NEW NBS PUB
JUL 14 1988

NBSIR 88-3772

IMPLEMENTATION OF THE INSPECTION ROBOT CONTROLLER

April 21, 1988

By:
Howard T. Moncarz Bruce Borchardt

U.S. DEPARTMENT OF COMMERCE National Bureau of Standards Gaithersburg, Maryland

IMPLEMENTATION OF THE INSPECTION ROBOT CONTROLLER

Howard T. Moncarz
Bruce Borchardt

April 21, 1988

This publication was prepared by United States Government employees as part of their official duties and is, therefore, a work of the U.S. Government and not subject to copyright.

Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

1. WHAT THIS DOCUMENT IS ABOUT 1

2. AUDIENCE 1

3. OVERVIEW 1

II. TOP LEVEL DESCRIPTION OF THE ROBOT CONTROLLER 3

1. DESCRIPTION OF THE ROBOT 3

2. LOCATION IN THE WORKSTATION ARCHITECTURE 3

3. MAIN CONTROLLER FUNCTIONS 5

4. WORK ELEMENTS AND STATUSES 5

III. DEFINING POINTS FOR THE ROBOT 7

1. REFERENCE FRAMES 7

2. POINTS ASSOCIATED WITH REFERENCE FRAMES 7

3. TYPES OF LOGICAL POINTS 7

4. PATH TRAVERSAL 8

5. PHYSICAL POINTS 9

IV. DATA STRUCTURES 11

1. AMRF DATA 11

2. LOCAL DATA 11

V. TASK DECOMPOSITION 13

1. wsc_irc 13

2. task 13

3. e_move 15

4. machine 16

Page

5. am_robot 18
5.1. Handshake (lines 1-18) 18
5.2. Initialization (lines 19-31) 18
5.3. State Table Parser (lines 32-82) 18
5.4. Execution Of State Table (lines 83-124) 18
5.5. Utility Subroutines (lines 124-169) 18
5.6. Command Subroutines (line 170-368) 19
5.7. Read Serial Port (lines 369-383) 19

VI. PROCEDURE MODULES 21

1. irc_lib 21

2. irc_types 21

3. irc_funcs 21

4. rpt_funcs 21

5. get_data 21

VII. INTERFACE TO EQUIPMENT 23

1. MODULES THAT INTERFACE TO EQUIPMENT 23

2. DETAILS OF THE CURRENT IMPLEMENTATION 23

3. CHANGES REQUIRED FOR EQUIPMENT SUBSTITUTION 23

VIII. INITIALIZATION AND SHUT DOWN 25

1. START UP 25

2. SHUT DOWN 25

IX. ERROR HANDLING 27

1. CONTROLLER PROGRAM HANGS 27

2. MOVE ERROR 27

3. EMERGENCY STOPS 28

4. ERRORS UNDER MANUAL CONTROL 28

X. USER INTERFACE 31

1. EXTRA MODULES REQUIRED FOR TESTING 31

2. USER COMMANDS 31

Page

XI. FUTURE PLANS 33

1. SOFTWARE DEVELOPMENT 33

2. NEW HARDWARE 33

3. PROBLEM AREAS 33
3.1. Locating the Part 33
3.2. Move Errors 33
3.3. Starting the Robot 34
3.4. Time Delays 34
3.5. Flipping the Part 35

APPENDICES

A. IWS DOCUMENTATION LIST 37

B. REFERENCES 39

C. GLOSSARY (and abbreviations) 41

D. FLAT FILE SPECIFICATIONS 47

READER COMMENT FORM 53

LIST OF FIGURES

Page

Figure 1. Logical Configuration of the IWS..... 4

Figure 2. Task Decomposition for the Robot Controller..... 14

IMPLEMENTATION OF THE INSPECTION ROBOT CONTROLLER

I. INTRODUCTION

1. WHAT THIS DOCUMENT IS ABOUT

This document describes the implementation specifics of the Inspection Robot Controller (IRC) program. This program runs under the control of the ECS program that is described in the document IMPLEMENTATION OF THE EXECUTION CONTROL SYSTEM OF THE INSPECTION WORKSTATION. The controller program consists of state machine modules that "customize" the controller for its particular application -- i.e. supervising the Robot.

2. AUDIENCE

Anyone who needs to understand the internals of the IRC software should read this document. This includes anyone who will continue the development of the IRC or make modifications to it.

The document ARCHITECTURE AND PRINCIPLES OF THE INSPECTION WORKSTATION describes the principles that the ECS and the IRC programs utilize. It is recommended that that document be read first.

3. OVERVIEW

Chapter II gives a top level description of the Robot Controller. It describes the equipment, specifies the location of the IRC in the IWS control hierarchy, and describes the main functions the controller performs.

Chapter III explains how locations are specified to IRC and how dynamic path generation is accomplished during controller execution.

Next, Chapter IV describes the main data structures, both global to the AMRF as well as local to the IWS, that the controller program uses. The specific task decomposition that the IRC incorporates is explained next in Chapter V. Additionally, procedure modules used by the main tasks discussed in Chapter V are described in Chapter VI.

The actual interface to the Robot is specified in Chapter VII. Specific details used in the start up and shut down procedures are described in Chapter VIII. Errors that can occur during operation are listed and explained in Chapter IX. Chapter X desribes the user interface to the IRC.

Finally, Chapter XI discusses future development plans for the IRC.

The appendices include further information and implementation details. Appendix A lists the entire IWS documentation set. References specific to the Robot and IRC are listed in Appendix B. Appendix C contains a glossary of terms used in this document. Appendix D specifies the internal file formats used to contain all the data for the IRC.

Completing the document is a reader/comment form. You are encouraged to write down your comments and mail the attached form to the address specified.

II. TOP LEVEL DESCRIPTION OF THE ROBOT CONTROLLER

The two primary duties of the Inspection Robot Controller (IRC) are to control the robot and to supervise the Surface Roughness Instrument (SRI) Controller.

The design of the Horizontal Workstation's robot controller (RCS) was studied and used as a starting point for the IRC. In particular, the RCS task decomposition strategy was adapted for the IRC, as well as the definition of the task level commands. (Please note that the task level commands refer to the commands specified in the task module.)

Several commands were added to the IRC's list of task commands to customize the IRC to its special requirements. However, those commands were defined so that they could be considered general robot commands, rather than specialized commands used only in an inspection workstation.

The task decomposition is discussed in Chapter V. The task level commands are discussed in Section 2 of that chapter.

1. DESCRIPTION OF THE ROBOT

The robot used at the IWS was manufactured by American Robot, now named American Cimflex. The IWS Robot is used to transfer parts from one place in the workstation to another, and to work as a unit with the SRI to position a part in front of it as required.

The IWS Robot is a standard robot that is custom mounted upside down on a gantry to permit it to reach any place in the IWS that is required. The robot system consists of the robot itself, the robot computer, the teach pendant, and the monitor. The robot has two modes of movement: position and track. In position mode, the robot can be moved at six joints -- the waist, shoulder, elbow, wrist roll, wrist flex, and hand roll. In track mode, the robot can be moved along the gantry track. The robot cannot move in both modes simultaneously.

The robot has its own computer built into it that is considered part of the robot itself. This computer runs a program written in the robot's language (AR Smart) that interfaces it to the HP computer that acts as its controller.

