SYSTEM BASED LADDER LOGIC SIMULATION AND DEBUGGING

by

Krishna Kumar Krishnan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree
of
Master of Science
in
Industrial and Systems Engineering

Approved:

De Meter, E.C., Co-Chairman
Deisenroth, M.P., Co-Chairman

O. K. Eyada

May, 1991
Blacksburg, Virginia
SYSTEM BASED LADDER LOGIC SIMULATION AND DEBUGGING

by

Krishna Kumar Krishnan

Committee Chairmen: Edward C. De Meter

& Michael P. Deisenroth

Industrial Engineering

(ABSTRACT)

PLCs are extensively used for the discrete and continuous control of non-intelligent shop-floor devices. The debugging phase of ladder logic development for PLCs is very cumbersome and difficult. Most often on-line debugging which is expensive and time consuming is used for debugging. Computer simulation techniques applied to this problem, leaves much to be desired. The best technique developed for ladder logic debugging is the use of ladder-based triggers. A ladder-based trigger is a function which suspends simulation execution whenever a vector of ladder variables equates to a vector of predefined states.

System-based debugging facilities are those which aid a programmer in error detection at the system level. System based triggers will identify system faults and set traps within a simulation model to detect their occurrence. This approach will provide information necessary for a faster correction of the ladder logic once a trigger is activated.
The system based debugging tool developed is capable of scanning a boolean representation of a PLC program with input coils, counters, timers, "and" conditions, "or" conditions and output coils.

The program provides the following facilities:

1. Graphics programs can be attached to the simulation program for better visualization.

2. The simulation program allows interactive control over the test bed developed. In a non-interactive simulation it can be executed in a timed sequential mode or random mode.

3. Triggers can be set by the user depending on the conditions that are to be monitored.

4. The program stops execution whenever a trigger is activated.

5. The program provides a trace of the output that caused the trigger and also of the inputs to this output, along with their state values at the time of activation.

The use of system based techniques and graphics in the debugging of PLC ladder logic is demonstrated. Further the use of an object oriented frame work in the development of the debugging software is also demonstrated.
Acknowledgements

I would like to express sincere thanks and deep gratitude to my advisor and chairman, Dr. Edward C. De Meter, for his endless patience and encouragement.

Thanks are also due to Dr. M.P. Deisenroth for serving as co-chairman in the committee and providing valuable criticism.

Special thanks are due to Dr. O.K. Eyada for serving on my committee and for his constant encouragement.

I would also like to express great gratitude to Dr. Myklebust for providing me with the facilities and the unbridled use of the Computer Aided Design lab, Dr. Jayaram, Uma and Kolady for providing timely and valuable help in the programming phase and the staff of CAD lab for their cooperation.

I wish to thank Dr. Dhande for his help during the initial phase of the thesis in literature review for graphics packages and for serving on my committee during the proposal stage of my thesis.
I would like to thank Joni and Lavonda for their constant support and for the administrative assistance during the writing of my thesis. I also would like to thank Mrs. Lovedia Cole for her help during the past three years.
# Table of Contents

1.0 Introduction ............................................. 1  
   1.1 Problem statement ...................................... 4  
   1.2 Research objective ..................................... 6  
   1.3 Significance of research ................................. 9  

2.0 LITERATURE REVIEW ......................................... 11  

3.0 METHODOLOGY .................................................. 14  
   3.1 Automated Conveyance System Description .............. 15  
   3.2 Object-Oriented Model Specification .................... 19  
   3.3 System-Based Trigger Development ....................... 25  
   3.4 3-D Graphical Animation Development ................. 27  
   3.5 Tool Demonstration Environment Development .......... 27  
   3.6 Hardware and Software Used ............................. 29  

4.0 SYSTEM DESIGN .................................................. 30  
   4.1 Format of PLC program ................................. 31  
   4.2 Initialization of Switches and Boolean File .......... 40  
   4.3 Interactive Phase of Setting Up Simulation .......... 40
4.4 Graphic Modules ........................................ 52
4.5 Execution of the Simulation .............................. 55
4.6 System-Based Debugging .................................. 58

5.0 TEST BED .................................................. 62
5.1 Feature of Test Bed developed ............................ 62
5.2 Setup Procedure for the Execution
   of Simulation Program ...... 65
5.3 Execution of the Simulation program .................... 68
5.4 Sample PLC Programs ...................................... 69

6.0 SUMMARY AND RECOMMENDATIONS ....................... 71
6.1 Conclusions ................................................. 71
6.2 Future Work ............................................... 72

7.0 REFERENCES ................................................. 74

Vita .................................................................

vii
LIST OF ILLUSTRATIONS

Figure 1. A Typical Ladder Diagram ...................... 2
Figure 2. Transaction Flow Network ..................... 5
Figure 3. Schematic Diagram of Converter Model ........... 16
Figure 4. State Diagram for Converter Model ............... 18
Figure 5. Activity Entity Object Class ................... 21
Figure 6. Passive Entity Object Class .................... 22
Figure 7. GIBSS Language Frame Work .................... 23
Figure 8. Classification for Test Bed Objects ............... 26
Figure 9. PLC Program Format ............................ 32
Figure 10. Equivalent 'OR' Condition ..................... 35
Figure 11. Counter Representation in PLC .................. 36
Figure 12. Timer Representation in PLC ................... 39
Figure 13. Instructions for Use of Software ................ 42
Figure 14. Attaching the Graphics Module .................. 43
Figure 15. Selecting the Interaction Mode .................. 44
Figure 16. Random or Timed Mode .......................... 45
Figure 17. Selecting the Triggers ......................... 47
Figure 18. Graphical Model of the Conveyance System ...... 49
Figure 19. The TRACER File .............................. 59
Figure 20. The RESULT File .............................. 60
LIST OF APPENDICES

1.0 Appendix 1 -- Source Code

2.0 Appendix 2 -- PLC Program With Format Error

3.0 Appendix 3 -- PLC Program With System Based Error

4.0 Appendix 4 -- PLC Program With No Error
1.0 Introduction

A programmable logic controller (PLC) is a microprocessor-based computer which is designed specifically to execute functions such as logic sequencing, timing, counting, and arithmetic to control various types of machines or processes through digital or analog input/output modules.

Today, PLCs are extensively used for the discrete and continuous control of non-intelligent shop-floor devices. Source code for PLCs is written in a primitive language known as ladder logic. A ladder logic program (Figure 1) contains networks of contact and coil symbols which represent Boolean equations. In addition, ladders also include symbols for representing variables of various data types and functions used for counting, timing and math.

Ladder programs created for industrial applications tend to be quite large due to the primitive nature of ladder
Figure 1. A Typical Ladder Diagram.
constructs. Programs consisting of over 500 networks are not uncommon. As a result, one of the most difficult stages in ladder logic development is that of debugging and validation.

Frequently, ladder debugging is conducted on-line. Unfortunately, on-line testing interrupts production and can be dangerous to both humans and machinery. In addition, it places the burden of error detection on the code developer, who must visually monitor for system failures and track ladder faults which lead to their occurrence.

An alternative to on-line debugging is the use of computer simulation. Computer simulation is a descriptive modeling technique, which when applied to the analysis of a controller and a controlled system, provides insight into their interactions.

The potential advantages of using computer simulation for ladder debugging are:

1. it is an off-line technique, and thus does not require system down-time, 2. it is flexible, and thus may be applied to any industrial system, 3. it is relatively inexpensive to execute, and 4. if properly structured, it
relieves a programmer of the burden of error detection and ladder fault tracking.

1.1 Problem statement

Over the past ten years, simulation modeling environments have been prototyped to facilitate ladder logic debugging. Unfortunately, these environments fail to employ modeling and debugging techniques which fully exploit the usefulness of computer simulation. Currently, the most sophisticated environment requires users to model industrial systems with transaction flow networks such as the one illustrated in Figure 2. While an accurate means of representing discrete event behavior, the use of these networks for modeling complex systems is labor intensive, error prone, and requires considerable modeling expertise on the part of a user. In addition, transaction flow networks are inadequate for modeling physical behavior, which typically requires the use of differential or difference equations in combination with continuous simulation techniques.

With respect to ladder debugging facilities, the most sophisticated tool offered is the ladder-based trigger. A
Figure 2. Transaction flow network.
ladder-based trigger is a function which suspends simulation execution whenever a vector of ladder variables equates to a vector of predefined states. Through the use of a trigger, a user can identify variable states which may result in system failure, and set traps within a simulation model to detect their occurrence. When activated, a trigger identifies the last ladder rung executed and automatically causes a trace of previously executed events to be stored. While an invaluable tool, the use of ladder-based triggers requires a programmer to correlate system failures to combinations of ladder variable states. While tractable for small ladders, it is extremely difficult for ladders of any size or complexity.

1.2 Research Objective

The objective of this research is to investigate new modeling and debugging techniques which may enhance the use of computer simulation as a ladder logic debugging tool. In particular, this research will demonstrate the usefulness of object-oriented model specification and system-based debugging techniques.
1.2.1 **Object-Oriented Model Specification**

Currently object-oriented simulation program generation is a widely used tool in simulation analysis. An object-oriented simulation program generator (SPG) is an interactive software tool that translates a model described by the logical assemblage of objects into the code of a simulation language, general purpose language, or machine language.

A simulation object is a modeling construct which is similar in abstraction to the entity which it represents and which hides the mechanics of its operation from a modeler. Currently, object-oriented SPGs are used in robotic programming environments such as IGRIP [1], ROBOTICS [2], and Robot-Sim [3].

When modeling with an object-oriented SPG, a user selects, instantiates, and links objects from a model-base to represent entities within a system of study. The SPG accepts this specification, and translates it into an executable model. While methods of translation vary, a popular method used for both commercial and prototype SPGs is object-oriented translation. Using this scheme, a
generator creates object code by scanning a specification for object references and the identification of object dependencies. Using this information, it instantiates objects from a model-base and links them together to form an executable model. With regard to object-oriented translation, a simulation logic framework dictates how objects are combined, executed, and interacted.

In this research, the use of an object-oriented logic framework for the specification of a ladder logic debugging simulation model is demonstrated. In particular, the GIBSS (Generalized Interaction Based Simulation Specification) [9] framework will be utilized to model the entities and interactions within a PLC controlled, automated conveyance system.

1.2.2 System-based Debugging Techniques

System-based debugging facilities are ones which aid a programmer in error detection at the system level. Two system-based techniques which will be investigated and demonstrated are system-based triggers and 3-D graphical animation. System-based triggers are identical to ladder-based triggers with the exception that they monitor system
events rather than ladder variable states. Using system-based triggers, a user will be able to identify system faults and set traps within a simulation model to detect their occurrence. Two facets of system-based triggers which will be investigated are their manner of specification in an object-oriented framework and the types of information they will provide upon activation.

3-D graphical animation involves the real-time, dynamic, graphical display of simulation objects. It facilitates a user's understanding of an executing simulation by providing visual information which is directly associable to the system being simulated. Currently, 3-D animation is a standard output of robotic programming environments such as the ones previously listed. This research investigates how 3-D graphical animation can be combined with system-based triggers and manual simulation control to enhance error detection and program validation.

1.3 Significance of the Research

Currently no software tool fully supports the ladder logic development cycle. This research is a major step toward the development of such a tool. The knowledge gained
from modeling a PLC controlled system in an object-oriented format will provide a major step toward the development of object-oriented SFGs capable of facilitating PLC/industrial system modeling. The development of system-based triggers will, in most cases, result in a tool capable of detecting errors faster than real time. Finally, the development of 3-D graphical animation as a debugging aid will provide a means of visually demonstrating the validity of ladder programs.
2.0 Literature Review

This literature review will provide a history of the development of simulation environments created for ladder logic debugging.

Lampen [4] was one of the first to use computer simulation as a tool for ladder logic debugging. He created and executed a simulation model of a manufacturing system and its controlling PLC. The model of the PLC executed the Boolean equivalent of a ladder program. The main thrust of Lampen's work was to illustrate how ladder execution could be simulated. Consequently, his work did not address issues such as debugging facilities or the general modelling of industrial systems.

Le Quoc, Cheng, and Athanasoulas [5] were one of the first to prototype a ladder simulation and debugging environment. However, their system is restricted to PLC
simulation. To describe manufacturing equipment I/O, a user creates a text file using structured language constructs. Their environment combines this data with a ladder diagram to create an executable model of a PLC. During model execution, a user simulates machine inputs by providing values to the environment. Likewise, the user reads simulated PLC output from a terminal screen.

Le Quoc and Cheng [6] developed a pc-based environment which permits the iconic modeling of simple ladder networks using electrical components. Their system, PC-SIM, is user-oriented and features interactive graphics programming and the interactive setting of input device states. During simulation execution, it provides a graphic display of current-carrying devices and output states.

Galgocy [7] developed an interactive simulation environment for debugging ladder programs used to control an iconic model of a flexible manufacturing system. His system allows a user to import and/or edit a ladder program. This program is then translated into an executable model of a PLC and linked to an executable model of the FMS. To aid in ladder debugging, Galgocy’s environment provides sequential event lists, time-phase diagrams, ladder-based triggers, and breakpoints.
During simulation, Galgocy's environment graphically displays and stores a time-ordered listing of FMS and ladder execution events. By visually scanning this data, a user can track the performance of the ladder and system over simulated time. To complement this event list, the environment also stores a history of ladder variable states. A user may examine this data either in tabular form or through the use of graphically displayed time-phase diagrams.

Mecker [1] added constructs to the SIMSCRIPT simulation language to facilitate the transaction flow modeling of ladders and the systems which they control. In addition, he created a set of graphical routines which display time-phase diagrams of ladder variables and system attributes.
3.0 Methodology

To investigate the modeling and debugging techniques discussed in Chapter 1, a test-bed approach was taken. Using this methodology, all algorithms created in this research were developed, coded, and tested relative to a specific application. In the case of this research, a PLC controlled, iconic model of an automated conveyance system, served as a test bed. The steps which were taken in this research were:

1. the modeling of the PLC controlled conveyance system using the GIBSS framework,

2. the development of system-based triggers using the GIBSS framework and their integration into the conveyance system model,
3. the integration of 3-D graphical animation with the system-based triggers previously developed, and

4. the creation of a tool demonstration environment.

Each of these steps are discussed after a description of the test-bed system.

3.1 Automated Conveyance System Description

A schematic diagram of the automated conveyance system is provided in Figure 3. This model currently exists within the PLC lab in Whittemore Hall. The conveyance system transports boxes to bins via a conveyor and diverts the boxes into the bins via rams. The conveyor is driven by a constant speed motor.

Two types of boxes, type A and type B, move down the conveyor. Part A is the smaller of the two. These boxes are diverted by rams A and B respectively. A photo detector is used in combination with a conveyor drive tooth counter to identify boxes and track their movement.
Figure 3. Schematic diagram of Converter Model.
When operational, the conveyance system is considered to be in one of three states, as seen in Figure 4. While in the "Emergency Stop" state, the conveyor motor is off, the rams are retracted, and the "Emergency Alarm" light is on. The system enters this state whenever the "Emergency Stop" switch is thrown on. The system remains in this state until the "Emergency Stop" switch is in the off position and until a pulse is received from the "System Reset" button. Note that the system powers up in the "Emergency Stop" state.

While in the "Conveyor Feed" state, the rams are retracted, and the conveyor motor is on. Any box that passes the photo detector while the system is in the "Conveyor Feed" state is not diverted. Instead it is allowed to run off the conveyor into the scrap bin. This also holds true for any box that is on the conveyor when the system enters the "Conveyor Feed" state. The "Emergency Alarm" light is also on.

While in the "Normal Operation" state, the system identifies boxes entering into the system, tracks them, and diverts them with the appropriate rams. To exit this state, the "Emergency Stop" switch is thrown on, sending the system back into the "Emergency Stop" state.
Figure 4. State Diagram For Converter Model.
Inputs from the conveyance system are:

1. "System Reset" Switch,
2. "Converter Feed" Switch,
3. "System Start" Switch,
4. "Emergency Stop" Switch,
5. Photo Detector Input, and
6. Tooth_Counter Input.

Outputs to the conveyance system include:

1. "Motor_On" signal,
2. "Ram_A_Extend" signal,
3. "Ram_A_Retract" signal,
4. "Ram_B_Extend" signal,
5. "Ram_B_Retract" signal, and

3.2 Object-Oriented Model Specification

The automated conveyance system and its controlling PLC are modeled using the GIBSS logic framework. GIBSS was
chosen due to its ability to integrate continuous simulation and discrete event simulation in an object-oriented format. Using GIBSS, objects are classified as active entity, passive entity, or interaction. An active entity object (Figure 5) represents system entities whose behavior is relevant enough to simulate on an individual basis. Active entity objects change their internal state in response to the passage of time or stimuli from their environment. Stimuli are introduced by the direct manipulation of their attributes by interaction objects.

Passive entity objects (Figure 6) represent system entities whose attributes are relevant to an analysis, but whose behavior is not relevant enough to simulate on an individual basis. A passive entity object has the characteristic of being able to change the number and types of attribute sets it possesses. Each attribute set represents a different level of abstraction. Manipulations by interaction objects ultimately dictate which attribute sets are used.

Interaction objects (Figure 7) represent the interactions between entities. To simulate interactions, they monitor the passage time or changes in the internal states of entity objects. If the conditions of an
Active Entity Object:

- Represents an entity whose behavior is relevant enough to simulate on an individual basis
- Alters its attributes in response to the passage of time or changes in its attributes
- Executes a combination of timed-based activities, conditional activities, and time-based attribute functions to simulate entity behavior to the desired level of abstraction
- Determines its future occurrence time using either a fixed increment or variable increment time flow mechanism
- Uses its interaction buffer to identify pending interaction with dynamic passive objects

Figure 5. Active Entity Object Class.
PASSIVE ENTITY OBJECT CLASS

Passive Entity Object:

- Represents an entity whose attributes are relevant but whose behavior is not relevant enough to simulate on an individual basis.

- Changes the number and types of attribute sets which it possesses to meet the requirements of its pending interactions.

- Uses a relational database to cross reference attributes whenever an attribute value is modified by an interaction object or when a new attribute set is established.

- If static, uses an interaction buffer to identify pending interaction with dynamic entity objects.

- If dynamic, does not use an interaction buffer.

Figure 6. Passive Entity Object Class.
**GIBSS LANGUAGE FRAMEWORK**

**ACTIVE OBJECT 1**
- Interaction Buffer
- Object 3

**DYNAMIC PASSIVE OBJECT 3**

**ACTIVE OBJECT 2**
- Interaction Buffer

**INTERACTION OBJECT 4**
- Clock
- Simulation Logic
- Interaction Buffer
- Object 1
- Object 2

**Interaction Object**:

- Simulates the actions between groups of entity objects
- Monitors the passage of time or changes in the internal states of entity objects
- Determines if entity objects are within a state of pending interaction
- Determines if interactions occur
- Simulates interactions by modifying the attributes of entity objects
- Determines its future occurrence time using either a fixed increment or variable increment time flow mechanism

*Figure 7. GIBSS Language Framework.*
interaction are met, they simulate the results of the interaction by modifying the attributes of the interacting entity objects. For instance, interaction between box A and ram A, would require the output 'y2' to be true and the box A to be in front of it. The 'Master simulation object' or the driver monitors the conditions. The conditions are fed to ram A, depending on which, ram A is activated. All the interaction modules are called from within the active entity object.

An executable CIBSS model is comprised of instantiated classes of active entity, passive entity, and interaction objects bound together via a master simulation object. Active entity objects and interaction objects use logic flows which permit them to execute combinations of time-based activities, conditional activities, and time-based attribute functions. When combined with the logic flow of a master simulation object, a composite logic flow resembles the three-phase approach modified to handle the execution of time-based attribute functions. Passive entity objects execute a logic flow which enables them to manage and simultaneously update multiple attribute sets. To do so, they utilize a relational database which links attributes and their associated values.
A projection of how the conveyance system is modeled is illustrated in Figure 8. In this the PLC, conveyor, photo detector, and rams are modeled using active entity objects while the boxes traveling down the conveyor are modeled using passive entity objects. Interactions such as the arrival and departure of boxes from the conveyor, the ejection of boxes by the rams, and the movement of boxes by the conveyor are modeled using interaction objects.

3.3 System-based Trigger Development

System-based triggers were developed by first examining the ladder/conveyance system and deducing an appropriate set of traps. Traps can be set for monitoring the improper firing of rams, lighting of the Emergency Alarm light, and running of the conveyor motor. Next, the types of information which were provided by triggers during activation were investigated. Triggers identify the ladder rungs which cause errors, and identify the inputs to these ladder rungs. In addition, each of these inputs, if they are coils are traced further. Next, the triggers are specified and coded in an object oriented format. They are treated as interaction objects, since they strictly perform object monitoring operations. Finally, the triggers are
Active Entity Objects

PLC
Conveyor
Photo Detector
Rams

Passive Entity Objects

Boxes

Interaction Objects

Box Arrival
Box Departure
Ejection of Boxes
Box Movement

Figure 8. Classification for Test Bed Objects.
integrated into the previously created conveyance model for testing and evaluation.

