality Conf., Washington, DC, February, 1990.

6.2. Software Related

[22] Fitzgerald, M. L., Barbera, A. J., "A Low-level Control Interface for Robot Manipulators," NBS-Navy NAV/SIM Workshop on Robot Standards, June, 1985.

[23] Fiala, J. C., Lumia, R., Albus, J. S., "Servo Level Algorithms for the NASREM Telerobot Control System Architecture," SPIE Vol. 851, Proc. SPIE Conf. -- Space Station Automation III, Cambridge, MA, November, 1987.

[24] Leake, S., "A Comparison of Ada and C on Sun and microVAX," Proc. 11th IASTED Intl. Symp. Robotics & Automation, Santa Clara, CA, May, 1988.

[25] Chaconas, K. J., Nashman, M., "Visual Perception Processing in a Hierarchical Control System: Level 1," NIST Tech. Note 1260, June, 1988.

[26] Lumia, R., Fiala, J. C., Wavering, A. J., "An Architecture to Support Autonomy, Teleoperation, and Shared Control," Proc. IEEE Conf. Sys. Man & Cybernetics, Beijing, China, August, 1988.

[27] Fiala, J. C., "Interfaces to Teleoperation Devices," NIST Tech. Note 1254, October, 1988.

[28] Fiala, J., "Manipulator Servo Level Task Decomposition," NIST Technical Note 1255, NIST, Gaithersburg, MD, October, 1988.

[29] Wavering, A. "Manipulator Primitive Level Task Decomposition," NIST Technical Note 1256, NIST, Gaithersburg, MD, October, 1988.

[30] Wavering, A., Lumia, R., "Task Decomposition Module for Telerobot Trajectory Generation," SPIE Conf. -- Space Station Automation IV, Boston, November, 1988.

[31] Lumia, R., Fiala, J. C., Wavering, A. J., "NASREM: Robot Control System and Testbed," Proc. 2nd Intl. Symp. Robotics & Manufacturing Research, Education & Applications, Albuquerque, NM, November, 1988.

[32] Lumia, R., Fiala, J. C., "The Flight Telerobotic Servicer: From Functional Architecture to Computer Architecture," Proc. Conf. Space Telerobotics, Pasadena, CA., January, 1989.

[33] Lumia, R., Wavering, A. J., "Trajectory Generation for Space Telerobots," Proc. Conf. Space Telerobotics, Pasadena, CA, January, 1989.

[34] Leake, S., Green, T., Cofer, S., Sauerwein, T., "Hierarchical Ada Robot Programming System (HARPS): A Complete and Working Telerobot Control System Based on the NASREM Model," Proc, IEEE Conf. Robotics & Automation, Scottsdale, AZ, May, 1989.

[35] Lumia, R., "Sensor-Based Robot Control Requirements for Space," Proc. NATO's Advanced Research Workshop on Sensors, Maratea, Italy, August, 1989.

[36] Nashman, M., Chaconas, K., "Visual Perception Processing for the Flight Telerobotic Servicer Control System," Intelligent Controls Group Internal Report, October, 1989.

[37] Kelmar, L. "Manipulator Servo Level World Modeling," NIST Tech. Note 1258, NIST, Gaithersburg, MD, December, 1989.

[38] Fiala, J., "Note on NASREM Implementation," NIST Internal Report 89-4215, NIST, Gaithersburg, MD, December, 1989.

[39] Kelmar, L. "Manipulator Primitive Level World Modeling," NIST Tech. Note 1273, NIST, Gaithersburg, MD, December, 1989.

[40] Fiala, J. C., Wavering, A. J., "Implementation of a Jacobian-Transpose Algorithm," NIST Internal Report 90-4286, NIST, Gaithersburg, MD, March, 1990.

[41] Chaconas, K., "Range from Triangulation Using an Inverse Perspective Method to Determine Relative Camera Pose," NIST Internal Report, April, 1990.

