WRAPS: a programming and simulation software tool for integrated robotic welding

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:


Metadata Record: https://dspace.lboro.ac.uk/2134/28002

Publisher: © K.H. Goh

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
W.R.A.P.S.
A Programming and Simulation Software Tool for Integrated Robotic Welding

by
Kim Huat Goh
MPhil BSc

A Doctoral Thesis
submitted in partial fulfilment of the requirements
for the award of
Degree of Doctor of Philosophy
of the Loughborough University of Technology
May 1988

(c) by K H Goh 1988
The outcome of this research is a low-cost off-line programming and simulation system called WRAPS for robotic arc-welding. WRAPS is an acronym for Welding Robot Adaptive Programming and Simulation. The package was developed on a Future FX20 microcomputer with 640 kbytes RAM under the Concurrent CP/M-86 operating system. It is written in the 'C' language and graphics capabilities is provided via the Graphics System extension (GSX) of Digital Research. WRAPS is menu-driven and utilises windows and icons through custom software for user interface.

WRAPS is composed of 4 basic modules - Modelling, Programming & Simulation, Online and Expert, which enables modelling of the workplace and components, off-line program generation which includes path and process features, communication with welding database, communication with robot controllers and other peripherals such as cassette data tapes, and simulation. In addition it is provided with interfaces and "front-ends" for interaction with CAD and future expert data management systems based on an expert system shell called Knowledge Engineering System (KES). The WRAPS package forms the basis for further development as a supervisory controller for a flexible self-adaptive robotic welding cell in a computer integrated environment. The basis of such a cell is described herein.

WRAPS was developed around Cincinnati Milacron Ltd robot systems but is extendible to a wide range of robots. Several software tools have been developed, eg a graphics toolkit, utilities for generating custom windows and icons, and for inter-computer or peripherals communication. The major modules of the prototype system are complete and demonstrable and are being further developed to commercial status and extended to include expert systems for data management and for real-time distributed process and quality control. The prototype package is being transported
over to IBM AT computers.

The research has shown that a low-cost off-line programming and simulation system could offer a practical and economically viable solution for the creation of robot programs with optimised welding procedures, and could be developed further and used as a low-cost tool for welding integration purposes.
The author has great pleasure to acknowledge the assistance and encouragement from various colleagues and friends. In particular, to J E Middle, who as a supervisor, has provided his thoughtful criticisms and suggestions which have greatly added to the value of the thesis. His immense expertise in welding and experience in the batch fabrication industry have enabled the author to comprehend quickly the multitude of problems associated with robotic welding applications. The author is also grateful to his wife, Madeline, and their children who have unreservedly offered their warm friendship over the past four years.

Special thanks go to R Parker and D Hardwick who have been friends as well as helpful colleagues. Their continuous suggestions and skill involved in the manufacture of many test equipment are greatly appreciated. Heartfelt Gratitude is also due to T W Smith for helping to prepare some of the photographic work in this thesis. The author have also found continuous support and encouragement from all the other technicians in the department, especially R Temple and J Singh.

The author is grateful to members of staff of the Audio Visual Services, especially K Pugh, A Hamilton and J Topham, for the splendid photographs produced from the computer screen for Chapter Five, and also for some conference papers throughout the research period.

A lot of the detailed information about robots was acquired as a result of the steadfast support which the author obtained from C P Jones and other members of Cincinnati Milacron Ltd, the collaborator in the first stage of the project. Their contribution is greatly appreciated.

The integrating aspects of the WRAPS project would have taken more time to be demonstrated if not for the involvement of F J Yong. Despite the very short time available for a Master of Science project, he had been able to design the electronics and build the simulator console used for the demonstration. His contribution towards testing out the concept
has been extremely useful to the WRAPS project.

The author is also greatly indebted to J Folkes and J Eckersley, who despite their hectic involvement with the Loughborough Balloon Club, have found time to proof-read the thesis and offer their valuable comments. Thanks are also due to F Marquis for her care and encouragement, and help in the final preparation of the thesis.

The list of acknowledgements would not be complete without special thanks mentioned for the SERC who have funded the WRAPS research project, and to the Welding Institute for their continuing collaborative effort. The author is particularly thankful to J Weston of the Welding Institute and the SERC Directorate in giving unreserved support, together with J E Middle, in gaining the work permit to enable the author to work on the project.
this thesis is dedicated to
my family and to eleni
who have been my main
source of inspirations
and loving support.
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D, 3D</td>
<td>2, 3 Dimensional</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BASIC</td>
<td>Beginner's All-purpose Symbolic Instruction Code</td>
</tr>
<tr>
<td>BCD</td>
<td>Binary Coded Decimal</td>
</tr>
<tr>
<td>bit</td>
<td>binary digit</td>
</tr>
<tr>
<td>byte</td>
<td>A sub-division of a computer word, usually comprising of 8 bits</td>
</tr>
<tr>
<td>C-DOS, C-DOS'XM</td>
<td>Concurrent DOS and Concurrent DOS Expanded Memory, microcomputer operating systems by Digital Research Inc, USA.</td>
</tr>
<tr>
<td>CAD/CAM</td>
<td>Computer-Aided Design/Computer-Aided Manufacture</td>
</tr>
<tr>
<td>CIM</td>
<td>Computer-Integrated Manufacturing</td>
</tr>
<tr>
<td>CCP/M</td>
<td>Concurrent version of CP/M</td>
</tr>
<tr>
<td>CML</td>
<td>Cincinnati Milacron Limited, USA, robot manufacturer.</td>
</tr>
<tr>
<td>CP/M</td>
<td>Control Program for Microcomputers</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analogue Convertor.</td>
</tr>
<tr>
<td>DDCMP</td>
<td>Digital Data Communications Message Protocol, of Digital Equipment Corp USA.</td>
</tr>
<tr>
<td>GKS</td>
<td>Graphical Kernel System, an emerging graphics standard.</td>
</tr>
<tr>
<td>GSX</td>
<td>Graphical System Extension, of Digital Research Inc, USA.</td>
</tr>
<tr>
<td>IBM Ltd</td>
<td>International Business Machines Limited.</td>
</tr>
<tr>
<td>IEEE</td>
<td>US Electronic standards body for the 488 and 728 documents covering the General Purpose Interface Bus</td>
</tr>
<tr>
<td>IGES</td>
<td>Initial Graphics Exchange Specifications</td>
</tr>
<tr>
<td>KES</td>
<td>Knowledge Engineering System, an expert system shell of Software &amp; Engineering, USA.</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>MAP</td>
<td>Manufacturing Automation Protocol, an emerging communication standard</td>
</tr>
<tr>
<td>GMAW</td>
<td>Gas Metal Arc Welding</td>
</tr>
<tr>
<td>mm</td>
<td>millimetre, 1/1000th of a metre - International System Organisation measurement standard.</td>
</tr>
<tr>
<td>MS-DOS</td>
<td>Microsoft Disk Operating System</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PC-DOS</td>
<td>Personal Computer Disk Operating System</td>
</tr>
</tbody>
</table>
Abbreviations

PDES   Product Data Exchange Specifications
pixel  picture element
PLC    Programmable Logic Controller
RAM    Random Access Memory
RS-232, An EIA recommended standard (RS) for connecting
data processing devices.
RS-232C
s      second, unit of time.
SERC   Science and Engineering Research Council
SAA    Systems Application Architecture, IBM's architectural framework for developing programs that operate together in distributed environments.
SNA    Systems Network Architecture, IBM's total description of logical structure, format protocols and operational sequence for transmitting information units.
TCP    Tool Centre Point
TIG    Tungsten Inert Gas (Welding)
TOP    Technical and Office Protocol, an emerging communication standard
UK     United Kingdom
USA    United States of America
VDU    Visual Display Unit
WRAPS  Welding Robot Adaptive Programming and Simulation
CONTENTS

TITLE
DECLARATION
ABSTRACT
ACKNOWLEDGEMENTS
DEDICATION
ABBREVIATIONS
CONTENTS

PREFACE

P.1 Organisation of Thesis xv
P.2 Conventions Used
  P.2.1 Text Format xvi
  P.2.2 Numbering of Pages xvii
  P.2.3 Numbering of Figures xviii
  P.2.4 System of Units xvi
  P.2.5 Abbreviations xviii
  P.2.6 References xviii

CHAPTER ONE
Introduction to WRAPS

1.1 Introduction to Chapter 1
1.2 Background of WRAPS 1
1.3 The Significance of WRAPS 2
1.4 Objectives of the WRAPS Research 3
1.5 Special Notes about the WRAPS Research 3

CHAPTER TWO
Literature Survey

2.1 Introduction to Chapter 4
2.2 Background Survey 4
  2.2.1 Automation of the Welding Process 5
2.2.2 Motivation for Off-line Programming 6
2.2.3 Suitability Analysis of Available Off-line Programming Packages 21
2.2.4 Integration of Welding Databases and Knowledge 25
2.2.5 Expert Systems in Manufacturing and Process Control 30

2.3 Outcome of Survey 36

CHAPTER THREE
Design Considerations of WRAPS

3.1 Introduction to Chapter 37
3.2 Specifications of WRAPS 37
3.3 WRAPS System Development Considerations 38
3.3.1 Achieving the Task Functionality 39
3.3.2 Achieving Basic Low-cost of WRAPS 41
3.3.3 General Issues in Designing WRAPS 43
3.3.4 The Hardware and Software for WRAPS 48
3.3.4.1 The Microcomputer System 48
3.3.4.2 The Software Implementation 49
3.3.4.2.1 Achieving Software Portability 49
3.3.5 Fitting WRAPS into a Microcomputer 56
3.3.5.1 Minimising Software Code 56
3.3.5.2 Minimising Size of WRAPS Package 58
3.3.5.2.1 Minimising Program Size 58
3.3.5.2.2 Minimising Data Memory 61
3.3.6 Providing Help to WRAPS Users 61
3.3.7 WRAPS Limitations 64
3.4 WRAPS Layout 71
3.4.1 Software 71
3.4.2 Screen 72
3.5 Graphical Representation of Objects 73
3.5.1 Rendering Edges to the Object 76

CHAPTER FOUR
Implementation of WRAPS

4.1 Introduction to Chapter 79
4.2 The Modelling Module 79
5.7 Economic Evaluation 188

CHAPTER SIX
Process Integrating Capability of WRAPS

6.1 Introduction to Chapter 191
6.2 The Integrating Environment 191
   6.2.1 Integrating WRAPS and Hardware Through A Network 192
   6.2.2 Integrating WRAPS with other Software Packages 200
6.3 The Process Planning Interface 203
6.4 Weld Process Integrating Capability of WRAPS 205
   6.4.1 Pre-weld Inspection 206
   6.4.2 Welding and Inspection 206
   6.4.3 Post-weld Inspection 207
   6.4.4 Data Acquisition 208
   6.4.5 Simulated WRAPS Flexible Robotic Welding Cell 211
      6.4.5.1 Equipment used and Setup 213
      6.4.5.2 Software Building for the System 218
      6.4.5.3 Demonstration of the Simulated Cell 221

CHAPTER SEVEN
Future Work

7.1 Introduction to Chapter 229
7.2 Further Research and Development Work 229

CHAPTER EIGHT
Conclusion

8.1 Introduction to Chapter 247
8.2 Conclusion 247

REFERENCES R-1

APPENDICES
A1 Robot Manufacturers Mentioned in Thesis A1-1
A2 An Example of a Robot Search Operation for a Horizontal Fillet Weld A2-1
Chapter One  Introduction to WRAPS
Chapter Two  Literature Survey
Chapter Three  Design Considerations of WRAPS
Chapter Four  Implementation of WRAPS
Chapter Five  Test Evaluation of WRAPS
Chapter Six  Process Integrating Capability of WRAPS
Chapter Seven  Future Work
Chapter Eight  Conclusion

The treatise on the research topic starts in Chapter One where a
general introduction is made. The assessment of the background leading
to the research and development of WRAPS is made in Chapter Two. It
includes the literature survey to determine the state-of-the-art
technology relating to the topic in order to assess what was happening
and what needs to be done. The research leading to the development of
the overall WRAPS project, primarily of a software nature, requires the
consideration of the functionality and the user interface aspects and
these are discussed in Chapter Three. Chapter Four describes the
implementation of the WRAPS software on the FX20 microcomputer system,
followed by a practical evaluation in Chapter Five. The advanced stage
of the WRAPS project is conceptual but inherently linked to the initial
work, and this is discussed in Chapter Six. This is carried out in
conjunction with a simulated integrated WRAPS cell consisting of an IBM
ATx microcomputer, input/output control board, a custom-designed
simulator console, and software linked with an expert system shell
called KES. A network system to link all the entities under WRAPS is also proposed. Chapter Seven contains discussions on future work that needs to be carried out so that the initial limitations of WRAPS will be overcome and used as a viable laboratory and industrial tool.

P.2 Conventions Used

The conventions used in this thesis are divided into 6 areas; text format, numbering of pages, numbering of figures, system of units, abbreviations and references.

P.2.1 Text Format

There are 3 main conventions used:

i) WRAPS software functions are all embraced within single quotes, eg 'display_robot_function' which prints the name of a robot function onto the computer screen. But overlay modules are typed in capital letters, for example 'REDRAW' which re-draws every object in the workplace.

ii) Any other references to special words or names may be between apostrophes. For example, "Hello There!" refers to a string of characters.

iii) Non-WRAPS software functions mentioned in the text, for example those belonging to the "C" language, the expert shell KES and graphics drawing functions under the GSX graphics extension, are in italics, eg putchar() which is the "C" library function to output a character to the computer screen.
P.2.2 Numbering of the Pages

All the pages in this thesis are numbered consecutively, starting with Chapter One up to the last chapter, Chapter Eight. The exceptions are as follows:

- pages before Chapter One are numbered with the roman lettering system.
  
  Example: ix refers to page (9) of the pages before Chapter One.

- a page number in the References section is preceded with a letter "R" and a hyphen.
  
  Example: R-5 refers to page 5 of the References section.

- a page number in the Appendices is preceded by a letter "A", an Appendix number, and a hyphen.
  

P.2.3 Numbering of the Figures

The figures, or illustrations, are numbered consecutively, starting anew with each chapter. A figure number is embraced by parenthesis and is preceded by a chapter number and a hyphen.

Example: Fig(2-3) refers to 1 Chapter Two Figure 3.

In certain cases, a figure may contain more than 1 item. In this, the item number is specified after the normal figure number, being separated by a slash.

Examples: Fig(2-3/a) refers to 1 Chapter Two Figure 3 item (a).
           Fig(2-3/4) refers to 1 Chapter Two Figure 3 item 4.
P.2.4 System of Units

The SI standard of units are used where possible throughout this thesis. However, in some cases where it is deemed appropriate to quote old units, eg units as still used in some equipment, the preferred SI equivalent is provided in brackets.

Example: 1 inch (25.4 mm)

P.2.5 Abbreviations

The abbreviations used in this thesis are for practical reasons, or are abbreviations which are commonly used. Some of the uncommon abbreviations are explained at those points in the thesis where they occur. A list of the abbreviations use are on page (viii).

P.2.6 References

All references to scholarly or industrial literature are classified under References rather than Bibliography. References to companies by names which are inevitable in this research, for example to link a product to a company, are also cited in the References section. References are quoted between square parentheses, viz. [reference item number(s)].

A slight deviation from the above is when the names of companies are mentioned in relation to a specific robot(s), for example CML T3 robot or FANUC robot. All robot names are given in capital letters. Because robot names are often referred to, a special list of all the robot-to-manufacturer relationships is given in Appendix(A1). This is in the form of the robot name, followed by the supplier’s or manufacturer’s address.
Chapter One

INTRODUCTION TO WRAPS

1.1 Introduction

This chapter provides a general introduction to the background of the WRAPS Project. It also describes the special features which makes WRAPS unique from other off-line programming and simulation packages that were available at the time when the Project was initiated and until the time of writing this thesis.

1.2 Background of WRAPS

WRAPS is an acronym for Welding Robot Adaptive Programming and Simulation. WRAPS has evolved from a project funded by a SERC grant (GRC/C/83326 - 1984) in association with the UK subsidiary of Cincinnati Milacron Ltd (CML), concerned with off-line programming and seam tracking for welding applications. The aspect of off-line programming and low-cost welding process integration is the major topic of this research. As the detailed survey in Chapter Two will show, early research in the application of robots to welding indicated on-line programming to be a significant cost factor in the economic viability of application to the small batch manufacturing environment.

An off-line programming system, because it runs off a remote computer, offers the obvious versatility of integrating with other domains of application functions and expertise. Consequently, it was considered conceptually and technically feasible to integrate the various aspects of the welding process and its related activities. The process of system integration, in this respect, requires a vehicle - a suitable computer-aided system which has developed into WRAPS. It was necessary to adopt a systematic approach to analyse the requirements of such a system. The analysis, which is carried out in the next chapter,
will show the following requirements:

i) a low-cost off-line programming system for the generation of off-line programs for robots assigned to specific manufacturing processes.

ii) a simple and sufficient package to suit the application.

iii) the use of expert knowledge to guide in the selection of welding procedures.

iv) closing the forward and feedback loop of the complete welding process to achieve a flexible welding cell within a CIM environment.

WRAPS is conceived as a package with suitable facilities that will allow some of the specifications to be achieved.

1.3 The Significance of WRAPS

The research into off-line programming has led WRAPS to be developed as a low-cost easy-to-use off-line programming system whereby a welding task may be programmed using a remote computer terminal away from the actual robot installation. The final program produced from WRAPS is to contain all the necessary and appropriate welding parameters, eg current, voltage and weldgun angle orientation, to control the actual process performed by the welding robot.

WRAPS offers the following built-in concepts:

i) The basis of a flexible integrated robotic welding cell.

ii) Expert knowledge integration to facilitate process control and information management.

iii) Process planning extension.

iv) Off-line simulation with process optimisation.

v) Possible flexible post-processing to enable output to different robot systems.
vi) Import of data from CAD and other packages.

vii) A generic tool to be used for programming in other processes.

1.4 The Objectives of the WRAPS Research

a) To produce a low-cost robotic programming and simulation package that satisfies points (i) to (iv) of the specifications under Sect 1.3 and to demonstrate the feasibility of application of the package to robotic arc-welding.

b) To test the concepts and to provide facilities for system expansion to satisfy points (v) to (vii) of the specifications and the operation of WRAPS within CIM.

1.5 Special Notes about the WRAPS Research

Published Information

The basis of the WRAPS concept was first published at the National Production Research conference [1]. In addition, internal reports and progress reports to SERC, have been written that charts the development of the whole project. The latest paper [2,3] presented by the author at an international conference on automated welding includes a discussion of the overall WRAPS integration within a CIM environment.

Software Confidentiality

All functions described herein are mentioned only by their names. The Welding Institute [4] are currently evaluating the system with a view to collaborating in its commercial development and marketing of the WRAPS package when completed. However, all the software source files are well maintained and available for examination purposes.
CHAPTER TWO
2.1 Introduction to Chapter

This chapter contains the background study into the various areas that support the work on WRAPS. It investigates into the area of off-line programming, its requirements, and the benefits and disadvantages associated with this technique of robot programming. It also describes the involvement in graphical 3D object modelling, particularly in the CADCAM areas. The development of off-line programming from a fully textural type to a wire-frame modelling type is described. The reasons leading to the development of WRAPS are investigated.