3.4 3-D Graphical Animation Development

The development of 3-D graphical animation as a debugging aid was conducted as follows. First, various types of visual information which enhance a programmer's recognition of system failures were identified. Next, the manner in which visual information was tied with the executing entity objects, interaction objects, and system-based triggers was ascertained. Finally, graphical animation code was created and integrated with the conveyance system model code.

3.5 Tool Demonstration Environment Development

To demonstrate the models and debugging tools developed in this research, a simple demonstration environment was created. This environment allows:

1. a user to specify the requirements of a graphics program to be attached,
2. a user to interactively specify the triggers to be set for the program,

3. a user to define the nature of interaction for the executing program, i.e., either a interactive or a non-interactive nature, and if non-interactive, whether in random mode or a timed mode.

Subsequently, the environment will:

1. accept a boolean representation of a ladder program,

2. simulate execution of the ladder and conveyance system,

3. allow a user to schedule the arrival of a box into the system,

4. permit interactive control of the simulation,

5. automatically suspend simulation execution upon activation of a trigger, and
6. provide a graphical animation of the executing simulation upon request of the user.

3.6 **Hardware and Software Used**

Software development in this research was done primarily with the C programming language. The exception to this was the development of animation code. Several graphics/ graphical packages like CADAM, CATIA, and HOOPS were considered for use. CADAM has good shading features and allowed the creation of 3-D objects, but it does not support graphical animation. CATIA allows 3-D graphical animation, but interfacing with software written in C was found to be difficult. Hardware facilities for running HOOPS were not available. As a result, the GRAPHICS standard library was used to develop the animation code.

An IBM-RT serves as the hardware platform in combination with an IBM-5080 graphics terminal.
4.0 **SYSTEM DESIGN**

This chapter describes the various functions developed for the simulation environment. It describes the PLC program format which is to be debugged, the PLC scanning function and the interactive setup facilities. It also describes the various graphics modules developed. In addition, it describes the interaction modules required for the various simulation modules and the overall execution of the program.

The PLC scan program is also used to perform an initial debugging of the PLC program. This debugging checks for format errors, such as repeated specification of output coils or outputs which are not referenced before. The interactive setup procedures allow the user to define the different modes in which the simulation is executed. The program may be executed in an interactive or non-interactive mode. The graphics module need not be attached for
executing the simulation program. Triggers can be set so as to trace the occurrence of predefined conditions.

In an interactive mode, the user can select different switches which are represented on the screen. A selection of a switch will cause a change in the state value of the switch. Any change in the system attributes will cause PLC scan function to be executed and cause an updating of all the coil values.

4.1 Format of PLC program

A PLC program format was defined to facilitate scanning by the C-program developed for interacting with the PLC program file. A format of the PLC program is shown in Figure 9. The PLC program is capable of having inputs like switches, counters, timers, coil outputs and inputs. Input coils or switches are represented by x’s and n’s, where n represents x-nots. Relay coils are represented by c’s and C’s, where C represents a c-not. Coil outputs which represent the final output sent to the shop floor to control the hardware are represented by y and Y (where Y represents y-not). An "or" condition is represented by "o". Timers are represented by T’s and counters by S’s.
Figure 9. PLC Program Format.
A ladder program can have eleven inputs and one output coil for each rung of the ladder. The inputs are placed starting from column 1 and can have a maximum column size of 5. The space between the beginning of each adjacent input is separated by five columns. The column numbers 55 to 60 are reserved for the output coil. In case the number of inputs is less than eleven, a '0' is placed at the starting point of each of the possible input positions. The program is stored in a file called "ptr".

4.1.1 Representation of inputs and coil outputs

All inputs and outputs are represented by Ixx where I is the type of input or output and the second and third x following each representation are integers or Instruction Reference Numbers, which uniquely identify the input or output.

Any change in the state of the switch caused by the user selecting the switch will result in the boolean value of this switch being changed in the file "bol". The PLC scan program is also executed immediately after the change so as to update the values of the coils. Pulse switches are
implemented by changing the state value corresponding to the switch, followed by the execution of the PLC scan program, and then resetting the state value to the original value followed by another execution of the PLC scan program.

4.1.2 Representation of the 'or' condition

An "or" condition is represented by 'o'. The equivalent of the 'or' condition represented in line #9 of Figure 9 is shown in Figure 10. The 'or' condition is represented such that if all the inputs before the 'o' or if the inputs after 'o' hold true the output coil will be activated. For instance, in Figure 9, c9 will be activated if c1, x20 and x16 has the value 1, or if c2, and x16 has value 1 and x17 has value 0.

4.1.3 Representation of a Counter

The counter representation in a PLC is shown in Figure 11. When the enable input line is on, the counter increases its value by one each time the input line goes from off to on. When the current count is equal to the preset count P, the output line from the box turns on. When
Figure 10. Equivalent ‘OR’ Condition.
Figure 11. Counter Representation in PLC.
the enable input turns off, the current count is reset to zero. In a PLC program the counter is represented by Sxx, where xx refers to the counter number or the instruction reference number. The counters and timers are defined in a common data structure STD and contains information as to the name of counter or timer, the inputs/coils for the reset/enable input line and the value of the reset line so that if it is true it has a value 1 and if it is false it has a value 0. A total of seven input coils can be assigned to the enable/reset line. Since the data structure for a counter and timer are the same, care must be taken to see that the counter number is not the same as that for a timer. This is in keeping with the manner in which PLC's allocate memory space to timers, and counters.

At the beginning of program execution, the counter value is initialized to zero. The value that the counter has to attain is assigned to the variable cnt, which is defined in the data structure for a counter. The counter is activated only if the enable line is true. For a change of the input line from off to on, the value of the counter is increased, by increasing the value of count corresponding to the counter. The value of count is then compared with that of the preset value, stored as cnt in the data structure for
counter. If the preset value and the counter value are the same, the output coil is activated.

4.1.4 Representation of a Timer

The timer in a PLC is represented as shown in Figure 12. When the enable input line and the start input line of the timer box is on, the timer decrements time from the preset time $P$ down to zero. If the start input line turns off and the enable input line remains on, the timer stops. If the start input is turned back on with the enable input still on, the timer resumes timing. If the enable input line is turned off during, or after the $P$ equals zero, the timer resets to the preset time $P$. When $P$ attains the value zero the output line from the timer box is activated.

In a PLC program a timer is represented by $Txx$, where $xx$ refers to the timer number or the instruction reference number. When both the input line and enable line for the timer has a true value the value of preset time cnt is decreased as the simulation proceeds. If the input line to the timer is turned off and the enable input line remains on the timer stops decreasing the time. It will resume decreasing the time when the input line is turned back on.
Figure 12. Timer Representation in PLC.
Whenever the enable input line is off the value of cnt is set back to the preset time. When the value of cnt is zero the output coil connected to the timer is activated.

4.2 **Initialization of Switches and Boolean file**

The boolean values representing the high/low states for the switches are stored in the file "bol". The file "bol" contains a series of 0's and 1's. The values for the switches are stored such that the position of the boolean value corresponding to the switch nxx is the xx th position. For instance, the boolean value for switch x23 will be stored in the twenty third position from the start of the file "bol".

4.3 **Interactive Phase of Setting up the Simulation**

Before the actual execution of the simulation, the program allows the user to define the requirement of display, mode of execution, triggers to be set type of interaction required, and the type of non-interactive execution. The program prompts the user through the various inputs that are required for setting up the simulation. The
program contains a series of interactive setups through which the user is led, before the actual execution of the simulation. The user may go from one setup to the next by selecting the box with the display "NEXT SCREEN".

The first screen contains descriptions of the software and the instructions required for the use of this software (Figure 13). It describes the different types of inputs required from the user and the different modes in which the program can be run. This screen may be exited by selecting the "NEXT SCREEN".

The second screen allows the user to determine whether to have graphics attached to the simulation program or not (Figure 14). Even if the graphics program is not attached, the program displays the switches and boxes and allows the execution of the program in an interactive mode if the user desires to.

The simulation environment allows the user to specify the interactive nature of the program (Figure 15 and Figure 16). The program can be set to execute the simulation in an interactive manner allowing the user to select the switches and the boxes to be placed on the conveyor at random. It can also be set up to execute with no user interaction, wherein any of the switches may be turned on or off by the
LADDER LOGIC DEBUGGING SOFTWARE

WELCOME to the ladder logic debugging software
THIS SOFTWARE IS DESIGNED TO READ A PLC
PROGRAM IN A SPECIFIED FORMAT AND CHECK FOR
ERRORS IN A SYSTEM BASED MANNER. BEFORE YOU
EXECUTE THE SOFTWARE, MAKE SURE OF THE FOLLOWING:
1. SPECIFY THE VALUE OF N, WHERE N IS THE
   # OF LINES IN THE PLC PROGRAM
2. SPECIFY THE ENABLE INPUTS TO THE COUNTER

SELECT THE 'NEXT SCREEN' TO PROCEED

Figure 13. Instructions for Use of Software.
Figure 14. Attaching the Graphics Module.
LADDER LOGIC DEBUGGING SOFTWARE

INTERACTIVE MODE

NON-INTERACTIVE MODE

NEXT SCREEN

INTERACTIVE OR NON-INTERACTIVE MODE?

De Meter
&
Krishna Kumar
VPI & SU

Figure 15. Selecting the Interaction Mode.
Figure 16. Random or Timed Mode.
program, in a random manner without any input from the user. The user will have no control over the program whatsoever. The program may also be run in a timed sequential fashion, wherein again the user has no interactive control over the switches. But the switches are turned on/off in a set sequence, which is set to execute in a predefined order.

In the next screen the user is allowed to select the triggers which the user would like to set for debugging the PLC program (Figure 17). The triggers can be set by selecting the appropriate boxes set up for this purpose.

The various triggers that can be set are listed below:

1. Ram A pushing Box A down,

2. Ram A pushing Box B down,

3. Ram B pushing Box A down,

4. Ram B pushing Box B down,

5. Motor on while in emergency stop state,

6. Ram A being activated with no box in front of it,
LADDER LOGIC DEBUGGING SOFTWARE

1. RAM A
   WITH
   BOX A

2. RAM B
   WITH
   BOX A

3. RAM A
   WITH
   BOX B

4. RAM B
   WITH
   BOX B

5. MOTOR
   ON
   IN
   E. STOP

6. RAM A
   WITH
   NO BOX

7. RAM B
   WITH
   NO BOX

8. EMERG
   IN
   E. STOP

NEXT
SCREEN

PLEASE SELECT THE TRIGGER CONDITION

De Meter
&
Krishna Kumar
VPI & SU

Figure 17. Selecting the Triggers.
7. Ram B being activated with no box in front of it,

8. Emergency lamp is on while still in emergency stop state.

4.3.1 **Attaching the Graphics Program.**

The program allows the user to execute the program with or without the graphics program attached to the executing program. If the graphics program is attached to the executing program the graphical representation of the conveyance system is displayed on the screen (Figure 18). A conveyor with two rams positioned along the conveyor, a photo detector input device, the emergency lamp and the two boxes which the user can choose to be placed on the conveyor are displayed on the screen.

The following switches or inputs to the system are displayed on the screen as menu items:

1. Emergency stop switch. (E.STOP)
2. System reset switch. (S.RESET)
3. Conveyor feed switch. (C.FEED)
Figure 18. Graphical model of the Conveyance System.
4. System start switch. ( S.START )

In addition another switch, STOP.SIM, is provided to let the user exit out of the program any time he desires so. The switches can be activated by selecting any of the displayed menu items.

4.3.2 Setting up of Triggers.

Triggers can be set by the user for debugging and tracing the coils which activates the trigger in the eventuality of its occurrence. The triggers can be set for:

1. Motor being switched on in the emergency stop condition, or before the system reset button switch is selected,
2. Movement of ram A in emergency stop state,
3. Movement of ram B in emergency stop state,
4. Movement of ram A in conveyor feed state,
5. Movement of ram B in conveyor feed state,
6. Movement of ram A when box 1 is not in position,
7. Movement of ram B when box 2 is not in position,
8. Activation of emergency light when not in emergency state.
Any trigger condition when selected will write a value of '1' into a boolean file 'trigger' set up for this purpose. Any trigger condition selected can be removed from the trigger list by selecting the trigger condition again. There are no triggers set at the beginning of the simulation and it is left to the user to decide the triggers to be set.

In the event of any trigger condition being activated, the execution of the simulation program is stopped and will execute the tracing function which will trace the output which caused the trigger to be activated and the inputs to this output.

4.3.3 **Setting up Interaction Mode for Simulation Program.**

The program prompts the user for the interaction required. The program can be executed in a non-interactive or interactive mode. In the interactive mode the program allows the user to select the switches at random and also allows the user to determine the type and arrival of boxes into the system. In the non-interactive mode, the program can be executed in one of two ways. It can be set to run
in a timed sequence wherein the program goes through a predetermined and fixed sequence of activation of switches. In this mode the emergency stop button is first set off. Then the system reset switch is selected, after which the system moves in to the conveyor feed state. The arrival of box A into the system is then executed followed by the change of the system to the System start/Normal operation state. As soon as it is in the system start state, the arrival of box B into the system is scheduled.

4.4 Graphics Modules

A number of different graphical modules have been developed. The code for graphics modules are written using graPHIGS and the C language. They are classified as:

1. Static modules, which represent structures that do not interact physically with the other objects.

2. Menu modules, which represent structures that can be selected and are provided for the interactive running of the debugging software (example: switches, boxes to be placed on conveyor).
3. Dynamic modules, which represent structures that provide animation capabilities and at the same time do not have any physical interaction with the other modules (example the conveyor when there is no box placed on it).

4. Interaction modules, which represent structures that have two or more physical devices interacting to provide graphical animation and allow visual debugging. These are attached to the active entity which is part of the interaction.

All the modules executed in the emergency stop state will be represented by static modules and menu modules. In case of animation the structures that undergo change are deleted and then the animation modules execute the new structure for the same device. For example, in the emergency stop condition, the conveyor is represented by a static module and then when it moves to the conveyor feed state the static structure is deleted and the animation module is activated.
4.4.1 Graphical Interaction Modules

Several graphical interaction functions have been defined for the purpose of describing the interaction between different objects. The interaction between the following entities are defined:

1. box 1 and the conveyor when box 1 is placed on the moving conveyor.

2. box 1 and the conveyor when box 2 is placed on the moving conveyor.

3. box 1 and ram A when ram A is activated and if box 1 is in front of ram A.

4. box 2 and ram A when ram A is activated and if box 2 is in front of ram A.

5. box 1 and ram B when ram B is activated and if box 1 is in front of ram B.

6. box 2 and ram B when ram B is activated and if box 2 is in front of the ram B.
The graphics screen displays different menu items representing switches to be used as inputs. The switches are set to the system being simulated and will prompt the user through the different stages of the system simulation.

4.5 Execution of the Simulation.

The executable program is stored in the file "kri". The first screen that is displayed contains the instructions necessary for the execution of the program. If the "NEXT SCREEN" display is selected, the second screen which prompts the user for graphics display to be attached or not is displayed. In the next screen, the nature of interaction of the executing simulation program can be specified, i.e. as to whether it is interactive or non-interactive and if non-interactive, whether the program should be executed in a random manner or in a timed sequence.

4.5.1 Preliminary Debugging

After the setup is completed, the PLC scan function is executed before simulation is started. This PLC scanning
program is used to conduct a preliminary debugging. The initial debugging checks for non-standard input or output names so as to ensure that the PLC program uses only standard input and output variables as specified in the previous paragraphs. It also checks for a duplicate assignment of output coils. If the program is in the predefined format, it will assign values to the inputs and the outputs based on the values stored in the boolean file "bol". The file "bol" is initialized such that the values correspond to that of the desired initial condition.

### 4.5.2 Execution of the Main Program

The graphics part of the demonstration environment is executed only if the initial debugging does not hit any error. The graphics, once displayed allows an interactive or non-interactive approach to the debugging phase depending on what the user has requested. If the program is being executed in an interactive fashion, a change in the system state can be caused by the appropriate selection of the switches displayed on the menu item. Any change in the system which is caused by the selection of any switch will cause a change in the boolean value stored in the file "bol". Unlike in a regular PLC, the scanning function for
the PLC program is executed only when there is a change in the system. This scanning will in turn update the state values stored for each of the input or output variable. The scanning of the PLC program is repeated until there is no change in the values of the output values from one scan to the next.

4.5.2.1 Monitoring Function

A master simulation function is used to monitor all the various conditions that occur during the execution of the simulation. This function provides the appropriate conditions to the various entities. It monitors the state values of the output coils and the counter values. These conditions are used by the entities to select the appropriate graphics modules and to select the interaction modules. After activation of the appropriate graphics subroutine it checks for the input switch values and confirms the state condition of the overall system as to whether it is in emergency stop state, or conveyor feed state etc.
4.6 **System Based Debugging**

The second phase of debugging which is system based debugging, is done whenever the program hits a trigger. At this juncture, a function to trace the output coil which activated the trigger is executed. This function traces and identifies the outputs and inputs that caused the trigger condition to be executed. Further, it traces any c-coils to the line where it is an output, thus completing the tracing.

As soon as a trigger is activated the graphical program as well as the simulation is stopped and the function trace outputs the results of the trace to two files. The first file called "TRACER" contains all the lines responsible for the output, along with their line numbers. A typical "TRACER" file is shown in Figure 19. The second file called "RESULT" contains the state values of each of the inputs responsible for the trigger. A typical "RESULT" File is shown in Figure 20. The subroutine trace is a recursive function. It takes the output which caused the trigger to be activated, identifies the ladder rung to which the output belongs, and it checks the inputs to this ladder rung. The ladder rung is written out to the "TRACER" file. The inputs to this ladder rung and their corresponding states are
Figure 19. The TRACER File.
The output y2 is identified as belonging to line #12
The input c1 belongs to line #0 and is in state 1
The input c5 belongs to line #4 and is in state 1
The input c3 belongs to line #2 and is in state 1
The input x4 is in state 0
The input c9 belongs to line #9 and is in state 1
The input n1 is in state 0
The input x4 is in state 1
The input n1 is in state 0
The input x3 is in state 1
The input x2 is in state 0
The input n1 is in state 0
The input c7 belongs to line #6 and is in state 1
The input x4 is in state 1
The input x7 is in state 0
The input x6 is in state 1

Figure 20. The Result File.
outputed to the file "RESULT". If any of the inputs are c-coils or c-nots or y's, they are further traced by using the same function trace() to the ladder rung where they are outputs. All the inputs to the ladder rung which caused the error can be traced along with their corresponding states at the time of hitting the error.
5.0 TEST BED SETUP

The program created in this research allows the debugging of a PLC program, developed in the format specified, using system based triggers and graphical animation. It allows the interactive specification of triggers and allows the user to define the interactive nature of the execution simulation. This chapter describes the features of the test bed developed for this research and the mechanics of usage of the programs created for the demonstration of the debugging tools. The source code is provided in Appendix 1.

5.1 Features of test bed developed

The test bed uses a model of the conveyance system described in Chapter 3. The program can be executed either with graphics attached or without graphics attached. It
allows the user to specify the interactive nature of the executing program, i.e. either an interactive or non-interactive mode. In the non-interactive mode the program may be executed either in a random mode or in a timed mode. In the random mode the switches are activated in a random fashion, while in the timed mode the switches are activated in a timed predetermined sequence. System-based triggers, defined for easy debugging of the ladder program, have been set which the user can specify interactively before the actual execution of the simulation.

In an interactive and graphical mode of execution of the program, the graphical representation of the conveyance system is displayed on the screen (Figure 18). A conveyor with two rams positioned along the conveyor, a photo detector input device, the emergency lamp and the two boxes which the user can choose to be placed on the conveyor are displayed on the screen. The user can rotate the relevant graphical modules of the system about any three of the axes, by turning the valuator input dials attached to the graphical modules. This will provide the user with an additional facility of viewing the system from any desired position.
The following switches or inputs to the system are displayed on the screen as menu items:

1. Emergency stop switch. (E.STOP )
2. System reset switch. (S.RESET )
3. Conveyor feed switch. (C.FEED )
4. System start switch. (S.START )

In addition another switch, STOP.SIM, is provided to let the user exit out of the program any time he desires so. The switches can be activated by selecting any of the displayed menu items.

The output coils are attached to the devices as follows:

1. y1 to "Motor_On" signal,
2. y2 to "Ram_A_Retract" signal,
3. y3 to "Ram_A_Extend" signal,
4. y4 to "Ram_B_Retract" signal,
5. y5 to "Ram_B_Extend" signal and
6. y6 to "Emergency_Alarm_Light" signal.
5.2 Setup Procedures for the Execution of Simulation Program

The PLC program that is to be debugged should be filed under the name "ptr". The number of lines in the program should be defined in the file "def.c" as N. The enable line to each counter in the PLC program should be set in the structure STD. The enable inputs are to be assigned to the array "enable" in this structure.

