



ROBOT IMPLEMENTATION LOGISTICS

IN

CONSTRUCTION ROBOTIC EQUIPMENT  
MANAGEMENT SYSTEM

BY

CARLOS M. CORNEJO



December 1990

**Division of Construction  
Engineering and Management  
School of Civil Engineering  
Purdue University  
West Lafayette, Indiana 47907**

**ROBOT IMPLEMENTATION LOGISTICS**  
**in**  
**CONSTRUCTION ROBOTIC EQUIPMENT**  
**MANAGEMENT SYSTEM**

A Special Research Problem  
Presented To

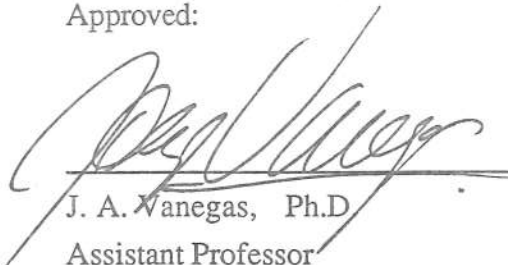
The Faculty of the  
Area of Construction Engineering and Management  
School of Civil Engineering  
Purdue University

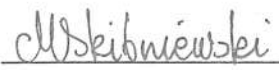
by

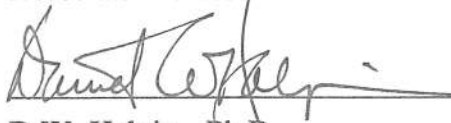
Carlos M. Cornejo

In Partial Fulfillment  
of the Requirements for the Degree of  
Master of Science in Civil Engineering

Approved:

  
J. A. Vanegas, Ph.D. Nov. 30, 1990.  
Assistant Professor Date

  
M. J. Skibniewski, Ph.D. Nov. 30, 1990  
Associate Professor Date

  
D. W. Halpin, Ph.D. 30 Nov 1990  
Area Head, CEM Date

## **ABSTRACT**

Projections indicate that an increasing number of construction operations are being automatized, and construction firms are acquiring more automated equipment for their daily operations. The automated equipment is highly expensive but cheaper than normal human labor operations in the long run. Several factors such as operation, maintenance, and management are involved with the automation of construction tasks, in order to get the maximum efficiency from the equipment.

Management is one of the crucial factors for assuring a high productivity rate. Automated equipment, like any other construction resource, needs to be scheduled within different projects. This report discusses the progress and software development of the Robot Implementation Logistics Module prototype (*RILM*), which is one of the modules of the Construction Robotic Equipment Management System (*CREMS*).

The *RILM* program is implemented in the *HyperCard* environment on the Macintosh microcomputer, and the main body of *RILM* was developed in the *QuickBASIC* programming language. Care has been taken to assure consistency of *RILM* with the other modules of *CREMS*.

To understand *CREMS*, a summary has been included in the introduction section. This summary explains the general functions *CREMS* and the interaction with *RILM*. Section 4 contains an example application session of *RILM*, demonstrating the features of the current prototype.

## TABLE OF CONTENTS

ABSTRACT .....	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES .....	iv
LIST OF TABLES .....	vi
SECTION 1: INTRODUCTION	
Problem Statement.....	1
State of the Art .....	2
Objective of the Study.....	3
Methodology .....	4
Construction Robotic Equipment Management Systems (CREMS).....	4
Background of Robot Implementation Logistics .....	6
Reader's Guide.....	9
Benefits of the Prototype.....	9
SECTION 2: THE RILM MODULE	
Implementation Logistics Issues .....	11
Assumptions of RILM .....	12
Architecture of RILM .....	13
SECTION 3: THE RILM SOFTWARE	
Software Configuration .....	18
Program Flowcharts	
RILM Flowchart within HyperCard .....	21
Preliminary Scheduling (QB-PRO15) .....	22
Critical Segment Analysis (QB-Seg-PRO16).....	22
Resource Leveling Procedure (QB-PRO16) .....	23
The Non-MTPR Submodule.....	27
SECTION 4: EXAMPLE APPLICATION SESSION	
Installation of the RILM Module.....	30
Example Application of RILM .....	31
Interpretation of Results.....	35
SECTION 5: CONCLUSIONS AND RECOMMENDATIONS	
Conclusions .....	40
Recommendations .....	41
Acknowledgements.....	41
REFERENCES.....	42
APPENDIX A: Glossary.....	43
APPENDIX B: RILM Cards.....	46

## LIST OF FIGURES

Figure 1.1	Configuration of the Principal Modules of CREMS .....	5
Figure 1.1	Conceptual Framework of RILM within CREMS.....	8
Figure 2.1	Configuration of RILM.....	13
Figure 2.2	Resource Leveling Flowchart (PRO-16) .....	15
Figure 2.3	Non-MTPR Submodule .....	17
Figure 3.1	RILM Configuration and Files Interaction .....	20
Figure 3.2	RILM Flowchart within HyperCard .....	21
Figure 3.3	Detailed Resource Leveling Flowchart (PRO16) .....	25
Figure 3.4	Detailed MTPR Process Flowchart.....	26
Figure 3.5	Non-MTPR Process Flowchart.....	28
Figure 4.1	Contents of the Final CREMS Folder.....	31
Figure 4.2	Project Gantt Chart for Basic Scheduling .....	32
Figure 4.3	Project Downhill Chart for Basic Scheduling.....	33
Figure 4.4	Result of Resource Leveling .....	34
Figure 4.5	Non-MTPR Solution Options.....	37
Figure 4.6	Result of Resource Leveling for the Three Solution Options .....	39
Figure A.1	Scheduling Charts.....	44
Figure B.1	Compile List of Projects.....	46
Figure B.2(a)	Logistics Parameters .....	46
Figure B.2(b)	Gantt Chart for Basic Scheduling (QuickBASIC screen) .....	47
Figure B.3	Logistics Parameters (Cont.).....	47
Figure B.4	Continue Card .....	47

## *Robot Implementation Logistics*

Figure B.5(a) Preliminary Scheduling.....	48
Figure B.5(b) Project Downhill Chart for Basic Scheduling.....	48
Figure B.6 Identifying Critical Segments (Critical Segment 1).....	48
Figure B.7 Project Leveling (Project Information for Critical Segment 1) .....	49
Figure B.8(a) Result of Project Leveling (Critical Segment 1) .....	49
Figure B.8(b) Downhill Chart for Scheduling of the Critical Segment 1 .....	49
Figure B.9 Project Leveling (Cont.).....	50
Figure B.10 Identifying Critical Segments (Critical Segment 2).....	50
Figure B.11 Project Leveling (Project Information for Critical Segment 2) .....	50
Figure B.12(a) Solution Options. QuickBASIC Screen.....	51
Figure B.12(b) QuickBASIC screen for the Switch Option after clicking on the "Apply changes & graph view" button.....	51
Figure B.13 Result of Project Leveling (Critical Segment 2) .....	51
Figure B.14 Project Leveling (Cont.) Final Card.....	52

## **LIST OF TABLES**

Table 2.1	Example Preference Criteria and Rules for Basic Robot Scheduling .....	16
Table 3.1	Description of Files.....	20
Table 3.2	List of Variables in Flowcharts.....	29
Table A.1	Tactical Rules.....	44



## **SECTION 1: INTRODUCTION**

Construction robot research and development resulted in large amounts of resources committed by the firms which undertook such efforts. One of the most important implied assumptions made in R&D investment decision making was one of future economic payoff on each such robot investment. The economic payoff can be achieved only if robot implementation decisions are made on a sound engineering basis.

The construction industry needs to handle large and expensive equipment for its daily operations. This equipment represents a large portion of the assets of a construction firm; therefore an adequate management of it is necessary. Among the heavy construction equipment, which is classified as the most expensive machinery, are the highly specialized machines such as scrapers, loaders, bulldozers, and so forth. Automated construction equipment is considered part of the expensive and highly specialized group of construction equipment. With the increase of the competitive market, the management team urges for computerized systems dedicated to solve the economical aspects of the acquisition and utilization of the equipment as well as the resource allocation and scheduling aspects.

### **Problem Statement**

The management of specialized construction equipment involves different issues such as an economical feasibility study for using the new technology compared to traditional ways of executing the construction tasks, allocation and assignment of the resource, implementation of the equipment in the jobsite, and scheduling of the resource among the different projects of the firm's portfolio competing for the use of this expensive equipment.

As a response to the need to effectively manage diverse robots on future construction sites, the Construction Robotic Equipment Management System (*CREMS*) analyzes the match between construction job tasks required on specific project sites and the capabilities of robots (Skibniewski et al. 1989a, Russell et al. 1990). Four basic modules constitute the *CREMS* architecture: *CTAM* (Construction Task Analysis Module), *RCAM* (Robot Capability Analysis Module), *REEM* (Robot Economic Evaluation Module), and *RILM* (Robot Implementation Logistics Module) (Skibniewski et al. 1989b, Skibniewski et al. 1990b). This paper is concerned with the research on *RILM* (Skibniewski et al. 1990c).



## **State of the Art**

Economical allocation of highly expensive and specialized construction equipment is an area in which many firms developed computer softwares. The applications in the market for resource scheduling are vast, however the results presented by each of them are very similar and varies only in the output format (e.g. more graphic capabilities). The majority of the current softwares in the market are dedicated to the scheduling of activities and resources for one project at the time. Given the characteristics for the activities such as earliest and latest start times, float times, and resource utilization, the software returns a scheduling chart with the critical path and the best combination of resource allocation within the project. The principal characteristics of the two most known computer applications in resource scheduling are described below:

- Fleetmatch (Caterpillar): Caterpillar supplies a software application package to buyers of Caterpillar equipment as an aid in management the equipment fleet. This program is to evaluate various equipment solutions. It also helps determine the most economical selection of equipment. Once data is entered, output is displayed graphically by pies, tables, graphs, bar charts, or the output is listed. The type of output is as follows: number of haulers needed, fleet availability, costs, affects of haulers costs. It also gives total cost and profit potential for a fleet as well as productivity. Input data may be as follows: number of loaders, purchase price, down payment, trade-in, book value of trade-in, depreciation, tire cost, and replacement cost (Caterpillar, Inc. 1990).
- Primavera Project Planner: The resource leveling and resource cost management issues are handled by two subprograms. For Primavera the amount of resource usage is not as crucial as the timely completion of the project; therefore it attempts to utilize any positive float to smooth resource usage without delaying the project. If there are insufficient resources to complete the required work, Primavera expects an increase in resource availability for completing the project. Primavera delays activities until the needed resources are available (Primavera Systems, Inc. 1987).

These two softwares are the most representative resource leveling applications. In general, these applications assign and allocate resources within projects after analyzing the total float times and profitability of their utilization.

The implementation of robots, as scarce resources, in construction sites requires a detailed study of the characteristics of each project and the profitability of using the robot compared to the non-robot solution or traditional execution. The software presented above aids in the assignment process of resources to projects but lacks of detailed analysis to provide the most optimum results with a user interface within a set of different projects.

The market requires a software able to assign highly specialized construction equipment, such as robots, to different tasks and project-sites within the project firm's portfolio. A goal of the analysis is to increase the profit of the company related to the implementation of robots to the different sites. This profit is the difference of the cost of executing the task with the robot and the cost of executing the same task by the conventional ways. The analysis should take into account task and robot characteristics for solving a resource leveling and implementation issues of the robot for the construction firm.

### **Objective of the Study**

The main objective of this study is to develop a computer program, which when executed, allocates a scarce resource within a set of projects. The program developed was tend to prove the concept of *RILM*; therefore the current version is on its first phase. Future research in the area may enhance the current version and develop more efficient *RILM* prototypes.

The *RILM* program assisted by the user evaluates a set of projects currently in a firm's portfolio which compete for the use of robots and allocates robots based on the optimal solution. The first three modules of *CREMS* generate some of the information required by *RILM*, such as robot cost and non-robot cost for a certain activity. The rest of the data, such as individual project information, are provided by the user. The main objectives of *RILM* are: 1) minimizing the idle time of the robot, 2) leveling the use of labor craft resources, and 3) reducing the total cost of logistic scheduling for all projects in a firm's portfolio.

The basic methodology of robot assignment to construction tasks can also be applied to the assignment of other, more conventional types of construction equipment treated as scarce resources by contractors, such as tower cranes, excavators, bulldozers, and other.

## **Methodology**

The allocation and implementation issues of robots in the construction industry required a detail investigation of commercially available programming applications and a detailed study of mathematical models, scheduling techniques, and heuristic-based models for developing the algorithms. The applications selected for the programming environment are HyperCard and QuickBASIC. The robot implementation logistics module is part of a construction robotic equipment management system, which was developed in HyperCard; therefore **RILM** had to keep consistency with the whole package. For the programming of the scheduling issues, a high level programming language was required. Because the familiarity of the author with the BASIC programming environment, QuickBASIC was selected. The current version of **RILM** is at a prototype stage. The tools selected were utilized for developing the prototype and proving the concept of robot implementation logistics. The author believe that better programming environments can be use and future efforts can drive the current prototype into a faster and user-friendly commercial product.

### *Construction Robotic Equipment Management System (CREMS)*

**CREMS** is a project focused on the planning and management of robotic equipment selected for construction tasks (Skibniewski et al. 1989a, Russell et al. 1990). The implementation of robots in the construction industry has made necessary the development of a system capable of managing and scheduling a robot fleet within a company's project portfolio. The principal objective of **CREMS** is to develop a comprehensive construction robot management system, and it is divided into four major sections:

1. Analysis of construction tasks to evaluate how feasible is the use of the robot.
2. Analysis of the robot capabilities to perform the task
3. Economic analysis to determine if the use of the robot is cost effective, and
4. Scheduling and logistics for the optimal use of a robot fleet.

These four areas of analysis serve as a basis for dividing **CREMS** into a four module system:

1. Construction Task Analysis Module (**CTAM**)
2. Robot Capability Analysis Module (**RCAM**)
3. Robot Economics Evaluation Module (**REEM**)
4. Robot Implementation Logistics Module (**RILM**)

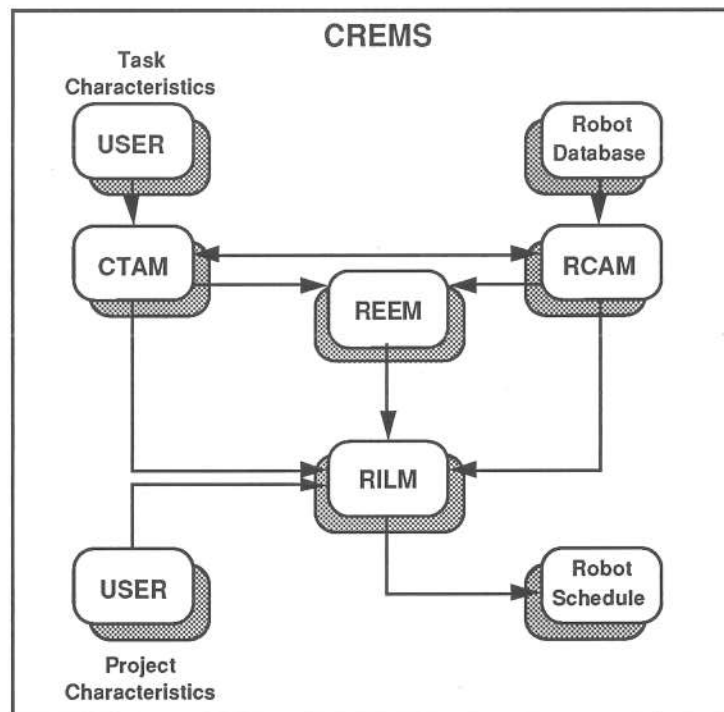
The interaction among the four modules is shown in Figure 1.1. These modules are described briefly next.