This chapter includes an examination of the welding process in relation to automation, its requirements to achieve automation and what automated welding can offer. The various aspects of an integrated flexible welding cell are also investigated. Various welding parameters in the form of weld procedures are discussed in order to justify the inclusion of expert knowledge-based system within WRAPS. Also, the requirements of a flexible robotic welding cell in terms of the data acquisition and "intelligent" usage of such data are assessed.

2.2 Background Survey

The primary aim of this research is to assess and develop an economical and practical off-line programming tool for robotic arc-welding applications to include the quality control of the weld, and to aid in the integration of various aspects of robotic welding with other processes, typically in a CIM environment. The arc-welding considered is the Gas Metal Arc Welding (GMAW) process [5].
Although the technology was becoming available commercially at the beginning of this research to achieve the various tasks for a closed-loop weld quality control, the tasks have to be carried out separately. The tasks necessary to achieve integration in a robotic welding cell are data acquisition with expert knowledge based interpretation, and networking. The former requires sensors for data capture, and equipment for signal conditioning. The latter is required for inter-module communication to enable transfer of data and information for control, and data archive.

To prepare the groundwork for WRAPS, it was deemed necessary to investigate (i) the automation of the welding process, (ii) the motivation for off-line programming, (iii) the suitability of available systems, and (iv) the integration of welding database and knowledge.

2.2.1 Automation of the Welding Process

The phrase "automatic welding" refers to a welding process where the weld seam generation is achieved using reprogrammable welding equipment, attached to reprogrammable manipulators which are taught to execute a certain task. Automatic welding may be run attended or un-attended. The former requires an operator to keep control of the process, often intervening to alter the path or weld control parameters required to achieve a desired weld. Often the equipment involved are dedicated welding machines. The latter requires no operator involvement apart from the initial setting up. Once initiated, the welding equipment executes the welding process slavishly until a desired batch count is reached, or an interrupt is made to stop it. Ideally, the process is self-adaptive and adjust its positional and weld control parameters, to match the changing workpiece characteristics. Intermediate loading and unloading of workpieces is also carried out by automatic means.

The advantages of using robots in automatic welding applications have been apparent for the past decade or so. With their inherent controllable, repeatable movements, robots are capable of producing more consistent welds, hence higher quality products and increased arc-on time, than could be attained by manual methods. It is also capable of
high-speed continuous welding and using large currents. These are conditions which would cause fatigue to a human welder due to the heat, arc glare and the weight of a heavy duty weldgun, or weld torch as it is also called. Throughout this thesis, the term weldgun is used. In addition to increased productivity, other advantages include increased economical use of plant space and facilities, improved safety and reduced environmental concerns [6,7].

A robot system equipped with a suitable arc-welding machine is used in many fabrication applications, but often little or no feedback is used or available for the safe-guarding of the quality of the weld obtained by closed-loop control. Sensors that could inspect the welds that are being formed or after, are necessary to gather information which is otherwise un-recorded and wasted. Such information may include, e.g. geometry of the weld, cracks, run appearance etc.

Traditionally, geometrical data from an engineering drawing is transferred into a robot program by manual means, typically through interaction of the robot manipulator with a physical product via teach-pendant programming methods which are discussed further below. There is enormous potential for such geometrical data, and also welding data that governs the weld quality, to be organised into a format that permits automatic access from a programming system.

There is hence a great potential to integrate these components together as part of a welding cell. Fig(2-1/a) shows the concept of such an integrated system.

2.2.2 Motivation for Off-line Programming

Since their introduction, all commercially manufactured servo-controlled industrial robots have been equipped for on-line programming. It is necessary for the robot operator to "teach" the robot its program by guiding it through its assigned tasks, typically using a hand-held pendant having the necessary keys to control and program the robot [81. Usually this is performed in conjunction with a keyboard which is linked to the robot system controller for additional data
FIG (2-1) Block Diagram of an Integrated Robot Welding System
(a) Without and (b) With Off-line Programming and Simulation and (c) With an Expert System.

FIG (2-2) The Robot Calibration Method Used by the SELSPINE Robot Check System.
entry. In contrast to on-line, the off-line programming technique can be defined as a task of creating a robot task remote from the actual robot hardware, typically using a remote computer to generate the necessary robot point coordinates, function data and program cycle logic that is in the natural or translated format required to function on the actual robot system. The integration of off-line programming to a robotic welding cell is shown in Fig(2-1/b).

Teach pendant programming methods hinder the use of robots for welding in several ways. Research undertaken in the Department of Manufacturing Engineering of Loughborough University and elsewhere has shown that conventional teach pendant programming can be a significant limiting factor in terms of productivity and costs [9-12].

The robot system could not be used for production work when a task is being taught. The time taken to teach a task may vary from a few hours to several weeks, depending on the complexity of the components [13,14]. This would in particular adversely affect the costs justification of small batch fabricators where a lot of programming is required in lieu of the small quantity, large variety of components. Programming time can only be insignificant if it could be apportioned over a large number of identical components.

Pendant programming also places the operator in a hazardous environment and visual location of the welding wire tip relative to the joint can introduce significant positioning errors. This could affect the need to improve and assure the accuracy of parts manufacturing, assembly and the jigging required to hold the parts for welding [13,15,16]. In the cases studied through a Teaching Company Program [17], because of the high cost of jigging, the expected total investment for a robot welding implementation can be as much as £140,000.

Advantages and disadvantages of off-line programming methods over teach pendant methods are now well documented [18-21]. Those which are related to robotic arc welding are discussed below:
Advantages

i) Improved efficiency - by increasing robot availability. Does not require a robot to be taken off productive work for creating the robot program. Also, if a robot is being serviced or repaired, the robot program could still be generated off-line.

ii) Improved safety - the human operator does not have to be in the working area, unless a "fine-tuning" of a robot program is required. Collisions may be avoided by pre-checking of program by simulation.

iii) Overcome the difficulties of pendant programming. A lot of the programming steps could be simplified. Also, all the necessary programming instructions could be accessed from the off-line programming computer rather than have to press some buttons on the teach pendant at some instance and then some keys on the robot control console.

iv) Increased flexibility - eg a variety of programming approaches may be checked out. This would help in the creation of an optimised robot program. Also, it would be possible to interact with other software packages, eg, to draw information from a weld database management package. This flexibility itself provides the capability to integrate into a CIM environment.

v) Spread operational load on programmers. A partially completed robot program may be saved and completed in later sessions.

vi) Off-line robot programs, which are high-level robot programs, are relative easier to comprehend.
by humans and to translate from one format to another, than compared with robot programs at the robot control level.

vii) Path generation - saves time and effort for programming circular weld paths or other paths of a defined geometrical nature. Other custom weld paths could also be generated, stored and re-used as a macro sequence, i.e., a series of commands strung together and invoked by a single command.

viii) Demonstrated best "accuracy" of off-line taught points producible is approximately +/- 3mm when replayed through the robot system [17]. Errors in the actual robot geometry arise because of various reasons, eg insufficient component fit-up and robot-to-work-positioner calibration requirements, and distortion of components during welding. Means of overcoming this limitation of the off-line program are now possible, especially, by the use of sensors for seam tracking. This is discussed further below.

ix) A high level of welding knowledge is not required as this can be embedded into the software through the use of artificial intelligence systems, in particular, knowledge based systems which can be created with the aid of expert system shells. This is discussed further below in Sect 2-2-5. Manual welders made redundant by the introduction of automated robot welding systems could be re-trained to perform off-line programming.

Disadvantages

i) No transfer of (welding) skill to operator from the off-line programming system.
ii) Needs thorough knowledge of the process operation, although the introduction of intelligent knowledge based systems can reduce this requirement.

iii) Seam-tracking requires additional robot software and hardware requirements. Sensors available today are expensive and restricted in their range of application. For example, the MetaTorch [22] in 1984 costed around £25000 for a basic system and is limited to certain joint types in sheet metal.

Off-line programming for robotic arc welding can offer additional advantages. The orientation of the weldgun at a weld point and along a welding-seam can be determined automatically from within the programming package. This, together with other weld data in relation to the type of joint, gap size, etc could be retrieved from a database and incorporated directly within the final robot program which is downloaded to the robot controller. Off-line programming system can also provide for rapid cost estimates of welding cycles, and enables selection of the lowest cost alternative from a number of possible robot weld programs.

Off-line programming requires the following:

a) - knowledge of the task.
   - knowledge of robot.
   - a method of programming, whether textual or graphical.
   - interface to the human programmer.

b) - 3D world model of workplace and work.
   - interface to the robot controller.

c) - verification of the robot program.
   - interface to the 'world'.
Chapter Two

(a) These 4 requirements are easily achieved with some software engineering. Knowledge about a process or robot could be incorporated into a program, either as the usual inflexible top-down conventional coded program or a program interfaced to an expert system with flexible knowledge and data manipulation.

The method of programming and human interface are discussed in Chapter Three Sect 3-3-3.

(b) There can be no exact matching of the modelled environment to that of the real situation due to many contributing factors, such as temperature effects, relative positioning errors between robot and component fixtures, accumulative production errors in the robot joints, backlash in drives train, etc. In the first place, the movements of the robot modelled in the off-line programming system has to be verified in relation to the actual robot system. This could be obtained, eg by using a sophisticated laser scanning system, eg based on the Selspine Robot Check System [23]. The typical system involves mounting light-emitting diodes (LEDs) at selected points on the robot being monitored, Fig(2-2). Two cameras, each equipped with a unique opto-electronic detector receive the light from the LEDs to determine its position on the robot in an X-Y plane. The 2D information from the 2 cameras are then transformed into a 3D format which may be displayed numerically or graphically and stored for future use. The system is claimed to be capable of measuring robot variables such as positioning, velocity, acceleration, vibrations, path curve accuracy and over-shooting. Conversely, by mounting the 2 cameras on the robot and the LEDs on the equipment to be calibrated, eg a work positioner, their relative positions could be calibrated. The measured results are then used to update the alignment of the modelled environment. For the model of the robot system itself, discrepancies with the real system may mean modifications to the geometrical setup and the kinematic and/or dynamic algorithms used to simulate its motion.

An alternative is to use other measuring probes attached to the
robot as an end effector. Some examples of probes that could be used are electronic touch trigger probes [24,25], laser triangulation sensors [24,25] and linear differential transducers (LVDT). The principle and operation of a touch probe is shown in Fig(2-3/a). The non-contact method of measuring using a probe that is based on a laser triangulation sensor [24,25] is shown in Fig(2-3/b). The availability of a valid/invalid signal output from such a sensor made it possible to be used as in a ON/OFF situation analogous to touch probe triggering. A 'valid' signal occurs when the component to be measured is positioned within the measurement range of the sensor otherwise an 'invalid' signal occurs. To detect an edge of a component, assume initially the sensor is positioned at a 'valid' location (1) as shown in Fig(2-4). By moving it over to (2), the sensor becomes out of range and the signal register 'invalid'. When this happens, the robot coordinates are recorded. With 1 edge detection, a point in space is defined and hence with 2 edge detections produces a length. Several other techniques are used to measure the height and width of a component, or the size of a hole. Details on this topic could be found in Goh [24].

Fig(2-5) shows an instance when a robot is equipped with a trigger probe to calibrate the position of a work bench relative to its coordinate system. The robot is used to measure strategically located points on the bench, which in this case, are represented by precision cubes. The results are sent to the off-line programming system which then takes these measurements into account and re-locate the modelled work bench in the simulated environment. In this simple example, only 3 points were required to determine a plane to align the top working surface of the bench with the equivalent points defined in the model. More accurate results are attainable, if for instance, a volumetric frame is attached to the object, and measurements are than made of this frame.

A calibrated real welding system may still have local variations in terms of the weld seams and component points due to various reasons, eg errors in jig and fixturing to hold the component for welding, and manufacturing tolerances in preparing the components. Sensing methods are also available to compensate for these discrepancies.
The probe consists of a stylus and mounting holder, which comprises 3 rollers mounted 120 degrees apart. This assembly is located on 3 sets of V-cylinders by a spring. An electric circuit is made through the cylinders, the resistance of the circuit changes due to the reduction in surface contact area on 1 or more of the cylinders as soon as any force from any direction is applied to the ball. This gives a characteristics of resistance versus stylus deflection which enables repeatabilities of less than half a micron to be achieved using this basic principle. Once measurement has been taken, complete lift safeguards the probe against overtravel. The measurement system is not a break contact switch but an omni-directional strain device giving a trigger point which is not subject to contact erosion associated with mechanical contacts. The adjustable spring pressure holds down the probe tip at null position so that as soon as the contact with the part ceases, the probe returns to its null state.

(a) Principle of Renishaw Touch Trigger Probe.

The beam from laser 'L' falls on point 'X' on the workpiece. Diffusely reflected radiation from the light spot 'X' is captured by lens 'O' and projected on a linear detector 'D'. The detector is divided into a row of separate elements with an element number. The spot on which is projected, determines the height 'H'. After the detector is read, the height is known.

(b) Laser-based Triangulation Principle.

FIG(2-3) Two Types of Probes that Could be Used in the Calibration of the Work Cell for Robotic Off-line Programming.
At (1) the sensor detects a 'valid' signal. When moved over to (2), the sensor becomes out of range and the signal is 'invalid'. The electronics detecting this change of signal status, immediately freezes the reading of the robot or measuring system coordinates. Hence a reading is obtained for a point. One point each on two edges of a component will give a length reading.

FIG(2-4) Edge Detection Using a Laser-based Triangulation Sensor.

FIG(2-5) A Method of Calibrating a Work Bench Position Relative to a Robot Coordinate System. The Measurements Taken will be Fed into the Modelled Environment for Alignment with the Equivalent Reference Points on the Model Bench.
The most direct method uses tactile sensing [26] in which a mechanised probe rides in the weld joint just ahead of the weldgun. Variations in the probe's position are translated into proportional error signals and fed back to the robot controller to allow corrective actions to be taken. One example in the use of a tactile seam tracker in a robotic welding environment is the Cyclomatic Robotracker [27] which was used for GMAW of roof panels in car body assembly. However, the intrinsic mechanical difficulties associated with a contact probe prohibit its use. Other probe systems used are ultrasound [28], eddy current measurements [29], electro-magnetic sensing [30] and visual guide-line tracking [31]. References [32,33] describe joint search and seam-tracking using a variety of optical, inductive and through-the-arc sensing systems.

A more advanced method is to use a vision system, e.g. a solid-state or vacuum-tube closed-circuit TV camera or a laser scanning system [34-36]. There are at least 3 methods of altering the robot coordinates to compensate for spatial discrepancies. The most straightforward method uses the vision system to provide a final position offset from a taught point, e.g. at a hole or the edge of a component. By combining this information with previously taught coordinates, a new location is generated to which the robot is moved. The second method uses the vision system to define the final position of the robot end effector in either the world (normally at the centre of the base) or the tool (normally at the centre at the end effector) reference frame. The robot is made to move to this point without reference to any pre-taught point. The third method uses the vision data to perform a frame shift of the robot world reference frame. Data obtained by the vision system provides the final new position of the end effector relative to some defined reference frame. By using the appropriate transformation, the offset information is generated. This information is used to translate and rotate the world coordinate system of the robot which then executes the un-altered program relative to a new set of axes.

The calibration problem for arc welding may well be overcome if the robot system in use has search and seam-tracking facility. Fig(2-6) shows the principle of laser scanning for seam tracking. Other systems use a stripe sensor for similar processing algorithms, e.g. the Liverpool
The laser beam and the radiation from spot 'X' are both deflected by mirrors mounted on an axis which has an oscillating motion. This causes the laser beam to scan cross-wise over the seam while the radiation received from 'X' remains aligned on the detector. Each scan gives a complete profile of the weld seam, typically constructed out of 200 height data points and 200 mirror angle positions per scan. This therefore provides a 2-dimensional profile of the seam.

To obtain a 3-dimensional picture of the seam, the sensor is moved along the seam in tandem with the weldgun. The direction of motion, being perpendicular to the height profiles taken, identifies the third dimension, and therefore enabling the topography of the workpiece to be constructed. The weldgun is then able to anticipate the geometry of the seam and adapt accordingly. However, the weldgun has to be positioned correctly over the seam and the scan is obtained some distance ahead of the weldgun to allow for processing time to arrive at the decision based on the obtained geometrical information.

FIG (2-6) Principle of Seam Tracking; Used for Adapting to Variations in the Welding Seam Using a Laser-based Sensor.
[Oldelft SEAMPILOT Seam Tracking System]
University FAST seam tracker [37]. Many advanced robot systems, eg FANUC, KOMATSU TOMKAT and CML T3 746 series, have through-the-arc sensing as standard. The principle of through-the-arc seam-tracking, eg using the TOMKAT, is shown in Fig(2-7/a) for position control in the lateral direction, and Fig(2-7/b) in the vertical direction.

In the first case, when the weldgun deviates to the right or left from the weld line, the currents measured at left and right are different from each other. The weldgun is centralised along the weld line by making the differential current zero.

In the second case, the position control is achieved by moving the weldgun until a target current value, as shown in Fig(2-7/c) is obtained. This is based on the principle that a change in the arc current is proportional to a change in the stickout length of the weldgun.

With this self-adapting capability, the off-line programming environment could therefore be permitted to have inaccuracies. The maximum tolerance of errors is dependent on the robot system. For example, for the TOMKAT it allows up to +/- 20 mm/metre while the CML T3 746 robot Militrac system was +/- 15 mm/metre.

If a robot system has seam-tracking it is also usually capable of searching to locate a position on a component, eg the start point of a weld. The principle of the search as shown in Fig(2-8) uses the welding wire as the sensor. When the wire is applied with a voltage and comes into contact with the metal to be welded, a short-circuit occurs and the voltage drops to zero. When this occurs, the robot controller records the position of the point. There are various mathematical strategies and programming steps involved in the determination of a searched point. The respective procedures are documented usually in the robot user's manual. However, for completeness, Appendix(A2) provides an example to search for the starting point of a horizontal fillet weld.

External computer interfacing is a common function of all state-of-the-art robot systems. This usually takes the form of an RS-232C interface [38] and each system has an associated protocol for
The torch is weaving right and left. When the torch deviates to right or left from the weld line, currents at right and left become different from each other. By adjusting the torch position to make this differential current zero, the torch position is corrected to the weld line.

**Position Control in the Z Direction**

Position control in direction Z utilizes the arc characteristic that a change in arc current is proportional to a change in distance between torch and base metal (stick out length).

**Arc Current vs. Stick Out Length**

In other words, the torch height is controlled so that a target current in the figure on the right is achieved.

**FIG (2-7) Principle of Thru-the-Arc Seam Tracking [TOMKAT Robots]**
The unit sensor utilizes the fact that when the welding wire applied with a certain voltage comes in contact with the work (base metal), a short circuit occurs and the voltage becomes zero. Generally, when this takes place in an electric circuit, a large current flows, causing burning or other trouble. To prevent such trouble from happening during the search, a high resistance (R) is installed to limit current. In the above figure, a tester is shown to measure the voltage between the welding wire and the base metal. The search unit of the welding robot plays the role of this tester. The system takes as position data the point where the welding wire comes into contact with the base metal.