Before the actual execution of the simulation, the program allows the user to define the requirement of display, mode of execution, triggers to be set, type of interaction required, and the type of non-interactive execution. The program prompts the user through the various inputs that are required for setting up the simulation.

5.2.1 Attachment of the Graphics Module.

The program allows the user to execute the program with or without the graphics program attached to the executing program. If the graphics module is to be attached the user may select the appropriate menu display. If the graphics
module is attached, the graphical representation of the conveyance system is displayed on the screen (Figure 18).

The following switches or inputs to the system are displayed on the screen as menu items:

1. Emergency stop switch. (E.STOP)
2. System reset switch. (S.RESET)
3. Conveyor feed switch. (C.FEED)
4. System start switch. (S.START)

In addition another switch, STOP.SIM, is provided to let the user exit out of the program any time he desires so. The switches can be activated by selecting any of the displayed menu items.

5.2.2 Defining the Interaction mode of Simulation Program.

The program prompts the user for the interaction mode required. The program can be executed in a non-interactive or interactive mode. The user may select the interaction mode that he requires. If no interaction mode is selected, the default is to execute the program in the interactive
mode. In the interactive mode the program allows the user to select the switches at random and also allows the user to determine the type and arrival of boxes into the system. In the non-interactive mode, the program can be executed in either of two ways. It can be set to run in a timed sequence wherein the program goes through a predetermined and fixed sequence of activation of switches. In this mode the emergency stop button is first set off. Then the system reset switch is selected, after which the system moves into the conveyor feed state. The arrival of box A into the system is then executed followed by the change of the system to the System start/Normal operation state. As soon as it is in the system start state, the arrival of box B into the system is scheduled.

5.2.3 Triggers Set up.

Triggers can be set to be activated for debugging and tracing the coils which caused the trigger to occur. The trigger conditions which can be set are:

1. Motor being switched on in the emergency stop condition, or before the system reset button switch is selected,
2. Movement of ram A in emergency stop state,
3. Movement of ram B in emergency stop state,
4. Movement of ram A in conveyor feed state,
5. Movement of ram B in conveyor feed state,
6. Movement of ram A when box 1 is not in position,
7. Movement of ram B when box 2 is not in position,
8. Activation of emergency light when not in emergency state.

The user may select any of the trigger conditions. If no trigger conditions are selected, the default is to have no triggers. In the event a trigger condition which has been selected is activated, the execution of the simulation program will be stopped. It will then execute the tracing function which will trace the output which caused the trigger to be activated and the inputs to this output.

5.3 Execution of the Simulation Program.

The executable program is stored in the file "kri". The first screen that is displayed contains the instructions necessary for the execution of the program. If the "NEXT SCREEN" display is selected, the second screen which prompts
the user for graphics display to be attached or not is displayed. In the next screen, the nature of interaction of the executing simulation program can be specified, i.e. as to whether it is interactive or non-interactive and if non-interactive, whether the program should be executed in a random manner or in a timed sequence.

5.4 Sample PLC Programs.

Three different sample PLC programs have been generated to demonstrate the various features of the debugging facilities. The programs are set so as to check for format errors, which demonstrate the initial debugging phase and also for system errors which will demonstrate the system based debugging features. Also a program with no error and which would demonstrate the simulation execution of the conveyance system in the intended manner is also generated.

5.4.1 PLC Program #1.

The first program contains format errors (Appendix 2). This will demonstrate the initial debugging phase, which checks for repetitive coil outputs, whether the program uses
c-coils which it has not referenced before and other similar coding or formatting errors. The output will be written on to the "RESULT" file.

5.4.2 PLC Program #2.

The second program contains a system error (Appendix 3). This error has been set so as to activate the ram A when the box 2 is in front of it, instead of box 1. However, this error will be traced only if the trigger condition corresponding to this error is requested to be activated. This error is one of the conditions for which a trigger has been created to demonstrate the second debugging phase.

5.4.3 PLC Program #3.

The third program is the correct PLC program (Appendix 4). This program will execute the system simulation as it is expected to be run. The program was created to demonstrate the simulation execution of the conveyance system.
6.0 Conclusions and Recommendations

6.1 Conclusions

A debugging tool for Programmable Logic Controller programs was developed. It is capable of scanning PLC programs with input switches, input coils, counters, timers, "and" condition, "or" condition and output coils. The environment developed will accept a boolean representation of a ladder diagram and will simulate execution of the conveyance system. The program can be executed with or without graphics programs attached to the simulation program. It will permit a user to specify system based triggers which are set to identify occurrences of specified events, the occurrence of which stops the simulation and traces the output which caused the event to occur and in addition the inputs to this output coil. The simulation developed can be executed in an interactive or non-interactive mode. In the interactive mode and with the graphics program attached, the simulation program will allow
the user to schedule the arrival of a box into the system, permit interactive control of the system, provides a graphical animation of the executing simulation and automatically suspends simulation upon activation of a trigger. On hitting a trigger, the system will suspend simulation and trace the output coil which caused the error. It will also trace the inputs which caused the error along with their state values. Further if the inputs are $c^-, C^-, y^-$ or $Y^-$ coils, they are traced to the lines in the program which caused the coils to be in this state.

6.2 **Future work**

The debugging tool should be updated so as to be able handle other inputs such as A/D converters, drums and the full range of PLC devices. Modules, static or dynamic should be specified interactively, allowing the user to specify the properties of each module and take advantage of the features of the object-oriented paradigm. The program should be able to tie any input switch to an user defined input coil name, and also tie an output coil name to the device it controls. For instance, the program should allow the user to have Ram A being activated to any output coil
name, say y₁, y₂, or y₅. This would allow a greater amount of flexibility in the usage of input variable names used.

The graphics program can be enhanced by letting the graphic icons to be specified interactively. This would allow the user to specify the number of icons of a type required, its position and other parameters. For instance it would allow the user to specify any number of rams and would also allow other attributes like size, and position to be specified interactively, allowing the user to treat the various modules as icons. In addition it should allow the specification of the interaction modules associated with the icon. It should also allow the user to define the triggers to be set. This would result in an environment which would allow the modeling of systems.
References


APPENDIX 1
# include "def.c"

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ MAIN PROGRAM FOR CONVERTER SIMULATION $ */
/* $ Written by: Krishna Kumar K. $ */
/* $ $ */
/* $ $ */
/* $ $ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

/* *********************** GLOBAL VARIABLES *********************** */

int i, j, r, I, k, X, P, q, r, m, M, AJ, IC, IK, SCAN, aq;
int LENGTH, IDL, tt;
static int DATA;
int aSiZe[3], numb, status, CI[100][12];

/* *********************** CHARACTER VARIABLES *********************** */

char TMSG[8];
char TEXT1[31], TEXT2[5], TEXT3[6], TEXT4[7], TEXT5[8], TEXT6[7], TEXT7[8];
char TEXT8[19], TEXT9[19], TEXT10[19], TEXT11[19], TEXT12[19], TEXT14[19];
char TEXT15[15], TEXT16[13], TEXT17[13], TVPI[6], TNAME[13], TEXT41[3], TEXT42[2];
char TEXT43[4], TEXT44[6];
static char DATA[]="";
char strl[]="";

/* *********************** FLOAT VARIABLES *********************** */

float intensity, x1;
float PTL[9];
float uX, uY, aX, aY;
float P[78][2], x, y, z, l, w, h;
float xys[3];
float HIGH, MATRA[4][4], MATRB[4][4], MATRC[4][4], MTRL[4][4], MATR2[4][4];
float csize[3], a1, a2, a3;
float vaX, vaY, valz, vall, C[3], R1[3];
float XY[3], CI[3], CVAL, TOTAL, HIGH1;
float mix[3];
float area[6], area1[6], area2[6], area3[6], area4[6];
float INTEN = 10.;

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ COLOR TABLE VALUE DEFINITION $ */
/* $ $ */
/* $ $ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

static float AR[27] =
(0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ DEFINITION OF POSITIONS FOR TEXT $ */
/* $ $ */
/* $ $ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
static float POS[2] = {0.71, 0.45}; /* POSITION E.STOP */
static float POS1[2] = {-0.9, 0.85}; /* POSITION TITLE */
static float POS2[2] = {0.73, 0.65}; /* POSITION MENU */
static float POS3[2] = {-.25, .6}; /* POSITION PROMPT 1 */
static float POS4[2] = {0.71, 0.25}; /* POSITION SYSTEM RESET */
static float POS5[2] = {0.71, 0.05}; /* POSITION CON.FEED */
static float POS6[2] = {0.71, -.14}; /* POSITION SYS.START */
static float POS7[2] = {0.71, -.34}; /* POSITION STOP */
static float POS8[2] = {-.99, -.9}; /* POSITION MESSAGES */
static float POS9[2] = {-.25, .5}; /* POSITION PROMPT 2 */
static float POS0[2] = {-.25, .4}; /* POSITION PROMPT 3 */
static float POSN[2] = {0.54, -.69}; /* POSITION Krish--- */
static float POS41[2] = {-.2, .2}; /* POS WITH GRAPHICS */
static float POS42[2] = {-.2, -.3}; /* POS WITHOUT GRAPHICS */
static float POS43[2] = {.8, -.5}; /* POS WITHOUT GRAPHICS */
static float POS44[2] = {-.82, -.55}; /* POS WITHOUT GRAPHICS */

/* ******************* DEFINITION OF VIEWPORT AND WINDOWS ******************* */

static float wind1[4] = (0.0, 0.75, 0.0, 0.75);
static float V1[6] = (0.0, 0.75, 0.11, 0.86, -2.3, 0.75);
static float wind2[4] = (0.0, 1.0, 0.0, 0.13);
static float V2[4] = (0.0, 1.0, 0.0, 1.0);
static float wind3[4] = (0.0, .75, 0.0, 0.1);
static float V3[4] = (0.0, .75, 0.0, 0.1);
static float wind5[4] = (0.0, 0.15, 0.0, 0.6);
static float V5[4] = (0.85, 1.0, 0.26, 0.86);
static float PT[3] = (0.00, 0.00, 60.0);

/* ********** BEGINNING EXECUTION OF THE MAIN PROGRAM ********** */

main()
{
    rite2(1, '1'); /* INITIALIZATION OF THE BOOLEAN FILE */
    rite2(2, '0');
    rite2(3, '0');
    rite2(4, '0');
    rite2(5, '0');
    rite2(6, '0');
    rite2(7, '0');
    rite2(8, '0');
    rite2(9, '0');
    rote2(1, '0'); /* INITIALIZATION TO NO RANDOM SEQUENCE */
    rote2(2, '0'); /* INITIALIZATION TO NO TIMED SEQUENCE */
    rote2(6, '1'); /* INTERACTION REQUIRED */
    rote2(7, '0'); /* NO INTERACTION REQUIRED */
    rote2(11, '1'); /* WITH GRAPHICS */
    rote2(12, '0'); /* WITHOUT GRAPHICS */
}

NPT[0]=3;
NP[0]=4;
aq = 0;
LENGTH = 1;
valx = -15.5;
valy = 23.3;
valz = -13.3;
TNOW = 0.;

GPOPPh (ERFILE,0); /* Open graphHIGS */
GPOPWS (2,CONNID,WSTYPE);
GPOPCh (11,10);
GPOFCF (11,111);
GPOFCO (11,111);

/* GPVMPI (11,wind1,V1,1,PT,0.0,10.0,-10.0); */
GPVCH (11,12,112,0,1,0,2); /*

TEXT (); /* DESCRIBES TEXT DATA */
GPCR (1,0,9,AR); /* LOAD COLOR TABLE */
ASOV(); /* ASSOCIATE VIEWS */
iinit_val(); /* INITIALIZE DEVICES */
picitems1(); /* DISPLAY MENU AND OTHER DATA */

GPUPWS(1,2);
err(.,-95,-7,03, "WELCOME to The ladder logic debugging software.");
err(-95,-6,025, "THIS SOFTWARE IS DESIGNED TO TAKE A PLC");
err(-95,-5,025, "PROGRAM IN A SPECIFIED FORMAT AND CHECK FOR");
err(-95,-4,025, "ERRORS IN A SYSTEM BASED MANNER. BEFORE YOU");
err(-95,-3,025, "EXECUTE THE SOFTWARE MAKE SURE OF THE FOLLOWING: ");
err(-95,-2,025, " 1. SPECIFY THE VALUE OF K, WHERE K IS THE");
err(-95,-1,025, " 2. SPECIFY THE ENABLE INPUTS TO THE COUNTERS");
err("SELECT THE NEXT SCREEN TO PROCEED");
do 
{
setu1(); /* SETS UP INITIAL DISPLAY SCREEN */
GPUPWS(1,2);
GPPKF (11,1,incl1,1,excl1);
/* PUT PICK FILTERS */
GPANCE(110,14,wsid,clas,devid); /* AWAIT EVENT */
if (clas == 5)
GPTDF (1,ADPH,PPATH);
GPUPWA (1);
GPFLV (1,5,1);
} while (PPATH[1] !=43);
GPEST (20);
GPDRV (1,0,20);
GPEST (40);
GPDRV (1,0,40);
grap(); /* SETS UP INITIAL DISPLAY SCREEN */
GPEST (40);
GPDRV (1,0,40);
if (vrot (11) == '1')
{
GPVMPI (11,wind1,V1,2,PT,0.0,10.0,-10.0);
GPVCH (11,12,1,12,0,1,0,2);
}
inact();
if (vrot (4) == '0')
{
ratim ()
}
trigr ();
picitems2 (); /* DISPLAY MENU AND OTHER DATA */
err("SELECT ANY SWITCH"); /* DISPLAY PROMPTS */
GPUPWS (1,2);
if (trig() ==1) return;
FORT();
printf("AFTER NO ! 2");
emer1();
va1x = va1x*22./7.360.;
valy=valy*22./7./360.;
valz=valz*22./7./360.;
GPROXY(valx,MATRA);
GPROTY(valy,MATRB);
GPROTZ(valz,MATRC);
GPCKT3(MATRA,MATRB,MATR1);
GPCKT3(MATRC,MATR1,MATR2);
GPVM3(l,l,MATR2);
GUPWA(l);
BOX();

if( vtrg(5) == 'l') {
  if ((vseek(1) == 'l') & getca("y1") ==)
    Trace ("y1");
    err("SELECT STOP.SIM SWITCH");
}

if( vtrg(8) == 'l') {
  if ((vseek(1) == 'l') & getca("y5") == l)
    Trace ("y5");
    PPATH[1] = 4;
    GPMSONG(1,8,TMSG): /* TERMINATION MESSAGE */
    GPCLPH(): /* CLOSE graphigs */
    return;
}

init();
printf("AFTER NO ! 5");
do {
  GPPKF(l,l,incl1,l,excl1): /* PUT PICK FILTERS */
  GRANMV(l,l,&wsid,&clas,&devid): /* AWAIT EVENT */
  if (clas == 3) {
    if (devid == 1)
      GPGTVL(valx); /* UPDATE WORKSTATION */
    GUPWA (l);
    GPFEVL(1,3,1);
  }
  if (devid == 2)
    GPGTVL(ivaly);
    GUPWA (l);
    GPFEVL(1,3,1);
  }
  if (devid == 3)
    GPGTVL(ivalz);
    GUPWA (l);
    GPFEVL(1,3,1);
  }
  if (vrot(2) == 'l') {
    ```
for (t=0; t < 1000000000; t++)
{
  printf("TNOW = %.2f", TNOW);
  TNOW = TNOW + 0.5;
  if (TNOW >= 30)
  {
    ID = 0;
    if (t1 == 2) DRIVE2();
    DRIVE(ID);
  }
  if (TNOW >= 60)
  {
    ID = 1;
    if (t1 == 2) DRIVE2();
    DRIVE(ID);
  }
  if (TNOW >= 90)
  {
    ID = 2;
    if (t1 == 2) DRIVE2();
    DRIVE(ID);
  }
  if (TNOW >= 190)
  {
    ID = 3;
    if (t1 == 2) DRIVE2();
    DRIVE(ID);
  }
  if (TNOW >= 290)
  {
    ID = 5;
    if (t1 == 2) DRIVE2();
    DRIVE(ID);
    DRIVE2();
  }
  if (TNOW >= 300)
  {
    ID=4;
    PPATH[1] = 4;
    GPMMSG(1,8,TMSG);
    GPCLPH();
    return;
  }
}
}
t1 = 3;
if (class == 5 && vrot(2) == '0')
{
  if (t1 == 2) DRIVE2();
  GPGTPK(1, &DEPTH, PPATH);
  ID = PPATH[1];
  printf("IDx1 = %d",ID);
  if (t1 == 2) DRIVE2();
  if (DRIVE(ID) == 1) PPATH[1] = 4;
}
/* *************** EXIT OUT OF THE PROGRAM *************** */
} while (PPATH[1] != 4);
/* LOOUP UNTIL A BREAK OCCURRS */
GPMMSG(1,8,TMSG);
GPCLPH();
/* TERMINATION MESSAGE */
/* CLOSE graphigs */
/* ******************************************************* */
/*
/* This subroutine is used for the initial display of the conveyor rams */
/* and emergency lamp. */
/* ******************************************************* */

init(valx, valy, valz)
float valx, valy, valz;
{
    GPEST (3);
    GPDVR (1, 0, 3);
    GPEST (16);
    GPDVR (1, 0, 16);
    conv1(.1,.02,.5);
    Ram2();
    photo();
    emer1();
    GPUPWS (1, 2);
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
extern "C" {
    char name[4];
    int value;
    } X1[N][13];

struct X2
{
    int valu;
    } X3[N][13];

struct CNTER
{
    char name[3];
char reset0[3];
char reset1[3];
char reset2[3];
char reset3[3];
char reset4[3];
char reset5[3];
char reset6[3];
int cnt;
int count;
int enline;
int inline;
float TNOW1;
};
static struct CINTER STD[CC];
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* The program DRIVE.c is used to drive the various interaction modules */
/* depending on the outputs from the PLC file. This also calls the various */
/* subroutines to check whether the input states are consistent with the */
/* output */
/* */
/* include "def.c" */

char y2,y3,y4,y5,y6;
float valx, valy, valz;
int t6;

DRIVE(WW)
int WW;
{
    printf("PP= %d ",WW);
    err("IN DRIVE");
    /* t1=2; */
    sel_menu(WW);
    if (t1 == 1) return 1;
    DRIVE2();
    if (t1 == 1) return 1;
    if (t1 == 0)
    {
        return t1,1D;
    }
    if (t1 == 2)
    {
        return t1;
    }
    return t1;
}

/* The program selmenu is used for switching to the various functions */
/* depending on the inputs selected from the menu */
/* */
/* */

sel_menu(XXX)
int XXX;
{
    err("IN SEL MENU");
    switch(XXX)
    {
    case 0: /* IF EMERGENCY STOP BUTTON IS SELECTED */
    {
        rite(1);
        tri();
        FORT();
        vseek(1);
        if(vseek(1) == '1') /* IF EMERGENCY SWITCH IS STILL ON */
        {
            err("CHANGING THE BOOLEAN VALUE IN FILE");
            rite2(4,'0');
            rite2(2,'0');
            rite2(3,'0');
            tri();
        }
    /* CHANGING THE BOOLEAN VALUE IN FILE */
    /* UPDATE DATA STORED IN ARRAY STRUCTURE */
    }
FORT();
GPUPWS(1,2);

} else
   /* IF EMERGENCY SWITCH IS OFF */
   err("YOU MAY SELECT SYSTEM RESET BUTTON NOW");

} break;

case 1:
   /* IF SYSTEM RESET SWITCH IS SELECTED */
   if(vseek(1) == '1')
      /* CHECKS FOR EMERGENCY STOP STATE */
      err("STILL IN EMERGENCY STOP CONDITION");
      break;
   else
      /* IF NOT IN EMERGENCY STOP STATE */
      err("YOU MAY SELECT CONVEYOR FEED BUTTON NOW");
      rite2(2,'1'); /* CHANGE STATE VALUE FOR SYSTEM RESET SWITCH */
      tri();
      FORT();
      rite2(2,'0'); /* CHANGE STATE VALUE BACK TO OFF */
      tri(); /* SINCE IT IS A PULSED SWITCH */
      FORT();
      GPUPWS(1,2); /* UPDATE WORKSTATION */