#### CONSTRUCTION TASK ANALYSIS MODULE ( *CTAM* )

This module gathers and analyzes all the relevant information regarding a specific construction task. The program analyzes, among others, resource constraints, productivity requirements, work quality, and characteristics of the work environment. The final evaluation of the module will decide if the task is suitable for robotic performance.

#### ROBOT CAPABILITY ANALYSIS MODULE ( *RCAM* )

Throughout the analysis it is necessary to examine the capabilities of a given robot or a group of robots for the execution of the given construction task. This module analyzes robot characteristics such as robot weight, power supply, payload, available tools, control and sensory systems. The required information is extracted from a robot database designed within HyperCard. Such a database works as a knowledge base used for matching these capabilities with the requirements of the construction task and the jobsite.



**Figure 1.1. Configuration of the Principal Modules of CREMS**

## ROBOT ECONOMICS EVALUATION MODULE (*REEM*)

This module produces a detailed analysis of expected robot profitability at the given jobsite location. Some of the factors included within the analysis are the cost for the non-robotic performance, robot operational costs, labor and time savings, and quality considerations.

## ROBOT IMPLEMENTATION LOGISTICS MODULE (*RILM*)

*RILM* is the module on which this report is focused. It develops a schedule for optimal use of the robotic equipment throughout the company's projects over one year. The scheduling functions include robot work time, resource allocation, transportation, set-up, and disassembly costs. The current version of *RILM*, described in this report, is at a prototype stage. It takes into account only information provided by the *REEM* module such as the robot and non-robot task performance cost and construction project characteristics. This version develops a feasible schedule for one robot. Future versions of *RILM* will include comprehensive resource allocation and transportation cost analysis.

Detailed explanation and development procedures of *CTAM*, *RCAM*, and *REEM* can be found in Progress Reports I and II to the Obayashi Corporation (Skibniewski et al. 1989b, Skibniewski et al. 1990b).

### *Background of Robot Implementation Logistics*

*RILM* evaluates a set of projects currently in the firm's portfolio which compete for the use of the robot and allocates the robot within the task-projects (equal subunits of a project) based on a feasible solution (Cornejo et al. 1991). The first three modules generate part of the information required by *RILM*, such as robot cost and non-robot cost for a certain activity. In addition, the user must provide the *RILM* module with project data such as earliest start times, latest start times, and crash times.

*RILM* is divided into four procedures described below and shown in Figure 1.1:

- Procedure 1: Compilation of a list of the projects.
- Procedure 2: Establishment of logistic parameters and acquisition of information from *CTAM*, *RCAM*, and *REEM*.
- Procedure 3: Preliminary scheduling for robot implementation logistics.
- Procedure 4: Resource leveling for robot implementation logistics.

In procedure 3 the preliminary scheduling is performed by means of two types of scheduling charts: 1) a Gantt chart defining the precedence relationships of projects listed in procedure 1, and 2) a downhill chart which provides input to procedure 3.

The program develops a downhill chart from a Gantt chart, which contains the original data collected by the computer from the user and the other three modules. The downhill chart is used to assign task-projects to different phases. The phases are the different horizontal levels of the downhill chart. In the first phase, called the critical phase, **RILM** allocates the robot to the task-projects in the critical phase in order to avoid idle periods. Only task-projects associated with the critical phase are performed by the robot. There is one prerequisite for consider the use of a robot on a project. It needs a minimum of three task-projects in the critical phase of the downhill chart to be considered a Minimum Task-Project Requirement (MTPR).

The critical phase analysis of **RILM** requires at least one task-project in the critical phase. If the project does not meet the MTPR requirement it is considered a Non-MTPR project. Projects not included in the critical phase are analyzed in a second phase analysis for non-robot activities. **RILM** executes resource leveling for the critical segment by shifting backward and forward, switching between phases, and crashing the project duration in order to increase the number of task-projects included in the critical phase or place the whole project in the critical phase if possible. The MTPR condition is calculated through the analysis of a profit / loss break-even point between the robot and human craft work, or is obtained from prior robot implementation experience.

Resource leveling results in the most economical use of the robot, maximizing the contractor's profit resulting from the utilization of robots across all projects currently in the firm's portfolio. The robot dispatcher must consider the net benefit of utilizing the robot across all projects at the same time, if feasible. Project activity shifting may be required to accommodate limited robot availability even at the cost of either crashing an activity performance time or delaying overall project completion, as long as the net profit to the firm remains positive when compared with the option of performing the tasks without the use of robots. Detailed explanation and development characteristics of **RILM** can be found in Progress Reports III to the Obayashi Corporation (Skibniewski et al. 1990c).

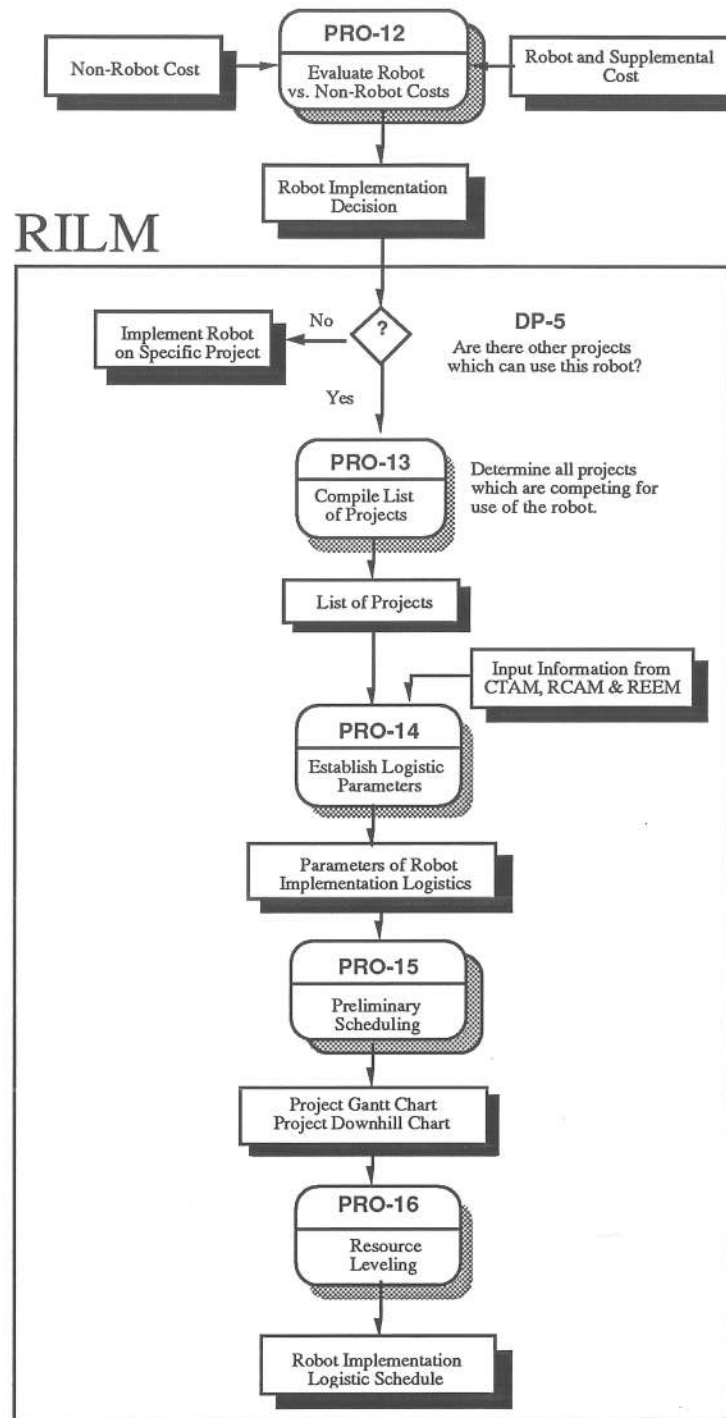


Figure 1.2. Conceptual Framework of RILM within CREMS.



## **Reader's Guide**

This Report is organized by sections and subsection as presented in the table of contents. For facilitating the reading of the report, figures and tables are included through the whole report. Appendix A has a glossary and definition of all the important terms utilized in this report for an easy comprehension of the work.

The first section, *Introduction*, presents the problem and the need identified in the industry for the resource allocation and scheduling of highly specialized construction equipment. The objectives as well as the methodology followed in this research is also presented in the first section.

The second section, *The RILM Module*, discusses the implementation logistics issues, the module architecture proposed for *RILM*, and the assumption considered for this first phase of the research.

*The RILM Software* is described in the third section. This section describes the configuration and the software utilized for this research. The flowcharts proposed for the development of *RILM* are also explained in detail within the software section.

The section four presents an *Example Application Session*, which can be regarded as the user's manual for operating *RILM*. The example takes the reader by the hand through each of the cards of the program. The cards are presented in the Appendix B. This section also includes a subsection in which the results obtained in the example are explained.

The section five has the *Conclusions and Recommendations* extracted from the research. At the end of the section the report includes recommendations for future investigation in the area, which looks very promising at this time.

## **Benefits of the Prototype**

The prototype performs project resource leveling independent of the task and the robot, even though the user has to define the initial project parameters, which are not available from the other three *CREMS* modules and are unique for the analysis within *RILM*. The prototype has proven its ability to handle the parameters of any project but only one robot for each analysis.

The application of the prototype can go beyond the robot scheduling. It can be used in today's construction projects for logistic implementation and scheduling of expensive construction and heavy equipment.

The different products in the market for cost analysis and resource scheduling only analyze the costs and scheduling for one machine. The advantage of **CREMS** is that it can analyze many task configurations for the non-robot option and compare the most favorable of them with the cost of the designated robot for the same task. The company has the opportunity to compare both options and make the right decision. That is what **CREMS** and **RILM** are all about, helping the company executives in the decision making process by helping them in the approach and analysis of the problem.

## **SECTION 2: THE RILM MODULE**

The variety of decision issues for robot assignment to construction sites are complex and includes task work volumes, task durations and sequences, job site locations and associated travel distances, robot setup, and operation and dismantling resources needed. As an aid in such decision, a decision support architecture for the performance of optimal robot assignment plans was presented in (Skibniewski et al. 1990a). The RILM architecture includes both theoretical methodology and formulations of robot implementation criteria.

### **Implementation Logistics Issues**

Once a given robot is selected to perform a specific construction task, the robot can be assigned to that task and dispatched. However, as it is frequently the case with a large construction company, more than one project site will compete for the use of the robot at any given time. Thus, considering the robot a scarce resource, timely decisions must be made based on a careful weighing of each assignment's advantages and disadvantages.

The study of implementation logistics involved a combination of three models: 1) mathematical model, 2) traditional scheduling techniques, and 3) heuristic-based approaches. The linear programming assignment modeling involves an establishment of a robot implementation cost objective function, subject to constraints representing the fact that any given robot can be assigned to only one project site at a given time. The traditional scheduling tools are highly applicable and relevant to the robot assignment issue. In particular, the Critical Path Method (CPM) combines the information on project task duration and task precedence relationships. These relationships result in the determination of critical activities and projects as well as allowable slack times available for optional activities. A heuristic approach, such as expert systems, offers the advantage of utilizing the experience of skilled robot and conventional construction equipment managers in making assignment decisions.

Combining these approaches requires the fulfillment of the objective of maximizing the the contractor's profit resulting from the utilization of robots in regard to all projects currently in the firm's portfolio. This implies that, in making an assignment decision, the robot dispatcher must take into account the net benefit of utilizing robot across all projects at the same time, if feasible. Thus, project activity shifting may be required to accommodate limited robot availability even at the cost of either crashing an activity performance time or delaying overall project completion, as long as the net profit to the firm

will be positive when compared with the option of performing the tasks without the use of robots.

### **Assumptions of RILM**

Like any prototype, *RILM* was developed under certain assumptions, which are described below, for proving its concept. *RILM* is a module for resource leveling and allocation, therefore it is not robot or task dependent. This first phase of *RILM* can allocate only one robot to one construction site at a given time assuming that the contractor has only one robot. *RILM* collects a listing of all project sites where use of a robot is economically justified. All projects and their associated tasks must have been screened previously by the cost feasibility analysis module, *REEM*. The task-projects are assumed to have equal initial duration. This condition is necessary with the current prototype version. It is assumed that the company using the implementation logistics module has all the required information about each project such as project duration, earliest and latest start times, crash time available, and incremental costs due to crashing or modifying the duration of each task-project.

The prototype is able to be executed on a standalone basis. A file created externally with the appropriate format and the required data (robot and non-robot cost for each project) can supply *RILM* with the necessary information for resource leveling and even for performing a sensitivity analysis.

*RILM*, at its prototype stage, performs only resource leveling for the critical phase. The second phase is only analyzed for utilizing time slots left by the critical phase resource leveling. The transportation, robot set-up and dismantling costs are included within the robot performance cost for a project; therefore the robot cost reflects the total costs that can be incurred by implementing the robot on a specified site.

## Architecture of RILM

The **RILM** prototype was developed in the HyperCard environment to maintain consistency with the other modules of **CREMS**. One of the features of HyperCard is the capability to link easily with programs written in other programming languages. This capability is utilized by **RILM** to use the QuickBASIC program for most of the MTPR and Non-MTPR analysis.

The implementation of QuickBASIC shortened the programming scripts and made the program much faster for executing subprograms and searching for internal data-files. The configuration of **RILM** is based on five blocks, as shown in Figure 2.1. The HyperCard script controls the analysis, interacts with QuickBASIC for executing specific processes, and creates data-files. Some files are created for the exclusive need of the QuickBASIC programs, others for HyperCard only, and the rest are used by both programs.