[42] Chaconas, K., Kelmar, L., and Nashman, M., "A NASREM Implementation of Position Determination from Motion," NIST Internal Report 90-4293, NIST, Gaithersburg, MD, May, 1990.

[43] Lumia, R., "Short Term Evolution for the Flight Telerobotic Servicer," Intelligent Controls Group Internal Report ICG-25, March, 1990.

[44] Kelmar, L., Lumia, R., "World Modeling for Sensory Interactive Trajectory Generation," Third Intl. Symp. on Robotics in Manufacturing, Vancouver, July, 1990.

[45] Michaloski, J., Wheatley, T. E., "Design Principles for a Real-Time Robot Control System," IEEE Intl. Conf. Systems Eng., Pittsburg, PA, August, 1990.

[46] Nashman, M., Chaconas, K., "Three Dimensional Position Determination from Motion," SPIE Conf. Intelligent Robots, Boston, November, 1990.

6.3. Hardware Related

[47] Dagalakis, N., "Robot Characterization Testing," Intelligent Controls Group Internal Report ICG-6, October, 1987.

[48] Akhavian, R., Catoe, R., Lumia, R., Smith, D., Staken, P., Watzin, J., "A Computer and Communications Architecture for the FTS," Unpublished document, June, 1988.

[49] Nashman, M., Chaconas, K. J., "Low Level Image Visual Processing Techniques Using the Pipeline Image Processing Engine in the Flight Telerobotic Servicer," NASA Conf. Pub. 3009, Goddard Conf. Space Applications of Artificial Intelligence, Goddard Space Flight

Center, Greenbelt, MD, 1988.

[50] Michaloski, J. L., Wheatley, T. E., Lumia, R., "Requirements for Implementing Real-Time Control Functional Modules on a Hierarchical Parallel Pipelined System," Conf. on Space Telerobotics, Pasadena, CA, January, 1989.

[51] Michaloski, J. L., Wheatley, T. E., Lumia, R., "Timing Analysis for a Parallel Pipelined Hierarchical Control System," 9th Real-Time Systems Symp.

[52] Wheatley, T. E., "Mapping Processes to Processors for Space-Based Robot Systems," Proc. IEEE Conf. Systems Eng., Wright State Univ., Dayton, OH, August, 1989.

[53] Fiala, J., "A High-Speed Serial Interface to the Robotics Research Corporation Servo Controller," Intelligent Controls Group Internal Document, November, 1989.