} break;

case 2:
   {
      if(vseek(1) == '1')
         err("STILL IN EMERGENCY STOP CONDITION");
      else
      {
         if(X1[2][11].value==0)
            {
               err("SELECT SYSTEM RESET BEFORE CONVEYOR FEED BUTTON");
            }
         else
         {
            if(vseek(3) != '1')
            {
               rite2(3,'1'); /* CHANGE STATE VALUE FOR CON. FEED SWITCH */
               tri();
               FORT();
               err("YOU MAY MOVE NOW TO THE SYSTEM START STATE ");
            }
            else
            {
            }
         }
      }
   }
   break;

case 3:
   /* CONVEYOR FEED SWITCH */
   if(vseek(1) == '1')
      /* CHECK EMERGENCY STATE CONDITION */
      err("STILL IN EMERGENCY STOP CONDITION");
   else
   {
      if(X1[2][11].value==0)
      {
         err("SELECT SYSTEM RESET BEFORE SYSTEM START BUTTON");
      }
else
{
    if(vseek(3) == '0')
    {
        err("SELECT CONVEYOR FEED BUTTON BEFORE SYSTEM START BUTTON");
    }
    else
    {
        if (vseek(4) != '1')
        {
            rite2(4,'1');  /* CHANGE STATE VALUE FOR CON. FEED SWITCH */
            tri();
            FORT();
            err("YOU ARE IN SYSTEM START STATE");
        }
    }
}
break;
}
case 4:
{  
t1 =1;
    return t1;
}
break;
case 5:
{
    if(vseek(3) == '0')
    {
        err("CONVEYOR NOT ON");
    }
    else
    {
        rite2(7,'1');
        tri();
        FORT();
        GPUPWS(1,2);
    }
}
break;
case 6:
{  
    if(vseek(3) == '0')
        err("CONVEYOR NOT ON");
    else
    {
        rite2(8,'1');
        tri();
        FORT();
        GPUPWS(1,2);
    }
}
break;
case 8:
{
}
break;
}
GPSTLV(1,5,1);
return;
DRIVE2()
{
    photo();
    if (vtrg(5) == '1')
    {
        if ((vseek(1) == '1') && getca("y1") == 1)
        {
            Trace("y1");
            t1 = 1;
            err("SELECT STOP.SIM SWITCH");
            return t1;
        }
    }
    if (vtrg(8) == '1')
    {
        if ((vseek(1) == '1') && getca("y6") == 1)
        {
            Trace("y6");
            t1 = 1;
            err("SELECT STOP.SIM SWITCH");
            return t1;
        }
    }
    if(getca("y1") == 0)
    {
        /* err("init"); */
        init();
    }
    if(getca("y6") == 1)
    {
        emerg();
    }
    if(getca("y2") == 0 && getca("y3") == 1)
    {
        /* If output Y3 is ON RAM A is to be */
        /* retracted if it is extended or retain */
        rrama();
        /* the ram at the retracted condition */
    }
    if(getca("y4") == 0 && getca("y5") == 1)
    {
        /* retracted if it is extended or retain */
        rramb();
        /* the ram at the retracted condition */
    }
    if(getca("y1") == 1 && getca("y2") == 0 && getca("y4") == 0 &&
       vseek(7) == '0' && vseek(8) == '0')
    {
        /* If output Y1 is ON motor is ON. Hence */
        /* start moving the Conveyor. */
        /* err("IN DRIVE 2"); */
        if (vtrg(5) == '1')
        {
            if(getca("y1") == 1 && vseek(1) == '0')
            {
                Trace("y1");
            }
        }
    }
    mconv();
    return t1, ID,
if (getca("y2") == 1 && getca("y3") == 0 &&
((vseek(7) == '0') || (vseek(7) == '1' && CHK2(1) == 0))
&& ((vseek(8) == '0') || (vseek(8) == '1' && CHK2(2) == 0)))
  /* If output Y1 is ON motor is ON. Hence */
  /* start moving the Conveyor. */

  err("IN DRIVE 2");
  mram();
  if (vtrg(6) == '1')
  {
    Trace("y2");
    err("SELECT STOP.SIM SWITCH");
    t1 = 1;
    return t1;
  }

if (getca("y4") == 1 && getca("y5") == 0 &&
((vseek(7) == '0') || (vseek(7) == '1' && CHK2(1) == 0))
&& ((vseek(8) == '0') || (vseek(8) == '1' && CHK2(2) == 0)))
  /* If output Y1 is ON motor is ON. Hence */
  /* start moving the Conveyor. */

  err("IN DRIVE 2");
  mram();
  if (vtrg(7) == '1')
  {
    Trace("y4");
    err("SELECT STOP.SIM SWITCH");
    t1 = 1;
    return t1;
  }

if (getca("y1") == 1 && getca("y2") == 0 && getca("y4") == 0 &&
(vseek(7) == '1'))
  /* If output Y1 is ON motor is ON. Hence */
  /* start moving the Conveyor. */
  err("IN MBOX1");
  /* Trace("y1"); */
  mbox1();
  rite2(7, '0');
  if (vtrg(4) == '1')
  {
    t1 = 1;
    err("SELECT STOP.SIM SWITCH");
    return t1;
  }
  t1 = 2;
  return t1;
}

if (getca("y1") == 1 && getca("y2") == 0 && getca("y4") == 0 &&
(vseek(8) == '1'))
  /* If output Y1 is ON motor is ON. Hence */
start moving the Conveyor.

err("IN MBOX2");

mbox2();
rte2(8,'0');
if(vtrg(4)=='1') {
    t1 = 1;
    err("SELECT STOP.SIM SWITCH");
    return t1;
}

if(getca("y2")==1 && getca("y3")==1) {
    err("THE RETRACT AND EXTEND OUTPUTS FOR RAM A ARE ON. AMBIGUOUS LOGIC");
    Trace("y2");
    Trace("y3");
    t1=1;
    err("SELECT STOP.SIM SWITCH");
    return t1;
}

if(getca("y4")==1 && getca("y5")==1) {
    err("THE RETRACT AND EXTEND OUTPUTS FOR RAM B ARE ON. AMBIGUOUS LOGIC");
    Trace("y4");
    Trace("y5");
    t1=1;
    return t1;
}

return;
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ */
/* $ AUXILIARY PROGRAMS FOR CONVERTER SIMULATION */
/* $ */
/* $ */
/* $ */
/* $ */
/* $ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

/* **************************************** GLOBAL VARIABLES **************************************** */

#include "def.c"

FILE *in, *bool, *root, *trig;

/* **************************************** INTEGRAL VARIABLES **************************************** */

int i, j, k, L, r, s, k, p, q, Q, m, M, AJ, IC, IC, IC, SCAN, aq;

int LENGTH, ID1;

int DATAL;

int ASIZE[3], numb, STATUS, CI[100][12];

/* **************************************** CHARACTER VARIABLES **************************************** */

char TMSG[8];  // TERMINATION MESSAGE

char TEXT[1][5], TEXT[2][5], TEXT[3][6], TEXT[4][7], TEXT[5][8], TEXT[6][7], TEXT[7][8];

char TEXT[8][19], TEXT[9][19], TEXT[10][19], TEXT[11][19], TEXT[12][19], TEXT[14][19];

char TEXT[15][15], TEXT[16][13], TEXT[7][13], TVPI[6], TNAM[13], TEXT[41][3], TEXT[42][2];

char TEXT[43][4], TEXT[44][6], TNAMZ[8], TAND[2];

static char DATA[] = {""};

/* **************************************** FLOAT VARIABLES **************************************** */

float intensity, xl;

float PTL[9];

float ux, uy, AX, AY;

float P[7][8], x, y, z, l, w, h;

float XYZ[3];

float HIGH, MATRA[4][4], MAIRB[4][4], MATRC[4][4], MATR1[4][4], MATR2[4][4];

float CSIZE[3], a1, a2, a3;

float VALX, VALY, VALX, VALUE, C[3], R1[3];

float XY[3], CI[3], CVAL, TOTAL, HIGH;

float MIX[3];

float AREA[6], AREAL[6], ARE2[6], ARE3[6], ARE4[6];

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ */
/* $ COLOR TABLE VALUE DEFINITION */
/* $ */
/* $ */
/* $ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

static float AR[27] = {0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0};

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ */
/* $ DEFINITION OF POSITIONS FOR TEXT $ */
/* $ */
/* $ */
/* $ $ */
static float POS[2] = {0.71, 0.45}; /* POSITION E.STOP */
static float POS[2] = {-0.9, 0.85}; /* POSITION TITLE */
static float POS[2] = {0.73, 0.65}; /* POSITION MENU */
static float POS[2] = {0.25, 0.6}; /* POSITION PROMPT 1 */
static float POS[2] = {0.71, 0.25}; /* POSITION SYSTEM RESET */
static float POS[2] = {0.71, 0.05}; /* POSITION CON.FED */
static float POS[2] = {0.71, -0.14}; /* POSITION SYS.START */
static float POS[2] = {0.71, -0.34}; /* POSITION STOP */
static float POS[2] = {-0.9, -0.9}; /* POSITION MESSAGES */
static float POS[2] = {-0.25, 0.5}; /* POSITION PROMPT 2 */
static float POS[2] = {-0.25, -0.4}; /* POSITION PROMPT 3 */
static float POS[2] = {0.54, -0.73}; /* POSITION Feet... */
static float POS[2] = {0.54, -0.63}; /* POSITION DEMET */
static float POS[2] = {0.54, -0.73}; /* POSITION Krish--- */
static float POS[2] = {0.63, -0.68}; /* POSITION & */
static float POS[2] = {0.58, -0.89}; /* POSITION VPI & SU */
static float POS[2] = {-0.82, -0.5}; /* POS WITH GRAPHICS */
static float POS[2] = {-0.82, -0.55}; /* POS WITHOUT GRAPHICS */

/* $ */
/* $ DEFINITION OF VIEWPORT AND WINDOWS $ */
/* $ */
static float windl[4] = {0.0, 0.74, 0.0, 0.74};
static float wind2[4] = {0.0, 0.73, 0.12, 0.85, 0, 1};
static float wind3[4] = {0.0, 1.0, 0.0, 0.13};
static float wind[4] = {0.0, 1.0, 0.87, 1.0};
static float wind[4] = {0.0, 1.0, 0.87, 1.0};
static float wind[4] = {0.0, 0.75, 0.0, 0.1};
static float wind[4] = {0.0, 1.0, 0.0, 0.60};
static float wind[4] = {0.0, 0.15, 0.0, 0.60};
static float PT[3] = {0.00, 0.00, 0.60};

/* */
/* This subroutine helps to rotate the conveyor and other icons along */
/* X, Y, and Z axis for better visualization. */
/* */
/* */

ROT()
{
    valx = valx * 22.//7. / 360.;
    valy = valy * 22.//7. / 360.;
    valz = valz * 22.//7. / 360.;
    GPROTX(valx, MATRA);
    GPROXY(valy, MATRB);
    GPROTZ(valz, MTRC);
    GPCMT3(MATRA, MTRB, MATR1);
    GPCMT3(MATRA, MTRC, MATR2);
    GPVMT3(1, 1, MATR2);
    GDPVWS(1, 2);
}

/* */
/* CREATE TEXT REQUIRED FOR THE PROGRAM */
/* */
/* */
/* */
GPTXFO (11);
GPTX2 (POS1,31,TEXT1);
GPEF(2);
GPELT(1);
GPECI(1);
GPI(2);
GPICI(3);
GPTXFO (10);
GAPADCN(1,NAME1);
GPPKID (8);
GPPG2 (1,NP,2,box8);
GPTXCI(2);
GPTCHH (.026);
GPTX2 (POSN,13,TNAM);
GPTX2 (POSN2,6,TNAM2);
GPTX2 (FAND,2,TAND);
GPTCHH (.04);
GPTX2 (FOVP,6,TVPI);
GPEF(1);
GPCLIST();

/* Activate the font */
/* TEXT TITLE */
/* Edge flag on for the box */
/* SET COLOR INDEX */
/* SET COLOR INDEX */
/* Activate the font */
/* COLOR FOR TEXT */
/* set character height */
/* TEXT Kris-- */
/* TEXT Kris-- */
/* TEXT Kris-- */
/* set character height */
/* TEXT VPI & SU */
/* Close structure */

/***********************
CREATE MENU ITEMS AND OTHER SCREEN DISPLAYS
***********************

picitems2()
{
    static float box3[8] = [0.7,0.4,0.7,0.55,1.0,0.55,1.0,0.4];
    static float box4[8] = [0.7,0.2,0.7,0.35,1.0,0.35,1.0,0.2];
    static float box5[8] = [0.7,0.0,0.7,0.15,1.0,0.15,1.0,0.0];
    static float box6[8] = [0.7,-.2,0.7,-.05,1.0,-.05,1.0,-.2];
    static float box7[8] = [0.7,-.4,0.7,-.25,1.0,-.25,1.0,-.4];
    static float box8[8] = [0.51,-1.0,0.51,-.49,1.0,-.49,1.0,-1.0];
    GPOFST (1);
    GPTXFO (10);
    GPTXPR (2);
    GPTXCI (2);
    GPTCHH (.05);
    GPTX2 (PO2,5,TEXT2);
    GPEF(2);
    GPELT(1);
    GPECI(1);
    GPI(2);
    GPICI(3);
    GPTXFO (10);
    GPTCHH (.025);
    GPTXCI (1);
    GPI(2);
    GPAADCN(1,NAME1);
    GPPKID (0);
    GPPG2 (1,NP,2,box3);
    /* TEXT E.STOP */
    GPTX2 (POS,6,TEXT3);
    GPPKID (1);
    GPPG2 (1,NP,2,box4);
    /* TEXT SYSTEM RESET */
    GPTX2 (POS4,7,TEXT4);
    GPPKID (2);
    GPPG2 (1,NP,2,box5);
    /* TEXT CON. FEED */
    GPTX2 (POS5,8,TEXT5);
    GPPKID (3);
    GPPG2 (1,NP,2,box6);
}
TEXT()
{
    strcpy(TEXT1,"LADDER LOGIC DEBUGGING SOFTWARE"); /*Title*/
    strcpy(TEXT2,"MENU "); /* Display text */
    strcpy(TEXT3,"E.STOP"); /* Display text */
    strcpy(TEXT4,"S.RESET"); /* Display text */
    strcpy(TEXT5,"CON.FEED"); /* Display text */
    strcpy(TEXT6,"S.START"); /* Display text */
    strcpy(TEXT7,"STOP SIM"); /* Display text */
    strcpy(TEXT8,"Krishna Kumar"); /* Display text */
    strcpy(TEXT9,"&"); /* Display text */
    strcpy(TEXT10,"De Meter"); /* Display text */
    strcpy(TEXT11,"VPI&SU"); /* Display text */
    strcpy(TEXT12,"NEXT"); /* Display text */
    strcpy(TEXT13,"SCREEN"); /* Display text */
}

/* ***************************************************** */
/* ASSOCIATE VIEWS */
/* ***************************************************** */

ASOV()
{
    /**********
    GPVMP3 (1,1,wind1,V1,2,PT,0.0,100.0,-100.0);/*
    GPVCH(1,1,2,1,1,2,0,1,0,2); /*
    GPVMP2 (1,2,wind2,V2); /*
    GPVCH(1,2,2,1,1,2,3,2,127,2); /*
    GPVMP2 (1,3,wind5,V5); /*
    GPVCH(1,5,2,1,1,2,3,2,127,2); /*
    GPVMP2 (1,3,wind3,V3); /*
    GPVCH(1,3,2,1,1,2,3,2,127,2); /*
    GPVP(wsid,3,1,1); /*
    GPVP(wsid,0,1,1); /*
    GPVP(wsid,2,1,1); /*
    GPARP (1,0,1,1); /*
    GPARP (1,0,2,1); /*
    GPARP (1,0,40,1); /*
    GPARP (1,0,70,1); /*
    GPARP (1,0,80,1); /*
}

/* ***************************************************** */
/* CREATE MENU ITEMS AND OTHER SCREEN DISPLAYS */
/* ***************************************************** */
picitems1()
{
    static float box3[8] = {0.7,0.4,0.7,0.55,1.0,0.55,1.0,0.4};
    static float box4[8] = {0.7,0.2,0.7,0.35,1.0,0.35,1.0,0.2};
    static float box5[8] = {0.7,0.0,0.7,0.15,1.0,0.15,1.0,0.0};
    static float box6[8] = {0.7,-2.0,7,-0.05,1.0,0.05,1.0,-2};
    static float box7[8] = {0.7,-4.0,7,-25,1.0,-25,1.0,-4};
    static float box8[8] = {0.51,-1.0,0.51,-49,1.0,-49,1.0,-1.0};
    GPOPST (1); /* Open a structure */
    GPTXFO (10); /* Activate the font */
    GPTXPR (2); /* Set the text precision */
    GPCCH (.058); /* set character height */
    GPTXCI (2); /* COLOR FOR TEXT */
/* TEXT SYS. START */
GPTX2 (POS6,7,TEXT6);
GPPKID (4);
GPPG2 (1, NP, 2, box7);
GPTX2 (POS7,8,TEXT7);
GPEF(1);
GPCLST();
/* TEXT STOP SIM */
}
/* Close structure */

/**************************************************************************/
/* INITIALIZE STRING, VALUATOR AND PICK DEVICE */
/**************************************************************************/

ini_val()
{
  int errind, units;
  GPOADS (1,&errind,&units,csize,asize);
  GPOPEF(1,1,1,NAME1,0,DATA);
  GPOSTM(1,1,1,1,2); /* PUT STRING IN REQUEST MODE */
  GPOPDM(1,1,1,1,2); /* PUT PICK IN REQUEST MODE */
  GPOVML(1,1,1,1); /* PUT VALUATOR IN REQUEST MODE */

  /**************************************************************************/
  /* SET AREA FOR VALUATOR */
  /**************************************************************************/

  area3[0] = 0.3*csize[0];
  area3[1] = csize[0];
  area3[2] = 0.11*csize[1];
  area3[3] = csize[1];
  area3[4] = 0.;
  area3[5] = csize[2];
  GPINVL(1,1,valy,1,area3,0.0,360.0,DATALL); /* INITIALIZE VALUATOR */
  area3[0] = 0.3*csize[0];
  area3[1] = csize[0];
  area3[2] = 0.11*csize[1];
  area3[3] = csize[1];
  area3[4] = 0.;
  area3[5] = csize[2];
  GPINVL(1,2,valy,1,area3,0.0,360.0,DATALL); /* INITIALIZE VALUATOR */
  area3[0] = 0.3*csize[0];
  area3[1] = csize[0];
  area3[2] = 0.11*csize[1];
  area3[3] = csize[1];
  area3[4] = 0.;
  area3[5] = csize[2];
  GPINVL(1,3,valy,1,area3,0.0,360.0,DATALL); /* INITIALIZE VALUATOR */

  /**************************************************************************/
  /* SET AREA FOR PICK DEVICE */
  /**************************************************************************/

  area1[0] = 0.;
  area1[1] = csize[0];
  area1[2] = 0.;
  area1[3] = csize[1];
  area1[4] = 0.;
  area1[5] = csize[2];
  GPINPK(1,1,0,PPATH,1,area1,0,DATA,2); /* INITIALIZE PICK DEVICE */

  /**************************************************************************/
  /* SET AREA FOR STRING DEVICE */
  /**************************************************************************/

  area2[0] = 0.5*csize[0];
  area2[1] = csize[0];
area2[0] = 0.8*csize[1];
area2[1] = csize[1];
area2[2] = csize[2];
GPINST(1,1,0,DATA,1,area2,50,1,1,DATA); /* INITIALIZE STRING */
GPVLMO(1,1,3,1); /* PUT VALUATOR IN EVENT MODE */
GPVLMO(1,2,3,1); /* PUT VALUATOR IN EVENT MODE */
GPVLMO(1,3,3,1); /* PUT VALUATOR IN EVENT MODE */
GPFLSMO(1,1,3,2); /* PUT PICK IN EVENT MODE */

FUNCTION FOR ERASING RESULT STRUCTURES

erase()
{
    GPEST (6);
    GPDRV (1,0,6);
    GPEST (4);
    GPDRV (1,0,4);
    GPEST (8);
    GPDRV (1,0,8);
    GPEST (10);
    GPDRV (1,0,10);
    GPEST (11);
    GPDRV (1,0,11);
}

FUNCTION FOR DISPLAYING PROMPTS

err(L)
char L[];
{
    int LTEX;
    LTEX = strlen(L);
    GPEST(9);
    GPDRV (1,0,9);
    GPOPST(9);
    GPTXFO (10); /* ACTIVATE THE FONT */
    GTXCI (2); /* COLOR FOR TEXT */
    GPCIH (.03); /* SET CHARACTER HEIGHT */
    GPTX2 (POS8, LTEX, L);
    GPCLST();
    GEARV (1,0,9,1,1); /* ASSOCIATE STRUCTURE TO VIEW */
    GPUPWS (1,2); /* UPDATE WORKSTATION */
}

FUNCTION FOR DISPLAYING PROMPTS
erri(xx,yy,zz,L)
char L[];
float xx,yy,zz;        /* DEFINES POSITION FOR TEXT TO BE */
                 /* DISPLAYED AS MESSAGES */
{
    int LTEX;
    float POSE[2];
    POSE[0]=xx;
    POSE[1]=yy;
    LTEX = strlen(L);
    GPSPSST(20);
    GPTXFO (10);
    GPTXCI (2);        /* ACTIVATE THE FONT */
    GPCHH (zz);        /* COLOR FOR TEXT */
    GPTX2 (POSE,LTEX,L);
    GPCLST();
    GPARV (1,0,20,1.);        /* SET CHARACTER HEIGHT */
    GPUPWS (1,2);        /* TEXT MESSAGES */
    /* ASSOCIATE STRUCTURE TO VIEW */
    /* UPDATE WORKSTATION */
}