FIG(2-8) Principle of Weld-Start-Point Search Using a Robot [TOMKAT Robot Systems].
communication. Although in practice it is not a strict standard adhered to in present industry, it enables off-line programs to be transmitted bi-directionally between robot controller and a remote computer once the proper wire-to-wire configurations have been set up.

(c) An off-line program which requires no external signals to control its execution could be verified from within the robot controller. However, a program in the real industrial world, has interaction with other equipment, eg (i) a program pauses until it receives a 'high' signal (binary 1) from a sensor channel or (ii) just to turn on the air to a pneumatically activated gripper. In order not to have a test run of the robot program with the actual system, a simulated hardware environment is required. Since all external interaction occurs through digital or analogue signals, and there are custom-designed microchips or boards available to convert signals both ways, it is possible to produce an environment whereby an off-line program could be tested thoroughly prior to usage on the robot system. This should assist to eliminate bugs and provide possible optimisation for an efficient error-free off-line program.

2.2.3 Suitability Analysis of Available Off-line Programming Packages

Many companies that want to remain competitive are turning to CIM systems to gain them that competitive edge. In the early days, it would have been an extremely costly and risky venture to test a new robotic manufacturing cell by building prototype systems. The use of graphical simulation systems, eg GRASP [39-41], ROBOT-SIM [42], SIMULATIONS [43], ROBCAD [44] and PLACE [45], have reduced both the expenses and risks involved in implementing new facilities. Examples are given by Bell [46], Hansen [47], Mattis [48] and Dooner [39] showing the evaluation of the installation of a robotic system. Such systems have the capability to examine the reach, geometry, and task schemes prior to commissioning. However, at the time when WRAPS was conceived, these graphical simulation packages were beginning to be used only for the simulation of robotic work cells; none were reported to have been used for actual
off-line programming application although the potential was there. Some of these packages are now known to be adapted to a certain extent to make robotic welding programming and other applications possible, eg GRASP [49], Integraph, AutoSimulations, McDonnel Douglas, Cimstation [50,51], ROBCAD [52] and IGRIP [53].

Nevertheless, the application of commercially available graphical simulation packages, for the programming of individual manufacturing processes, such as for robotic arc welding, is difficult to justify economically for many companies whose only requirement is off-line programming. Not all the capabilities within such packages are required and utilised. These packages normally run on graphics workstations and can provide impressive animation. Fig(2-9) shows some examples of state-of-the-art systems. Typical costs of such software systems ranged from £25,000 to £100,000 for a single user station, exclusive of computing hardware used. A multi-user system may be as much as £300,000. The costs were obtained from various surveys in CADCAM International (EMAP Publication) during 1987/8. CADCAM implementation involves huge investments, as well as a total reappraisal of factory processes and long-term employee education. One example is quoted [54] where the initial cost of a system is £200,000 and the total running cost over a ten-year period may amount to more than £500,000. After a few years of CADCAM introduction in the 1980's, some feedback as to the payback as a result of such implementations are known. In the same paper, it was reported that 54% of CADCAM users in manufacturing achieved moderate-to-high pay-back and 46% achieved "negative, none or low".

Under unfavourable circumstances, if a company cannot justify a simulation or CADCAM package, there are basically only 3 alternatives:

1) No implementation. However, for companies hoping to remain competitive in a few years' time, the risk of not implementing may be even higher.

2) Use a skeletal system. Not many CADCAM vendors have provided this option where only a very minimal amount of the overall software package is tailored and offered for specific applications.
Top Left: ROBCAD
Simultaneous simulation of all moving elements in a robotic cell in 3D colour shaded solid models.

Top Centre: INTERGRAPH
Simulation of an industrial spot-welding application using a combination of wireframe and shaded 3D model.

Top Right: GRASP
Wireframe model of a KUKA robot and a Torsteknik manipulator in a simulated welding of a saddle joint for a boiler component at the UK Welding Institute.

Left: MCDONNELL DOUGLAS
Simulation of a multi-task cell with one robot applying sealant and the other manipulating the windshield for installation on a car body.

Most vendors do not want to keep track of many different versions of their software just to obtain some down-market sales. However, such an implementation may still incur considerable initial outlay because of minimum minicomputer hardware requirement.

3) Use the services of a bureau [55]. This has the disadvantage that there would be no in-house experience of the program generation process and a dependence on external resources. Services based on telephone/modem time hire on the CADCAM package could be expensive in the long run, and current CAD services have already revealed possible corruption of information transmitted and excessive computer wait time. Company's data security cannot be guaranteed.

Small batch fabricating companies have always presented special problems to technical and economic feasibility of robotic arc welding, including as explained previously the possible large time and costs associated with programming. Implementation in such companies has been slow. However, this is by far the largest sector of the fabrication industry offering considerable potential for robot applications and requiring, perhaps, only a few successful company implementations as a catalyst for the rest. It is anticipated that many of these companies, will find a less sophisticated off-line programming system applicable and affordable. Discussions with CML and batch manufacturers have revealed that a subjective target cost for a complete package, inclusive of hardware and software, would be £10,000 or lower.

A low-cost and practical system will find easier acceptance in developing countries, especially in the Pacific rim like Singapore, Malaysia and Thailand, which are rapidly moving into robotics and CADCAM applications [56].
2.2.4 Integration of Welding Databases and Knowledge

The adaptation of available simulation packages to off-line program generation for robotic arc welding is one step but still far from the requirements of a fully computer integrated robot welding system. The welding process is being automated but there is a necessity to automate also the entire flow of information required for the welding process to function: the planning, the design, the costing, the purchasing, the scheduling. The aim should be to harness all the diverse bits of information that pervade a welding fabrication company. This concept may be considered a sub-set of the whole CIM environment which is summed up as "the bringing together of separate islands of information, such as computer-aided design, process planning, material requirements planning, production scheduling, flexible manufacturing systems and other computer-assisted function into a unified system" [57].

The quality of a weld created as a result of an arc welding process is influenced by many variables. Expert knowledge is needed to determine a specification of how to produce an acceptable weld for a given application. This specification is known as a welding procedure [58] and shall include the following items:

(a) Classification, type and size of electrodes and, for automatic welding, the process materials.
(b) For manual welding, the size of electrode, welding current, open circuit voltage and length of run per electrode or fillet weld leg length and number of runs. For automatic welding and semi-automatic welding, the size of electrode, welding current, arc voltage, speed of travel, and rate of flow of gas and/or consumption of other process materials, as appropriate.
(c) Sketch showing the number and arrangement of runs in multi-run welds.
(d) Welding positions.
(e) Welding sequence.
(f) Pre-heating.
(g) Post-weld heat treatment.
Any fabrication company will create, in the course of normal production operations, a vast amount of data relating to welding procedures employed and their performance in practice. Invariably, in the current practice of most fabricators, much of this information is unrecorded or not readily retrievable. Low-cost technology is now available for automatic collection of much of this data. For example, many robotic arc welding seam tracking devices can be adapted, as already discussed in Sect 2-2-2 above, to automatically record the welding conditions and the state of the joint being welded. Other
required data can also be effectively recorded with a management information system. This has largely been made possible by board-level systems which plug directly into an expansion slot of a PC, or front-end boxed add-on modules and standalone units [62-64]. Some of these boards follow the CAMAC standard [62,65] which is the de facto standardised approach accepted for real time applications documented under IEEE/ANSI standards. It specifies mechanical, electrical and functional characteristics of systems modules, controllers, parallel and serial data highways and software, as well as those of hardware link between the other modules and the computer or external data highways. At the time of writing, the data acquisition hardware area was maturing and the future trend of data acquisition is aimed at the influence of fiberoptics and multi-vendor standard communication. As more senior management recognise the potential benefits of an integrated approach towards process control and manufacturing, computer-assisted data acquisition will proliferate [66]. In addition, there are also other systems which were specially designed and built for weld data monitoring. One example is the Cranfield 'Arc Guard' [67].

Data once stored, is of no use if it is not utilised effectively. A major factor militating against the use of flexible automation for welding in small batch operations, and indeed welding in general, is the problem of establishing or selecting the best available welding procedure for each particular joint or category of joint.

Therefore, there is a need to develop and implement a welding process information strategy, eg, new data may be collected and handled in a knowledge based system incorporating rules and heuristics. There is a need to automate the task of storing and retrieving of qualified welding procedure information. The expert knowledge-based system extension is shown in Fig(2-2/c) and would provide for a self-optimising process data base and the generation of process control algorithms. This can also be governed by expert rules to avoid some pitfalls related to data acquisition, eg excessive data transformation and manipulation which increases the risk of losing or changing the meaning of the information. Expert systems have emerged from the laboratory into industrial applications. The discussion about the use of such systems in engineering follows in Sect 2-2-5 below. In addition, there are many
factors to consider for eventual data storage, eg what information structure should be used so that the exchange of data with other systems is possible. It is emphasized that an ill-planned and un-coordinated attempt to introduce computerised data measurement and management can result in disaster. Neill in his paper [68] explains what could go wrong and how to avoid the pitfalls.

In conjunction with off-line programming, an expert knowledge-based data management system will ensure that data selected for a required procedure is maintained at an optimal state based on a knowledge of historical performance of similar or related procedures. The programmer in practice may be offered a variety of options. The options may be eg:

(a) discard current job considered for robotic welding for manual welding. This situation arises when there is absolutely no known data or the job, eg may require special welding techniques which may not be possible on the robot.

(b) Manual input of welding parameters. This may occur when a job is considered robot weldable and there are possible weld parameters to be tried from experience.

(c) Use empirically determined parameters when there are no historical data. Many researchers have over the years developed some empirical formulae which are applicable in certain environments [69-72].

(d) Use a known procedure when there is one, or a procedure nearest to the ideal could be used. Stored procedure should incorporate a known weld performance reference.

Conceptually, robot sensors which may initially provide only a seam following function can collect information on joint variability which, together with procedure and performance measures can be used in
statistical modelling. Process models created would be enhanced as more related data is collected. They would be subjected to a confidence testing and simulation until an acceptable algorithm is predicted. This would in turn be an option offered to the programmer.

Available simulation packages have shown that its off-line programming required considerable typed keyboard input. The dialogue between system and the user was mainly verbose questions and answers type. In addition, the user is expected to learn a particular syntax of input command statements in order to use the system. These were characteristics of even the high-cost simulation packages, eg GRASP, PLACE, and ROBCAD. At the other end of the off-line programming packages, is the fully textural type.

Fully textural off-line program generation demands even more from the programmer. The individual coordinate points and weldgun orientation angles at all robot points, in addition to all relevant command syntax, must be supplied to the system in a conforming structure. This method is time-consuming and error prone. The work carried out by Lee [73] clearly shows these limitations. ROPS [74] is one example where the program is comprised of blocks of text information describing the logic, robot point coordinates and process parameters for the programmed task. VAL-II [75,76], KAREL [77], SIL [78] and ROWEL [79] are examples of the high-level programming language type of off-line programming, eg compare with PASCAL. Bume and Jakob in a recent book [76] examined and compared the following programming languages: SRL, PASRO, AL, AML, VAL, HELP, SIGLA, and ROBEX. There are now also off-line programming systems whose language syntax is similar to the common BASIC language (eg the AR-BASIC for the Rediffusion Reflex robot [80] and the YAMAHA Robot Language for the YAMAHA robot [81]), or similar to the FORTH language (eg the MSP FORTH for the RTX robot [82] and ROBOFORTH II for the PROMETHEUS robot [83]). In addition to manual input of robot coordinates data, the user would also be expected to have some programming experience. The author anticipated that a compromise can fill the gap between a fully textural off-line programming system and other high-powered graphical simulation systems.

The main users of off-line programming systems for arc welding
applications were expected to be manual welders, welding engineers or personnel at technician level trained to generate the off-line program. It is therefore not expected that a great level of computer expertise is required. Users would be more inclined to utilise a package whereby the programming of a robot to weld a component is simplified by using an approach via a user-friendly interface.

2.2.5 Expert Systems in Manufacturing and Process Control

Expert knowledge-based systems are becoming more widely utilised both in the financial and manufacturing sectors of industry. Some indications of real life applications to which expert systems can be applied is provided by the work of the nine Expert System Awareness Clubs established by the Alvey Directorate [84-86]. The growth of expert systems applications in manufacturing is particularly significant. Expert systems are expected to play a major part of the predicted 40% application of artificial intelligence systems in manufacturing by the 1990s [84,87]. However, although expert systems have the potential of being adapted to many manufacturing processes, its usage should be approached with caution as expert systems were particularly suitable for solving problems that could be described with well-formulated heuristics. At least this was true for earlier expert systems.

The subject of expert systems, and particularly in engineering applications involving process control, is relatively new. The reader is referred to the following papers which provide some theory and background into expert systems: [88-91]. There are also now many books dealing with the subject, in particular [92-94] are good references for background knowledge.

The development of an expert system may involve the following types of people:

1) Domain expert - this is a specialist in a particular field with the ability to apply that knowledge to solve problems and make decisions.
Knowledge engineer - "Knowledge engineers are concerned with identifying the specific knowledge that an expert uses in solving a problem. Initially, the knowledge engineer studies a human expert and determine what facts and rules of thumb the expert employs. Then the knowledge engineer determines the inference strategy that the expert uses in an actual problem-solving situation. Finally, the knowledge engineer develops a system that uses similar knowledge and inference strategies to simulate the expert's behaviour" [94].

The knowledge engineer may also be responsible for translating the information obtained from the domain experts into a required knowledge base format as dictated by a particular expert shell. This role is also known as knowledge base author. It is possible that all the functions required to develop an expert system are performed by one person. Most manuals that accompany the expert system shell provide some guidance on the role and work carried out by the knowledge engineer, but two guide booklets [95] published by Expertech are particularly useful.

The applications of expert systems in manufacturing industries have produced the following benefits:

1) Speed up fault-finding and repairs.
2) Reduce downtime and production losses.
3) Boost throughput yet reduce energy consumption while raising automation levels.
4) Relying less on expert operating and maintenance staff.

An Expert-based system may be developed by using [96]:

(i) a conventional programming language like BASIC, FORTRAN or "C".
(ii) a declarative language like LISP or PROLOG or
(iii) an expert system shell.
There are several reports [97,99] that provide some guidance as to which approach should be taken. Using the first 2 approaches [99], there is complete flexibility in developing the expert software but this is offset by several disadvantages:

i) Since declarative languages like LISP or PROLOG were relatively new there would be difficulties in obtaining expertise.

ii) Expert systems developed are normally domain dependent. Hence only a small portion of the software for the application will be retained when moving onto another. In expert shells, only the knowledge for a new application needs to be acquired and formalised.

iii) Earlier versions of PROLOG or LISP did not support floating-point arithmetic.

iv) Commercial versions of such AI languages run comparatively slower in computer machines other than those which were specifically designed for them.

v) Prototyping of the expert system would require a considerably longer learning curve.

An expert system shell offers the comparative ease with which the knowledge base may be read and subsequently updated by welding engineers as knowledge advances. Rules created by a knowledge engineer, eg using the production-rule inference mechanism of an expert system shell, express knowledge in the form of "if..then" relationships. "if" is followed by a condition, and finally followed by a conclusion. This is in a form similar to the way that knowledge is often represented. An expert shell will also enable deterministic knowledge to be separated from company specific knowledge which may be required to be kept confidential. There are other knowledge representation techniques, eg
fuzzy-logic, and Bayesian. Additional information on these topics are found in the books on expert system theory mentioned previously. In addition, an evaluation report [100] carried out by the Department of Civil Engineering of Loughborough University in 1985 contains valuable information as well as details of the following expert system shells at the time: Xi, ESP/Advisor, APES, TESS, SAVOIR, EX-TRAN 7, KES and ENVISAGE/SAGE.

It is becoming possible to introduce a low-cost integration exercise of expert systems for off-line programming and data management systems as more expert systems were being made available on microcomputers [101-103]. The interface of an expert system to an application software is possible by software embedding. This technique will be examined in Chapter Four Sect 4-5-3.

Until recently, not much has been published on the application of expert systems for manufacturing processes. Earlier applications of expert systems were limited to consultative or diagnostic activities involving no immediate external feedback into the expert systems to affect its inferencing. Some applications, eg were:

(i) MYCIN - is an expert system which provides physicians with expert consultation on antibiotics use for the treatment of infectious diseases. It is one of the earlier expert system that has helped to bring its technology out of the laboratory environment.

(ii) PROSPECTOR - for exploratory geology. It can accept geological survey data and expertly produces maps and site evaluations, eg for ore prospecting.

Other application areas are (extracted from [91]):

Medicine: CABNET, INTERNIST, PIP, Digitals Therapy, Advisor, IRIS, EXPERT, PUFF, HODSKINS, HEADMED, VM, ONCOCIN.

Mathematics: MACSYMA.

Education: SCHOLAR, SOPHIE, EXCHECK, WHY, WEST, WUMPUS, GUIDON, BUGGY, LOGO, SMALLTALK, DIRECTOR.

Design and Fault Diagnosis: XCON, EURISTO, EL & SYN, PALLADIO, CMU-DA, DAA & EMUCS, SMX-Cogitor, CRIB, FAŁOSY.

Business and Office: TAXADVISOR, AUDITOR, XSITE, IMACS, ISA, IPMS, XPRESS, ILRPS, XSEL.

Others: MOLGEN, DART, SPEAR, SACON, AGE, LDB, CALLISTO, AIRPLAN, HYDRO, WAVES, GENESIS, CADECEUS, TATR.

There are considerable documentation to these earlier applications. Good sources of references are [89,91]. However, the use of expert systems are now found in many varied areas as Appendix (A3) will show.

The application of expert systems in industry, and particular in manufacturing, is an emerging technology. The most sophisticated system to date might be R1(XCON) [104,105] which is used by Digital Equipment Corporation (USA) to configure their VAX computer systems according to customers' requirements. R1(XCON) was reported at one stage to have over 3300 rules in its knowledge base and around 5500 component descriptions [106], but have since recently expanded to 6000 rules and component descriptions of 20000 [107]. R1(XCON) not only detects errors in the computer system configuration, but it also corrects them and makes any addition that are necessary to complete the system. R1(XCON) is being extended into the second system called XSEL.
For the discussions here, the emphasis is on the usage of expert knowledge-based systems for industrial control involving active feedback. This is the area of related research for WRAPS. The main goals for applying a real-time or pseudo real-time expert system-based control system with WRAPS are as follows:

- to optimise the welding process automatically, or as on-line and off-line advice to the operator, as regards to quality and overall production planning control.

- to quickly identify problems with the welding in progress, and execute immediate remedial actions that will normally require expert knowledge and data interpretation experience which may be beyond the capability of the robotic cell operator.

These goals will necessitate various sub-goals to be addressed, such as the interpretation of in-coming signals from the welding equipment to assess the prevailing situation in order to diagnose malfunctions and corrective measures needed to adjust improve on the given situation. These problems have been long addressed by the classical algorithmic approach, but with limited success. The main difficulty within this high level of control lies in obtaining a complete model of the welding process to be controlled or optimised. A generalised complete numerical model of welding process, considering even just for GMAW, is not available thereby preventing an effective algorithmic solution. An extended discussion to this subject is given in Chapter Four Sect 4-5-2. As an alternative, expert systems techniques are however, by their very nature, designed to deal with incomplete or vague or only qualitatively describable information. Furthermore, the operator is usually not interested in the very low-level sensor data which is difficult to collect and combine for interpretation. Modern control systems provide quite extensive data interpretation such as trend analysis and forecast, but this technique has approached the limit of its task as they do not enable the operator to understand the immediate situation and respond swiftly. Conventional systems lacked the understanding of the causal relationships of the measured data.
Earlier expert system shells were unsuitable for manufacturing applications because they lacked the external interfacing capability. However, recent shells have overcome this hindrance by their capability to exchange data with other applications. This is achieved by external read/write commands executed within the shell, and is now a common feature with all commercial shells. Other systems offer additional enhanced external interfacing capability through embedded software. Examples of such shells are KES [108], and recently Crystal [109] and MUSE [110]. This topic will be discussed further in Chapter Four Sect 4-5-3-1.