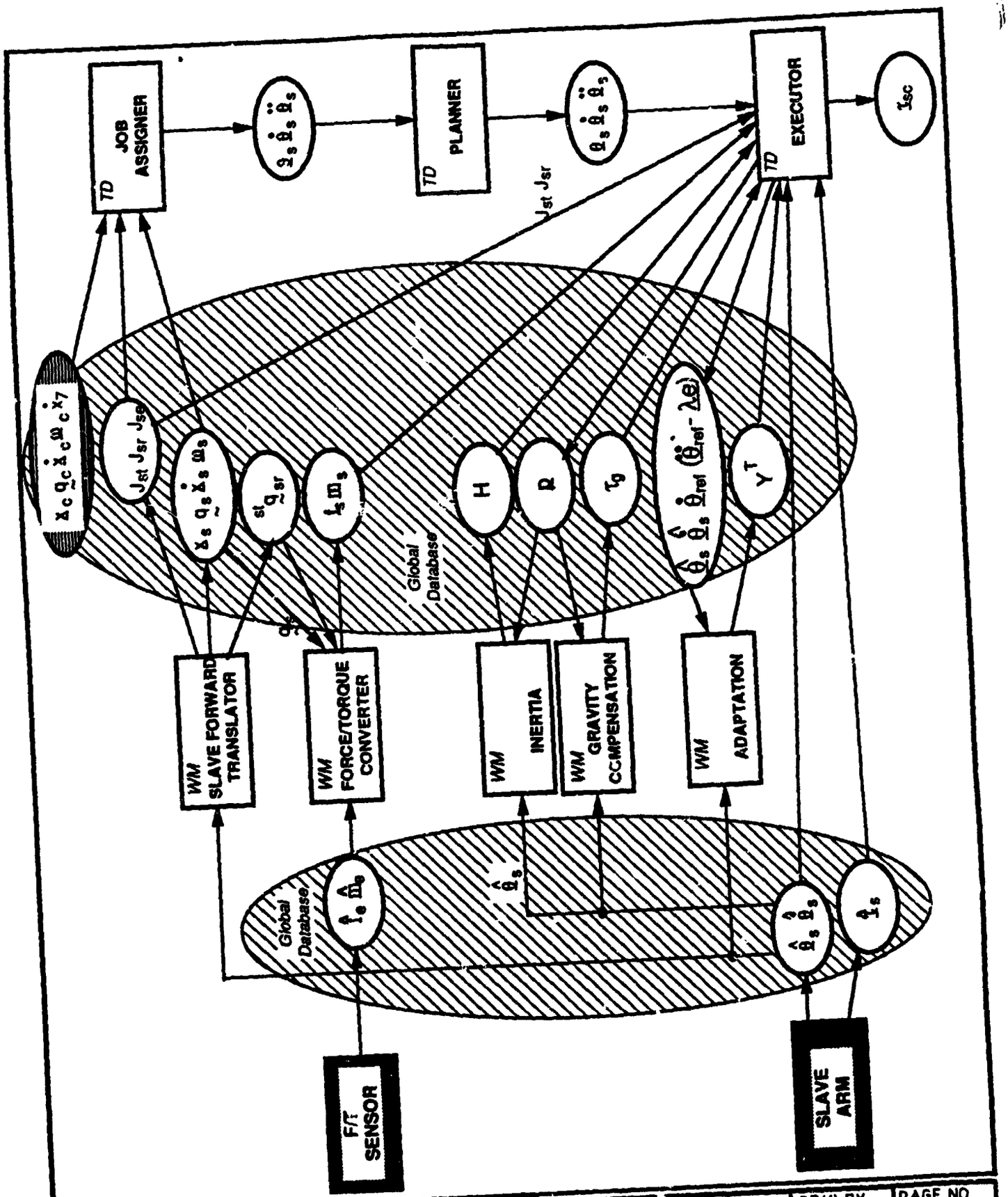
[54] Michaloski, J. L., Wheatley, T. E., Lumia, R., "Analysis of Computational Parallelism with a Concurrent Hierarchical Robot Control System," NIST Internal Report 90-4251, January, 1990.

[55] Lumia, R., "Integrating Sensors into a Standard Control Architecture for Robotic Applications," IEEE Instrumentation & Measurement Tech. Conf., San Jose, February, 1990.