/* */
/* FUNCTION FOR RETAINING THE INPUT VALUES ON THE SCREEN */
/* */
/* */
/* */

retain(L,i)
char L[];
int i;
{
    int LT;
    float PO8[2];
    PO8[0] = .15;
    if (i == 1) PO8[1] = .6;
    if (i == 2) PO8[1] = .5;
    if (i == 3) PO8[1] = .4;
    LT = strlen(L);
    GPOPST(11);
    GPTXFO (10);
    GPTXCI (2);
    GPCHH (.04);
    GPTX2 (PO8,LT,L);
    GPCLST();
    GPARV (1,0,11,1.);        /* ASSOCIATE STRUCTURE TO VIEW */
    GPUPWS (1,2);        /* UPDATE WORKSTATION */
}

/* */
/* This program will create a box or cube like structure depending on the */
/* of inputs given. The inputs required are the x, y, z coordinates of the */
/* lower left edge point, the length of the box ( measured along the X-axis) */
/* the width of the box ( measured along the Y-axis ), and the height of the */
/* box ( measured along the Z-axis ). */
/* */
void box (x,y,z,l,w,h)
float x,y,z,l,w,h;
{
    static float point1[12];
    static float point2[12];
    static float point3[12];
    static float point4[12];
    static float point5[12];
    static float point6[12];
    point1[0] = x;
    point1[1] = y;
    point1[2] = z;
    point1[3] = x;
    point1[4] = y;
    point1[5] = z+h;
    point1[6] = x+l;
    point1[7] = y;
    point1[8] = z+h;
    point1[9] = x+l;
    point1[10] = y;
    point1[11] = z;
    point2[0] = x+l;
    point2[1] = y;
    point2[2] = z;
    point2[3] = x+l;
    point2[4] = y;
    point2[5] = z+h;
    point2[6] = x+l;
    point2[7] = y+w;
    point2[8] = z+h;
    point2[9] = x+l;
    point2[10] = y+w;
    point2[11] = z;
    point3[0] = x+l;
    point3[1] = y+w;
    point3[2] = z;
    point3[3] = x+l;
    point3[4] = y+w;
    point3[5] = z+h;
    point3[6] = x;
    point3[7] = y+w;
    point3[8] = z+h;
    point3[9] = x;
    point3[10] = y+w;
    point3[11] = z;
    point4[0] = x;
    point4[1] = y+w;
    point4[2] = z;
    point4[3] = x;
    point4[4] = y+w;
    point4[5] = z+h;
    point4[6] = x;
    point4[7] = y;
    point4[8] = z+h;
    point4[9] = x;
    point4[10] = y;
    point4[11] = z;
    point5[0] = x;
    point5[1] = y;
    point5[2] = z;
point5[3] = x+1;
point5[4] = y;
point5[5] = z;
point5[6] = x+1;
point5[7] = y+w;
point5[8] = z;
point5[9] = x;
point5[10] = y+w;
point5[11] = z;
point6[0] = x;
point6[1] = y;
point6[2] = z+h;
point6[3] = x+1;
point6[4] = y;
point6[5] = z+h;
point6[6] = x+1;
point6[7] = y+w;
point6[8] = z+h;
point6[9] = x;
point6[10] = y+w;
point6[11] = z+h;

GPIS(2):
GPPG3 (1,NP,3,point1);
GPPG3 (1,NP,3,point2);
GPPG3 (1,NP,3,point3);
GPPG3 (1,NP,3,point4);
GPPG3 (1,NP,3,point5);
GPPG3 (1,NP,3,point6);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* This program will create a box or cube like structure depending on the */
/* of inputs given. The inputs required are the x, y, z coordinates of the */
/* lower left edge point, the length of the box (measured along the X-axis) */
/* the width of the box (measured along the Y-axis), and the height of the */
/* box (measured along the Z-axis). */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

box2(u1,v1,u2,v2,u3,v3,u4,v4)
float u1,v1,u2,v2,u3,v3,v4;
{
float pt1[12],pt2[12],pt3[12],pt4[12],pt5[12],pt6[12];
pt1[0] = u1;
pt1[1] = v1;
pt1[2] = 0.;
pt1[3] = u2;
pt1[4] = v2;
pt1[5] = 0.;
pt1[6] = u3;
pt1[7] = v3;
pt1[8] = 0.;
pt1[9] = u4;
pt1[10] = v4;
pt1[11] = 0.;
pt2[0] = u2;
pt2[1] = v2;
pt2[2] = 0.;
pt2[3] = u3;
pt2[4] = v3;
pt2[5] = 0.0;
pt2[6] = u3;
pt2[7] = v3;
pt2[8] = 0.9;
pt2[9] = u2;
pt2[10] = v2;
pt2[11] = 0.5;
pt3[0] = u2;
pt3[1] = v2;
pt3[2] = 0.5;
pt3[3] = u3;
pt3[4] = v3;
pt3[5] = 0.5;
pt3[6] = u4;
pt3[7] = v4;
pt3[8] = 0.5;
pt3[9] = u1;
pt3[10] = v1;
pt3[11] = 0.5;
pt4[0] = u1;
pt4[1] = v1;
pt4[2] = 0.5;
pt4[3] = u4;
pt4[4] = v4;
pt4[5] = 0.5;
pt4[6] = u4;
pt4[7] = v4;
pt4[8] = 0.0;
pt4[9] = u1;
pt4[10] = v1;
pt4[11] = 0.0;
pt5[0] = u1;
pt5[1] = v1;
pt5[2] = 0.0;
pt5[3] = u2;
pt5[4] = v2;
pt5[5] = 0.0;
pt5[6] = u2;
pt5[7] = v2;
pt5[8] = 0.5;
pt5[9] = u1;
pt5[10] = v1;
pt5[11] = 0.5;
pt6[0] = u4;
pt6[1] = v4;
pt6[2] = 0.0;
pt6[3] = u3;
pt6[4] = v3;
pt6[5] = 0.0;
pt6[6] = u3;
pt6[7] = v3;
pt6[8] = 0.5;
pt6[9] = u4;
pt6[10] = v4;
pt6[11] = 0.5;
GPIS(2);
GP3G (1, NP, 3, pt1);
GP3G (1, NP, 3, pt2);
GP3G (1, NP, 3, pt3);
GP3G (1, NP, 3, pt4);
GP3G (1, NP, 3, pt5);
BOX()
{
    static float BOX[8] = {0.15, 0.6, 0.225, 0.25, 0.325, 0.6, 0.825, 0.95};
    static float BOX2[8] = {0.3, 0.6, 0.4, 0.6, 0.4, 0.65, 0.6, 0.65};
    GPOPST (2);
    GPARY (1,0.2,1.);
    GPADCN (1, NAME1);
    GPICI (4);
    GRIS (2);
    GPPKID(5);
    GPPG2 (1, NP, 2, BOX1);
    GPPKID(6);
    GPPG2 (1, NP, 2, BOX2);
    GPCLOT();
}

rite(JR)
long JR;
{
    char vr,ch;
    bool = fopen("bol","r");
    fseek(bool,JR,0);
    vr = getc(bool);
    fclose(bool);
    if (vr==0')
    {
        bool = fopen("bol","r+");
        fseek(bool,JR,0);
        ch = '1';
       putc(ch,bool);
        fclose(bool);
    }
    else
    {
        bool = fopen("bol","r+");
        fseek(bool,JR,0);
        ch = '0';
       putc(ch,bool);
        fclose(bool);
    }
}
rite2(JY,v)
long JY;
char v;
{
    char vr, ch;
    bool = fopen("bol","r+");
    fseek(bo1,JY,0);
    ch = v;
pus(ch,bool);
fclose(bool);
return;
}

rote2(JY,v)
long JY;
char v;
{
    char vr, ch;
    rort = fopen("ranti","r+");
    fseek(rort,JY,0);
    ch = v;
pus(ch,rort);
fclose(rort);
}

vseek(JR)
long JR;
{
    char vs;
    bool = fopen("bol","r");
    fseek(bool,JR,0);
    vs = getc(bool);
fclose(bool);
return vs;
}
vseekl(JR)
int JR;
{
    char vs;
    bool = fopen("bol","c");
    fseek(bool,JR,0);
    vs= getc(bool);
    fclose(bool);
    return vs;
}

getca(c)
char c[];
{
    int L;
    L = strlen(c);
    for(j=0;j<N;j++)
    {
        if(strcmp(c,X1[j][11].name)==0)
        {
            printf(" getca= \%d",X1[j][11].value);
            return X1[j][11].value;
        }
    }
}

setup1()
{
    static float BOX41[8] = {-.3,0.1,0.3,0.1,.3,0.4,-.3,0.4};
    static float BOX42[8] = {-.3,-.1,0.3,-.1,.3,-.4,-.3,-.4};
    static float BOX43[8] = {-.85,-.65,-.55,-.65,-.55,-.35,-.85,-.35};
    GPO3ST (40);
GPTXFO (10);                       /* Activate the font */
GPTXPR (2);                        /* Set the text precision */
GPEF (2);                          /* Edge flag on for the box */
GPELT(1);                          /* */
GPECI(1);                           /* */
GPIS(2);                           /* */
GPICI(2);                          /* */
GPTXCI (8);                         /* */
GPADCN (1,NAME1);                   /* */
GPCHH (.13);                        /* */
GPARV (1,0,40,1.);                  /* */
GPPKID(43);                         /**/
GPCHH (.03);                        /**/
GPPG2 (1,NP,2,BOX43);              /* TEXT NO */
GPTX2 (POS43,4,TEXT43);             /* TEXT NO */
GPTX2 (POS44,6,TEXT44);             /* */
GPCLST();                           /* */
GPUPWS(1,2);                        /* */
}

;/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* The subroutine setup1 is for the display of the menu screen and other */
/* graphical outputs */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

setup1()
{
    static float BOX41[8] = {-.3,.1,.3,.0,1,.3,0.4,-.3,0.4};
    static float BOX42[8] = {-.3,.1,.0,3,-.1,.3,-4,-.3,-4};
    static float BOX43[8] = {-.65,.65,-.55,-.65,-.55,.35,-.65,-.35};
    GPNOST (40);
    GPTXFO (10);                       /* Activate the font */
    GPTXPR (2);                        /* Set the text precision */
    GPEF (2);                          /* Edge flag on for the box */
    GPELT(1);                          /* */
    GPECI(1);                           /* */
    GPIS(2);                           /* */
    GPICI(3);                          /* */
    GPTXCI (2);                         /* */
    GPADCN (1,NAME1);                   /* */
    GPCHH (.13);                        /* */
    GPARV (1,0,40,1.);                  /* */
    GPIS(2);                           /* */
    GPPKID(41);                         /* */
    GPPG2 (1,NP,2,BOX41);              /* TEXT NO */
    GPPKID(42);                         /* */
    GPPG2 (1,NP,2,BOX42);              /* TEXT NO */
    GPPKID(43);                         /* */
    GPCHH (.03);                        /* */
    GPPG2 (1,NP,2,BOX43);              /* */
    GPTX2 (POS43,4,TEXT43);             /* */
    GPTX2 (POS44,6,TEXT44);             /* */
    GPCLST();                           /* */
    GPUPWS(1,2);                        /* */
}

;/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
Program TRI() for reading the PLC program and initial debugging

This program is used to read any PLC program specified in the format defined in the Chapter 4 of the thesis. It performs the initial debugging for format errors as well.

#include "def.c"

************** DEFINITION OF INTEGER VARIABLES **************

int i, ii, iii, I, j, J, K, k, l, L, m, M, r, R, p, s;
long val, val1, val3;

************** POINTERS TO FILE **************

FILE *in, *bool, *RES;

**************

tri()
{

STD[0].name[0] = 'S';
STD[0].name[1] = '1';
STD[0].cnt = 14;
STD[0].reset[0] = 'X';
STD[0].reset[1] = '4';
STD[0].reset[2] = 'C';
STD[0].reset[3] = '3';
STD[0].reset[4] = '0';
STD[0].reset[5] = 'I';
STD[0].reset[6] = '0';
STD[0].enln = 0;
STD[0].inln = 0;

STD[1].name[0] = 'S';
STD[1].name[1] = '2';
STD[1].cnt = 20;
STD[1].reset[0] = 'X';
STD[1].reset[1] = '4';
STD[1].reset[3] = '1';
STD[1].reset[4] = '0';
STD[1].reset[5] = '0';
STD[1].reset[6] = '0';
STD[1].enln = 0;
STD[1].inln=0;
STD[2].name[0]=’T’;
STD[2].name[1]=’3’;
STD[2].cnt=4;
STD[2].reset0[0]=’x’;
STD[2].reset0[1]=’4’;
STD[2].reset1[0]=’c’;
STD[2].reset1[1]=’1’;
STD[2].reset2[0]=’0’;
STD[2].reset2[1]=’0’;
STD[2].reset4[0]=’0’;
STD[2].reset5[0]=’0’;
STD[2].reset6[0]=’0’;
STD[2].TNOW1 = 1;
STD[2].enln=0;
STD[2].inln=0;
STD[2].name[0]=’S’;
STD[2].name[1]=’4’;
STD[2].cnt=2;
STD[2].reset0[0]=’y’;
STD[2].reset0[1]=’3’;
STD[2].reset1[0]=’y’;
STD[2].reset1[1]=’5’;
STD[2].reset2[0]=’0’;
STD[2].reset3[0]=’0’;
STD[2].reset4[0]=’0’;
STD[2].reset5[0]=’0’;
STD[2].reset6[0]=’0’;
STD[2].enln=0;
STD[2].inln=0;

in = fopen("ptr", "r"); /* Reading the PLC FILE */

for(j=0; j<N;j++)
{
  K=11;
  for(l=0; l<13;l++)
  {
    fscanf(in,"%s",X1[j][l].name); /* Scanning ptr file */
    if(l<11)
      {  
        if(X1[j][l].name[0]==’X’||X1[j][l].name[0]==’n’)
          GetB();
        else if (X1[j][l].name[0]==’S’)
          COUNTER();
        else if (X1[j][l].name[0]==’T’)
          TIMER();
        else if (X1[j][l].name[0]==’o’)
          K = 1;
        else if (X1[j][l].name[0]==’c’||X1[j][l].name[0]==’y’||X1[j][l].name[0]==’C’)
          {
            if(( GETCY()==1))
              {  
                fclose(in);
                return 1;
              }
          }
      }
  }
  if(l==11)
  {
    
  }
}
if (coilv() == 1)
{
    fclose(in);
    return 1;
}

fclose(in);

/* if(COMP() == 2) tri(); */
printf("LEAVE TRI COUNTVALUE \d \d ", STD[0].count, STD[1].count);
return X1[N-1][1].value, X1[N-2][1].value, STD[0].count, STD[1].count;

/* ********************************************* */
/*
/* This function is used to get the boolean value of the switch from the    */
/* file "bol" corresponding to the number of the switch                */
/* ********************************************* */

GetB()
{
    char copy1[], vl;
    l = strlen(X1[j][i].name);
    l = 0;
    for (l=1; l < I+1;++l)
    {
        copy1[l] = X1[j][i].name[I];
        ++l;
    }
    val = atoi(copy1);
    bool = fopen("bol","r");
    fseek(bool, val, 0);
    vl = fgetc(bool);
    if (vl == 'o')
        X1[j][i].value = 0;
    else
        X1[j][i].value = 1;
    fclose(bool);
    return;
}

/* ********************************************* */
/*
/* This function is used to assign the value of the coil to the input coil */
/* from the coil value previously assigned as output                    */
/* ********************************************* */

GETCY()
{
    printf("IN GETCY");
    if(X1[j][i].name[0]=='C' || X1[j][i].name[0]=='c')
    {
        for(r=0; r<j+1; r++)
        {
            L = 1;
        }
        for(r=0;
do {
    if(X1[r][11].name[0] != 'y')
    {
        if(X1[r][11].name[1]==X1[j][i].name[1])
        {
            if(L==2)
            {
                X1[j][i].value=X1[r][11].value;
                return;
            }
            L++;                
        }
        else
        L =3;
    }
} while(L < 3);
if(r==j+1)
{
    RES = fopen("RESULT","a");
    fprintf(RES,"The output \%s \%d \%d is not referenced before ",X1[j][i].name,j,r);
    fclose(RES);
    return 1;
}
}
for(r=0;r<j+1;r++)
{
    if(strcmp(X1[r][11].name,X1[j][i].name)==0)
    {
        X1[j][i].value=X1[r][11].value;
        return;
    }
    else if(r==j+1)
    {
        RES = fopen("RESULT","a");
        fprintf(RES,"The output \%s \%d \%d is not referenced before ",X1[j][i].name,j,r);
        fclose(RES);
        return 1;
    }
}

/*******************************************************************************/
/*
/* This function is used to get the boolean value of the switch from the */
/* file "bol" corresponding to the number of the switch */
/*
/*******************************************************************************/

coilv()
{
    for(m=0;m<11;m++)
    {
        if ((X1[j][m].name[0]=='x') || (X1[j][m].name[0]=='n') || (X1[j][m].name[0]=='0')
            || (X1[j][m].name[0]=='c') || (X1[j][m].name[0]=='s') || (X1[j][m].name[0]=='d')
            || (X1[j][m].name[0]=='y') || (X1[j][m].name[0]=='o') || (X1[j][m].name[0]=='t')
            || (X1[j][m].name[0]=='c'))
            else
            {
                RES = fopen("RESULT","a");
            }
fprintf (RES, "There is an unknown character %s in the ptr file on line %d ",
X1[j][i].name, j);
    fclose (RES);
    return 1;
}
if (m==10)
    {
    if (DIRV()==1) return 1;
    return;
}
}

/* ***************************************************/
/* */
/* This function is used to get the coil value for the output by checking */
/* values for the inputs */
/* */
/* ***************************************************/

DIRV()
{
    for (R=0;R<N;R++)
    {
        if (strcmp (X1[R][11].name, X1[j][11].name)==0 && R != j)
        {
            RES = fopen("RESULT", "a");
            fprintf (RES, "The coil %s is referenced as outputs in lines %d and %d ",
X1[j][11].name, R, j);
            fclose (RES);
            return 1;
        }
    }
    for (I=0;I<11;I++)
    {
        CHKUM(j,I);
        if (((X1[j][I].name[0]==’x’ && X1[j][I].value==1) && (X1[j][I].name[0]==’n’ &&
X1[j][I].value==0) || (X1[j][I].name[0]==’0’) || (X1[j][I].name[0]==’S’ &&
CHK2 (val1)==1)) ||
        ((X1[j][I].name[0]==’y’ && X1[j][I].value==1) ||
        (X1[j][I].name[0]==’C’ && X1[j][I].value==0)) ||
        (X1[j][I].name[0]==’S’ && CHK2 (val1)==1 ) ||
        (X1[j][I].name[0]==’D’ && CHK2 (val1)==1 ))
        {
        if (I==K-1)
        {
            if (X1[j][11].name[0] == ’y’ || X1[j][11].name[0] == ’c’ )
            {
            X1[j][11].value=1;
            printf("%s %s 1 c1\d ",X1[j][11].name, X1[j][11].value);
            return;
            }
        }
    }
    else if (K==11)
    {
        for (r=K+1;r<11;r++)
        {
        if (((X1[j][r].name[0]==’x’ && X1[j][r].value==1) ||


X1[j][r].value==0) || (X1[j][r].name[0]=='0') || (X1[j][r].name[0] == 'S' && CHK2(val1)==1) ||
(X1[j][r].name[0]=='c'&&X1[j][r].value==1) ||
(X1[j][r].name[0]=='y'&&X1[j][r].value==1) ||
(X1[j][r].name[0]=='C'&&X1[j][r].value==0) ||
(X1[j][r].name[0]=='S'&&CHK2(val1)==1)
|| (X1[j][i].name[0]=='D' && CHK2(val1)==1 )
{
    if(r==10)
    {
        if(X1[j][11].name[0] == 'y' || X1[j][11].name[0] == 'c' )
        {
            X1[j][11].value=1;
            return;
        }
    }
    else
    {
        X1[j][11].value=0;
        return;
    }
}
else
{
    X1[j][11].value=0;
    return;
}
}

/ * *************************************************************** */
/ *
/ * This function is used to get the numerical part of the switch name so as */
/ * to find from the file "bol" its state value */
/ *
/ * *************************************************************** */

CNUM(j,i)
int j,i;
{
    char copy2[1];
    J=strlen(X1[j][i].name);
    l=0;
    for(L=1;L<J+1;L++)
    {
        copy2[l]=X1[j][i].name[L];
        l++;
    }
    val1 = atoi(copy2);
    return val1;
}

/ * *************************************************************** */
/ *
/ * This function is for the counter logic representation */
/ *
/ * *************************************************************** */
COUNTER()
{
    R=0;
    CNUM(j,i);
    ENABLE1(val1);

    if (enbl(val1)==0)
    {
        STD[val1-1].count = 0;
        return;
    }

    if (inline(val1)==1)
    {
        if (STD[val1-1].inline==0)
        {
            STD[val1-1].count++;
            STD[val1-1].inline=1;
            return STD[val1-1].count;
        }
        else if (inline(val1)==0)
        {
            STD[val1-1].inline=0;
            return;
        }
    }

