P00011 to DLA900-87-D-001 Page 47 of 48 ②

SECURITY CLASSIFICATION OF THIS PAGE

REPORT D

AD-A229 477

1a. REPORT SECURITY CLASSIFICATION Unclassified	
2a. SECURITY CLASSIFICATION AUTHORITY	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Clemson University Clemson Apparel Research	6b. OFFICE SYMBOL (If applicable)
7a. NAME OF MONITORING ORGANIZATION Defense Personnel Support Center	
6c. ADDRESS (City, State, and ZIP Code) 500 Lebanon Road Pendleton, SC 29670	7b. ADDRESS (City, State, and ZIP Code) 2800 South 20th Street P. O. Box 8419 Philadelphia, PA 19101-8419
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Defense Logistics Agency	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DLA900-87-D-0017, Delivery Order 0016	
8c. ADDRESS (City, State, and ZIP Code) Room 4B195 Cameron Station Alexandria, VA 22304-6100	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
78011S			

DTIC ELECTE DEC 05 1990 S E D

11. TITLE (Include Security Classification) Automated Guided Vehicles

12. PERSONAL AUTHOR(S) Dr. John C. Peck

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 1/4/90 TO 10/2/90	1990 October 17	23

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
11	05	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A considerable cost is incurred in today's manufacturing industry in the process of manually moving components between work stations. One possible means of reducing this cost may be found through the use of computer-controllable Automated Guided Vehicles (AGVs). However, development of suitable computer software for the specific requirements of a plant, in a cumbersome, machine-oriented AGV Operational Language, has hindered the widespread us of AGVs. Development of such software is virtually impossible for the typical AGV user, a non-programmer.

Various approaches considered for developing such a tool are analyzed in the final report. Finally, an imprementation of one approach at the Clemson Apparel Research plant is discussed.

[illegible]

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Christine W. Jarvis	22b. TELEPHONE (Include Area Code) (803) 646-8454 / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

College of Sciences
DEPARTMENT OF COMPUTER SCIENCE

CLEMSON UNIVERSITY

October 17, 1990

Accession For		
NTIS GRA&I	✓	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

Ms. Sally DiDonato
DPSC-FTCA-2
2800 S. 20th Street
P.O. Box 8519
Philadelphia, PA 19101-8419

Dear Ms. DiDonato:

Please find enclosed a copy of the final report on contract DLA900-87-D-0017, Problem Solving for Apparel Manufacturers. Final accounting records associated with the contract are now being completed in our Office of Business and Finance at Clemson University.

I have mailed a copy of this report to Mr. Dan Gearing (COTR) at Cameron Station Virginia.

Sincerely,

J C Peck

John C. Peck
Principle Investigator

cc: Mr. Dan Gearing
Dr. Christine Jarvis

90 [illegible] 014

Problem Solving for Apparel Manufacturers

Automated Guided Vehicles

Final Report

Clemson Apparel Research

John C. Peck

October 1990

Executive Summary

This contract resulted in the development of a tool to facilitate the use of Automated Guided Vehicles (AGVs) in American manufacturing industries and thereby assist in increasing the competitiveness of these industries in a global economy. Using an AGV manufactured by Litton Industries, Mr. Krishnamachari Devarajan, a graduate student in Computer Science at Clemson University, with Dr. Peck developed an simple system to control the action the AGV in factory applications. The AGV was taken to a local apparel plant where it was tested in an actual application. This final technical report is a scholarly paper, prepared by Mr. Devarajan and used to fulfil part of his requirements for the MS degree in Computer Science.

One of the objectives of the AAMTD contract between the Defense Logistics Agency and Clemson University was to demonstrate and improve the operation of advanced manufacturing technologies in the domain of apparel manufacturing. As a means of attaining this objective, an Automated Guided Vehicle (AGV) was donated by the International Business Machines Corporation to the Clemson Apparel Research Project. This device consists of a remotely controlled vehicle which can perform operations like picking ups boxes, carrying them to predefined locations and disposing of them at these locations. A guide path, consisting of a reflective tape adhered to the floor of a factory, is used to define the route for the AGV. The path consists of forks and other changes in direction as well as floor code marks to identify locations within the factory. Instructions are transmitted to the AGV from an FM radio base station connected to a microcomputer which makes decisions concerning the conditional actions to be taken by the AGV.

The programming method used by most factory personnel is to write very low level code to control the AGV. This method is extremely time consuming and expensive, is difficult to change and maintain, and usually requires a computer programming professional to perform. Consequently, AGV usage is many times restricted to companies large enough to employ a sufficiently large programming staff.

The primary goal of this project was to develop a software system through which an end-user, with no programming experience, can specify a plan of activities to be performed by an AGV. Using this information, the system automatically generates a computer program required to implement the desired plan.

The document which follows describes how this goal was accomplished.

A USER INTERFACE SYSTEM FOR AUTOMATED GUIDED VEHICLES

by

Krishnamachari Devarajan
Department of Computer Science
Clemson University, South Carolina.

ABSTRACT

A considerable cost is incurred in today's manufacturing industry in the process of manually moving components between work stations. One possible means of reducing this cost may be found through the use of computer-controllable Automated Guided Vehicles (AGVs). However, development of suitable computer software for the specific requirements of a plant, in a cumbersome, machine-oriented AGV Operational Language, has hindered the widespread use of AGVs. Development of such software is virtually impossible for the typical AGV-user, a non-programmer.