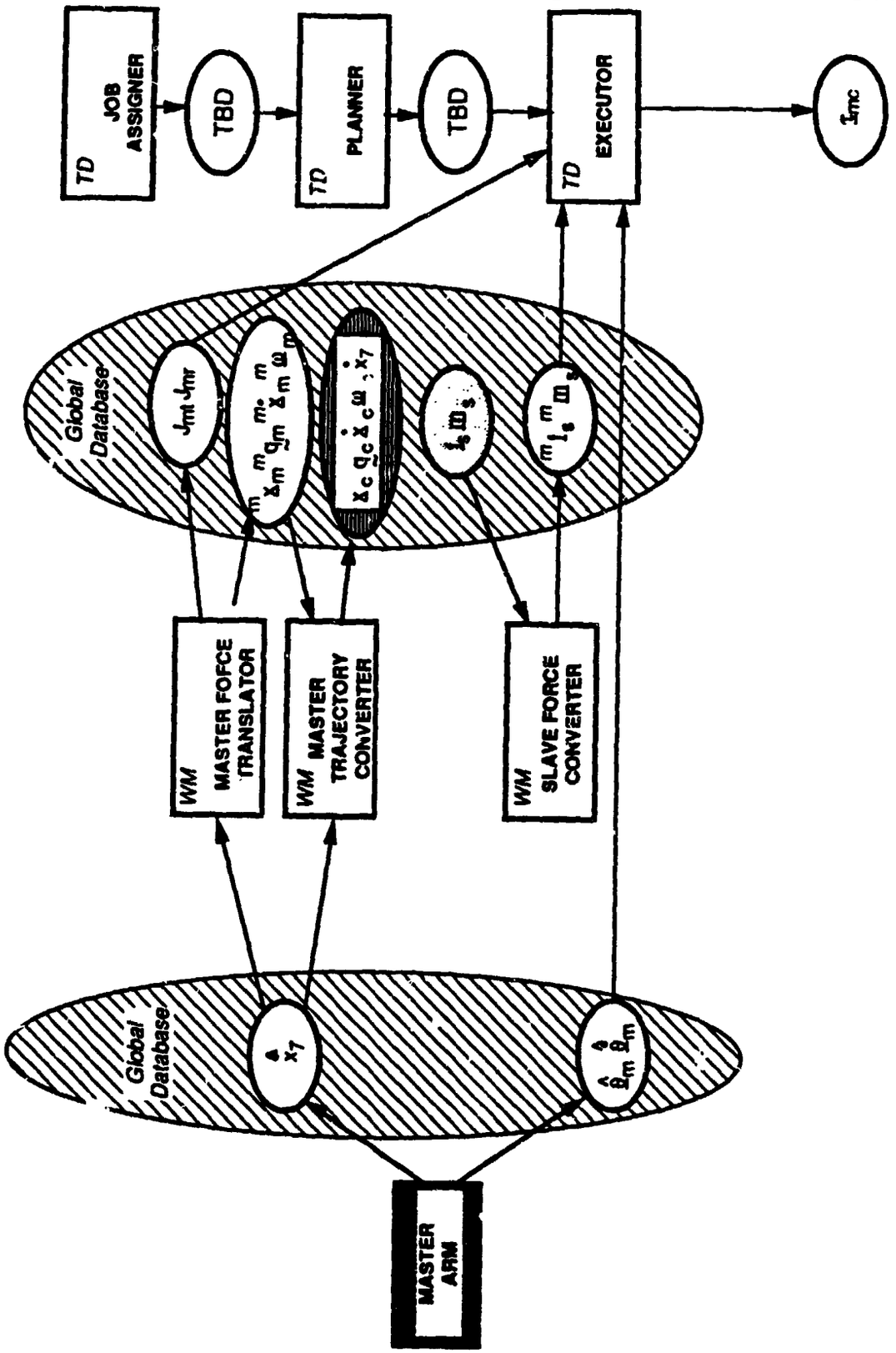
[56] Wheatley, T. E., Michaloski, J., "Configuration and Performance Evaluation of a Real-Time Robot Control System: The Skeleton Approach," IEEE Intl. Conf. Systems Eng., Pittsburg, PA, August, 1990.

Appendix 1.

The Multiple Algorithm Robot Control System (MARCS) is currently under development in Goddard Space Flight Center's robotics laboratory. This system is designed to run three different teleoperation algorithms, a Martin Marietta-style algorithm, a JPL teleoperation algorithm, and an algorithm developed at Goddard. The following pages are taken from early design efforts in the project and are presented here only to show a consensus of viewpoint among two labs working on telerobot systems development. The first two figures show the Servo Levels for the manipulator subsystem and the hand-controller subsystem. The last figure shows a preliminary hardware architecture to support these two Servo Levels.



