NIST Technical Note [illegible]

NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)

[illegible]

Robot [illegible]
[illegible]
[illegible]
[illegible]

Sponsored by:
NASA [illegible]
[illegible]

[illegible]

April [illegible]

[illegible]

U.S. Department of [illegible]
Robert A. [illegible]

National Institute of [illegible]
[illegible]

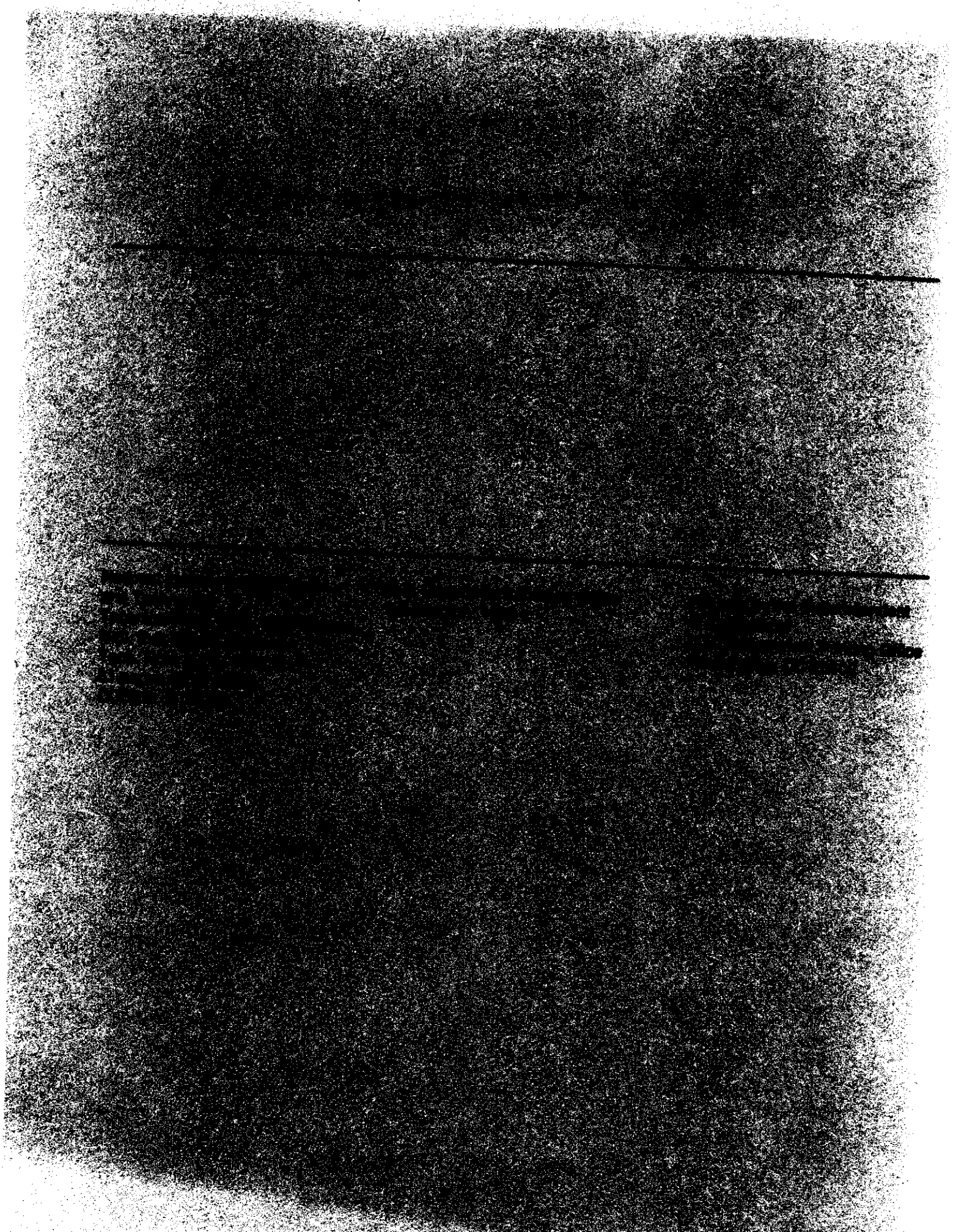


TABLE OF CONTENTS

NASA/NBS
STANDARD REFERENCE MODEL FOR
TELEROBOT CONTROL SYSTEM ARCHITECTURE
(NASREM)

1. INTRODUCTION

1.1 Scope of This Document

This document describes the NASA/NBS Standard Reference Model (NASREM) Architecture for the Space Station Telerobot Control System. It defines the functional requirements and high level specifications of the control system for the NASA Space Station IOC Flight Telerobot Servicer. It is to be used as a reference document for the functional specification, and a guideline for the development of the control system architecture, of the IOC Flight Telerobot Servicer.

The NASA/NBS Standard Reference Model (NASREM) defines a logical computing architecture for telerobotics, derived from a number of concepts developed in previous and on-going research programs, including the NASA OAST telerobotics research program at JPL, Langley Research Center, Oak Ridge National Laboratory, Johnson Spaceflight Center, Marshall Spaceflight Center, the Artificial Intelligence program at Ames Research Center, the Intelligent Task Automation program sponsored by DARPA and Wright Patterson Air Force Base, supervisory control concepts pioneered by Sheridan at MIT, and the hierarchical control system developed for the Automated Manufacturing Research Facility at the National Bureau of Standards [1-34]. This latter system was developed for simultaneously controlling a number of robots, machine tools, and materials transport systems in a machine shop, and has recently been extended to the control of multiple autonomous undersea vehicles, and to battle management for SDI.

The NASREM telerobot control system architecture described in this document incorporates many AI concepts such as goal decomposition, hierarchical planning, model driven image analysis, blackboard systems, and expert systems. It integrates these into a framework that also includes modern control concepts such as multivariate state space control, reference model adaptive control, dynamic optimization, and learning systems. The framework also readily accommodates concepts from operations research, differential games, utility theory, and value driven reasoning [35-46].

The NASREM telerobot control system architecture defines a set of standard modules and interfaces which facilitate software design, development, validation, and test, and make possible the integration of telerobotics software from a wide variety of sources. Standard interfaces also provide the software hooks necessary to incrementally upgrade future Flight Telerobot Systems as new capabilities develop in computer science, robotics, and autonomous system control.

The NASREM telerobot architecture has been reviewed and discussed in three meetings of the FTS Architecture Working Group. This version incorporates the revisions and extensions suggested by the members of that working group.

1.2 Background

The NASA/NBS Standard Reference Model telerobot control system architecture is hierarchically structured into multiple layers, as shown in Figure 1a, such that a different fundamental mathematical transformation is performed at each layer. At layer one, coordinates are transformed and outputs are servoed. At layer two, mechanical dynamics are computed. At level three obstacles are observed and

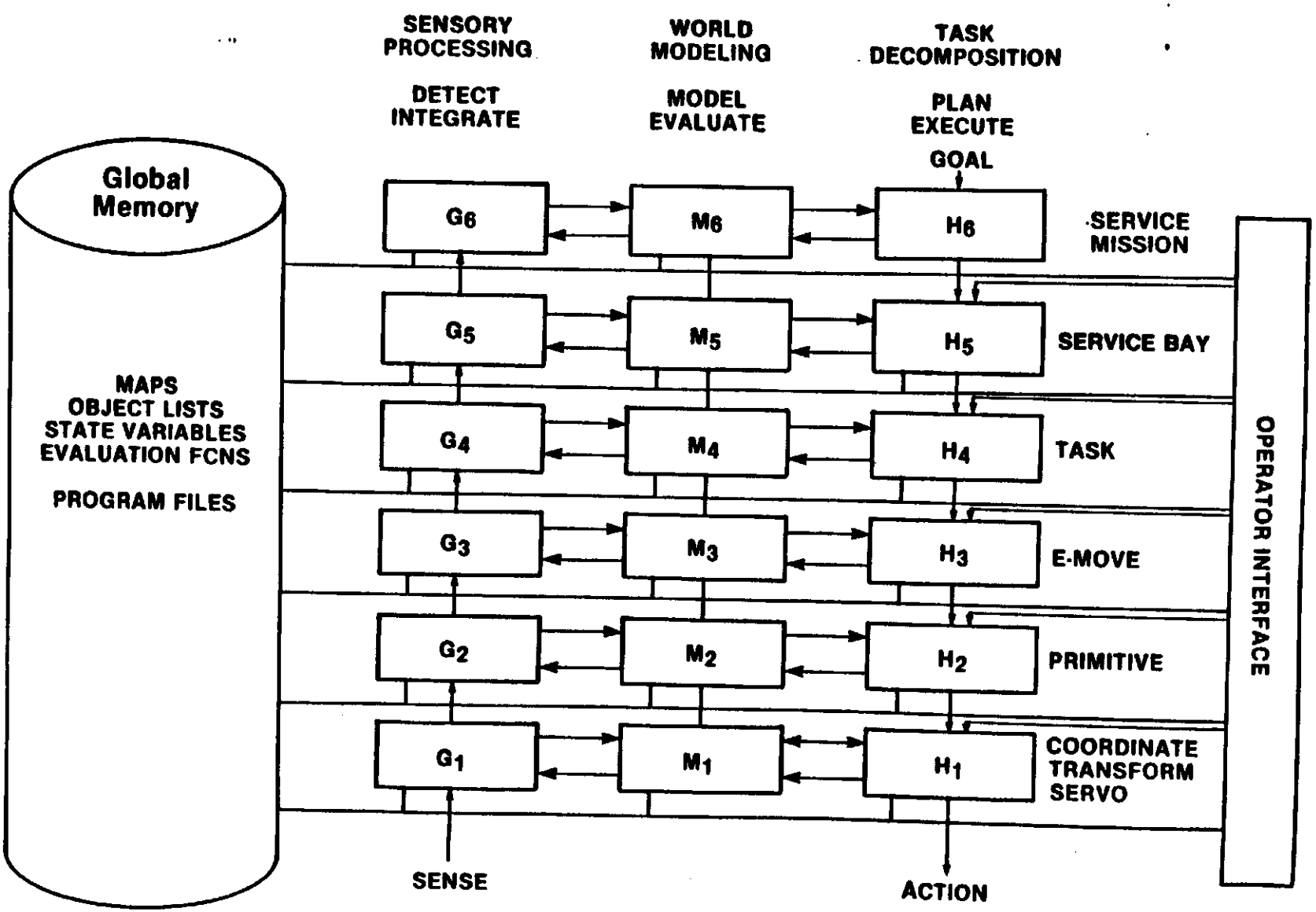


FIGURE 1a: **A hierarchical control system architecture for telerobots.**

avoided. At level four, tasks on objects are transformed into movements of effectors. At level five tasks tasks on groups of objects are sequenced and scheduled. At level six objects are batched into groups, resources are assigned to worksites, and parts and tools are routed and scheduled between worksites. Higher levels are possible, and have been implemented and studied in the NBS AMRF. These levels are described in greater detail in Sections 3 and 8.

Hierarchical control is not new. It has been used by military, government, and business bureaucracies for centuries. The application of hierarchical control to real-time computer control systems is a recent development which is most mature in industrial computer integrated manufacturing systems. Real-time hierarchical control concepts are also now being implemented in advanced aircraft flight controllers and modern smart weapons systems.

The NASREM telerobot control architecture is also horizontally partitioned into three sections: Task Decomposition, World Modeling, and Sensory Processing. Task decomposition includes planning and task monitoring, value driven decisions, servo control, and interfaces for operator input. World modeling includes Computer-Aided-Design (CAD) models of objects and structures, maps of areas and volumes, lists of objects with their features and attributes, and tables of state variables which describe both the system and the environment. Sensory processing includes signal processing, detection of patterns, recognition of features, objects, and relationships, and correlation and differencing of observations vs. expectations. These functions are described more thoroughly in Sections 2, 5, 6, and 8.

1.3 Hierarchical vs. Horizontal

The NASREM telerobot control architecture has both hierarchical and horizontal communications.

The flow of commands and status feedback is hierarchical. High level commands, or goals, are decomposed both spatially and temporally through a hierarchy of control levels into strings and patterns of subcommands. Each task decomposition module represents a node in a command tree. It receives input commands from one and only one supervisor, and outputs subcommands to a set of subordinate modules at the next level down in the tree. Outputs from the bottom level consist of drive signals to motors and actuators.

The sharing of data is horizontal between modules at the same level. Both in terms of volume and bandwidth, there is much more information flowing horizontally between modules at the same level than flowing vertically between levels along the branches of the command tree.

The task decomposition modules at each level in the control hierarchy are made up of a number of job assignment modules, planner modules, and executor modules. Each of these communicates voluminously with a world modeling module at the same level.

The world modeling module is made up of a set of processes that maintain geometric models of the workspace, update lists of objects and their attributes, keep state variables current, generate predictions and compute evaluation functions based on hypothesized or planned actions. Each world modeling module is constantly in communication with a set of sensory processing modules which compute spatial and temporal correlations, differences, convolutions, and integrations; comparing predictions generated by the corresponding level modeling module with observations detected by lower level sensory processing modules.

The sensory processing modules are programmed to filter, detect, recognize, measure, and otherwise extract from the sensory data stream the information necessary to keep the world model at each level updated. The flow of information between sensory processing, world modeling, and task

decomposition modules at each level is mainly horizontal and the bandwidth of the information flowing horizontally completely dwarfs the amount flowing vertically.

The sharing of information between world model, task decomposition, and sensory processing modules at the same level is, however, not necessarily strictly horizontal. All input and output variables to all of the modules at all levels are globally defined, and exist in a global memory. This facilitates interaction with the user, promotes good programming practice, and allows debugging of the system. Conceptually, there is no logical restriction prohibiting any module at any level from making a query of, or obtaining information from, the world model at any level. These variables are available to any process that wishes to post a query or read a value. There of course, may be practical limitations dictated by the physical implementation of the distributed computing hardware. The global memory may thus be distributed over a number of physically distinct memories.

There exists a communications process which allows shared access to information in global memory. It is transparent to the computing modules and makes the global memory appear to the various computing modules as if it were a single common memory.

Although the flow of commands through the hierarchical task decomposition command tree is strictly enforced (no telerobot and no command subtree ever reports to more than one supervisor at any instant in time), the command tree is not necessarily stationary. For example, at the Satellite Service Bay level and above, the command tree may be reorganized from time to time so as to reassign telerobots to different service bays for various tasks. In the AMRF, this idea corresponds to the "virtual cell" which is described by McLean [47]. When the command tree is reconfigured it is done instantaneously, and the control structure always remains a tree; with one root node at the top, where the longest term strategy is pursued and the highest level priority is determined.

2. A FUNCTIONAL SYSTEM ARCHITECTURE

The highest level block diagram of the NASREM telerobot architecture is shown in Figures 1a and 1b. The NASREM control system architecture is a three legged hierarchy of computing modules, serviced by a communications system and a global memory, and interfaced to operator and programmer workstations.

The task decomposition H modules perform real-time planning and task monitoring functions, and decompose task goals both spatially and temporally. The sensory processing G modules filter, correlate, detect, and integrate sensory information over both space and time so as to recognize and measure patterns, features, objects, events, and relationships in the external world. The world modeling M modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in global memory. Global memory is a database which contains the system's best estimate of the state of the external world. The world modeling modules keep the global memory database current and consistent.

2.1 Task Decomposition - H Modules (Plan, Execute)

The task decomposition hierarchy consists of H modules which plan and execute the decomposition of high level goals into low level actions. Task decomposition involves both a temporal decomposition (into sequential actions along the time line) and a spatial decomposition (into concurrent actions by different subsystems).

Each H module at each level consists of three sublevels:

1) a job assignment manager JA,
2) a set of planners PL(i), and
3) a set of executors EX(i).

These three sublevels decompose the input task into both spatially and temporally distinct subtasks as shown in Figure 2.

2.2 World Modeling - M Modules (Remember, Estimate, Predict, Evaluate)

Def. 1: World Model
The "world model" is the system's best estimate and evaluation of the history, current state, and possible future states of the world, including the states of the system being controlled. The "world model" includes both the M modules and a knowledge base stored in global memory where state variables, maps, lists of objects and events, and attributes of objects and events are maintained.

By this definition, the world model corresponds to what is widely known throughout the artificial intelligence community as a "blackboard" [48].

The world modeling leg of the hierarchy consists of M modules which model (i.e. remember, estimate, predict) and evaluate the state of the world.

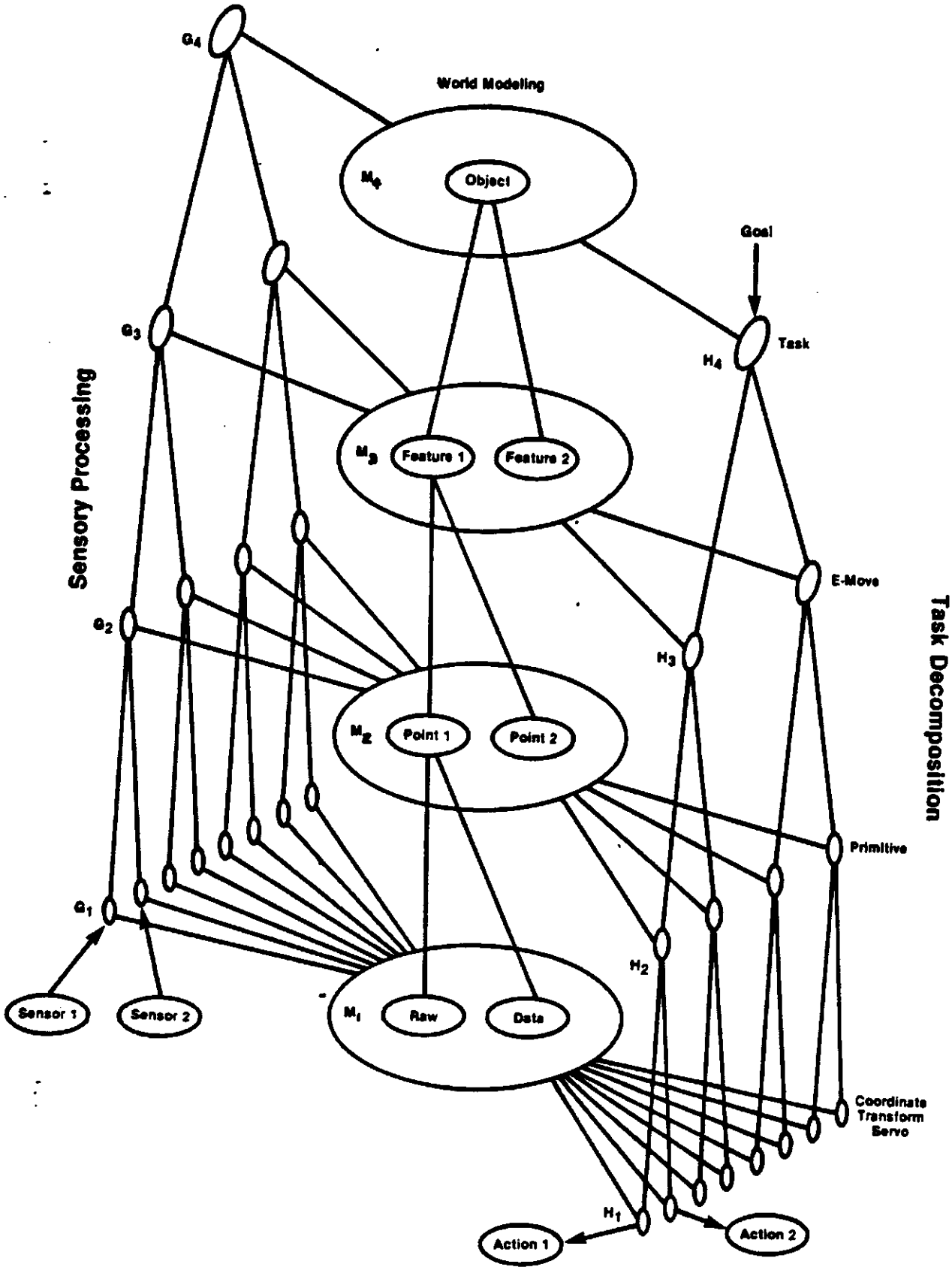

G4
World Modeling
M4
Object
Goal
G3
H4
Task
Sensory Processing
M3
Feature 1
Feature 2
E-Move
G2
H3
M2
Point 1
Point 2
Task Decomposition
Primitive
G1
H2
Sensor 1
Sensor 2
M1
Raw
Data
Coordinate
Transform
Servo
H1
Action 1
Action 2

FIGURE 1b:

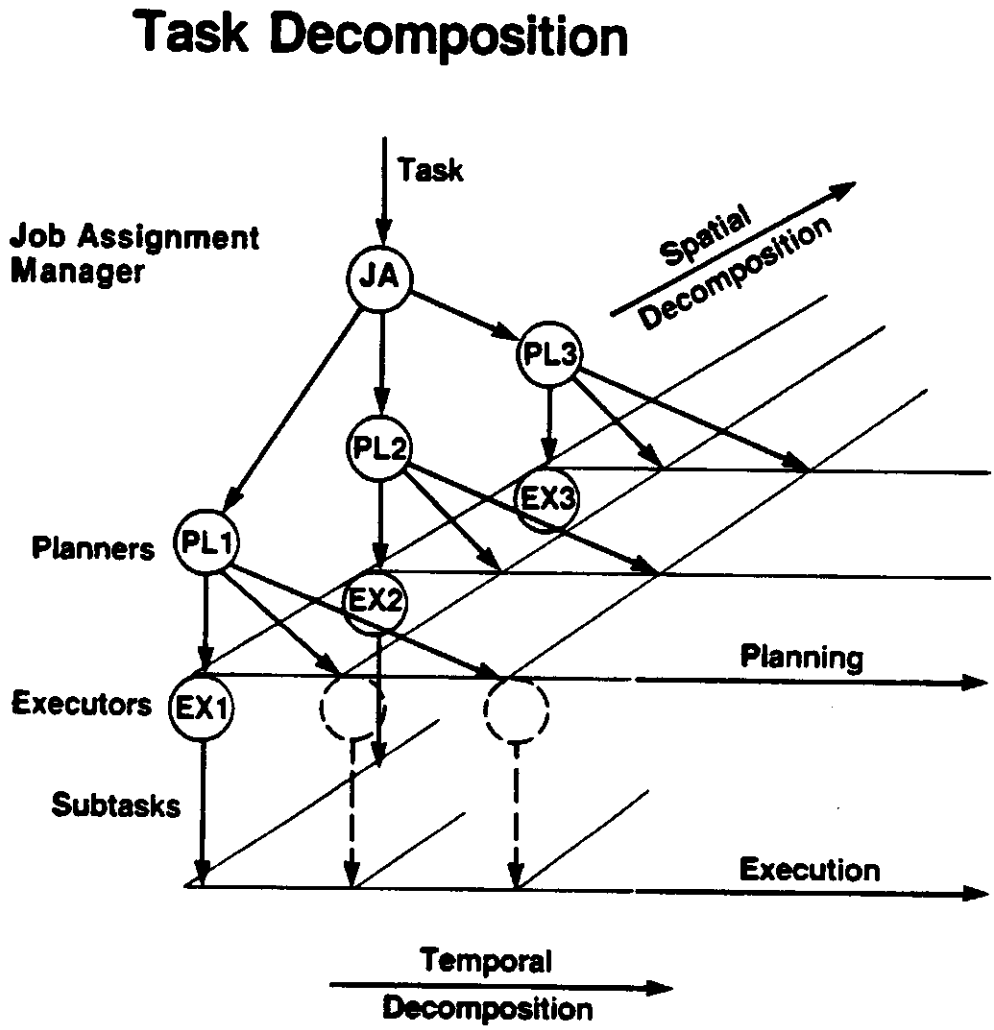


FIGURE 2: The job assignment JA performs a spatial decomposition of the task. The planners pl (j) and executors EX (j) perform a temporal decomposition.

As shown in Figure 3, the M modules at various levels:

2.2.1 Maintain Knowledge Base

Maintain the global memory knowledge base, keeping it current. The M modules update the knowledge base based on correlations and differences between model predictions and sensory observations. This is shown in more detail in Figure 4.

2.2.2 Provide Predictions

Provide predictions of expected sensory input to the corresponding G modules, based on the state of the task and estimates of the external world. This is shown in Figure 4.

2.2.3 What Is?

Answer "What is?" questions asked by the planners and executors in the corresponding level H modules. The task executor requests information about the state of the world, and uses the answers to monitor and servo the task, and/or to branch on conditions to subtasks that accomplish the task goal. This is shown in more detail in Figure 5.

2.2.4 What If?

Answer "What if?" questions asked by the planners in the corresponding level H modules. As shown in Figure 6, the M modules predict the results of hypothesized actions.

2.2.5 Evaluate Situations

Evaluate the current situation and potential future consequences of hypothesized actions by applying evaluation functions to current states and to future states expected to result from hypothesized actions. The evaluation functions define a set of values over the state-space defined by state variables in the global memory. These evaluation functions can be used to compute priorities, cost-benefit values, risk estimates, and pay-off values of states of the world. Thus, working together with the world model, the planners are able to search the space of possible futures, and choose the sequence of planned actions that produce the best evaluation. The executors are able to apply value judgments to moment by moment behavioral decisions.

2.3 Global Memory

Def. 2: Global memory is the database wherein is stored knowledge about the state of the world including the internal state of the control system.

2.3.1 Contents of Global Memory

The knowledge in the global memory consists of:

a) Maps which describe the spatial occupancy of the world.

A map is a spatially indexed database showing the relative position of objects and regions. At different levels the maps have different resolution. The maps at different levels may be represented in a pyramid structure. Map overlays may also contain value functions such as utility, cost, risk, etc. to be used in path planning and safety.

World Modeling

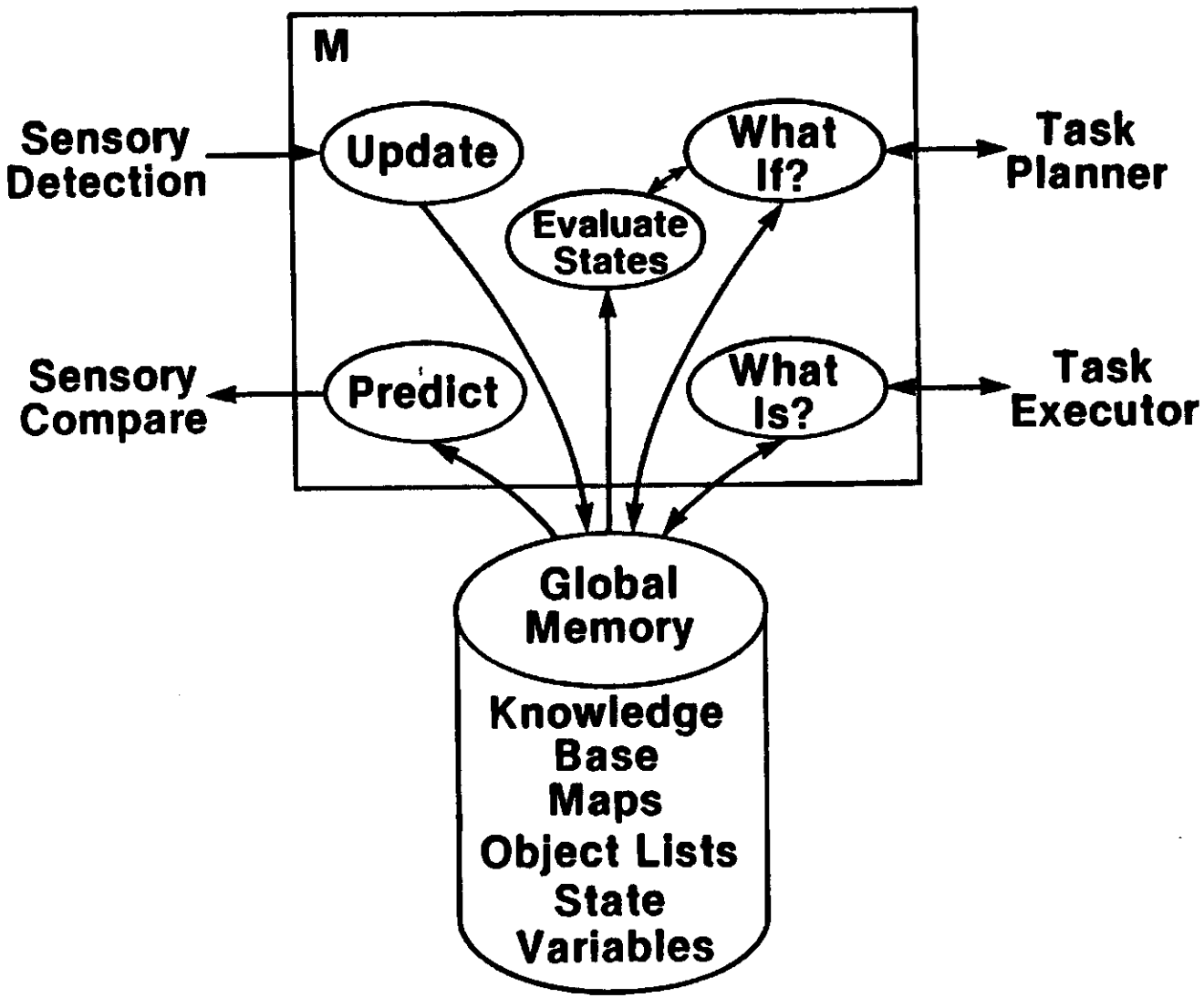


FIGURE 3: Functions performed by M modules in the world model.