This paper presents a tool that enables a non-programmer to specify an AGV's functions for a dynamic manufacturing plant and automatically generates the required control software. Various approaches considered for developing such a tool are analyzed. Finally an implementation of one approach at the Clemson Apparel Research plant is discussed.

INTRODUCTION

In today's manufacturing industry, a substantial amount of manual effort is being spent on transferring sub-assemblies from work station to work station during the various stages involved in producing a finished product. Even the most advanced plants usually go only to the extent of installing high technology machines to automate the individual phases of the manufacturing process. They normally leave the task of constantly moving components among the work stations to manual labor, incurring a considerable cost.

An Automated Guided Vehicle (AGV) can greatly reduce the manual effort required for transfer of goods within a manufacturing plant. An AGV is basically a remotely controlled vehicle that can perform operations like picking up boxes, carrying them along a pre-defined collection of paths and delivering the boxes to specified locations. The AGV is instructed by a host computer which broadcasts packets of instructions through a Radio Frequency (RF) transmitter. Inquiries can be made by the host to the AGV regarding its status and location.

A computer program is required to periodically determine the next set of instructions to be transmitted to the AGV, based upon the following main factors:

1. *Status of the work stations:* The status of a work station determines the type of service to be rendered by the AGV.
   (e.g.) "Ready for picking up of goods"
   "Waiting for delivery of goods"
   This status can be conveyed to the computer by some external communication device like a lazer gun, door bell, bar code reader, etc.

2. *Status of the AGV:* This includes the current physical location of the AGV within the plant and the current load on the AGV. These can be obtained by the host computer by inquiring to the AGV.

Thus, control of an AGV requires an AGV Control Program (AGV-CP) which can "listen to" the work stations regarding their status, inquire to the AGV about its location and other important information, and command the AGV to perform a clearly defined set of operations. The AGV-CP must constantly perform these tasks throughout the time when the plant is in production.

Generally speaking, manufacturing personnel, who use AGVs to assist in material movement within plants, are not programmers. They need a facility to assist in specifying the AGV's functionality in their plants and to convert these specifications to a computer program which transmits the AGV's operational language.

This paper presents a User Interface System (UIS) - an interface between a non-programmer and the AGV - that accepts a definition of the functional specifications of the AGV from a non-programmer and performs transformation of these specifications to a computer program which instructs the AGV accordingly. This interface system is, in effect, a tool that automates development of a suitable AGV-CP starting from a non-programmer's specifications.

For the purposes of illustration, an apparel manufacturing plant is described, since UIS was developed and demonstrated in an apparel plant, viz. the Clemson Apparel Research plant (CAR), in Pendleton, South Carolina. The AGV used was a Litton Series 800 Automated Guided Vehicle, donated by IBM to CAR.

In what follows, the concepts and working principles of UIS are discussed. Section 1 presents an overview of the AGV and its complex communication procedure, which was a motivation for development of this tool. Section 2 identifies the required characteristics of the UIS, through a step-by-step process, from a fundamental view point. Section 3 proposes possible approaches to development of an interface system. Section 4 describes, in detail, the working procedure of UIS and finally, Section 5 concludes the paper.

For the reader's convenience, a list of acronyms used in this paper is given below:

AGV	-	Automated Guided Vehicle.
AGV-CP	-	Automated Guided Vehicle Control Program to control the AGV.
CAR	-	Clemson Apparel Research plant where the interface system was developed and demonstrated.
HLL	-	High Level Language.
TM	-	Transfer Mechanism of the AGV that transfers boxes.
UIS	-	User Interface System, the title of this paper.
UOL	-	User Operations List, a list of AGV operations built by the user for each LOCATION CODE during the specification entry.
USF	-	User Specification File, a text file containing user's specifications in the form of HLL statements.
USU	-	User Specification Utility.
VSC	-	Vehicle System Controller for communicating with the AGV.

SECTION 1.
AN OVERVIEW OF AGV AND THE OPERATIONAL LANGUAGE

1.1 Features of the Litton 800 AGV:

The Litton 800 AGV consists of a Guidance System, a Power Train, and a Control System. The control system incorporates a microprocessor giving the vehicle decision making capabilities and the ability to follow commands from a Vehicle System Controller (VSC), otherwise known as a Host Computer. The vehicle receives its commands and responds with status by communicating with the VSC via a UHF radio data communications link. The Guidance System receives information from the *Scanning* Sensor Head *sub-system and* controls the steering servo sub-system. The Power Train is a servo controlled motor that drives the front wheel of the tricycle AGV.

The basic features of the AGV include the following:

1. A Multi-Load Transport with a Transfer Mechanism (TM) that loads and unloads up to four boxes. The TM has an elevator that can transfer boxes to external devices at various height positions.

2. An Addressable Transmitter allows devices external to the vehicle to be selectively activated and deactivated. These could be door openers, annunciators or other devices.

The guide path for the AGV consists of a line of fluorescent tape (masking tape works fine) that is used to form a route for the AGV to follow. The path can have right or left branches to alternate AGV's routes. Codes are marked along the guide path to identify vehicle location. These codes will be referred to as LOCATION CODEs in the following sections. These codes are *necessary to keep track of the* vehicle's location and to designate where a particular command is to be executed.

The AGV executes lists of commands keyed to LOCATION CODEs as the codes are read along the guide path (Deferred Mode). In addition, commands can be implemented immediately upon receipt by the vehicle (Immediate Mode).

1.2 The Communications Protocol:

The Communications Protocol between the AGV and the VSC is initiated by a radio transmitting a message frame containing either the commands for the vehicle to perform or an inquiry request of vehicle status. The message frame has the following format:

Header Field, m1, m2, ..., m20, Trailer Field.