2. LOCATION IN THE WORKSTATION ARCHITECTURE

As shown in Figure 1, the Robot controller is subordinate to the Workstation Controller and is the supervisor to the Robot.

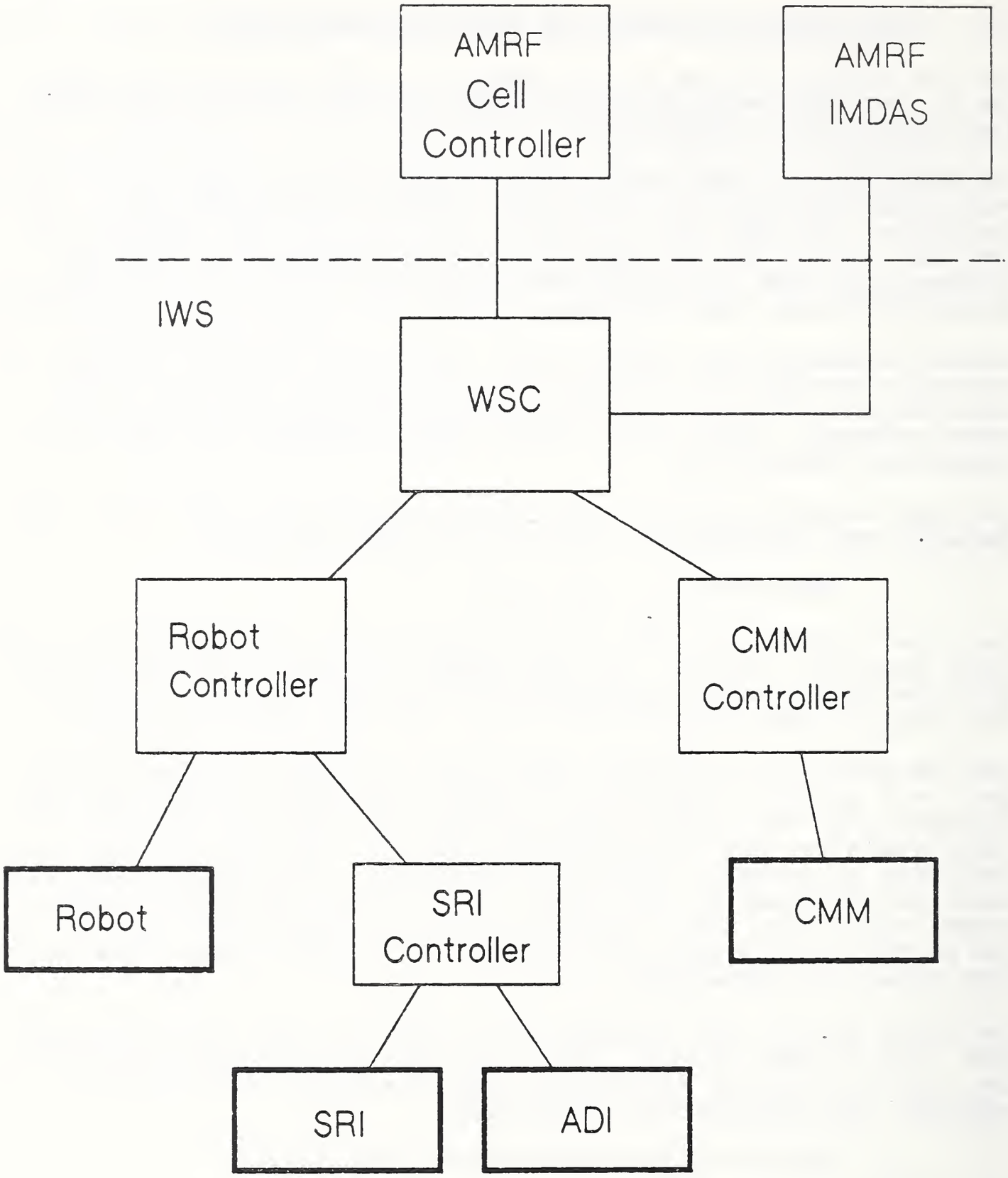

Figure 1. Logical Configuration of the IWS

Additionally, the Robot Controller is the supervisor to the SRI Controller. In a future implementation, the Robot Controller will be able to access the IMDAS (Integrated Manufacturing Data Administration System), the distributed data system which provides common interfaces to the AMRF's user programs and underlying databases [B.5, B.6].

3. MAIN CONTROLLER FUNCTIONS

The main functions of the Robot controller are the following:

Respond to Workstation Controller (WSC) commands and return status information.

Retrieve data mapping out the IWS from the IMDAS.

Control the robot's motion.

4. WORK ELEMENTS AND STATUSES

The Robot Controller receives commands from the Workstation Controller. These commands are either transition commands (involved in the start up/shut down protocol), or work order commands (for operating in ready state), which contain the work elements that command the main functions that the Robot Controller is responsible for doing.

The work elements executed by the Robot Controller are listed below:

INITIALIZE

LOAD_DATA

ACQUIRE

CLEAR

MOVE

RELEASE

TRANSFER

GET_DIAL_READING

LOAD_SRI_DATA

INSPECT_WITH_SRI

Details for these work elements are discussed in Chapter V, Section 2.

Statuses returned by the Robot Controller to the Workstation Controller are:

WORKING

DONE

ERROR

III. DEFINING POINTS FOR THE ROBOT

IRC commands refer to locations in the IWS. To generalize the IRC, so it can be interfaced to other robots, locations are specified by logical points, and are read in as data. A logical point consists of a reference frame and a point number. This is discussed in Sections 1 and 2.

Path traversal between any two logical points is specified in Sections 3 and 4.

Ultimately, a logical point refers to the actual location in the IWS, known as the physical point. Physical points are discussed in Section 5.

1. REFERENCE FRAMES

An IRC reference frame is used to group a collection of points together by their physical proximity. A convenient choice is to associate a reference frame with each piece of equipment at the IWS.

The particular reference frames used are specified by data, and are read in as character strings. For the benchmark IWS, the reference frames used are: 'TRAY', 'CMM', 'SRI', 'DIAL', and 'ROBOT'.

Currently there is only one tray at the IWS. If a second tray is added, the reference frames for the two trays could be specified as 'TRAY1' and 'TRAY2' (or whatever names are desired). The 'DIAL' reference frame refers to points around the dial indicator. 'ROBOT' refers to the reference frame assumed when the robot is first started up -- which could be considered the absolute reference frame for the IWS.

2. POINTS ASSOCIATED WITH REFERENCE FRAMES

The points associated with a reference frame are identified by numbers -- using a different number for each point. They do not have to be in consecutive order. Also, the same numbers can be used to specify points in different reference frames.

3. TYPES OF LOGICAL POINTS

Logical points are grouped into three separate categories -- destination points, individual approach points, and group approach points.

Destination points include all those points that the robot could end up at the completion of a command. These points could include

positions on the tray, positions on the coordinate measuring machine, positions in front of the surface roughness instrument, etc.

Every destination point has an individual approach point associated with it. The robot must always go through the individual approach point to go to the destination point. Conversely, the robot must always pass through the individual approach point after coming from the destination point.

It is possible for the robot to complete its movement at an individual approach point. In that case the approach point must still have an individual approach point. However, an approach point may have itself defined as its approach point.

Finally, every reference frame has one point associated with it that is called the group approach point. Whenever the robot leaves or arrives at a reference frame, it must pass through that reference frame's group approach point.