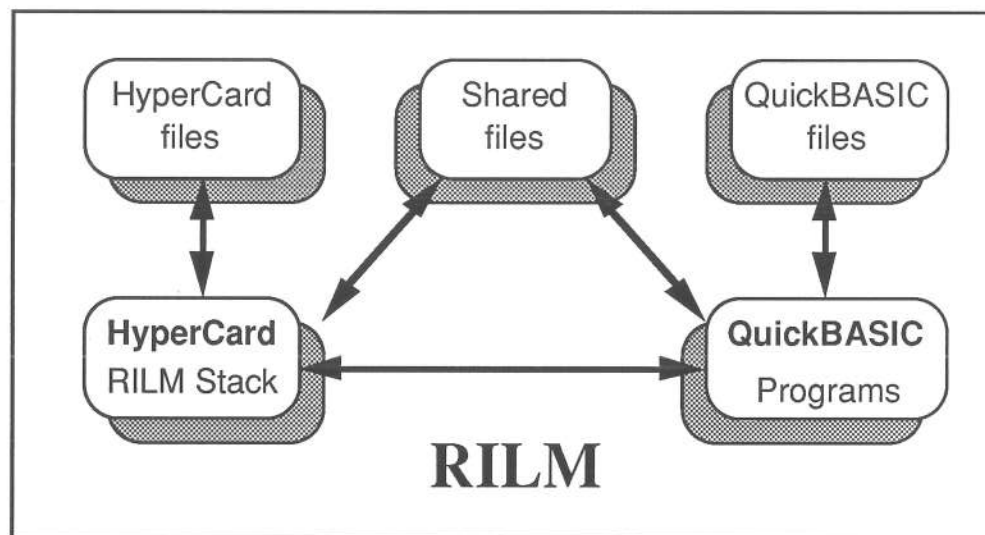


Figure 2.1. Configuration of RILM

**RILM** is basically composed of four processes, as shown in Figure 1.2. **PRO-13** collects the information from data files of the projects and creates a list of projects. The next process, **PRO-14**, establishes the logistic parameters with information collected from a file containing the results of executing **CTAM**, **RCAM**, and **REEM** once for each project. The logistic parameters provide input for the preliminary scheduling and the

generation of a project Gantt chart and a project downhill chart. The preliminary scheduling is executed in process **PRO-15**, in which the projects are located in different phases. The result of **PRO-15** is represented in a downhill chart. **PRO-16** is the most important part of *RILM*. The resources are leveled and allocated for robot implementation logistics. The process generates a final table and a final downhill chart for simplifying the understanding of the table and the final allocation of the robot. **PRO-16** minimizes the idle period of the robot and levels the use of labor crafts. Figure 2.2 shows the programming flowcharts of **PRO-16** and is explained in the following paragraphs. The Non-MTPR process alone is illustrated in Figure 2.3.

The flowchart for **PRO-16** is divided into six submodules: 1) Critical Segment Submodule, 2) Non-MTPR submodule, 3) Project Selection Submodule, 4) Selected Project Submodule, 5) Project to Right Submodule, and 6) Second Phase Leveling Submodule.

The *Critical Segment Submodule* analyzes the projects until the identification of the critical segments is completed. A critical segment is determined by a project with a fixed start date and high decision-maker preference, therefore the critical project is part of the critical phase. The critical segments are the segments between restricted projects, the beginning of the year, or the end of the year. This submodule also identifies the task-projects that are part of the critical phase and after this analysis also checks for the MTPR condition of each project. If a project does not fulfill the MTPR requirements, it will be analyzed on an independent basis in the *Non-MTPR Submodule*.

The *Project Selection Submodule*, the *Selected Project Submodule*, and the *Project to Right Submodule* perform the MTPR analysis. The *Project Selection Submodule* analyzes the preference order of the projects. The preference rules are presented in Table 2.2. At the user's discretion the ranking can be modified for other reasons such as robot profitability in a project. After checking the ranking, the submodule selects the task-projects of the project with highest decision-maker preference.

The *Selected Project Submodule* considers the project selected in the Project Selection Submodule and attempts to place all of its task-projects into the critical phase by shifting, crashing, or rearranging them. This is accomplished by checking for crash time availability and calculating the associated incremental cost. The incremental cost is an additional cost of the project due to crashing the task-projects duration a certain amount of time in order to shift all of the task-projects into the critical phase. The incremental cost is compared with the robot work profit (RWP) of the project under consideration.

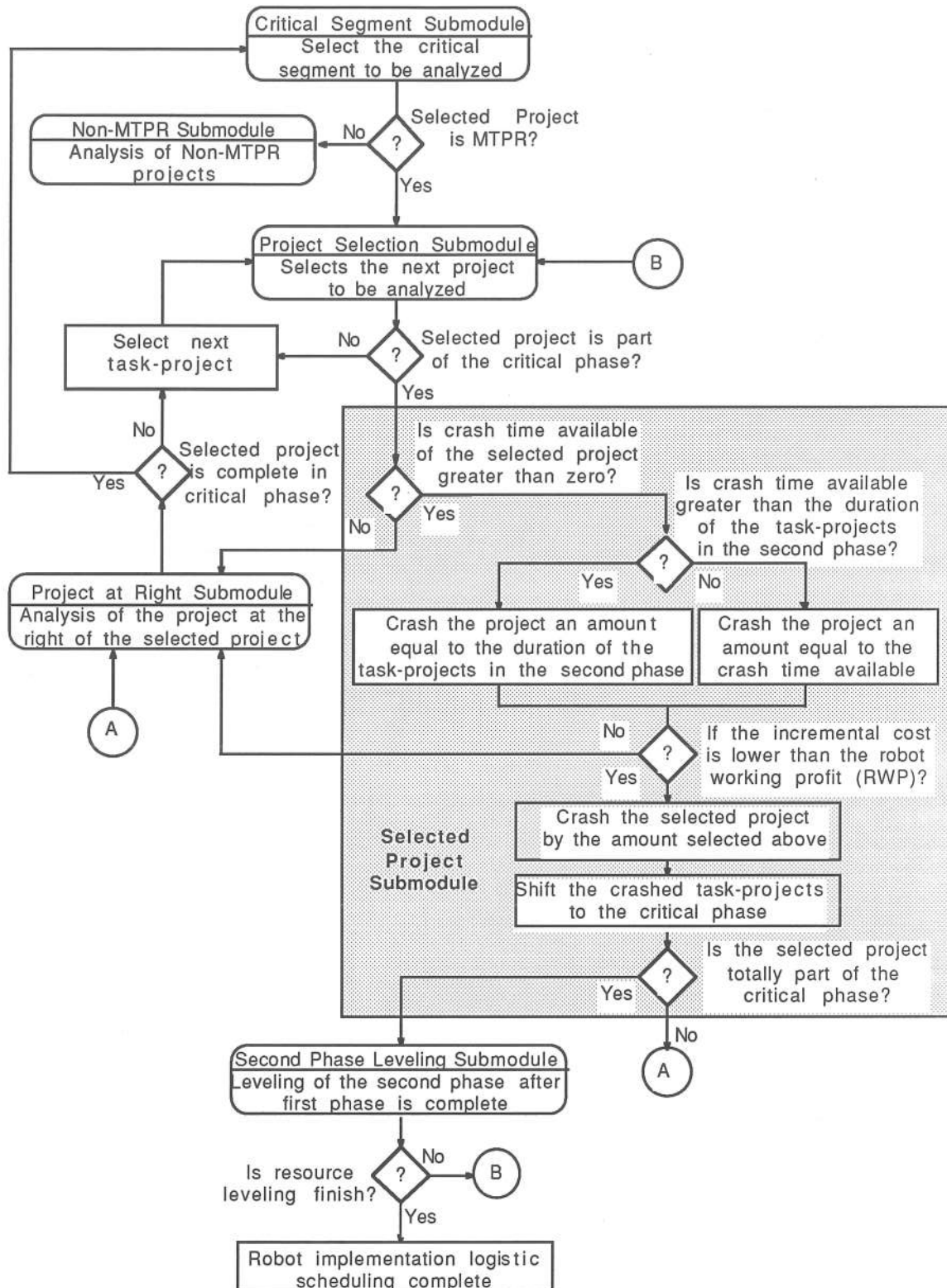


Figure 2.2. Resource Leveling Flowchart (PRO-16)



If the Incremental Cost is lower than Robot Work Profit then the project is crashed or its duration is reduced and the task-projects rearranged. If one of the decision points is not true or after this analysis the project is still not 100% part of the critical phase, the analysis proceeds to *Project at Right Submodule*. This submodule attempts to rearrange the project at the right of the current project in order to leave enough time for shifting the selected project into the critical phase.

The last submodule, *Second Phase Leveling Submodule*, analyzes and rearranges the levels above the critical phase in the downhill chart. It covers vacant scheduling slots left by the prior submodules during the resource leveling analysis.

When compared to the six established submodules for the resource leveling process, the Non-MTPR Submodule is the most complex in nature, therefore a section of this report is dedicated to the Non-MTPR process alone. When a project does not fulfill the minimum conditions required for the use of the robot, such as having less than three task-projects (Preference Rule PR2, see Table 2.2) as part of the critical phase, a HyperCard script transfers the command to a QuickBASIC program. The Non-MTPR QuickBASIC program is also divided into subroutines for facilitating the understanding and programming of this submodule. This program analyzes the Non-MTPR projects and presents different solution options, to let the user decide among the most economical and feasible options. The process and flowchart diagram of the Non-MTPR Submodule is shown in Figure 2.3. A detailed description of the Non-MTPR submodule is presented in the next section, The *RILM* Software.

Preference Rule No. (1)	Preference Criterion (2)	Example Rule (3)
PR1	Robot Work Profit (RWP) in Comparison with Non-Robot Work: $RWP = \frac{(\text{Non-Robot Cost}) - (\text{Robot Cost Plus Supplementary Cost})}{\text{Number of Task\_Projects Involved in a Project}}$	if $RWP \leq \text{¥ } 100,000$ per project then do not use robot
PR2	Number of Robotable "Task_Projects" Involved in a Project	if number of "task_projects" < 3 then do not use robot
PR3	Total Project Duration	if total project duration $\geq 3$ months then consider sharing robot with other projects simultaneously
PR4	Robot Effectiveness = $\frac{(\text{Total Area}) - (\text{Area Not Executed by Robot})}{\text{Total Area}}$	if robot effectiveness $\leq 70\%$ then give preference to another "task_project"

Table 2.2. Example Preference Criteria and Rules for Basic Robot Scheduling.

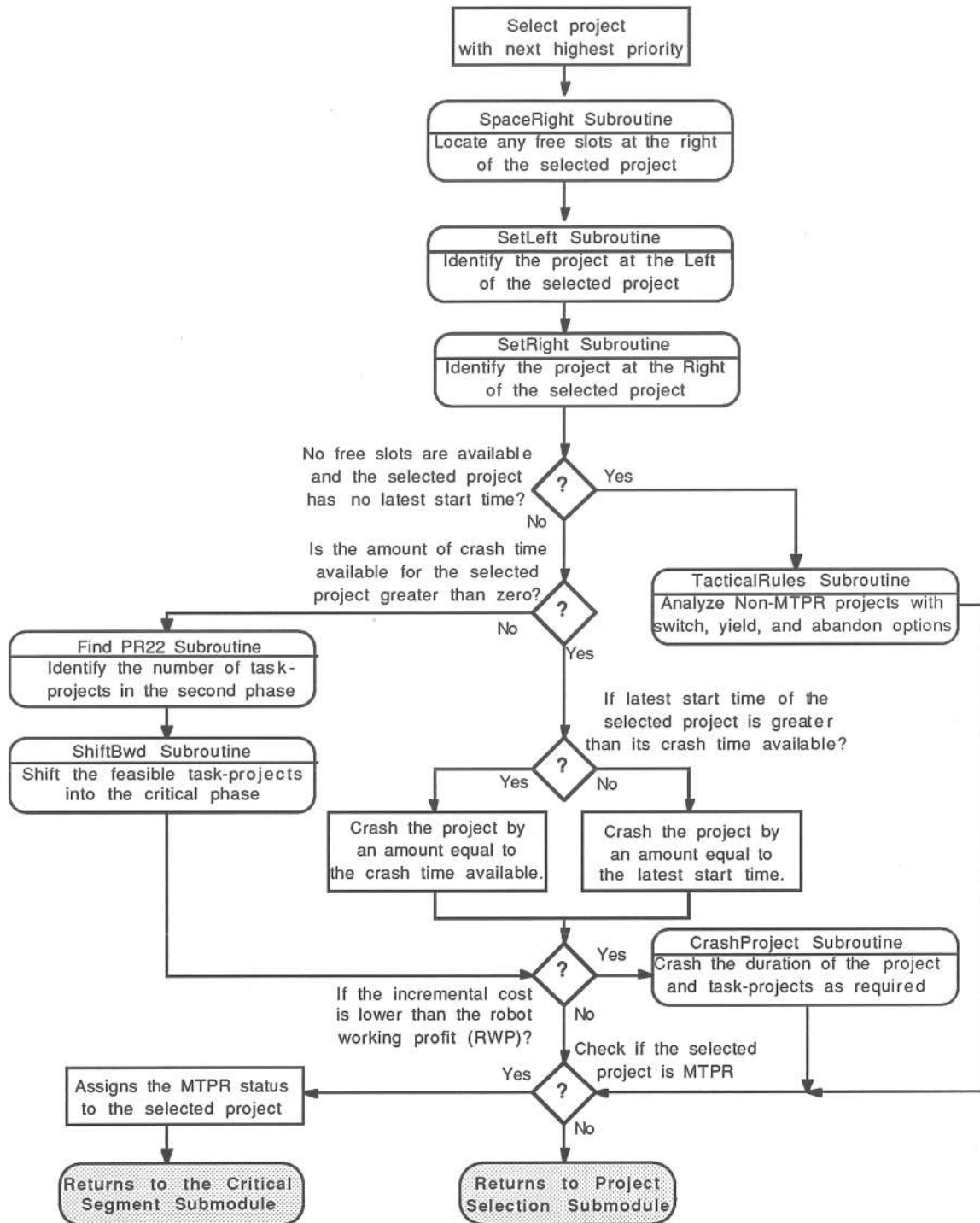


Figure 2.3. Non-MTPR Submodule

## SECTION 3: THE RILM SOFTWARE

This Section is dedicated to the description of the software of **RILM**, including the configuration and programming aspects. The flowcharts for the major processes are presented and explained in detail.

### Software Configuration

As mentioned in Section 2, the main body of **RILM** was developed in HyperCard to maintain consistency with **CREMS**, but the computationally complex submodules were written in QuickBASIC. The HyperCard and QuickBASIC applications, used for the development of **RILM**, combined with the graphic capabilities of the Apple Macintosh microcomputers made a perfect combination for a user friendly software. The selection of these two programming environments was done initially because of the availability of commercial packages, the short time required to learn the languages, and above all, the major goal of the project was to *prove the concept* of RILM rather than doing a programming exercise. Future versions can be reprogrammed in other programming languages with better capabilities.

HyperCard programs are made out of cards and these cards are put together in stacks. Information contained in the cards can be transferred to other cards through a transparent flow. Some of the reasons for using HyperCard are: 1) Excellent database management capabilities, 2) Ease of interfacing multiple data files with external programs, 3) Simplicity to use for a relatively inexperienced construction field personnel, 4) Rapid prototype development, and 5) Commercial availability of software. The major disadvantage of HyperCard is the lack of high level programming capabilities; therefore the interface with a high level programming language was required for the scheduling programming issues.