The Header and Trailer bytes contain items like opening and closing flags, address of the vehicle, hand-shaking information, etc. The Data Field (m1, ..., m20) is a sequence of 1 to 20 message blocks each of which is made of 9 bytes. A message block is basically one vehicle command. Its component bytes have the following significance:

Byte 1 : Immediate/Deferred mode indicator
Byte 2 : LOCATION CODE (For Deferred Commands)
Byte 3 : Command Code
Byte 4 : Delay Specification byte
Byte 5 : Implement Delay Value
Byte 6 : Termination parameter
Byte 7 : Auxiliary parameter
Byte 8 : 1st command parameter
Byte 9 : 2nd command parameter

1.3 Translating a User Requirement to an AGV Message:

Constructing the AGV messages for every operation to be performed by AGV is quite tedious. To illustrate this, an example is presented below:

To perform a Transfer-A-Box operation from the current position of the AGV, the message to be used is,

Byte-1	*Byte-2*	*Byte-3*	*Byte-4*	*Byte-5*	*Byte-6*	*Byte-7*	*Byte-8*	*Byte-9*
1a	*00*	*0b*	*01*	*14*	*00*	*00*	*11*	*10*

The preceding are hexadecimal bytes whose individual bits carry valuable information to the AGV. For instance, the significance of byte-8's bits are,

bit6	- Nature of the transfer
bit5	- Whether the transfer is a load or unload

bit4	- Which side of the AGV to perform the transfer
bit3 through bit0	- The number of boxes to be transferred.

Prior to the transfer operation, the AGV must be physically routed to the desired location and prepared for the transfer by several other messages, the construction of which will require equally tedious procedures.

Thus, the conventional approach of developing a custom-made program for AGVs for the specific requirements of a plant layout would involve considerable amounts of tedium and programming effort and is error-prone. Besides, modification to this program, when changes or enhancements are required, will also be difficult and error-prone.

The purpose of the UIS presented in this paper is to provide a better human interface to the AGV. A non-programmer will easily be able to accomplish the task of defining the AGV's functions as well as incorporating future enhancements.

SECTION 2.
CHARACTERISTICS OF THE UIS

The UIS has two major components to communicate the user's needs to the AGV:

1. An interactive utility that interfaces with the user to obtain the specifications of AGV's activities. This utility is referred to as the User Specification Utility (USU).

2. A suitable AGV-CP, produced by the USU, to interface witn the AGV.

To automate creation of the AGV-CP, a model for such a control program must be developed. This model will provide insight into the facilities to be incorporated into the USU. The following sections identify these requirements:

2.1 Analysis of a typical manufacturing plant

A typical manufacturing plant has several work stations, each acting on one phase of the entire manufacturing process. For example, in a shirt manufacturing plant, the individual work stations may perform operations like *stitching pockets, buttons, sleeves, cufflinks,* assembling the sub-components, producing the finished product, packaging the product, etc.

The state of each work station will be constantly changing. The work station may be waiting for its input components, ready for its products to be picked up, or even both simultaneously. Each of these states may require service from the AGV. The commands to be executed by the AGV in order to accomplish this service will depend on the AGV's current location within the plant, since these commands must include the ones for taking the appropriate branches along the guide path and skipping the intermediate LOCATION CODEs.

2.2 Methods for Identifying AGV's functions:

Considering the two main factors, namely the state of the plant and the location of the AGV, the AGV's operations in a dynamic manufacturing plant can be identified in two possible formats:

1. Use the state of the plant as a major reference, and for each value of the state, identify the AGV operations for each location where the AGV can possibly be positioned at that time.

2. Use the AGV's location as the major reference and define the operations based on the various states of the plant.

The former approach is more difficult to visualize since the major reference, viz. the state of *the plant, is a collection of numerous* combinations of the individual work station's states. On the other hand, in the latter approach one has a clearly defined major reference, the set of LOCATION CODEs. Operations to be performed at the current location can be identified for all states of the particular work station adjacent to the current location. Once the adjacent work station is serviced, the next location to be serviced by the AGV can be determined based on the states of upcoming work stations.

2.3 AGV's Role in an Example Plant Set up:

An example is presented below to illustrate the second approach presented in section 2.2:

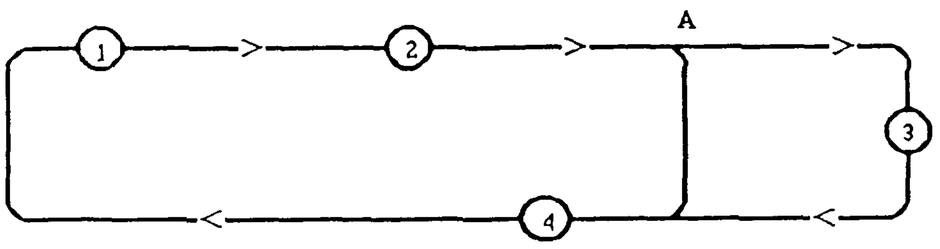


Figure 1. Layout of work stations of a plant

Consider the plant layout shown in Figure 1. The following important aspects form the basis of the AGV's functions in this plant:

- Work stations at LOCATION CODE 2 and LOCATION CODE 3 (referred to as work station 2 and work station 3) manufacture products which require pick up by the AGV when they are ready. The work stations have some communication device to inform the host computer of their status.

- LOCATION CODE 4 denotes the warehouse where the products must be delivered.

- LOCATION CODE 1 is where the AGV normally stays when there is nothing to be done. This acts as a starting point for the AGV.