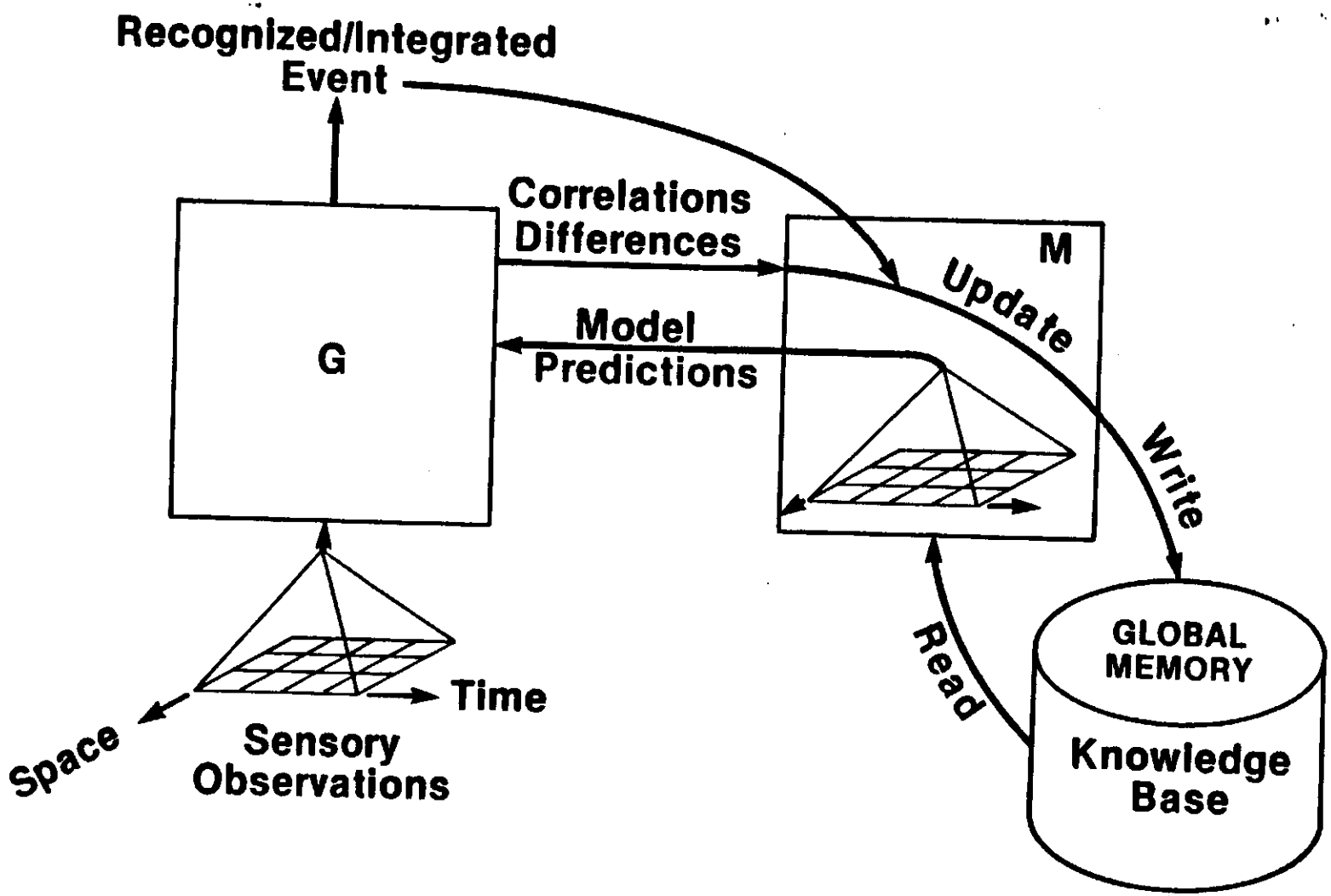


FIGURE 4: **Role of M module in predicting sensory input and in up-dating knowledge base based in correlations and differences between predicting and observations.**

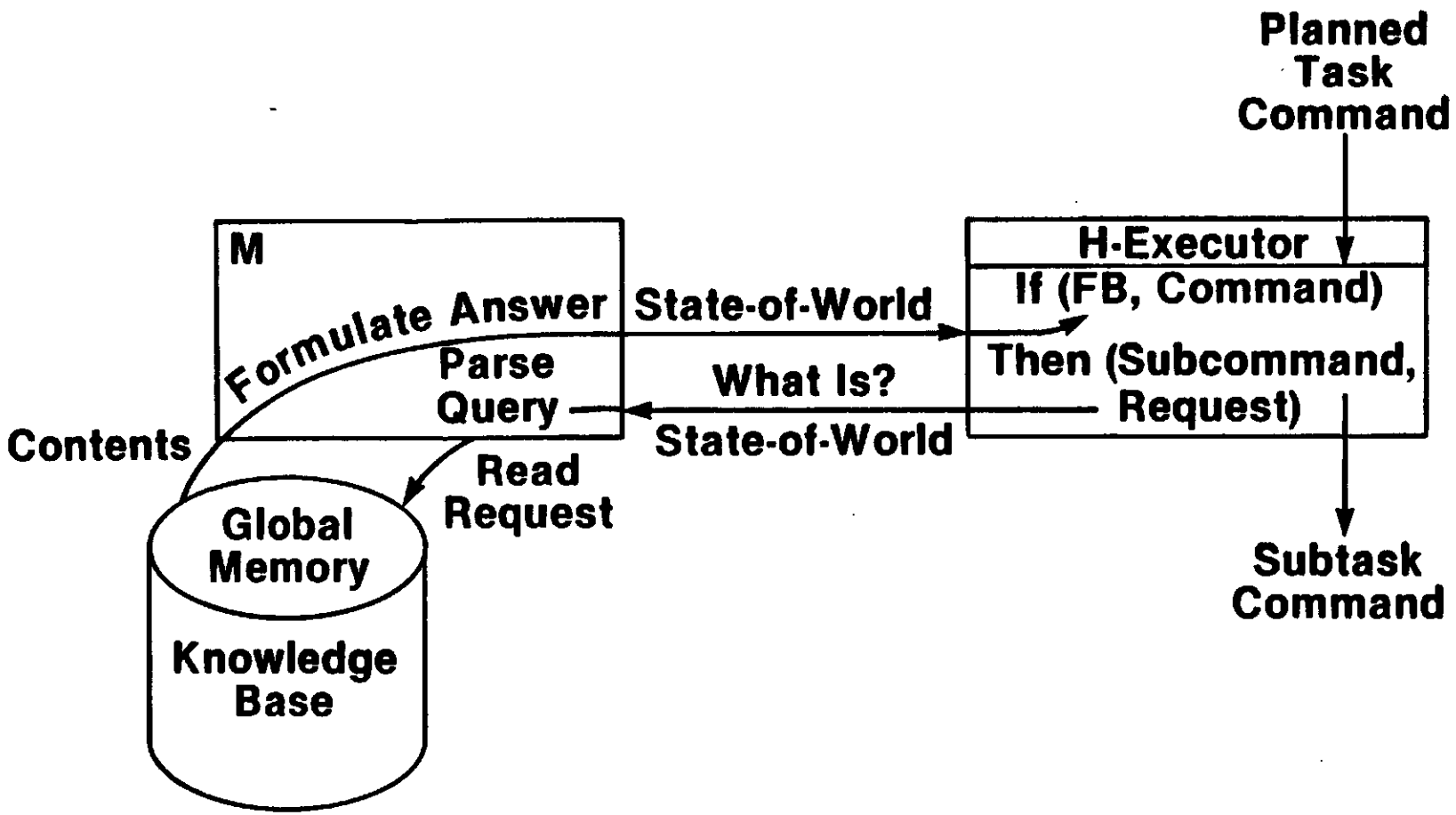


FIGURE 5: **Role of M modules in responding to H module executor "What Is?" questions.**

Role of World Model in Planning

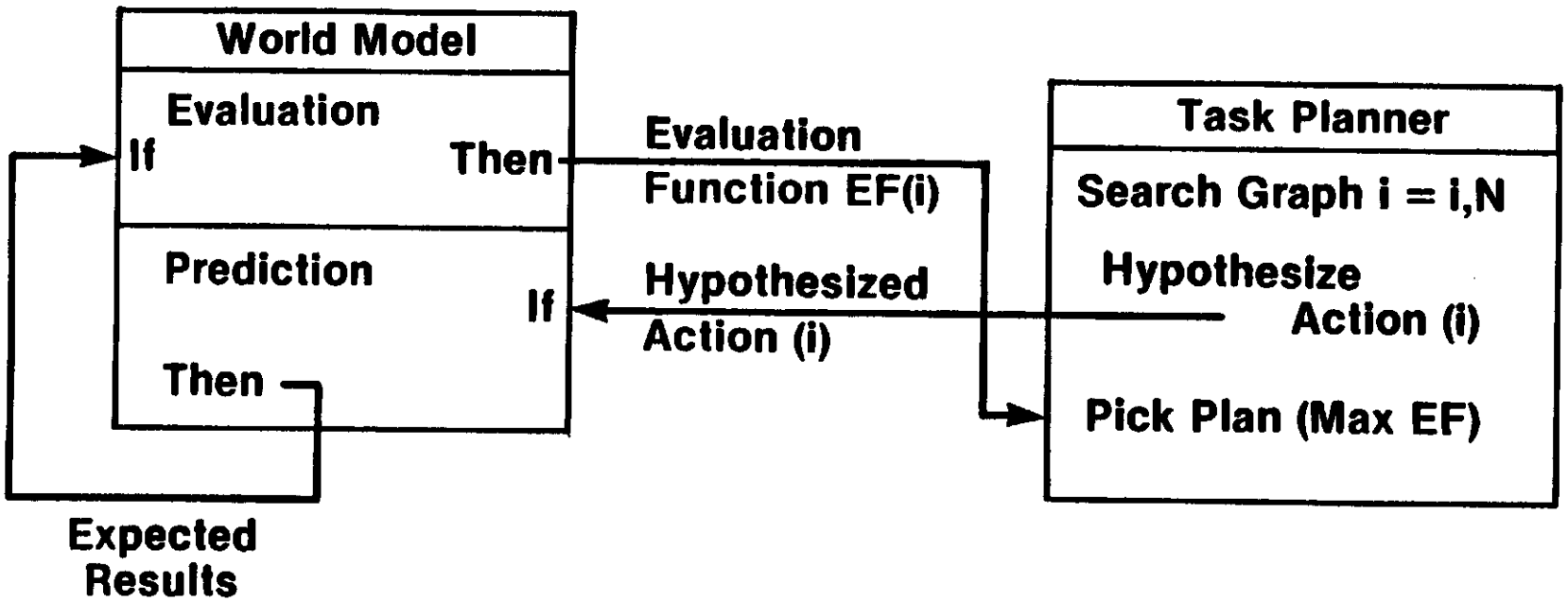


FIGURE 6: Role of world model in planning hypothesized actons are "What If?" questions.

b) Lists of objects, features, relationships, events, and frames containing their attributes.

This database is indexed by name. Object and feature frames contain information such as position, velocity, orientation, shape, dimensions, reflectance, color, mass, and other features of interest. Event frames contain information such as start and end time, duration, type, cost, payoff, etc. Recognized objects and events may also have confidence levels, and degrees of believability and dimensional certainty.

At different levels, object frames have different levels of detail and spatial resolution, and event frames have different levels of temporal resolution. Typically, class labels of the lower level are considered as primitives for the higher level.

c) State variables which identify particular situations.

The state variables in global memory are the system's best estimate of the state of the world, including both the external environment and the internal state of the H, M, and G modules. Data in global memory is available to all modules at all levels of the control system.

2.3.2 Implementation of Global Memory

Global memory is not necessarily implemented as a physically contiguous single block of memory. Global memory may, in practice, be distributed over a variety of media in physically disparate locations. Parts of global memory may be on dual- ported RAM located on H, M, or G module processor boards on a multiprocessor bus. Other parts may be on disk or bubble memory at a variety of physical locations, on the telerobot, in the space station, or even on the ground. What is important is that the variables in global memory are globally defined, that they can be called symbolically by the computational modules that either read or write them, and they can be accessed with acceptable delay. It is recognized that each level of the hierarchy has substantially different requirements for delay because each runs roughly one order of magnitude slower than the level below. This must be taken into consideration during the implementation of global memory so that information can flow as required. There should be an on-line data dictionary so that numerical values can be bound to symbolic names during execution.

It, of course, is important that there exist automatic protection mechanisms to prevent array variables from being simultaneously read and written, and to prevent corruption of data by program bugs such as indexing outside of arrays, etc. If global memory is distributed such that multiple copies of data exist, then there must be mechanisms for assuring consistency between multiple copies. Implementation must also take into account the timing requirement of the computational modules that make use of the data for real-time control. Typically, every global variable has only one process that writes it, while many processes may read it. This greatly simplifies the problem of preventing inadvertent corruption of global memory.

2.4 Sensory Processing - G Modules (Filter, Integrate, Detect, Measure)

The sensory processing leg of the hierarchy consists of G modules which recognize patterns, detect events, and filter and integrate sensory information over space and time. As shown in Figure 7, the G modules also consist of three sublevels which:

1) compare observations with predictions
2) integrate correlation and difference over time
3) integrate correlation and difference over space

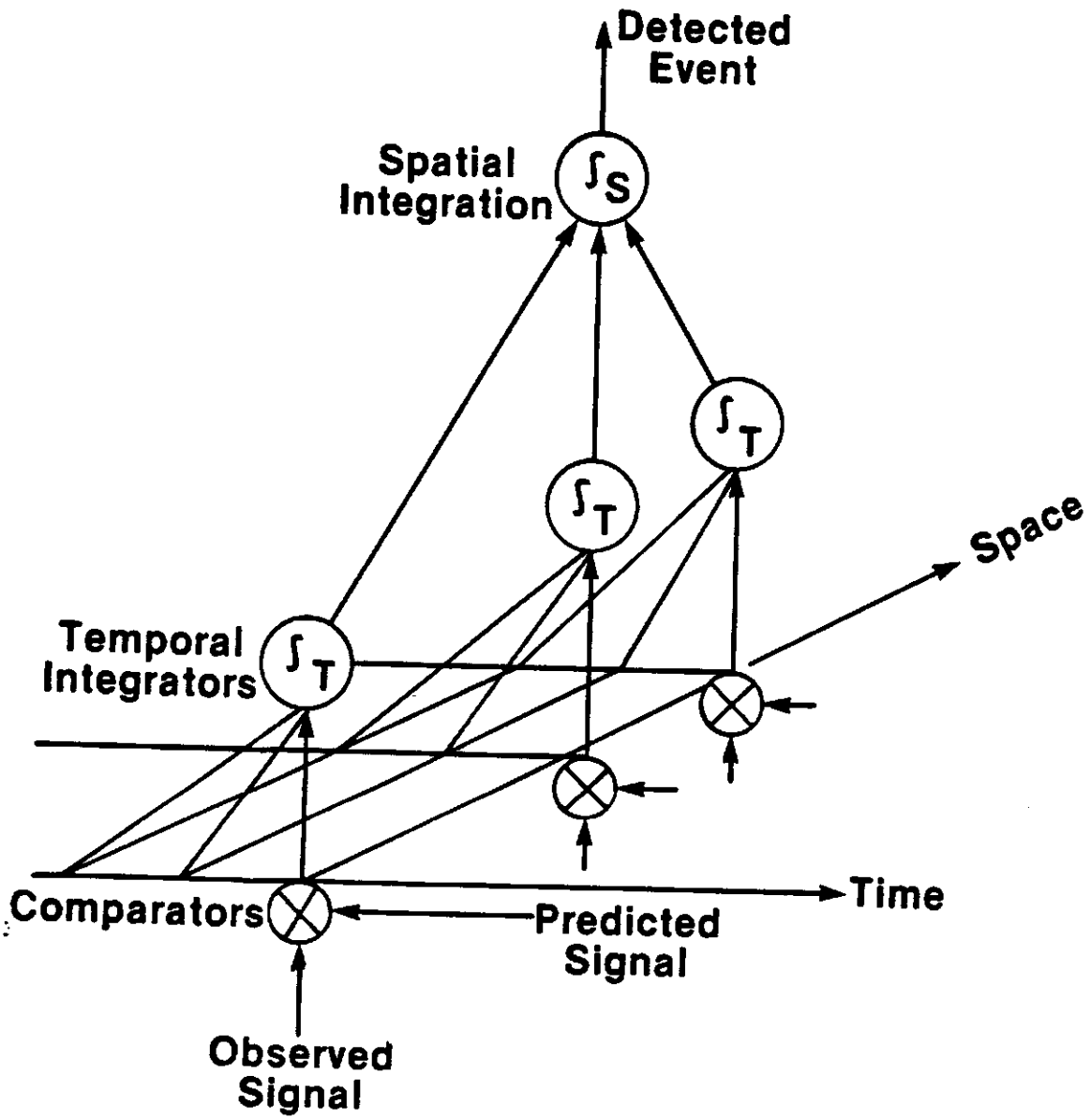


FIGURE 7: Each sensory processing G modules performs a comparison and both temporal and spatial integration.

These spatial and temporal integrations fuse sensory information from multiple sources over extended time intervals. Newly detected or recognized events, objects, and relationships are entered by the M modules into the world model knowledge base in global memory, and objects or relationships perceived to no longer exist are removed. The G modules also contain functions which can compute confidence factors and probabilities of recognized events, and statistical estimates of stochastic state variable values.

2.5 Operator and Programmer Interfaces(Control, Observe, Define Goals, Indicate Objects, Edit Programs and Data)

The control architecture defined here has operator and programmer interfaces at each level in the hierarchy.

2.5.1 Operator Interface

The operator interface provides a means by which human operators, either in the space station or on the ground, can observe, supervise, and directly control the telerobot. Each level of the task decomposition hierarchy provides an interface where the human operator can assume control. The task commands into any level can be derived either from the higher level H module, or from the operator interface. Using a variety of input devices such as a joystick, mouse, trackball, light pen, keyboard, voice input, etc., a human operator can enter the control hierarchy at any level, at any time of his choosing (within restrictions imposed by synchronization and data integrity constraints), to monitor a process, to insert information, to interrupt automatic operation and take control of the task being performed, or to apply human intelligence to sensory processing or world modeling functions. (Operator interrupts will not literally be allowed at "any time", but at frequent points in time where operator interrupts can be synchronized to coincide with state clock increments or subtask completion events.)

The operator interface terminal provides the input devices (joystick, mouse, trackball, light pen, keyboard, or voice input) whereby the human operator can input the information needed for designating tasks at that level. The operator interface processor provides the necessary translators and string generators to format human inputs into the proper format, and to verify, validate, and synchronize them with ongoing processes at the appropriate level of levels. The operator interface processor also provides the necessary synchronization mechanisms necessary so that automatic operations can be resumed from the point in time and space where the human operator leaves off, or restart automatic operations from the point where the human interrupted.

The sharing of command input between human and autonomous control need not be all or none. The combining of automatic and teleoperator modes can span an entire spectrum from the one extreme, where the operator takes complete control of the system from a given level down so that the levels above the operator are disabled, to the autonomous mode where the operator simply loads a given program and puts the telerobot on automatic. In between these two ends of the spectrum, is a broad range of interactive modes where the operator supplies some control variables and the autonomous system provides others. For example a human might control the orientation of a camera while the robot automatically translates the same camera through space. It is also within the state of the art to compute control inputs by a polynomial, which multiplies human and automatic input variables by relative percentages, and sums the result so that both human and autonomous inputs share in influencing the position, velocity, force, and stiffness of the manipulator end effector. Even in cases where the operator takes complete control, some of the higher level safety and fault protection functions should remain in operation.

2.5.2 Operator Control Interface Levels

If the human operator enters the task decomposition hierarchy in the middle of level 1 (at the input to the servos), he/she must use a replica master, or individual joint position, rate, or force controllers.

If the human enters the task decomposition hierarchy above level 1, he/she can use a joy stick to perform resolved motion force/rate control.

If the human enters above level 2, he/she can simply indicate safe motion pathways, and the robotic system will compute dynamically efficient movements.

If the human enters above level 3, he/she can graphically or symbolically define key poses, or using a menu, call for elemental manipulator or telerobot transport movements (E-moves) such as <position-telerobot-at X>, <move-gripper-to-pose Y>, <approach-grip-point Z>, etc. This may be done using an interactive graphics display with a joystick, mouse, track ball, light pen, or voice input.

If the human enters above level 4, he/she can indicate objects, and call for tasks to be done on those objects, such as <remove- module M>, <insert-refueling-hose-in R>, <fixture-object X in- clamp W>, etc. This may be done using cursors and graphic images overlaid on television images.

If the human enters above level 5, he can reassign telerobots to different service bays, insert, monitor, or modify plans that describe servicing task sequences, define repair part and tool kits, etc.

If the human enters above level 6, he can reconfigure servicing mission priorities, change servicing requirements, enter or delete jobs, and change the mission operations schedule.

The operator control interface thus provides mechanisms for entering new instructions or programs into the various control modules. This can be used on-line for real-time supervisory control, or in a background mode for altering autonomous telerobot plans before autonomous execution reaches that part of the plan. The operator control interface can also provide look- ahead simulation of planned moves so as to analyze the consequences of a prospective motion command before it is executed.

2.5.3 Operator Monitoring Interfaces

The operator interfaces allow the human the option of simply monitoring any level. Windows into the global memory knowledge base permit viewing of maps of service bay layout, geometric descriptions and mechanical and electrical configurations of satellites, lists of recognized objects and events, object parameters, and state variables such as positions, velocities, forces, confidence levels, tolerances, traces of past history, plans for future actions, and current priorities and utility function values. These may be displayed in graphical form, for example using dials or bar graphs for scalar variables, shaded graphics for object geometry, and a variety of map displays for spatial occupancy. Time traces can be represented as time line graphs, or as stick figures with multiple exposure and time decay. State graphs with windows into nodes and edges can be used to display the state of the various modules in the control system and the conditions required for state transitions.

Sequences of past actions or plans for future action can be represented as state graphs, with windows into nodes to display the state of the various modules in the control system at different times, and windows into edges to display the conditions required for state transitions. Geography and spatial occupancy can be displayed as a variety of maps, vectors, or stick figures, or shaded graphics images. Object geometry can be represented as wire frames or 3-dimensional solid objects. The operator may also have a direct television image of the robot's environment with graphics overlays which display the degree of correlation between what the robot believes is the state of the world, and what the human

operator can observe with his own eyes.

2.5.4 Operator Sensory Processing/World Modeling Interfaces

The operator interface may also permit interaction with the sensory processing and/or world modeling modules. For example, an operator using a video monitor with a graphics overlay and a light pen or joystick might provide human interpretative assistance to the vision/world modeling system. The operator might interactively assist the model matching algorithms by indicating with a light pen which features in the image (e.g. edges, corners) correspond to those in a stored model. Alternatively, an operator could use a joystick to line up a wireframe model with a TV image, either in 2-D or 3-D. The operator might either move the wireframe model so as to line up with the image, or move the camera position so as to line up the image with the model. Once the alignment was nearly correct, the operator could allow automatic matching algorithms to complete the match, and track future movements of the image.

The human operator can thus monitor, assist, and if he wishes, interrupt autonomous operation, at any time (within the restrictions noted above), for any reason, at any desired level, to take control, to stop the robot, to slow it down, to back it up, or to substitute the human's judgment by directly entering commands or other information to replace what the robot had otherwise planned to do.

2.5.5 Programmer Interface

The programmer interface allows a human programmer to load programs, monitor system variables, edit commands and data, and perform a broad range of debugging, test, and program modification operations.

There are a variety of levels of programmer interface corresponding to various levels of programming skill and system change authority. The lowest level allows only monitoring of system variables. The next higher level permits debugging tests to be performed while applications programs are running. The third level allows on-line editing of data variables and applications program code. The fourth and highest level allows editing of the FTS operating system itself.

Both the operator and programmer interface formats could be defined in ASCII strings, so that information flowing either direction can be easily read by either man or machine. This convention greatly facilitates debugging and system integration. However, other information formats which follow an object oriented approach are also applicable.

2.6 Safety System

The FTS control system should incorporate a safety system which can prevent the FTS system from entering forbidden volumes, both in physical space and in state space. This safety system should always be operational so as to prevent damage to the robot or surrounding structures or humans during all modes of operation: teleoperation, autonomous, and shared.

The safety system should have access to all the information contained in the world model of the control system, but should also maintain its own world model, updating it with redundant sensors. The safety system should periodically query the control system to test its state and responsiveness. Conversely, the control system should also periodically query the safety system to test it. Observed states should be constantly compared with predicted states and differences noted. If either system detects an anomaly in the other, error messages should be sent, and appropriate action taken.

The sophistication of the safety system may approach, if not exceed, that of the control system itself.

This will be necessary if the safety system is to adequately protect the FTS system from failure of sensors, controls, and operator error.

3. LEVELS IN THE CONTROL HIERARCHY

The NASREM system architecture described here for the Flight Telerobot System is a six level hierarchy, as shown in Figure 8. At each level in this hierarchy a fundamental transformation is performed.

Level 1 transforms coordinates from a convenient coordinate frame into joint coordinates. This level also servos joint positions, velocities, and forces.

Level 2 computes inertial dynamics, and generates smooth trajectories, and servos the end effector in a convenient coordinate frame.

Level 3 decomposes elementary move commands (E-moves) into strings of intermediate poses (or trajectory knot points). E-moves are typically defined in terms of motion of the subsystem being controlled (i.e., transporter, manipulator, camera platform, etc.) through a space defined by a convenient coordinate system. E-move commands may consist of symbolic names of elementary movements, or may be expressed as keyframe descriptions of desired relationships to be achieved between system state variables. E-moves are decomposed into strings of intermediate poses which define motion pathways that have been checked for clearance with potential obstacles, and which avoid kinematic singularities.

Level 4 decomposes object task commands specified in terms of actions performed on objects into sequences of E-moves defined in terms of manipulator motions. Object tasks typically define actions to be performed by a single multiarmed telerobot system on one object at a time. Tasks defined in terms of actions on objects are decomposed into sequences of E-moves defined in terms of manipulator or vehicle subsystem motions. This decomposition checks to assure that there exist motion freeways clear of obstacles between keyframe poses, and schedules coordinated activity of telerobot subsystems, such as the transporter, dual arm manipulators, multifingered grippers, and camera arms. (Coordination at this level consists of scheduling the starting and ending of E-moves, not of instant-by-instant real-time synchronization of movements. This type of tight movement synchronization is accomplished by sharing of system state variables through global memory at levels 1 through 3.) Level 4 corresponds to the Equipment level in the NBS AMRF.