Tests were made to confirm the easy interaction between HyperCard and **RILM**. The major tests executed were about external data files interaction and graphic capabilities. QuickBASIC proved to be an easy understandable programming language with fast execution times compared to HyperCard.

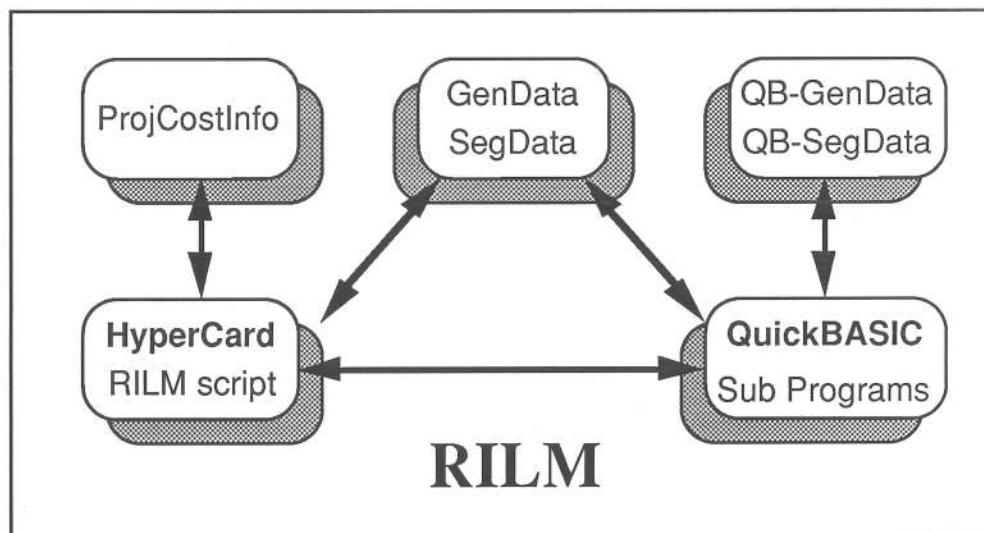
The external data files are the media selected for the exchange of information within the different submodules. Between HyperCard cards, the flow of data is handled through the cards. Due to the presence of another programming environment, QuickBASIC, the data is transferred by external data files.

The five blocks in which the configuration of **RILM** is divided are shown in Figure 3.1. The interaction of files follows the arrow lines between the blocks. As represented in Figure 2.1 the file blocks are HyperCard files, shared files, and QuickBASIC files. Each of these blocks summarizes the principal function of the files. Table 3.2 illustrates the files already classified by name, type, and description. These files are the actual files that **RILM** creates for its use. The type identification for each file indicates the location of each file within the blocks diagram in Figure 3.1.

The main script or program is written in HyperCard. The script uses HyperCard cards to present the results on the screen and to call the QuickBASIC programs. The programming within HyperCard involves only the opening and closing of files and the flow of data through the module. The nature of this application does not provide high programming capabilities; therefore the interface with a programming language, such as QuickBASIC, was necessary for the scheduling programs.

<u>File Name</u>	<u>Type</u>	<u>Description</u>
RILM script	HyperCard Stack	Main Script
QB-PRO15	QB Subprogram	Preliminary Scheduling
QB-Seg-PRO16	QB Subprogram	Critical Segment Information
QB-PRO16	QB Subprogram	MTPR and Non-MTPR Scheduling
QB-Graph-Gantt	QB Graph Subprg.	Gantt Chart
QB-Graph-GenData	QB Graph Subprg.	Downhill Chart
QB-Final-Graph	QB Graph Subprg.	Comparison of Preliminary Downhill Chart and Final Downhill Chart
ProjectCostInfo	Data File	Project Costs Data
GenData	Data File	Project General Data
SegData	Data File	Critical Segment Data
QB-GenData	Data File	Modified Project Data
QB-SegData	Data File	Critical Segment Data with MTPR Status

**Table 3.1 Description of Files**



**Figure 3.1 RILM Configuration and Files Interaction**

## RILM Flowchart within HyperCard

The main script of RILM receives data from the first three modules of CREMS and is transferred to RILM through external files. The flowchart of RILM illustrating the HyperCard cards, the QuickBASIC graphic programs, and the QuickBASIC programs is presented in Figure 3.2. The screen copies of the RILM cards are shown in Appendix A.

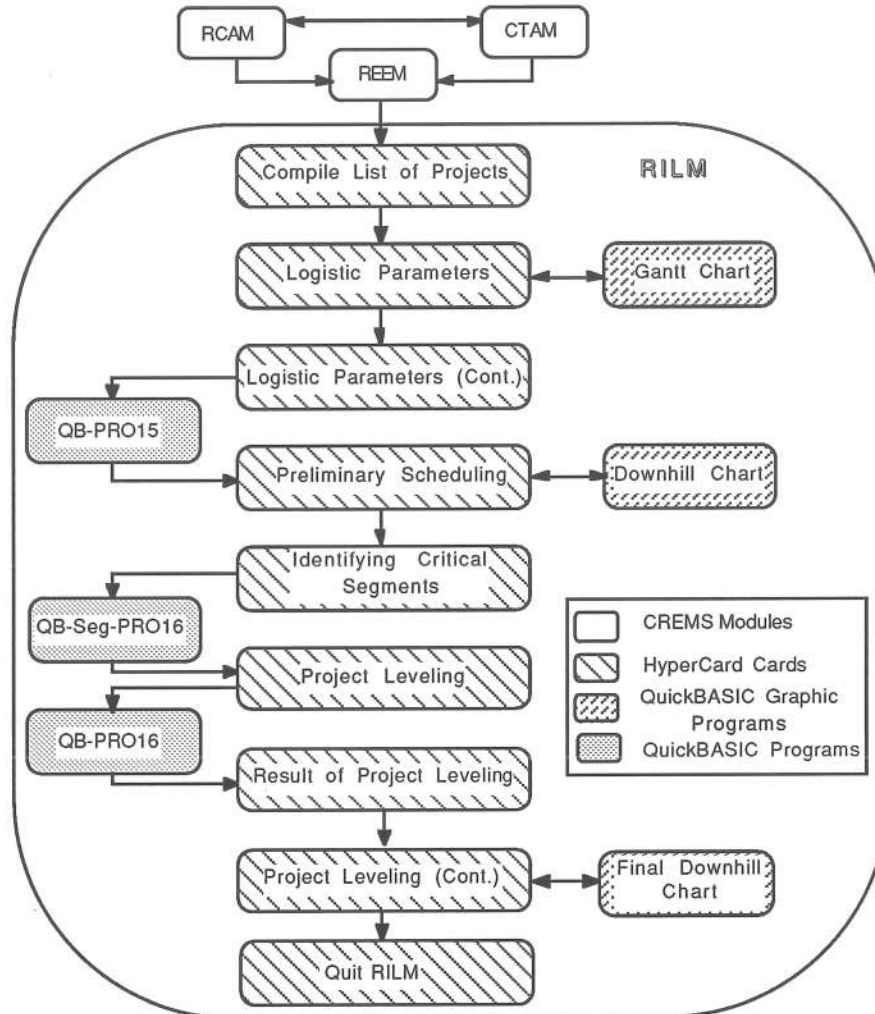


Figure 3.2. RILM Flowchart within HyperCard

The first card, *Compile List of Projects* (See Appendix B, Figure B.1), compiles a list of projects competing for the use of the robot. The task information about each of the projects is entered in the next card named *Logistic Parameters* and a Gantt chart helps to visualize the list of projects (See Appendix B, Figure B.2(a)). Project cost information

summarized from *REEM* is shown in the next card, *Logistic Parameters (Cont.)* (See Appendix B, Figure B.3). RILM activates at this time the QuickBASIC program, *QB-PRO15*, for preliminary scheduling and shows the results in the next HyperCard card, *Preliminary Scheduling* (See Appendix B, Figure B.5(a)). The preliminary scheduling translates the Gantt chart (See Appendix B, Figure B.2(b)) into a downhill chart and analyzes the preferences of each of the projects as explained below (See Appendix B, Figure B.5(b)). The critical segments are identified in the card, *Identifying Critical Segments* (See Appendix B, Figure B.6), and the program *QB-Seg-PRO16* analyzes the MTPR status of each project involved in the critical segment and records the information on a file, which is presented in the next card, *Project Leveling* (See Appendix B, Figure B.7). This card activates the most important program of the module, *QB-PRO16*, in which the projects are logistically leveled and the solution is presented in a table, (See Appendix B, Figure B.8(a)). A downhill chart aids the user in visualizing the results in the table (See Appendix B, Figure B.8(b)). The *Project Leveling (Cont.)* card (See Appendix B, Figure B.9) allows the selection of another critical segment and the *Quit RILM* card directs the user to restart or exit the program.

### **Preliminary Scheduling (QB-PRO15)**

This QuickBASIC program transfers the information shown in the initial Gantt chart into a downhill chart. For defining which task-project goes into the critical phase in case of overlapping, the program analyzes the preference assigned to each project and the task-project with the least preference goes to the critical phase. The task-projects are arranged with the one with the least preference at the bottom and the one with the highest preference at the top of the list of projects to be executed, and the process continues.

### **Critical Segment Analysis (QB-Seg-PRO16)**

The *QB-Seg-PRO16* program obtains the information about which critical segment was selected and retrieves the information about the projects involved in the selected critical segment. The program searches for the information it needs in the *QB-GenData* file and after analyzing each project it returns the information of the projects that are active in the segment with their MTPR status. The MTPR status indicates whether the project fulfills the MTPR requirements, if it is a Non-MTPR, or if it has no task-projects in the critical phase. The project is considered active for the analysis if it has at least the MTPR or Non-MTPR status.



## Resource Leveling Procedure (PRO-16)

As mentioned in the previous section, PRO-16 is divided into six submodules: 1) Critical Segment Submodule, 2) Non-MTPR Submodule, 3) Project Selection Submodule, 4) Selected Project Submodule, 5) Project to Right Submodule, and 6) Second Phase Leveling Submodule. The flowchart PRO-16 in Figure 3.3 illustrates the inter-relationships between these six submodules and describes in detail the algorithm of the most significant Selected Project Submodule in the PRO-16. The algorithm of *Non-MTPR Submodule* is illustrated in Figure 3.5 and described in the next section.

Before describing the details for each submodule in Figure 3.3, two additional terms need to be defined. First, projects in which the earliest start time is equal to the latest, (i.e., with no "total float time"), are designated as "fixed projects." Second, a fixed project which is in the critical phase is called a "restricted project." Since a restricted project possesses high preference and the fixed start point of the project duration, it may force the user to adjust the competing task-project schedules associated with the predecessor or successor projects against its own constraints. Therefore, the clarification of where the restricted project is located in the critical phase is a prerequisite for resource leveling in the other remaining projects.

The *Critical Segment Submodule* receives the information from QB-PRO15 and QB-Seg-PRO16 with each segment's information. The submodule identifies the segment and the task-projects that are part of the critical phase. A detailed flowchart for each submodule is illustrated in Figure 3.4. If the project selected fulfills the MTPR requirements, the *Project Selection Submodule* orders the projects by preference and selects the project with the highest priority. The project selected with the least preference is the project with the highest priority. If the project selected has at least one task-project in the critical phase the program goes to the *Selected Project Submodule*. If this is not the case, the program selects the project with the next priority.

In order to improve the level robot utilization, after processing a Non-MTPR project, the *Selected Project Submodule* focuses on further increasing the number of task-projects in a critical segment by utilizing the available crash times. Therefore, first and second decision points in this submodule check for crash time availability involved in the selected project (see Figure 3.4). Before the third decision point, both parameters of incremental cost and robot working profit (RWP) are calculated.

RWP is defined in the following formulation:

$$RWP = \frac{(\text{Non-Robot Cost}) - (\text{Robot Cost Plus Supplementary Cost})}{\text{Number of Task-Projects Involved in a Project}}$$

If the incremental cost is lower than RWP, the selected project is crashed until the required number of the task-projects exists in first phase. Then the task-projects in second phase are shifted to the vacant scheduling slots in critical phase that are provided by crashing the selected project.

The *Project at Right Submodule* checks whether the project at the right of the selected project has latest start time and crash time available. If either of both are negative then none of the prior changes are made effective and the analysis starts again with the selection of the project with the next highest priority. The submodule then sets the crashing durations for the project and verifies if the incremental cost is lower than the robot working profit. If the last decision point results in a positive answer, the submodule crashes the project and actualizes its new starting date.

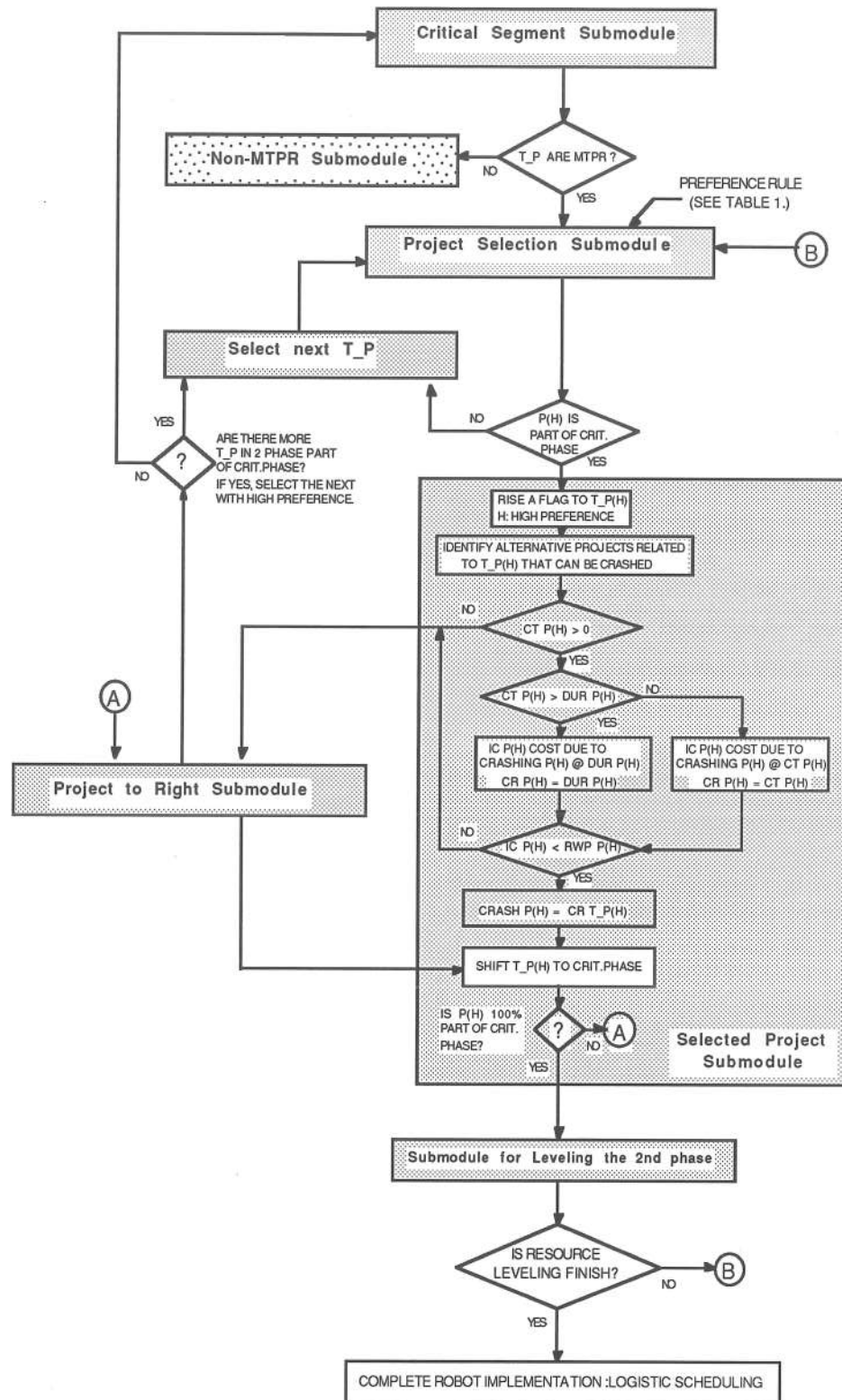


Figure 3.3 Detailed Resource Leveling Flowchart (PRO16)