- Work station 2 is of higher priority than work station 3, meaning, if both are ready at the same time, the former is serviced first; work station 3 gets service only when work station 2 demands no service.

With these parameters, the AGV's functions can be identified as shown in Figure 2.

```
In case the AGV is currently stopped at LOCATION CODE 1
    if LOC2_IS_READY or LOC3_IS_READY
        proceed to LOCATION CODE 2 and stop

In case AGV is currently stopped at LOCATION CODE 2
    if LOC2_IS_READY
        pick up goods
        proceed to LOCATION CODE 4 and stop
    else
        proceed to LOCATION CODE 3 and stop

In case AGV is currently stopped at LOCATION CODE 3
        pick up goods
        proceed to LOCATION CODE 4 and stop

In case AGV is currently stopped at LOCATION CODE 4
        deliver goods
        proceed to LOCATION CODE 1 and stop
```

Figure 2. AGV's functions in the example plant

LOC2_IS_READY and LOC3_IS_READY denote the corresponding work stations being ready for service. Please note that, since the LOCATION CODE is the major reference for AGV's next set of instructions, each of these sets must be terminated with an instruction that stops the AGV at a new LOCATION CODE.

2.4 Characteristics of AGV-CP:

In order to control the AGV for the above operations the required AGV-CP must include the instructions shown in Figure 3:

```
In a repetitive loop, do the following:

-   Inquire to the AGV to see if it is stopped at any particular location.

-   Check for occurrences of any asynchronous interrupts to set conditions like
    LOC2_IS_READY, to TRUE

-   If the AGV has stopped, then

        If the current LOCATION CODE = 1
            if LOC2_IS_READY or LOC3_IS_READY
                Set the AGV in motion
                Ask AGV to stop when LOCATION CODE 2 is encountered

        If the current LOCATION CODE = 2
            if LOC2_IS_READY
                Position the AGV's TM to appropriate height
                Activate a stationary device to feed a box
                Perform Load-A-Box operation
                Set the AGV in motion
                Make the AGV take the Right Branch at point A
                Ask the AGV to stop when LOCATION CODE 4 is encountered
                Set the condition LOC2_IS_READY to FALSE
            else
                Set the AGV in motion
                Make the AGV take the Left Branch at point A
                Ask the AGV to stop when LOCATION CODE 3 is encountered

        If the current LOCATION CODE = 3
            Position AGV's TM to the appropriate height
            Activate a stationary device to feed a box
            Perform Load-A-Box operation
            Set the AGV in motion
            Ask the AGV to stop when LOCATION CODE 4 is encountered
            Set the condition LOC3_IS_READY to FALSE

        If the current LOCATION CODE = 4
            Position the AGV's TM to appropriate height
            Activate a stationary device to accept a box
            Perform Unload-A-Box operation
            Set the AGV in motion
            Ask the AGV to stop when LOCATION CODE 1 is encountered
```

Figure 3. AGV-CP for the example plant

This is a typical sequence of instructions to be executed by the AGV-CP. The following general characteristics of the AGV-CP are now clear. The AGV-CP must,

1. Remain in a major loop while the plant is in operation.

2. Obtain the state of the work stations and the AGV. If the AGV is currently in motion, pause for a while and get back to the beginning of the loop.

3. Go through a test of the AGV's current LOCATION CODE. This can be implemented through *Case Statements* in a High Level Language (HLL) like Pascal.

4. Whenever the AGV gets to a new LOCATION CODE and stops, transmit a collection of commands based on the state of the work stations. These commands will eventually route and stop the AGV at a new LOCATION CODE.

5. Three types of statements are essential:
   - Function calls for constructing AGV message frames for the required commands.
     (e.g.) Set_the_AGV_in_motion
   - Statements for testing conditions. (e.g.) IF LOC2_IS_READY
   - Statements for setting conditions. (e.g.) LOC2_IS_READY := FALSE

2.5 Required Features of the USU:

The general format of the AGV-CP and the type of instructions required imply that the USU must include the following essential features:

- Accept the list of LOCATION CODEs where the AGV must stop to perform operations and, for each LOCATION CODE, accept the operations to be performed.

- Allow the user to specify any of the AGV operations and accept the associated parameters. For example, if *Take-A-Left-Branch* operation is to be specified, the position of the branch in the guide path with reference to either a LOCATION CODE or the current position of the AGV is a necessary parameter.

- Enable the user to include conditional performance of operations. This means a condition like *If LOC2_IS_READY* must be accepted by the USU and checked during run time by the AGV-CP. Consequently, inputting the associated *Else* and *End-If* must also be possible.

- Setting conditions at one location which might affect the operations at another location must be possible. In the above example, *Set LOC2_IS_READY to FALSE* was essential, or work station 3 would never have received service.

The USU, after accepting the input from the user, must output the specifications in some format that will form the core of the AGV-CP. With these basic guidelines, some possible approaches to an interface system are discussed in Section 3.

SECTION 3.
POSSIBLE APPROACHES TO AN INTERFACE SYSTEM

The general characteristics of the user's needs, the USU and the AGV-CP have been analyzed in section 2. The next step is to determine a method to save the user's input and utilize it in a suitable AGV-CP to accomplish the necessary functions. There are two possible approaches for this:

3.1 The Interpretive Approach:

The USU can accept the user's specifications and output a file of data that can be interpreted by a suitable *interpreter at run time* (Figure 4). This approach requires development of an interpreter with the following features:

- The method of constructing AGV messages from user's parameter data may be incorporated into the interpreter.