Level 5 decomposes actions to be performed on batches of parts into tasks performed on individual objects. It schedules the actions of one or more telerobot systems to coordinate with other machines and systems operating in the immediate vicinity. For example, Level 5 decomposes service bay action schedules into sequences of object task commands to various telerobot servicers, astronauts, and automatic berthing mechanisms. Service bay actions are typically specified in terms of servicing operations to be performed by all the systems (mechanical and human) in a service bay on a whole satellite. This decomposition typically assigns servicing tasks to various telerobot systems, and schedules servicing tasks so as to maximize the effectiveness of the service bay resources. (Detailed real-time synchronization, again, is accomplished at lower levels.) This level corresponds to the Workstation level in the NBS Automated Manufacturing Research Facility (AMRF).

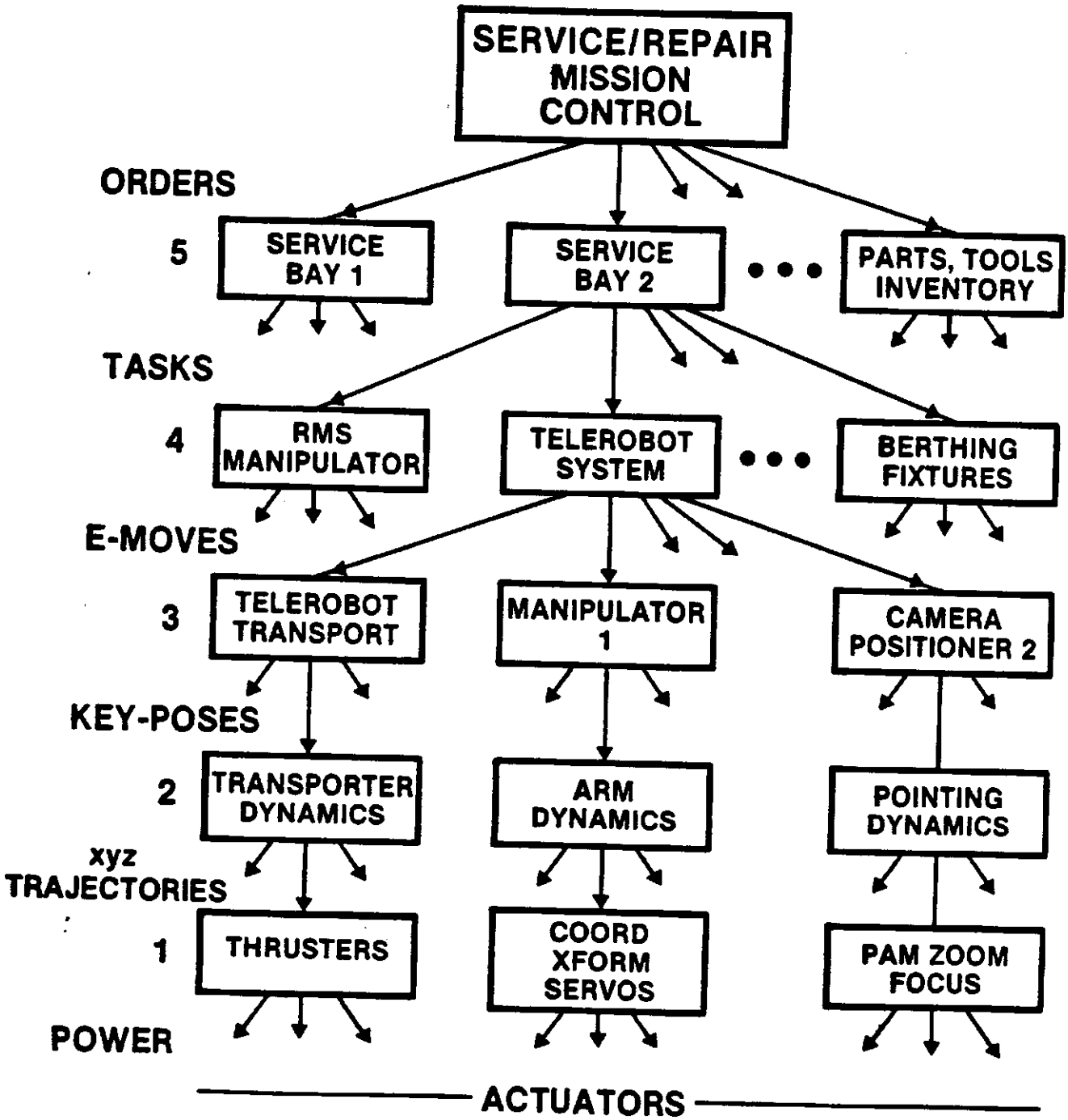


FIGURE 8: **A six level hierarchical control system proposed for telerobots.**

Level 6 decomposes the satellite servicing mission plan into service bay action commands. Mission plans are typically specified in terms of satellite servicing priorities, requirements, constraints, and mission time line. The level 6 decomposition typically assigns satellites to service bays, sets priorities for service bay activities, generates requirements for spare parts and tool kits, and schedules the activities of the service bays so as to maximize the effectiveness of the satellite servicing mission. To a large extent the level 6 mission plans will be generated off line on the ground, either by human mission planners, or by automatic or semiautomatic mission planning methods. This level corresponds to the Cell level in the NBS AMRF.

4. COMMUNICATIONS

The H, M, and G modules at all levels of the NASREM architecture can be viewed as state machines which periodically read input variables, compute some function of their input and state, write output variables, and go to a new state. This requires a communications mechanism by which output variables computed by the various modules at time t=i become available as input variables at time t=i+1.

4.1 Communications Timing

One possible implementation of the NASREM architecture would be a discrete time system in which a state clock is incremented at one millisecond intervals. Between each state clock increment, there would exist a data-transfer/compute cycle as shown in Figure 9.

As soon as the state clock is incremented, the communication process moves data from all output buffers that are ready to the global memory, and from thence to those input buffers that are ready. The routing of data by this communication process may be controlled by the request data in the output buffer of the computation module.

During the compute period, all state variables in global memory are effectively frozen, and represent a snapshot in time of the state of the world at the time of the state clock transition. The H, M, and G functions can read from their input buffers, or from global memory, and compute functions on both local and global variables. Output is stored in output buffers until the next increment of the state clock. Any process that does not finish computing by the end of the compute period will continue until it does finish. Its output buffers will not be moved by the communication process until the process is finished and the output buffers contain new data. Each output buffer therefore carries a ready flag which is set to busy when the computations process begins and is set to ready when the computation process is finished and the output buffers contain fresh data.

A variety of mechanisms for message passing through global memory have been studied. For example, computing modules may communicate with global memory by defining local mailboxes as described in [49,50]. Mailgrams are posted in the mailboxes or read from the mailboxes by the local processes. Delivery of the mailgrams is accomplished by a data administration system which periodically picks up messages from mailboxes that have new information, and deposits the messages at their specified destinations.

Timing requirements for process synchronization vary at different levels of the hierarchy. At level one, synchronization within a few milliseconds is important. At level two, sync within tens of milliseconds is adequate. At level three, sync within tenths of a second; level four, within seconds; level five within tens of seconds; and at level six, sync within minutes is sufficient.

4.2 Communications Through Global Memory

Although there are many methods for implementing a real-time multiprocess communications system, it is conceptually useful to think of passing variables through a global memory. Assume for example, that the NASREM control hierarchy is supported by a global memory in which all state variables, including all input and output variables, are globally defined. It is highly desirable that the global memory have a 32 bit (4 gigabyte) address space. Then communication can consist simply of each computing module reading its inputs from global memory and writing its output back into global memory. Each computing module needs only to know where in global memory its input variables are stored, and where in global memory it should write its output variables. The read and write functions in the system G, M, and H modules then define the communication interfaces.

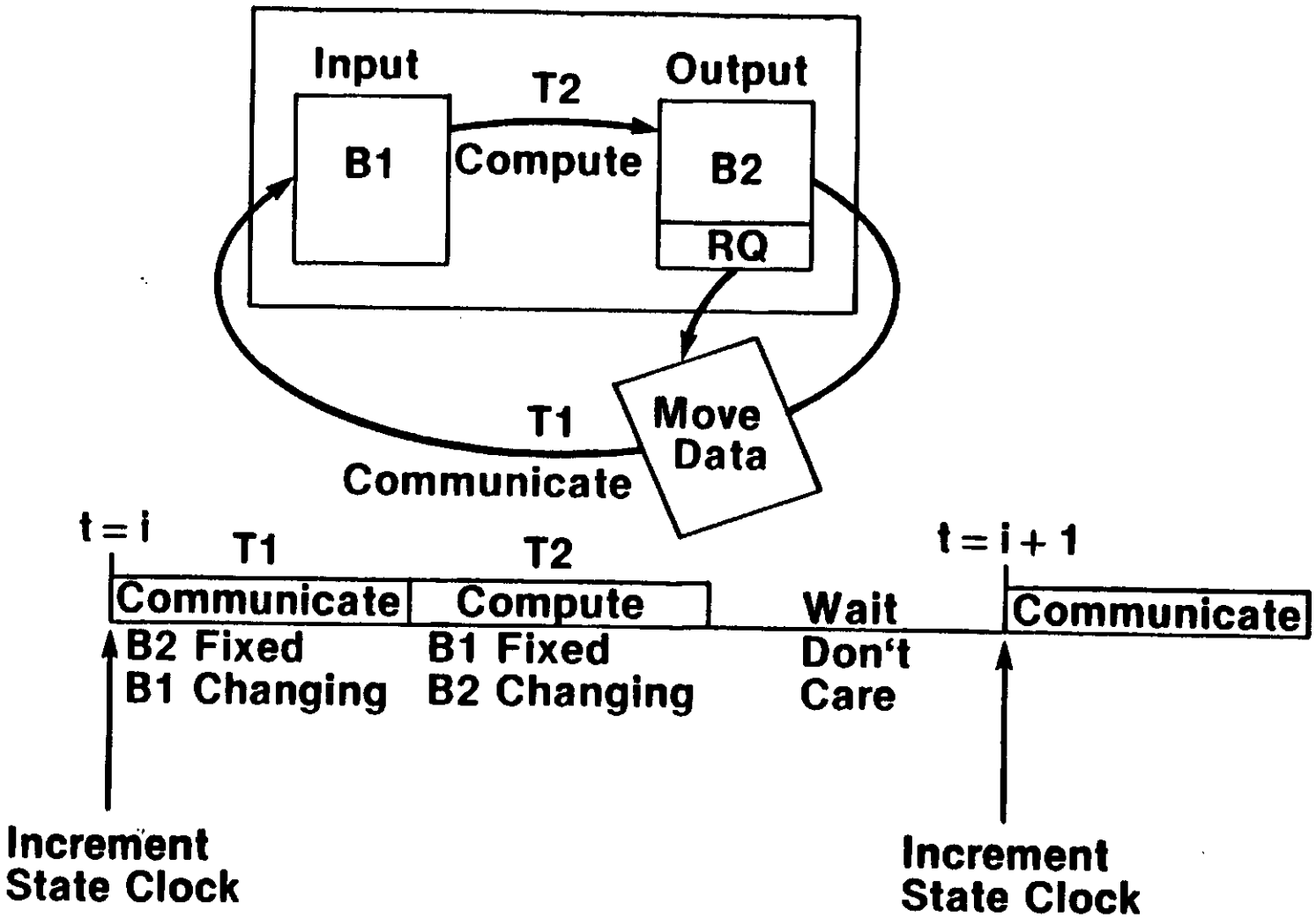

Input
T2
Output
B1
Compute
B2
RQ
T1
Move
Data
Communicate
t = i
T1
T2
t = i + 1
Communicate
Compute
Wait
Communicate
B2 Fixed
B1 Changing
B1 Fixed
B2 Changing
Don't
Care
Increment
State Clock
Increment
State Clock

FIGURE 9:

The global memory approach not only readily supports interprocessor communications, it also provides a clean interface for the operator/programmer workstation. The operator displays read the variables they need from the locations in global memory. If the operator wishes to take control of the system, he writes command variables to the appropriate locations in global memory. The control modules that read from those locations need not know whether their input commands derived from a human operator, or from the next higher level in the autonomous control hierarchy.

If a programmer wishes to monitor or modify a data variable, a sensor input, or a drive signal output, he can simply execute the equivalent of a "PEEK" or "POKE" into the global memory.

The global memory also supports modular development of software. Any system modules can be replaced with a functionally equivalent module by merely respecting the address definitions for the input and output data.

5. DETAILED STRUCTURE OF THE H MODULES

The H module at each level consists of three parts as shown in Figure 10:

1) a job assignment manager JA,
2) one or more planners PL(s), and
3) one or more executors EX(s).

For each level:

5.1 Job Assignment

The job assignment manager JA is responsible for partitioning the task command TC into s spatially or logically distinct jobs to be performed by s physically distinct planner/executor mechanisms. At the upper levels the job assignment module may also assign physical resources against task elements. The output of the job assignment manager is a set of job commands JC(s), s=1, 2, ..., N where N is the number of spatially, or logically, distinct jobs.

5.2 Planners

For each of these job commands JC(s), there exists a planner PL(s) and a executor EX(s). Each planner PL(s) is responsible for decomposing its job command JC(s) into a temporal sequence of planned subtasks PST(s,tt) as shown in Figure 11.

Planning typically requires evaluation of alternative hypothetical sequences of planned subtasks. As shown in Figure 6 the planner hypothesizes some action or series of actions, the world model predicts the results of the action(s) and computes some evaluation function EF(s,tt) on the predicted resulting state of the world. This evaluation function is sometimes called a cost-benefit analysis or objective function. The hypothetical sequence of actions producing the best evaluation function EF(s,tt)max is then selected as the plan PST(s,tt) to be executed by the executor EX(s). We may express the plan PST(s,tt) as the result of a function PL(s) operating on the parameters JC(s) and EF(s,tt)max, i.e.

$$PST(s,tt) = PL(s)\ [JC(s),EF(s,tt)max]$$

where tt is the time sequence index for steps in the plan. tt may also be defined as a dummy time variable, or a running temporal index in planning space.

$$tt = 1, 2, ..., th$$

where th is the value of the tt index at the planning horizon. The planning horizon is defined as the period into the future over which a plan is prepared. Each level of the hierarchy has a planning horizon of one or two expected input task time durations. The replanning interval should be one order of magnitude less than the planning horizon (or about equal to the expected output subtask time duration). Thus the planning horizon grows exponentially at each successively higher level of the hierarchy as illustrated in Figure 12.

5.3 Executor

Each executor EX(s) is responsible for successfully executing the plan PST(s,tt) prepared by its respective planner PL(s). If all the subtasks in the plan PST(s,tt) are successfully executed, then the goal of the original task will be achieved. The executor operates by selecting a subtask from the current queue of planned subtasks and outputting a subcommand STX(s,t) to the appropriate

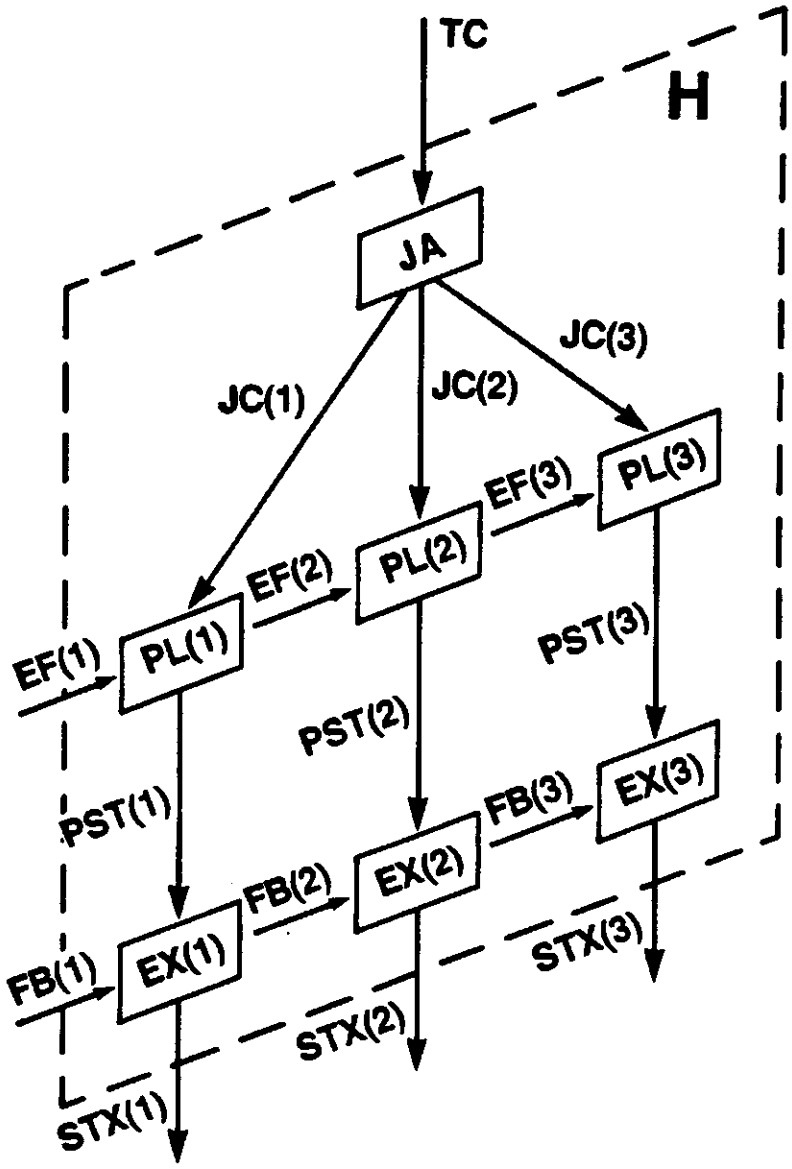


FIGURE 10: **The H module at each level has three parts: A job assignment module JA, Planners PL, and a set of executors EX.**

FIGURE 11

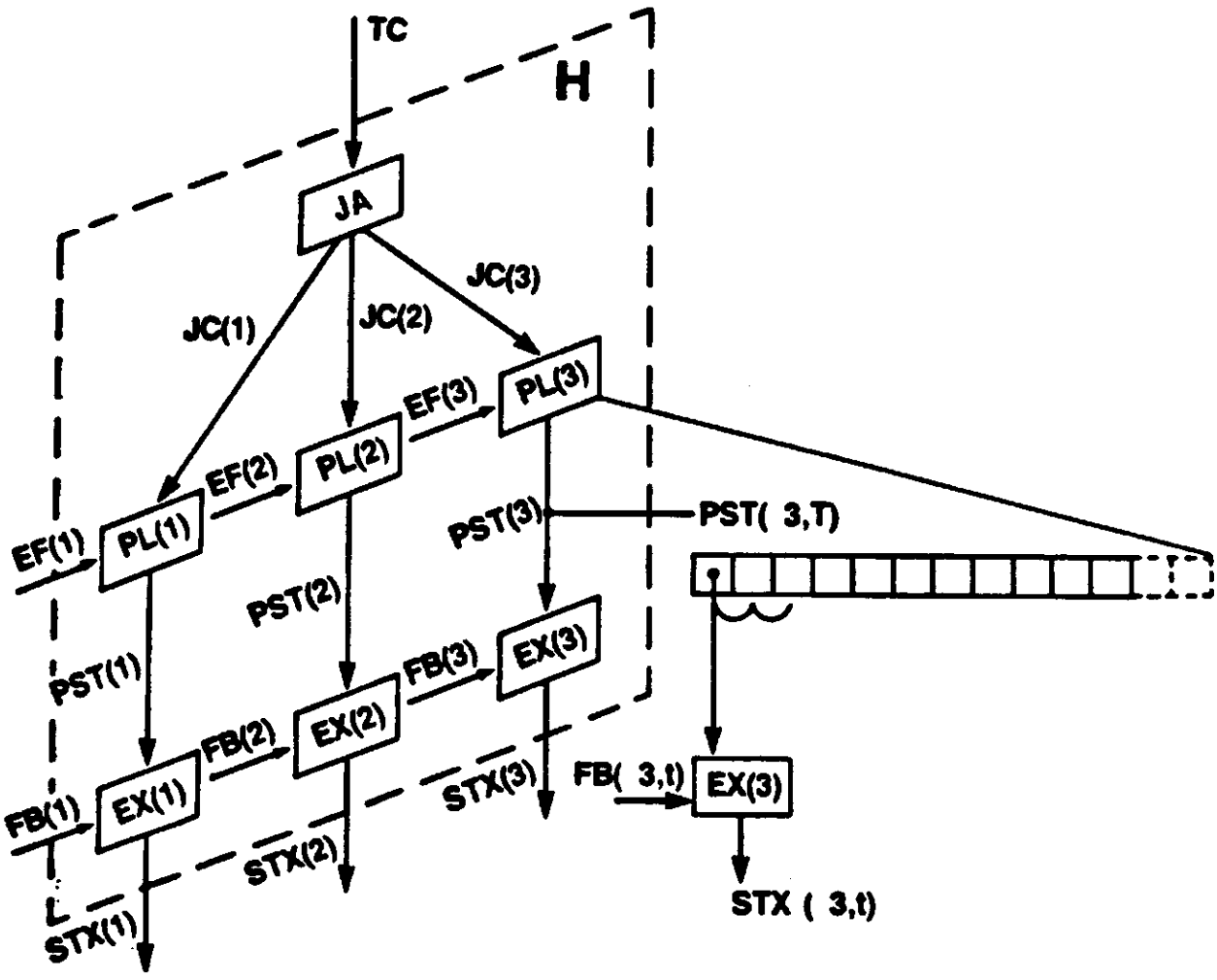


FIGURE 11: Each Planner PL (j) produces a string of planned subtasks PST (j,t). Ht Time t the Executor EX (j) reads the planned task PST (j,t). The feedback FB (j,t) and computers and output STX (j,t).

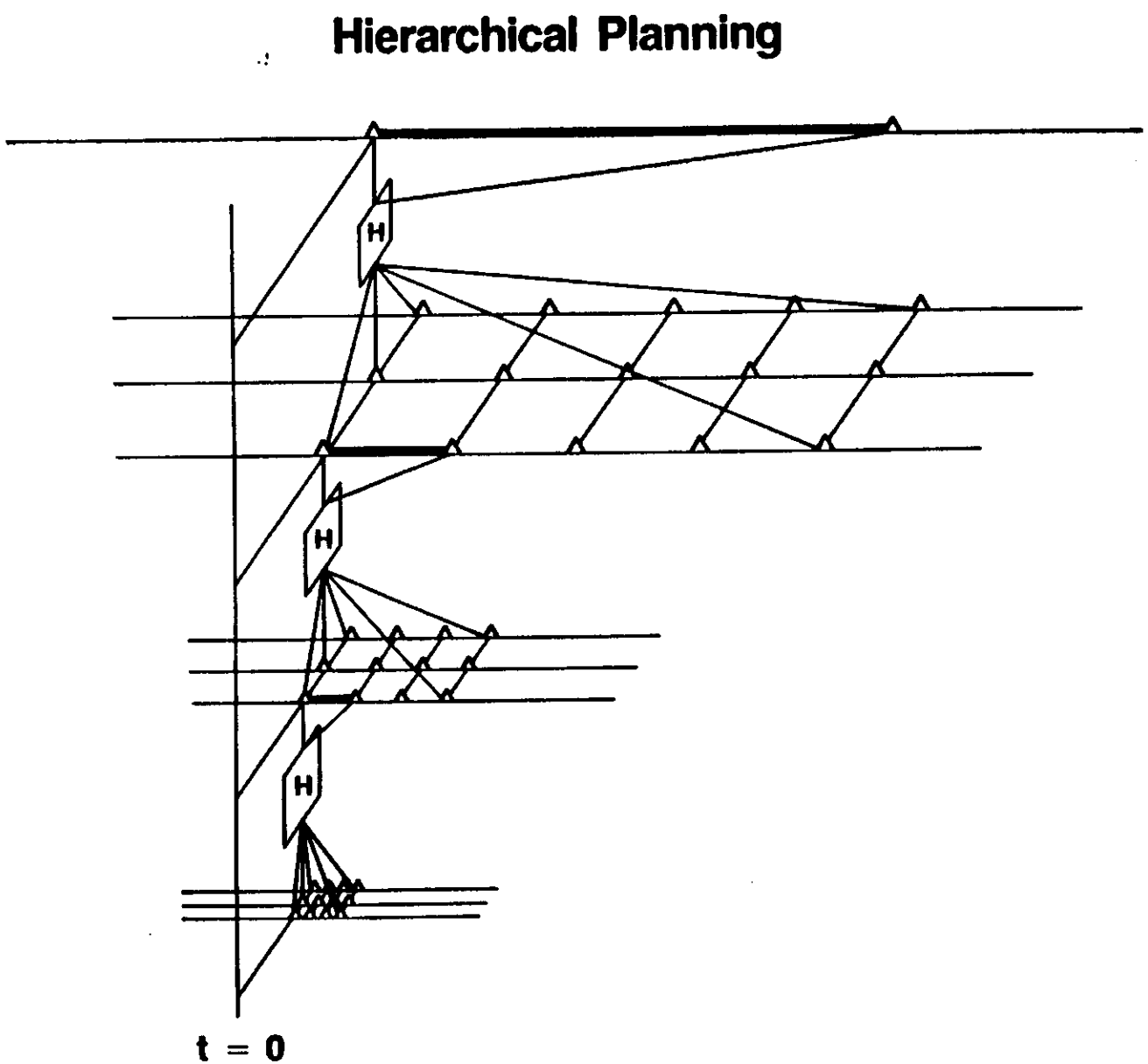


FIGURE 12: **Three levels if real-time planning illustrating the shrinking planning horizon and greater detail at successively lower levels of the hierarchy.**

subordinate H module at time t. The EX(s) module monitors its feedback FB(s,t) input in order to servo its output STX(s,t) to the desired subtask activity. The executor output may then be expressed as the function EX(s) operating on the parameters PST(s,t) and FB(s,t), i.e.

$$STX(s,t+n) = EX(s)\,[PST(s,t),FB(s,t)]$$

where n = the number of state clock periods required to compute the function EX(s). n typically equals 1.

The feedback FB(s,t) also carries timing and subgoal event information for coordination of output between executors at the same level. When the executor detects a subgoal event, it selects the next planned subtask from the queue.

Executor output STX(s,t) also contains requests for information from the world model M module, and status reports to the next higher (i+1) level in the H module hierarchy. The feedback FB(s,t) contains status reports from the H module at the i-1 th level indicating progress on its current task. As a minimum, these reports provide a handshaking acknowledgment of receipt of the subtask command and an echo of the unique identification number of the command currently being executed. This enables the EX(s) process to know that the subtask output given has been received and is being executed. The EX(s) process generates error reports if time-outs or failures in handshaking with the H module at the i-1 th level occur.

The data buffers forming the input and output buffers to the H module at the i-th level are shown in Figure 13.