All these point towards the adoption of a suitable expert system shell for the WRAPS research.

2.3 Outcome of Survey

From this investigative study, the following needs have been identified:

i) a low-cost off-line programming system for the generation of off-line programs for robots assigned to specific manufacturing process.

ii) a simple and sufficient package to suit the applications.

iii) the use of expert knowledge to guide in the selection of welding procedures.

iv) closing the forward and feedback loop of the complete welding process to achieve a flexible welding cell within a CIM environment.
CHAPTER THREE
3.1 Introduction to Chapter

This chapter details the specifications of WRAPS, and the overall design considerations undertaken in creating WRAPS as a low-cost package to run on a variety of microcomputers. Some of the general issues, eg user-friendliness and a readable format of the off-line program, are also considered in developing this software package which has commercial application potential.

3.2 Specifications of WRAPS

In order to satisfy many of the problems discussed in Chapter One Sect 1-2, the general specifications of WRAPS are as follows:

i) The system should be capable of standing alone as a low-cost off-line programming package. A target cost ceiling of £10,000 was proposed including, if needed, the provision of suitable computer hardware.

ii) The system should be portable such as to obviate the need for particular computer hardware.

iii) The system should be transportable in terms of the variety of robot systems it should support.

iv) The system must be easy to use.

v) The robot program produced must incorporate process parameters necessary for process control.
and defined in normal process terms.

vi) There must be a calibration capability to overcome differences between the modelled environment and the "real" world.

vii) The system must provide facilities for program storage and communication in forms acceptable to the user.

viii) The system should be capable of expansion through interfacing with CAD/CAM facilities and with sensors, and capability of receiving feedback from the welding process to facilitate development of knowledge-based rules and quality control, etc within a CIM environment.

ix) WRAPS is designed specifically for robotic arc-welding applications but the system's software core should be adaptable with additional process knowledge for use in other areas.

Although not specifically required at the first stage of development, it is also important to consider that the above specifications would function in multiple robots as well as in single robot welding environment.

3.3 WRAPS System Development Considerations

The design of the WRAPS software had gone through a systematic definition. It had to be decided, (a) how the functionality of the task is to be achieved, (b) how the whole system costs could be kept to a minimum, and (c) what are the general issues involved in the design of a software package that has potential industrial application. It was thus important to consider the type of computer hardware and software tools to be used in order that a potentially large application could fit into a microcomputer as available in 1984. Also, because the WRAPS software
was intended to be used by inexperienced computer operators, it was also important to consider the software user interface.

3.3.1 Achieving the Task Functionality

It was required to consider at which level the off-line programming is to be performed. There are 4 levels to consider:

a) Joints - express position and orientation of individual robot linkages to affect position and orientation required of the tools for the task.

b) Robot - express movements in terms of world positions of robot.

c) Object - express task in terms of position and movement of objects within the robot working space.

d) Objective - express task in terms of actual work required to be performed, eg, "assemble box" "spray panel", or "weld seam".

In teach-pendant mode, the robot programmer drives the robot, i.e. the TCP, by a set of special keys programmed to move the robot in specific directions (in, out, up, down, left or right) and orientations (pitch, yaw, roll). The TCP is usually located straight out from the face-plate centre line of rotation at some distance commonly designated as the Tool Dimension. Some robot systems allow for TCPs which are offset from the centre line. Fig(3-1) shows some examples of TCP. Whatever movements the robot has been taught, it is always the TCP which is referenced in these moves. Tool orientation is how the tool is positioned angularly as related to a "global" frame of reference. If a TCP is moved and the tool orientation remains unchanged, the tool will retain the same global orientation throughout the move. Fig(3-2) describes this concept, which is discussed in many textbooks on robotics [21,111].
The robot moves from point (A) to point (B) with no orientation change. Two points are taught, (A) and (B).

A Move to change the tool orientation from point (B) to point (C). The Tool Centre Point did not change but there are still two taught points, (B) and (C).

The tool moves from point (A) to point (B) with no orientation change. Then the tool orientation changes from (B) to (C) with no change in the tool centre point. Next the tool centre point is moved while the tool orientation is changed to point (D) from point (C). Changes to the tool centre point and orientation can occur at the same time. Four points (A), (B), (C) and (D) are taught.

FIG(3-1) Examples of Robot Tool Centre Point and Tool Dimension.

FIG(3-2) Examples of Tool Centre Point Move and Tool Orientation Change.
Chapter Three

The programmer would also need to know the coordinates of the start and end points of a weld, and the required orientation of the weldgun relative to a seam at all points along the seam. WRAPS emulates this programming procedure but with added programming utilities to improve the speed and efficiency of the robot program generation. The programmer is not required to know how much the individual robot joints have to move in order to achieve the positioning coordinates required.

For special welding functions, such as weaving or crater-filling, the required program steps normally required individually are grouped as a macro. The command then to execute the weave function is called objectively, i.e. "weave-weld seam". The technique adapted for WRAPS is predominately levels (b) and (d) as described above. Effectively, Level (d) is derived from level (b) by accumulating the various steps into a macro. This is useful for robot functions like "move circle", "tack-weld seam" or "crater-fill". Level (c) could be used for the movements of component and/or equipment within the workplace.

Therefore, all 3 levels (b), (c) and (d) are used for WRAPS on various occasions. Level (a) is not implemented at this phase of the project.

3.3.2 Achieving Basic Low-cost of WRAPS

There are 2 main factors which contribute to the basic costs of the overall package – computer hardware and initial development software tools, and the development time.

Minimum hardware costs could be achieved by targeting the software to run on a system that contains the minimum specifications. The more recent progress made in microcomputers has attained a high performance/cost/size ratio and the decision to use a microcomputer other than a minicomputer or workstation, was inevitable.

Since the early days of computer technology when the mainframe computer was predominant, the advent of the microcomputer has changed
Chapter Three

the situation. The strengths and weaknesses of a microcomputer compared against a mainframe are as follows [112]:

Strengths of a microcomputer

- A range of very high quality general purpose software which is easy to use, powerful, low cost and which many users are already familiar.

- Increasing powerful hardware, complete with local disk storage and printing, in a low cost package.

- Rapid processing, good responsiveness (esp with the latest hardware and operating systems) and high quality graphics.

- Low development costs.

Weaknesses of a microcomputer

- Limited file sizes, in both operating system and physical capacity.

- Limited number of users - often only one.

- Poor support from many suppliers.

Strengths of a mainframe

- A corporate-wide source of data.

- Powerful and well controlled multi-user operating systems.

- Very powerful large scale processing and storage facilities.

- Excellent access security features.
- Excellent backup security features.

Weaknesses of a mainframe

- High costs in almost all aspects - hardware, software, support, access and development.

- Difficult and often long winded access.

- Slow responses.

Software development costs could be kept low by effective and easily maintainable software code while maintaining maximum flexibility for future development and expansion. The overall concept of WRAPS was fully developed and it became evident that the development time induced would be mainly due to transforming the ideas into software program.

3.3.3 General Issues in Designing WRAPS

The issues considered here are related to the needs and characteristics of intended users and to the task for which WRAPS will be used. In the early stages of the research, some of the issues that may affect hardware areas, such as the type of computer, operating system or choice of input device (keyboard, mouse etc), were less important as they were to a large extent dictated by availability at the time. The issues will be concentrated in the areas of dialogue and screen design. In actual commercial applications hardware considerations are important, and will be considered together with the environment in which it will be used and a host of other pre-marketing factors [113].

In designing the software user interface, the main criteria which the author believed to be of most importance were:

1) Conceptual structure - finding the most effective method of affecting the user task. Alternatives
considered include fully textural off-line programming, fully graphical programming, vocal programming or a combination of each.

Fully textural systems have zero representation of the equipment and environment in which the programmer can visualise. Thus a thorough knowledge of the process, equipment and environment is assumed. Data input to a textural system to form a robot program is error-prone as all the robot functions, individual coordinate point values and other parameters must be entered.

A fully graphical system would provide maximum visualisation, but there are certain data parameters that are best met by textural input, eg, entering decimal values for welding voltages or wire feed speed, or filenames for storage of data.

Vocal programming was still in its infancy and it would be useful if, eg, a programmer who is physically handicapped.

A mainly graphical approach to the programming technique, with textural input where necessary, appeared to be the best compromise. Fig(3-3) shows an approximate relationship of costs/graphical-sophistication/automation associated with the degree of off-line programming and simulation. The data was obtained from manufacturers concerned or extracted from several surveys carried out in the CADCAM International (EMAP Publication) and represented graphically. It is obvious that the costs of a system increases substantially with increased sophistication in graphics capability and the degree of increased programming performance. The extreme of graphical programming

Wire-frame modelling of robots/components/workplace. Mainly for workcell simulation with collision detection, but with robot program generation capability. Uses some textual input.

Fair amount of graphics (skeletal models) to aid in establishing coordinates of eg welding seams. Mainly textual input of data and commands. Simple model of objects but capable of providing some view control by revolving the object.

Some windowing graphics to facilitate user friendliness.

Verbose entry. User need to define coordinates of eg weld seams, and enter all necessary commands.

Legend:
○ Expected Position of WRAPS

may be to use the type of advanced dynamic 3D animation involving stereo images with ray-tracing to show the reflective properties and hidden surfaces [114], for example as demonstrated in motion pictures like *TRON* and *The Last Starfighter* [115,116]. It is certain no manufacturer could afford such techniques in the near future as for example, the latter motion picture required a Cray X-MP supercomputer, currently the world's fastest computer, to create its 40-minutes worth of simulation for some of its sequences. The type of graphical representation used for WRAPS is discussed in Sect 3-5 below.

In addition, a graphics concept based on windows, icons, mouse and pull-down menus (or WIMPS as is now commonly known) was believed more efficient. Although this concept was not widely publicised, the impetus to adopt it for WRAPS was obtained from a reference [117] of the work carried out at the Palo Alto Research Laboratory in the United States. This concept is now widely adopted in the microcomputer industry and has been implemented on various operating systems, eg GEM, TAXI, TOPVIEW, MacIntosh ToolBox, X-Windows, NeWs, and Microsoft WINDOWS [118,119]. The graphics-based user interface is also predicted to have wider applications in commercial software packages in the future [120].

ii) Dialogue type - where choices include questions and answers, menu selection or the use of command line languages. If the latter is considered, how terse or verbose should the interaction language be, and what style should be used. Simplicity and ease of use was a major factor, considering that the users of the WRAPS package would possibly be personnel who may not have substantial exposure to
computer usage. It was considered that a menu selection type would be more appropriate. The amount of textural input, typically via the computer keyboard, is minimised.

Also, it was considered whether the software would be used in a foreign language. The structure of the WRAPS package could cater for such a requirement, eg, by isolating all messages in a separate file, which are read and installed into the system when the package program is started.

iii) Extent of user control - how much should be pre-determined by the software. There was a need to consider what should be given parameters and what the user should be allowed to select. In most respects, the programmer should be given as much information as available to reduce the time and skill needed for creation of a robot program.

v) Type of screen presentation - monochrome or colour in low (300 x 200 pixels), medium (600 x 400 pixels) or high resolution (1024 x 1024 pixels). In an ideal situation, a high-resolution colour version would enable better visualisation as well as impression. The distinction between the objects being programmed and the equipment used could be better. One object could be easily differentiated from another. However, the costs of high-resolution colour hardware equipment were a magnitude of at least twice the monochrome lower-resolution system. A monochrome medium-resolution screen was the compromise choice in terms of achieving the functionality and low-cost element of the WRAPS package.
3.3.4 The Hardware and Software for WRAPS

The resources for the project, both hardware and software, are discussed next.

3.3.4.1 The Microcomputer System

The computer used for the software development was the Future FX20 model. It was marketed by Encotel Systems Ltd [121]. It is a 16-bit system based on the Intel 8088 chip, and was claimed to be an IBM PC lookalike. The FX20 comprised of a 25 rows by 80 columns monochrome monitor, supplied with 256 Kbytes RAM (which was later expanded to 640 Kbytes in the later stage of the project), twin 800 Kbytes floppy disks, and a QWERTY style keyboard. It was supplied with CP/M-86 operating systems as standard and MS-DOS as an option. However, the initial MS-DOS version was suitable only for reading of IBM PC single-sided floppy disk. In view of the software tools available for the FX20 at the time, especially the language compiler and graphics utilities, only the CP/M-86 was suitable for an early start on the project.

The computer was more of a business applications orientated machine and graphics capability was not immediately available. However, the choice of the computer was constrained by project financial resources and was preferred in comparison to the IBM PC, an industrially accepted machine, because the FX20 appeared to have technical superiority in certain aspects and was over £1000 cheaper. At the time, the Macintosh computer was available and was the best system for graphical applications because of its WIMP environment, but again its costs was prohibitive. Another suitable contender was the Commodore Amiga, which was suitable in terms of costs and specifications, but it was only appearing in the US and was not to be available in the UK some 9 months later after the start of the WRAPS project. Although the time delay was acceptable for the fact that it will not prolong the overall software development in real terms, it was not taken on-board for development of the WRAPS software for political reasons established in conjunction with the research industrial collaborators, CML.
3.3.4.2 The Software Implementation

The software development environment follows the classical edit-compile-link sequence. In other words, an ASCII text file containing the program is first created. This is fed to a language compiler which produces an object file. The object file, and if needed in combination with additional object files and/or library files, then serves as input to a linker which produces an executable file. WRAPS was intended to be used across a wide range of microcomputers, and it was important to consider its portability.

3.3.4.2.1 Achieving Software Portability

Portable software code is code that will compile and run correctly with little or no modifications in a variety of languages compiler or operating system or hardware environments. The portability element of the WRAPS project includes the graphics environment as well as the primary language in which WRAPS is coded. The primary language considered here is the "C" language and the graphics environment is the Graphics Extension System (GSX).

(i) The "C" Language

"C" [122-126] was developed as a system programming language, designed to make the process of writing software for things like operating systems quicker and more efficient. It is descended from BCPL, which has not been popular to the same extent. Both BCPL and "C" have been used to write operating systems, eg UNIX [127] and AMIGADOS [128]. These were developed with a very small amount of Assembler, eg in UNIX about 95% of the code was written in "C".

Assembler languages work directly with a computer's built-in instruction set. The programmer will have to think in terms of the hardware and specify every operation in the machine terms, eg move these bits into a register and add them to the bits in that other register, then place the result in memory at this location and so on. They are very tedious to write and are prone to programming errors. Compared to higher level languages, the programmer can think in terms of the problem
at hand rather than in terms of the hardware. A logical structure could not be visibly imposed on the program in Assembler. For example, writing $a = b + c$ in a higher language does the same function as writing

LHLD .c
PUSH H
POP B
LHLD .b
DAD B
SHLD .a

in the 8080 assembly language.

Compared even with BASIC, "C" is quite weak, particularly as regards to string handling. It is also inclined to be cryptic, and tends to make very heavy use of pointers. It is quite possible to write phenomenally incomprehensible code in "C". "C" is also distinctly lackadaisical about typing even though it is a typed language. It is possible to convert types easily using an operation called "casting".

Despite all these seemingly disadvantages, "C" has grown into a major language. At the beginning of the WRAPS project in 1984, "C" was hardly used in the UK commercial world. Even at Loughborough University there was no resident expert on the "C" language. However, the author's felt compelled to favour "C" as the developing language for WRAPS on the following reasons:

1) "C" is like a "super-assembler". It is relatively low-level to enable a programmer to specify every detail in a program's logic to achieve maximum computer efficiency. It is also high-level enough to hide the details of the computer architecture thus promoting programmer's efficiency. If one has appreciation of what the compiler actually does, it is possible to write very efficient code in "C" in terms of overhead and execution speed. It is easy to link "C" functions with Assembler routines where an extra 20 to 30% in speed is
vital. "C" has standard functions that enable bit manipulations, and declare variables to be stored specifically in machine registers. "C" provides facilities for structures and pointers and especially the latter, is almost the same as a machine address which is used frequently in Assembler.

Being reasonably low-level, "C" does not support direct operations on arrays, strings and sets. There are also no input/output parameters built in to the language. Input/output operations are performed by functions in the standard input/output library which are part of the compiler's package.

"C" lends itself much better than Assembler to writing in a "structured" manner, with independent intercommunicating functions. Structured programming is a discipline intended to make programs easier to read and write, based mainly on a modular structure. Because of the modularity, "C" has the ability to build complex programs out of simple function elements. A modular structure lends itself to a developing environment in which programs are written independently by members of a team and compiled and debugged separately. It is easy to build a customised library of useful functions and repetitively use them to reduce development time.

Contrary to some popular belief, it is possible to write readable "C" code to aid in program maintenance and debugging.

"C" is very much the native language on many operating systems. These include, eg TRIPOS, AMIGADOS, UNIX, and XENIX. Using "C" it is
particularly easy to make full use of the facilities provided by such operating systems. By taking the "C" route, it is possible to be working with the computer rather than fighting against it, eg, using the "C" approach, the software will not kill the multi-tasking of the AMIGADOS operating system.

iv) "C" is supposed to be a portable language, though in practice it is never 100% portable. However, "C" will at least ease the task of transporting the software across a variety of computers. This is important in respect of developing a package like WRAPS which has commercial potential and could be required to be used on a variety of computers and operating systems.

v) The time saved in writing a software package in "C" rather than in Assembly can be very significant. Assembler programmers on large projects who have switched to "C" report that after a few months experience, they can be 10 times more productive generating working debugged code. The resulting code may run a bit slower than if it was written in Assembler, but in many applications the time penalty is acceptable.

The "C" language allows portable code to be written, but the programmer is not forced to do so. Writing portable code requires practice and a certain amount of knowledge about the targetted computer and its associated operating system. Programmers tend not to write portable code because it is a little more difficult to write than it is just to write a program that performs its function.

Efficient coding in the "C" language may often mean functional but illegible code that requires a considerable level of "C" expertise to read. Many good "C" language books and articles, eg see [129-132] refer to such dangers, and also provide some examples and guidance in writing
efficient and readable software.

The following general approaches were taken to achieve and maintain portability for WRAPS:

- no huge programs. All programs were divided into smaller units and where possible used as overlays. This also facilitates debugging.

- the size of data objects are not assumed. Where data size is necessary and used within the software, eg in dynamic allocation routines involving malloc(), calloc() etc, it is initially determined automatically by the sizeof() function. Hence, for example, in allocating memory space to an integer, even if it is known that the size for a specific computer is 2 bytes, use malloc( sizeof(int) ) instead of malloc(2).

- declare all function return types. It is common "C" practice that functions which return an integer do not need to be declared, but make sure that the return statement still return a value, eg 0, even though the returned may not be used. If a function returns nothing, declare the function as of type void.