- For conditional execution of instructions, a format for specifying the conditions must be established and the interpreter must test the conditions and take appropriate actions at run time.

- For Setting conditions, again, a suitable syntax must be established and the interpreter must be able to assign a value to a variable at run time.

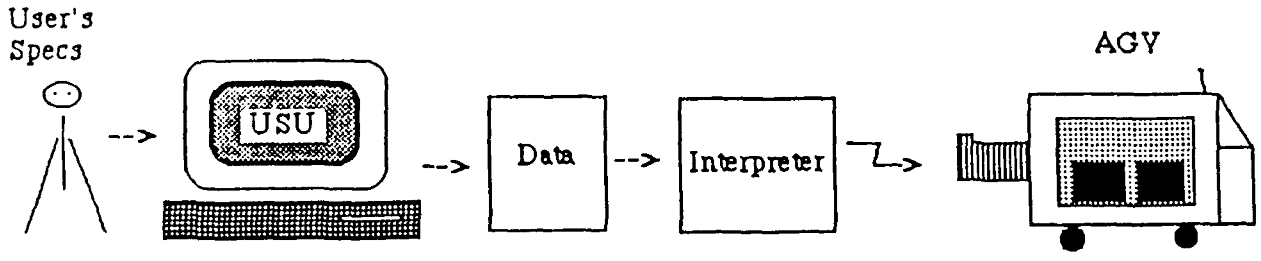


Figure 4. The interpretive approach.

Developing an interpreter involves a considerable amount of programming effort. Incorporating possible enhancements to the interface system (e.g. *While* statements), will also be tedious. A compiled approach solves these problems.

3.2 The Compiled Approach:

In this approach the USU outputs a file containing statements in a HLL, which can form the core of a target program (AGV-CP) written in the same language (Figure 5). The AGV-CP thus formed can be compiled and executed to serve the required purpose. This approach avoids the necessity of developing an interpreter and instead makes use of existing HLL compilers.

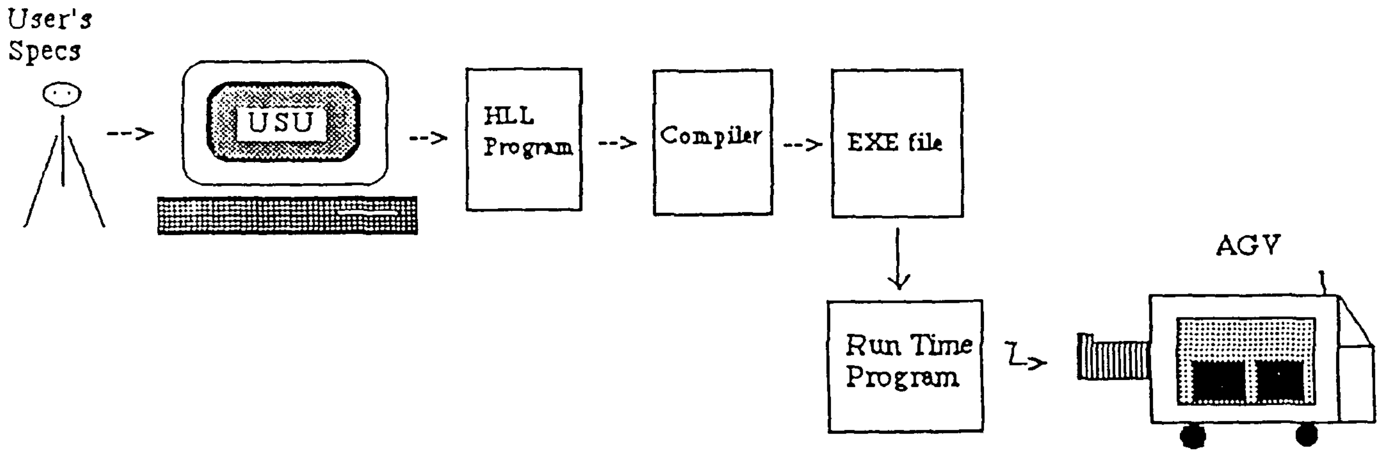


Figure 5. The compiled approach.

For each AGV operation in the user's specification, the USU can output a HLL function call. For example if FORTRAN IV is the language of the AGV-CP, then for each AGV operation, the USU can produce a statement,

CALL TRANSMIT (B1, B2, B3, B4, B5, B6, B7, B8, B9)

where *B1, ..., B9* are constants that have been calculated by the USU, based on the user input. *TRANSMIT* can be a sub-program that merely transmits the given 9 bytes. For *Test Condition* and *Set Condition* operations the USU can produce the equivalent *if* statements and *assignment* statements. In addition to being easier to implement and maintain, the compiled approach produced a program whose performance is far superior to the interpretive approach.

SECTION 4.
WORKING PROCEDURE OF THE UIS

The compiled approach was adopted for the UIS and Pascal was chosen to be the language of the AGV-CP. In UIS, the two major components identified in section 2 work as follows:

1. The USU accepts the user's input and generates a User Specification File (USF), which is a text file containing Pascal statements.

2. A model AGV-CP program, written in Pascal, has statements for all the house-keeping operations like polling the AGV, etc, and includes the USF with a *{$I ..}* compiler directive to form the complete AGV-CP.

4.1 Working Procedure of the USU:

The USU goes through two important phases, The User Input Phase and The Code Generation Phase.

4.1.1 The User Input Phase:

In this phase the user specifies the operations to be performed at various LOCATION CODEs, along with their required parameters, using a User Specification Input Screen (Figure 6.)