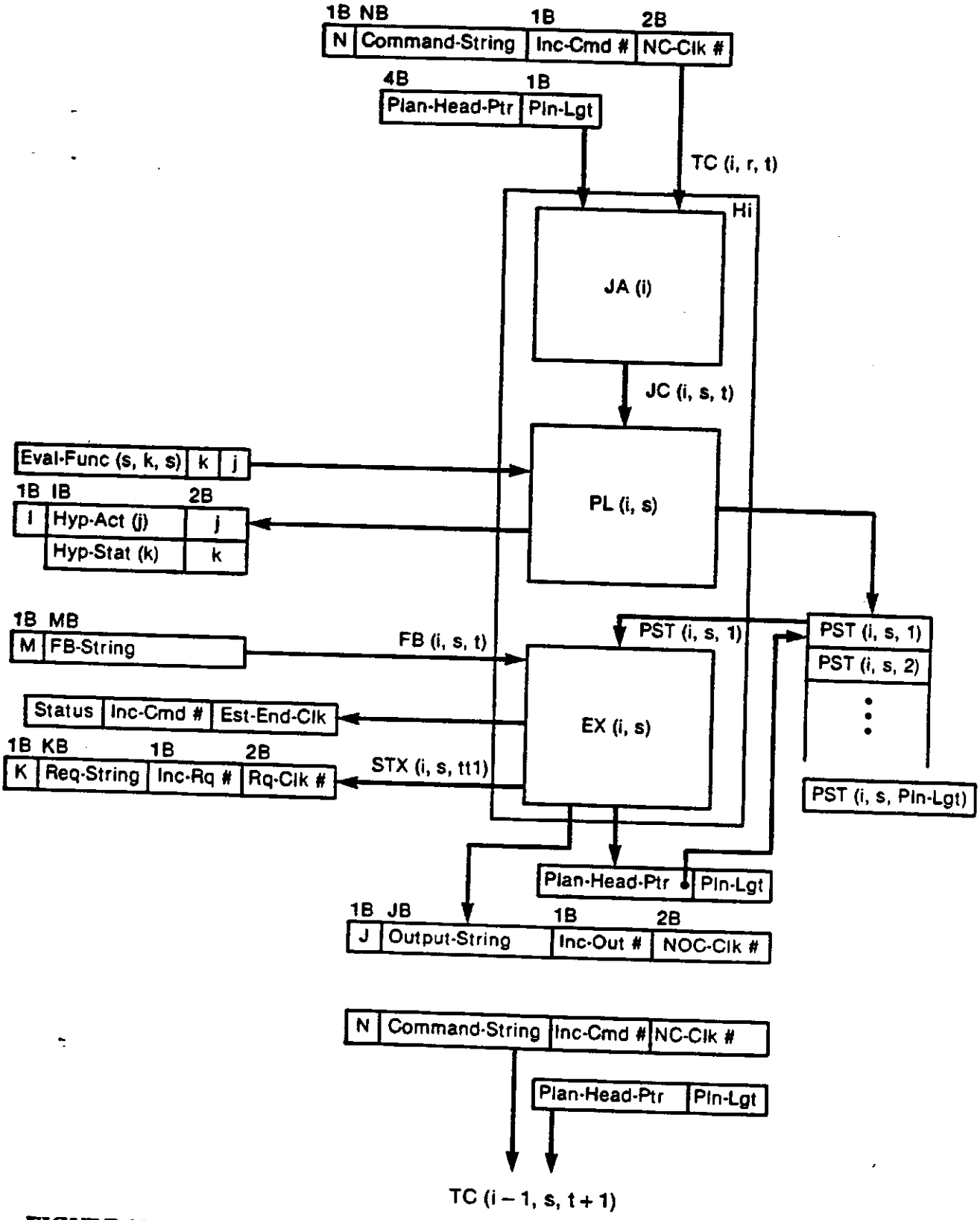

1B NB
N
Command-String
1B
Inc-Cmd #
2B
NC-Clk #
4B
Plan-Head-Ptr
1B
Pln-Lgt
TC (i, r, t)
Hi
JA (i)
JC (i, s, t)
Eval-Func (s, k, s)
k
j
1B IB
2B
I
Hyp-Act (j)
j
Hyp-Stat (k)
k
PL (i, s)
1B MB
M
FB-String
FB (i, s, t)
PST (i, s, 1)
PST (i, s, 1)
PST (i, s, 2)
Status
Inc-Cmd #
Est-End-Clk
EX (i, s)
1B KB
1B
2B
K
Req-String
Inc-Rq #
Rq-Clk #
STX (i, s, tt1)
PST (i, s, Pln-Lgt)
Plan-Head-Ptr
Pln-Lgt
1B JB
1B
2B
J
Output-String
Inc-Out #
NOC-Clk #
N
Command-String
Inc-Cmd #
NC-Clk #
Plan-Head-Ptr
Pln-Lgt
TC (i – 1, s, t + 1)

FIGURE 13:

6. TASKS AND PLANS

Def 3: Task

As shown in Figure 14, a task is an activity which begins with a start-event and is directed toward a goal. A goal is an event which terminates the task. A task command is an instruction to achieve a goal event of the form

DO <Task> AFTER <Start Event> UNTIL <Goal Event>

or

TASK COMMAND := DO <Task>
WHEN (Start Event)
DO (Task)
UNTIL (Goal Event)
END-DO

Def 4: Plan

A plan is a set of activity-event pairs which lead to the desired goal event. Each activity in the set leading to the goal is a subtask, and the event terminating each of the subtasks is a subgoal. The final event in the plan is the goal event. This is illustrated in Figure 15.

A plan may involve the scheduling of several machines to simultaneously perform different activities on different objects as illustrated in Figure 16. These subtasks may depend on each other for results, for example, if an operation on an object in one machine cannot begin before another machine finishes its operation on that same object.

Complex plans may involve conditional branching, or even probabilistic decision rules. Plans may also include provisions for branching to error correction activities and reporting failure in the case of lack of progress toward the goal.

In some cases, plans can be represented by mathematical functions of time and/or state variables such as distance from target, velocity, coordinate position, etc. For example, a path planner may compute a straight line trajectory from the current point to a goal point, or as illustrated in Figure 17, the planning function may compute acceleration and deceleration profiles as a function of time or position along the planned trajectory.

A plan can be represented in a number of different notations. The series of actions and events illustrated in Figure 16 is the form of a Gantt chart. Plans can also be represented as a graph of states and state-transitions in the form of a Pert or Critical Path Method (CPM) chart, as a Petri network, or any of several other methods for representing trajectories through state space, such as state-graphs, finite-state-automata (fsa) grammars.

In fact, any program or procedure designed to accomplish a goal is a form of plan. Plans typically are prepared before action begins, and are used to sequence activities in pursuit of the goal.

Def 5: Planning

Planning is the preparation of a plan. Planning can be done off-line (well before the action begins), or in real-time (immediately before the action begins or as the action is proceeding).

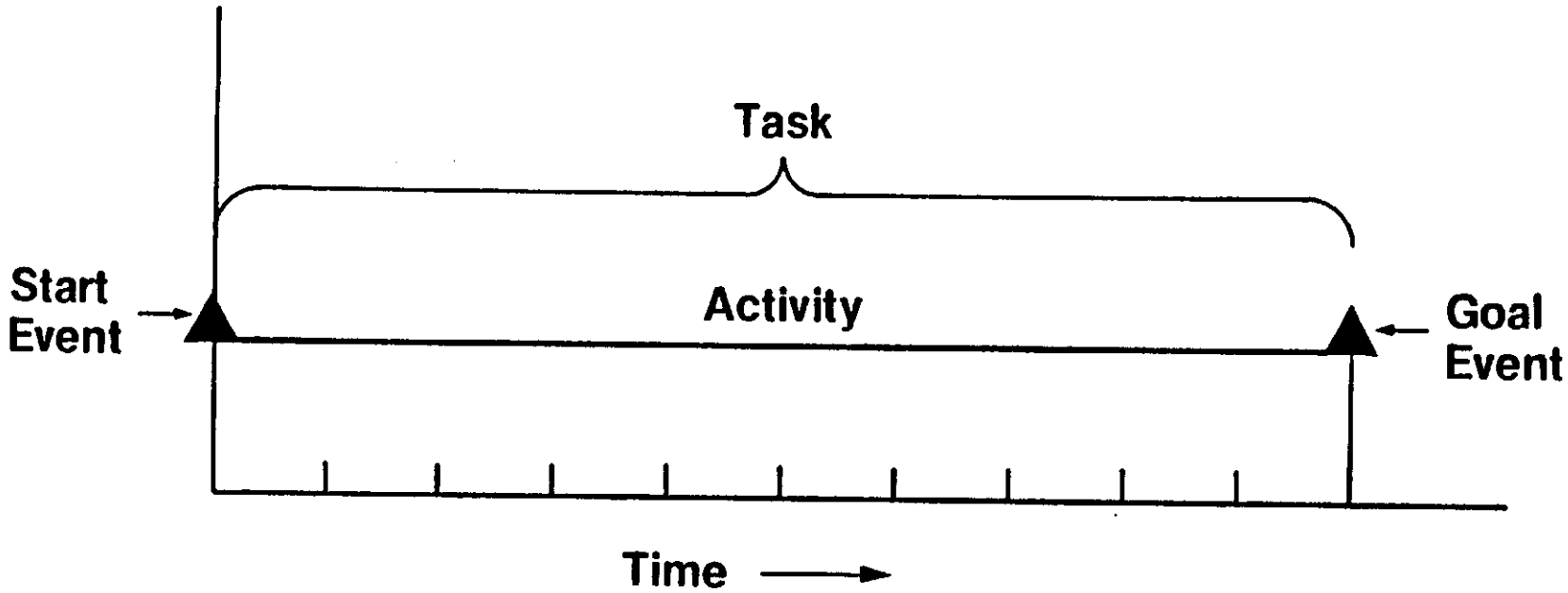


FIGURE 14: **A task is an activity directed toward a goal.**

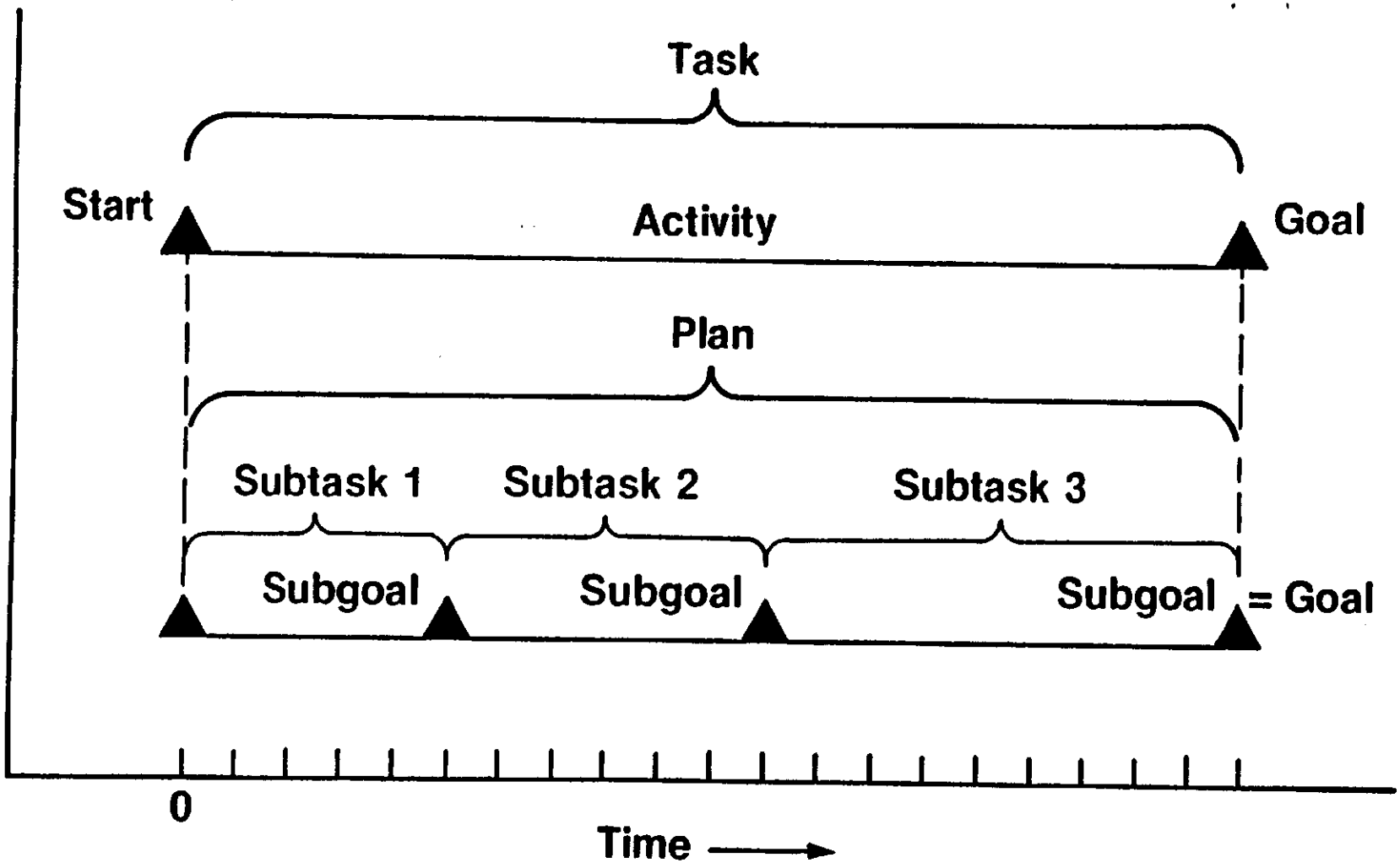


FIGURE 15: **A plan is a set of activity-event pairs or subtasks which achieve the goal event.**

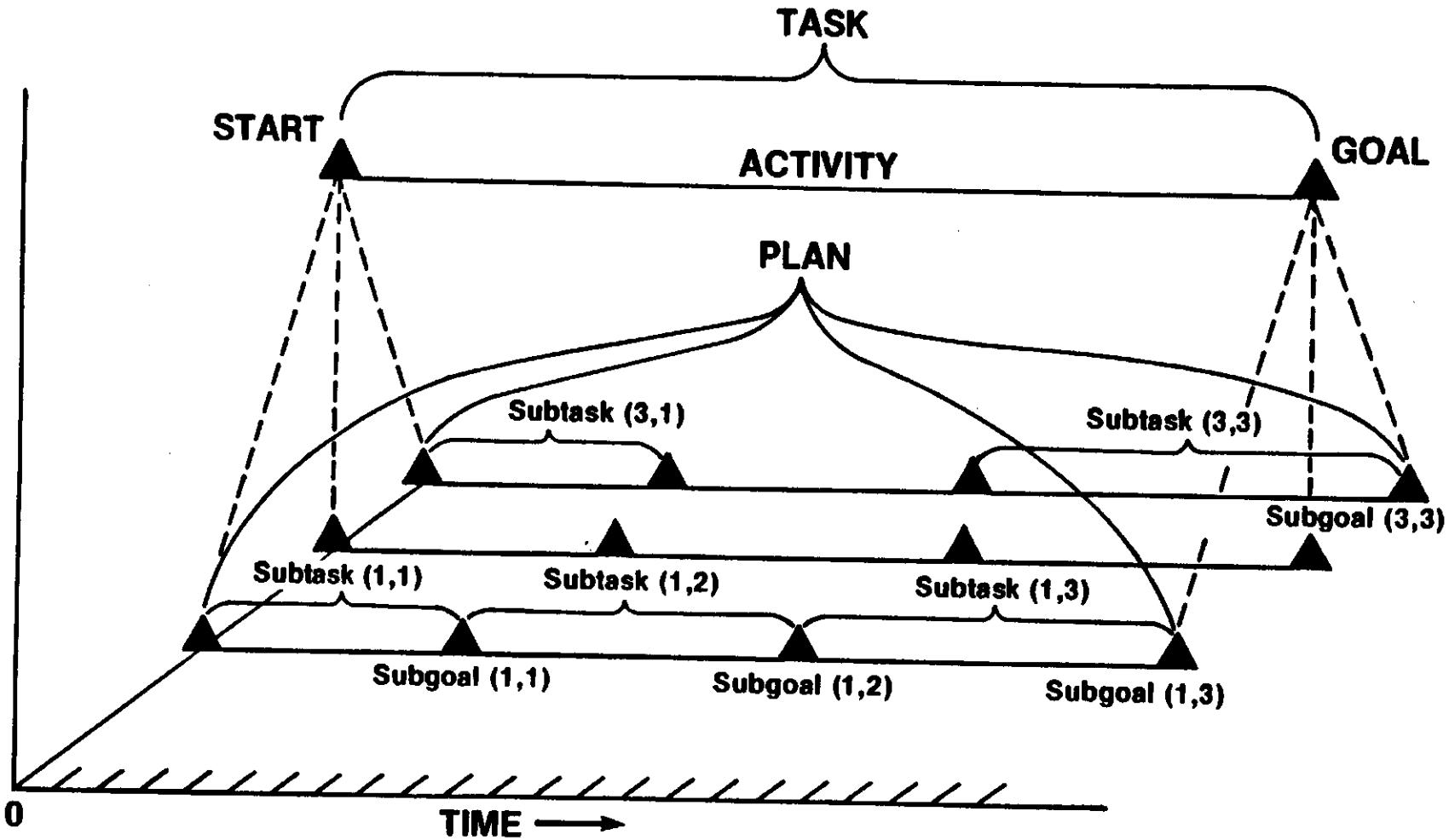
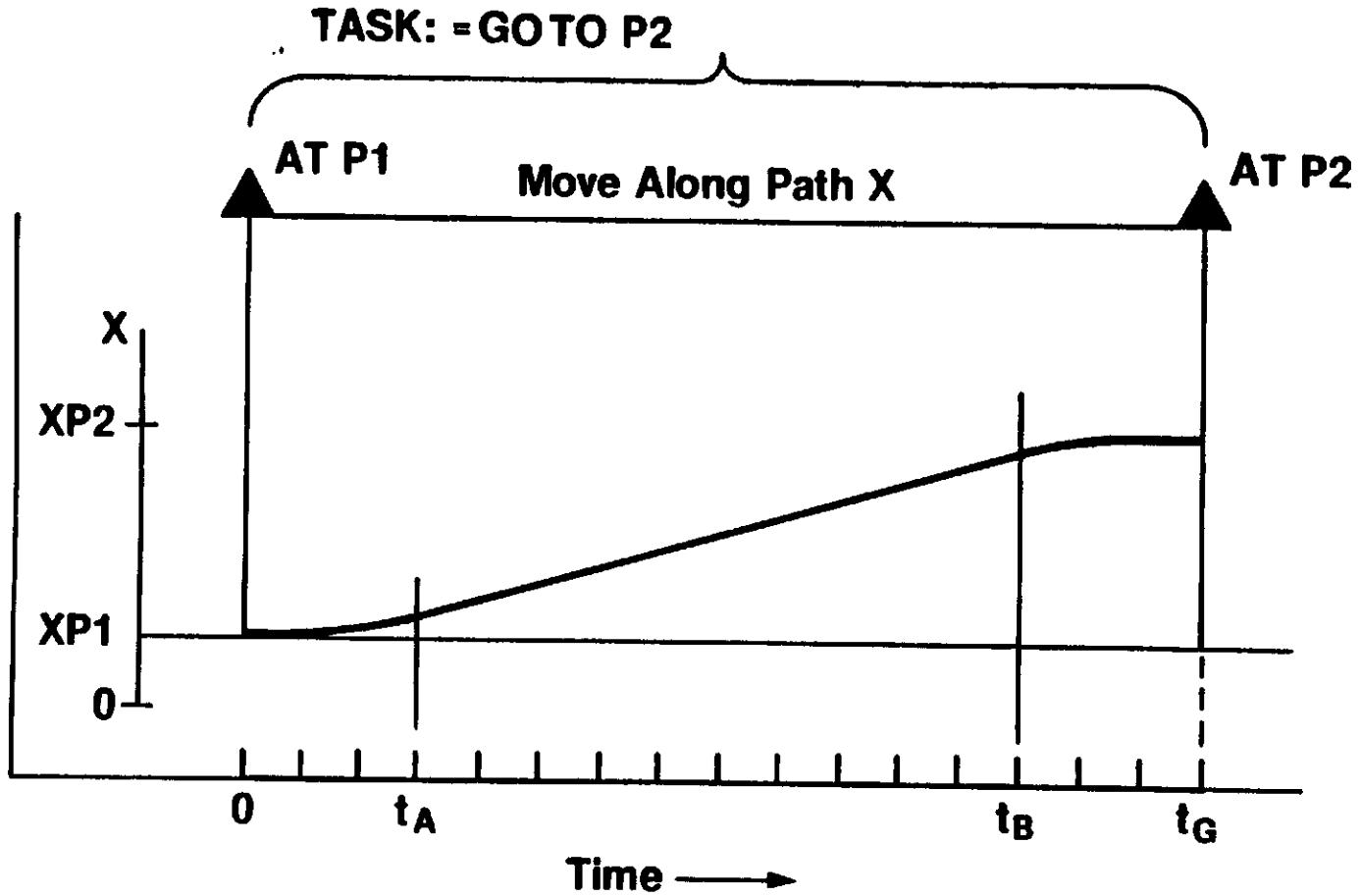


FIGURE 16: **A plan may consist of several concurrent strings of subtasks which collectively achieve the goal event.**



$$X = \begin{cases} \text{for } 0 < t \leq t_A & X = XP1 + K_1 t^2 \\ \text{for } t_A < t \leq t_B & X = K_2 t + K_3 \\ \text{for } t_B < t \leq t_G & X = XP2 - K_1 (t - t_G)^2 \end{cases}$$

FIGURE 17: **An example of a path plan for moving from point P1 to P2. Only the X component of the plan is shown.**

Of course, planning may combine off-line and real-time elements. For example, off-line planning may be used to develop a library of prefabricated plans, and real- time planning can then select a particular plan, or modify a prefabricated plan in order to fit the conditions that exist at, or near, execution time.

6.1 Gantt Notation

The Gantt chart notation explicitly represents the time axis, and can conveniently represent parallel simultaneous activities along the time axis. This is convenient for graphically visualizing what is happening in a control system. For example, Figure 2 illustrates how the planners in each H module generate a temporal decomposition. Figure 18 shows how the set of subgoals which terminate the plan combine to fulfill the goal event of the input task command.

Figure 12 shows three levels of planning activity. The activity represented by the Gantt chart at the highest level is input to the top level H module as a task command. This task is decomposed by the job assignment manager and three planners of the top H module into three simultaneous plans consisting of four activity-event pairs each. The first executor of the top level H module outputs the current subtask command in its plan to a second level H module. This second level task command is decomposed by the job assignment manager and three planners in the second level H module into three plans, again consisting of four subtasks each. The first of the second level executors outputs the current activity in its plan to a third level H module, which further decomposes it into three plans of four subtasks. At each level the final subgoal events in the plans correspond to the goal of the input task. At each successively lower level, the planning horizon becomes shorter, and the subtasks become more detailed and fine structured.

Planning is done top-down. The highest level plan covers the entire backlog of work to be done. At each lower level, plans are formulated (or selected) in real-time to accomplish the next step in the plan of the level immediately above. Thus, a goal directed control system such as is described here always has a hierarchy of plans in place. If the work goes as planned, each level of the control the system will always be able to anticipate the next subtask, and there is no need to pause to replan. However, if unexpected events cause a plan to become obsolete, the system may suddenly find itself without a plan. This condition can be described as a state of "confusion", in which one or more levels has no plan available for execution.

If the activity-event pairs at each level are displayed as illustrated in Figure 19, the resulting Gantt chart has the form of a musical score. This form suggests a possible notation for programming multiple cooperative tasks. It may even be possible to develop a programming system using a computerized form of musical scoring, or ballet notation such as Labans.

Each activity on such a chart can be described by a frame. If the proper software tools are available, it is possible to bring up a window containing a frame describing the activity simply by pointing to the activity with a cursor. This is illustrated in Figure 20. The slots in the frame can be edited by a human process planner. The process planning system developed by Brown and McLean [51,52] for the NBS AMRF contains most of the tools required for a space station telerobot task planner.

6.2 State-Graph Notation

The state-graph notation has the advantage that it can be directly translated into a finite state automata (fsa).

$$\text{fsa} = \{\text{states, transition table, inputs, outputs}\}.$$

The nodes of the state graph are states of the fsa, inputs are planned subtask commands plus feedback

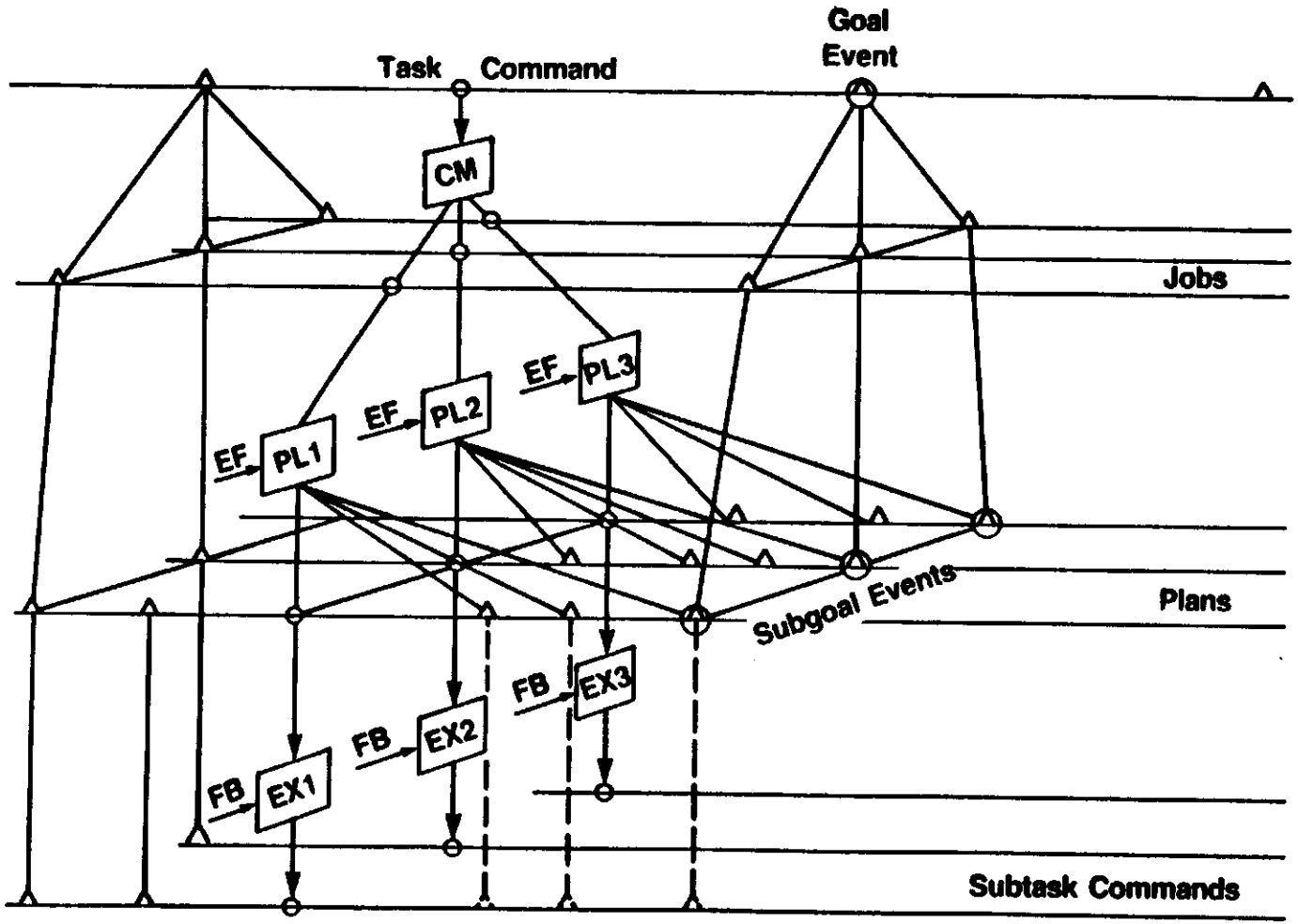


FIGURE 18: **The job assignment manager decomposes a task command into a set of jobs. The planners PL (j) decomposes jobs into plans, and the executors EX (j) translates plans into subtask commands.**