- document other functions, eg specific calls to the host operating system, that may present portability problems and isolate these within a library file.

(ii) The GSX Graphics Standard

GSX [133] is the graphics system extension to an operating system providing graphics capability through standard operating system calls. It is based on the ANSI Virtual Device Interface (VDI) to provide portability across systems and devices and is functionally based on the
Graphical Kernel System (GKS) concept [133-136]. But GKS, which had a better chance for universal acceptance, was not available for the Future FX20 computer. GSX manages the differences among these devices by providing a library of device drivers (special software programs) which communicate with the computer.

GSX comprises of 2 parts as shown in Fig(3-4).

i) Graphics Device Operating System (GDOOS)
   It provides the standard interface between the application program and the specific devices. The application access GDOOS in much the same way that it access the disk operating system.

ii) Graphics Input/Output System (GIOS)
   It provides the standard interface between GDOOS and the specific devices. It contains device specific code required to interface to a particular device to the GDOOS. A particular device driver may be loaded dynamically into RAM when required by an application.

The GIOS perform the graphics primitive of GSX which is consistent with the inherent capabilities of the particular device. The graphics primitives or op codes are all defined in the GSX manuals. Some common functions of primitives are:

- open_workstation - initiate a graphics device, i.e it loads the device driver if necessary.
- polyline - output a polyline.
- text - output text at a specified location.
- set attributes - this may, eg, set the text size, type of line, colour, type of shade, etc.
- fill area - display and fill a polygon.

The GSX implementations on the Future FX20 had some minor
GSX is the graphic system extension to the CP/M family of operating systems, providing graphic output through standard O/S calls. GSX is required by all Digital Research graphics products which load it from disk prior to execution. GSX includes a set of supported, graphic output device drivers.

Three major components make up GSX: the Graphics Device Operating System (GDOS), the Graphics Input Output System (GIOS), and the GENGRAF utility.

- The Graphics Device Operating System (GDOS) is analogous to the Basic Disk Operating System (BDOS) in the CP/M system and contains the device independent part of GSX.
- The Graphics Input Output System (GIOS) is the device-dependent module that provides the glue between GDOS and the protocol of a specific graphics device.
- The GENGRAF utility configures a graphics application to run in the GSX environment.

The GDOS and GIOS path to graphics devices is essentially parallel to the BDOS and BIOS path (See Figure 1). BDOS handles logical operating system calls, such as reading or writing to the console or a disk drive. BIOS provides the device-dependent interface. GDOS intercepts and services graphics calls, by first loading the required device driver module and then passing it on to the GIOS.

GSX insures that the proper driver is loaded as needed. Thus, only one device resides in memory at any time. Requests for new work stations are handled automatically, maximizing the memory available for the application programs.
limitations. A special program called 'GSXTEST' was written to determine the conformity of the computer graphics hardware to GSX. The output of 'GSXTEST' for the French-made Leonard graphics board inside the FX20 is shown in Fig(3-5). Compared with the GSX manual [137, pages C-4 to C-9], it is obvious that full requirements are not met, but the FX20 system has the most basic primitives implemented, i.e. drawing a dot, a marker, straight line, polygon and fill. All the other primitives, for example arcs, circles or display special text characters or bold lines, could be simulated from software. Colour could be implemented if a colour graphics card/colour monitor combination is used.

3.3.5 Fitting WRAPS into a Microcomputer

The major consideration here was to minimise (i) the total software code, and (ii) the size of the eventual package.

3.3.5.1 Minimising Software Code

This could be achieved by efficient coding, and careful use of library functions. The importance of efficient coding and its influence on software portability has already been discussed in Sect. 3-3-4 (b).

Careful use of the standard library functions which are supplied with the compiler, could reduce the size of the executable code. To quote a simple example from experience consider the following 2 "C" programs:

(a) #include <stdio.h> /* Header File. */
main() /* Main function. */
{
    /* Start of main function. */
    putchar('K'); /* Calls putchar function */
    /* to output the character */
    /* "K" to the screen. */
}
/* End of main function. */
FIG(3-5) Output of Program 'GSXTEST' Showing the Graphical Capabilities of the Leonard Graphics Board fitted in the FX20 Microcomputer.
Chapter Three

(b) `#include <stdio.h>
main()
{
    printf("%c", 'K'); /* Calls printf function */
    /* to output character "K" */
    /* to the screen. */
}

Both the functions, `putchar()` and `printf()`, are part of the standard library. The above 2 programs write the character "K" to the screen. But when compiled with the small memory model, and linked with the necessary functions from the library to form the executable code, (b) was 52% more in program code size than (a). In addition, there was an increase of 1% more data in (b).

3.3.5.2 Minimising Size of WRAPS Package

Two approaches were taken to make the overall package.

3.3.5.2.1 Minimising Program Size

An estimation of the eventual software program size was made to anticipate the amount of computer RAM space required. The minimal code and data storage required for a basic prototype of WRAPS was some 300 kbytes. See Appendix (A4) for this estimation.

In early microcomputers, the majority of operating systems reside in RAM space, rather than in ROM which some modern microcomputers have. The amount of RAM occupied varies according to the operating system used, eg a Concurrent CP/M-86 system may take up about 100 kbytes.

Thus, it would appear that a microcomputer with up to 400 kbytes was required just to run a very basic WRAPS system. A few years ago, a microcomputer RAM size of 256 kbytes was considered very large, and moreover, not many microcomputers at the time had that amount of RAM installed. In view of the large memory required for WRAPS, a modular design approach was adopted. The individual modules of WRAPS are
discussed in detail in Chapter Four (Sect 4.3.2). This involves the creation of the WRAPS program which is comprised of several files called overlays [138].

By using a modular design, a large program need not reside in memory all of the time. For example, a user may select one of a number of functions from a menu-driven application. Since the functions are separate and invoked non-concurrently, it is not necessary for them to reside in memory simultaneously. When the function currently called is complete, control is returned to the menu portion of the program from which the user selects the next function. When using overlays, a program is divided into sub-programs and loaded from disk only when required.

Fig (3-6) shows the concept of overlays. If a menu-driven application program consists of 3 separate user-selectable functions, each function requires 30 kbytes and the menu portion 10 kbytes, then the total memory required is 100 kbytes (a). However, if the 3 functions are implemented as overlays (b), the program only requires 40 kbytes.

There are some restrictions on program overlays, being defined as Overlay Method 1 and 2 in the programming guide. The method used in WRAPS is Overlay Method 1. The 2 prime restrictions are (i) all overlays must be on the default disk drive and (ii) the overlay names are determined at compile time and cannot be altered during run-time. Design and maintenance of overlays by Method 1 is simpler and the restrictions did not hinder the development of WRAPS.

The modularity offers WRAPS users a phased investment program should WRAPS become developed as an integrated software package. WRAPS could be provided with a module for every need, so long as they are integrated through a common style data structure. This enables an attractive low-cost startup by purchasing the modules which are initially needed thereby realising early return for each investment step.
FIG (3-6) Using Overlays in a Large Program to Reduce Computer Memory Requirements.

FIG (3-7) Use of Text-Blocks Pointers to Access 'HELP' Information From a File in an Overlay Environment.
3.3.5.2.2 Minimising Data Memory

The reservation of RAM space for arrays requires a size to be declared. This has a limitation. Consider an example where the complete data from a disk file of unknown size is to be transferred to computer memory. It is apparently difficult to estimate how much of array storage is to be declared. If too much of memory space has been declared to hold the file, the computer RAM is wasted. Specifying too small an array would cause some data to be lost, and in some cases, could cause catastrophic failure of the program.

The solution was to use dynamic memory storage which was possible through memory allocation functions provided, eg malloc(), calloc() etc [1391], which are part of the standard library. Considering malloc(), for example, allocates a specified number of bytes in RAM, large enough to hold the required data. Another library function named free() releases memory which was reserved by the memory allocation functions, thereby allowing the memory to be used elsewhere.

Data that may need to stay in memory, eg messages which are only occasionally used could be stored permanently on disk and read into the program when required. This should be used when a certain amount of delay in response from the computer is acceptable, eg accessing help information.

3.3.6 Providing Help to WRAPS Users

The author examined the "help" facilities on many standard software packages and found that most were not efficient and help needed is not directly accessible or always available. In some cases, there were insufficient help and it was necessary to refer to the manuals for further information before continuation. A user should be provided with help for all options available in a particular menu-type selection so that even a complete novice could be guided in using the software without referring to thick manuals.

In commercial software packages, the user is normally expected to
press a "help" key and a menu of help options appear. The user then selects the choice of help required, or maybe selects another sub-choice again before the relevant information is drawn and displayed from a file.

The technique used for all the WRAPS modules is different. The user could press a dedicated "help" key anytime when there are options to be selected from, and instantly the assistance required is extracted and displayed without going through a further help option selection. This reduces the computer response time and makes the software more efficient to use.

The structure of all the main system modules are based on overlays as discussed in the previous section. All the modules are based on menu options selection and providing assistance at all times could be achieved in 2 ways:

(a) Each option being linked to a file that contains the specific help information for that option. Consider the Programming Module at the time of writing, there were 45 module units and each module has 12 options to choose from. Hence, if this method was used, there would be 45 x 12 = 540 individual files, just for help information. Ordinary microcomputer floppy disks would certainly be unable to hold that many file names.

(b) Store all the help information on file, but use direct file pointer addressing to access the right section of the file for the particular help needed. The file pointer to or within a disk file is synonymous to the exact address of the location of the relevant information within a file.

The method (b) is also used in a similar way in the "help" facilities examined in some commercial packages, where the "help" file is divided into distinguishable partitions. Such an example could be found in the "help" information file of the CCP/M system [140]. However, to make the "help" facility available for all options, at all times and for all the separately linked modules, a special method has to be improvised.

Page 62
Consider a simple example to illustrate the method used. Fig(3-7) shows 3 different software modules and the method in which the "help" facility is accessed. The main module could call and use one of the overlays at any time. Each module has 3 options.

The help information for all the options are initially prepared using an ordinary word-processor in a simple format. An example for the main module format is shown as follows:

This option permits you to exit the program. To activate option, press <RETURN> key to return to the host operating system. You will be asked to confirm whether you really want to exit the software. You may answer yes or no by typing a "Y" or "N" when asked.

This option enables you to rotate the modelled object in 3D space so that you could view it from any angle. You can also zoom in to examine the object in more detail.

To rotate the object, use the arrow keys on the keyboard. You can rotate the object to the left, right, up, down. Use the keys labelled "LEFT", "RIGHT", "UP" or "DOWN" respectively.

To zoom in, press the key labelled "IN". To zoom out, press the key labelled "OUT".

This option deletes a file from the floppy disk. You will be asked the name to be removed. The validity of the file name will be checked to see if it exists. If it does, you will then be asked to confirm again if you really want to delete it. Remember, once a file is deleted you will not be able to recover or use it, unless you
The character "\" is used as a de-limiter to separate the individual blocks of help information which may be of any number of bytes. This "help" file is then "parsed" by a software utility (called W-INDEX) specially written to count the number of characters within each block of information. Note that a count in a "help" file may not correspond to a count on another operating system, even though the "help" files are supposedly identical. The reason is because some operating systems, eg UNIX, store a new line as a simple ASCII linefeed character but on others, eg CP/M and MSDOS, as a combination of an ASCII carriage return and a single ASCII linefeed. It is therefore necessary to use W-INDEX to create the required index file on the targeted operating system.

The count information is then stored in a "help" index file (called PROGHELP.HLP, MODHELP.HLP, ONLHELP.HLP and EXPHELP.HLP for the Programming, Modelling, On-line and Expert module respectively) which is subsequently accessed when required during operation.

The action of the menu options in all the software modules, including overlays, are controlled by a software function ('choose_option' in WRAPLIB library) which checks which option key has been pressed. If the "help" key was pressed, it passes the relevant block information to the "help" function. The blocks of information may be in any order so long as the right block address is passed to the option select routine which calls the "help" function.

If the "help" information file needs to be updated, the editing again is carried out with a word-processor. It is then only required to run the parsing utility which automatically creates the INDEX.HLP file.

3.3.7 WRAPS Limitations

Any microcomputer implementation, has its limitations. On earlier microcomputers, graphics capability was one of them. Microcomputer
systems generally had a defined resolution of 600 x 400 pixels which was considered high-resolution in 1984/5. Typical high-powered commercial graphical systems can go up to 1024 by 1024 pixels supported by powerful computing capabilities to give fast pictorial representations, even to 3D solid modelling applications. But these systems are expensive at about £10,000 each. The graphics obtained on a microcomputer, without special enhanced graphics add-on boards, may not be as impressive as graphics workstations, but they suffice for the task in hand and it offers the basic low-cost element required for WRAPS. It was planned to upgrade to a higher resolution computer graphics system when low-cost enhanced graphics system were available. This is greatly facilitated by the use of a standardised graphical system extension, such as the 85X of Digital Research Inc. Since the start of the research, many enhanced graphics systems, such as an IBM ATx with an enhanced graphics adaptor (EGA), is available at reasonably low costs, ranging between £1500 to £4000. In Chapter Seven, the porting of the prototype WRAPS package over to an IBM ATx with enhanced graphics is discussed.

To improve graphical simulation in WRAPS, and because of the restraint imposed by the initial basic microcomputer dynamic memory (RAM) limitation of 256K, only the robot end-effector is displayed. The RAM of the FX20 was eventually upgraded though, to 640K to meet the requirements of other WRAPS functions, eg more off-line program space and modelling of more complex components.

The robot joints are not displayed to reduce computational time involved in graphical simulation and for other reasons as will be outlined below. This has enabled a smoother animation of the weldgun movements within the robot programming space. Simulation of the weldgun alone does not affect the off-line programming capability of WRAPS. In robot controllers, the movements and orientation of the robot TCP are governed by a mathematical control algorithm. Several control models are used, eg, geometrical, kinematic and dynamic [9,141-143], the most common being the kinematic whilst dynamic control algorithm was still very much a research topic. Accepted methods proposed, eg, based on the methods developed by Denavit-Hartenberg [144,145] will suffice for the T3 and other 6-axis jointed robot arm. This method, generally termed as forward kinematic solution, derives the position of the robot arm in
space, given the 6 joint variables. In the majority of robotic arc welding implementations, the programmer mostly manoeuvres the weldgun and the focus would be at the tip of the weldgun. This requires the inverse kinematic solution [146,147], which is thus given the TCP, find the joint variables to achieve it.

The inverse kinematic modelling solution has been found for most commonly used industrial robots and could be incorporated into the WRAPS system if required. It is also reported that a generalised solution is found using a modified iterative algorithm based on the Newton-Raphson technique [148]. The temporary omission of kinematic control of the robot model in WRAPS is justified since the main intention is to research and develop a low-cost off-line robotic programming system that can provide many other special features. The concept-proving of off-line programming and these other special features was deemed more important. In teach-pendant programming, the robot points are taught only when the arm has come to a rest, i.e. on a point-to-point basis. This is to be the simulated programming procedure for WRAPS.

Adding full kinematic modelling at an early stage of the project would not have helped to prove the concept of WRAPS but would only prolong development time of the overall package. Furthermore, an inverse kinematic model of a robot cannot guarantee the accuracy of the joint movements. One would really need to consider the dynamics of the robot system. In fact, in similar reasoning to the inverse kinematic model, it becomes necessary to consider the inverse dynamic model. The accuracy and repeatability of a robot system is affected by many factors, some examples are the elasticity of the joints, play in the gears and bearings due to wear and tear and tolerances involved in the manufacture of the components, friction in moving parts, and variations in the motor torque to drive the joints. The topic is beyond the scope of WRAPS initial aims.

Impossible robot movements unknown during off-line programming can be detected by a "dry cycle" run. Later, on-line adjustment could be made if necessary, during a replay of the robot program via a manual step-by-step mode (or more commonly known as teach mode). This enables a check for collisions and is safe because the robot program should be
tested at a fraction of the actual robot automatic service speed. However, a limited amount of robot programming control is built into WRAPS to provide a check on the infringement of the extremes of the robot reach capability. Collisions are likely to occur between the weldgun and the component to be welded. The provision of software functions to allow zoom-in and viewing angle adjustment of the modelled environment, especially around the weldgun/component region, can assist in the visual detection of collision.

Thus only the geometric model of the weldgun, assumed attached at the end of the wrist, was used in WRAPS. As the robot joints manoeuvrability depends largely on the control algorithm, this in turn has limited the use of different tool dimensions during a particular session. Currently, the TCP of the robot system assigned in WRAPS must always be defined at the tip of the wire. In practice, if a user moves a jointed robot arm through the teach pendent, more steps will be required to move the TCP to another location at a fixed orientation because of the number of axes involved. But this will have negligible influence in the creation of the off-line program, since all the required coordinate data will be generated automatically by the software. The limitation on the tool dimension will be removed once the robot control algorithm is implemented.

The prototype WRAPS was developed around CML robot systems but it is relatively straightforward to extend the system to a wide range of robots, and applications. For example, (i) welding domain functions can be grouped and made available as a library, (ii) the input/output signals are general in application as they can be linked to control any peripheral device of choice so long as the signal is, or made to be compatible, and (iii) the robot functions names could be replaced by an appropriate set. The control algorithm when implemented for the CML T3 robot can be replaced by another algorithm, eg for a SCARA type of robot. A library of different robot models can be made available.

Every robot point must have a function assigned to it. Currently only the following robot functions are supported by WRAPS:

\[ \text{Page 67} \]
DELAY - pauses the robot at the taught point for a specified duration of time before proceeding to the next taught point. This is a frequently used function for process actions such as arc initiation, tacking and crater-filling.

INPUT - reads the digital signal from an input port, eg "INPUT 1" reads the digital status of input port channel 1.

POSITIONER - has options for switching power to a programmable work positioner ON or OFF, i.e., "POSITIONER ON" or "POSITIONER OFF". Also, "POSITIONER TILT" is used to record the positioner coordinates after it has been tilted by a certain number of degrees from the horizontal position.

The term "positioner" or "work positioner" is used throughout this thesis to represent work orientation devices which are also known as table positioner, work manipulator or table manipulator. It is preferred to "manipulator" which is best used to reference devices which are akin to robots and similarly categorised systems.

MOVE - this function specifies a robot coordinate shift. Future implementation will include the options of normal move (NORM) or Continuous (CONT). NORM is used when the robot moves from a point to another, decelerating and stopping at the target point, then accelerating to reach the next point. CONT is used when the robot arm is not required to stop at a point but to move continuously on to the next point.

NO_MOVE - This function holds the robot at its current
position whilst some other function is being executed, eg, as required with the (CML) T3 robot, to adjust the orientation of the work positioner.

TOOL - this function toggles the state of a digital signal at the appropriate channel, eg "TOOL 1" toggles the state of output channel 1. It is assumed this signal is used to drive a relay which controls the operation of some form of auxiliary device, eg a gripper, or to start a conveyor system.

PERFORM - enables a given task to be considered and used as a subroutine, eg "PERFORM 01" where the program steps contained in a subroutine 01 is executed. This is taught once and then used whenever required by calling a sequence number. The function may be used with conditions, eg "PERFORM 01 IF condition" but at the present stage of development, only a "PERFORM always" is implemented.