At the completion of a robot move command, the robot may be ordered to go to the safe position for its current destination point. The destination point's individual approach point is used for that safe position.

4. PATH TRAVERSAL

A predetermined strategy is used in going from one destination point to the next. This strategy is implemented in the module e_move.

In general, the robot will move from its current point to that point's individual approach point. From there it will go to the group approach point for its current reference frame. Next, it will move to the group approach point of the destination's reference frame. From that point, it will move to the destination's individual approach point, and finally to the destination point itself.

It is assumed that the robot can always move from a destination point to that point's individual approach point and vice versa, directly. Between any other two points, a path may be defined. If a path is defined, the robot is directed to move to intermediate points between the end points of the path. If a path is not defined, then it is assumed that the robot can move safely between those two points.

If a reference frame is specified as 'safe' (by data), then the robot can move from any approach point within that reference frame to any other approach point within that reference frame directly,

without checking to see if a path has been defined. If the destination point is in the same reference frame as the current point, then the group approach point can be skipped in the path traversal. I.e. the entire path would be from the current point to its individual approach point to the destination's approach point to the destination point.

The robot is always commanded to move at its approach speed between a destination point and its approach point -- either coming or going. Between any other two points, the robot is directed to move at its faster free space speed. Also, the robot is directed to move in its 'continuous mode' to all points except a destination point. In 'continuous mode', the robot does not pause at all in moving to the point following the immediate point it is approaching. In going to a destination point, the robot is directed to move in its 'stop at point mode.' In this mode, the robot comes to a complete stop before it moves to the next point.

5. PHYSICAL POINTS

Eventually, all logical points refer to physical points -- actual locations in the IWS. These points are all taught ahead of time, but are loaded in as data when the IRC receives the 'LOAD_DATA' command. The paths taken in going from one point to another are dynamic, within the limitations described in the previous section.

By simply redefining the taught points, small adjustments can be made to the actual locations throughout the IWS, without affecting the logical points or paths at all.

Additionally, the IRC has the capability to specify any absolute point in the IWS work space (either a taught point or an arbitrarily specified or computed point) as a local reference frame for moves relative to that point. (Note that a point includes position as well as orientation components for its specification. The IRC uses this capability when it adjusts the position of the part in front of the SRI, based on feedback from the SRI Controller. In this case, the points moved to by the robot are dynamically determined.

IV. DATA STRUCTURES

1. AMRF DATA

Before the robot can be directed to its main tasks (moving material around the IWS), the Robot Controller must retrieve the data that maps out the workstation. This data is specified in Appendix D.

The data is retrieved after IRC receives the command to 'LOAD_DATA'. This command is passed down to the task module, and this module directs the data retrieval.

The 'LOAD_DATA' command is packaged with two arguments. The first argument specifies which set of files contains the data required, and the second argument specifies which mapping within that data should be used. This second argument is used, because one set of data may contain several mappings. Each mapping is referred to as a robot plan.

In the current implementation, the data is stored locally and is not retrieved from the IMDAS. The task module prepares the local data structures so that the data files for the robot plan required will be used by the Robot Controller. When the data is retrieved from the IMDAS, each set of data will contain one robot plan, and consequently the one argument to 'LOAD_DATA' will be the name of the robot plan.

2. LOCAL DATA

The local data is stored in a flat file system -- each relation is stored in a separate file. A relation contains key fields that are used to find the record required. Then, the data fields in that record are retrieved. The full specification for all the IRC flat files is in Appendix D. The description of how the flat file system is implemented is in IWS document Implementation of the Execution Control System of the Inspection Workstation.

V. TASK DECOMPOSITION

The state machine modules used to implement the Robot Controller are shown in Figure 2. These are described below.

1. wsc_irc

The wsc_irc module implements the UVA Protocol [B.2] in interfacing the Robot Controller to the Workstation Controller. This module accepts commands from the WSC and passes those commands down the task decomposition hierarchy. It receives statuses from its subordinate, the task module, and returns statuses to the WSC.

2. task

The IRC task module supervises the main functions performed by the Robot Controller. These are the work elements listed in Chapter II, Section 4. Additionally, task takes care of starting itself up and shutting itself down upon command from its superior (wsc_irc), as all state machine modules do in the IWS implementation.

Currently, task retrieves data from data files residing locally on the Robot Controller. Eventually, this data will be retrieved from the IMDAS. This data will then be parsed and stored in local data files in the same format as the data that is currently being used.

The implementation of the work elements is described below. Some of the work elements require arguments, and these are enclosed in parentheses. Optional arguments are enclosed in brackets. If the part name is not given, it is assumed that the robot is already holding the proper part. The part name is abbreviated as OBJ. If the location is not given, it is assumed that the location is the safe point for the previous location referenced. The locations are abbreviated as A, B, and C.

INITIALIZE
Initialize the robot and robot parameters.

LOAD_DATA (PLAN_NAME)
Load the PLAN_NAME robot data.

ACQUIRE (OBJ [at A])
Get the part (OBJ) at location A. If A is not specified, get OBJ from the current location.