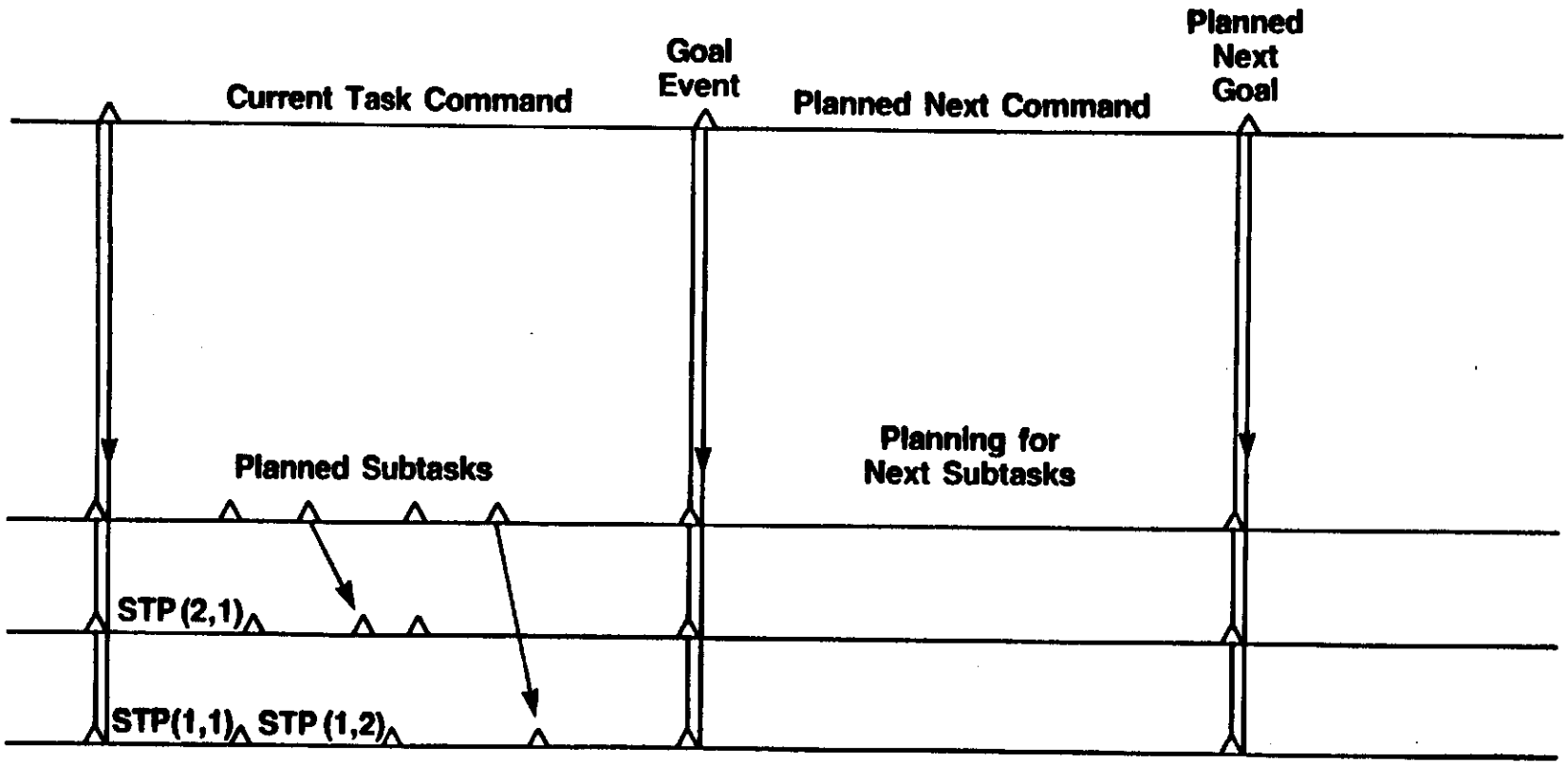


FIGURE 19: **The gantt chart notation for planning is a potential programming interface.**

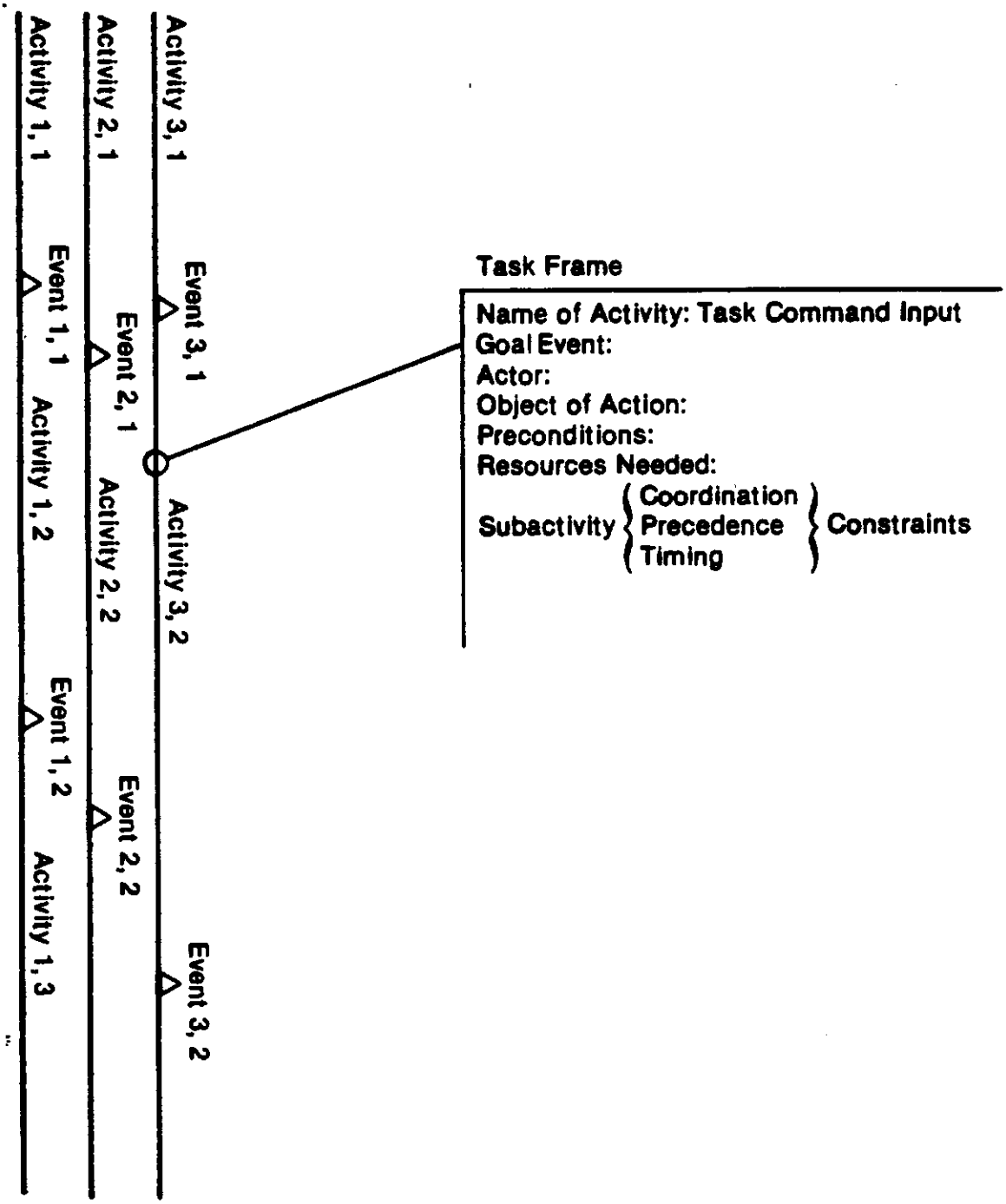

Activity 3, 1
Activity 2, 1
Activity 1, 1
Event 3, 1
Event 2, 1
Event 1, 1
Activity 1, 2
Activity 2, 2
Activity 3, 2
Event 1, 2
Event 2, 2
Event 3, 2
Activity 1, 3
Task Frame
Name of Activity: Task Command Input
Goal Event:
Actor:
Object of Action:
Preconditions:
Resources Needed:
Subactivity
Coordination
Precedence
Timing
Constraints

FIGURE 20:

PST(s,t) + FB(s,t), outputs are the executor outputs STX(s,t). Edges are the lines in the transition table which define the IF/THEN rules for subtask selection [53].

There is an important distinction to be made between states of the control system and states of the external world. Figure 21 illustrates this distinction. States of the world are transition conditions for the control system, and states of the control system produce actions that cause transitions to occur in the state of the world. Therefore, the state graph of the world is a dual of the state graph of the control system. The state graph of the world can be viewed as a Gantt chart, where states are nodes and activities are edges. The state graph of the control system can be viewed as a Pert chart, where nodes correspond to states, and edges correspond to events in the world that cause the control system to transition between states.

If plans are expressed in state-graph form, EX(s) is the fsa defined by the state-graph. The state of EX(s) corresponds to the currently active node in the state graph. The output of EX(s) at time t is STX(s,t). EX(s) monitors its input PST(s,t) + FB(s,t), and discovers which line (or lines) in the fsa state transition table match the current situation. EX(s) then executes the appropriate line in the state table; i.e. it goes to the next state called for by that line, computes the functions called, and outputs the STX(s,t) subtask output commands selected.

In the ideal case where the task decomposition works according to the plan, a planner PL(s) merely needs to add one new activity- event pair to the end of the current plan on average as often as the Executor EX(s) achieves a sub-goal event and steps to the next activity in the current plan.

However, in cases where the task execution does not go as planned, the current plan may need extensive modification, or a completely new plan may need to be generated (or selected). The time required to generate a new plan is an important system requirements parameter, and what the system does while a new plan is being computed is an important issue in error recovery and restart.

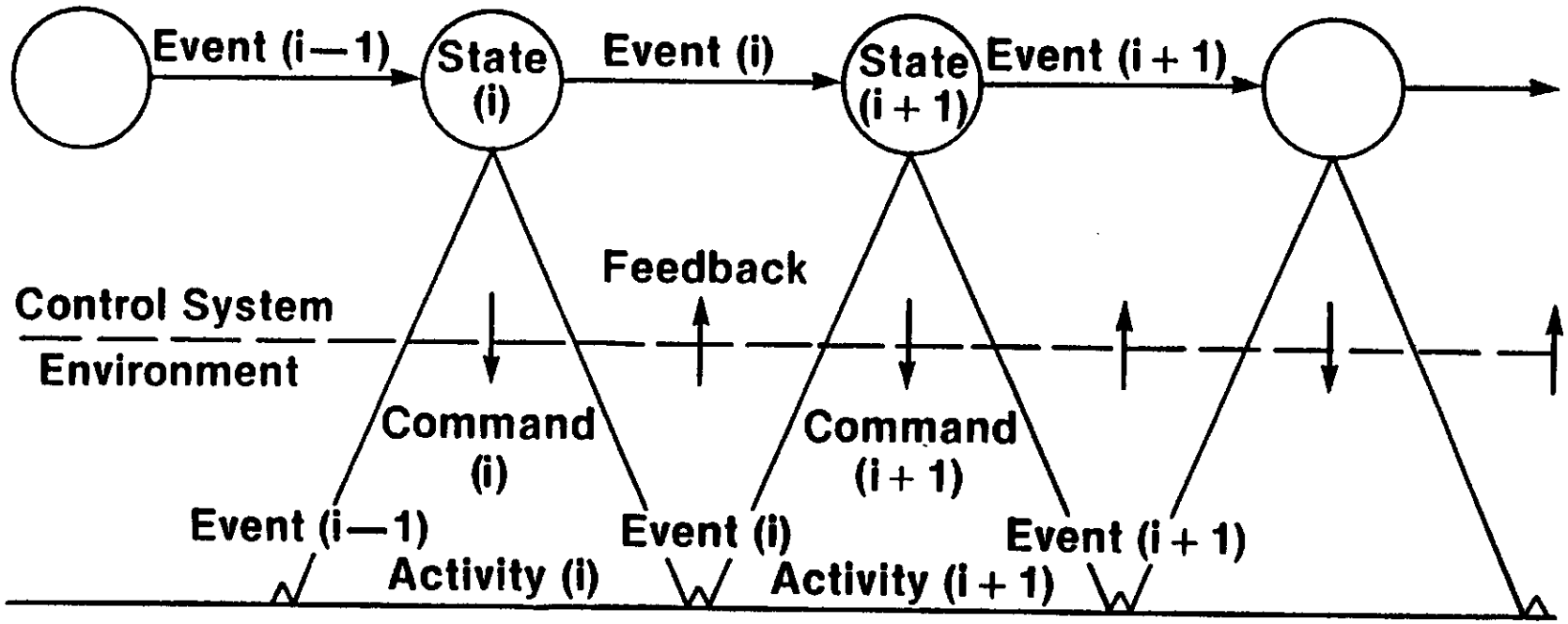

Event (i—1)
State (i)
Event (i)
State (i + 1)
Event (i + 1)
Feedback
Control System
Environment
Command (i)
Command (i + 1)
Event (i—1)
Activity (i)
Event (i)
Activity (i + 1)
Event (i + 1)

FIGURE 21:

7. AN EXAMPLE IMPLEMENTATION

An example of how the NASREM control hierarchy might be implemented is illustrated in Figure 22. The VME bus supports high bandwidth communication between sensory processing, world modeling, task planning, and task execution modules at each level of the hierarchy. These modules can exist on separate single board computers for high speed parallel computation. The commands and status feedback between various levels of the hierarchy requires much lower bandwidth, and could be passed through gateways between separate buses. A high speed bus for vision sensors may be required at the lower levels of the image processing hierarchy.

This type of implementation can accommodate tens, or even hundreds of single board computers. It therefore can support extremely complex control computations, such as those required for multiple manipulators, multi-fingered hands, etc. It can also support special purpose computing elements, such as pipeline image processors and vector accelerators, as long as they have a VME bus interface.

A software development and simulation environment similar to that shown in Figure 22 is extremely important. A variety of software development tools, such as Lisp machines, workstations with bit-mapped screens, graphics engines, and supercomputers for dynamic modeling and simulation should be provided. Translators and cross compilers should be provided so that software developed in this environment can be downloaded into the target hardware for real-time execution. Activity at the Service Mission level is sufficiently non-time-critical that the development/simulation environment could be used for run-time execution at this level.

7.1 Timing

The rate of subtask completion, and hence the rate of subgoal events, increases at the lower levels of the hierarchy, and decreases at upper levels of the hierarchy. If the planners at each level generate plans containing an average of ten steps, the average period between changes in output at each level will increase an order of magnitude at each higher level in the control hierarchy.

At the lowest level, the servo subtask output STX(1,s,t) is perfectly regular, one output per millisecond. At the Primitive level the executor subtask output STX(2,s,t) will be computed sixteen times slower. Hence the lowest level planner has to compute a new plan (interpolation profile) every 16 milliseconds.

At the E-Move level and higher the subtask durations are variable, because subtask goals correspond to sensed events in the external world. E-Move outputs are trajectory knot points which are not necessarily evenly placed in space or time. The Primitive level is responsible for inertial dynamics, and hence needs to produce outputs synchronized with time. The primitive level thus adds more or less interpolation points to compensate for the non-regular nature of the E-Move level output. The Primitive level, therefore, functions as a time synchronizer.

Above the E-Move level, the non-regularity of subtask output duration becomes more and more pronounced, and there are long periods during which much of the control system is in a WAIT mode. For example, there may be periods of hours or even days during which the telerobot is waiting for a new job. During these periods the telerobot may be stowed, and its control system placed in a WAIT mode.

There, of course, will also be periods of activity during which all levels of the control hierarchy are fully engaged. During those periods a six level system of the design outlined here will have approximately the following rates of change in output, average replanning intervals, and planning horizons. A replanning interval, as defined here, is the time required to add one additional step to an existing plan, or to slightly modify an existing plan to reflect a new piece of information from the world

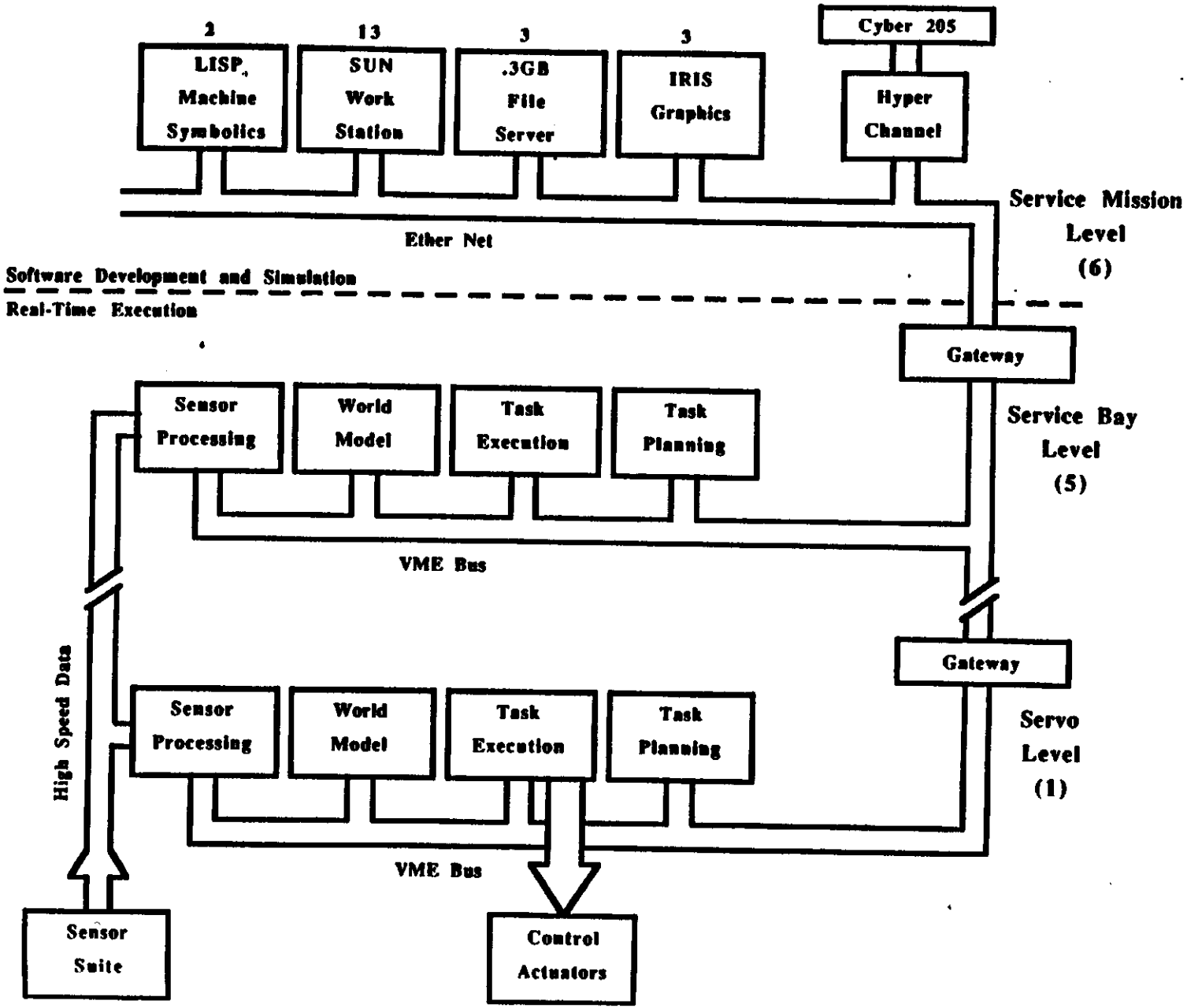

2
LISP Machine Symbolics
13
SUN Work Station
3
.3GB File Server
3
IRIS Graphics
Cyber 205
Hyper Channel
Ether Net
Service Mission Level (6)
Software Development and Simulation
Real-Time Execution
Gateway
Sensor Processing
World Model
Task Execution
Task Planning
Service Bay Level (5)
VME Bus
Gateway
High Speed Data
Sensor Processing
World Model
Task Execution
Task Planning
Servo Level (1)
VME Bus
Sensor Suite
Control Actuators

FIGURE 22:

model. For situations where new information makes the current plan obsolete, thereby requiring a completely new plan, the replanning interval may considerably exceed the average replanning interval.

	Average rate of change in output		Average replanning interval		Planning horizon	
Servo	1000	KHz	2	millisecond	15	msec
Primitive	100	Hz	30	"	300	msec
E-Move	10	Hz	200	"	2	sec
Object/task	1	Hz	3	second	30	sec
Service Bay	0.1	Hz	1	"	> 10	min
Mission	0.01	Hz	6	minutes	> 1	hour

A subtask at any level can be altered on any state clock cycle, so that the minimum subtask period at all levels is state clock period.

Figure 22a provides an example of a timing diagram for all six levels of the NASREM hierarchy for task decomposition and sensory processing. The times shown have been chosen to illustrate the relative timing between levels. Specific design timing requirements are manipulator dependent.

The highest level input command is to accomplish the mission. The mission plan covers the entire backlog of work to be done, and the planning horizon of the highest (mission) level is the end of the entire mission. At each lower level, plans are formulated (or selected) in real-time to accomplish the current and next task in the plan of the level immediately above. Each plan in the higher level plan is decomposed into a lower level plan of at least two, and typically less than ten, subtasks. The planning horizon thus shrinks exponentially at each successively lower level of the hierarchy.

Similarly, the rate of subtask completion, and hence the rate of subgoal events, increases at the lower levels of the hierarchy, and decreases at upper levels of the hierarchy. If the planners at each level generate plans containing an average of ten steps, the average period between changes in output at each level will increase by an order of magnitude at each higher level in the control hierarchy.

Replanning is done either at cyclic intervals, or whenever emergency conditions arise. The cyclic replanning interval is about equal to the command update interval at each level, growing to about one percent of the planning horizon at the mission level. Thus the real-time planner must generate a new plan about as often as the executor puts out a new output command. Emergency replanning begins immediately upon the detection of an emergency condition.

The timing diagram in Figure 22a illustrates the duality between the task decomposition and the sensory processing hierarchies. A sensory event at one hierarchical level can be defined as a sequence of events at the next lower level. At each level in the past hierarchy, the sensory processing modules look back into the past about as far the planner modules look forward into the future. At each level, future plans have about the same detail as historical traces.

The goal events which terminate each subtask in the plan, when achieved at time t=0, become the observed events that make up the historical trace. To the extent that a historical trace is but a time shifted duplicate of a former future plan, the plan was followed and every task was accomplished as planned. To the extent that a historical trace deviates from the plan, there were surprises.

Figure 22b illustrates the command and feedback variables present at the first three levels of the NASREM hierarchy, and their relative update rate along the time line, for one specific algorithm.

NASREM TIMING DIAGRAM

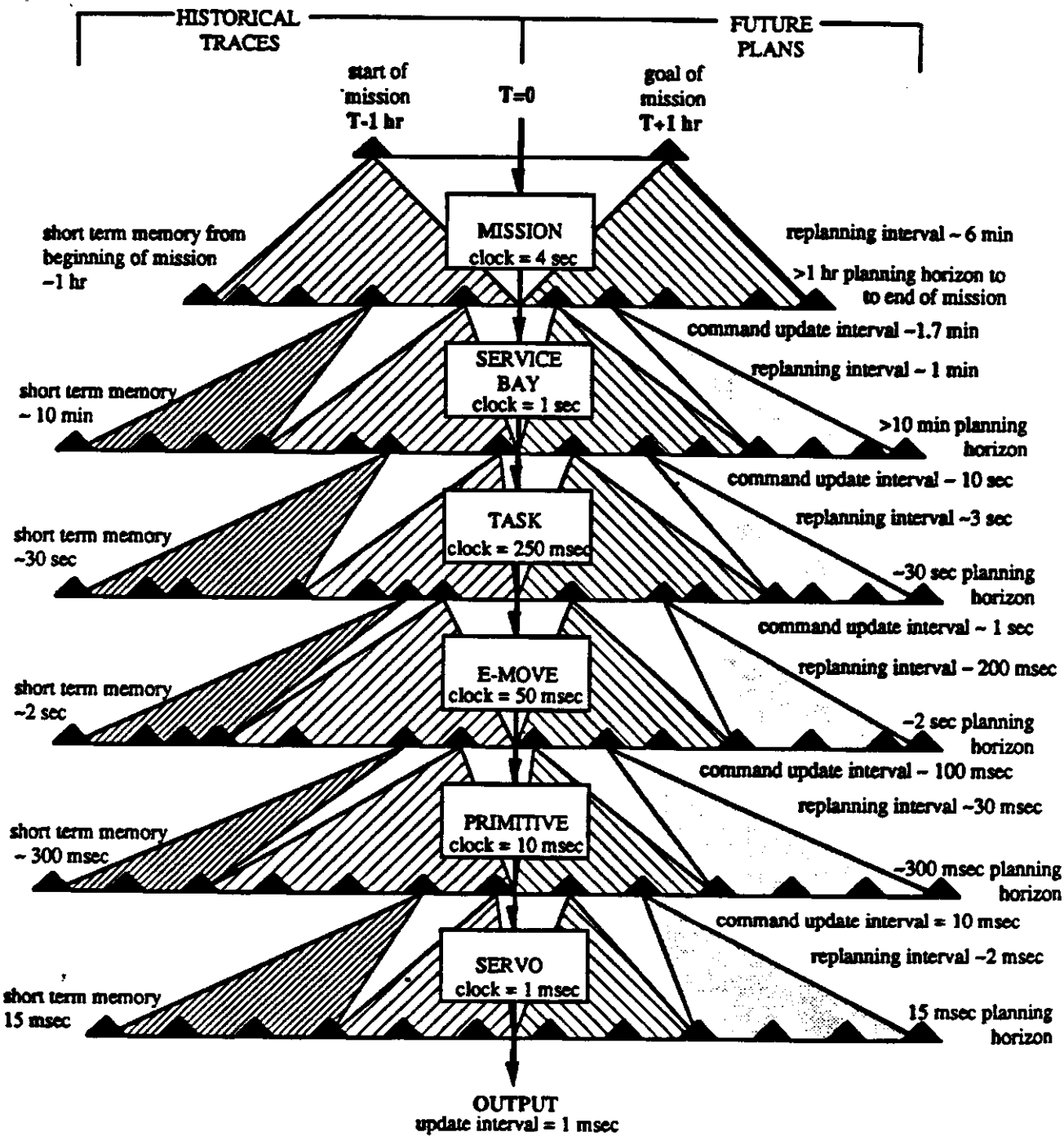


FIGURE 22a:

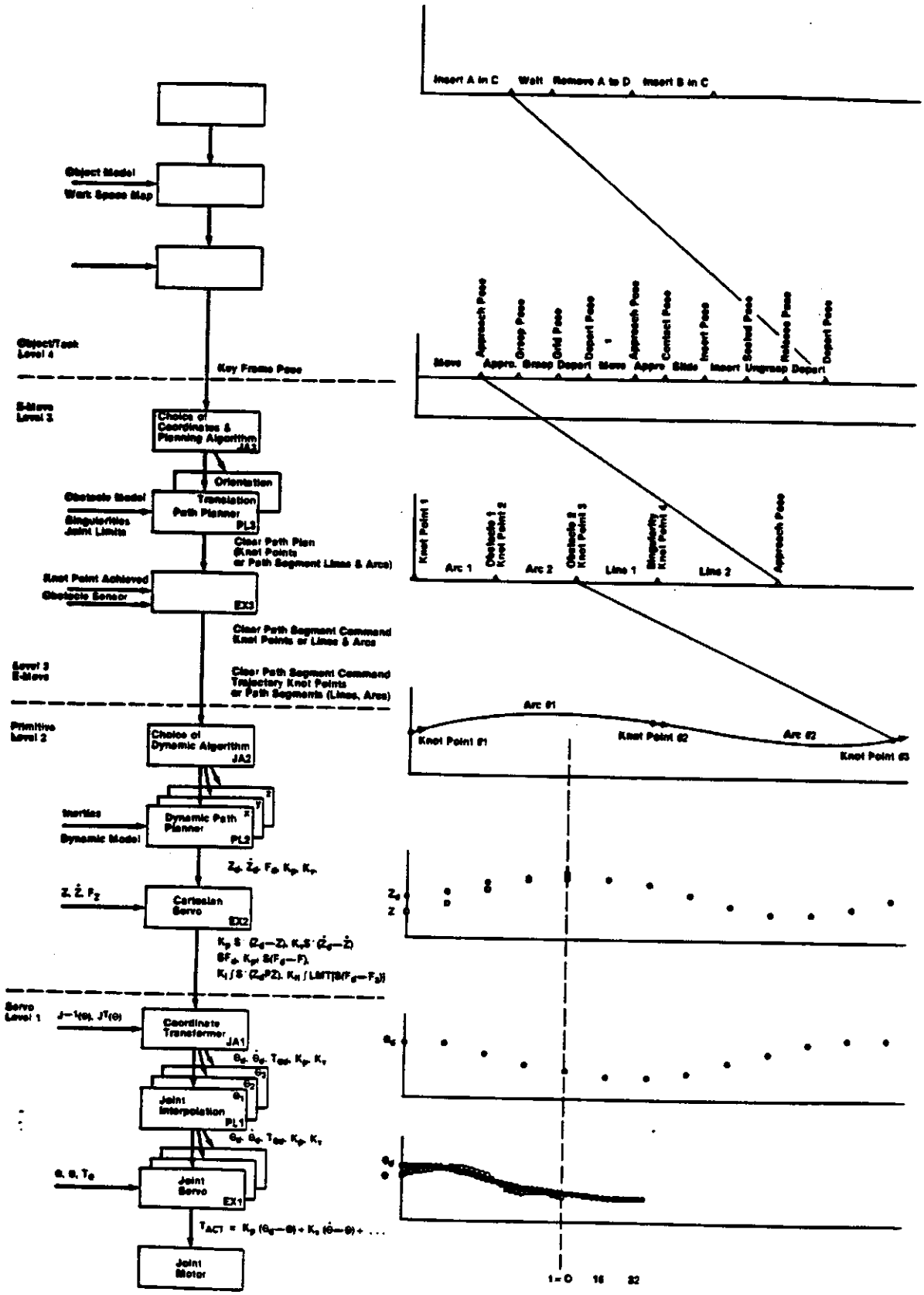

Insert A in C
Wait
Remove A to D
Insert B in C
Object Model
Work Space Map
Object/Task Level 4
Key Frame Pose
S-Move Level 3
Choice of Coordinates & Planning Algorithm JA3
Orientation
Translation
Path Planner PL3
Obstacle Model
Singularities Joint Limits
Clear Path Plan (Knot Points or Path Segment Lines & Arcs)
Knot Point Achieved
Obstacle Sensor
EX3
Clear Path Segment Command (Knot Points or Lines & Arcs
Level 3 S-Move
Clear Path Segment Command Trajectory Knot Points or Path Segments (Lines, Arcs)
Primitive Level 2
Choice of Dynamic Algorithm JA2
Inertias
Dynamic Model
Dynamic Path Planner PL2
Cartesian Servo EX2
Servo Level 1
Coordinate Transformer JA1
Joint Interpolation PL1
Joint Servo EX1
Joint Motor
Approach Pose
Grasp Pose
Grid Pose
Depart Pose
Contact Pose
Insert Pose
Sealed Pose
Release Pose
Move
Appro.
Grasp
Depart
Appro
Slide
Insert
Ungrasp
Depart
Knot Point 1
Arc 1
Obstacle 1 Knot Point 2
Arc 2
Obstacle 2 Knot Point 3
Line 1
Singularity Knot Point 4
Line 2
Approach Pose
Arc #1
Knot Point #1
Knot Point #2
Arc #2
Knot Point #3
1
0
16
32

FIGURE 22b:

NASREM is designed as a functional architecture which can support many algorithms. Figure 22b shows an instantiation of NASREM for a certain set of algorithms. It should not be assumed that other algorithms cannot be supported as well.