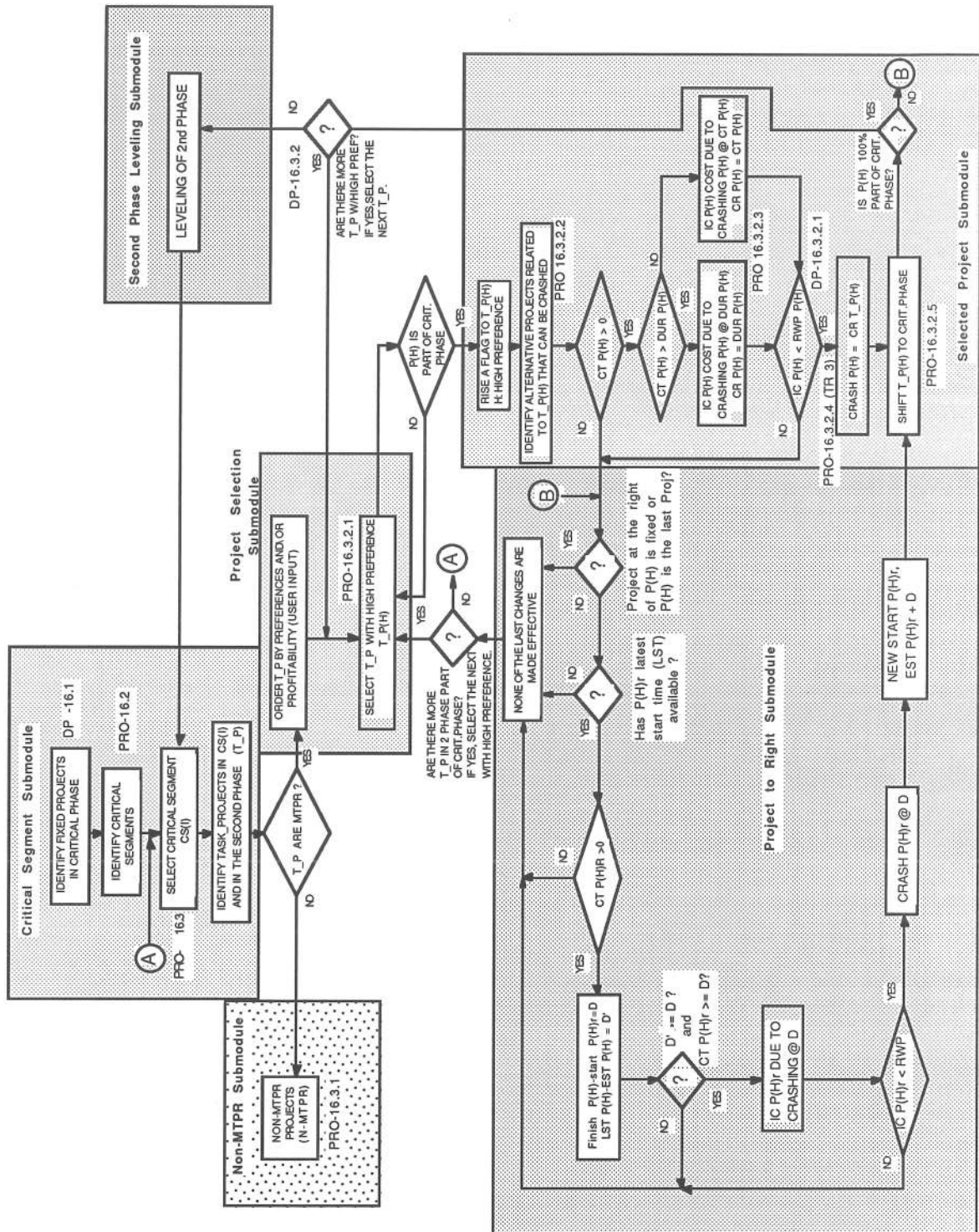


Figure 3.4. Detailed MTPR Process Flowchart

### **The Non-MTPR Submodule**

The Non-MTPR Submodule is subdivided into subroutines (See Figure 3.5 and Table 3.2 for a definition of terms). The first subroutine to be executed is the *SpaceRight Subroutine* in which any free slots at the right of the selected projects are available. These vacant slots can be there because of previous arrangements of higher preferred projects or idle periods found in the initial downhill chart. The next steps are the *SetLeft* and *SetRight Subroutines* in which the physical projects at the left and at the right of the selected project, respectively, are identified from the latest rearranged downhill chart. If no free slots are available, the selected project cannot be shifted forward, and the program executes the *TacticalRules Subroutine* in which it suggests three alternatives. The selection of an alternative overrules the basic assumptions of RILM and gives the Non-MTPR project the MTPR status based on the approval of the user. If there are any free slots at the right of the selected project, the program checks also for crash time available. If the last decision statement is positive, then the next decision depends on whether the latest start time available is greater than the crash time available. This step indicates the program the amount of time by which the project will be crashed. If there were not any crash time available, then the *FindPR22* and *ShiftBwd Subroutines* are executed. The number of task-projects left in the second phase is identified and the program shifts to the critical phase those task-projects that match with the duration of the vacant slots. The task-projects of the selected project that were not able to be shifted to the critical phase remain in the second phase but are shifted forward to the locations of the task-projects that were shifted. These changes are reflected in an increment to the cost. If the incremental cost is lower than the robot working profit (RWP), then the changes are executed within the *CrashProject Subroutine*. If the incremental cost is higher than the RWP, then no changes are made effective. The program checks the new status of the project. If the status is still Non-MTPR, then it returns to the Project Selection Submodule in the main script and selects the next project with highest preference. If the project is MTPR then it returns to the Critical Segment Submodule in which the analysis for an MTPR project continues within the main script.

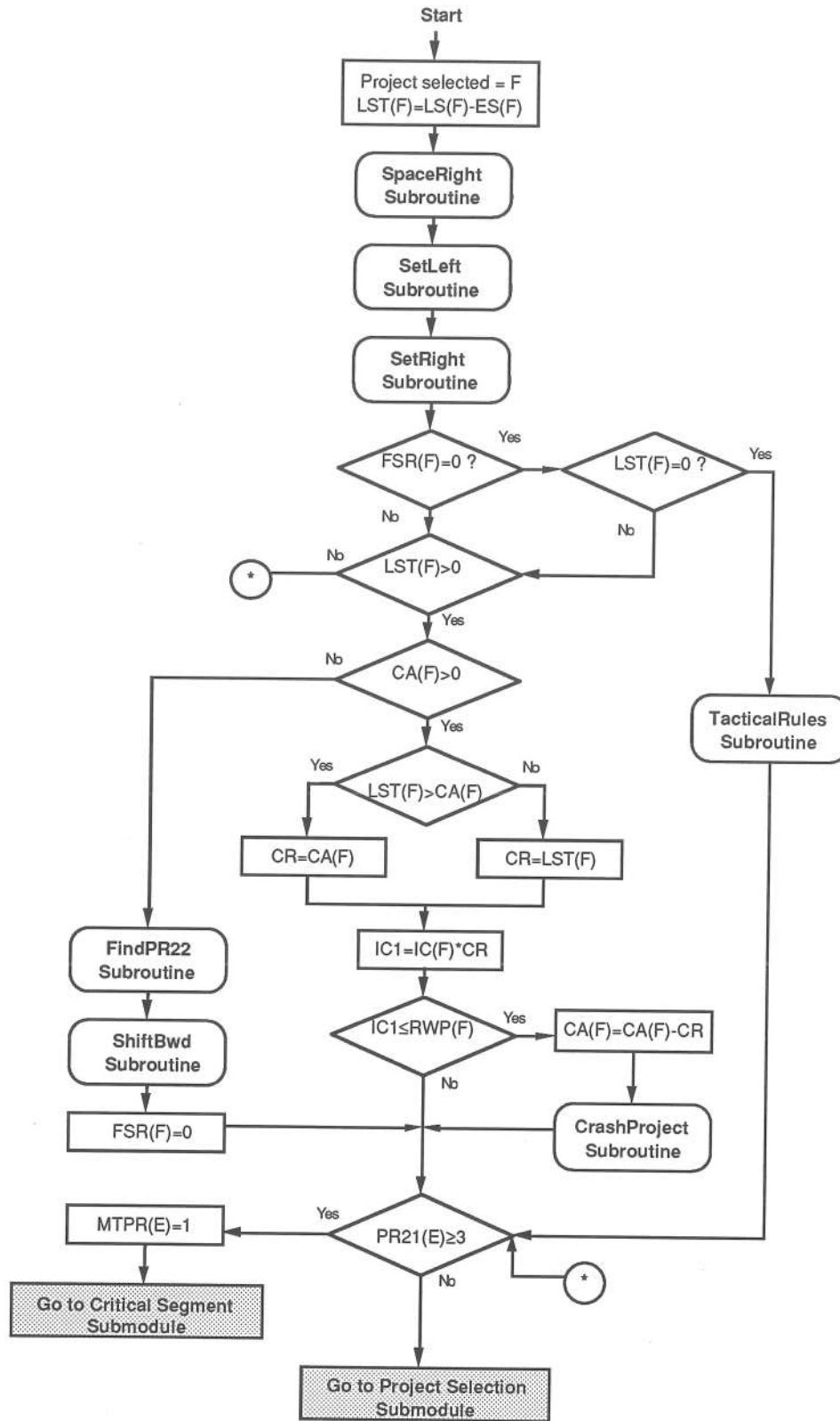


Figure 3.5. Non-MTPR Process Flowchart

MTPR	: Minimum task project requirements
N-MTPR	: Non-MTPR projects
FP (i)	: Fixed project i
P (i)	: Preference of project i
CS (i)	: Critical segment i
P(H)	: Project with high preference or profitability
P(H)r	: Project located at the right of P(H)
CT (i)	: Crash time for project i
IC (i)	: Incremental cost due to crashing project i
DUR (H)	: Duration of task-project of P(i), in second phase
OPP(i)	: Opportunity loss for project i (Non-Robot Cost)-(Robot Cost + Supplementary Cost)
RWP(i)	: Robot work profit for project i $RWP = \frac{(\text{Non-Robot Cost}) - (\text{Robot Cost} + \text{Supplementary Cost})}{\text{Number of task-projects involved in a project}}$
CR (H)	: Amount of time to be crashed for P(H)
EST (i)	: Earliest start time for project i
LST (i)	: Latest start time for project i
START (i)	: Start date for project i
FINISH (i)	: Finish date for project i

**Table 3.2. List of Variables in Flowcharts**



## SECTION 4: EXAMPLE APPLICATION SESSION

This section can be regarded as the user's manual for operating *RILM*. The first part of the section explains the installation of *RILM* on an Apple Macintosh microcomputer, the second part presents an example session described step by step, and then the results obtained are explained in this section.

### Installation of the *RILM* Module

*RILM* programs are contained on one 3 1/2" floppy disk labeled *RILM-QB*. The *RILM* prototype runs on Apple Macintosh microcomputers under MAC-OS version 6.0 or higher. The Mutifinder has to be installed and activated. The HyperCard and Microsoft QuickBASIC applications for Apple Macintosh Systems must also be on the Macintosh to be ready to run *RILM*. If corrections were necessary to a QuickBASIC program, the program corrected should be compiled in the *RILM* folder.

The *RILM* folder contains one folder with the original QuickBASIC programs called "*QB-RILM Programs*", the HyperCard Stacks, working files, and the QuickBASIC applications.

Open the *CREMS* folder that has a copy of the original *CREMS*. Insert the diskette with *RILM*, open the *RILM* folder, and select all the items in the *RILM* folder. Copy all the items to the *CREMS* folder.

Open the HyperCard folder containing the HyperCard application, and duplicate the HyperCard Stack and the HOME Stack. Transfer the duplicated stacks into the *CREMS* folder. Figure 4.1 shows the contents of the folder *CREMS*.

This version of *RILM* can be run without the *CREMS* stacks but cannot run without the HyperCard application and the HOME stack.

For running *RILM* only, click twice on the *RILM* icon.

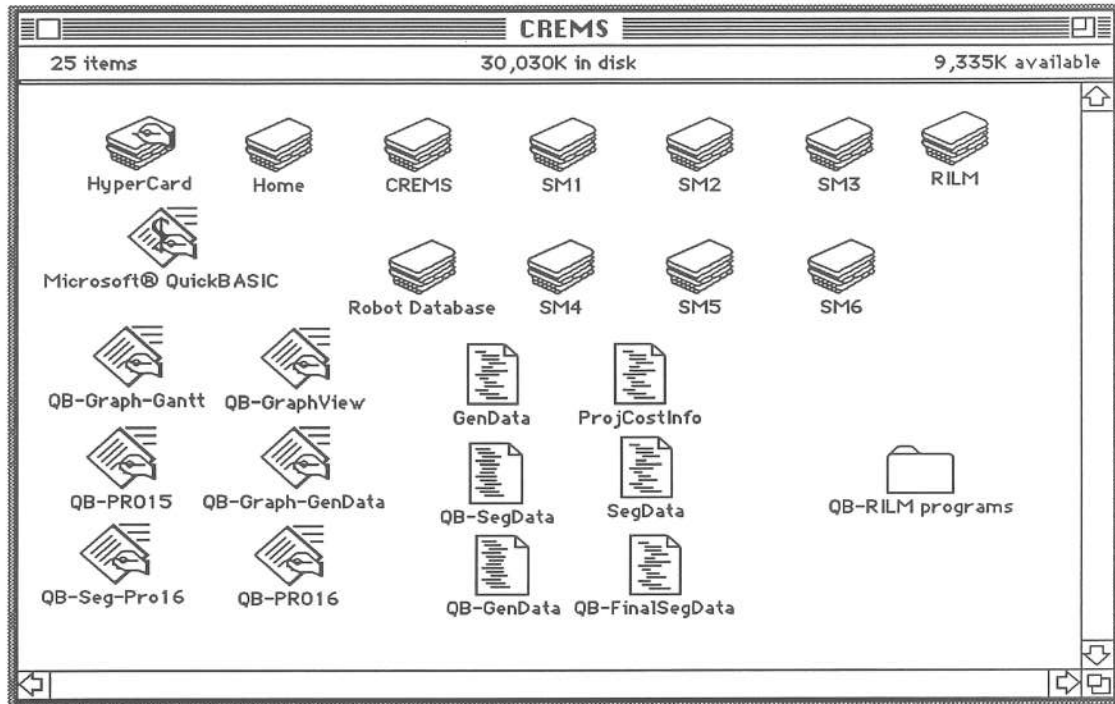


Figure 4.1. Contents of the final *CREMS* folder

### Example Application of *RILM*

The current version of *RILM* is a standalone prototype that works with data for an eight project example. *RILM* data for the projects include the task-project codes, earliest and latest start times, project durations, preferences, and available crash times. The preference order in this example was determined by the number of task-projects in each project. Figure 4.2 shows the data on a Gantt chart. In the example session the projects are numbered with a numeric code system but the task-projects have the codes as shown in Figure 4.2.