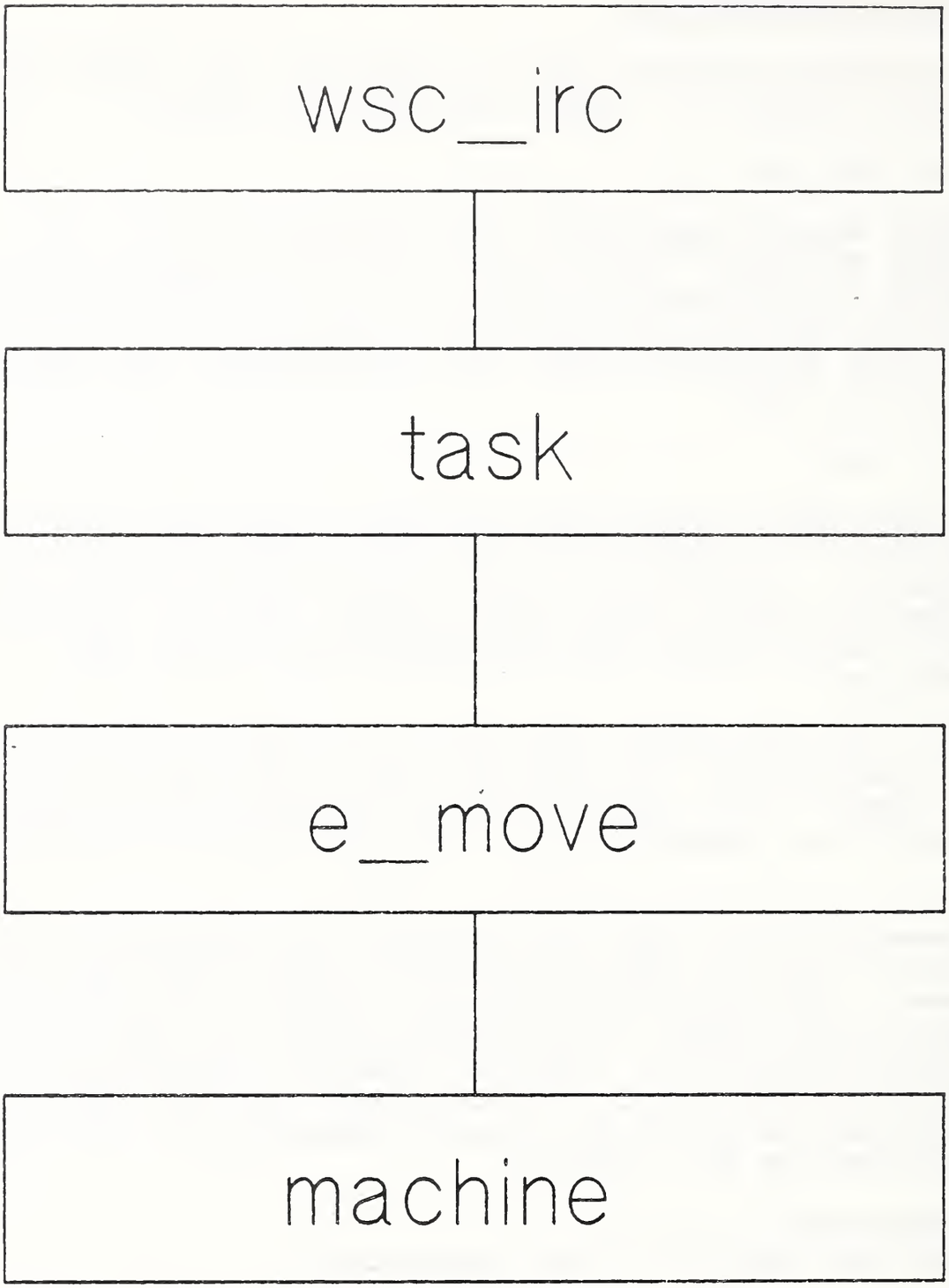

Figure 2. Task Decomposition for the Robot Controller

CLEAR (drop-at A [end-at B])
Drop off OBJ at A. If B is specified, end up at B, otherwise end up at the safe point for A (A's individual approach point).

MOVE ([OBJ from A] to B)
If A is specified, get OBJ from A. Then move to B.

RELEASE [end-at A]
If A is specified, then go to A. Release OBJ.

TRANSFER ([OBJ from A] to B [end-at C])
If A is specified, get OBJ from A. Go to B. Then, if C is specified, go to B, otherwise go to the safe point for B (B's individual approach point).

GET_DIAL_READING ([OBJ from A] to B, towards C)
If A is specified, get OBJ from A. Go to B. Then go in the direction from B towards C and move a distance equal to the peck length. Repeat this move until reading is obtained from the ADI. Total dial reading is equal to the (number of pecks - 1) times the peck length plus the dial reading.

LOAD_SRI_DATA (OBJ, PLAN_NAME)
Tell the SRI Controller to load SRI data for OBJ, using data from PLAN_NAME.

INSPECT_WITH_SRI ([OBJ at A] to B, towards C)
If A is specified, get OBJ from A. Go to B. Then go from B towards C and up to a distance of SRI_Clearance from the SRI (using the reading obtained from GET_DIAL_READING). Now, send command to the SRI Controller (SRIC) to begin the SRI inspection. At this point, move to the positions specified by the SRIC (track) until the SRIC reports back DONE. This completes the work element.

3. e_move

The term e_move stands for elementary move. This module contains procedures that are required for each move segment (see Chapter III, Section 4). The main e_move procedures that are currently implemented are: xqt_pickup, xqt_move_to, xqt_move_to_obj, xqt_release, xqt_load_robot_data, and xqt_init_mach. These are described below.

Xqt_pickup closes the gripper to grasp a part.

Xqt_move_to moves the robot to the point specified by its argument. The argument specifies the logical point in terms of its reference frame and its point number. This command uses the path traversal strategy described in Chapter III, Section 4, given

the current location of the robot and destination point specified by the argument.

Xqt_move_to_obj first opens the gripper to prepare it to grasp a specified part, then executes the move specified above.
Xqt_release opens the gripper to release a part.
Xqt_load_robot_data loads the robot data that specifies the layout of the IWS in terms of reference frames used, point locations, important dimensions, etc., as well as robot parameters such as approach speed, free space speed, etc.

Finally, xqt_init_mach establishes synchonization between the robot and the HP computer, and establishes default values for certain robot parameters.

4. machine

The IRC machine module provides the interface between the bottom level IRC commands, that are still robot independent, to the American Robot -- a specific robot.

The main commands implemented at the machine module (that are important for the Robot Controller) are: INIT_HP_TO_ROB, INIT_HANDSHAKE, INIT_MACHINE, IDLE, MOVE_TO_POINT, SET_SPEED, SET_MOVE_MODE, CHANGE_GRIP_POSITION, SET_RF, and SET_TOOL. These commands are described below.

When the Robot Controller is first started, certain initializations are required. The INIT_HP_TO_ROB command initializes the interface boards on the HP computer in preparation for communicating with the robot. The INIT_HANDSHAKE command then establishes synchonization with the robot. Next the INIT_MACHINE command sets default values for several robot parameters. After this command is executed, the robot is ready to receive the normal operating commands. These are described below.

The IDLE command is used to put the robot is its 'IDLE' state, from which it can receive new commands. This state is required after the completion of each command. In a future version of the software this state will be eliminated and replaced by a handshaking protocol.

The MOVE_TO_POINT command moves the robot to the point specified by its seven coordinates -- track and six coordinates for the arm. The robot moves to the track position first, and then to the arm position. (The track and arm moves are independent.) The first argument for this command (before the coordinates) specifies whether the move is absolute or relative. If absolute, the coordinates are measured from the robot power up reference frame.

Otherwise the coordinates are measured from the reference frame specified by the SET_RF command (described below).

The SET_SPEED command sets the speed of the hand. E_move changes the speed between the free space speed and the approach speed back and forth as required. The speed set in the machine module is an index -- the program on the robot (am_robot) determines what the speed actually is. If the speed specified by this command is the one currently being used, no command will be sent to the robot.

SET_MOVE_MODE has arguments to set several parameters for the robot which it uses while it is moving between points. The first argument specifies whether the move is to be joint-coordinated or straight-line mode. The next argument specifies whether the move is to be continuous, or if the robot will stop at the point to which it is moving. If it is continuous, the robot will go at full speed to that point and immediately proceed to the next point. Otherwise, if the 'stop at point' mode is used, the robot will come to a complete stop at the specified point before it will accept another command. The third argument specifies whether the elbow is up or down during arm moves. Whenever the SET_MOVE_MODE command is issued, the machine module determines what the current settings of the three modes affected are, and only sends commands to the robot to set those modes that must be changed.

The CHANGE_GRIP_POSITION command is used to open or close the gripper. It has one argument which specifies the gripper position -- either opened or closed. This command is only sent to the robot if the position specified is different from the current gripper position.

The first argument of the SET_RF command determines whether the robot reference frame is absolute (the reference frame on start up) or if a local reference frame is to be used. In the latter case, the next six arguments specify a point (for the arm only) that will be used as local reference frame from here on (until the local reference frame is cancelled by a new SET_RF command which specifies the absolute frame). For a clear understanding of this command and the SET_TOOL command (to be described below), consult the American Robot manual.

The arguments for the SET_TOOL command are similar to the SET_RF command. The first argument specifies whether to cancel the tool or else to define a new tool. If a new tool is specified, the next six arguments give the point location (of the arm) to use to define the new tool. The SET_RF command sets a reference frame that is used to specify positions for the center of the hand. The SET_TOOL command provides an offset to that position.