VOLT_SELECT - this function within WRAPS specifies the welding voltage used for the welding. This is equivalent to DAC 1 for CML T3 robot but a translation is required to equate and output the actual voltage value since DAC 1 is based on an digital-to-analogue conversion. The concept of using DACs and its conversion is discussed in detail in Sect 4.6.

WELD - This has 2 options, "WELD START" and "WELD END". The former switches the arc on and the latter turns it off. When the arc is on, it is assumed that the wire feed is also initiated. This function assumes that the weldgun has been located at the appropriate
start or end weld point and also that the welding conditions has already been selected (using VOLT_SELECT and WFS_SELECT).

WFS_SELECT - enables selection of the wire feed speed. In the CML T3 robot this is equivalent to DAC 2 and similar to DAC 1 above, requires a conversion to output the actual value to the wire feed equipment. Conversion details are discussed in Sect 4.5.

These were sufficient to enable an off-line program created on WRAPS for welding and indeed also for other applications, such as assembly or sealing. The robot function names as required for specific robot systems or applications are definable by the user. For example, instead of "weld" which performs arc-welding, the replacement name of "seal" would initiate the glue application equipment. In general, the commands used in WRAPS are universal by definition and are therefore only dependent on the post-processing for formatting into the correct robot program structure. Other robot control parameters are also selectable or modifiable through WRAPS programming commands:

velocity - rate at which the robot TCP is moved.

flags control - a flag is a variable whose value, indicates the attainment of some designated state or condition by an item of equipment or a program. The flag is subsequently used as a basis for conditional branching and similar decision processes. Typical values used for a flag is 0 or 1.

tool dimension - distance of the TCP from the face and along the centre line of the roll plate of the robot arm. Previously explained in Chapter Three Sect 3-3-1.
WRAPS at the present stage, permits the modelling of a single robot and a single work positioner as its working environment, herein referred to as "active equipment". Any other model of "inactive" tools or objects that require no manipulative control (i.e. requiring only positioning within the working environment) can be modelled and used providing there is sufficient computer memory to generate and store the models.

All models are displayed in 3D wireframe. The representation to create the wireframe objects is discussed in Sect 3-5 below. Wireframe models are defined only in terms of points and lines and as they contain no knowledge of the surfaces lying between such lines or of which parts are "solid" and which are in "fresh air" and hence cannot be used reliably to detect clashes between parts. Hidden line capability is not automatic in wireframe models and is also not implemented in the prototype WRAPS. Not all options displayed on the menus are presently implemented but are included to project the whole concept of WRAPS and its intended capabilities. All options are intended to be incorporated and enhanced in the future. The creation of all aspects of the WRAPS package at this stage will exceed the time available for the project.

3.4 WRAPS Layout

The software structure and screen layout are discussed below.

3.4.1 Software

When the WRAPS project was initiated in 1984, graphics capabilities, including software tools for graphics implementation was in its infancy on microcomputers in the UK. On the Future FX20 there was no graphics being used even by its manufacturers. It has therefore been necessary to write special graphics application routines to execute all necessary graphics including 3D transformations. At the time, the author believed that icons and windows would dominate the future. Hence utilities were specially written to generate what is now commonly known as the WIMPS (Windows, Icons, Mouse and Pull-down menus).
All the software developed as part of the WRAPS project may be classified under 2 main groups - Application and Development.

(1) Application Software

The WRAPS package comprises 4 main system Modules which were conveniently labelled according to functions:

i) Modelling
ii) Programming
iii) On-line
iv) Expert

(2) Development Software

"C" language functions which are repeatedly used in various software units are sub-grouped to form a library which is linked with the application object code to form the run time program Module. There are 5 main library groupings. These are:

i) GRAFLIB - GSX graphics drawing routines. This includes pop-out menus and windows.
ii) MATHLIB - Mathematical and 2D and 3D transform routines.
iii) MISCLIB - General utilities routines, including low-level bit-by-bit assignment routines.
iv) ALFALIB - ASCII escape code routines, mainly screen functions in alpha mode.
v) WRAPLIB - Other special routines related to WRAPS.

In addition, in the Programming and Modelling Modules, each has its own special library called PROGLIB and MODLIB respectively. Lastly the "C" programming language itself requires its own application library called CLEARL.LB6 for large memory model [139] and CLEARS.LB6 for small memory model applications.

Other software tools were also developed which are used as aids in
using parts of the WRAPS. Three most important tools are (1) Byte-by-Byte Editor, (2) Hexadecimal listing of a disk file and (3) Icon Design Generator.

3.4.2 Screen

While in the graphics mode, the maximum display area covers 510 by 320 pixels which is dictated by the graphics board and its associated device driver, in this case the Leonard Graphics Board supplied through Future Computers. This resolution is mapped to 32767 x 32767 device units under GSX. The simulated environment is displayed in the viewport defined by 32568 x 20800 in GSX device units. The boundary layout is static except for the values associated with the various parameters that are associated with the simulated environment. During a picture redraw, only the viewport area is erased. Also, the screen may alternate between alpha and graphics mode depending whichever is required.

3.5 Graphical Representation of Objects

Every year sees a huge increase in the amount of literature dealing with various aspects of computer graphics. Keeping up to date can be a time consuming activity. For basic understanding, the author referred to at least 5 books [149-153]. To keep abreast with developments, the following journals/magazines were found to have especial value:

- Computer Graphics and Applications (IEEE Publication)
- CADCAM International (EMAP Publication)
- Computer Graphics World (Penwell Publication)
- 3D (EMAP Publication)

A computer screen has a finite number of points making up each line across the screen. The maximum number of distinguishable points which a line may have is a measure of the resolution of the device. The greater the number of points, the higher the resolution. Each point is called a pixel (short for picture element) and is the smallest addressable screen element. Each pixel has an address which corresponds to a defined
coordinate system attached to the screen. The GSX graphics system has the zero reference point of the coordinate system at the bottom left corner of the screen and maximum addressable pixel at the top right most corner of the screen.

Computer graphics images are made by setting the intensity and colour of the pixels which compose the screen. For example, a line segment is created by setting the brightness, i.e. by turning it on, of a string of pixels between a starting pixel and an ending pixel. The screen may be considered as an array of pixels which holds the internal representation of the image. This array is also sometimes called the frame buffer.

The 3D images are generated from a series of line segments which are represented by the coordinates of their endpoints relative to a cartesian frame of reference. Any alteration in the image is obtained by performing mathematical transformation operations on their coordinates. These transformations may be scaling, translation or rotation about 3D space, and also projection of a 3D image onto a flat screen for perspective display. The mathematics involved in WRAPS are standard techniques involving homogenous coordinate matrixes and are found in many textbooks [149,150,154]. However, the implementation technique to create, and procedure of transformation of WRAPS objects, are different to take into consideration the software and hardware resources, especially the GSX graphics extension.

An object can be held as a numerical representation in many ways in computer memory. Once a model exists, it becomes a straightforward task to perform the necessary mathematical transformation(s) to produce arbitrary views for projection onto a computer screen for visualisation. Several schemes are available to represent an object. These are wireframe, surface model, and solid modelling. The latter could be further grouped under several approaches, for example, constructive solid geometry, boundary representation, and faceted modelling. Their relative disadvantages and merits are summarised in Fig(3-8). There are now many published articles and textbooks dealing with graphics and modelling schemes and the reader is referred especially to [155-160] for further information. A paper published by Nichols [161] covers a broad
Advantages | Disadvantages
---|---
Fast model creation | Ambiguous models with possible nonsense objects
Fast viewing | Picture representation deficiencies - silhouette edges - hidden line removal
Low demand on processor | Limited geometric construction capability
Quick editing of individual points/edges | No general mechanical property calculation facilities
Small data storage requirement | User is expected to visually detect interferences between wireframe objects

Table 1 The wireframe scheme

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast model creation</td>
<td>No connectivity between surfaces leaving ambiguities</td>
</tr>
<tr>
<td>Can model general surfaces</td>
<td>Mechanical property calculations limited to single faces</td>
</tr>
<tr>
<td>Generates curve of intersection between 2 surfaces</td>
<td>Cannot automatically section an object</td>
</tr>
<tr>
<td>Fast editing of surfaces small data requirement</td>
<td>User is expected to visually detect interferences between bodies</td>
</tr>
</tbody>
</table>

Table 2 The surface scheme

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No ambiguities</td>
<td>High demand on computer</td>
</tr>
<tr>
<td>Automatic hidden line removal and sectioning</td>
<td>Geometric coverage does not yet include sculptured surfaces</td>
</tr>
<tr>
<td>Mechanical property calculations of parts and assemblies</td>
<td>Medium to large data storage requirements</td>
</tr>
<tr>
<td>Automatic interference detection</td>
<td>Considerable future potential</td>
</tr>
</tbody>
</table>

Table 3 Solid modelling scheme

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguous models with possible nonsense objects</td>
<td>Picture representation deficiencies</td>
</tr>
<tr>
<td>Limited geometric construction capability</td>
<td>No general mechanical property calculation facilities</td>
</tr>
<tr>
<td>User is expected to visually detect interferences between wireframe objects</td>
<td>High demand on computer</td>
</tr>
</tbody>
</table>

Surface models overcome many of the wireframe model ambiguities.

Facetted modelling uses hundreds and thousands of planar surfaces to approximate the actual shape.

area of the various model representations.

With reference to Fig(3-8), it is obvious that solid modelling was not suitable for the Future FX20 computer initially available for the WRAPS project. A compromise which encompasses basic wireframe to the facetted modelling technique was used. This technique is discussed in the next section.

3.5.1 Rendering Edges to the Object

The following techniques used are capable of modelling objects using basic wireframe, surface and facetted representation. Where the objects are simple, for example a cuboid, the wireframe is used. A complex object is created similar in principle to the Construction Solid Geometry or Combinational Geometry technique. For cylindrical or cones, a surface or facetted type of representation is used. This is important as pipe intersections which are frequent in welding fabrication work, could be modelled this way.

Two schemes are used to render edges to an object, depending on the geometry of the object being modelled. The first scheme is for a polyhedra object which may be of a regular or a irregular face of constant cross-section. With this scheme, a face section of the object, is initially constructed from essential dimensions requested from and provided by the user. Consider an example in modelling a hexagonal bar shown in Fig(3-9/a). The user is asked to input the desired location of the origin of the bar, the number of sides required (in this case 6), the size of the circumscribing radius around the hexagonal shape, and finally the length of the bar. The vertices forming the face of the hexagon bar, is then constructed with reference to the starting point. The other face of the bar is then projected from the first face by the length of the bar.

The sides, or "flesh" of the hexagonal bar is created by linking the vertices between the 2 faces. Two methods could be used here. The first linking method starts off from an initial vertex, then linking up all the other vertices of the first face. Next the second face is linked
Both Sequences Require 24 linking Steps. But the Number of Coordinate Points Need Only be 12.

FIG(3-9) Rendering Edges to a Hexagon Bar.

For a Polyprism, the Rendering Sequence is Defined by the User.

FIG(3-10) Rendering Edges to a Polyprism.
to the first in the fashion shown in Fig(3-9/b). The second linking method is shown in Fig(3-9/c). Both methods are equally suitable because a computer plotting algorithm to link up all the vertices is easily derived and implemented.

The second scheme of rendering edges to the object is used for polyprisms. All the vertices of a polyprism being modelled has to be defined consecutively and are linked in the same order. Some points would need to be doubly defined in order that all necessary edges are made. Fig(3-10) shows an example of this scheme.

A common technique used to form a representation of an object on the computer screen is to link all the point vertices in a defined manner by making individual calls to a line drawing function. However, this method is not used for WRAPS. The 2 schemes used in rendering edges to the WRAPS object make the best use of a GSX function called polyline() which draws a line between the points stored in 2 arrays, one containing the x-coords and the other the y-coords. This contrasts with the use of the drawline() function which is called every time a link between 2 points is required. These coordinates are the transformed coordinates of the perspective view to be displayed on the computer screen from the actual x-y-z object coordinates. Each object segment in WRAPS is therefore stored in 2 arrays for the visualisation stage. WRAPS uses variable size arrays which are dynamically allocated using calloc() to optimise memory utilisation. Hence to draw an object segment, only 1 call is made to the function. This enhances the animation which was slow and limited on earlier computers and was highly desirable for the Future FX20 computer used for this research.
CHAPTER FOUR
Chapter Four

CHAPTER FOUR

IMPLEMENTATION OF WRAPS

4.1 Introduction to Chapter

The implementation of the 4 main modules of WRAPS are discussed (Sect 4-2 to Sect 4-5). The Programming module creates the off-line program which is described in full detail (Sect 4-3-2). A calibration procedure to compensate for the differences between the modelled and the real world is also discussed.

The problem associated with the incompatibility of various software input into WRAPS, and WRAPS output to differing robot systems are also examined (Sect 4-4-1). It describes a possible solution to this incompatibility for the low-cost system. The communication capability available in the On-line module are discussed in Sect 4-4-2. The protocols used in communication between robot systems are also described.

Finally, interfacing methodology of the expert system shell, KES, to the WRAPS applications are discussed in Sect 4-5-3.

4.2 The Modelling Module

This module allows the definition of work components, tooling and the welding system. The current Module is capable of modelling the following object segment shapes, Fig(4-1):

(a) Straight lines.
(b) Regular polyhedra - cuboids, cylinders.
(c) Irregular polyhedra - irregular face, constant cross-section objects.
(d) Polyprism.
FIG(4-1) Primitives Used in the Modelling of Objects with the Modelling Module.
(e) Pre-defined shaped, eg, T-sections, L-sections.

The basic methodology involved in definition of an object is shown in Fig(4-2). In WRAPS, a model always remains in memory in 3D and at full scale. The displayed picture is a 2D representation and scaled down or up to fit into the WRAPS screen. A complicated object is divided into segments which are created individually and stored on disk. The segments may then be joined using an interactive function to form an overall object.

Regular constant cross-section objects may also be created by selection from a pre-defined face which is stored as an icon. When an icon for a required shape is selected, all the necessary dimensions for such an object are entered interactively to the computer.

Weld-points depicting the start and end of seams may be defined and saved in the modelled object. This may then be used in the Programming Module for direct homing of the weldgun tip. Examples of weld points creation is given in Chapter Five Sect 5.2.2.

Each point is created and stored dynamically to overcome unnecessary memory storage associated with using global pre-defined memory arrays. This is important in a memory limited environment such as a microcomputer, and is explained in Chapter Three Sect 3-4.

When the system is in operation, a menu is displayed where the user can make a selection. Errors encountered are reported instantaneously on the screen. The process is totally interactive with the minimum of user keyboard entry requirements. Disk files holding object data may also be read into the module for editing or proofing. Data files are stores in ASCII format so that simple geometrical data files may also be created in the required format using ordinary word-processing software.

The software is modular in construction and can be upgraded with more facilities. These additional enhancements are suggested in Chapter Seven Sect 7-2. Each option is structured in a hierarchical tree and could therefore be extended with more options when required. Fig(4-3) shows the overlay structure of the Modelling Module. Appendix(AS)
FIG4-21 The Sequence in the Creation of a Model of an Object with the Modelling Module.
These modules are on the same 2nd Overlay level.

FIG(4-3) Overlay Software Modules of the Modelling Module.
Chapter Four

describes the functions of each of the main software units.

The module also provides some utilities to interact with the operating system, eg deleting a file on disk, show directory of files, or sending data to the screen or to the printer.

4.3 The Programming Module

This module manipulates the object models and accepts welding programming commands to produce a robot program which incorporates all the necessary welding parameters retrieved from a database. The models are displayed in 3D perspective form. A robot program consists of many taught points which are programmed in a logical sequence to enable a task to be carried through. In this respect, a taught point is a point in space at which the TCP is located and where the tool orientation is defined in a specific way with an associated function to perform at a specified velocity of the TCP. A function is the activity that occurs at the point being taught. It can range from a simple move to a complex perform-on-multiple-conditions instruction.

The Programming Module software is a tree-like structure as shown in Fig(4-4). The main menu unit, PROGRAM, remains permanently in RAM. The other units are overlays which are loaded into RAM as and when required. There are 3 levels of nested overlays. The functions performed by the individual units are discussed in Appendix (A6).

When the Programming Module is started, a title is displayed to introduce the user to the Module. This is followed by a disclaimer, stating the use of WRAPS for generating off-line robot programs and their use for safe operation on a robot system is entirely the responsibility of the user. The main menu unit then reads in 3 information files:

i) GENERAL.DAT - contains data values that determine the maximum number of (a) model points, (b) segment units to form a model, and (c) weld points.
FIG (4-4) The Overlay Tree-Structure of the Software Modules of the Programming Module.

Page 85
The main menu unit uses this information to allocate RAM to hold these data requirements. (a) is used to reserve space for the GSX arrays [137] in order that the desired maximum number of model points that forms the object model may be displayed. It is stressed here that the storage RAM for the actual model is allocated only before the data points are read in a later overlay unit, READALL. The RAM space for the exact number of object model points is reserved. This makes efficient RAM usage, as explained in Chapter Three Sect 3-4.

ii) SETWORLD.DAT - contains information that organises the work environment: relative positions of all the equipment of the cell, default view parameters, default robot base position, default robot and weld process parameters.

iii) PROGHELP.DAT - contains a list of addresses of the starting section within a disk file that holds all the information that is used to aid the user. The technique used here to extract the assistance required is unique from most software packages examined and has been described in Chapter Three Sect 3-3-6.

The creation of the robot off-line program is based on a three-legged branching tree structure as shown in Fig(4-5). The main sequence branches out to another sequence if required, and this branching can also branch out again from the new sequence and so on. Each robot point within a sequence can also be referred to as a "node" in normal tree-like representation description. The reader unfamiliar with tree structures will find [162] informative. At the moment, the software limits the maximum number of branches to 29 (hence a total of
FIG(4-5) Three-legged Structure of Off-line Robot Program Points.

```c
struct robot_program {
    double program_num;
    struct robot_coords tcp;
    struct process_parameters weld_params;
    struct robot_program *main_seq;
    struct robot_program *feed_of_path;
    struct robot_program *sub_seq;
}

struct robot_coords {
    double rob_x;    /* Robot x coord */
    double rob_y;    /* Robot y coord */
    double rob_z;    /* Robot z coord */
    double rob_d;    /* Robot pitch D angle */
    double rob_e;    /* Robot yaw E angle */
    double rob_r;    /* Robot roll R angle */
}

struct process_parameters {
    int current_robot_function;  /* robot function */
    int current_velocity;        /* Velocity Table */
    int current_arc_voltage;     /* DAC 1 */
    int current_feed_speed;      /* DAC 2 */
    int current_tool_diam;       /* only 1 dia is implemented */
    int current_variable;        /* variable used */
    int current_flag;            /* flag used */
    int current_output;          /* output channel, tools */
    int current_input;           /* input channel */
}
```

FIG(4-6) "C" Structure of a Robot Program Point which is Created Dynamically in Computer Memory.
30 sequences including the main sequence). In theory, there is no limit to the number of branches from each robot point. The modularity of the off-line program which is based on sequences, offers a distinct advantage because a complete program could be built out of various separate modules or macros. The benefits begin to emerge when i) a big job cannot be completed in one go, and ii) a particular robot task is to be repeated at other times.