    /* ***************************************************/
    /* */
    /* This function is for the down counter */
    /* */
    /* ***************************************************/

    CDOWN()
    {
    }

    /* CDOWN() */
    {
    R=0;
    CNUM(j,i);
    printf("val5 \%d ",val1);
    for(I=0;I<i;I++)
    {
        printf("\%d ",val1,1,Cl[val1][I]);
        Cl[val1+5][I] = Cl[val1][I];
        printf("\%d ",I,Cl[val1+5][I]);
        if(X1[j][I].value==Cl[val1+5][I])
            printf("good COUNTVALUE\%d ",COUNT[val1]);
        else
        {
            Cl[val1][I]=X1[j][I].value;
            printf("THIS IS 2nd Cl[%d][%d]= \%d ",val1,I,Cl[val1][I];
            Cl[val1+5][I] = Cl[val1][I];
            R++;
        }
    }
    }
    if(R>0)
for (m=0; m<i; m++)
{
    C1[vall][m] = X1[j][m].value;
    printf("m C1[vall][%d] = %d ", m, C1[vall+5][m]);
}
for (M=0; M<m; M++)
{
    printf("exec cdown %d ", vall);
    if((X1[j][M].name[0] == 'x' && X1[j][M].value == 1) ||
       (X1[j][M].name[0] == 'n' &&
        X1[j][M].value == 0) ||
       (X1[j][M].name[0] == 'c' && X1[j][M].value == 1) ||
       (X1[j][M].name[0] == 'y' && X1[j][M].value == 1) ||
       (X1[j][M].name[0] == 'C' && X1[j][M].value == 0))
    {
        if (M==10)
        {
            CNT[vall]--; 
            if (CCHK[vall] == 1)
            {
                X1[j][M].value = 1;
                printf("COUNTVALUE%d ", COUNT[vall])
            } 
            R=0;
            return CNT[vall];
        }
    }
    else
    {
        printf("RET 1 CNTVALUE%d ", CNT[vall]);
        R=0;
        return CNT[vall];
    }
}
/*
* *********************************************** *
*/
/* This function is used for the checking of the counter value
*/
/* *********************************************** */

CCHK(v6)
long v6;
{
    if ( STD[v6-1].count == 0)
    {
        err("CCHK 1");
        return 1;
    }
    else
    {
        err("CCHK 0");
        return 0;
    }
}
/* *********************************************** */
/*
 * This function is used for the checking of timer value
 */

CCHK3(v6)
long v6;
{
    if (STD[v6-1].TNOW1 == 0)
    {
        err("CCHK 1");
        return 1;
    }
    else
    {
        err("CCHK 0");
        return 0;
    }
}

/* 
 * This function is for the counter
 */

CCHK2(v7)
long v7;
{
    if (STD[v7-1].cnt == STD[v7-1].count)
    {
        err("CCHK 3");
        return 1;
    }
    else
    {
        err("CCHK 4");
        return 0;
    }
}

/* 
 * This function is used define the logic for the TIMER.
 */

TIMER()
{
    R=0;
    CNUM(j,i);
    ENABLE1(vall);
    if (enbl(vall)==0)
    {
        printf("IN TIMER 3");
        STD[vall-1].TNOW1 = STD[vall-1].cnt;
        return;
    }
if (inline(vall)==1)
{
    printf("IN TIMER 4 inline =1\n");
    FORT();
    return STD[vall-1].TNOW1;
}
else if(inline(vall)==0)
{
    STD[vall-1].inln=0;
    return;
}

/*==============================================================================
/*
/* This function is used for checking the enable line for the counter and timer */
/*
/*==============================================================================

ENABLE1(vv)
long vv;
{
    for (ss=0;ss<7;ss++)
        {
            if (enable(vv,ss) == 1)
                {
                    if(ss==6)
                        {
                            STD[vv-1].enln=1;
                            return;
                        }
                    else
                        {
                            STD[vv-1].count=0;
                            STD[vv-1].enln=0;
                            return;
                        }
                }
        }
}

/*==============================================================================
/*
/* This function is used for the counter and timer for enable input line */
/*==============================================================================

int enable(iiiii,iii)
int iii;
long iiiii;
{
    static char UUU[];
    if(iii==0)
        {
            UUU[0]=STD[iiiii-1].reset0[0];
        }
UUU[1] = STD[iiii-1].reset0[1];
UUU[2] = STD[iiii-1].reset0[2];
}
if (iii==1)
{
    UUU[0] = STD[iiii-1].reset1[0];
    UUU[1] = STD[iiii-1].reset1[1];
}
if (iii==2)
{
    UUU[0] = STD[iiii-1].reset2[0];
    UUU[1] = STD[iiii-1].reset2[1];
}
if (iii==3)
{
    UUU[0] = STD[iiii-1].reset3[0];
    UUU[1] = STD[iiii-1].reset3[1];
}
if (iii==4)
{
    UUU[0] = STD[iiii-1].reset4[0];
    UUU[1] = STD[iiii-1].reset4[1];
}
if (iii==5)
{
    UUU[0] = STD[iiii-1].reset5[0];
    UUU[1] = STD[iiii-1].reset5[1];
}
if (iii==6)
{
    UUU[0] = STD[iiii-1].reset6[0];
    UUU[1] = STD[iiii-1].reset6[1];
}
if (UUU[1]=='c' || UUU[0]=='c' || UUU[0]=='C' || UUU[0]=='y')
{
    if (encol(UUU) == 1)
    {
        return 1;
    }
}
if (UUU[1]=='x' || UUU[0]=='x' || UUU[0]=='n')
{
    if (getbc(UUU) == 1)
    {
        return 1;
    }
}
if (UUU[1]=='c' || UUU[0]=='c' || UUU[0]=='C' || UUU[0]=='y')
{
    if (encol(UUU) == 1)
    {
        return 1;
    }
}
if (UUU[0]=='0')
{
    return 1;
}
return 0;
enbl(sss)
int sss;
{
    /* check the data array for matching counter and then return the value which is given in the enline */
    if(STD[sss-1].enline==1)
        return 1;
    else
    {
        STD[sss-1].count = 0;
        return 0;
    }
}

int getbc(ttt)
char ttt[];
{
    char cpy[],v2;
    I = strlen(ttt);
    l = 0;
    for(L=1;L < I+1;++L)
    {
        cpy[l]=ttt[L];
        ++l;
    }
    val3 = atoi(cpy);
    bool = fopen("bol","r");
    fseek(bool,val3,0);
    v2 = getc(bool);
    if (v2 =='0' && ttt[0]=='x')
        || (v2=='1' && ttt[0]=='n'))
    {
        fclose(bool);
        return 0;
    }
    else
    {
        fclose(bool);
        return 1;
    }
}
int encoile(tttt)
    char tttt[];
    {
        for(r=0; r<N; r++)
        {
            L = 1;
            do {
                if(tttt[0] == 'c' || tttt[0] == 'C')
                {
                    if(X1[r][11].name[L] == tttt[L])
                    {
                        if(L == 2)
                        {
                            if ((X1[r][11].value== 0 && tttt[0] == 'C')
                                || (X1[r][11].value== 1 && tttt[0] == 'C'))
                            {
                                return 1;
                            }
                            else if((X1[r][11].value== 1 && tttt[0] == 'C')
                                || (X1[r][11].value== 0 && tttt[0] == 'c'))
                            {
                                return 0;
                            }
                        }
                        L++;
                    }
                    else
                        L = 3;
                }
                if(tttt[0] == 'y')
                {
                    if(X1[r][11].name[L] == tttt[L])
                    {
                        if(L == 2)
                        {
                            if (X1[r][11].value==0 && tttt[0] == 'y')
                                return 0;
                            else
                                return 1;
                        }
                        L++;
                    }
                    else
                        L = 3;
                }
            }while(L < 3);
        }
        printf("The enable input %s is not referenced before ", tttt);
        return 0;
    }
```c
/* *******************************************/
/* This function is used for the input line to the counter and timer */
/* *******************************************/

inline(liii)
long liii;
{
    iiii=liii-1;
    for(m=0;m<11;m++)
    {
        if ((X1[i][j].name[0]=='x') || (X1[i][j].name[0]=='n') || (X1[i][j].name[0]=='0')
            || (X1[i][j].name[0]=='c') || (X1[i][j].name[0]=='s') || (X1[i][j].name[0]=='d')
            || (X1[i][j].name[0]=='t')
            || (X1[i][j].name[0]=='y') || (X1[i][j].name[0]=='o') || (X1[i][j].name[0]=='c'))
            else
            {
                RES = fopen("RESULT", "a");
                fprintf(RES,"There is an unknown character \%s in the ptr file on line \%d ",
                        X1[i][j].name,j);
                fclose(RES);
                return 1;
            }
        if(m==10)
        {
            if (line(liii)=1) return 1;
            return 0;
        }
    }
}

/* *******************************************/
/* This function is used to check the complete input line for the counter */
/* and timer */
/* *******************************************/

line(liii)
long liii;
{
    for(I=0;I<11;I++)
    {
        if(((X1[i][j].name[0]=='x'&&X1[i][j].name[0]=='n'&&
            X1[i][j].name[0]=='0') || (X1[i][j].name[0]=='s') || (X1[i][j].name[0]=='d')
            || (X1[i][j].name[0]=='t')
            || (X1[i][j].name[0]=='y') || (X1[i][j].name[0]=='o') || (X1[i][j].name[0]=='c'))
            else
            {
                if(X1[i][j].name[0]=='s' || X1[i][j].name[0]=='t')
                {
                    return 1;
                }
            }
    }
    else
    {
    }
```
return 0;

/* *****************************/
/* This function is used for the continuous updating of the timer */
/* when it is activated */
/* *****************************/
FORT()
{
    for (iii = 0; iii < CC; iii++)
    {
        if (STD[iii].name[0] == 'T')
        {
            if (enbl[iii+1] == 1)
            {
                if (inline(iii+1) == 1)
                {
                    STD[iii].TNOW1 = STD[iii].TNOW1 - .5;
                    /* printf("in I val 1 -%d X= %d ",vall,K); */
                    if (STD[iii].TNOW1 < 0.)
                        STD[iii].TNOW1 = 0.;
                }
                else
                {
                    STD[iii].TNOW1 = STD[iii].cnt;
                }
            }
        }
    }

    /* *****************************/
    /* This function is used for checking if the values of the coil outputs */
    /* are completely updated for the inputs */
    /* *****************************/
    COMP()
    {
        for (ii = 0; ii < N; ii++)
        {
            if (X1[iii][ii].value != X3[ii][ii].valu)
            {
                X3[ii][ii].valu = X1[ii][ii].value;
                return 2;
            }
        }
    }

    /******************************/
121
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ */
/* $ GRAPHICS OR NO GRAPHICS */
/* $ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

#include "def.c"

FILE *rort;

/* **************************************** CHARACTER VARIABLES **************************************** */

char TEXT43[4], TEXT44[6];

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $ */
/* $ DEFINITION OF POSITIONED FOR TEXT */
/* $ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

static float POS51[2] = { -2.2 }; /* POS WITH GRAPHICS */
static float POS52[2] = { -2.3 }; /* POS WITHOUT GRAPHICS */
static float POS53[2] = { -8.5 }; /* POS WITHOUT GRAPHICS */
static float POS54[2] = { -82.55 }; /* POS WITHOUT GRAPHICS */

void grap() {
    err("DETACH THE GRAPHICS MODULE?");
    setup2(); /* SETS UP INITIAL DISPLAY SCREEN */
    disp6();
    GUPWS(1, 2);
    do {
        GPPKF(1, 1, incl1, excl1); /* PUT PICK FILTER */
        GPAWEV(110.1, &wsid, &clas, &devid); /* AWAITS EVENT */
        if (clas == 5)
            sel_gra();
        GPUPWA(1);
        GFLE(1, 1.5, 1);
    } while (PPATH[1] != 43);
    GPEST(70);
    GPDRV(1, 0, 70);
    GPEST(20);
    GPDRV(1, 0, 20);
}

/* ****************************************** */
/* FUNCTION FOR SELECTING MENU ITEMS AND SWITCHING TO APPROPRIATE FUNCTIONS */
/* ****************************************** */

sel_gra() {
    if (clas == 5)
    {
        GPGTPK(1, &DEPTH, PPATH);
    }
ID = PPATH[1];
printf("PP= %d ", ID);
GFLEVE(1,5,1);
switch(ID)
{
    case 41: /* IF EMERGENCY STOP BUTTON IS SELECTED */
    {
        rot(11); /* CHANGING THE BOOLEAN VALUE IN FILE */
        rot(12);
        if(vrot(11) == '1')
            err("GRAPHICS MODULE IS ATTACHED");
        else
            err("GRAPHICS MODULE NOT ATTACHED");
    }
    break;
    case 42: /* IF EMERGENCY STOP BUTTON IS SELECTED */
    {
        rot(11);
        rot(12); /* CHANGING THE BOOLEAN VALUE IN FILE */
        if(vrot(12) == '1')
            err("NO GRAPHICS MODULE ATTACHED");
        else
            err("GRAPHICS MODULE IS ATTACHED");
    }
    break;
    }
GFLEVE(1,5,1);
}
GFLEVE(1,5,1);

disp6()
{
    errl(-.215,.315,.04," ATTACH");
    errl(-.225,.195,.04," GRAPHICS");
    errl(-.225,-.205,.04,"DETACH");
    errl(-.225,-.325,.04,"GRAPHICS");
    GFUPWS(1,2);
}
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
# include "def.c"

FILE *rort;

char TEXT41[3], TEXT42[2];
char TEXT43[4], TEXT44[6];

static float POS51[2] = { -2., 2.};  /* POS WITH GRAPHICS */
static float POS52[2] = { -2., -3.};  /* POS WITHOUT GRAPHICS */
static float POS54[2] = { -82., -55.}; /* POS WITHOUT GRAPHICS */

void inact()
{
  err("INTERACTIVE OR NON-INTERACTIVE MODE?");
  setup5();  /* SETS UP INITIAL DISPLAY SCREEN */
  disp2();
  GPUPWS(1,2);
  do
    { GPFKF(1,1,1,xcll,1,excll);  /* PUT PICK FILTERS */
      GPAREV(110.1, &wsid, &clas, &devid); /* AWAIT EVENT */
      if (clas == 5)
        sel_iact();
      GPUPWA (1);
      GPPLEV(1,5,1);  /* UPDATE WORKSTATION */
    } while (FPATH[1] != 43);
  GDPRET(80);
  GDPREV(1,0,80);
  GDPREV (20);
  GDPREV (1,0,20);
}

/* $$$$$$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* The subroutine setup5() provides graphics for the triggers to be setup */
/* allowing the user to decide which interactive modules should be setup as */
/* trigger conditions */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

setup5()
{
  static float BOX81[8] = { -3.0, 0.0, 0.3, 0.10, 0.30, 0.40, -0.30, 0.40};
  static float BOX82[8] = { -3., -0.3, 0.10, 0.03, -0.30, 0.03, -0.30, -0.40};
static float BOX43[6]= (-.85,-.65,-.55,-.65,-.55,-.35,-.85,-.35);
GPOPST (80);
GPTXFO (10); /* Activate the font */
GPTXPR (2); /* Set the text precision */
GPELT (1);
GPECI (1); /* SET COLOR INDEX */
GPIS (2);
GPICI (3); /* SET COLOR INDEX */
GPTXCI (2); /* COLOR FOR TEXT */
GPADCN (1, NAME1); /* PUT CLASS NAME */
GPCHH (.13); /* set character height */
GPARV (1,0,50,1.); /* Associate structure to view */
GPIS (2);
GPKID (81);
GPPG2 (1, NP, 2, BOX81);
GPKID (82);
GPPG2 (1, NP, 2, BOX82);
GPKID (43);
GPCHH (.03); /* set character height */
GPPG2 (1, NP, 2, BOX43);
GPTX2 (POS33,4,TEXT43); /* TEXT NO */
GPTX2 (POS54,6,TEXT44); /* TEXT NC */
GPCLST();
GPUPWS (1,2);
}
disp2 ()
{
  err1(-.25,.315,.025," INTERACTIVE");
  err1(-.25,.195,.025," MODE");
  err1(-.275,-.205,.025," NON-INTERACTIVE");
  err1(-.25,-.325,.025," MODE");
  GPUPWS (1,2);
}

/* *************************************************************/
/* */
/* FUNCTION FOR SELECTING MENU ITEMS AND SWITCHING TO APPROPRIATE FUNCTIONS */
/* */
/* *************************************************************/

sel_lact()
{
  if (clas == 5)
  {
    GPGTPK (1, &DEPTH, PPATH);
    ID = PPATH[1];
    printf("PP= %d ", ID);
    GPFLEV (1.5, 1);
    switch (ID)
      {
      case 81: /* IF EMERGENCY STOP BUTTON IS SELECTED */
      {
    rot (6); /* CHANGING THE BOOLEAN VALUE IN FILE */
    rot (7);
    if (vrot(6) == '1')
      err("YOU HAVE SELECTED INTERACTIVE MODE");
    else
      err("YOU ARE IN NON-INTERACTIVE MODE");
      }
break;
case 82: /* IF EMERGENCY STOP BUTTON IS SELECTED */
{
    rot(6); /* CHANGING THE BOOLEAN VALUE IN FILE */
    rot(7); /* CHANGING THE BOOLEAN VALUE IN FILE */
    if(vrot(7) == '1')
        err("YOU HAVE SELECTED NON-INTERACTIVE MODE");
    else
        err("YOU ARE IN INTERACTIVE MODE");
} break;
}
GPFLEV(1.5,1);
}
GPFLEV(1.5,1);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* The subroutine rainbl is the interaction module for demonstrating the */
/* interaction between Ram A and Box 1. */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

#include "def.c"

rainbl()
{

int i;

static float A = -0.114;
static float B = .1515;
static float C = 0.5;
static float D = .05;
static float E = .05;
static float F = 1.1;

/* Draw the A1's first, then draw the A2 box and then the A box */

GPEST(20);
GPDRV(1,1,20);
    conv1(.1,.02,.5);
    photo();
for(i=0;i<5;i++)
{
    err("in rainbl");
    GPEST(22);
    GPDRV(1,1,22);
    GPEST(16);
    GPDRV(1,1,16);
    GPST(22);
    GPARV(1,1,22,1.);
    GPIS(2);
    GPLIC(8);
    box(A,B,C-(i*.05),D,E,F);
    GPLIC(4);
    box(-.139,.1462,0.25,.075,.05,.05);
    GPCLST();
    GUPWS(1,2);
}

for(i=0;i<7;i++)
{
    GPEST(22);
    GPDRV(1,1,22);
    GPEST(16);
    GPDRV(1,1,16);
    GPST(22);
    GPARV(1,1,22,1.);
    GPIS(2);
    GPLIC(8);
    box(A,.25-(i*.05),D,E,F);
    GPCLST();
    GPST(16);
    GPARV(1,1,16,1.);
    GPIS(2);
    GPLIC(4);
    box(-.139,.1462,0.2-(i*.05),.075,.05,.05);
    GPCLST();
}
GPUPWS(1,2);
}

for (i=0; i<29; i++)
{
    GPEST(16);
    GPDRV(1,1,16);
    GPEST(16);
    GPARV(1,1,16,1.);
    GPIS(2);
    GPICI(4);
    box(-.139,.1462-.01+i,-.1-.01*i,.075,.05,.05);
    GPCLST();
    GPUPWS(1,2);
}
GPEST(16);
GPDRV(1,1,16);
GPUPWS(1,2);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
*/
/* The subroutine rainb2 is the interaction module for demonstrating the */
/* interaction between Ram A and Box 2. */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

rainb2()
{

    int i;

    static float A = -0.144;
    static float B = .1515;
    static float C = 0.5;
    static float D = .05;
    static float E = .05;
    static float F = 1.0;
    /* Draw the A1's first, then draw the A2 box and then the A box */
    GPEST(20);
    GPDRV(1,1,20);
    convl(.1,.02,.5);
    phot();
    for(i=0; i<5; i++)
    {
        err("in rainb2");
        GPEST(22);
        GPDRV(1,1,22);
        GPEST(16);
        GPDRV(1,1,16);
        GPEST(22);
        GPDRV(1,1,22,1.);
        GPICI(8);
        box(A,B,C-(i*.05),D,E,F);
        GPICI(4);
        box(-.139,.1462,.25,.1,.05,.05);
        GPCLST();
        GPUPWS(1,2);
}
for(i=0;i<7;i++)
{
  err("in rainb2");
  GPEST(22);
  GPDVRV(1,1,22);
  GPEST(16);
  GPDVRV(1,1,16);
  GPOPST(22);
  GPARV(1,1,22,1.);
  GPICI(8);
  box(A,B,.25-(i*.05),D,E,F);
  GPCLST();
  GPOPST(16);
  GPARV(1,1,16,1.);
  GPICI(4);
  box(-.139,.1462,2-(i*.05),1,.05,.05);
  GPCLST();
  GPUPWS(1,2);
}

for(i=0;i<29;i++)
{
  GPEST(16);
  GPDVRV(1,1,16);
  GPOPST(16);
  GPARV(1,1,16,1.);
  GPICI(4);
  box(-.139,.1462-.01*i,-.1-.01*i,1,.05,.05);
  GPCLST();
  GPUPWS(1,2);
}