This is useful when the robot is holding a part in front of the SRI. If a SET_TOOL command is used to define the offset of the part surface to the robot's hand, and then that same point on the part (which is in front of the SRI) is defined as the local reference frame to use, a move may be specified (i.e. a pure roll motion) about that point.

5. am_robot

Am_robot is a program, written in the AR Smart language, that runs on the robot's computer, and interprets and executes commands issued by the IRC. The program is implemented as a state table interpreter. The following describes the program, line by line.

5.1. Handshake (lines 1-18)

This section checks the HP GPIO port (on the controller) to be sure that communications are working. It raises its line, then checks the line from the controller. After waiting one second, it checks again, and if the controller has lowered its line in that second, continues. This assures that both sides are "live."

5.2. Initialization (lines 19-31)

This fills up the registers with appropriate values (mostly zero). This section is just insurance; the program would probably work fine without it, as the registers are normally filled in the program.

5.3. State Table Parser (lines 32-82)

This section decides which line of the state table we're on, based on the command (register 8) and the current status (register 20). The state table line number is placed in register 21; 50 is added to this number; and the appropriate auxiliary commands are executed by subroutines 51-57.

5.4. Execution Of State Table (lines 83-124)

Here is where the state table housekeeping is done. The various subroutines (120 to 196) do some necessary chores (filling registers, output and input lines, etc.). The execution of subroutine 54 is where the actual robot commands are carried out.

5.5. Utility Subroutines (lines 124-169)

Here are the housekeeping subroutines.

5.6. Command Subroutines (lines 170-368)

This section executes specific commands, numbered 1 through 8 (commands 9-15 are null; they could be written later). Each is treated as a subroutine. These subroutines work entirely through robot commands, except for calls to the serial port for data.

i. Move Track. Notes the current position, gets 1 number from the serial port, loads it into register 1, decides whether the move is long enough to require the three second wait, branches to the move with wait or move without wait routine, defines a point from register 1, and moves to it.

ii. Move Arm. Gets 6 numbers from the serial port, loads them into the registers, defines a point from them, and moves to it.

iii. Move Track and Arm. Gets 7 numbers from the serial port, loads them into registers, defines a track point from register 1, moves to it, puts the x-coordinate back into register 1, defines an arm point from the 6, and moves to it.

iv. Set Speed. Reads one number from the serial port, and sets the arm speed and track speed to a number close to it (0.5 to 40). This round off is necessary because of the limited choices accessible to the Set Speed command.

v. Set Move mode. Sets the robot to the move mode passed as the command argument.

vi. Open/close gripper. Looks at the command argument (0 or 1) and opens the gripper for a 0 or closes it for a 1.

vii. Set Frame. Gets 6 numbers, defines a point, and sets frame to it.

viii. Set Tool. Gets 6 numbers, defines a tool, and sets it.

5.7. Read Serial Port (lines 369-383)

This gets one number from the serial port and puts it into the accumulator. It includes handshaking to be sure that the control computer sends only one number.

VI. PROCEDURE MODULES

The modules in the task decomposition (state machine modules plus the machine module) use procedures that are packaged into separate modules. These modules are described in this section.

1. irc_lib

This library module contains procedures from the HP library that are not used by the other controllers. The particular modules included here are used to access the Hewlett Packard Interface boards.

2. irc_types

This module includes data structure types that are specific to the Robot Controller, and are referenced by state machine modules as well as procedure modules throughout the Robot Controller program.

3. irc_funcs

This module contains some general functions that are specific to the Robot Controller and are referenced by modules throughout the Robot Controller program.

4. rpt_funcs

This module exports the data structures and functions to allow the Robot Controller to track the orientations requested by the SRI controller.

5. get_data

This module contains the procedures that search the locally stored data files to return the data values needed by the state machine modules as they are running.

VII. INTERFACE TO EQUIPMENT

1. MODULES THAT INTERFACE TO EQUIPMENT

The machine module interfaces the HP computer to the robot, and is the only module to do so.

2. DETAILS OF THE CURRENT IMPLEMENTATION

Details for the machine module are described in Chapter V, Section 4 above.

3. CHANGES REQUIRED FOR EQUIPMENT SUBSTITUTION

To substitute a robot from a different manufacturer, the machine module must be modified to implement the functions specified in Chapter V, Section 4 for that robot.

VIII. INITIALIZATION AND SHUT DOWN

The top level module of the Robot Controller, wsc_irc, implements the UVA initialization and shut down protocol. This protocol implementation is discussed in Chapter VI on the Workstation Controller implementation.

This section discusses what is done in the Robot Controller when it receives these transition commands. It ignores the interface between wsc_irc and the Workstation Controller, which is discussed in Chapter VI.

1. START UP

On cold start up, i.e. when the Robot Controller program is first started, the hardware interface to the robot is initialized, and then a handshake is done to establish synchronization between the controller and the robot. Additionally, common memory mailboxes are established.

After these events occur, IRC is ready to receive commands. When the 'WARM_STARTUP' command is received, task issues a 'WARM_STARTUP' command to the SRI Controller, and waits until that controller acknowledges that it is started. The task module then goes into its 'IDLE' state, reports 'DONE' back to wsc_irc, and is then ready to receive order action commands. (A LOAD_DATA command is required before any other order actions can be issued.)

2. SHUT DOWN

On receiving the 'WARM_SHUTDOWN' command, the command is passed on to the SRI Controller, and task goes to the 'SHUTDOWN' state. When the SRI Controller signals 'DONE' back, task reports 'DONE' back to wsc_irc.

IX. ERROR HANDLING

There are only a few error conditions that can be handled without restarting the system. The most common diagnostic that appears on the robot's screen (not the IRC) is "extra status." This is a non-error which requires no action.

1. CONTROLLER PROGRAM HANGS

This error has two main sources. First of all, when the Controller is communicating with the robot, if the handshaking between the two does not conform to the proper protocol, the Controller program can hang. It will continually wait for a certain response from the robot that will never occur. One cause for this failure is a noisy interface between the Controller and the robot.

Secondly, any bug in the control structure of the program can force the Controller to go into an undetermined domain and effectively crash the program.

Currently, the Robot Controller cannot recover from this error. The IWS must be restarted.

2. MOVE ERROR

During operation, there should in principle be no move errors. Move errors occur when the robot is commanded to a position that it calculates to be impossible, for instance one that is outside its reach. They can also occur for other reasons, which should not happen while executing our usual program. One that happens is "failure to reach end position." This indicates that the position was not reached, but fails to show why not. Usually, a MOV command will succeed in moving the robot to the position that the previous command failed to reach. Other move errors can also occur, but the reasons are not clear. In a perfect system, since we are moving only to points well-known to be legal, they should never occur under program execution.

The only response to the move errors is to restart the robot program at a point which does not require an initial handshake. This is the procedure given in the operations manual. The philosophy behind it is that the program can be made to begin execution at a particular line, but only from the PGM (program) mode. The technique is to issue, from immediate mode, 'GTS 19'. (go to statement 19. The decimal point is required.) This is a line which is after the initial handshake, so the program will immediately start executing its state table. Next, issue PGM to put the robot in the program mode, and issue RUN from that mode. This is by no means an absolutely certain restart. If the robot

controller is not in a suitable state, or if the robot's registers have unusual data in them, the restart may well not work. When you try this, be sure to have your finger on the stop button.