8. FUNCTIONAL DESCRIPTION OF CONTROL LEVELS

At this point in the discussion, we will add the level index to the state variables TC, JC, PST, STX where TC is the task command, JC is the job command, PST is the planned subtask, and STX is the executor output. For example, STX(1,2,3) means output from level 1 of executor #2, at time t=3. The level index will also be added to the functions JA, PL, and EX. For example, EX(2,4) is the fourth executor at level 2.

8.1 Level 1 -- Servo/ Coordinate Transfer Level

Level 1 transforms coordinates and servos output.

Inputs consist of commands designed to null the error between desired and observed positions, orientations, velocities,and forces of manipulators, grippers, transporters, and sensor platforms in the coordinate system of choice.

The function of the Servo Level is to handle motion 'small in a dynamic sense'. This means that the level executes a specific algorithm for approaching a command attractor set, which consists of positions, velocities, accelerations, acceleration rates, forces, force rates, coordinate system, etc. This attractor set explicitly expresses the desired state of the manipulator.

Outputs consist of electrical voltages or currents to motors and actuators.

8.1.1 Input Commands

Input commands to level 1 are designated TC(1,r) r = 1,2,..., M, where M is the number of subsystems being controlled. A subsystem is defined as a group of actuators which combine their actions to move a single end effector, such as a gripper, a tool, a camera, a laser beam, etc. A camera subsystem might consist of pan, tilt, zoom, focus, and iris actuators. A tool subsystem might consist of the set of actuators that move the arm holding the tool. A multi-fingered gripper subsystem might consist of the set of actuators that move the fingers to as to manipulate the position and orientation of an object held in the fingers. A multi-fingered gripper subsystem might be carried on the end of an arm subsystem.

For purposes of this discussion, assume a FTS system with six subsystems, where:

subsystem 1 is the set of actuators or thrusters on the FTS transport system

subsystem 2 is the pointing, zoom, focus, and iris actuators on the left camera

subsystem 3 is the set of actuators (including gripper actuators) on the left manipulator arm

subsystem 4 is the set of actuators on the stabilizer foot

subsystem 5 is the set of actuators (including gripper actuators) on the right manipulator arm

subsystem 6 is the pointing, zoom, focus, and iris actuators on the right camera.

For transporter system arm or vehicle thrusters, level 1 input commands TC(1,1) define desired FTS platform positions and orientations, velocities, and forces in a coordinate system of choice.

For camera pointing, level 1 input commands TC(1,2) and TC(1,6) define desired pointing vectors, zoom resolution, and focus and iris settings for the left and right cameras.

For the manipulator arms and the stabilizer foot, level 1 input commands TC(1,3) and TC(1,5) define

desired positions, velocities, forces, and stiffnesses of the end effectors in a coordinate system of choice.

8.1.2 Task Decomposition - The H Module

The H module consists of Job Assignment, Planner, and Executor modules.

8.1.2.1 Job Assignment Module

The job assignment modules JA(1,r) at level 1 perform kinematic coordinate transformations, from a convenient coordinate system in which the control problem is most easily expressed, into joint coordinates. At least four different coordinate systems should be selectable:

1) a coordinate system fixed in the manipulator (or subsystem) base,
2) one fixed in the end effector of the manipulator,
3) one fixed at a convenient point in work space,
4) one fixed in an object of interest such as an electronic module to be services, or a part to be manipulated.

Any of these coordinate systems may be either moving or stationary. For example, if a coordinate system is chosen fixed in a module to be replaced on a spacecraft, that module may be rotating with the spacecraft of which it is a part.

The kinematic coordinate transformation often is not unique. For example, a commanded position of a manipulator end effector can often be achieved by more than one kinematic configuration of the manipulator arm. In these cases, the desired configuration must be specified in the level 1 input command, or a default configuration assumed. The choice of configuration is probably best made at the E-Move level as a part of the obstacle and singularity avoidance computations. In order for the E-Move level to specify the desired configuration, the information as to the current configuration must be available to it from the world model.

A new coordinate transformation is computed for every level 1 input command, i.e., once every 16 milliseconds. The level 1 Job Assignment module level must be able to work equally well with all coordinate systems of choice, and to switch readily back and forth between coordinate systems within the interval between level 1 input commands. The choice of coordinate system for each subsystem is probably best made at the Object/Task level where the tasks to be performed on objects are transformed into sequences of effector movements.

8.1.2.2 Planner Modules

The servo level planners PL(1,s) interpolate trajectory points (straight line, circular, or spline) in joint coordinates between level 1 command updates. Planned joint trajectory points PST(1,s,tt) provide smoothly varying commands to the executors EX(1,s), one command for each time the feedback FB(1,s,t) is sampled. The servo level planners must also deal with force command inputs or hybrid force/position control inputs.

Planner outputs may include coefficients for position, integral, and differential terms in the servo loops. These are derived from stiffness and damping factors specified in the level 1 input.

8.1.2.3 Executor Modules

The level 1 executors EX(1,s) are servos which compare the current observed joint positions, velocities, and forces with the commanded (or planned) positions, velocities, and forces. The errors between planned and observed values are used to compute outputs designed to null the difference between planned and observed values. Command and feedback input is sampled by the executors every millisecond.

Speed can be achieved by parallel computations. Each joint actuator is servoed to a trajectory of set points developed by its respective planner. All terms representing dynamic interactions between coupled manipulator joint linkages or vehicle rotations change slowly compared with servo output requirements. Thus, the coefficients of equations required to compute each joint output can be updated at rates comparable to level 1 inputs, or at least once every 16 milliseconds.

8.1.3 Output Subcommands

Output from the level 1 executor modules EX(1,s) consist of electrical voltages or currents as shown in Figure 23. These outputs directly drive power amplifiers for mechanical actuators such as manipulator joint motors, machine tool axes, camera pan, tilt, zoom, focus, and iris controls, clamps, pumps, motors, valves, and various other mechanical output devices. Level 1 outputs may also drive electrical and acoustic emissions such as radar, and laser ranging devices. There are thus N executors EX(1,s), s=1,2,3,...,N where N=the number of outputs to physical actuators.

The time required at level 1 for the EX(1,s) modules to compute an updated output is one millisecond. In other words, the servo level manipulator task decomposition module executor samples commands and feedback inputs each tick of a one millisecond control cycle clock. It then computes an output, writes that output to output registers, and waits for the next control cycle clock. During the wait interval, a communications process moves new data into all level 1 input registers.

8.1.4 World Modeling

At all levels, the world model consists of a modeling process M, and a block of global memory. State variables are maintained which represent measured, estimated, or a priori knowledge of both the external environment and the internal state of the control system. These are made available to H, M, and G processes.

Level 1 global memory contains observed positions, velocities, and forces of manipulator joints, and thrusters as measured by sensors. This information is scaled and filtered before being entered into the global memory. Global memory may also contain information such as inertial and frictional characteristics of the manipulator arms, g-forces, estimates of masses being manipulated, and estimates of cross products of inertia between degrees of freedom.

The level 1 world model also contains current kinematic and dynamic transformation matrices for the selected coordinate system. Since these transformation matrices change frequently, they should be recomputed about every 16 milliseconds.

Input to the level 1 world model comes from three sources:

1. From the task decomposition H module
   - Task state information
   - Requests for current or future joint positions, velocities, accelerations, torques, and frictions

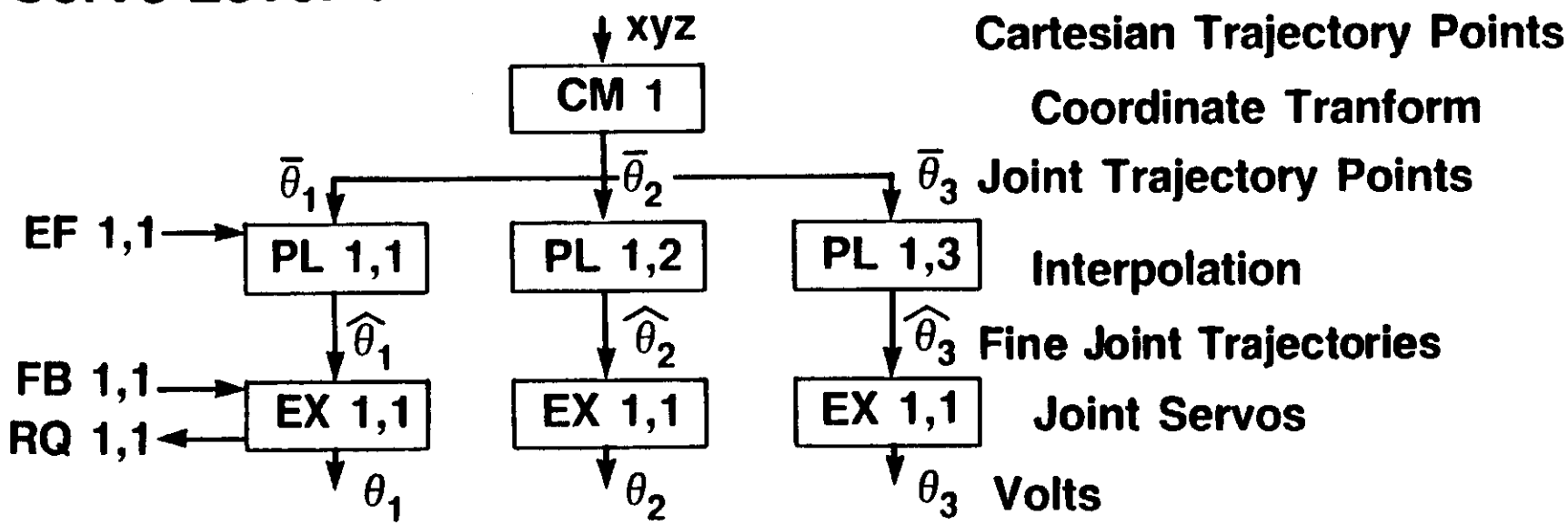


FIGURE 23: **The functions and interfaces in the level 1 H module.**

- Requests for current or anticipated future inertias, loads, and g-forces.

2. From the sensory processing G module

   - Detected, filtered, and scaled readings of sensors giving parameters such as positions, velocities, accelerations, torques, loads, frictions, and g-forces.

   - Correlations and differences between observed and predicted sensor readings.

3. From a priori information loaded during system initialization.

Requests to the level 1 M module consist of Read-Requests for the value of named variables. Delay between request and return of the information should be no more than a few bus read cycles. For a high performance manipulator, total loop delay at level 1, from sensory read, to actuator output should be less than 2 milli-seconds.

8.1.5 Sensory Processing

Level 1 sensory processing consists of scaling and filtering functions. Joint encoders are processed into radians or degrees. Tachometer and accelerometer readings are transformed into velocities and accelerations, and perhaps subjected to Kalman filtering to provide statistical best-estimates of measured variables. Limit checking is performed to detect out-of-spec conditions. Error flags are set when anomalies are detected.

Force, touch, and proximity sensor inputs can be scaled, filtered, and entered into the world model for sensory servoing. Use of such sensor data at the servo level requires that the data values be multiplied by a row in a Jacobian matrix in order to transform from sensor coordinates to joint coordinates. This matrix needs to be updated about every 16 milliseconds.

At level 1, emphasis is on short time delay. Data must be sampled, processed, entered into the world model, accessed and used by the servo modules in less than two control cycles. This implies that sensor readings and other sensor outputs should be synchronized with the servo level executor clock so as to minimize time delays between sampling and acting on the sampled data.

Vision processing at level 1 consists of scaling and filtering, histogram equalization, edge enhancement, and other local or point operators. Vision information does not enter the world model at level 1 as it typically takes 16 milliseconds to scan a single TV frame.

In general, if a particular sensor system does not produce data for a particular control level, the data flow will bypass that level.

8.2 Level 2 -- Primitive Level

The primitive level computes inertial dynamics, and generates smooth dynamically efficient trajectories in a convenient coordinate frame.

Input commands consist of intermediate trajectory poses which define a path which has been checked for obstacles and is guaranteed free of collisions.

Feedback input consists of measured position, velocity, rotation rates, rate of closure to obstacles, etc. Feedback input is sampled every 16 milliseconds.

Output consists of evenly spaced trajectory points which define a dynamically efficient movement. Outputs are produced every 16 milliseconds. Delay between sensory data being sampled and output response from the Primitive level should be less than 32 milliseconds.

8.2.1 Input Commands

Input commands to level 2 are designated TC(2,r) r = 1,2,..., M, where M is the number of subsystems being controlled, as shown in Figure 24. Level 2 input commands are updated on average, about five to ten times per second, but not necessarily equally spaced in space or time. Level 2 outputs subcommands are evenly spaced in time, i.e. every 16 milliseconds.

8.2.1.1 Manipulator Motion

Level 2 manipulator input commands define desired end effector poses (position, velocity, force, and roll, pitch, and yaw orientation, rates, and accelerations at trajectory knot points) expressed in the coordinate system of choice.

8.2.1.2 Motion of the Transport Vehicle

Level 2 transporter input commands define desired FTS poses at trajectory knot points in the coordinate system of choice. The coordinate system chosen to express transporter commands may be different than the coordinate system chosen to express manipulator commands.

In the early implementations of the NASREM telerobot control system, the transporter and the manipulator will not be activated concurrently. However, in later versions, both the transporter and the manipulators may operate simultaneously. For example, the transporter may move to keep the work volume optimally positioned within the reach envelope of the manipulators while the manipulators are functioning.

8.2.2 Task Decomposition - The H Function

8.2.2.1 Job Assignment Modules

The job assignment modules JA(2,r) at level 2 for the manipulator and vehicle guidance subsystems split the computational task into x, y, z, roll, pitch, and yaw components in the coordinate system of choice. This permits parallel computation of these components by the planners and executors.

8.2.2.2 Planner Modules

The primitive level planners PL(2,s) compute dynamically efficient trajectories between trajectory knot points JC(2,s) defined as goals by JA(2,r). These computations typically involve dynamic interactions between coupled manipulator joint linkages and vehicle inertial cross products. Speed can be achieved by parallel computations. Each axis (x, y, z, roll, pitch, yaw) can be computed separately using dynamic equations whose coefficients change slowly compared to level 2 outputs (every 16 milliseconds). New coefficients can be updated every 128 milliseconds.

Subcommands in the planned trajectories PST(2,s,tt) are synchronized so that smoothly coordinated motions of the vehicle and manipulator are produced. Dynamic trajectories planned at the Primitive level must never call for motions that transform into joint velocities or forces that exceed the physical limits of joint actuators. It is the responsibility of the PL(2,s) planners to check for joint position, velocity, and torque limits, and if necessary, scale back planned trajectories PST(2,s,tt) so that the output subcommands from the EX(2,s) executors to the level 1 servos are always within the range of capabilities of the servo level.

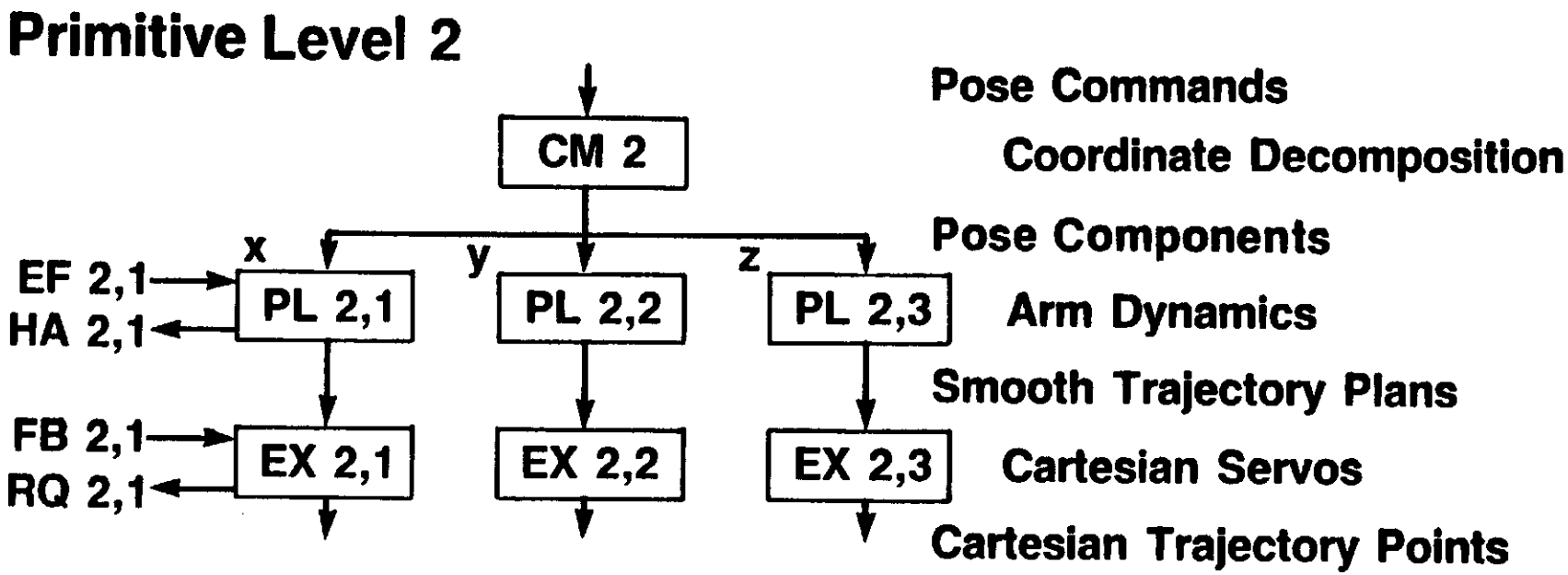


FIGURE 24: **The functions and interfaces in the level 2 H module.**

8.2.2.3 Executors

The planned trajectories PST(2,s,tt) from the planners PL(2,s) provide inputs to the executors EX(2,s). The primitive level executors EX(2,s) compare the current observed positions, velocities, forces, and stiffness in the coordinate system of choice with the commanded (or desired) positions, velocities, and forces defined by the planned trajectories PST(2,s,tt). The errors between the desired plan PST(2,s,tt) and observed values FB(2,s,t) are used to compute outputs designed to achieve the desired values. Level 2 executors thus perform position, velocity, force, and stiffness servoing in the coordinate system of choice. Synchronization between executor output subcommands STX(2,s,t) may be achieved by information exchanged through global memory.

8.2.3 Output Subcommands

Output subcommands from level 2, provide input commands to level 1. Level 2 outputs define desired subsystem trajectories in the coordinate system of choice. These outputs include stiffness and damping factors and other servo loop parameters. Feedback input FB(2,s,t) is sampled by the EX(2,s) executors every 16 milliseconds, and output subcommand values are updated every 16 milliseconds.

8.2.4 World Model

The world model at level 2 contains:

1. filtered parameters such as observed accelerations, velocities and positions of end effectors in a coordinate system of choice.

2. a dynamic model of the transporter system, with the ability to predict FTS accelerations and velocities in response to thruster forces.

3. a kinematic and dynamic model of the manipulators with similar predictive capabilities.

4. a computation process which computes kinematic and dynamic transformation matrices for the coordinate systems selected for the various subsystems. This modeling process must be able to develop a new set of transformation matrices about every 128 milliseconds.

Output from the model can be used by the JA(2) and PL(2,s) modules to plan and by the EX(2,s) modules to execute motion of the transporter and manipulators. Output from the model can also be used to predict sensory data.

8.2.5 Sensory Processing

Sensory processing modules at level 2 operate on filtered data from force, torque, and tactile sensors, accelerometers, rate gyros, and manipulator joint encoders. They compare observed forces, accelerations, velocities, and positions with predictions from the world model based on level 2 task commands. The sensory processing modules compute correlations and differences which are used by the world model to update the global memory. This updated information is used to compute better predictions for sensory processing, and to provide feedback to the planners and executors in the task decomposition module. Anomalous conditions enter error flags into the global memory.

Information from a variety of sensors is integrated over space and time to detect the 3-dimensional position, orientation, and dynamic motion of object features such as edges, corners, holes, and vertices.

Vision processing at this level consists of detection of 2-D image features such as edges and corners, and where possible, the transformation of these features into 3-D coordinate space. Level 2 vision

processing also integrates the motion of features into trajectories through space and time. These trajectories are expressed in the coordinate system of choice.

8.3 Level 3 - Elemental Move (E-Move)

Level 3 transforms symbolic commands for "elemental" movements (E-moves) into strings of intermediate poses which define motion pathways that are free of collisions and kinematic singularities.

Inputs consist of symbolic names of E-Moves, typically expressed as commands to achieve "key-frame" poses in the coordinate system of choice.

Outputs consist of trajectories of intermediate poses that avoid kinematic singularities and collisions with objects.

8.3.1 Input Commands

Level 3 of the task decomposition hierarchy shown in Figure 25 accepts elemental move commands which can be expressed as commands to achieve "key-frame" poses. (The term "keyframe" is derived from the field of cartoon animation. A keyframe in an animation sequence represents a particular relationship between the cartoon characters and objects in their environment at a key point in the story sequence. The keyframes define the story line, and are drawn by the principal artist and creator of the cartoon story. Intermediate frames are added by apprentice artists to fill in the action that connects the keyframes. [In the case here, the intermediate frames are added by the E-move level planners.] A string of keyframes can thus be viewed as a string of goal poses to be achieved by the characters in the cartoon. The E-move level takes each successive keyframe goal as an input command, and generates the string of intermediate poses needed to smoothly move the system from one keyframe to the next.)

Manipulator E-Move input commands TC(3) consist of trajectory segments such as <reach-to X>, <approach-grasp-point Y>, <grasp>, <move-to Z>, <depart W inches>, <track-edge E with-camera>, <pull-back-while-nulling-gripper-torques>, <apply-force-vector-F> etc.

FTS Transporter input commands TC(3) to the E-Move level call for movements such as <go-to-docking-approach X>, <yaw Z degrees>, <go-forward W inches>, etc.

The E-Move level decomposes these commands into paths or trajectories consisting of strings of intermediate poses, or trajectory knot points. These output strings of intermediate poses are not necessarily evenly distributed in time, but are chosen so as to steer the subsystem output trajectories around all problem areas such as joint limits, kinematic singularities, and obstacles.

8.3.2 Task Decomposition - The H Module

8.3.2.1 Job Assignment

The E-Move level Job Assignment modules JA(3,r) separate translation from rotation and assign the computation of intermediate poses and trajectory knot points to separate position and orientation planners PL(3,s) and executors EX(3,s). This permits parallel computation of intermediate trajectories for position and orientation. There is an E-Move level Job Assignment module for each subsystem (i.e. transporter, manipulators, camera platforms, etc.).

It is also the role of the E-move level job assignment module to select the coordinate system most appropriate for computing the execution of the commanded E-move.