The Available Operations List has all the possible AGV operations as well as provision for specifying *If/Else/End-If* statements and *Set-Condition* statements. The purpose of this screen is to build the User Operations List (UOL) from the Available Operations List for a given LOCATION CODE. The Editing Mode Menu provides all the necessary editing features like Append, Modify, Delete, Insert, etc for building the UOL.

Whenever a particular element in UOL is being edited, a Parameter Input Window (PIW) specific to the particular operation, appears (Figure 7). For example if the *Set Speed* operation is chosen, then the PIW accepts the value of the speed, the implementation delay, the duration of the implementation, etc.

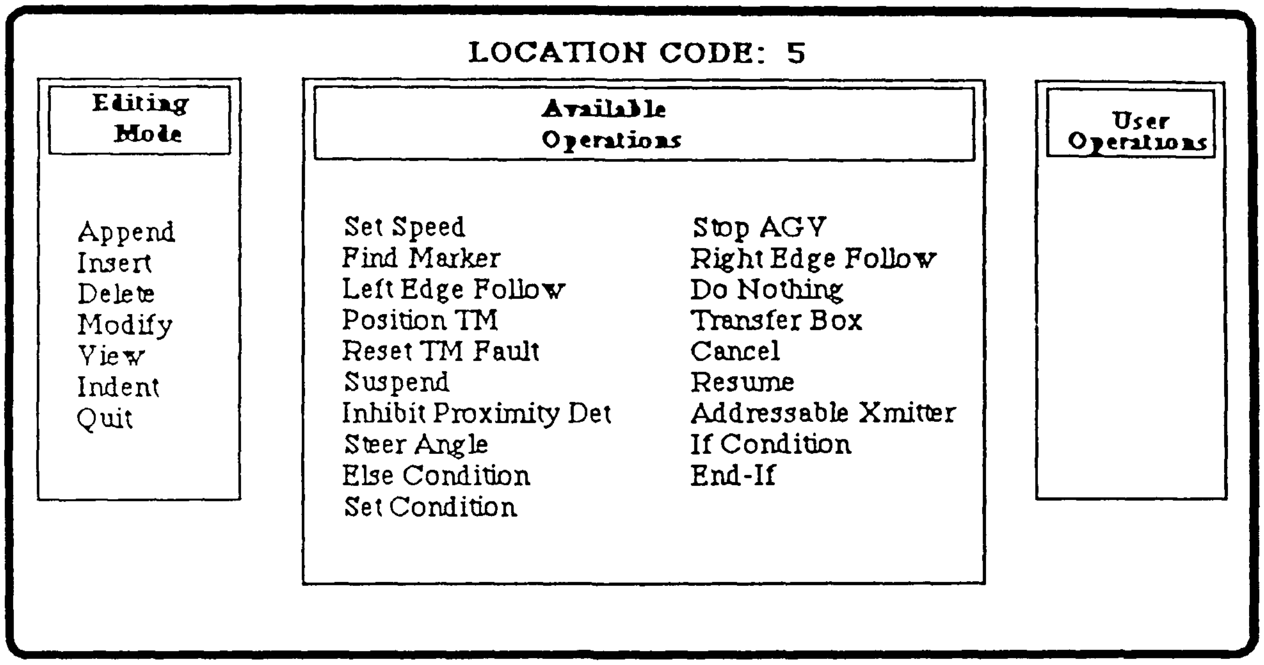
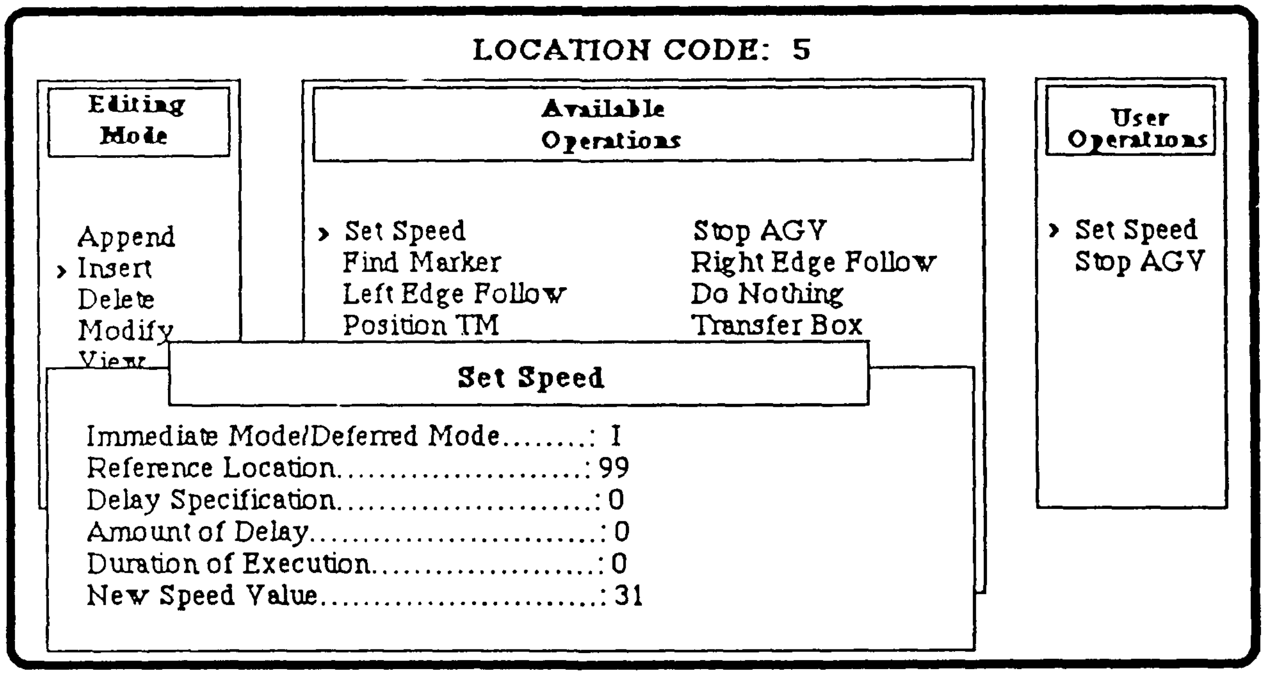