3. EMERGENCY STOPS

The system has several ways to stop it. None of them could be considered "soft" stops, in that all require a restart of at least the robot program.

The first thing to try might be the rocker switch 'Run/Stop' on the teach pendant. This will stop the robot program, and prevent the next move from occurring. Use this in a non-emergency situation, when there is plenty of time for the present move to finish. This stop can be recovered from with the GTS 19., PGM, RUN sequence.

Equal in ease of recovery are the next level of stops. This level includes the MOTOR button on the teach pendant and the fence circuit. Both of these shut off the robot motors immediately. This also stops the executing program automatically. To recover from these stops, it is necessary to restart the motors, and then do the program restart. The fence circuit has a locking feature which is controlled by the box on the northeast column of the gantry. If the fence is triggered, it must be reset with the black button before the motors will restart. The audio feature of the fence alarm can be turned on or off independently of the fence itself.

The next level up is the power off button on the front console. Pushing this button also stops the robot, but it necessitates the reloading of AR Smart, the recalibration of the robot arm and gantry, the reloading of the program, and the restart of the control program. It also makes the success of the GTS 19 procedure very uncertain, as none of the robot registers will be properly loaded, except by coincidence.

4. ERRORS UNDER MANUAL CONTROL

There are lots of ways to generate errors while under manual control. Besides soft move errors, where the requested move is known to be impossible, there are hard move errors from the end of travel of an axis, from the collision detector, and from all sorts of moves. The recovery is similar to that under program control, except that once the motors are live, you are already under manual control. The only additional tip concerns axis end of travel:

sometimes the axis will be stuck at the end of travel (this seems to happen especially with the elbow axis.) In this condition the joystick controls will sometimes not work; the solution is to turn off the motors and manhandle the arm away from the position, then turn the motors on again.

X. USER INTERFACE

1. EXTRA MODULES REQUIRED FOR TESTING

In integrated mode, wsc_irc is the highest level module in the Robot Controller. It receives commands, via the local network, from the Workstation Controller. If the Robot is to be run in stand-alone mode, the operator needs to enter commands to the controller and have those commands transferred to the wsc_irc module. The interface that provides this connection between the user and the wsc_irc module is contained in three modules -- irctest, user_io, and xqt_file.

Irctest is a state machine module, analogous to cmmtest in the CMM Controller program. The actual commands available are contained in the procedure module user_io, and are exported to irctest. Likewise, the procedures in xqt_file are also exported to irctest. These procedures allow the user to execute a file of commands in one batch.

2. USER COMMANDS

The first level of commands available to the user are 'ABORT', 'SHUTDOWN', 'STARTUP', and 'EXECUTE'. These are the transition commands. The first three commands do not have arguments. The arguments for 'EXECUTE' are the work orders and their respective arguments.

The user must first choose 'STARTUP' to transition the Robot Controller to the ready state. Next, the user selects 'EXECUTE' and is presented with the work order commands. The user may continually request work order commands until he chooses the command 'QUIT', which returns the program to the first level of commands.

XI. FUTURE PLANS

1. SOFTWARE DEVELOPMENT

The primary effort is involved with fully integrating IRC into the AMRF. This entails software development so that IRC will retrieve the data it requires from the IMDAS, and then translate that data into the internal data structures required.

The next area of software development will be to allow the robot to pick up a part at an offset and orientation from the destination point where it is located. That transformation from the destination point is referred to as a grip point for the part, and each part will have one or more grip points assigned to it as initial data. Currently, the grip point is assumed to be aligned with the destination point. All destination points are at fixed locations in the workstation, and the robot cannot compensate if the part's grip point is not exactly where it is expected.

2. NEW HARDWARE

It is planned to add a vision system to the IWS that will aid the robot in picking up the part. The vision system will tell IRC the relative position and orientation of the part from its destination point, and with the software development described above, it will be a simple matter for the robot to pick up the part at the proper grip point for it.

3. PROBLEM AREAS

3.1. Locating the Part

The main problem in integrating the IWS to the AMRF and making it fully automatic is to be able to determine a part's exact location and orientation on the tray, and to pick it up at its proper grip point. When the AGV delivers a tray to the IWS, it is very likely (based on current experience) for a part to be considerably offset from its proper location. This leads to problems. The robot may not be able to pick up the part, or even if it can, the part may be put so out of place on the CMM that the CMM cannot inspect it. The same is true for the SRI inspection. The planned development described in the two sections above should solve this problem.

3.2. Move Errors

To make the IRC truly data driven, all destination points should be determined offline, and not need to be taught ahead of time by the operator. With the current hardware, a move from one legal point to another (both taught ahead of time) may result in a move error. (This is indicated on the robot teach pendant.)

Unfortunately, this stops the robot program (am_robot) and can crash the IWS. Even if the IWS can be recovered without rebooting the system, the recovery involves an awkward procedure. To be sure a move error doesn't occur during workstation operation, all paths must be tested ahead of time. Obviously, this limits the level of automation that can be achieved with this robot.

It should be noted before leaving this section, that there are two other distressing problems concerning robot moves. First of all, some moves that were tested offline, and in fact worked properly over and over again, sometimes suddenly don't work, again due to a move error. (This is despite following the manufacturer's instructions.)

The second problem was discovered in moving between two points that are very close to each other. Certain moves that are legal and consistenly work take so long for the robot to accomplish that they should be avoided. This move can be dramatically sped up by moving to an appropriate intermediate point. The problem moves, as well as the solution paths, can only be determined by experiment insofar as we have been able to ascertain.

3.3. Starting the Robot

Every time the robot is turned on, it must be calibrated. This is an awkward procedure relative to other IWS start up procedures, and an automatic calibration on power up would be very worthwhile.

3.4. Time Delays

The robot does not move at speeds necessary for commercial implementation, and there are time delays that seem excessive. For proof of concept, the slow speed of the robot is not a problem. The control structure could be implemented with sufficient speed. The delays are caused for the following reasons:

The control structure was not optimized for speed. Significant improvements are obtainable.

The retrieval from the local database is slow. A better implementation can solve this.

The current network implementation is very slow. Its upgrade will be much faster.

Communications between IRC and the robot's computer is slow. This is limited by the current robot computer.

Motion of the robot along its track is independent of the robot's arm motion. The robot does not signal IRC when a track move is completed, and an artificial time delay is necessary in the IRC program to account for this. Without the built in delay, the arm could start moving while the robot was moving along its track, and this could be disastrous.

3.5. Flipping the Part

Currently, there is no provision for flipping over a part. This is necessary to allow the robot to present more surfaces to the SRI, as well as to give more flexibility in placing the part on the CMM. An automatic vise would be needed to erase this limitation, but this has not yet been planned.

APPENDICES

A. IWS DOCUMENTATION LIST

1. H. T. Moncarz, Architecture and Principles of the Inspection Workstation, to be published as an NBSIR.

2. H. T. Moncarz, Implementation of the Execution Control System of the Inspection Workstation, to be published as an NBSIR.

3. H. T. Moncarz and T. H. Hopp, Implementation of the CMM Controller, to be published as an NBSIR.

4. H. T. Moncarz and T. V. Vorburger, Implementation of the SRI Controller, to be published as an NBSIR.

5. H. T. Moncarz and B. Borchardt, Implementation of the Inspection Robot Controller, NBSIR 88-3772, April 21, 1988.

6. S. A. Osella, Implementation of the Workstation Controller, to be published as an NBSIR.

7. J. Zimmerman, Inventory of Equipment in the Inspection Workstation, to be published as an NBSIR.

8. H. T. Moncarz, S. A. Osella, B. Borchardt, and R. Veale, Operations Manual for the Inspection Workstation, NBSIR 88-3766, April 21, 1988.