GPEST(16);
GPDVRV(1,1,16);
GPUPWS(1,2);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* The subroutine rbinbl is the interaction module for demonstrating the */
/* interaction between Ram b and Box l. */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

rbinbl()
{
  int i;
  static float D = .05 ;
  static float E = .05 ;
  static float F = 1. ;
  static float A3 = .216;
  static float B3 = .1515;
  static float C3 = 0.5;
  GPEST(21);
  GPDVRV(1,1,21);
    conv1(1,.02,.5);
    plot();
    for(i=0;i<5;i++)
    {

err("in rbibn1")
GPEST(23);
GPDVR(1,1,23);
GPEST(16);
GPDVR(1,1,16);
GPOPST(23);
GPARV(1,1,23,1.);
GPICI(8);
box(A3,B3,C3-(i*.05),D,E,F);
GPICI(4);
box(.191,.1462,.25,.075,.05,.05);
GPCLST();
GPUPWS(1,2);
}

for(i=0;i<7;i++)
{
    GPEST(23);
    GPDVR(1,1,23);
    GPEST(16);
    GPDVR(1,1,16);
    GPOPST(23);
    GPARV(1,1,23,1.);
    GPICI(8);
    box(A3,B3,C3-(i*.05),D,E,F);
    GPCST();
    GPOPST(16);
    GPARV(1,1,16,1.);
    GPICI(4);
    box(.191,.1462-.01*i,.5-.01*i,.075,.05,.05);
    GPCST();
    GPUPWS(1,2);
}

for(i=0;i<29;i++)
{
    GPEST(16);
    GPDVR(1,0,16);
    GPOPST(16);
    GPARV(1,1,16,1.);
    GPICI(4);
    box(.191,.1462-.01*i,-.1-.01*i,.075,.05,.05);
    GPCST();
    GPUPWS(1,2);
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* The subroutine rbibn2 is the interaction module for demonstrating the */
/* interaction between Ram b and Box 2. */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

rbibn2()
{
    int i;
static float D = .05 ;
static float E = .05 ;
static float F = 1. ;
static float A3 = .216;
static float B3 = .1515;
static float C3 = 0.5;
gPEST(21);
GDPRV(1,1,21);
   conv1(.1,.02,.5);
   photo();
for(i=0;i<5;i++)
{
   err("in rbinb2");
   GPEST(23);
   GDPRV(1,1,23);
   GPEST(16);
   GDPRV(1,1,16);
   GPOPST(23);
   GPARV(1,1,23,1.);
   GPICI(8);
   box(A3,B3,C3-(i*.05),D,E,F);
   GPICI(4);
   box(.191,.1462,.25,.1,.05,.05);
   GPCLST();
   GPUPWS(1,2);
}
for(i=0;i<7;i++)
{
   err("in rbinb2");
   GPEST(23);
   GDPRV(1,1,23);
   GPEST(16);
   GDPRV(1,1,16);
   GPOPST(23);
   GPARV(1,1,23,1.);
   GPICI(8);
   box(A3,B3,.25-(i*.05),D,E,F);
   GPCLST();
   GPOPST(16);
   GPARV(1,1,16,1.);
   GPICI(4);
   box(.191,.1462,.2-(i*.05),.1,.05,.05);
   GPCLST();
   GPUPWS(1,2);
}
for(i=0;i<29;i++)
{
   GPEST(16);
   GDPRV(1,1,16);
   GPOPST(16);
   GPARV(1,1,16,1.);
   GPICI(4);
   box(.191,.1462-.1*i,-.05-.01*i,.1,.05,.05);
   GPCLST();
   GPUPWS(1,2);
}
} 

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
FILE *rort;

char TEXT41[3], TEXT42[2];
char TEXT43[4], TEXT44[6];

static float POS51[2]= {-.2, .2};   /* POS WITH GRAPHICS */
static float POS52[2]= {-.2,.3};     /* POS WITHOUT GRAPHICS */
static float POS53[2]= {-.8,-.5};   /* POS WITHOUT GRAPHICS */
static float POS54[2]= {-.82,-.55}; /* POS WITHOUT GRAPHICS */

void ratim(){
    err("PLEASE SELECT THE INTERACTION MODE");
    setup4(); /* SETS UP INITIAL DISPLAY SCREEN */
    disp1();
    GPUPWS(1,2);
    do {
        PPATH[1] =43;
        GPKF(1,1,1,incl1, excl1);  /* PUT PICK FILTERS */
        GPaweV(110.1, &wsid, &clas, &devid); /* AWAIT EVENT */
        if (clas == 5)
            sel_rot();
        GPUPWA(1); /* UPDATE WORKSTATION */
        GPFILEV(1,5,1);
    } while (PPATH[1] != 43);
    GPEST (70);
    GFDV (1,0,70);
    GPEST (20);
    GFDV (1,0,20);
}

rot(JR)
long JR;
{
    char vr,ch;
    rort = fopen("ranti","r");
    fseek(rort, JR, 0);
    vr= getc(rort);
fclose(rort);
if (vr==\'0\')
{
    rort = fopen("ranti","r+");
    fseek(rort,JR,0);
    ch = '\l';
    putc(ch,rort);
    fclose(rort);
}
else
{
    rort = fopen("ranti","r+");
    fseek(rort,JR,0);
    ch = '\0';
    putc(ch,rort);
    fclose(rort);
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/*
/* The subroutine vrot(JR) looks up in the boolean file at position JR the */
/* value stored and returns the value as a character */
/*
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

vrot(JR)
long JR;
{
    char vr;
    rort = fopen("ranti","r+");
    fseek(rort,JR,0);
    vr= getc(rort);
    fclose(rort);
    return vr;
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/*
/* The subroutine setup4() provides graphics for the triggers to be setup */
/* allowing the user to decide which interactive modules should be setup as */
/* trigger conditions */
/*
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

setup4()
{
    static float BOX71[8]= {-.3,.0,.10,.0,.30,.0,.30,.0,.30,-.30,.0,.40};
    static float BOX72[8]= {-.3,-.10,.0,.3,-.10,.0,.3,-.40,-.3,-.40};
    static float BOX43[8]= {-.85,-.65,-.55,-.65,-.55,-.35,-.85,-.35};
    GPOST (70);
    GPTXFO (10);        /* Activate the font */
    GPTXPR (2);        /* Set the text precision */
    GPEF(2);        /* Edge flag on for the box */
    GPELT(1);
    GPECI(1);        /* SET COLOR INDEX */
    GPIS(2);
    GPICI(3);        /* SET COLOR INDEX */
GPTXCI (2);
GPADCN (1,N1AME1);
GPCHH (.13);
GFARV (1.0,70.1.);
GFIS (2);
GPPKID(71);
GPPG2 (1, NP, 2, BOX71);
GPPKID(72);
GPPG2 (1, NP, 2, BOX72);
GPPKID(43);
GPPCH (0.03);
GPPG2 (1, NP, 2, BOX43);
GPTX2 (POSS3,4, TEXT43);
GPTX2 (POSS4,6, TEXT44);
GPCLST();
GPUPWS(1,2);
}

disp1()
{
err(-.215,.315,.04," RANDOM");
err(-.225,.195,.04," MODE");
err(-.225,-.205,.04," TIMED");
err(-.225,-.325,.04," MODE");
GPUPWS(1,2);
}

/*************************************************************************/
/* FUNCTION FOR SELECTING MENU ITEMS AND SWITCHING TO APPROPRIATE FUNCTIONS */
/*************************************************************************/

sel_rot()
{
if (clas == 5)
{
    GPGTPK(1,DEPTH,PPATH);
    ID = PPATH[1];
    printf("PP= %d ",ID);
    GPFLEV(1,5,1);
    switch(ID)
    {
        case 71: /* IF EMERGENCY STOP BUTTON IS SELECTED */
        {
            err("RANDOM MODULE NOT ATTACHED");
        }
        break;
        case 72: /* IF EMERGENCY STOP BUTTON IS SELECTED */
        {
            rot(2); /* CHANGING THE BOOLEAN VALUE IN FILE */
            if(vrot(2) == '1')
            {
                err("YOU HAVE SELECTED TIMED SEQUENCE MODE");
            }
            break;
        }
    GPFLEV(1,5,1);
    }
    GPFLEV(1,5,1);
}
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
# include "def.c"

FILE *trig;

/* *********************** CHARACTER VARIABLES *********************** */

char TEXT41[3], TEXT42[2];
char TEXT43[4], TEXT44[6];

/* *********************** DEFINITION OF POSITIONS FOR TEXT *********************** */

static float POS51[2] = -.2, .2; /* POS WITH GRAPHICS */
static float POS52[2] = -.2, -.3; /* POS WITHOUT GRAPHICS */
static float POS53[2] = -.8, -.5; /* POS WITHOUT GRAPHICS */
static float POS54[2] = -.82, -.55; /* POS WITHOUT GRAPHICS */

void trigr()
{
    err("PLEASE SELECT THE TRIGGER CONDITION");
    setup3(); /* SETS UP INITIAL DISPLAY SCREEN */
    disp();
    GRUPWS(1, 2);
    do {
        PPATH[1] = 43;
        GPPKF(1, 1, 1, incl1, excl1); /* PUT PICK FILTERS */
        GPAWEV(110.1, &wid, &clas, &devid); /* AWAIT EVENT */
        if (clas == 5)
            sel_trg();
        GRUPWA (1); /* UPDATE WORKSTATION */
        GPPELV(1, 5, 1);
        while (PPATH[1] != 43);
        GPEST (50);
        GPDVR (1, 0, 50);
        GPEST (20);
        GPDVR (1, 0, 20);
    }
}
static float BOX54[8] = {0.3, 0.35, 0.6, 0.35, 0.6, 0.65, 0.3, 0.65};
static float BOX55[8] = {-7.5, -1.5, -4.5, -1.5, -4.5, 0.15, 8.75, 0.15};
static float BOX56[8] = {-4.5, -1.5, -1.5, -1.5, -0.15, -4.0, 0.15};
static float BOX57[8] = {-0.5, -1.5, 0.25, -1.5, 0.25, 0.15, 0.05, 0.15};
static float BOX58[8] = {0.3, -1.5, 0.6, -1.5, 0.6, 0.15, 0.3, 0.15};
static float BOX43[8] = {-8.65, -0.65, -0.65, -0.55, -0.35, -0.35, -0.35, -0.35};
GPOPST (50);
GPTXFO (10);
GPTXPR (2);
GPEF (2);
GPELT (1);
GPEC1 (1);
GPTIS (2);
GEC1 (3);
GPTXCI (2);
GPAADCN (1, NAME1);
GPCHH (.13);
GPARV (1, 0, 50, 1,);
GPTIS (2);
GPPKID(51);
GPPG2 (1, NP, 2, BOX51);
GPPKID (52);
GPPG2 (1, NP, 2, BOX52);
GPPKID (53);
GPPG2 (1, NP, 2, BOX53);
GPPKID (54);
GPPG2 (1, NP, 2, BOX54);
GPPKID (55);
GPPG2 (1, NP, 2, BOX55);
GPPKID (56);
GPPG2 (1, NP, 2, BOX56);
GPPKID (57);
GPPG2 (1, NP, 2, BOX57);
GPPKID (58);
GPPG2 (1, NP, 2, BOX58);
GPPKID (43);
GPCHH (.03);
GPPG2 (1, NP, 2, BOX43);
GPTX2 (POSS53, 4, TEXT43);
GPTX2 (POSS54, 6, TEXT44);
GPLST();
GPUPWS(1, 2);
}

disp()
{
errl(-.72, .55, .025, "1. RAM A");
errl(-.72, .5, .025, " WITH");
errl(-.72, .45, .025, " BOX A");
errl(-.38, .55, .025, "2. RAM B");
errl(-.38, .5, .025, " WITH");
errl(-.38, .45, .025, " BOX B");
errl(-.03, .55, .025, "3. RAM A");
errl(-.03, .5, .025, " WITH");
errl(-.03, .45, .025, " BOX A");
errl(0.32, .55, .025, "4. RAM B");
errl(0.32, .5, .025, " WITH");
errl(0.32, .45, .025, " BOX B");
errl(-.735, .05, .02, "5. MOTOR ON");
errl(-.72, .0, .025, " IN");
errl(-.72, -.05, .025, " E.STOP");
errl(-.38, .05, .025, "6. RAM A");
errl(-.38, .0, .025, " WITH ");
rtzg(JR)
long JR;
{
    char vr,ch;
    trig = fopen("trigger","r");
    fseek(trig,JR,0);
    vr = getc(trig);
    fclose(trig);
    if (vr=='0')
    {
        trig = fopen("trigger","r++");
        fseek(trig,JR,0);
        ch = '1';
       putc(ch,trig);
        fclose(trig);
    }
    else
    {
        trig = fopen("trigger","r++");
        fseek(trig,JR,0);
        ch = '0';
       putc(ch,trig);
        fclose(trig);
    }
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* The subroutine vtrg(JR) looks up in the boolean file at position JR the */
/* value stored and returns the value as a character */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

vtrg(JR)
long JR;
{
    char vr;
    trig = fopen("trigger","r");
    fseek(trig,JR,0);
    vr = getc(trig);
    fclose(trig);
    return vr;
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* The subroutine setup3() provides graphics for the triggers to be setup */
/* allowing the user to decide which interactive modules should be setup as */
/* trigger conditions */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