Some of the cards in *RILM* allow the user to enter data in scrolling fields that can handle more than eight projects. To scroll the fields, click on the arrow for the desired direction. The scrollbar between the arrows will not scroll the fields.

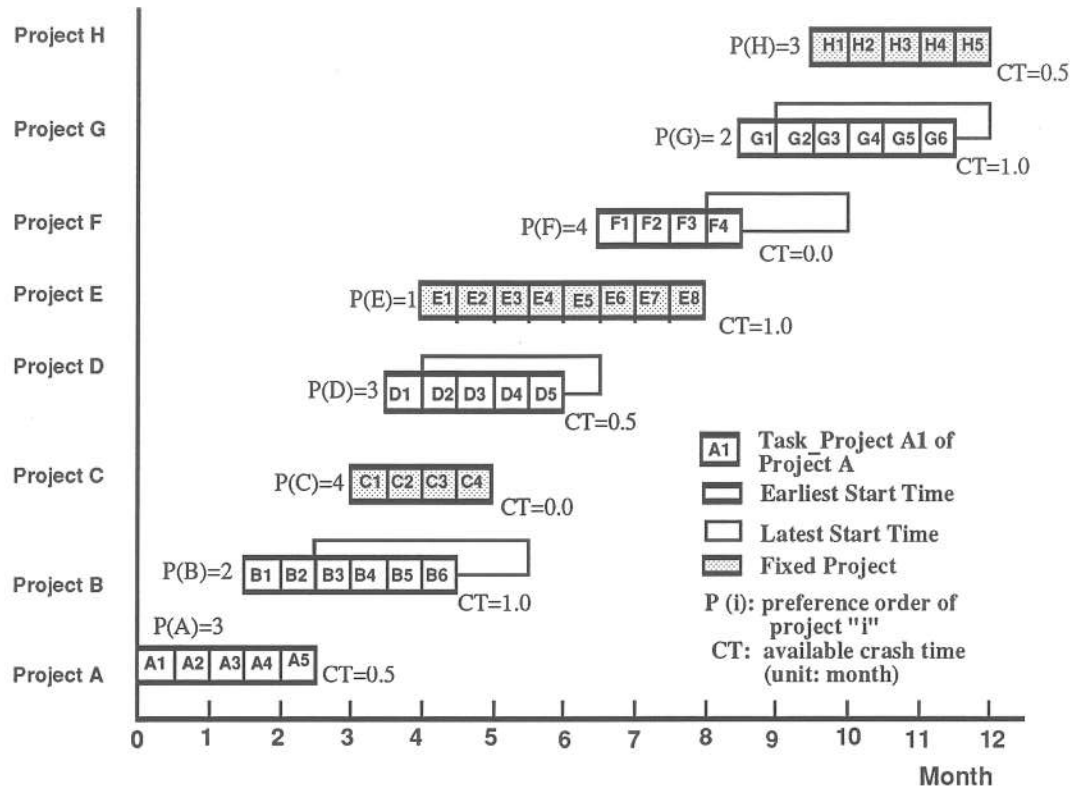


Figure 4.2 Project Gantt Chart for Basic Scheduling

For all the cards the Continue button causes the program to move to the next card, the question mark '?' button opens a message card, and the arrow button returns to the first card of **CREMS**, which is called the HOME card.

The second card, *Logistic Parameters*, receives data shown on the Gantt chart in Figure 4.2. The F in the Fixed Project column indicates that a project is fixed, thus its earliest and latest start times are equal.

The third card controls **PRO-14**. This card collects information about each project from the other three modules of **CREMS**. This card appears empty because the data are already entered for this version of the prototype. The value of Robot Working Profit was entered manually using arbitrary Robot and Non-Robot Costs.

When the Continue button is clicked, the *QuickBASIC* program *QB-PRO15* is executed and the screen will blink twice after a period of time. Then a card with a Continue button appears. The Continue button serves as a connection to the next card, *PRO-15*. When the downhill chart on card *PRO-15* is clicked, a *QuickBASIC* program is

executed. This program transfers the information contained in the card into a graphic representation, as shown in Figure 4.3. The Print Graph button allows the user to get a printout of the screen through appletalk. This convention was adopted for all the graphic charts.

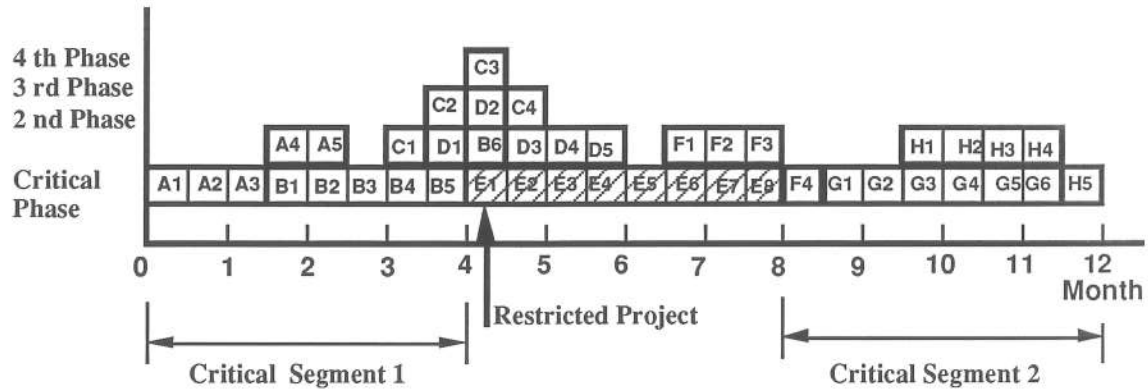


Figure 4.3 Project Downhill Chart for Basic Scheduling

The *Order* field shows the order of the task-projects starting from day zero, and the *Phase* field shows the location of each task-project in the horizontal level.

The next card, *PRO16.1*, identifies the critical segments and the restricted projects between the critical segments. The critical segment to be analyzed is automatically selected but the user is able to modify the selection. Clicking the OK button, results in execution of the QuickBASIC program *QB-Seg-PRO16*. The screen blinks twice and returns to the card *PRO16.1*. The OK button disappears and the Continue button appears. The user clicks the Continue button to proceed to the next card, *Project Leveling, PRO-16.3.2*.

The card, *Project Leveling*, summarizes information for each project contained in the selected critical segment. The MTPR column has three options: 1) MTPR, 2) Non-MTPR, and 3) Not in Critical Phase (No in C.Phase). The projects are ordered from lowest to highest preference. The last column indicates task projects, if any, that are in the critical segment. In the future this card will allow the user to change the preference of the projects without going back to the first card and running the prototype again. Only the OK button is shown at first. The user must click the OK button to confirm the chart. Then the Continue button is shown. When it is clicked, the QuickBASIC program *QB-PRO16* is executed.

If some of the projects are Non-MTPR and cannot be rearranged by the program, the QuickBASIC program opens a screen that shows information about the projects and possible solutions. Three buttons at the right side of the screen allow the user to view the charts and proceed with the analysis. The top button is clicked to show a possible solution after clicking one of the solution buttons on the left side of the screen. The middle button is clicked to accept the changes and proceed with the analysis. The bottom button causes two graphs to be displayed. The graph in the top half of the screen is the original downhill chart. The other graph shows all of the changes made in other critical segments up to that point.

The next card, *Result of Resource Leveling*, shows a chart with the result of the resource leveling analysis. It also includes a downhill chart button for displaying the result. The downhill chart summarizing the result is presented in Figure 4.4.

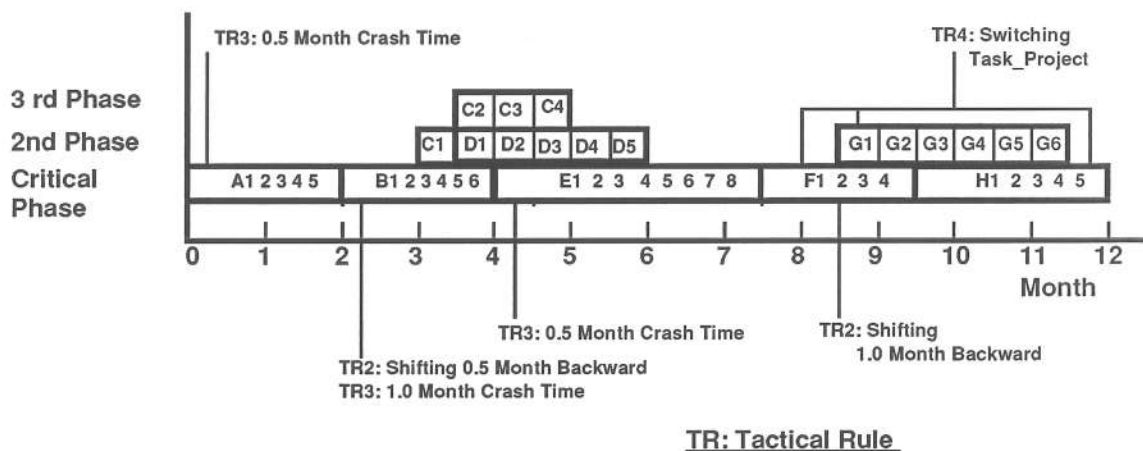


Figure 4.4 Result of Resource Leveling

The last card of RILM gives the user the options of proceeding to another robot or ending the analysis. If the segment selected at the beginning was the first one, then this card will select in the screen the second segment for approval of the user. If the segment selected was the last one, the only option left is finishing the program or reviewing the result of the resource leveling. To end the session, the user pulls down the file menu and clicks on the Quit.

## **Interpretation of Results**

The result of resource leveling analysis with the switch option is illustrated in Figure 4.4. A similar graph is obtained within the program by selecting the downhill chart button in the last card of *RILM*. This option presents two downhill charts. The chart at the top is the result of preliminary scheduling and the one at the bottom is the result of *RILM* at that stage of the analysis. If the selected segment is the last one, then the graph shows the final result.

The first line of the final chart, called the critical phase, includes all the task-projects that are suitable for robot use. The individual duration of some task-projects has been reduced due to a crash in the activity duration for that project. In order to fit other projects in the critical phase, a project start can be shifted forward or backward.

In the example in this section, there are two critical segments with one restricted project (Project E or 5). The first segment contains projects A, B, C, and D. Two projects are MTPR (A and B) and the other two are not part of the critical phase (C and D), and are not taken into consideration for the analysis of the critical phase only for the analysis of the second phase, once the first phase analysis is completed.

The analysis of critical segment number one starts with project B. This project has a crash time available of 1.0 months (see Figure 4.1: crash time of project B, CT=1.0) and only one task-project in the second phase with a duration of 0.5 months. The program crashes the project 0.5 months, reducing the task-project duration from 0.5 to 0.41 months and leaving a vacant scheduling slot between B5 and E1. B6 is moved into the first phase to fill the vacant scheduling slot, and all of the task-projects of project B are part of the critical phase.

Project A is the next project to be considered. It has two task-projects in the second phase with a duration of 1.0 months and a total crash time available of 0.5 months. The program crashes project A by 0.5 months and reduces the task-project duration from 0.5 to 0.4 months. The vacant scheduling slot generated in the first phase by crashing the project is not large enough to fit at least one of the task-projects in the second phase (A4 and A5). The program proceeds to analyze the project to the right of project A, in this case project B. The start time of project B can be moved one month and it still has 0.5 months of crash time left. Project A needs only 0.5 month to be shifted into the critical phase, therefore project B is crashed an extra 0.5 months (task-project B duration 0.33 months) and its start

point is moved 0.5 months from 1.5 to 2.0 month. The free space left by project B allows project A to fit completely in the critical phase.

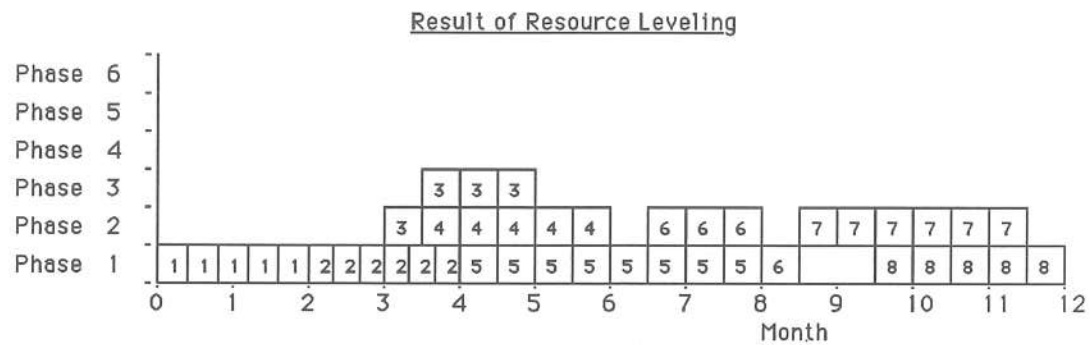
The projects that are not in the first phase are analyzed in the second phase. The second phase analysis fills the vacant scheduling slots left by switching task-projects into the critical phase. In this case C3 and D2 are lowered one phase to fill the slot left by B6.

The second critical segment is defined by projects F, G, and H of which projects F and H are Non-MTPR and project G is completely in the critical segment. The second critical segment analysis starts with project H. Project H has the second least preference order after project G, which is not counted because all its task-projects are part of the critical segment. Project H is fixed and is the last one, therefore it can not be shifted backwards.