9. J. Zimmerman, Recommended Technical Specifications for Procurement of Commercially Available Systems for the Inspection Workstation, to be published as an NBSIR.

B. REFERENCES

1. American Robot Corporation, Merlin Robot System Operators and User Guide, Revision 2.1, 15 February, 1984.

2. D. R. O'Halloran and P. F. Reynolds, Jr., "A Model for AMRF Initialization, Restart, Reconfiguration, and Shutdown", NBS/GCR 88/546, May 23, 1986.

3. Rodenstock Precision Optics, Inc., Operating Manual for the Optical Surface Finish Measuring System RM400, 1984.

4. A. J. Wavering and J. C. Fiala, The Real-Time Control System of the Horizontal Workstation Robot, NBSIR 88-3692, December 16, 1987.

5. D. Libes and E. Barkmeyer, "The integrated manufacturing data administration system (IMDAS)--an overview", International Journal of Computer Integrated Manufacturing, Vol. 1, No. 1, pp. 44-49.

6. C. Furlani, et al, "The Integrated Manufacturing Data Administration System (IMDAS)", to be published as an NBSIR, 1988.

C. GLOSSARY (and abbreviations)

ADI Abbreviation for the Automatic Dial Indicator.

AGV Abbreviation for Automatic Guided Vehicle.

approach speed
Speed the robot uses when traveling between the destination point and its individual approach point -- either to or from the destination point.

automatic dial indicator
Instrument used to measure the distance that a spring mounted stem is depressed.

automatic guided vehicle
Electric vehicle under control of the AMRF's MHS that conveys trays of parts to and from workstations.

common memory system
Manages communications between state machines.

CMM Abbreviation for the Coordinate Measuring Machine.

CMMC Abbreviation for the CMM Controller.

continuous mode
Robot move mode in which the robot does not stop at the point towards which it is traveling, but goes through that point immediately on to the next point to which it is directed. (The opposite mode is stop at point mode.)

controller
Supervises the operation of a mechanism, another controller, or both.

coordinate measuring machine
Machine used to measure the dimensions of a part.

data server
Software module that interfaces the controller it resides on to the data it requires.

destination point
Location within the IWS where the robot can stop after it finishes its command.

ECS Abbreviation for the execution control system.

elbow mode
: Robot move mode when the arm is moving -- the robot's elbow may be specified to be up or down.

execution control system
: Computer program that runs on each controller computer and implements the AMRF design principles. This program loads and executes those modules which determine which controller is actually being run.

free space speed
: Speed that the robot uses when traveling between approach points (whether individual or group)

FSM
: Abbreviation for finite state machine. Strictly speaking, the software control modules used in the IWS are state machines, not finite state machines. However, for convenience, the abbreviation FSM is kept.

gantry
: A track mounted above the IWS that supports the robot (mounted upside down), and allows the robot to glide along it. This track gives the robot one more dimension (along the track) than is normally provided with this robot.

group approach point
: Point within one of the IWS defined robot reference frames that the robot must pass through before going to a new robot reference frame.

individual approach point
: Point associated with a destination point. Whenever the robot goes to a destination point, it must first pass through the individual approach point, and vice versa when the robot leaves the destination point. Every destination point specified within the IWS has an individual approach point associated with it.

inspection workstation
: AMRF workstation that inspects parts for dimensional tolerance and surface finish.

integrated mode
: Mode of operation of the IWS in which the IWS is integrated to the AMRF -- receiving commands from the AMRF Cell, and obtaining data from the IMDAS.

IRC
: Abbreviation for the Inspection Robot Controller.

IWS
: Abbreviation for the Inspection Workstation.

joint coordinated mode
: A robot mode which allows the robot hand to move between two points in its most efficient manner. The alternative mode is straight line mode.

load file
: Data file that specifies what state machines ECS should load and execute.

logical architecture
: Specifies the direction of commands and statuses between controllers and between controllers and equipment.

logical point
: A point defined in the IWS that is specified by its reference frame name and its point number.

MHS
: Abbreviation for AMRF's Material Handling System.

network
: The connections (both hardware and software) that connect the IWS controllers together and to the rest of the AMRF. Local network refers to the former only.

network interface unit
: This device, connected to each controller in the IWS and to the AMRF, provides the network link (both hardware and software) to the IWS. (This is not currently implemented. The IWS network consists of direct RS232 links.)

NIU
: Abbreviation for network interface unit.

offline
: Mode in which the robot is controlled by an operator rather than the IRC. This mode is used when starting up the robot and calibrating it. It is also used to determine what points to use and to test the motion between those points.

online
: Mode in which the IRC is controlling the robot.

orientation mode
: Robot mode (under teach pendant control) in which the robot can move in yaw, pitch, and roll.

physical architecture
: Specifies the physical connections among the controllers and equipment.

physical point
: Point within the IWS that is specified by its seven components -- position x, y, and z; orientation yaw, pitch, and roll; and track.

position mode
: Robot mode (under teach pendant control) in which the robot can move in x, y, and z directions.

reference frame
: Coordinate system containing a collection of points defined in the IWS. Each specified reference frame should usually be associated with a piece of equipment, and it is appropriate to name it as such.

RF
: Abbreviation for one of the IWS defined robot reference frames.

safe point
: Point out of the way (safe) from a destination point. The safe point used is the destination point's individual approach point.

safety fence
: An electric eye that surrounds the robot at the IWS and detects if an object moves into the robot's workspace, and reacts according to its settings (either by audible alarm, blinking light, or shutting down the robot).

SRI
: Abbreviation for the Surface Roughness Instrument.

SRIC
: Abbreviation for the SRI Controller.

stand-alone mode
: Mode of operation of the IWS in which an operator commands the IWS, and the data used by the IWS is stored in local data files.

state machine
: Software control unit with outputs dependent on inputs to it plus its internal state. This is the building block for the IWS control software.

straight line mode
: A robot mode which forces the robot hand to move between two points in a straight line. The alternative mode is joint coordinated mode.

stop at point mode
: Robot move mode in which the robot stops at the point towards which it is traveling, before going on to its next point. (The opposite mode is continuous mode.)

surface roughness instrument
: Machine that measures the optical scattering off the surface of a part that can be correlated with its surface roughness.

teach pendant
: Hand held device with keyboard and display, used by an operator to communicate with the robot offline from IWS operation.

transition commands
: Commands used to transfer the IWS to a new state (specified by the UVA protocol).

UVA Protocol
: Model, proposed by research group from the University of Virginia and adopted by the AMRF, that specifies the start up and shut down sequence for the AMRF as a whole as well as every controller within the AMRF.

work order commands
: A command accepted by a controller when it is in ready state, and used to perform one of its main functions.

WSC
: Abbreviation for the Workstation Controller.

D. FLAT FILE SPECIFICATIONS

This appendix contains the specifications for the flat files used by the Robot Controller and contained in local disk files. For details concerning the implementation of the flat file system, see the IWS document IMPLEMENTATION OF THE EXECUTION CONTROL SYSTEM OF THE INSPECTION WORKSTATION. For a general description of what the flat files are used for, see Chapter IV of this document.