Figure 6. The User Specification Input Screen



```
Immediate Mode/Deferred Mode........: I
Reference Location......................: 99
Delay Specification.......................: 0
Amount of Delay..........................: 0
Duration of Execution.....................: 0
New Speed Value..........................: 31
```

Figure 7. The Parameter Input Window for *Set Speed* Operation

While the PIWs for all AGV operations are similar, the ones for the *If Condition* and *Set Condition* options are different. The *If Condition* option requires that the user enter a Boolean expression conforming to Pascal syntax that must be TRUE for execution of subsequent statements. The *Set Condition* option asks the user to enter a variable name and the value/expression to be assigned to the variable.

Several data items are declared in the model AGV-CP for use in the *If Condition* and *Set Condition* PIWs, and more may be added as needs arise. Presently some preliminary validations are made to the Boolean expression and assignment statements entered by the user.

4.1.2 The Code Generation Phase:

After UOLs for all the desired LOCATION CODEs are built, the USU enters the Code Generation Phase where it generates the USF. The contents of USF are shown in Figure 8.

```
Procedure Process_Loc_Operations;
begin
    Case Loc_Code of

        L1:  begin
             stmt 1.1
             stmt 1.2
             .
             .
             stmt 1.1'
             end {L1}

        L2:  begin
             stmt 2.1
             stmt 2.2
             .
             .
             stmt 2.2'
             end {L2}
        .
        .
        Ln:  begin
             stmt n.1
             stmt n.2
             .
             .
             stmt n.n'
             end {Ln}

    end; {case}
end; {Procedure Process_Loc_Operations}
```

Figure 8. The User Specification File

In the USF, *L1, L2, ..., Ln* are the LOCATION CODEs entered by the user. The UOL of each LOCATION CODE *Li* is translated to a block of Pascal statements, *<Begin stmt i.1; ...; stmt i.i'; End>* , each of which represents one operation in the UOL.

The rules for formulating these statements are as follows:

1. Every AGV operation is translated to a Pascal function call,
   formcmd (b1, b2, b3, b4, b5, b6, b7, b8, b9);
   where *b1,...,b9* are AGV instruction bytes computed by the USU based on the operation and the user's parameters.

2. An *If Condition* is translated to
   If <Boolean> then begin
   where *<Boolean>* is the Boolean expression entered by the user for the *If Condition* option.

3. An *Else* option forms the lines
   end
   else begin
   in the USF.

4. An *End_If* option forms the line
   end;
   in the USF.

5. The Set Condition is outputted in the USF as,
   <variable> := <value>;
   where *<variable>* and *<value>* are the variable name and the value entered by the user.

The procedure *Process_Loc_Operations* will be invoked by the AGV-CP during run time.

The user requirements outlined in Figure 3 were entered using the USU. The corresponding USF generated by the USU is shown in Figure 9. A one-to-one correspondence may be observed between the instructions in Figure 3 and the statements in Figure 9. A comparison of these two figures helps understand the rules for USF generation.

```
Procedure Process_Loc_Operations;

Begin
   Case Loc_Code of

      1: begin
         If (r2 or r3) then begin
            formcmd ($1a, $00, $85, $00, $00, $00, $00, $1f, $00);   {Set Speed}
            formcmd ($02, $02, $ff, $00, $00, $00, $00, $00, $00);   {Stop AGV}
         end;
         end; {1}
      2: begin
         If (r2) then begin
            formcmd ($1a, $00, $84, $00, $00, $00, $00, $dc, $00);   {Posn TM}
            formcmd ($1a, $00, $01, $01, $10, $14, $00, $84, $00);   {Xmitter}
            formcmd ($1a, $00, $0b, $01, $1e, $00, $00, $01, $00);   {Xfer Box }
            formcmd ($1a, $00, $85, $01, $40, $00, $00, $1f, $00);   {Set Speed}
            formcmd ($1a, $00, $83, $00, $05, $0a, $00, $00, $00);   {Rt Edge Fl}
            formcmd ($02, $04, $ff, $00, $00, $00, $00, $00, $00);   {Stop AGV}
            r2 := FALSE;
         end
         else begin
            formcmd ($1a, $00, $85, $00, $00, $00, $00, $1f, $00);   {Set Speed}
            formcmd ($1a, $00, $82, $00, $05, $0a, $00, $00, $00);   {Lf Edge Fl}
            formcmd ($02, $03, $96, $00, $00, $00, $00, $00, $00);   {Stop AGV}
         end;
         end; {2}
      3: begin
            formcmd ($1a, $00, $84, $00, $00, $00, $00, $bb, $00);   {Posn TM}
            formcmd ($1a, $00, $01, $01, $10, $14, $00, $82, $00);   {Xmitter}
            formcmd ($1a, $00, $0b, $01, $1e, $00, $00, $01, $00);   {Xfer Box }
            formcmd ($1a, $00, $85, $01, $40, $00, $00, $1f, $00);   {Set Speed}
            formcmd ($02, $04, $ff, $00, $00, $00, $00, $00, $00);   {Stop AGV}
            r3 := FALSE;
         end; {3}
      4: begin
            formcmd ($1a, $00, $84, $00, $00, $00, $00, $0a, $01);   {Posn TM}
            formcmd ($1a, $00, $01, $01, $10, $14, $00, $81, $00);   {Xmitter}
            formcmd ($1a, $00, $0b, $01, $1e, $00, $00, $21, $00);   {Xfer Box }
            formcmd ($1a, $00, $85, $01, $40, $00, $00, $1f, $00);   {Set Speed}
            formcmd ($02, $01, $ff, $00, $00, $00, $00, $00, $00);   {Stop AGV}
         end; {4}

   end; {case}
end; {Process_Loc_Operations}
```