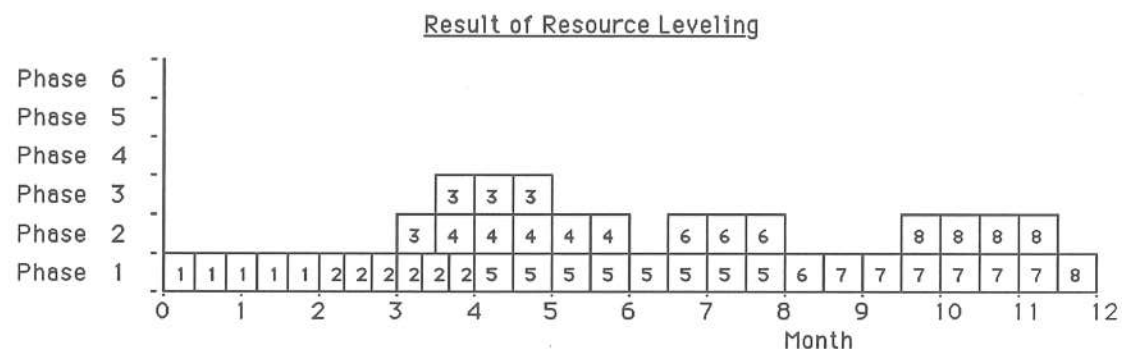
When a project faces all these difficulties the program presents an additional screen with different solutions that can be selected. The options are switch, yield, and abandon in order to convert Non-MTPR projects into MTPR. The Yield option will allow the Non-MTPR project to share task-projects between the phases, the abandon option will leave task-projects of the Non-MTPR project out of the critical phase, and the switch option will switch two projects between phases. The options are shown in Figure 4.5.

For this example the switch option was selected, see Figure 4.5 (a), a switch of phases between projects G and H took place, and project H became entirely included in the critical phase. Two vacant scheduling slots were left by G1 and G2 when they were switched to the second phase. No reduction of activity duration was done up to this stage. The program recognizes these free slots and changes to the next project in the preference ranking, project F, with 0.0 months of crash time available and 1.5 months of latest start time. Project F is switched backwards 1.0 months and switched into the critical phase, filling the free slots left by project G. Project F is now considered MTPR with F2, F3, and F4 part of the critical phase. At this point the program starts analyzing project F as MTPR. The goal now is to make project F fit completely in the critical phase. The project to the right of project F is project H and is a fixed project, therefore it cannot be shifted backward. Under these conditions the program tries to analyze the project to the left of the current project and finds that project E has 1.0 months of crash time available. Task-project F1 needs to be switched to the first segment. With this in mind, project E is crashed, keeping its starting point (fixed project) and reducing its task-project duration from 0.5 to 0.475 months. The solution for this option is shown in Figure 4.6 (a).

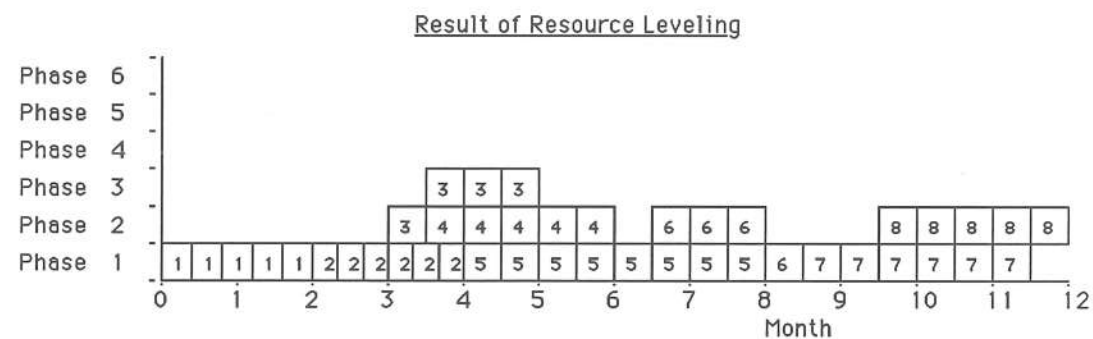




(a)



(b)

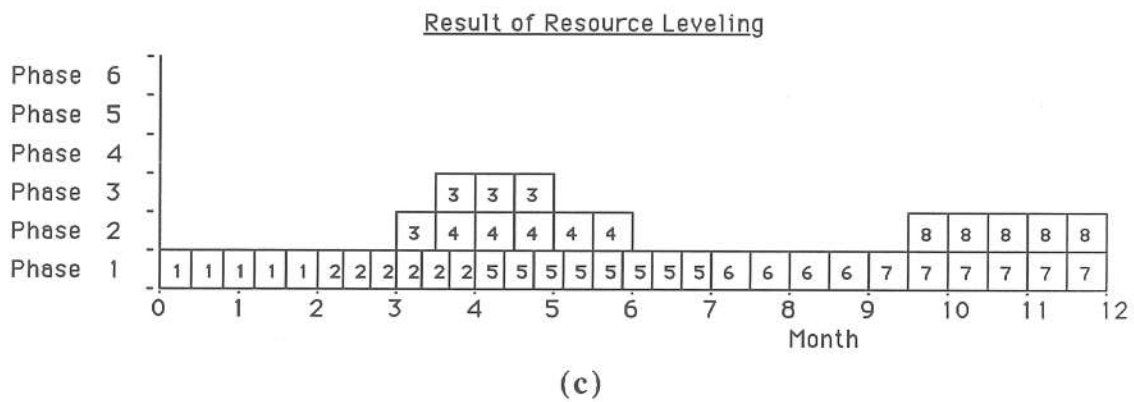
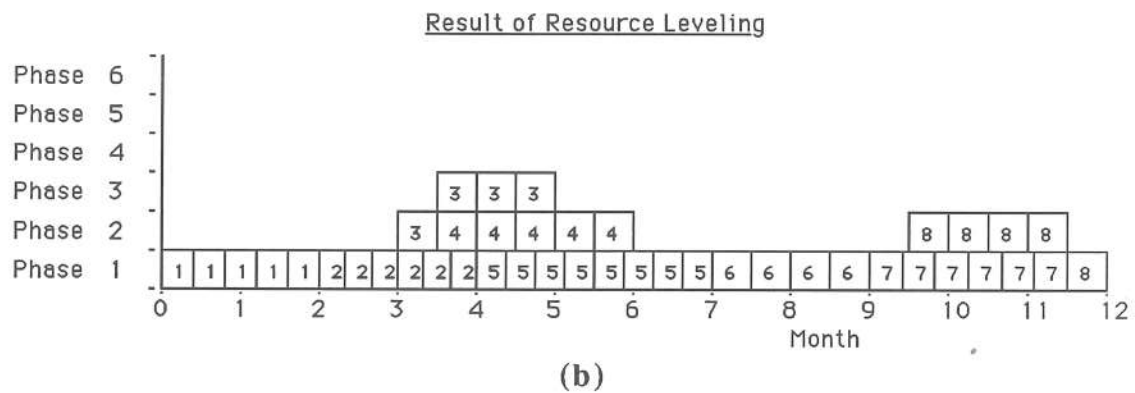
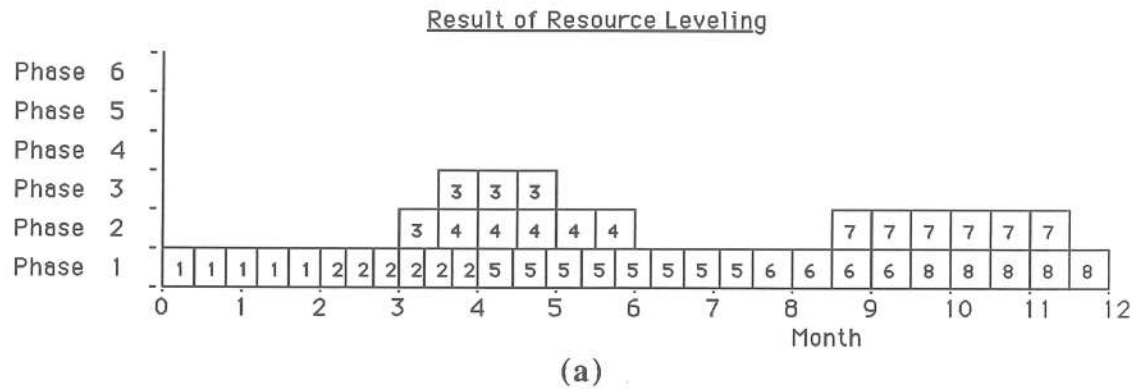


(c)

Figure 4.5. Non-MTPR Solution Options: (a) Switch Option, (b) Yield Option, and (c) Abandon Option

For the different solution options the program only returns one solution, current work is being done to have the program recognize all the possible solutions and analyze them to find the economically feasible solution.

At the end of the program all the projects involved in the analysis of the critical phase are completely part of it. If either the *abandon* option or the *yield* option had been chosen for project H, this would not have been the case. The solutions provided by the program for the three options are shown in Figure 4.6.



**Figure 4.6. Result of Resource Leveling for the Three Solution Options:**  
 (a) Solution for the Switch Option,  
 (b) Solution for the Yield Option, and  
 (c) Solution for the Abandon Option.

## SECTION 5: CONCLUSIONS AND RECOMMENDATIONS

### Conclusions

The goals established for this prototype of a resource scheduling application were accomplished. The first phase of **RILM** proved to be effective and user friendly. Besides the assumptions considered and limitations found, the concept of implementation logistics was proven to be a feasible and promiser area for further research and development.

As a part of the research on the Construction Robotic Equipment Management System (**CREMS**) , the prototype of the logistic scheduling software for the Robot Implementation Logistic Module (**RILM**) was developed within the **HyperCard** and the **QuickBASIC** programming environments. The flowcharts were presented in order to aid a user with a systematic solution to the complicated resource leveling problem by using an interactive computer procedure. The current direction of the research is to enhance the **RILM** prototype for solving the conflict of logistic schedule multi-type robots.

A construction firm can visualize **CREMS** and **RILM** as an advantage for cost analysis and logistic scheduling. The companies are searching for tools that could simplify the decision making process by helping them in the approach and analysis of the problem. As mentioned in Section 1, the actual similar softwares only analyze the cost and scheduling for one machine. The advantage of **CREMS** is that it can analyze many task configurations for the non-robot option and compare the most favorable of them with the cost of the designated robot for the same task.

As mentioned in Section 1, the current version of **RILM** is designed as a standalone prototype and is not robot dependent. This prototype allowed us to prove the functional concepts of **RILM** . The database files are created with the information stored in the first three cards of the **RILM** stack. Part of this information has to be entered by the user and the rest must be collected automatically by the program from the results of each run of the other three **CREMS** modules. The first three modules develop data for one project, therefore it is necessary to run the first modules as many times as there are projects in the firm's portfolio.

## **Recommendations**

Current development work focuses on connecting and interfacing *CTAM*, *RCAM*, and *REEM* with *RILM*. The final output from *REEM*, which is part of the information required by *RILM*, could be transferred by external data files.

The future direction of the research is to enhance the *RILM* prototype for solving the conflict of logistic schedule multi-type robots. *RILM* should be able to handle diverse robots for different construction applications; therefore the robot databases will have to be modified to handle more general and specific information in one file.

Further analysis should also be guided into the implementation of more than two robots simultaneously, and, for example, modifications would be necessary to convert the second phase of the downhill chart into a second critical phase based on the number of robots handled by a given construction contractor.

The future implementation of a heuristic-based approach, such as expert systems, is intended for allowing *RILM* to take decisions and provide suggestions to the contractor.

A change in the programming environment is also feasible. High level programming languages, such as "C" or Pascal, can be use to provide faster execution times and an optimal programming structure. HyperCard and QuickBASIC proved to be well structured programming environments for the prototype stage and new languages can replace them if the new selection proves to be more suitable for *RILM* and *CREMS*.

The author believe that this research will serve as a platform for further investigation efforts in the area, that has a lot of potential for research and future industry application.

## **Acknowledgments**

Support for this research provided by the Obayashi Corporation and the National Science Foundation under the Presidential Young Investigator Award No. MSM-8657142 is sincerely appreciated.

## REFERENCES

- Apple Computer, Inc. (1989). *HyperCard, User's Guide*, Cupertino, CA.
- Apple Computer, Inc. (1989). *HyperTalk Beginner's Guide: An Introduction to Scripting*, Cupertino, CA.
- Apple Computer, Inc. (1988). *HyperCard Script Language Guide: The HyperTalk Language*, Addison-Wesley, Reading, MA.
- Caterpillar Inc. (1990). "Fleetmatch: User's Manual".
- Cornejo, C., Tamaki, K., Posey, J., and Skibniewski, M. (1991). "Computer Assisted System for Construction Robot Implementation Logistics," to appear in the Proceedings of the *Seventh Conference on Computing in Civil Engineering*, ASCE, Washington D.C., May 6-8.
- Goodman, D. (1988). *The Complete HyperCard Handbook*, Bantam Books, New York, NY.
- Goodman, D. (1988). *HyperCard Developer's Guide*, Bantam Books, New York, NY.
- Primavera Systems, Inc. (1987). *Primavera Project Planner: User's Guide*, Bala Cynwyd, PA.
- Russell, J., Skibniewski, M., Vanegas, J. (1990). "Framework for Construction Robot Fleet Management System," *Journal of Construction Engineering and Management*, ASCE, Vol. 116, No.3, pp.448-462.
- Skibniewski, M., Vanegas, J., Russell, J. (1989a). "Construction Robotic Equipment Management System (CREMS)," Proceedings, *Sixth International Symposium on Automation and Robotics in Construction*, sponsored by the Construction Industry Institute, San Francisco, California, June, pp. 404-411.
- Skibniewski, M., Russell, J., Vanegas, J., Posey, J., Rembold, B. (1989b). *CREMS: Construction Robotic Equipment Management System, Progress Report I to the Obayashi Corporation*, Technical Report, Purdue University, School of Civil Engineering, Division of Construction Engineering and Management, June.
- Skibniewski, M., Tamaki, K., Russell, J. (1990a). "Construction Robot Implementation Logistics," Proceedings, *Seventh International Symposium on Automation and Robotics in Construction*, Bristol, England, June, pp.543-555.
- Skibniewski, M., Russell, J., Cartwright, T. (1990b). *CREMS: Construction Robotic Equipment Management System, Progress Report II to the Obayashi Corporation*, Technical Report, Purdue University, School of Civil Engineering, Division of Construction Engineering and Management, March.
- Skibniewski, M., Cornejo, C., Tamaki, K., Posey, J. (1990c). *CREMS: Construction Robotic Equipment Management System, Progress Report III to the Obayashi Corporation*, Technical Report, Purdue University, School of Civil Engineering, Division of Construction Engineering and Management, September.

## **Appendix A: Glossary**

**Button** - An object on a card for initiating a script or process. The script is executed by moving the arrow cursor over a button and clicking the mouse.

**Card** - One screen in *HyperCard*, subunit of a stack.

**Click** - To press down on the mouse button once. This is used where press may seem logical but click is more clear.

**CREMS** - Construction Robotic Equipment Management System.

**Critical phase** - A sequence of task-projects that use the robot. Is referred to the task-projects involved in the bottom line of the downhill chart.