The storage of the robot points (or nodes) for the robot program are created dynamically as they are programmed in. Each robot point follows the "C" language structure as shown in Fig(4-6). Each robot point has a point number s.nnn conforming to a format which will be described in Sect 4-3-2 below. "s" is the sequence number and "nnn" a robot point number within that sequence. Associated with each robot point, as in the real robot system, are the robot TCP coordinates and orientation angles. The other elements in the robot point structure are 3 pointers which points to the same structure as itself. The first pointer points to the next robot point in the same sequence. One robot point in the same sequence will point to another until the close loop instruction (CLOSE PATH for CML T3 robot) is encountered. The second pointer is used to assign the robot point to which the present sequence is to close loop to. If it is not a close loop instruction, then this pointer is zero otherwise it contains the robot point description "s.nnn". The third pointer points to the first robot point of a new sequence if required, otherwise it is set to zero.

Apart from the robot points tree-like structure being created, some additional data is also stored to keep track of the sequences. These data include the count of the number of robot points in that sequence, the welding voltages and wire feed speeds used in their respective order of appearance in that sequence, the order of PERFORM instructions issued etc. These are created and stored dynamically for each sequence in another "C" structure as shown in Fig(4-7). This is necessary to help create the flow of logic in the off-line program.

In operation, the user has graphical representation of the weldgun which is attached to the face plate of the robot wrist, although the robot itself could be displayed in the "home" position if required.
#define NUM_OF_PERFORM_SEQ 30
#define OPTIONSALS 5

struct per_sequence {
    double end_path_point;     // The pt where an end_path ends */
    int num_pts_in_seq;        // # Num of pts in the */
    int num_delay;             // # particular Sequence. */
    int order_delay_time[OPTIONSALS];  // # Keeping track of the order of */
                                       // # occurrence within the sequence */
                                       // # as there may be more than 1 */
                                       // # such function but with with */
                                       // # different time parameters. */
    int num_tilt;              // # Num of work positioner */
    double order_tilt_status[OPTIONSALS];  // # TILT commands in seq. */
    int num_rotate;            // # Num of work positioner */
    double order_rotate_status[OPTIONSALS];  // # ROTATE commands in seq. */
    int num_tool1;             // # Num of times tool 1 i.e. */
    int order_tool1_status[OPTIONSALS];  // # Order of ON/OFF status. */
    int num_tool2;             // # Num of times tool 2 i.e. */
    int order_tool2_status[OPTIONSALS];  // # power to arc-welding m/c */
                                       // # is turned On or OFF. */
    int num_input1;            // # Num of external devices */
    int order_input1_status[OPTIONSALS];  // # Order of ON/OFF status. */
    int num_input2;            // # their order of */
    int order_input2_status[OPTIONSALS];  // # occurrence within the */
    int num_input3;            // # sequence. */
    int order_input3_status[OPTIONSALS];  // */
    int num_sel_dac1;          // # Num of times the voltage */
    int order_sel_dac1[OPTIONSALS];  // # parameter is selected */
                                       // # within the sequence and */
                                       // # the order in which they */
                                       // # occur. */
    int num_sel_dac2;          // # Num of times the wire */
    int order_sel_dac2[OPTIONSALS];  // # feed rate is selected */
                                       // # within the sequence and */
                                       // # the order in which they */
                                       // # occur. */
    int num_perform;           // # Num of calls to sub-sequences */
    int order_perform_call_pt[NUM_OF_PERFORM_SEQ];  // # within this particular sequence */
    int order_perform_called[NUM_OF_PERFORM_SEQ];  // # Ordered list of */
                                       // # the pt name of */
                                       // # the points that */
                                       // # is carrying the */
                                       // # PERFORM command. */
                                       // */
    int order_perform_called[NUM_OF_PERFORM_SEQ];  // # Order of the sub- */
                                       // # sequences to be */
                                       // # performed. */

    ]);}

FIG(4-7) "C" Structure for Holding Additional Information About Individual Robot Program Sequences.
Absolute positioning and orientation of the weldgun is used and the joint constraints are evaluated only in terms of its reach capability. These reach constraints are user defined values which are read into the Programming Module from the data file called GENERAL.DAT described earlier under this same heading. View change (view rotate, view shift to the left or right, or zoom) of the working environment is achieved using defined arrow-cursor keys (in, out, left, right, up, down) on the keyboard.

Functions to inspect or alter parameters which affect the robot activity and immediate ancillary equipment are included. These are for velocity, DACs, flags, input/output signals and status of the work positioner.

An object model read in from the Modelling Module may not contain all the necessary weld points already pre-defined to aid in the homing in of the weldgun. It is also anticipated that in the future, pre-processed model data of objects imported from other CAD packages may be used by the Programming Module. Hence the Programming Module also permits the creation of start and end of weld seams.

Once the seam path is defined, the robot can be taught its path. For each seam, the associated weld parameters required, eg, voltage and welding speed, must also be entered. This can be entered directly through the keyboard, or search routines incorporated within the Module can be used to select from external database which may contain the most updated weld procedure to be used for the particular seam. This search capability is further expanded in the Expert Module, Sect 4-5-1.

Additional facilities to assist off-line programming are:

(a) Examine distance from the TCP to a point on the component.
(b) Examine distance between any 2 points on the component.
(c) Toggle display of a vector symbol to show location and orientation of the TCP, and also another symbol to show location of the global reference.
point.

Robot off-line programs created may be inter-nested to form a complicated multi-subroutines program as already shown in Fig(4-5). This structure resembles the general organisation of most compiled or interpreted BASIC language. The program consists of a main function which at some required point(s) branch out to call another function. In BASIC, this may, eg, be made via CALL or GOSUB statements. Upon completion of the called function, control is returned to another point in the main or at some other point as required. This may be achieved via RETURN or GOTO in BASIC.

One of the greatest benefits of off-line programming is manifested in the usage of previously created program sequences. For example, a "warm-up" cycle programmed once in an application could be saved as a "relocatable" sequence. A relocatable sequence could be re-used and inserted into a new off-line program, thus saving enormous amount of programming effort and time. The format is similar to the normal off-line program and is described in Sect 4-3-2-2. An example of the use of this powerful facility is illustrated in the programming exercise in Chapter Five Sect 5-2-2-2.

Before operating in automatic mode, the robot system loaded with an off-line robot program, is likely to require "fine-tuning" to remove any joint constraints or program path clashes occurring to correct any positioning errors or inaccuracies in the system. The restrictions on joints constraints and clash detection are already discussed in Chapter Three Sect 3-3-7.

4.3.1 World Calibration

In the linking of the modelled robotic system to the real-world system, it is vital that WRAPS has sufficient fundamental accuracy in its storage of geometrical data of the elements in the system. Mathematical evaluations within WRAPS could be guaranteed to within 8 decimal places, but it is the input of geometrical data into the WRAPS system which ultimately determine the accuracy of the whole simulation.
Chapter Four

The geometrical data includes measured data from the work floor and could expect to differ from actual dimensions. Other factors, which are discussed below, may also eventually affect the accuracy of the modelled environment. It is thus equally important to have all the modelled objects to be positioned and orientated accurately in relation to other objects within the robotic system.

It becomes necessary to calibrate and compensate for the differences between the modelled environment and real world. The real world is the welding cell as described in Chapter Five Sect 5-2-2. Several methods of calibration are possible and considered and have been already discussed in the literature survey of Chapter Two Sect 2-2-2. A simple prototype calibration procedure has been devised. It is simple but effective.

Every robot system has inherent errors built-in, and these errors deteriorate with usage. Errors are normally related to its positioning accuracy and repeatability. It is essential therefore that a robot system is calibrated when first installed for use, and subsequently at defined intervals of time during service. It is with intention that the calibration method developed would take into account this varying adverse condition.

The method developed involves using the robot system itself as the measuring instrument during calibration. This is important because no 2 robots, even of the same make, would perform an identical robot program without small deviations. This is caused by a combination of the control system calibration and the tolerance problems inherent in the manufacture of the robot linkages. The latter, including the lack of rigidity in the robot joints, give rise to variations in joint offsets which may be small but could compound to produce quite large errors at the TCP. There are no published results of calibration of work environment with a simulated environment but it is known that most techniques developed or being developed calibrate the modelled environment to the real environment at the off-line robot program level. ROPS [74] uses a method that falls into this category, as it make amendments to its eventual off-line robot program to run in the actual robot setup. The WRAPS method attempts to have a calibrated setup in the
modelled environment.

Repetitive tests have also made conclusive results that the accuracy and repeatability of a robot could vary from position to position within its working range [25]. The term position refers to both location and orientation of the TCP. In this respect, it appears important that the calibration of the robot arm should be carried out in the region in which the welding process is to occur. This is the basis of the calibration developed for WRAPS.

The method used now is suited to applications requiring positioning repeatability of +/- 1mm minimum. It was particularly developed for use on a robot system equipped with some form of seam-tracking. Most robot systems have this capability, especially the "through-the-arc" sensing method. This was to be also the initial project requirement but was later dropped due to change of equipment availability for the WRAPS project. Discussions with welding robot manufacturers, CML, TOMKAT and FANUC, revealed that positioning repeatability of up to +/- 10mm was sufficient for applications with a seam tracker.

The calibration method makes use of 2 sets of data, one from the known dimensions of a reference object and the other from the measurements of the reference object. Currently, the WRAPS calibration software contains the following "objects" for calibration references:

- a point, a line, a square planar frame, a cuboid, and a sphere.

The sets of data contain essential coordinates of the reference "object". Here the word "object" is within quotes because the object may be a real object or imaginary. Imaginary objects are especially referred to points, lines or planar frames. The other "objects" are real objects and are physically required to exist at the calibration location, such as a cuboid frame or a sphere. The following paragraphs describe the principle used, based on an example on the square planar frame.

Consider the square planar frame to be attached to the work positioner as shown in Fig(4-8). The frame may be attached to any object...
Dimensions of the Reference Frame (mm)
Tolerance: ± 0.5mm

Use of the Robot System as the Measuring Tool to Obtain Dimensions of the Reference Frame. The Measuring Point of the Robot is the Tool Centre Point Set at the Tip of the Welding Wire.

Chapter Four

and anywhere in the cell. However, the existing software can only position it at a location defined in the reference object data file, typically of file type *.REF. There will be facilities to adjust the position and orientation of the frame in its locality. The planar frame may be real or imaginary as already mentioned. The latter will provide a "near perfect" planar frame whilst the former will be a frame produced to specified tolerances. If it is imaginary it could be marked out on to the flat surface of the work positioner. The objective is to use the robot to measure the 4 corner coordinates of the frame.

The tip of the welding wire is made to coincide with the TCP of the robot for "measuring" purposes. The robot is then made to approach each corner of the planar frame at a fixed orientation angle, using the eye in this case, to locate the tip of wire to the corners of the frame. It is possible to locate to within a bandwidth of 1mm accuracy visually although other measuring instruments could be attached to the robot end to pick up or identify the coordinates of the planar frame.

The measurement from the robot may be recorded in 2 ways:

(i) by manual recording of the robot coordinates and storing it away in a file on a remote computer.
(ii) by automatic recording where the robot coordinates are transferred through a communication link between robot and remote computer. This data transfer is discussed in in Sect 4-4-2.

The "measured" data file is always of file type extension *.MEA and the "referenced" data *.REF. The measured file may be an average version of several sets of actual measurements to obtain better statistical acceptance. When both types of file are available to WRAPS, the actual calibration of the modelled environment to the real world could commence.

The first step involved averaging out the planar frame which is effectively in 3D space. It may be distorted as shown in Fig(4-9/a). The averaging method used is as follows.
Measured Units:
$\text{Pt1} = 1, 1, 1 \ [x, y, z]$
$\text{Pt2} = 0, 11, -1$
$\text{Pt3} = 11, 12, 0$
$\text{Pt4} = 10, 2, 1$

$\text{Pt1}_x_{\text{next}} = 0.5 \left( \text{Pt1}_x + \text{Pt2}_x \right)$
$\text{Pt1}_y_{\text{next}} = 0.5 \left( \text{Pt1}_y + \text{Pt2}_y \right)$
$\text{Pt1}_z_{\text{next}} = 0.5 \left( \text{Pt1}_z + \text{Pt2}_z \right)$

After every 2 averagings, the frame is returned to its original orientation but diminished in size. Further averaging will reduce frame to a near perfect square frame where its lengths are equal and deviation in depths of the frame is zero. Averaging is stopped after an acceptable accuracy or when a perfect square is obtained. For the example above, a 0.1 units away from a perfect square frame is reached after 10 iterations.

FIG(4-9) Averaging of a Robot Measured Frame As Used in the Calibration of the Top Working Surface of the Work Positioner. See Main for Explanation.
Consider the frame as shown in Fig(4-9/b). The first averaging step bisects the adjacent sides of the frame, reducing its size by a factor of 0.25. This bisecting continues which return a new frame to its original horizontal position but again reduced by another factor of 0.25. Thus for every 2 averages, the frame is in its original orientation but reduced size. The averaging process is carried out repetitively every 2 times until the measured frame approaches a square of specified limits of geometrical properties, eg the square must not be more than 0.1mm out of misalignment, and depth of distortion no more than the same amount. The limits criteria is more important on the "z" or depth axis of the 3D measured planar frame because the sides of the frame always approaches a "perfect" square first during the averaging process. Once the measured frame is acceptable at its reduced size, it is "stretched" back to about the size of its original.

The second step in the calibration method is to align the reference frame in the modelled environment to the measured frame. To align a planar frame to another planar frame it is only necessary to align 3 coordinates of the frames. Thus only points 0, 1 and 3 are used although any other 3 coordinates may also be applicable. Initially point 0 of the reference frame (and effectively the whole work positioner) is pulled towards point 0 of the measured frame. Next the reference frame is rotated about point 0 until its point 1 is in alignment with point 1 of the measured frame. Finally, point 3 of the reference frame is rotated about the plane formed by point 0 and point 1 of the 2 frames until to within an acceptance level currently set at 0.1 degrees of each other.

The calibration technique applies in general to other "objects". The only differences are in the measuring points used and the mathematical averaging involved according to the "object" used. It would thus appear that the point and the line reference "objects" are the easiest to use.

The calibration for the programming example in Chapter Five used a planar square frame.
4.3.2 The WRAPS Off-line Program

This is the "neutral" off-line program generated from the Programming Module of the WRAPS system before it is post-processed to a format that is usable by the robot system controller. The "neutrality" factor is important in satisfying specification points (iii) and (vii) which was listed in Chapter Three Sect 3-2 to achieve robot system independence and storage acceptance to the user. It facilitates translation to other formats since there is no current universally acceptable robot program format. There are efforts to develop a standardised format and is spearheaded by the CAM-I group as will be further described in Sect 4-4-1-2.

Some robot systems, especially the pick-and-place types, uses a top-to-bottom single sequence of instructions. A sequence is also sometimes known as a routine. The more recent robot systems are capable of having multiple sequences of instructions which are linked logically to each other by a branching instruction. The WRAPS off-line program also has this capability of inter-nested sequences.

There are 2 types of format for the off-line programs used for WRAPS - (i) the absolute format and (ii) the relocatable format.

4.3.2.1 The Absolute Format Sequence

This is the complete WRAPS off-line program produced that will be post-processed to be used with the actual robot system. Fig(4-10) shows a typical WRAPS off-line program. The program consists of 3 sequences, each containing a series of robot 'MOVE' commands. Comment lines are added into the off-line program to improve readability. Comment lines begin and end with an asterisk (*). The off-line program is stored in standard ASCII text format and could be created with ordinary word processors and also edited if minor changes are required. This allows the robot program to be easily interpreted by the user.

The off-line program is divided into blocks of information. The first bit of information contains the name of the off-line program which
**FIG(4-10)** Example of a WRAPS Off-line Robot Program Comprising of One Main and Two Sub Sequences. Continued Next Page.
FIG (4-10) Cont’d.

FIG (4-11) An Example of a Relocatable Sequence.
is also the name of the program known to the robot controller. In WRAPS this name may be any character from the standard ASCII set. However, certain robot controller systems may not recognise certain characters and are usually defined in the robot system manuals. For example, the acceptable character set for the CML T3 robot is given in Appendix (A7). To overcome this problem, the Programming Module could read in a default character set from a storage file that is recognised by the robot controller of a particular robot system during software system loading and initialisation.

The next main block of information after the off-line program lists the number of sequences and the respective number of robot points in each sequence. The sequences are stored in the order in which they are created during off-line programming.

Following this, there are 3 categories of information: (a) Program Sequence Logic, (b) Sequence Process Parameters and (c) Program Data.

(a) Program Sequence Logic

The length of this section is dependent on the number of sequences and the number of robot points programmed in each sequence. Each sequence is divided into columns of information as Fig (4-10) shows.

The first column contains the robot point number which is designated as s.nnn where s is a sequence number and nnn is a robot point number within that sequence. The sequence number in WRAPS can currently range from 0 to 29 and the maximum number of robot points in each sequence is 999. These suffice for most of the batch fabrication applications. Each robot point being programmed into the off-line program is created dynamically in RAM and in practice its number is also limited by the capacity of the computer memory. See the description of dynamic allocation of the robot points in Sect 4-3 and in Chapter Three Sect 3-3-5.

In the second column is the robot function to be executed. All the functions that are supported in the current WRAPS software have already been discussed in Chapter Three Sect 3-3-7. Some examples are:
Chapter Four

BASE - start at origin of robot.
MOVE - move the TCP to another location.
PERFORM - executes a given sequence.
DELAY - hold the robot for a specified time.
END_PATH - closes the loop of a sequence.

The third column describes the state that qualifies or expands on a particular robot function, eg if it is DELAY, this column contains the number of seconds the hold is supposed to take or if it is a PERFORM the number of the required sequence to be executed. If a robot function does not require further qualifications, this column is left blank.

The next 6 columns contain the coordinates (X, Y and Z) of the robot position and the orientation angles (D, E and R) of the TCP of each robot point. In the current software, the coordinates are specified in mm and the orientation angles in degrees. The WRAPS software may be modified in future to accommodate choice of units used.

The above format is repeated for each additional sequence contained in the off-line program. Apart from the first sequence which must start with "O", the rest of the sequences may be labelled in any order so long as it is within its given range.

(b) Sequence Process Parameters

This section contains 6 process control parameters that may be used directly in conjunction with a particular robot function. The section is similar to the Sequence Logic section described above - it is variable in length, is divided into columns, and there is a block of information for every sequence as in the Logic section. The first column is identical and corresponds to the section above in their respective sequences.

The second and third columns contain the welding voltage and the wire feed speed respectively. These are defined in actual required standard units. Some robot systems, eg the CML T3 robot, require these
parameters to be expressed in a converted digital-to-analogue format. This conversion is achieved in the post-processor for the CML T3 and is described in Chapter Five Sect 5-3. Some other robot system, eg the FANUC "S" series robots do not require this conversion value in their robot program.

The fourth column contains the absolute velocity of the TCP expressed in mm/min. This is the expected velocity to be used by the robot but in most cases only an averaged value is obtainable in practice.

The fifth column contains the tool dimension. Most robot systems provide a range of tool dimensions to choose from or could be specified. Some robot systems, eg the CML T3 and the PUMA 562, allows dimensional offsets for the TCP and hence affect the tool dimension. In WRAPS, the tool dimension is the absolute value taken from the centre line of the roll axis without any offsets. At the moment, only one fixed tool dimension is used as was explained in Chapter Three Sect 3-3-7.