setup3()
{
    static float BOX51[8]= {-.75,0.35,-.45,0.35,-.45,0.65,-.75,0.65};
    static float BOX52[8]= {-.4,0.35,-1.0.35,-1.0.65,-1.4,0.65};
    static float BOX53[8]= {-.05,0.35,0.25,0.35,0.25,0.65,-.05,0.65};
```c
err1(-.38,-.05,.025,"NO BOX");
err1(-.03,.05,.025,"7. RAM B");
err1(-.03,.0,.025,"WITH ");
err1(-.03,-.05,.025,"NO BOX");
err1(.32,.05,.022,"8. EMER.L");
err1(.32,.0,.025,"IN");
err1(.32,-.05,.025,"E.STOP");
GUPWPS(1,2);
}

/* **************************************************************************
/* FUNCTION FOR SELECTING MENU ITEMS AND SWITCHING TO APPROPRIATE FUNCTIONS */
/* **************************************************************************
/* **************************************************************************

sel_trg()
{
    if (clas == 5)
    {
        GPGTPK(1,6DEPTH,PPATH);
        ID = PPATH[1];
        printf("PP= %d ",ID);
        GPFLEV(1,5,1);
        switch(ID)
        {
        case 51: /* IF EMERGENCY STOP BUTTON IS SELECTED */
                    ttrg(1); /* CHANGING THE BOOLEAN VALUE IN FILE */
                    if(ttrg(1) == '1')
                        err("YOU HAVE SELECTED 1 TO BE A TRIGGER");
                    else
                        err("CONDITION 1 IS NOT A TRIGGER");
                    break;
        case 52: /* IF EMERGENCY STOP BUTTON IS SELECTED */
                    ttrg(2); /* CHANGING THE BOOLEAN VALUE IN FILE */
                    if(ttrg(2) == '1')
                        err("YOU HAVE SELECTED 2 TO BE A TRIGGER");
                    else
                        err("CONDITION 2 IS NOT A TRIGGER");
                    break;
        case 53: /* IF EMERGENCY STOP BUTTON IS SELECTED */
                    ttrg(3); /* CHANGING THE BOOLEAN VALUE IN FILE */
                    if(ttrg(3) == '1')
                        err("YOU HAVE SELECTED 3 TO BE A TRIGGER");
                    else
                        err("CONDITION 3 IS NOT A TRIGGER");
                    break;
        case 54: /* IF EMERGENCY STOP BUTTON IS SELECTED */
                    ttrg(4); /* CHANGING THE BOOLEAN VALUE IN FILE */
                    if(ttrg(4) == '1')
                        err("YOU HAVE SELECTED 4 TO BE A TRIGGER");
                    else
                        err("CONDITION 4 IS NOT A TRIGGER");
        }
    }
}
break;
case 55:          /* IF EMERGENCY STOP BUTTON IS SELECTED */
{                 /* CHANGING THE BOOLEAN VALUE IN FILE */
    rtrg(5);      /* CHANGING THE BOOLEAN VALUE IN FILE */
    if(vtrg(5) == '1')
        err("YOU HAVE SELECTED 5 TO BE A TRIGGER");
    else
        err("CONDITION 5 IS NOT A TRIGGER");
}                 /* CHANGING THE BOOLEAN VALUE IN FILE */
break;
case 56:          /* IF EMERGENCY STOP BUTTON IS SELECTED */
{                 /* CHANGING THE BOOLEAN VALUE IN FILE */
    rtrg(6);      /* CHANGING THE BOOLEAN VALUE IN FILE */
    if(vtrg(6) == '1')
        err("YOU HAVE SELECTED 6 TO BE A TRIGGER");
    else
        err("CONDITION 6 IS NOT A TRIGGER");
}                 /* CHANGING THE BOOLEAN VALUE IN FILE */
break;
case 57:          /* IF EMERGENCY STOP BUTTON IS SELECTED */
{                 /* CHANGING THE BOOLEAN VALUE IN FILE */
    rtrg(7);      /* CHANGING THE BOOLEAN VALUE IN FILE */
    if(vtrg(7) == '1')
        err("YOU HAVE SELECTED 7 TO BE A TRIGGER");
    else
        err("CONDITION 7 IS NOT A TRIGGER");
}                 /* CHANGING THE BOOLEAN VALUE IN FILE */
break;
case 58:          /* IF EMERGENCY STOP BUTTON IS SELECTED */
{                 /* CHANGING THE BOOLEAN VALUE IN FILE */
    rtrg(8);      /* CHANGING THE BOOLEAN VALUE IN FILE */
    if(vtrg(8) == '1')
        err("YOU HAVE SELECTED 8 TO BE A TRIGGER");
    else
        err("CONDITION 8 IS NOT A TRIGGER");
}                 /* CHANGING THE BOOLEAN VALUE IN FILE */
break;
}
GPFLEV(1,5,1);
GPFLEV(1,5,1);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
# include "def.c"

convl(l,w,h)

float l,w,h;
{
    int j;
    float MATR2[4][4];
    static float u,r;
    float x = -.8;
    static float z = 0.0;
    int i;
    r = (1/2.)*(1.005 + (1.0707));
    GPST(3);
    GPRV(1,1,3);
    GPOST(3);
    GPEF(2);
    GPEC(1);
    GPCI(2);
    i = 0;
    box (x-r,-1/2.,z,w,l,h);
    box2 = (.5907,.0625,-.820,.1157,.8341,.12984,.9048,.07699);
    box2 = (.5907,.0625,-.820,.1157,.8341,.12984,.9048,.07699);
    box2 = (.6017,.0625,.531,.1157,.5451,.12984,.6158,.07699);
    box2 = (.6017,.0625,.531,.1157,.5451,.12984,.6158,.07699);
    do
    {
        box (x,r,z,l,w,h);
        x = x+1+.01;
        i++;
    } while (i < 12);
    u = -r-.02;
    j = 0;
    x = -0.8;
    do
    {
        box (x,u,z,l,w,h);
        j++;
        x = x+1+.01;
    } while (j < 12);
    box (x+r,.03,-1/2.,z,w,l,h);
    GPRV(1,1,3,1.);
    GPCLST();
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* This program allows you to draw the conveyor using the subroutine box. */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

#include "def.c"

conv2(l, w, h)
float l, w, h;
{
    int i;
    static float u, v;
    float x = -.745;
    static float z = 0.0;
    int i;
    r = (1/2.)+.005 +(1*7.07);
    GPEST(3);
    GPDRV(1,2,3);
    GPST(3);
    GPARV(1,2,3,1.);
    GPEF(2);
    GPECI(1);
    GPICI(2);
    i = 0;
    box2(-.8566,.0998,-.176,1257,-.7652,145,-.860,.12);
    box2(-.8566,-.0998,-.176,-1257,-.7652,-145,-.860,-.12);
    box2(-.9228,.01,-.866,.0928,.883,1094,-.942,.022);
    box2(-.9228,.01,-.866,-.0928,-.883,-1094,.942,.022);
    box2(.5666,.0998,.47,1257,.4752,.145,.57,.12);
    box2(.5666,-.0998,.47,-1257,.4752,-.145,.57,-.12);
    box2(.6328,.01,.576,.0928,.593,1094,.652,.022);
    box2(.6328,-.01,.576,-.0928,.593,-1094,.652,-.022);
    do
        box (x, r, z, l, w, h);
        x = x+l+.01;
        i++;
    } while (i < 11);
    u = r-.02;
    j = 0;
    x = -.745;
    do
        box (x, u, z, l, w, h);
        j++;
        x = x+l+.01;
    } while (j < 11);
    GPCLST();
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* This program allows you to draw the Emergency lamp */

#include "def.c"

char T1[1];
static float P1[2]={-0.84,.525};
emerL()
{
  strcpy (T1,"L");
  GPEST(26);
  GPDV(1,0,26);
  if(getca("y6")=='1')
  {
    GPOPST(26);
    GPARV(1,0,26,1.);
    GPTXCI(1);
    GPCHH(.025);
    GPICI(8);
    box(-.85,.515,.7,.05,.05,.05);
    GPTX2(P1,1,T1);
    GPCLST();
    GPUPWS(1,2);
  }
  else
  {
    GPOPST(26);
    GPARV(1,0,26,1.);
    GPTXCI(1);
    GPCHH(.025);
    GPICI(7);
    box(-.85,.515,.7,.05,.05,.05);
    GPTX2(P1,1,T1);
    GPCLST();
    GPUPWS(1,2);
  }
}
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* This program allows you to move the boxes on the conveyor */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

#include "def.c"

int ii, jj;
float XX1, XX2;

mbox1()
{
    char y1, y2, y3, y4;
    XX1 = -0.8 - 0.055;

    for(ii=0; ii<23; ii++)
    {
        printf(" IN MBOX1 %d ", ii);
        rite2(6, '1');

        if (vseek(4) == '1')
        {
            /* err("IN CHECK1"); */
            XX1 = XX1 + 0.055;
            XX2 = XX1 + 0.075;
            if (XX1 < -0.643) && (-0.643 < XX2))
            {
                /* err("IN CHECK2"); */
                rite2(10, '1');
            }
            else
            {
                /* err("IN CHECK3"); */
                rite2(10, '0');
            }
        }
        tri();
        FORT();
        if (getca("y2") == 1 && CHK2(1) != 1)
        {
            /* err("In #1"); */
            mrama();
            if (vtrg(6) == '1')
            {
                Trace("y2");
                err("SELECT STOP SIM SWITCH");
                t1 = 1;
                return t1;
            }
            else
            {
                /* err("In #1"); */
                rite2(6, '0');
                tri();
                FORT();
                return;
            }
        }
        if (getca("y2") == 1)
        {
            if (CHK2(1) == 1)
/* err("In #2"); */
/* This is the only correct response */
/* */
/* printf(" IN MBOX1 ACTUAL %d ",ii); */
/* X[8][li].value=0; */
real(6,'O');
tri();
FORT();
return;
}
else
{
  rainb1();
  Trace("y2");
  err("SELECT STOP SIM SWITCH");
  t1 =1;
  return t1;
}
}

/* if ( getca("y4")==1 && CHK(2) !=1 ) */
if ( getca("y4")==1 && CHK(2) == 1 )
{
  if (vtrg(2)=='O')
  {
    /* err("In #4"); */
    rbinb1();
    return;
  }
  else
  {
    rbinb1();
    Trace("y4");
    err("SELECT STOP SIM SWITCH");
    t1 = 0;
    return t1;
  }
}
GPEST(16);
GPDRV(1.1,16);
GPEST(3);
GPDRV(1,0,3);
if(ii42==1)
{
  conv2(.1,.02,.5);
}
else
{
conv1(.1,.02,.5);
}
photo();
GPOP$T(16);
GPICI(4);
GPARV(1.1,16,1.);
box(-0.8+(li*0.055),.1462,0.25,.075,.05,.055);
GPUPWS(1,2);
GPCLST();
rite2(6,'O');
tri();
FORT();
GPUPWS(1,2);
}

/* STD[0].count= 0;
X1[8][11].value=0: */
rite2(6,'O');
tri();
FORT();
GPEST(16);
GPDIV(1,1,16);

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
# include "def.c"

int ii, j;
float XX3, XX4;

mbox2()
{
  char y1, y2, y3, y4;
  XX3 = -0.8 -.055;
  for(ii=0; ii<23; ii++)
  {
    /* printf(" IN MBOX1 %d ", ii); */
    rite2(6,'1');
    if (vseek(4) == '1')
    {
      /* err("IN Check 1"); */
      XX3 = XX3 + 0.045;
      XX4 = XX4 + 0.12;
      if (XX3 < -0.643 && (-0.643 < XX4))
      {
        rite2(10,'1');
        /* err("IN Check 2"); */
      }
      else
      {
        rite2(10,'0');
        /* err("IN Check 3"); */
      }
    }
    tri();
    FORT();
    if (getca("y2")==1 && CHK2(1) != 1)
    {
      /* err("In #1"); */
      mrama();
      if(vtrg(6)=='1')
      {
        Trace("y2");
        err("SELECT STOP SIM SWITCH");
        t1 = 1;
        return t1;
      }
      else
      {
        rite2(6,'0');
        tri();
        FORT();
        return;
      }
    }
    if (getca("y2")==1)
    {
      if (CHK2(1) == 1)
      {
        if(vtrg(3)=='0')
        {
/* err("In #2"); */
 rainb2(); /* This is the only correct response */
/* printf(" IN MBOX1 ACTUAL \%d ",ii); */
/* XI[8][11].value=0; */
 rite2(6,'0');
 tri();
 FORT();
 return;
}
else
{
 rainb2();
 Trace("y2");
 err("SELECT STOP SIM SWITCH");
 tl = 1;
 return tl;
}

/* if ( getca("y4") == 1 & & CCHK(2) !=1) */
{
 err("In #3");
 mrmb();
 if(vtrg(7)=='1')
 {
 Trace("y4");
 err("SELECT STOP SIM SWITCH");
 return;
 }
 else
 return;
} */
if ( getca("y4") == 1 & & CCHK2(2) == 1 )
{
 if(vtrg(4)=='0')
 {
 /* err("In #4"); */
 rbimb2();
 rite2(6,'0');
 tri();
 FORT();
 return;
 }
 else
 {
 rbimb2();
 Trace("y4");
 err("SELECT STOP SIM SWITCH");
 tl = 1;
 return tl;
 }
} GPEST(16);
GPDV(1,1,16);
GPEST(3);
GPDV(1,0,3);
if(lis2==1)
{
 conv2(.1,.02,.5);
}
else
{
 conv1(.1,.02,.5);
photo();
GPOPST(16);
GPICI(4);
GPARV(1,1,15,1.);
box(-0.8+(ii*.055),.1462,0.25,.1,.05,.05);
GPUPWS(1,2);
GPCLST();
rite2(6,'0');
tri();
FORT();
GPUPWS(1,2);
}
/*
  STD[0].count= 0;
  XL[8][11].value=0; */
rite2(6,'0');
tri();
FORT();
GPEST(16);
GPD RV(1,1,16);
*/
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* This program allows you to move the conveyor and calls the movc.c program */
/* which moves the conveyor */
/* */
#include "def.c"

mconv()
{
    int devid,clas, wsid;
    float TN;
    char y2,y3,y4,y5,y6;
    TN =0;
    for(t=0;t<1.0000000000;t++)
    {
        tri();
        FORT();
        if (vrot(2) == '1')
        {
            TN=TN+0.5;
        }
        if(TN > 10.0)
        {
            t1 =0;
            return t1;
        }
    }
    err("IN MOVE CONVEYOR");
    if(t%2==1)
    {
        GPDRV(1,0,16);
        GPDRV(1,27);
        GPDRV(1,1,27);
        GPDRV(3);
        conv1(.3,1,.01,.02);
        photo();
        GPUPWS(1,2);
        GPAWEV(.1,&wsid,&clas,&devid);
        if (clas ==5)
        {
            t1 = 0;
            return t1; /*, ID; DRIVE(); sel_menu(); */
        }
    }
    else
    {
        GPDRV(16);
        GPDRV(1,0,16);
        GPDRV(3);
        GPDRV(1,0,3);
        conv2(.3,1,.02,.05);
        photo();
        GPUPWS(1,2);
        GPAWEV(.1,&wsid,&clas,&devid);
        if (clas==5)
        {
            t1 = 0;
            return t1; /*, ID; DRIVE(); sel_menu(); */
        }
    }
}
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* */
/* This program is for the graphical representation of the movement of */
/* the RAMS */
/* */
/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */

#include "def.c"

int i;

static float A = -0.114;
static float B = 0.1515;
static float C = 0.5;
static float D = 0.05;
static float E = 0.05;
static float F = 1.0;
static float A3 = 0.216;
static float B3 = 0.1515;
static float C3 = 0.5;

/* Draw the A1's first, then draw the A2 box and then the A box */

void mrana()
{
    err("In mrana");
    GPEST(20);
    GPDRV(1, 0, 20);
    for (i = 0; i < 5; i++)
    {
        err("In mrana 2");
        GPEST(22);
        GPDRV(1, 1, 22);
        GPPOST(22);
        GPARV(1, 1, 22, 1.);
        GPSR(2);
        GPICI(8);
        box(A, B, C - (i * .1), D, E, F);
        GPCLST();
        GPUPWS(1, 2);
    }
    return;
}

void mramb()
{
    GPEST(21);
    GPDRV(1, 1, 21);
    for (i = 0; i < 5; i++)
    {
        GPEST(23);
        GPDRV(1, 1, 23);
        GPPOST(23);
        GPARV(1, 1, 23, 1.);
        GPSR(2);
        GPICI(8);
        box(A3, B3, C3 - (i * .1), D, E, F);
        GPCLST();
        GPUPWS(1, 2);
    }
    return;
} 

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
#include "def.c"

char T1[1];
static float P1[2]=(-.84,.525);
photo()
{
   GPEST(27);
   GPD RV(1,1,27);
   GPORST(27);
   GPARV(1,1,27,1.);
   GPICI(3);
   box(-.65,.15,.55,.075,.075,.075);
   GPICI(8);
   box(-.625,.175,.55,.025,.025,.025);
   if (getca("y7") ==1)
   {
      GPICI(8);
      box(-.625,.0175,.15,.025,.022,.405);
   }
   GPCLST();
}

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
/* This program is for the static graphical representation of the RAMS */

#include "def.c"

static float A = -0.114 ;
static float B = .1515;
static float C = 0.5;
static float D = .05 ;
static float E = .05 ;
static float F = 1. ;
static float A3 = .216;
static float B3 = .1515;
static float C3 = 0.5;
float MATR2[4][4];

/* Draw the A1's first, then draw the A2 box and then the A box */

Ram2()
{
    Ram0();
    Ram1();
}

Ram0()
{
    GPEST(22);
    GPDRV(1,1,22);
    GPOPST(20);
    GPARV(1,1,20,1.);
    GPIS(2);
    GPEF(1);
    GPECI(1);
    GPICI(5);
    box(A,B,C,D,E,F);
    GPCLST();
}

Ram1()
{
    GPEST(23);
    GP0RV(1,1,22);
    GPOPST(21);
    GPARV(1,1,21,1.);
    GPIS(2);
    GPICI(5);
    box(A3,B3,C3,D,E,F);
    GPCLST();
}

/* */
/* This program is for the graphical representation of the retraction of RAMS */

#include "def.c"

int i;

static float A = -0.114;
static float B = .1515;
static float C = 0.;
static float D = .05;
static float E = .05;
static float F = 1.;
static float A3 = .216;
static float B3 = .1515;
static float C3 = 0.0;

/* Draw the A1's first, then draw the A2 box and then the A box */

void rrama()
{
    GPEST(20);
    GPDRV(1,1,20);
    for (i=0; i<5; i++)
    {
        GPEST(22);
        GPDRV(1,1,22);
        GPST(22);
        GFARV(1,1,22,1.);
        GPIS(2);
        GPICI(5);
        box(A,B,C+(i*.01),D,E,F);
        GPCLST();
    }
    Ram2();
}

void rramb()
{
    GPEST(21);
    GPDRV(1,1,21);
    for (i=0; i<5; i++)
    {
        GPEST(23);
        GPDRV(1,1,23);
        GPST(23);
        GFARV(1,1,23,1.);
        GPIS(2);
        GPICI(5);
        box(A3,B3,C3+(i*.01),D,E,F);
        GPCLST();
    }
    Ram2();
}

/* */
This program is for the tracing of the variables and outputs for the error output.

#include "def.c"

int l,i,ii,iii, I, j, jj, J,ki, K, li, L, m, M, r, R, ccc, cnt;
int Cl[100][12], CMT, val5, mmmm;
long val1, val11, val4;

FILE *RES, *TRACK;
FILE *in, *bool;
char vv[] = {""};
char kk[] = {""};
char kr[] = {""};
char nnn[] = {""};
char nnn3[] = {""};
char nnn5[] = {""};
char nnn7[] = {""};

struct CCC {
    char TRACE[4];
};
static struct CCC CCCC[N];

Trace(U)
char U[];
{
    mmmm = N*4;
    get1(U);      /* Subroutine to find final line that caused the defect. */
    getn();
    ti = 1;
    return ti;
}

get1(C)
char C[];
{
    for (li = 0; li < 4; li++)
    {
        CCCC[0].TRACE[li]=C[li];
    }
    for (jj=0; jj<N; jj++)
    {
        if(strcmp(C,X1[jj][11].name)==0)
        {
            RES = fopen("RESULT","a"); /* Reading the PLC FILE */
            fprintf(RES,"The output %s is identified as belonging to line #%d ",C,jj);
            fclose(RES);
        }
    }
}
```c
TRACK = fopen("TRACER","a");
    for(ii=0;ii<l3;ii++)
    {
        fprintf(TRACK,"%s X1[j][i].name;
    }
    fprintf(TRACK," ");
    fclose(TRACK);
}

/******
** The subroutine getn() is used to find the inputs in the jth line which
** caused the defect
**
* getn()
{
    cnt =0;
    ccc=1;
    do(
        for (CMT=0; CMT < 100000; CMT++)
        {
            cnt++;
            TRACK = fopen("TRACER","r");
            fseek(TRACK, CMT, 0);
            fscanf(TRACK,"%s",nnnn);
            fclose(TRACK);
            if( (strcmp(nnnn,nnnn6)==0) && strcmp(nnn6,nnnn7)==0)
            {
                cnt=nnnn;
            }
            if((strcmp(nnnn,nnnn6)!=0))
            {
                for (li = 0; li < 4; li++)
                {
                    nnnn6[li]=nnnn[li];
                }
            }
            if( (strcmp(nnnn,nnnn6)==0) && (strcmp(nnnn6,nnnn7)!=0))
            {
                for (li = 0; li < 4; li++)
                {
                    nnnn7[li]=nnnn[li];
                }
            }
            for (li=0;li < N; li++)
            {
                if(strcmp(nnnn,CCCC[li].TRACE)==0)
                {
                    nnnn[0]='0';
                }
            }
            if(nnnn[0] == 'x' || nnnn[0] == 'n')
            {
                cnum(nnnn);
                val5 = val4;
                if ( (vseek(val5) == 'l'))
                {
                    vv[0]="1";
                }
            }
        }
    }
```
} 
else 
{ 
vv[0]='0'; 
} 
CMT = CMT+3; 
RES = fopen("RESULT","a"); /* Reading the PLC FILE */ 
fprintf(RES,"The input %s is in state %c",nnnn,vv[0]); 
fclose(RES); 
} 
if(nnnn[0] == 'c' || nnnn[0] == 'C') 
{ 
if(nnnn[0] == 'c') 
{ 
 nnnn3[0] = 'C'; 
 for(li = 1; li < 4; li++) 
 { 
 nnnn3[li]=nnnn[li]; 
 } 
} 
if(nnnn[0] == 'C') 
{ 
 nnnn3[0] = 'c'; 
 for(li = 1; li < 4; li++) 
 { 
 nnnn3[li]=nnnn[li]; 
 } 
} 
CMT = CMT + 3; 
for (li = 0; li < 4; li++) 
{ 
 CCCC[ccc].TRACE[li]=nnnn[li]; 
 } 
ccc++; 
for (ii=0;ii<N;ii++) 
{ 
if((strcmp(nnnn.XI[ii][11].name)==0) || (strcmp(nnnn3.XI[ii][11].name)==0)) 
{ 
 RES = fopen("RESULT","a"); /* Reading the PLC FILE */ 
 fprintf(RES,"The input %s belongs to line# %d and is in state %d ",nnnn,ii,XI[ii][11].name); 
 fclose(RES); 
 TRACK = fopen("TRACER","a"); 
 for(iii=0;iii<13;iii++) 
 { 
 fprintf/TRACK,"%s ",XI[ii][11].name); 
 } 
 fprintf/TRACK," "); 
 fclose/TRACK); 
 } 
} 
if(nnnn[0]=='y' || nnnn[0] == 'Y') 
{ 
if(nnnn[0] == 'y') 
{ 
 nnnn3[0] = 'Y'; 
 for(li = 1; li < 4; li++) 
 { 
 nnnn3[li]=nnnn[li]; 
 } 
} 
if(nnnn[0] == 'Y') 
{ 

struct X1[1][11].name == 0) || (strcmp(nnnn3,X1[ii][ll].name) == 0))
{
    RES = fopen("RESULT","a"); /* Reading the PLC FILE */
    fprintf(RES,"The input %s belongs to line# %d and is in state %d ",nnnn,ii,X1[ii][ll].name);
    fclose(RES);
    TRACK = fopen("TRACER","a");
    for(iii=0;iii<13;iii++)
    {
        fprintf(TRACK,"%s ",X1[ii][ll].name);
    }
    fprintf(TRACK," ");
    fclose(TRACK);
}

if(nnnn[0] == '0' )
{
...

if(nnnn[0] == 'S' )
{
    CMT = CMT +3;
    RES = fopen("RESULT","a"); /* Reading the PLC FILE */
    fprintf(RES,"A COUNTER %s IS IDENTIFIED AS AN INPUT ",nnnn);
    fclose(RES);
}

if(nnnn[0] == 'T' )
{
    CMT = CMT +3;
    RES = fopen("RESULT","a"); /* Reading the PLC FILE */
    fprintf(RES,"A TIMER %s IS IDENTIFIED AS AN INPUT ",nnnn);
    fclose(RES);
}

if(nnnn[0] == 'o' )
{
...
}

} while(cnt < mmmn);

cnum(C)
char C[];
char copy2[];
J=strlen(C);
l=0;
for(L=1;L<J+1;L++)
{
    copy2[l]=C[L];
    l++;
}
val4 = atoi(copy2);
return val4;

/* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ */
APPENDIX 2
<table>
<thead>
<tr>
<th>n1</th>
<th>M</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>c1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c2</td>
<td>1</td>
</tr>
<tr>
<td>x2</td>
<td>n1</td>
<td>o</td>
<td>c3</td>
<td>n1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>c1</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>x7</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>x4</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>x4</td>
<td>x10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c11</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c7</td>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c8</td>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c4</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c9</td>
<td>c11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>c9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c10</td>
<td>C11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>C10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td>o</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>c5</td>
<td>o</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y7</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX 3
<table>
<thead>
<tr>
<th>n1</th>
<th>M</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>c1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c2</td>
<td>1</td>
</tr>
<tr>
<td>x2</td>
<td>n1</td>
<td>o</td>
<td>c3</td>
<td>n1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c3</td>
<td>2</td>
</tr>
<tr>
<td>x3</td>
<td>c1</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c4</td>
<td>3</td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c5</td>
<td>4</td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c6</td>
<td>5</td>
</tr>
<tr>
<td>x4</td>
<td>x7</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c7</td>
<td>6</td>
</tr>
<tr>
<td>x8</td>
<td>x4</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c8</td>
<td>7</td>
</tr>
<tr>
<td>x6</td>
<td>x4</td>
<td>x10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c11</td>
<td>8</td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c7</td>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c9</td>
<td>9</td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c8</td>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c10</td>
<td>10</td>
</tr>
<tr>
<td>c1</td>
<td>c4</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y2</td>
<td>11</td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c9</td>
<td>c11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y1</td>
<td>12</td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>c9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y3</td>
<td>13</td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c10</td>
<td>C11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y4</td>
<td>14</td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>C10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y5</td>
<td>15</td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y6</td>
<td>17</td>
</tr>
<tr>
<td>c5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y7</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>n1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>n1</td>
<td>o</td>
<td>c3</td>
<td>n1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>c1</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>n1</td>
<td>c3</td>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>x7</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>x4</td>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>x4</td>
<td>x10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c11</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c7</td>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>c8</td>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c4</td>
<td>c3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c9</td>
<td>c11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>c9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>c5</td>
<td>c3</td>
<td>x4</td>
<td>c10</td>
<td>C11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>o</td>
<td>C10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>c4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>c5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>y7</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>
Vita

Krishna Kumar Krishnan was born on June 6, 1961 at Trivandrum, India. On completing his Bachelors in Mechanical engineering from Kerala University in August 1984, he joined the Indian Space Research Organization as a Scientific Officer. He was primarily involved with the research and development of new manufacturing technologies and the creation of quality control procedures for the first liquid engine booster ever to be developed by India.

In pursuance of higher studies he resigned from his job and began his graduate work in Industrial Engineering at Virginia Tech in Fall 1988. He will continue to pursue his doctoral degree under Dr. O. K. Eyada in the Industrial and Systems Engineering department at Virginia Tech.