Figure 9. USF for a test plant set up

In Figure 9, r2 and r3 are Boolean variables declared in the model AGV-CP and represent work station 2 and work station 3, respectively, being ready. Each *formcmd* procedure call includes the hexadecimal bytes calculated by the USU based on the AGV instruction and the data obtained in the PIWs.

4.2 Model AGV-CP:

The model AGV-CP has the following structure:

```
While Plant_is_in_operation do begin

    Inquire_AGV;  {To get the current LOCATION CODE and status of the AGV}

    If an External_Interrupt_has_Occurred
        Set_the_Work_Station_Status_Variables;

    If the AGV_is_Stopped then begin
        Form_Header_Part_of_the_AGV_Message_Packet;
        Process_Loc_Operations; {The procedure in the USF}
        Form_Trailer_Part_of_the_AGV_Message_Packet;
        Transmit_the_Packet_to_the_AGV;
        Wait_until_the_AGV_Starts_Moving;
    end;{If}

end;{While}
```

The primary components of the model AGV-CP are explained below:

- Inquiring of the AGV assigns the last read LOCATION CODE to the global variable *Loc_Code*.

- AGV being in motion would imply that it is still in the process of executing the previous message packet received, since every message packet is terminated with a *Stop-at-a-new-location* message. Thus, only when AGV has stopped will the AGV-CP send the next message packet.

- The *Case* statements in the *Process_Loc_Operations* procedure pass control to the appropriate block of the current *Loc_Code* where the necessary message packet for the particular *Loc_Code* is constructed through *formcmd* procedure calls. *formcmd* is a simple procedure in the model AGV-CP that appends the given nine bytes to the AGV message packet.

- On insertion of the header and trailer parts, the message packet becomes a valid message frame acceptable to the AGV, and is then transmitted to the AGV.

- The AGV-CP idles until the AGV starts moving from the current location, or transmission of the same message packet may be repeated at the current location. Once the AGV starts moving, the above cycle repeats.

- Meanwhile, the AGV-CP also maintains the variables indicating work station status values through external communications from the work stations.

SECTION 5. CONCLUSIONS

This paper presented a User Interface tool for specification of an AGV's function in a manufacturing plant. Even a non-programmer can easily interact with this tool since it provides descriptive explanations for the parameters as they are entered, together with a context-sensitive help screen. Through use of this tool, manufacturing specialists can concentrate on the tasks they are best trained for, those involving planning and direct manufacturing. The programming tasks associated with AGV operations are entirely eliminated. This tool avoids the necessity for learning the machine-oriented AGV Operational Language. Thus, an enormous amount of tedium, confusion and errors in developing and maintaining a suitable AGV-CP is avoided.

This tool provides a fundamental framework for the automation of AGV-CP creation. The tool at present outputs the AGV operations in the form of Pascal code; however, it could be easily modified to output code in any target language. The set of statements supported by the tool can be increased as new requirements are identified.

REFERENCES

1. G. Mayer, "Industrial Experience with an automated guided vehicle system", *Proceedings of the 2nd International Conference on Automated Guided Vehicle Systems and 16th IPA Conference, Stuttgart, W. Germany,* pp 73-84, 1983.

2. J.-O. Myhrman, "Flexible Manufacturing Systems (FMS) with AGVS," *Proceedings of the 2nd International Conference on Automated Guided Vehicle Systems and 16th IPA Conference, Stuttgart, W. Germany,* pp 147-164, 1983.

3. F. Schneider, "AGVS used as a total transport system in a new plant," *Proceedings of the 2nd International Conference on Automated Guided Vehicle Systems and 16th IPA Conference, Stuttgart, W. Germany,* pp 301-318, 1983.

4. Edward H. Frazelle, "Design Problems in Automated Warehousing," *IEEE 1986 International Conference On Robotics and Automation, Vol. 1,* pp 486-489, 1986

5. R. Bohlander, "Comparisons of Automated Techniques for AGV Control," *IEEE 1986 International Conference On Robotics and Automation, Vol. 1,* pp 496-503, 1986.

6. Litton UHS Automated Vehicle Systems, "Programming Guide Series 800 Vehicle and Multi Load Transport," 1983.

7. Litton UHS Automated Vehicle Systems, "User's Guide Series 800 Vehicle and Multi Load Transport," 1984.

8. Litton UHS Automated Vehicle Systems, "Operator's Manual Series 800 Automated Guided Vehicle model 9106A," 1983.

9. Litton UHS Automated Vehicle Systems, "Automatically Guided Vehicles for materials handling in light manufacturing," 1982.