**Critical segment** - The segment between restricted projects. All the projects within that segment, both MTPR and Non-MTPR, are included in the critical segment.

**Downhill Chart** - A basic scheduling chart for robot implementation logistics. The projects and task-projects are arranged in lines or phases. The task-projects in the first lines, also called critical phase, have preference to use the robot (See Figure A.1). This chart also provides the input to **PRO-16**, which is the main process of *RILM*.

**Effectiveness** - The amount of a project that a robot will be able to execute. Usually a ratio of the amount executable divided by the total amount of the project. (Amount is usually area)

**Field** - An object on a card that contains text and/or numbers. Used to pass information between the user and *CREMS*.

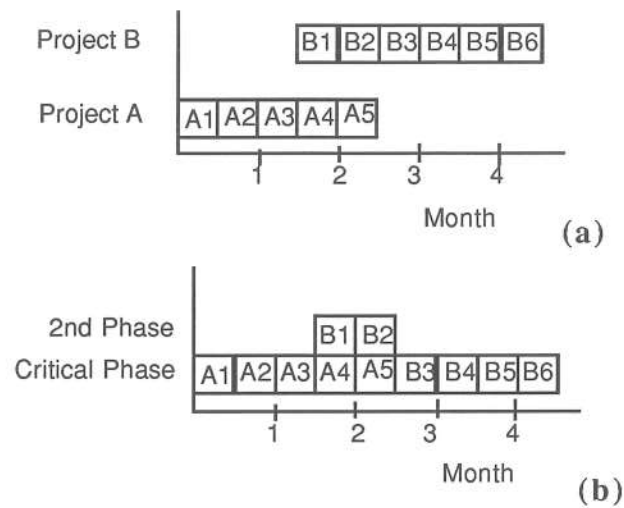
**Fixed project** - A project in which the earliest start time is equal to the latest start time. The total float time for a fixed project is zero.

**Gantt Chart** - A graphic bar chart that defines the precedence relationships of the projects listed for analysis by *RILM*. This chart includes the earliest and latest start times as well as finish times for each project (See Figure A.1).

**Module** - A logical subsection of *CREMS*.

**MTPR** - Minimum Task Project Requirement. The condition of a project when at least three task-projects are in the critical phase.





**Figure A.1. Scheduling Charts (a) Gantt Chart, (b) Downhill Chart**

**Non-MTPR** - The condition of a project when less than three task-projects are in the critical phase. If such a projects exist, its predecessor or successor project is shifted or rearranged following the tactical rules shown in Table A.1 in order to convert the Non-MTPR project into a MTPR project.

TR1	Shift task-project forward
TR2	Shift task-project backward
TR3	Reduce project duration by use of crash time / cost
TR4	Switch task-projects between the critical phase and the second phase
TR5	Yield partial task-projects in a current project to another project.
TR6	Abandon task-project implementation for the use of the robot

**Table A.1. Tactical Rules**

**Object** - A part of card. For example a button or a field

**Preference** - The ranking or preference order of the project for the analysis. Preference is defined by the user.

**Productivity** - A ratio of the amount of the project that the robot can execute divided by the total amount of time required for the robot to execute a project, including the non-operating time.

**PRO-16** - The process 16 is the most important process within *RILM*. It analyzes the downhill chart and executes the resource leveling for robot implementation logistics.

**Resource** - A component necessary to complete a task. Management, labor, equipment, and permanent and temporary materials.

**Restricted project** - A fixed project in the critical phase. A restricted-project has the highest preference and a fixed start point with no float time.

**RILM**.- Robot Implementation Logistics Module.

**Robot Supplemental...** - The parts of a task that are necessary to support, setup, dismantle, and complete the robots work.

**Script** - A programming routine in *HyperCard*.

**Stack** - A collection of cards. In *CREMS*: a submodule.

**Submodule** - A logical subsection of a module.

**Task-project** - A unit of a project used for scheduling activities.

## APPENDIX B: RILM CARDS

This section shows the cards of the current version of *RILM*. All the data fields are presented, the way the program would display them. The figure titles are the card names as contained in the current version of *RILM*.

Project code	List of Projects competing for use of the Robot
12001	Project A
12002	Project B
11003	Project C
21004	Project D
31005	Project E
32006	Project F
33007	Project G
41008	Project H

Figure B.1. Compile List of Projects

Project code	Number of T_P	Project Duration	Early Start	Latest Start	Fixed Project	Preference order	Crash Time
12001	5	2.5	0.0	0.0	No	3	0.5
12002	6	3.0	1.5	2.5	No	2	1.0
11003	4	2.0	3.0	3.0	F	4	0.0
21004	5	2.5	3.5	4.0	No	3	0.5
31005	8	4.0	4.0	4.0	F	1	1.0
32006	4	2.0	6.5	8.0	No	4	0.0
33007	6	3.0	8.5	9.0	No	2	1.0
41008	5	2.5	9.5	9.5	F	3	0.5

Dates in Months

Figure B.2 (a). Logistic Paramenters

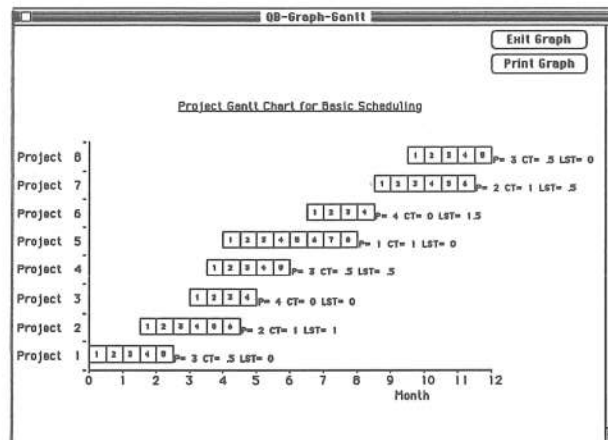


Figure B.2 (b). Gantt Chart for Basic Scheduling (QuickBASIC screen)

C.E.M. Faculty HD:RILM-QB:RILM

LOGISTIC PARAMETERS

RILM : PRO-14.1

Continue

Project code	Project Duration	Robot Cost	Non-Robot Cost	Crash Time	Crash Direct Cost	Incremental Crash Cost
12001	2.5			0.5		100000
12002	3.0			1.0		150000
11003	2.0			0.0		170000
21004	2.5			0.5		90000
31005	4.0			1.0		80000
32006	2.0			0.0		110000
33007	3.0			1.0		115000
41008	2.5			0.5		101000

Dates in Months

Figure B.3. Logistic Parameters (Cont.)

C.E.M. Faculty HD:RILM-QB:RILM

PRO 14.2

Continue

Figure B.4. Continue Card

C.E.M. Faculty HD:RILM-QB:RILM						
PRELIMINARY SCHEDULING						
RILM : PRO-15						
Project code	Task-Project	Order	Order of Preference	T_P start	T_P finish	Phase
12001	1	1	3	0	0.5	1
12001	2	2	3	.5	1	1
12001	3	3	3	1	1.5	1
12001	4	4	3	1.5	2	2
12001	5	5	3	2	2.5	2
12002	1	4	2	1.5	2	1
12002	2	5	2	2	2.5	1
12002	3	6	2	2.5	3	1
12002	4	7	2	3	3.5	1
12002	5	8	2	3.5	4	1
12002	6	9	2	4	4.5	2

Figure B.5 (a). Preliminary Scheduling

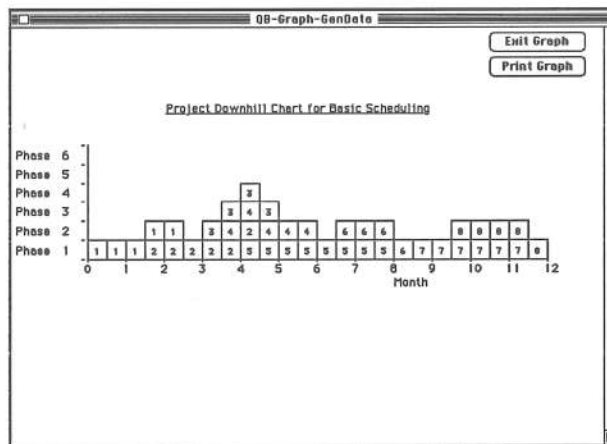


Figure B.5 (b). Project Downhill Chart for Basic Scheduling

C.E.M. Faculty HD:RILM-QB:RILM		
IDENTIFYING CRITICAL SEGMENTS		
RILM : PRO 16.1		
Codes of Critical Projects	Codes of Proj. of Critical Segments	Number of Critical Segments: 2
3 5 8	5	Select Critical Segment ? : 1 OK?

Figure B.6. Identifying Critical Segments (Critical Segment 1)

C.E.M. Faculty HD:RILM-QB:RILM

### PROJECT LEVELING

RILM : PRO 16.3.2

Critical Segment selected: 1

Projects in Critical Seg:1	Preference/ user input	MTPR status	Order by preference	No of T_P in C.Phase
1	3	MTPR	2	3
2	2	MTPR	1	5
3	4	No in C.Phase	0	0
4	3	No in C.Phase	0	0

Is the chart OK?

Figure B.7. Project Leveling (Project Information for Critical Segment 1)

C.E.M. Faculty HD:RILM-QB:RILM

### RESULT OF PROJECT LEVELING

RILM : PRO 16.3.2

Project code	Task-Project	Order	Order of Preference	T_P start	T_P duration	Phase
12001	1	1	3	0.00	0.40	1
12001	2	2	3	0.40	0.40	1
12001	3	3	3	0.80	0.40	1
12001	4	4	3	1.20	0.40	1
12001	5	5	3	1.60	0.40	1
12002	1	4	2	2.00	0.33	1
12002	2	5	2	2.33	0.33	1
12002	3	6	2	2.67	0.33	1
12002	4	7	2	3.00	0.33	1
12002	5	8	2	3.33	0.33	1
12002	6	9	2	3.67	0.33	1

Dates in Months

Figure B.8 (a). Result of Project Leveling (Critical Segment 1)

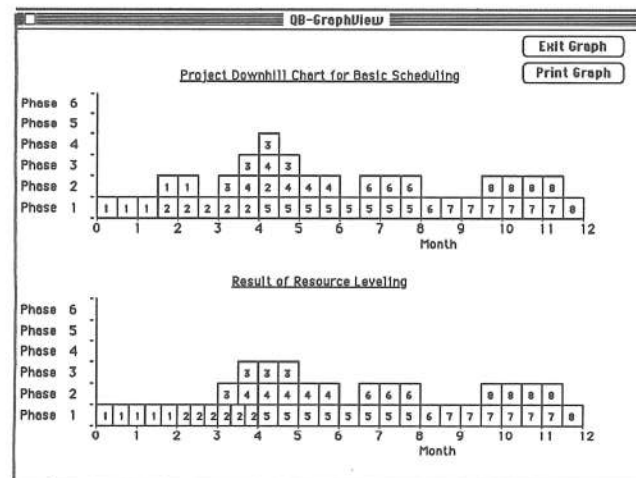


Figure B.8 (b). Downhill Chart for Scheduling of the Critical Segment 1

C.E.M. Faculty HD:RILM-Q8:RILM

**PROJECT LEVELING (Cont.)**

RILM : PRO 16

Critical Segment Selected: 1 out of 2

Select next segment

Down-Hill Charts

Figure B.9. Project Leveling (Cont.)

C.E.M. Faculty HD:RILM-Q8:RILM

**IDENTIFYING CRITICAL SEGMENTS**

RILM : PRO 16.1

Codes of Critical Projects	Codes of Proj. of Critical Segments
3	5
5	
6	

Number of Critical Segments: 2

Select Critical Segment ? : 2 OK?

Figure B.10. Identifying Critical Segments (Critical Segment 2)

C.E.M. Faculty HD:RILM-Q8:RILM

**PROJECT LEVELING**

RILM : PRO 16.3.2

Critical Segment selected: 2

Projects in Critical Seg:2	Preference/ user input	MTPR status	Order by preference in C.Phase	No of T_P
6	4	N-MTPR	3	1
7	2	MTPR	1	6
8	3	N-MTPR	2	1

Is the chart OK? OK?

Figure B.11. Project Leveling (Project Information for Critical Segment 2)



QB-PRO16

**Rules Structure in Resource Leveling**

Non MTPR Project : 41000 ( 0 )  
 Project at the right : 0  
 Project at the left : 7  
 Segment : 2

☐ Switch task\_projects  
☐ Yield partial task\_projects  
☐ abandon task\_project Robot impler

Apply changes & graph view  
 Proceed with changes  
 Graphic View w/o changes

Figure B.12 (a). Solution Options. QuickBASIC Screen.

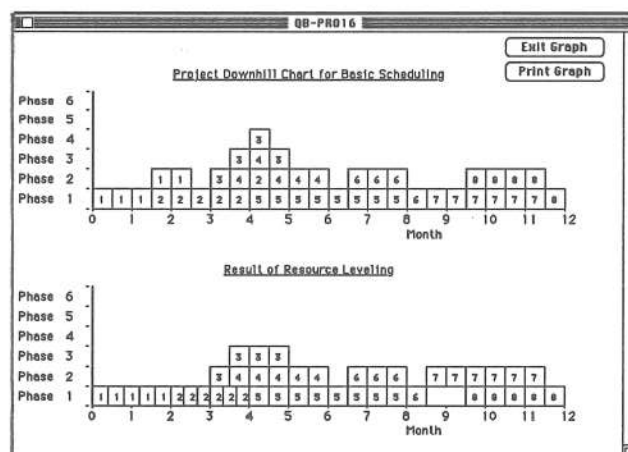


Figure B.12 (b). QuickBASIC screen for the Switch Option after clicking on the "Apply changes & graph view" button.

C.E.M. Faculty HD:RILM-QB:RILM

**RESULT OF PROJECT LEVELING** Continue

RILM : PRO 16.3.2

Project code	Task-Project	Order	Order of Preference	T_P start	T_P duration	Phase
12001	1	1	3	0.00	0.40	1
12001	2	2	3	0.40	0.40	1
12001	3	3	3	0.80	0.40	1
12001	4	4	3	1.20	0.40	1
12001	5	5	3	1.60	0.40	1
12002	1	4	2	2.00	0.33	1
12002	2	5	2	2.33	0.33	1
12002	3	6	2	2.67	0.33	1
12002	4	7	2	3.00	0.33	1
12002	5	8	2	3.33	0.33	1
12002	6	9	2	3.67	0.33	1

Dates in Months

Downhill Chart

Figure B.13. Result of Project Leveling (Critical Segment 2)

The image shows a screenshot of a software window. At the top, a title bar reads "C.E.M. Faculty HD:RILM-Q8:RILM". Below this, the window title is "PROJECT LEVELING (Cont.)". Inside the window, there is a box labeled "RILM : PRO 16". Below that, the text "Critical Segment Selected: 2 out of 2" is displayed. In the bottom right area of the window, there is a button labeled "Downhill Charts". A small arrow icon is visible in the top right corner of the window frame.

**Figure B.14. Project Leveling (Cont.) Final Card**