The format for specifying these files (referred to as relations) is described here. Each file is composed of ASCII characters that are broken up into records -- each record containing one or more fields. Records are separated by a carriage return and a line feed. Fields are separated by one or more spaces.

The description for each relation begins with the name of the relation. The name given here is the same as given in the computer program, except that in the computer program the name is prefixed by 'DS_'.

Next is given a brief description of what this relation is.

This is followed by the name of an example data file that is actually used. The data files referenced contain data to inspect the pipe clamp, which is one of the parts commonly manufactured at the AMRF.

Next is specified the task module from which this relation is retrieved.

This is followed by the names of the key fields. These fields are used to find the particular record in the relation that is required. The names of these fields often include the underscore character, so that a name will clearly specify a single field, even if it has more than one word. However, these names are not necessarily the same as they appear in the computer program. They are named as such to self document what they are.

Additionally, the data types for these fields are indicated by following the data field (or fields) by ' : ' and then the identification of the type. (I.e. many of the fields are of type integer.) Furthermore, comments are often included that elaborate on what the fields mean. These are distinguished by enclosing them between '{' and '}'.

Following the names for the key fields are the names for the data fields. The names are specified in the same manner as for the key fields. The remarks above concerning the data type information and commenting apply here as well.

Every piece of equipment in the IWS has a reference frame (RF) associated with it, and any point in the IWS can be identified by its RF and point number (PN) within that RF.

All RF's are measured relative to an absolute zero, which is the position of the robot after calibration. This point is assigned the designation ('ROBOT', 1), which means that it is in the 'ROBOT' reference frame, and its point number is 1.

1. PathData

Description:
Specifies the path (integer name for it and the number of points on it) to follow based on the current location and the destination location.

Example file name: path_tst

Retrieved from: e_move

Key fields:
Current_RF, Current_PN :(identifier, integer)
Destination_RF, Destination_PN :(identifier, integer)
{these refer to IWS_Points}

Data fields:
Path_Name, Path_Length : integer

2. IWS_Point

Description:
Specifies the seven components of the robot point based on the reference frame and point number. This relation includes all points in the IWS, including local destination points, approach points, path points, etc.

Example file name: pts_tst

Retrieved from: e_move

Keyfields:
RF_Name, Point_Number :(identifier, integer)

Datafields:
7 components of robot point in absolute coordinates : real
{track, x, y, z, yaw, pitch, roll}

3. PathPoint

Description:
Identifies the next point in the path, given the path name and the point number.

Example file name: ppts_tst

Retrieved from: e_move

Keyfields:
Path_Name, Path_Index : integer

Datafields:
Path_Point : (identifier, integer) {refers to IWS_Point}

4. DestinationPoint

Description:
Each Destination_Point has an Individual_Approach_Point associated with it that is used to travel to and from that Destination_Point. The Individual_Approach_Points are also the Safe_Points -- one for each Destination_Point. It is assumed that there is always a clear path between the Destination_Point and its Individual Approach_Point in either direction. This relation specifies which Individual_Approach_Point to use for a particular Destination_Point.

Example file name: dpt_tst

Retrieved from: e_move

Keyfields:
Destination_RF, Destination_PN :(identifier, integer)

Datafields:
Individual_Approach_Point_PN : integer
{It is assumed that the RF is the same as that for the Destination_Point.}

5. OneRowOf4x4

Description:
This relation specifies the 4x4 transformation for each RF from the absolute reference frame (based on the 'ROBOT' RF.) However, this relation is currently not used. All

points are given in absolute coordinates, and the relation of the relative RF's to the absolute RF is currently not necessary.

Example file name: xrow_tst

Retrieved from: (currently not used)

Keyfields:
RF_Name : identifier
row number of 4x4 transformation : integer

Datafields:
1 row (4 components) of 4x4 transformation : real

6. EquipmentPlan

Description:
Specifies relevant information for each RF identified in the IWS. In general, each RF is identified with a piece of equipment, and consequently the RF's are referred to by the equipment with which they are associated.

Example file name: epln_tst

Retrieved from: e_move

Keyfields:
EquipmentLayoutName : integer
{specifies a particular RF configuration}
EquipmentIndex : integer
{specifies a particular RF within the EquipmentLayout}

Datafields:
RF_Name : identifier {used to identify this RF to the IWS}
EquipmentType : identifier
{for future development, a particular EquipmentType may be handled in a specific generic way by the robot}
RF_ApproachPointName : integer {the group approach point}
ForbiddenVolumeFlag : (0, 1, -1) {designates whether this RF can currently be approached by the robot. This is not currently implemented.
0 if not defined for this RF_Name
1 if defined and currently active
-1 if defined and currently inactive }
SafePaths : boolean {TRUE if don't need a path to travel between two points, both within this RF; else FALSE}

7. RobotPlan

Description:
Specifies (directly or indirectly) all information required by the IWS robot, given the Robot_Plan_Name.

Example file name: rpln_tst

Retrieved from: e_move

Keyfields:
Robot_Plan_Name : integer

Datafields:

```
Equipment_Layout_Name : integer
          {used as a key in Equipment_Plan relation}
NumOf_RFs : integer {number of reference frames -- each is
                     represented in Equipment_Plan relation}
IorM : ('I', 'M')   {units used -- English or Metric
                     system}
FreeSpaceSpeed : integer {specifies robot speed between
                          approach points.  This is an index
                          to a range of allowable robot
                          speeds.}
ApproachSpeed : integer  {specifies robot speed in
                          approaching a destination point}
DialPeckLength : real    {specifies distance for each peck
                          to dial indicator}
SRIClearance : real {specifies distance for robot to
                     position part surface away from SRI}
```

READER COMMENT FORM

IMPLEMENTATION OF THE INSPECTION ROBOT CONTROLLER

This document is one in a series of publications which document research done at the National Bureau of Standards' Automated Manufacturing Research Facility from 1981 through March, 1987.

You may use this form to comment on the technical content or organization of this document or to contribute suggested editorial changes.

Comments: _____

_____

_____

_____

_____

_____

_____

_____

_____

If you wish a reply, give your name, company, and complete mailing address: _____

_____

_____

_____

What is your occupation? _____

NOTE: This form may not be used to order additional copies of this document or other documents in the series. Copies of AMRF documents are available from NTIS.

Please mail your comments to:

AMRF Program Manager
National Bureau of Standards
Building 220, Room B-111
Gaithersburg, MD 20899

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3722	2. Performing Organ. Report No.	3. Publication Date MAY 1988

4. TITLE AND SUBTITLE

"Implementation of the Inspection Robot Controller"

5. AUTHOR(S)

Howard T. Moncarz and Bruce Borchardt

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document describes the theory and implementation of the Inspection Robot Controller (IRC) program. This controller is part of the Inspection Workstation (IWS) in the Automated Manufacturing Research Facility (ARMF) at the National Bureau of Standards. The IRC is commanded by the Inspection Workstation Controller, and in turn, the IRC supervises the IWS robot and is also the supervisor to the Surface Roughness Instrument Controller. The configuration of the workstation as well as important points throughout it are specified to the IRC as data.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

AMRF; data-driven control; inspection robot; inspection workstation; IWS; robot controller; surface roughness

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

60

15. Price

$13.95

USCOMM-DC 6043-P80