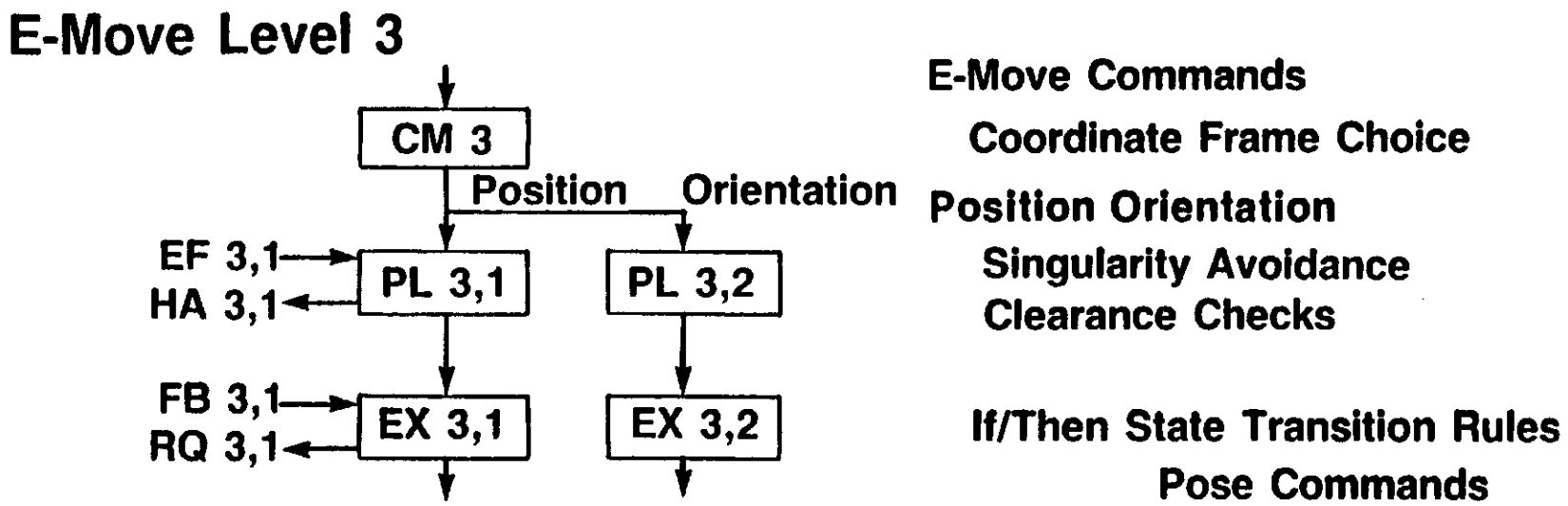


FIGURE 25: **The functions and interfaces in the level 3 H module.**

8.3.2.2 Planning

The E-Move level planners, PL(3,s) plan a sequence of intermediate poses for the vehicle or manipulator which will accomplish the commanded E-Moves. The E-Move planning modules PL(3,s) are responsible for generating problem-free trajectory segments that extend at least one second into the future. The planners check to see if there is clearance between the vehicle or manipulator and potential obstacles in the world. The planners also check whether any of the intermediate poses lie near kinematic singularities, or whether straight line trajectories between intermediate poses come near to obstacles or singularities. If so, the planners interject additional intermediate poses so as to safely skirt potential problem areas.

Each E-Move planning module PL(3,s) adds a new trajectory knot point to the end of the current plan on average about as rapidly as the corresponding E-move executor selects a new knot point from the beginning of the plan to output to the Primitive level. Thus, the E-move planners generate an updated plan about ten times per second, and the planning module always has prepared a plan which looks about one second, or ten trajectory knot points into the future.

E-Move trajectories can be be planned in real-time as they are being executed. In a known environment, such as in or around the space station, however, commonly used E-Move trajectories PST(3,s,tt) can be preplanned and recorded. These recorded trajectories can then be invoked by naming the file in which they are stored. During the execution of these recorded trajectories, the system automatically detects and avoids unexpected objects.

A plan PST(3,s,tt) may be defined as a path through a tree of potential futures. Each node of the PST corresponds to a planned action, and each edge corresponds to an expected result of the action. The edges can thus carry a probability and a cost- benefit value (or objective function) corresponding to the probability and benefit of the result. This enables a computation of the expected cost/benefit value of the planned sequence of actions. In planning graphs with multiple paths, the different traces can be evaluated relative to each other.

Edges can also carry a list of constraints and enabling, or disabling, conditions. This means that state graphs representing plans can represent a variety of possible conditions that can be invoked by the execution modules EX(3,s) at execution time.

Whether the E-Move plan PST(3,s,tt) is planned in real-time or pre-recorded, information from the world model about the current, or anticipated future state of the world, can be used by the executor EX(3,s) to modify these E-Move trajectories, to control branches, to vary parameters such as speed, or end-point position or velocity, and to effect synchronization and timing for smooth trajectories and coordinated maneuvers at end-points.

Real-time planning implies that a new plan is generated approximately once per output subcommand. If task execution proceeds exactly as planned, then each new plan generated uses all of the remaining previous plan. The planning process need only add one additional step onto the end of the current plan to make up for the one step taken off the front of the plan to be executed.

Of course, task execution does not always proceed as planned. Events can occur which change the state of the world, and hence the expected result of actions on the world. Events can cause changes in world model objective function parameters. This can change the cost-benefit value of states of the world.

Events can thus require changes in the current plan to produce a new plan. Some events require only modest changes in the current plan, such as a modification of the speed or acceleration profile -- a sort of mid-course correction. Others, require a completely new plan -- a completely new type of E-Move trajectory. In the latter case, it may become necessary to issue a <Pause> subcommand, or substitute a

preplanned <Error_Recovery> routine, until the new plan can be generated.

8.3.2.3 Execution

The execution submodules EX(3,s) are responsible for issuing the first intermediate pose in the current plan to the appropriate task decomposition modules at the Primitive level. The execution submodule also monitors the progress of the Primitive level as it attempts to reach the commanded trajectory points.

Output from the E-Move execution submodule consists of trajectory points, poses, and velocities in the coordinate system of choice. The output commands carry a field which designate the choice of coordinate system.

8.3.3 World Model

The world model at the E-Move level contains information defining the position and orientation of features such as edges, vertices, and bounding surfaces of objects. This information is used by PL(3,s) planners to check clearances and perform local obstacle avoidance, and to compute poses of the transporter and manipulator systems relative to objects; poses such as approach/depart points, grip orientations, station-keeping poses, dock and grasp poses, and the aiming of sensors.

The world model also contains representations of kinematic singularity points in a form that makes it convenient to detect potential problems.

The information about object features contained in the world model makes it possible to generate predictions of image features, such as edges, corners, contours, etc. to be used in the interpretation of image data. The interaction between the sensory processing system and the world model is described in more detail in [46,54].

8.3.4 Sensory Processing

The E-move level is the first level at which information derived from image processing is extensively used. The sensory processing module at the E-move level compares image features predicted by the world model with observed image features detected by the primitive level G module. Correlations and differences between predicted and observed image features are integrated and used to compute object features. These detected object features are used to update the global memory, and to provide input to the higher level task level G modules.

Information derived from vision systems about the position and orientation of object features such as edges, surfaces, corners, holes, etc. is used at the E-move level to avoid collisions, to approach objects to be manipulated, and track features on moving objects. This information may be used to servo camera pan, tilt, and zoom as well as to guide manipulators and end effector tooling.

8.4 Level 4 - Object/Task Level

The task level transforms goals defined in terms of desired actions to be performed on objects, or desired relationships to be achieved between objects in the world, into a series of control system E-moves designed to achieve those relationships.

Input consists of a command to perform a task on an object in order to achieve a desired relationship of that object relative to other objects in the world.

Output consists of a string of E-move commands to a transporter, manipulators, or cameras that will have the desired effect.

8.4.1 Input Commands

The Task Level shown in Figure 26 is the highest level in the individual Flight Telerobot System (FTS). The task level receives commands from the Service Bay controller (Level 5) to maneuver the FTS relative to some workplace, or target object, and to execute a particular task, or sequence of tasks, in an environment containing multiple objects, obstacles, and unexpected hazards. Examples are <Replace ORU X>, <Attach fueling mechanism Y>, <Cut insulation blanket>, <Inspect surface Z>, <Close/Open valve W>, etc.

8.4.2 Task Decomposition - The H Function

8.4.2.1 Job Assignment Module

The Job Assignment Module JA(4,r) at the task level is the FTS system coordinator. JA(4,r) receives commands from the Service Bay level executor EX(5,s), and interprets those commands in the context of what is present in the World Model. The JA(4,r) coordinator is an expert system which decides what jobs each of the FTS subsystems should do to accomplish the task level input command. It issues jobs to the planning modules PL(4,s) of the various FTS subsystems to generate plans as to the sequence of actions to be performed in order to achieve the desired result.

The JA(4,r) expert system examines the current state of the task and the object of the task, and issues job commands JC(4,s) to the level 4 Planners to generate the type of maneuver to be performed relative to the object.

The JA(4,r) coordinator breaks down the input task commands into a set of job commands to be decomposed into elemental move commands by a set of subordinate controller modules (Transport system, Left Camera, Left Arm, Stabilizer, Right Arm, Right Camera). These decompose their respective job commands into a temporal sequence of elemental move commands.

8.4.2.2 Planning

The subsystem planning and execution modules may be:

1. Transport system = {PL(4,1), EX(4,1)}
2. Left Camera = {PL(4,2), EX(4,2)}
3. Left Arm = {PL(4,3), EX(4,3)}
4. Stabilizer Arm = {PL(4,4), EX(4,4)}
5. Right Arm = {PL(4,5), EX(4,5)}
6. Right Camera = {PL(4,6), EX(4,6)}

The transport system is what moves the body, or shoulders, of the telerobot system. It may consist of a remote manipulator system (RMS), an orbital maneuvering vehicle (OMV), or some other mechanism. In early implementations, this transport system may simply move the telerobot system into position, and then remain stationary while the telerobot does its work. In later implementations, the control for the transport system can be integrated with the controls for the telerobot so that the motions of the transporter can be coordinated with those of the telerobot arms. This complicates the control computations, but greatly enhances the effective work volume of the telerobot.

The planners PL(4,s) may select predetermined, well practiced, and optimized plans (i.e. E-Move sequences) by simply naming the file in which they are stored. Generic plans, or scripts, can be selected from files, or E-Move sequences can be computed in real-time by artificial intelligence

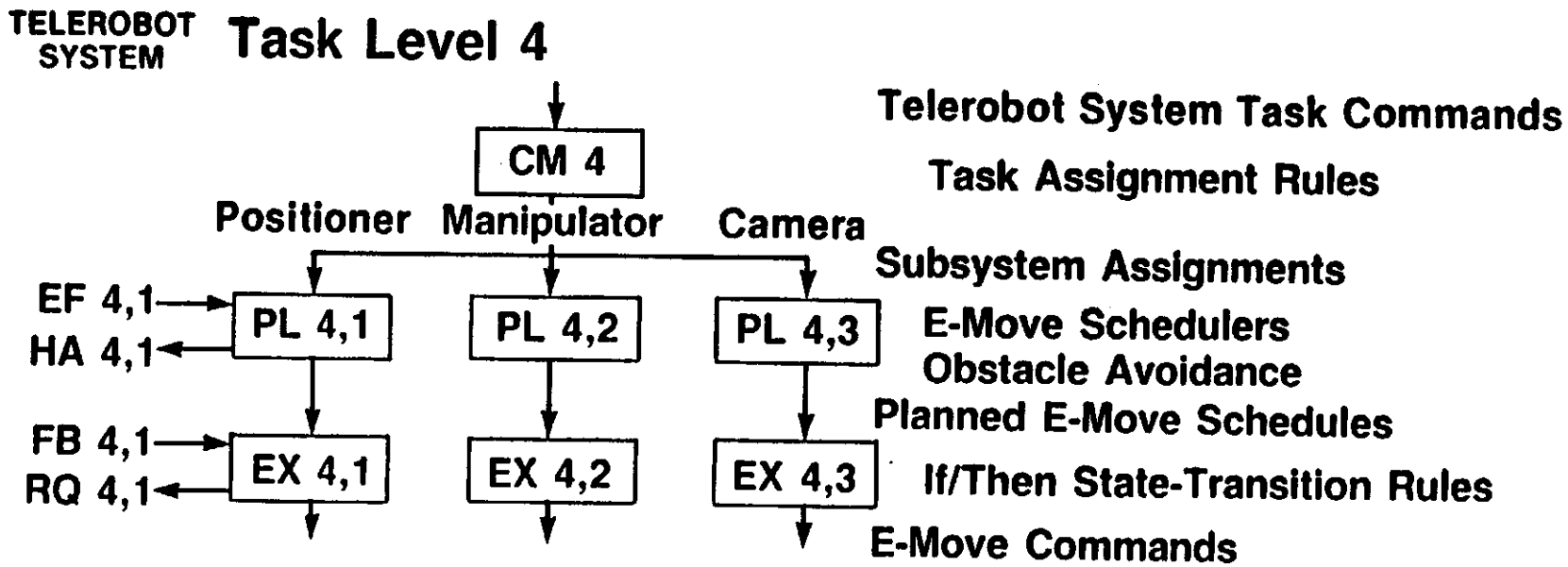


FIGURE 26: **The functions and interfaces in the level 4 H module.**

planning and search strategies, by operational research linear programming techniques, or by game theoretic methods of cost-risk analysis and utility theory. Planners at the object/task level look ahead about 7 ±3 E-Moves (or 7 ±3 E-Move time periods).

In order to facilitate planning, E-Moves may carry lists of preconditions, resource requirements, expected costs, expenditure of resources, and risk factors. These parameters may either be specified as constants or as functions of world model state variables to be evaluated in real-time.

Planned coordination of E-Moves between cooperating subsystems needed for transport system maneuvers, manipulator motions, and sensor coordination, pointing, and focusing is organized and synchronized at the object/task planning level. Synchronization can be carried out by including a timing field in the plans generated by the object/task level planners PL(4,s). The timing field may carry an <execute immediate> flag, a <begin on condition> flag, a <begin at clock time x> flag, a <begin after delay y> flag, a <begin with delay y after condition x> flag, an <end before clock time x> flag, a <do-until condition x> flag, or a <do-while condition y> flag.

The transport system planner PL(4,1) contains criteria for positioning the FTS system at an optimal work position. The evaluation function EF(4,1) may cause the PL(4,1) planner to generate E-Move sequences that satisfy least energy, or shortest time, or least risk criteria.

The manipulator planners PL(4,3) and PL(4,5) are responsible for turning manipulation job commands expressed in terms of what action should be performed on objects into a coordinated sequence of E-Moves expressed in terms of what elemental movements the manipulator grippers should make.

Generic manipulation plans for performing different kinds of tasks such as replacing ORUs, cutting, and attaching tasks can be developed ahead of time through the supervisory control teaching/learning techniques of Sheridan. Such generic tasks can carry symbolic variables that are converted to current geometric dimensional data by the real-time planners PL(4,s) and executors EX(4,s) using information supplied by the world model. The world model contains the geometric dimensions of objects as well as numerical values of position and orientation. It also contains knowledge of the position and orientation of object features, such as surfaces, edges, corners, and potential grip points. Using this information, the PL(4,s) manipulation planners can turn generic plans into specific plans in real-time for a whole class of objects.

The transport system planner PL(4,1) must coordinate its plan with the camera and arm planners PL(4,s) so that the FTS can maneuver itself into a position that is favorable for the manipulation task, and stay with the object the manipulator is working on.

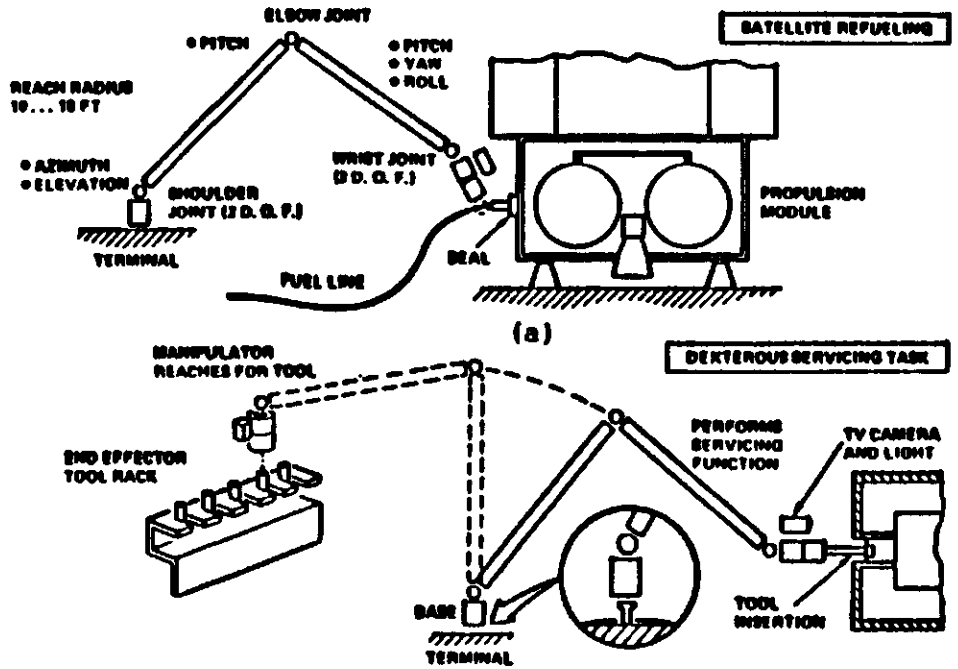
8.4.2.3 Execution

Each of the EX(4,s) executors can be viewed as a state sensitive expert system. This means that the rule base contains a state variable, and the rules can be executed in a state-transition table. The rules correspond to edges in a state-graph, and the nodes correspond to states.

In all plans, whether prerecorded or computed in real-time, information about the state of the world can be used by the EX(4,s) executors to modify planned E-Move sequences, to control branches, to vary parameters such as speed, to effect synchronization and timing for cooperative coordinated movements and synchronized maneuvers between multiple arms, or arms, eyes, and fingers.

8.4.3 World Model

As shown in Figure 27, the world model at the task level contains information defining the identity, position, and orientation of objects in the vicinity of the FTS system such as satellites to be serviced, the location of replaceable ORUs, positions of spare ORUs, tools, other FTS systems, and space



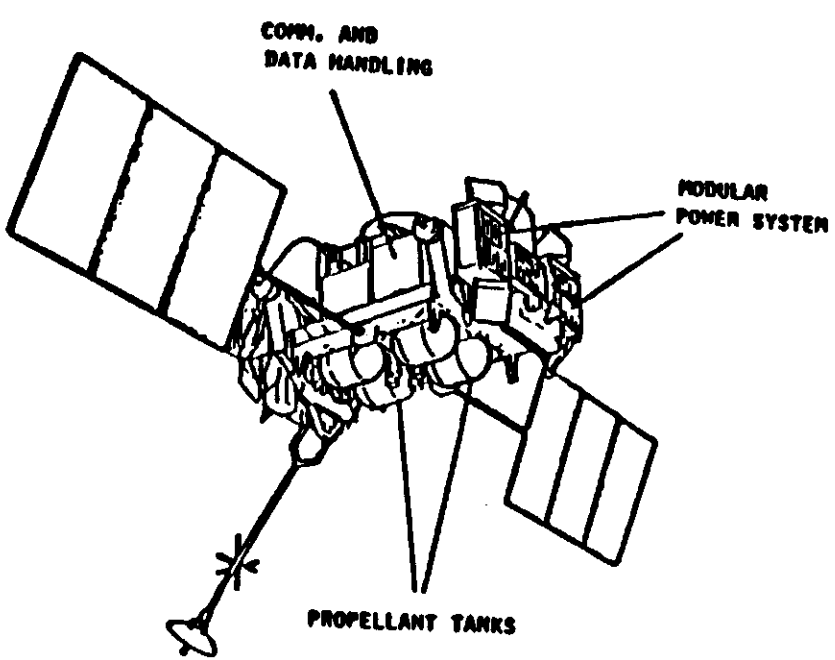


FIGURE 27: **Telerobot task level 4 world model maps.**
(a) refueling task
(b) servoing task
(c) module replacement task

station structures such as tool holders, struts, fuel containers, etc. This information is used by PL(4,s) planners to schedule E-Moves and compute E- Move parameters so as to avoid obstacles, plan efficient movement sequences, and efficiently carry out task assignments.

Information about objects is indexed both by position in space as denoted by the maps, as in Figure 27, and by name. Lists of the named objects contain pointers to lists of attributes for each named object. Attributes such as shape, size, velocity, type, condition, surface reflectance, and intended use are thus represented in the world model. Shape can be denoted by solid modeling techniques, as well as by wire frame, or bounded surface representations.

8.4.4 Sensory Processing

Sensory processing at the object/task level compares observed object features detected by the E-move level with predicted object features from the task level world model. Observed features may be derived from brightness images, range images, structured light images, and tactile sensors. Predicted object features are generated by the world model from maps, geometrical descriptions, system state variables, and lists of attributes of objects. Predictions include the position, orientation, and motion of object features. These predictions are sent to the task level sensory processing modules for comparison with observations of object features from the E-move level sensory processing modules.

The sensory processing modules compare predicted and observed object features. Correlations and differences are integrated and evaluated to detect object positions, orientations, and identities. This information is sent to the task level M module to update the world model, and is also relayed upward to the Service Bay level sensory processing modules.

8.5 Level 5 - Service Bay Control Level

The service bay level transforms goals defined in terms of repair and maintenance requirements for an entire spacecraft into sequences of actions to be performed on objects such as ORUs.

Input consists of commands to a service bay manager to perform a set of service and maintenance operations on specific spacecraft.

Output consists of a string of object task commands to one or more FTS systems, automatic berthing fixtures, materials transport mechanisms, and possibly one or more astronauts.

The service bay control level corresponds to the Workstation level in the NBS AMRF.

8.5.1 Input Commands

As shown in Figure 28, input to the Service Bay Control level consists of commands to carry out servicing tasks on specific spacecraft. This typically requires the coordinated actions of one or more telerobot servicing systems, berthing fixtures, and requires tools and parts to be delivered to the service bay at specific times by transfer pallet delivery mechanisms. The control of transfer pallets for delivering parts and tools is under a materials transfer control module which is also at the service bay control level.

Examples of input TC(5) to the Service Bay Control Level are commands such as <Repair Satellite X>, <Refuel Satellite W>, <Replace Subsystem Y on Satellite Z>, <Move Tool Kit K to Bay 2>, etc. Commands may take several minutes to hours to carry out.

There also exists a part and tool management system at the service bay control level. This system provides the storage and retrieval system and the transportation mechanisms to deliver kits of parts and tools to the proper buffer storage areas in the various service bays.

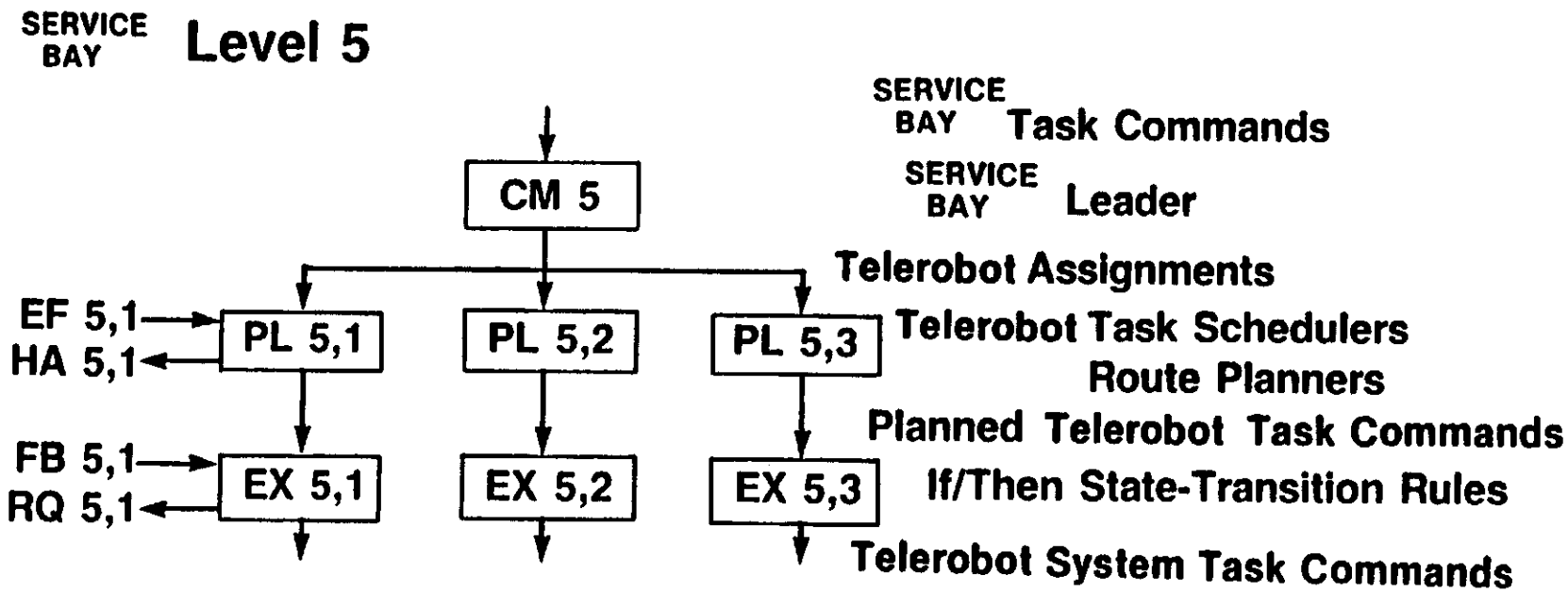


FIGURE 28: **The functions and interfaces at the level 5 H module.**

8.5.2 Task Decomposition - The H Module

8.5.2.1 Job Assignment

The job assignment managers at the Service Bay Control Level are the service bay managers. These are designated JA(5,r), r=1, 2, ... M where M is the number of service bays. These JA(5,r) service bay manager modules contain expert systems which partition the service bay commands TC(5) into telerobot and astronaut job assignments JA(5,s), s=1, 2, . . . N where N is the number of telerobotic systems in the service bays.

8.5.2.2 Planning

The planners PL(5,s) at the Service Bay level accept job assignments from their respective service bay manager, JA(5,r). The planners schedule tasks for individual telerobot systems, astronauts, berthing fixtures, and tool and part kit buffers. For example, in a service bay with two telerobots, an astronaut, a berthing fixture, and a kit buffer:

1. PL(5,1) would be the task scheduler for telerobot #1
2. PL(5,2) would be the task scheduler for telerobot #2
3. PL(5,3) would be the task scheduler for the astronaut
4. PL(5,4) would be the task scheduler for the berthing fixture
5. PL(5,5) would be the task scheduler for the kit buffer.

Task schedulers generate plans based on service bay resource utilization, satellite servicing priorities, tool and part availability, and fixturing requirements and constraints.

The service bay task planners PL(5,s) may select predetermined, well practiced, and optimized coordinated task plans for routine servicing operations by naming the file in which they are stored. However, task plans can be also be computed, or recomputed, in real-time by artificial intelligence planning and search strategies. Operational research linear programming techniques, or game theoretic methods of cost-risk analysis, utility theory, and value-driven decision methodologies can also be used. Planners at the Service Bay level look ahead about 7 $\pm$3 Tasks (or 7 $\pm$3 Task time periods).

In order to facilitate planning, telerobot system task commands may carry lists of preconditions, resource requirements, expected costs, expenditure of resources, and risk factors. These parameters may either be specified as constants or functions of world model state variables.

In all plans, whether prerecorded or computed in real-time, information about the state of the world can be used by the EX(5,s) executors to modify planned task sequences, to control decision points, to vary parameters such as speed, to effect synchronization and timing for cooperative coordinated movements and synchronized maneuvers between telerobot systems.

8.5.2.3 Execution

For each service bay planner PL(5,s), there is an executor EX(5,s). The service bay executors may be viewed as state sensitive expert systems that work from a set of IF/THEN state transition rules. When feedback FB(5,s) indicates that a subgoal in the PST(5,s,tt) plan has been achieved, the executor EX(5,s) selects the next vehicle task command PST(5,s,tt+1) in the planned vehicle task schedules. It then issues this planned command as an actual vehicle task command STX(5,s,t). Output from the service bay executors EX(5,s) consists of task commands to individual systems, i.e., telerobots, astronauts, fixtures, and buffers.

8.5.3 The Service Bay Level World Model

The world model at the service bay level contains layout maps of the service bay, such as shown in Figure 29, indicating the position and orientation of the satellite to be serviced, the positions of berthing fixtures, tool and part buffers, and other objects and structures within the service bay. These maps may represent objects in either telerobot or world coordinates, or both types of maps may be maintained. The world model should contain processes for translating the representation of any object, region, or feature in world coordinates into telerobot coordinates (or vice versa) within one input command update cycle at the service bay level. Maps contain representation of the location and boundaries of service bay features such as interior surfaces, storage bins, etc.

The world model also contains lists of objects indexed both by name and map coordinates. Task specific lists may also be constructed from time to time during execution using other indices such as range, color, size, or mass.

8.5.4 Sensory Processing

Sensory processing at the service bay level compares measured positions of service bay surfaces, objects, and other telerobot systems with information derived from the world model maps and lists of objects. At this level, sensory data from vision, tactile, force, and position sensors has been fused into observations of objects and descriptions of their spatial relationships to each other.