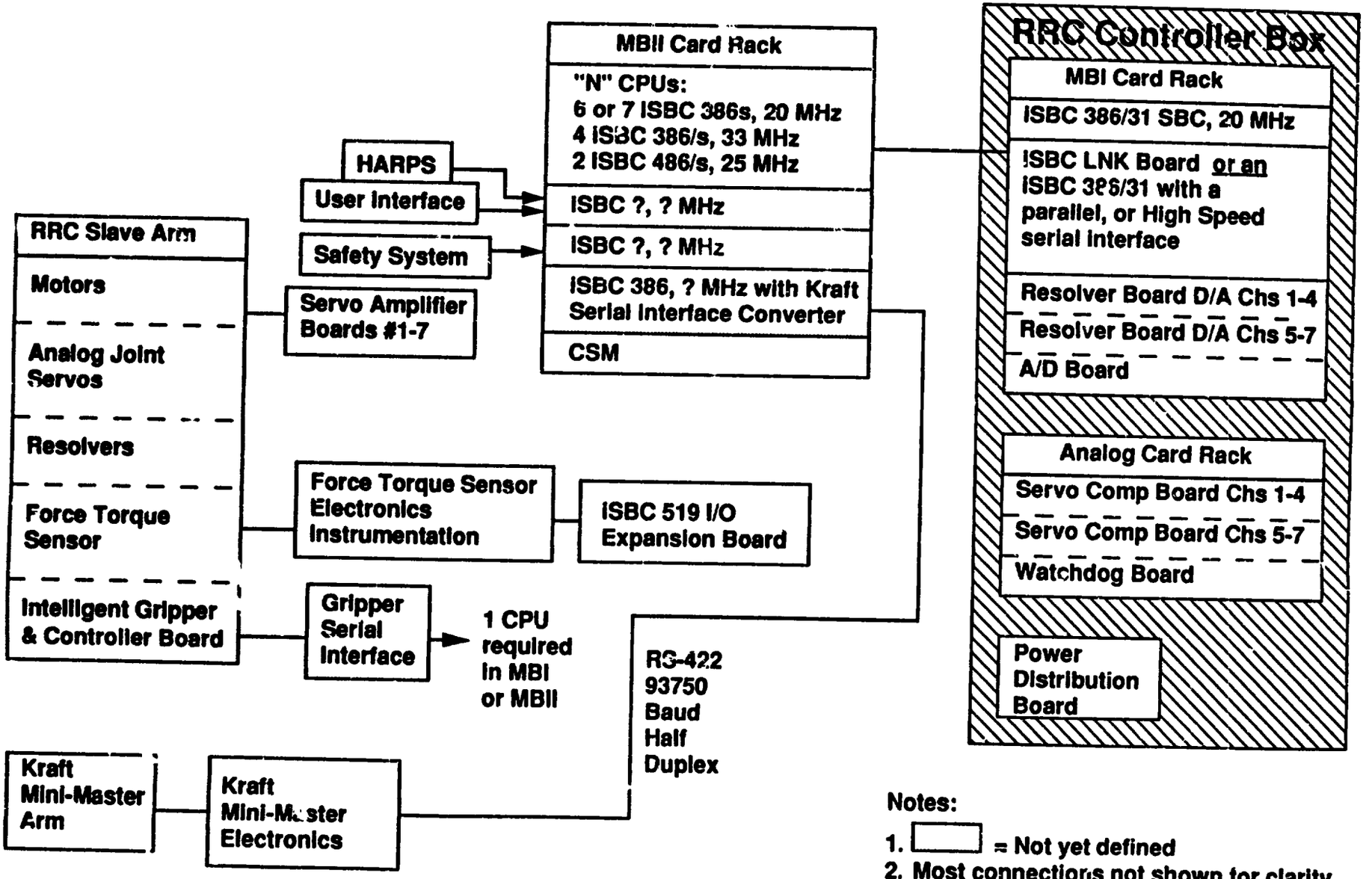
TITLE	DATE	REV	ORIG BY	PAGE NO.
Robot Servo Control	4/9/90	2	Swetz	1 of 2



TITLE	DATE	REV	ORIG BY	PAGE NO.
Servo Operator Control	4/9/90	2	Swetz	2 of 2

Hardware Requirements - 5

Essential MARCS Hardware Components



Notes:

1. ☐ = Not yet defined
2. Most connections not shown for clarity