8.6 Level 6 - Operations Control Level

Level 6 decomposes input commands expressed as prioritized lists of satellites requiring service into servicing schedules for the various service bays.

The operations control level corresponds to the Cell level in the NBS AMRF.

8.6.1 Input Commands

As shown in Figure 30, input commands to the H module at the operations control level come from the space station mission plan. They consist of commands to the operations control level to schedule the servicing of the entire backlog of of satellites awaiting service. The operations control input commands include priorities related to the space station satellite servicing mission objectives.

8.6.2 Task Decomposition - The H Module

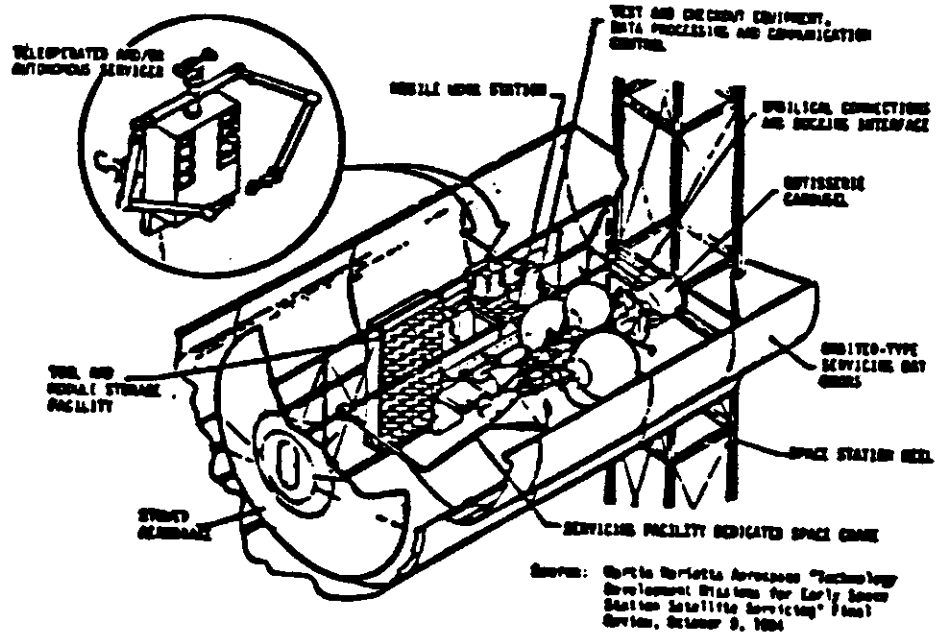
8.6.2.1 Job Assignment Manager

The job assignment manager JA(6) at the operations control level assigns satellites and servicing resources such as telerobot servicers, astronauts, parts, and tools to service bays. An example assignment might be:

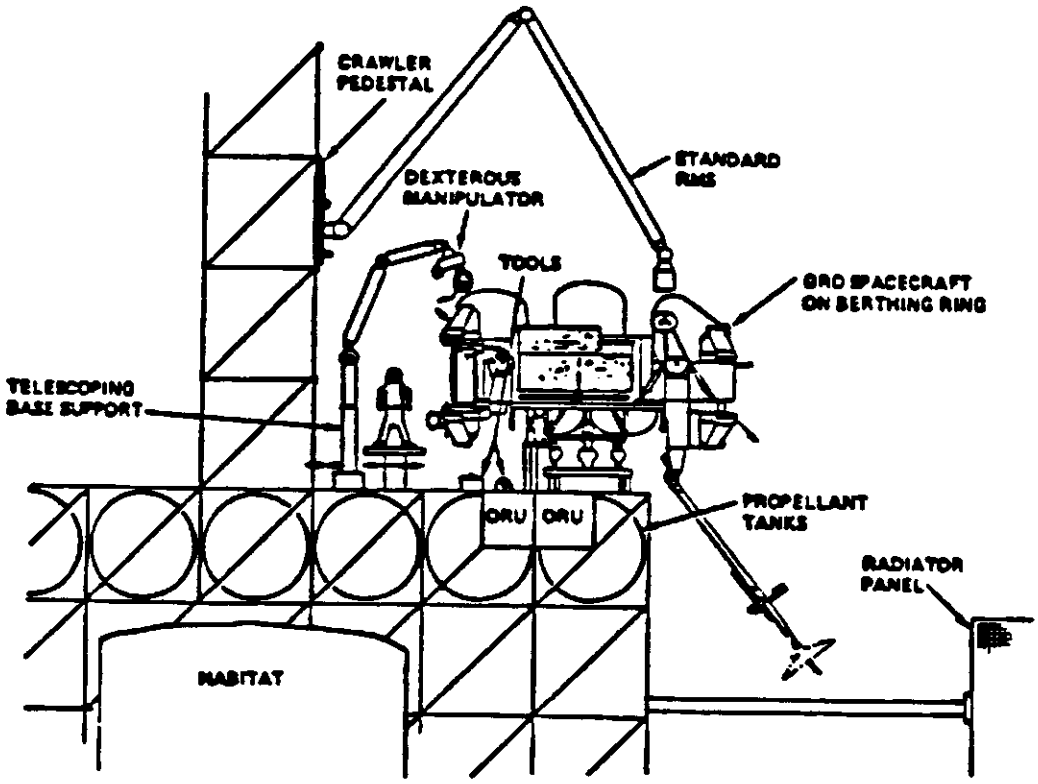
Service satellite A in bay 1
FTS #1 and #2 report to bay 1
Provide parts and tools as required

Service satellite B in bay 2
FTS #3 and astronaut #1 report to bay 2
Provide parts and tools as required

Service satellite C in bay 3



(a)



(b)

FIGURE 29: Service bay level 5 world model maps
(a) zoom out/oblique view
(b) zoom in/end view

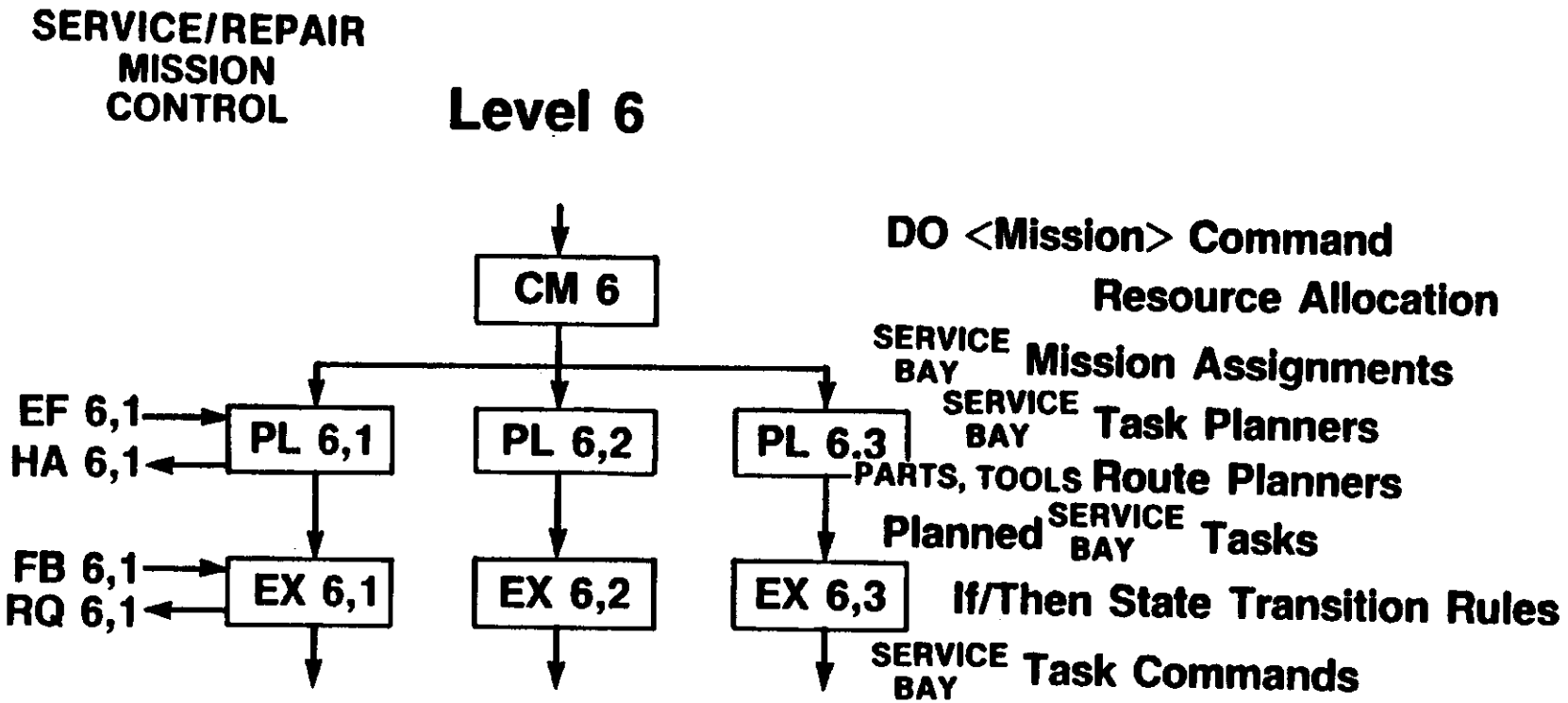


FIGURE 30: **The functions and interfaces in the level 6 H module.**

FTS #4 report to bay 3
Provide parts and tools as required

The assignment functions of the mission level job assignment manager may be done manually. The system specified here provides the interface tools for a human configuration manager to easily ask "What if?" questions of the world model, to display the results in graphical form, and thereby enable a human planner to generate the JA(6) group assignments.

The computer assisted process planning system developed at the National Bureau of Standards for planning manufacturing operations [55] has many features that could be used in designing the interface to the mission level planners.

8.6.2.2 Planning

The operations control level contains a planner PL(6,s) for each service bay. Each planner PL(6,s) generates a schedule of servicing activities PST(6,s,tt) that the s service bay must perform in order to accomplish the satellite servicing mission. The initial a priori form of the plans PST(6,s,tt) may be developed manually using the same type of interactive programming tools as used to develop the JA(6) assignments.

There is also a PL(6,s) planner for the parts and tools handling system. This planner plans kits of parts and tools for the various servicing tasks. These plans are based on the servicing requirements. The parts and tools planner then schedules the delivery of these kits to the proper buffer storage areas in the service bays.

8.6.2.3 Executors

For each service bay there is an executor EX(6,s) that monitors the state FB(6,s,t) of the servicing task, and steps through the plan, issuing subcommands STX(6,s,t) to the service bay level controllers at the proper times. There is also an executor for the delivery system which monitors the movement of parts and tools throughout the space station service bay complex.

8.6.3 World Model

The world model at the operations control level consists of maps of the space station including pathways for telerobot systems to move from one service bay to another. This model will change with time reflecting the construction. Figure 31 shows the map expected for the space station after 15 shuttle launches while Figure 32 shows the map at completion.

The world model also contains lists of telerobot systems, satellites awaiting service, and the location of other resources.

8.6.4 Sensory Processing

Sensory processing at the operations control level compares expected servicing completion times with observed progress. This information is used in service bay operations planning, and in the sequencing of satellites through the service bays.

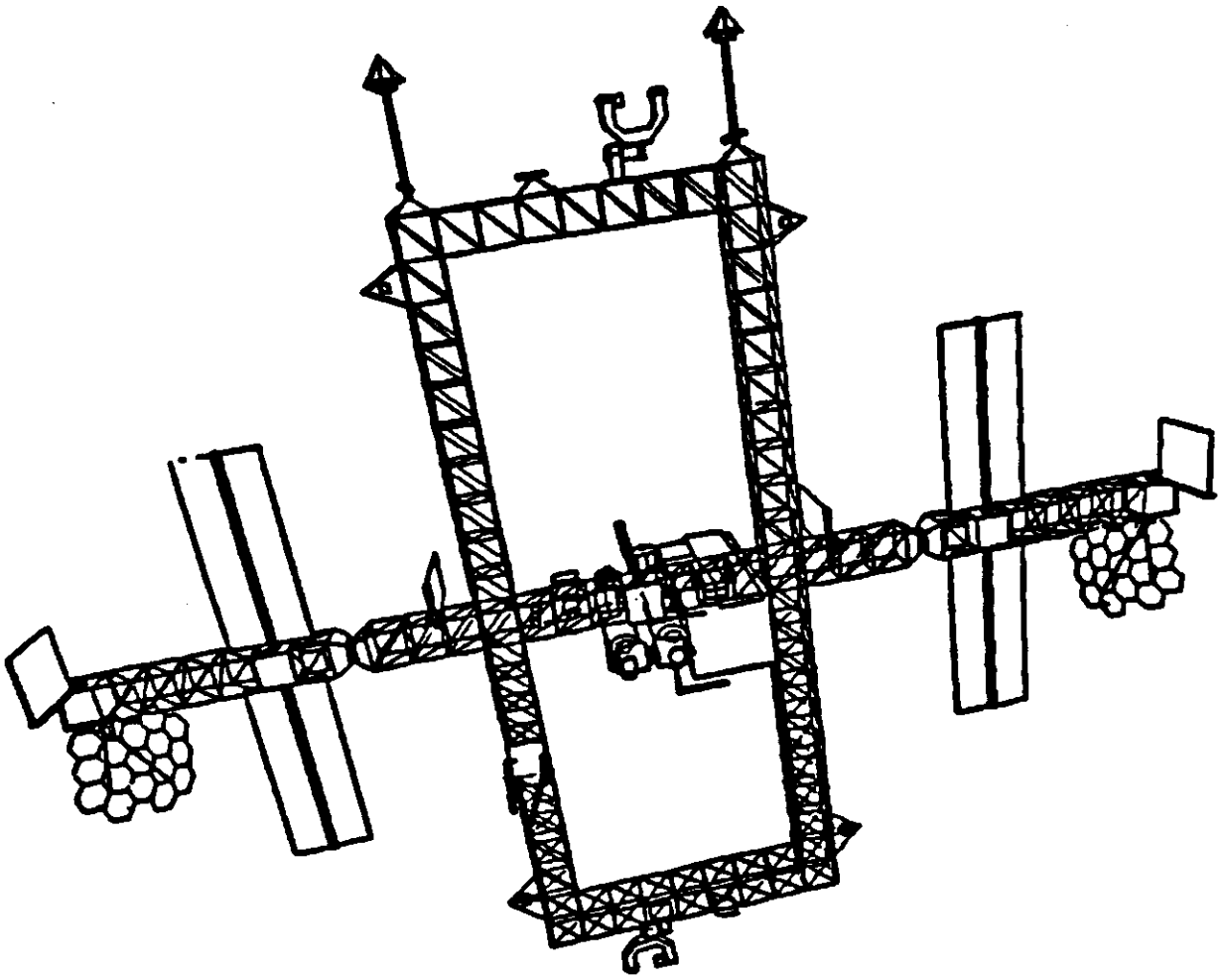


FIGURE 31: **Space station after 15 shuttle launches.**

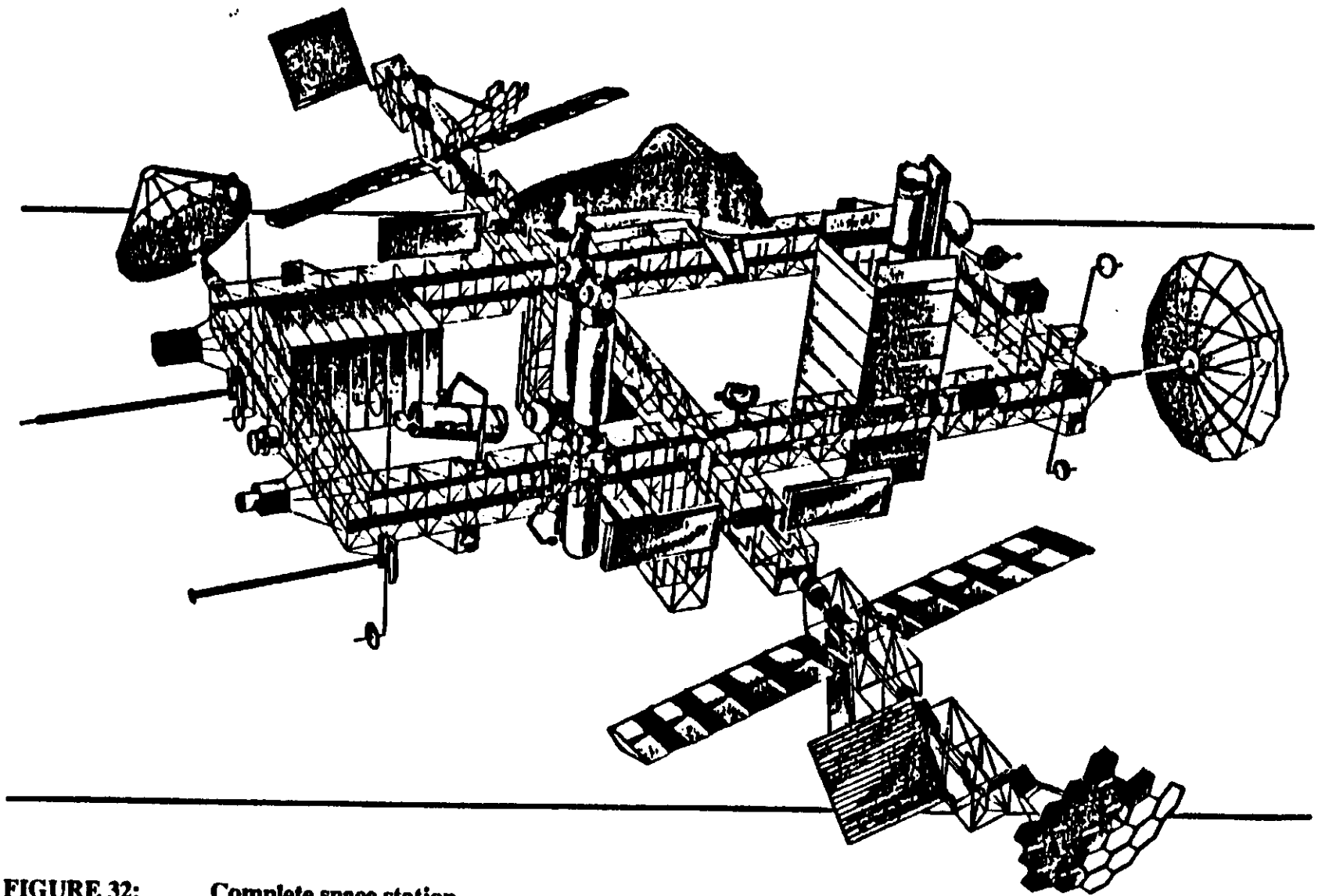


FIGURE 32: **Complete space station.**

9. REFERENCES

[1] V.D. Hunt, Smart Robots: A Handbook of Intelligent Robotic Systems, (New York, Chapman and Hall, 1985), p. 99.

[2] E. Johnsen, "Telesensors, Teleoperators, and Telecontrols for Remote Operation," IEEE Trans. on Nuclear Science, Vol. NS13, 1966, p. 14.

[3] R. Goertz, "Manipulation Systems Development at ANL," Proc. of 12th RSTD Conf., 1964, p. 117.

[4] J. Vertut and P. Coiffet, Teleoperation and Robotics Evolution and Development, (Englewood Cliffs, Prentice Hall, 1984).

[5] G. Kohler, Manipulators Type Book, (Munich, Verlag Karl Thieming, 1981).

[6] C. Flatau, "Compact Servo Master-Slave Manipulation with Optimized Communication Links," Proc. of 17th RSTD Conference 1969, p. 154.

[7] R. Mosher, B. Wendel, "Force Reflecting Electro-Hydraulic Servo Manipulation," Electro Technology, Vol. 66, 1960, p. 138.

[8] J. Charles, J. Vertut, "Cable Controller Deep Submergence Teleoperation System," Proc. of 2nd RMS Conference, 1975.

[9] P. Mosher, "Exploring the Potential of a Quadruped", SAE, SAE Report 690191, 1969.

[10] R. McGhee, "Control of Legged Locomotion Systems," Proc. of 18th Joint Automatic Control Conference, 1977, p. 205.

[11] M. Raibert, "Experiments in Balance with 3D One-legged Hopping Machine", Robotics Research, Vol. 3, #2, 1984, p. 75.

[12] M. Brady, et.al., ed. Robot Motion: Planning and Control, (Cambridge, MIT Press, 1982).

[13] D.E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Trans. Man-Machine Systems MMS- 10, 1969, p. 47.

[14] D.E. Whitney, "The Mathematics of Coordinated Control of Prostheses and Manipulators," Journal of Dynamic Systems, Measurement, Control, Dec. 1972, p. 303.

[15] R.P. Paul, "Manipulator Path Control," IEEE Int. Conf. on Cybernetics and Society, New York, p. 147.

[16] R.P. Paul, "Manipulator Cartesian Path Control," IEEE Trans. Systems, Man, Cybernetics SMC-9, 1979, p. 702.

[17] R.P. Paul, Robot Manipulators: Mathematics, Programming, and Control, (Cambridge, MIT Press, 1981.)

[18] R. H. Taylor, "Planning and Execution of Straight-line Manipulator Trajectories," IBM J. Research and Development 23 1979, p. 424.

[19] J.M. Hollerbach, "A Recursive Formulation of Lagrangian Manipulator Dynamics," IEEE Trans. Systems, Man, Cybernetics SMC-10, 11, 1980, p. 730.

[20] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-line Computational Scheme for Mechanical Manipulators," J. Dynamic Systems, Measurement, Control, 102, 1980, p. 69.

[21] C.S.G. Lee, P.R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 532.

[22] E.E. Binder, J.H. Herzog, "Distributed Computer Architecture and Fast Parallel Algorithm in Real-Time Robot Control," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 543.

[23] M.H. Raibert and J.J. Craig, "Hybrid position/force control of manipulators," J. Dynamic Systems, Measurement, Control, June, 1981, p. 126.

[24] H. Kazerooni, T.B. Sheridan, P.K. Houpt, "Robust Compliant Motion for Manipulators, Part I: The Fundamental Concepts of Compliant Motion," IEEE Journal of Robotics and Automation, Vol. RA-2, No. 2, June 1986, p. 83.

[25] H. Kazerooni, P.K. Houpt, T.B. Sheridan, "Robust Compliant Motion for Manipulators, Part II: Design Method," IEEE Journal of Robotics and Automation, Vol. RA-2, No. 2, June 1986, p. 93.

[26] D.F. Golla, S.C. Garg, and P.C. Hughes, "Linear State-Feedback Control of Manipulators," Mech. Machine Theory, 16, 1981, p. 93.

[27] F. Freund, "Fast Nonlinear Control with Arbitrary Pole- placement for Industrial Robots and Manipulators," Int. J. Robotics Research 1, 1, 1982, p. 65.

[28] W. Hamel and M. Feldman, "The Advancement of Remote Technology: Past Perspectives and Future Plans," Proc. 1984 National Topical Meeting on Robotics and Remote Handling in Hostile Environments, ANS, Gatlinburg, TN, April, 1984.

[29] J. Herndon and W. Hamel, "Analysis of the Options--Rationale for Servomanipulators in Future Reprocessing Plants," Proc. 1984 National Topical Meeting on Robotics and Remote Handling in Hostile Environments, ANS, Gatlinburg, TN, April, 1984.

[30] W. Hamel et. al., "Advanced Teleoperation in Nuclear Applications," 1984 ASME International Computers in Engineering Conf., Las Vegas, NV. August, 1984.

[31] W. Hamel and H. Martin, "Robotics-related Technology in the Nuclear Industry," Proc. SPIE, Vol. 442, August, 1983, pp. 97-107.

[32] A. K. Bejczy and J. K. Salisbury, "Controlling Remote Manipulators Through Kinesthetic Coupling," Computers in Mechanical Engineering, July, 1983, pp. 48-60.

[33] D. L. Grisham, "Monitors 1980: Now There are Two," Proc. 28th Conf. Remote Systems Technology, Vol. 2, American Nuclear Society, 1980, p. 83.

[34] A. Liegiois et. al., "Learning and Control for a Compliant Computer Controlled Manipulator," IEEE Trans. Auto. Control, Vol. AC-25, No. 6, Dec, 1980, pp. 1097-2003.

[35] W.A. Perkins, "A Model Based Vision System for Industrial Parts," IEEE Trans. on Computers, Vol. C-27, 1978, p. 126.

[36] G.L. Gleason, G.J. Agin, "A Modular Vision System for Sensor-controlled Manipulation and Inspection," Proc. 9th Int. Symposium on Industrial Robots, 1979, p. 57.

[37] M.R. Ward, et.al., "CONSIGHT An Adaptive Robot with Vision," Robotics Today, 1979, p. 26.

[38] J. Albus, E. Kent, M. Nashman, P. Mansbach, L. Palombo, M.O. Shneier, "Six Dimensional Vision System," SPIE, Vol. 336, Robot Vision, 1982, p. 142.

[39] R.C. Bolles, R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local Feature-Focus Method," Int. Journal of Robotics Research, Vol. 1, 1982, p. 57.

[40] R.C. Bolles, P. Horaud, M.J. Hannah, "3DPO: Three Dimensional Parts Orientation System," Proc. of The Int. Joint Conf. on Artificial Intelligence, August 1983, p. 1116.

[41] T.F. Knoll and R.C. Jain, "Recognizing Partially Visible Objects Using Feature Indexed Hypotheses," Proc. IEEE Conf. on Robotics and Automation, San Francisco, 1986, p. 925.

[42] J.C. Crowley, "Navigation for an Intelligent Mobile Robot," IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, 1985, p. 31.

[43] J.A. Simpson, R.J. Hocken, J.S. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing System, Vol. 1, No. 1, 1983, p. 17.

[44] R. Lumia, "Representing Solids for a Real-Time Robot Sensory System," Proc. Prolamat 1985, Paris, June 1985.

[45] M.O. Shneier, E.W. Kent, P. Mansbach, "Representing Workspace and Model Knowledge for a Robot with Mobile Sensors," Proc. 7th Int. Conf. Pattern Recognition, 1984, p. 199.

[46] M.O. Shneier, R. Lumia, E.W. Kent, "Model Based Strategies for High-Level Robot Vision," CVGIP, Vol. 33, 1986, p. 293.

[47] C. McLean, H. Bloom, T. Hopp, "The Virtual Manufacturing Cell," IFAC/IFIP Conf. on Information Control Problems in Manufacturing Technology, Geithersburg, MD., Oct., 1982.

[48] A. Barr, E. Feigenbaum, The Handbook of Artificial Intelligence, (Los Altos, William Kaufman, 1981).

[49] C. McLean, M. Mitchell, E. Barkmeyer, "A Computer Architecture for Small Batch Manufacturing," IEEE Spectrum, May, 1983, p. 59.

[50] M. Mitchell, E. Barkmeyer, "Data Distribution in the NBS Automated Manufacturing Research Facility", Proc. of the IPAD2 Conf., Denver, April, 1984.

[51] C. McLean, "Process Planning Research at the AMRF," Navy Manufacturing Technology Report, Nov., 1984, p.5.

[52] C. McLean, "An Architecture for Intelligent Manufacturing Control," 1985 ASME Computers in Engineering Conf., Boston, Aug., 1985.

53] J. Albus, C. McLean, A. Barbera, M. Fitzgerald, " An Architecture for Real-Time Sensory-Interactive Control of Robots in a Manufacturing Environment," 4th IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology, Gaithersburg, Oct., 1982.

[54] T. Hong and M. Shneier, "Describing a Robot's Workspace Using a Sequence of Views from a Moving Camera," IEEE Trans. PAMI, vol PAMI-7, no. 6, 1985, p. 721.

[55] P. Brown, "The Interactive Process Planning System," 1986 Winter ASME Conf., Anaheim, Dec., 1986 (submitted).

NIST