The sixth column contains a variable value that could be used for as a loop counter or in arithmetic evaluations. There are a total of 16 variables that could be used.

The last and seventh column contains the state of a flag that may be used within the off-line program. The flag function is was also discussed in Chapter Three Sect 3-3-7.

It is noted that even if the parameters are not used, they are still there. It could be used to show the change of state within that particular sequence in the off-line program. The values of the parameters are specified by a number which represents the element number of a standard table of information which is given in the next section.

(c) Program Data

This final section contains 6 blocks of data and is fixed in length. The first block contains 16 standard values for the welding
voltages that is available to the program. The next block contains 16 values for the wire-feed speeds and is followed by 16 values of robot velocities, 4 values of tool dimensions, 16 initial variable values, and lastly 16 initial status of flags.

The number of elements permitted in a particular robot varies from system to system. However, this is taken care of in the post processor, and only the required elements are used.

4.3.2.2 The Relocatable Sequence - Fixed Data Format

This is the format of a sequence which may be inserted into a program being generated. The sequence is linked up to the main program at the PERFORM command. For example, assume that a sequence labelled 4 contains the required task. If a main sequence has PERFORM call for sub-sequence 4, than the latter is called up and linked onto the calling PERFORM point.

The overall format remains unchanged as in the whole off-line program sequence. The layout of the logic is identical except that the program points are only labelled in integers of ascending order. The END_PATH point for the relocatable is not defined and is replaced by a series of question marks '?' . The software will ask for a required END_PATH point for the relocatable sequence to close loop to the current program. The robot system data tables, i.e. for velocities, tool dimensions, etc, are not used in the relocatable sequence as the values of the individual program points of such required data will take the value of those which are set in the main calling program. As Fig(4-11) will show, only the process parameters section forms part of the relocatable sequence to inform the system which elements within a data table is to be used.

4.3.2.3 The Relocatable Sequence - Macro

This is an extension of the idea based on the relocatable sequence but is considered as a special software function which contains
specialised robot related task, herein referred to as a macro. There are
2 major differences between a macro function and the fixed data
relocatable sequence mentioned above.

The first difference is that the coordinates for all the points in
the macro are not defined. These are defined only at programming time,
and its values depends on the given number of coordinates points, and
this in turn varies on the specific macro considered. For example, a
spot welding macro will generate 3 spot welds if the interval defined is
2, and if the interval is, say 5, the number of spot welds created
becomes 6. The other necessary intermediate points of the sequence, eg
the time delay instructions and the movements from one spot weld point
to another, are also generated by the macro.

The second difference is that each macro function is individually
coded and accessed as an overlay from the Programming Module. In
addition, the user is required to supply additional information in an
interactive manner as the macro is executed. The additional information
required varies, eg for the spot or tack welding, the user is asked
details such as:

- the start and end points of the spot welds.
- the number of intervals required.
- any offsets from the component/weldpts coordinates that
  has been chosen as the start and end spot welds.
- the END_PATH point for the macro.

Macros are also suitable for other specialised tasks such as
assembly, weaving and multi-pass welding and will be implemented in due
course.

4.4 The On-line Module

This module provides the communication functions between WRAPS and
other hardware equipment, basically the robot controller, cassette data
storage systems or a printer. The communication functions are discussed
further in Sect 4-4-2 below. A robot program stored on a floppy disk is
transferred to/from the robot controller direct by a RS232/Current-loop interface convertor link or indirectly via the cassette data tape recorder by a RS232 interface. The latter method has the advantage that a data tape for the robot system is also created in the process as a backup. Should the robot program in the robot controller be corrupted or erased, eg due to a power surge or other operating conditions, the program can be reloaded from the data tape using the cassette recorder.

The post-processor to convert a WRAPS off-line robot program to the target robot format (in this case, the CML T3 robot) is also part of this Module, but there is a standalone version. This post-processor is described in Sect 4-4-1 below.

Editing utilities are available in this module to assist in off-line program editing which goes down to the bit level. A hexadecimal listing of the robot program may also be made to the printer. The editing utilities are described in Chapter Five Sect 5-3-1.

Because this module carries the communication functions, the capability to record the measurements of a calibration object by automatic transmission from the robot controller is provided. The link-up to the robot controller is via the ADJUST specifications which is discussed in Sect 4-4-2-1 below. The calibration facilities are described in Sect 4-3-1 above.

This Module is intended to be the eventual intelligent supervisor of the overall flexible WRAPS controlled cell. Provisions are made for future on-line monitoring of the welding process to be used in conjunction with the Expert Module, which include communications with sensors. This expansion is discussed in detail in Chapter Six.

4.4.1 Post-processing

Before the off-line program is sent to the robot system, it has to be converted to the right form. This is performed through a software filter-and-reformat process which takes one program format and transform it to a required robot format.
4.4.1.1 Robot Program Formats of Different Robots

It must be clarified at this point that when a robot program is mentioned in this section, it refers to the native program structure that the robot controller system can understand. This is not another off-line program, eg from WRAPS, VAL or GRASP, that would still require post-processing before actual robot use.

The formats of a robot program differ among robot systems because each have more or less followed a different route of development. In addition to the CML T3, Robot programs structure of FANUC [164], YAMAHA [81], RTX [165], SANDS [83], PUMA [75], KUKA [166], KOMATSU [167], and ASEA [168] were investigated. The following observations were recorded:

a) The documents for each robot system were normally lengthy and extremely difficult to obtain from the manufacturers. On several occasions, the information supplied were erroneous and inadequate.

b) They differ in layout and meaning in the contents of arranged data, as well as from different manufacturers and even from different models of the same manufacturer.

c) Information within the robot program, i.e. robot system data and process related data, are arranged in groups of a specified fixed or variable length of computer bytes. Each byte of information may be sub-divided into bits of individual information for compactness, shown for example, for the FANUC S100 robot as in Fig(4-12) and Fig(4-13).

There is no intention to reproduce the various formats of the robot programs here, and the reader is referred to the respective manufacturers for further information.

Page 107
**Joint Coordinates Format**

<table>
<thead>
<tr>
<th>Byte</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>G72</td>
<td>F1 (HIGH)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F2 (LOW)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>AXIS 1 (θ)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>AXIS 2 (ω)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>AXIS 3 (ψ)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>AXIS 4 (γ)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>AXIS 5 (ϕ)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AXIS 6 (α)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>S CODE 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>S CODE 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>S CODE 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>S CODE 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**G98 Block—No Motion Only Service Codes**

<table>
<thead>
<tr>
<th>Byte</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Detail of F code**

F code is commanded in following two ways:

1) F1 digit

   In this case, only F2 (LOW) field is used.

   ![F (low)]

2) F4 digit

   In this case, both F1 (HIGH) and F2 (LOW) fields are used (11 bits in total).

   ![F (high)]

F4 digit is used with following G codes:

G01, G02, G03, G04, G05

**FIG(4-12) The Internal Program Structure of the FANUC S-series Robots [S100, S200, S300, S400].**

See FIG(4-13) for More Information.
FIG (4-13) Details of G Codes of FANUC S-series Robots
Showing the Bit Structure of Information Packed in Part of a Robot Instruction. See Also FIG (4-12).
4.4.1.1 Possible Universal Robot Interface Format

The adoption of a standard robot program format would considerably reduce interfacing efforts. Off-line programming systems offers several levels of standardisation possibilities [169]:

i) High level programming languages
   Possible only if the various robot manufacturers and software vendors of programming languages are willing to standardised. This is unlikely in the foreseeable future.

ii) Translated code
    Corresponds to the CILDATA interface used with Numerically-controlled (NC) machine tools.

iii) Robot specific code
    This uses an interface similar to the usage of paper tapes as in NC machines.

iv) Controller/Robot arm
    This is the lowest level allowing for possible exchange of controllers between robots, thereby, also transfer of programs across the robots.

There are presently 2 main working groups investigating the implications and requirements of the standardisation. One is a European tripartite effort [170,171] between the UK, France and Germany and the other the proposals by the special robotics interests group [172] of the Computer Aided Manufacturing-International (CAM-I). The "Industrial Robot Data" or IRDATA format [173] is the most advanced development to date, and it is based at level (ii) of the interface. An introduction of the IRDATA format is given in Appendix (A8).
4.4.1.1.2 The WRAPS Flexible Post-processor Concept

The general method to develop a robot format translator is to write a program for each make of robot, i.e. a "one-to-one" approach. It is unlikely that a single standard will be universally accepted, not at least in the foreseeable future. Even when it happens, some translation would still be required to make existing robot programming systems fit into the new standard whenever it arrives.

This robot program translator or post-processor as it is commonly known, accepts as input a program produced from another software, eg from WRAPS, and produces as output a version of that program which can be directly executed by the robot system. It is expected that the final program when executed, will have the same effect as dictated by the logic and data contained in the original program. This translation process will include verification of the syntax, rules and format of the source and target programs. A "one-to-one" approach will require that whenever a new translation is needed, a new program has to be written, normally demanding an experienced programmer-engineer to do the work.

As a parallel development to the core of the WRAPS project, the author has supervised two Master of Science degree projects to develop a flexible post-processor. Its concept is equally applicable as a flexible pre-processor. Some initial investigations and development work based on guidelines given were in accordance with the overall integrating concept of the WRAPS project.

In the first project, Rae [19] investigated and re-affirmed the need for a flexible post-processor for an off-line programming system and commenced basic development work, culminating in a translation of a simple VAL to CML T3 language type of format. In the second project, Law [20] extended the work from Rae and developed some software, based on the author's concept, which forms the basis of such a universal post-processor. He tested out the post-processing from an old version of a WRAPS single-sequence off-line program to a high-level descriptive form of the CML T3 robot program format. The latter is not a working program of the CML T3.
Chapter Four

The author's concept is an extension of the techniques adopted in lexical analysis or variations of it and of the many language compilers and program translators [174,175]. However, the flexible post-processor application requires a different approach.

One of the major criteria of the WRAPS concept is that the change of one post-processing task of an off-line/robot controller system to another, will only require an end-user to make the necessary modifications to the input data for the post-processing software to produce a required format. This concept dictates a common fixed software tool but different information being supplied to it. The concept post-processor tool is comprised of 2 main modules: set-up and program generator. Because the concept works with off-line programs and robot systems at about the interface levels (ii) and (iii) as described in the previous section, the universal post-processor would also fit in the arena where robot users want to translate their existing programs to the new standard when (and if) the time arrives.

To illustrate the concept, consider a translation of an old version of WRAPS program to another (any) robot target program, shown in Fig(4-14). It is apparent that the format may be divided into recognizable blocks of information. In the WRAPS concept, the smallest block is a single character.

Initially, the user uses the set-up module and interactively enters the required format of a target program. Looking from the user's point of view, it is like telling the post-processor what is required as the outcome, with the rest of the work performed or controlled by the post-processor. The format is to be defined in a block-by-block (eg CML T3, ROPS, KAREL) or a top-down line-by-line (eg KUKA, VAL) fashion. This will cover all, if not most, known robot program formats.

The set-up module will ask the user to define the physical layout, logics, syntactics and semantics of the output target. During the interactive session, the set-up module will ask if, eg the output format is composed of blocks of information or individual lines. Whether it is arranged in blocks or lines, the set-up module will have to ask until
*MY FILE NAME: wrap.dat*

**ROBOT PROGRAM**

<table>
<thead>
<tr>
<th>PROG NO</th>
<th>NO OF POINTS</th>
<th>Function</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0</td>
<td>Home</td>
<td>-</td>
</tr>
<tr>
<td>0.001</td>
<td>0</td>
<td>Tool 1</td>
<td>ON</td>
</tr>
<tr>
<td>0.002</td>
<td>0</td>
<td>No Op</td>
<td>-</td>
</tr>
<tr>
<td>0.003</td>
<td>0</td>
<td>Sel DAC1</td>
<td>1</td>
</tr>
<tr>
<td>0.004</td>
<td>0</td>
<td>Sel DAC2</td>
<td>2</td>
</tr>
<tr>
<td>0.005</td>
<td>0</td>
<td>Cyc Start</td>
<td>-</td>
</tr>
<tr>
<td>0.006</td>
<td>0</td>
<td>No Op</td>
<td>-</td>
</tr>
<tr>
<td>0.007</td>
<td>0</td>
<td>No Op</td>
<td>-</td>
</tr>
<tr>
<td>0.008</td>
<td>0</td>
<td>End Path</td>
<td>0.005</td>
</tr>
</tbody>
</table>

**COORDINATE DATA**

<table>
<thead>
<tr>
<th>PROG NO</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>D</th>
<th>E</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.001</td>
<td>925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.002</td>
<td>12925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>90.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.003</td>
<td>12925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.004</td>
<td>12925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>90.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.005</td>
<td>12925.0</td>
<td>0.0</td>
<td>860.0</td>
<td>90.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.006</td>
<td>61645.0</td>
<td>5575.0</td>
<td>4000.0</td>
<td>90.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.007</td>
<td>29645.0</td>
<td>2425.0</td>
<td>4000.0</td>
<td>90.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**PROCESS PARAMETERS**

<table>
<thead>
<tr>
<th>PROG NO</th>
<th>DAC1</th>
<th>DAC2</th>
<th>Velocity</th>
<th>Tool Dim</th>
<th>Variable</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.002</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.003</td>
<td>1</td>
<td>2</td>
<td>14</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.004</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.005</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.006</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0.007</td>
<td>1</td>
<td>2</td>
<td>14</td>
<td>14</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**DAC1**

| 40 | 111 | 0 | 0 |
| 0  | 0   | 0 | 0 |
| 0  | 0   | 0 | 0 |

**DAC2**

| 110 | 0 | 222 | 0 |
| 0   | 0 | 0   | 0 |
| 0   | 0 | 0   | 0 |

**VELOCITIES**

| 102 | 140 | 250 | 420 | 600 |
| 800 | 9000 | 1000 | 3000 | 10000 | 12000 |
| 14000 | 16000 | 18000 | 20000 |

**TOOL DIMENSIONS**

| 102 | 500 | 614 |

**VARIABLES**

| 0   | 0   | 0   |
| 0   | 0   | 0   |
| 0   | 0   | 0   |

**FLAGS**

| 0   | 0   | 0   |
| 0   | 0   | 0   |

FIG (4-14) A Very First Version of a WRAPS Off-line Program Used in the Robot Program Translation Based on the Flexible Post-processing Concept. See Law's Work [20].
every character space of significance within the program has been mapped. Some spaces within a robot program contains zeroes as padding to maintain a fixed length of data information and are not used by the appropriate robot controllers. It will have to find out where robot command names and process-related data are stored. Frequently, for the sake of compactness, it may be a single bit within a byte (a character) of the robot program. In addition, the special robot function names (eg MOVE, GRIP etc) and data identification parameters (eg welding voltages, Wire feed Speeds etc) will be required so that the post-processor will have a dictionary reserved for the particular robot system.

The author has not come across a robot program that contain comments, but only in the various off-line program formats. So the set-up program will ask for a character (eg it may be an "*"), or a series of characters (eg "*" or "REM") which are used to denote the opening of a comment. In due course, all comments which are recognised as comments are eliminated. Specific robot messages are however, treated as parameters to a robot controller function which is related to echoing a string of characters to the controller screen.

In the next phase, the set-up module will read in the source program and a lexical analysis is performed so that the input is separated into characters and/or words and given distinctive tokens. Elimination of un-wanted comments takes place. Then the post-processor links the tokens logically into syntactic structures which are trees whose leaves are tokens. For example, if a token represents a robot point number, then there must be a robot command associated with it, whether it is an active robot function, eg to move to a desired location, or a passive function that carries only process-related information. In addition, each point would normally have defined coordinates and orientation of the TCP for a move command or intentionally zero for a no-move command. Also, similarly if a token represents "Output", then this token will be linked to a sub-token, for example, represents the channel which it controls, and then another lower level token to represent the possible states of the token, eg ON or OFF. The post-processor then translate this information into a reference table containing the logic, syntactic and semantic information, say call this a LSS table. An output of such a LSS table,
taken from Law's work is shown in Fig(4-15). Then, the setup module creates formatted-code in the intermediate stage, based on a cross-referencing between the LSS table and the dictionary database. An output of the intermediate code, also from Law's work, is shown in Fig(4-16). For completeness, a description of the meaning of the LSS and the intermediate code is given in Appendix(A9).

The next phase is similar to the definition input for the target format. Based on the information given in the target format, the set-up module will ask for certain information, eg if there is a "grip" option in the target, than the set-up module will ask if there is an identical or replaceable command that needs to be taken into consideration of the user's input program. In this way, the post-processor can also check for unknown robot functions. Similarly as for the target program, the set-up module asks for the entry of the logics, syntactics and semantics of the source program if necessary. The format is also to be defined in a block-by-block (eg WRAPS, ROPS) or a language-type (eg VAL, GRASP) fashion. From this information, the set-up module creates another LSS table.

The program generator module, the second module of the concept post-processor, is then used to interpret the intermediate code in conjunction with the target LSS table to automatically produce the robot program that is required.

The author's approach for the flexible post-processor is demonstrated to a certain extent by Law and the reader should refer to his contribution for further details. Considerable development work need to be carried out before a universal post-processor becomes viable. The author is dedicated to oversee future work on this topic as it is one of the tools that should be made available within the whole WRAPS integration concept.

4.4.1.2 Robot Program Format of the CML T3 Robot

The storage form of the robot program of the T3 is called a (cassette) data tape, but herein it is referred to as the robot program
KY FILENAME IS: offlog.dat

0 0 2 1
& *
0 1 1 2
& *
0 2 1 3
Home No_op Tool_1 Sel_DAC1 Sel_DAC2 Cyc_Start End_Path *
0 3 1 4
ON OFF *
0 2 1
& *
1 1 5
& *
1 2 1 6
& *
1 3 1 7
& *
1 4 1 8
& *
1 5 1 9
& *
1 6 1 10
& *
2 0 2 1
& *
2 1 1 12
& *
2 2 1 13
& *
2 3 1 11
& *
2 4 1 14
& *
2 5 1 15
& *
2 6 1 16
& *
3 0 2 21
& *

FIG (4-15) The Corresponding LSS Table Data File for the WRAPS Off-line Program of FIG (4-14). Explanation to LSS Table is Found in Appendix (A9). For More Details, Refer [20].

Cont'd Next Page.
FIG (4-15) Cont'd. in (a) of the LSS Table.
(b) Shows the Dictionary Codes Which are Generated in Conjunction with the LSS Table.
Intermediate code data file

MY FILE NAME : int.dat

<table>
<thead>
<tr>
<th>Value</th>
<th>0.000</th>
<th>0.001</th>
<th>0.002</th>
<th>0.003</th>
<th>0.004</th>
<th>0.005</th>
<th>0.006</th>
<th>0.007</th>
<th>0.008</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000</td>
<td>0.100</td>
<td>0.200</td>
<td>0.300</td>
<td>0.400</td>
<td>0.500</td>
<td>0.600</td>
<td>0.700</td>
<td>0.800</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>1.100</td>
<td>1.200</td>
<td>1.300</td>
<td>1.400</td>
<td>1.500</td>
<td>1.600</td>
<td>1.700</td>
<td>1.800</td>
</tr>
<tr>
<td></td>
<td>2.000</td>
<td>2.100</td>
<td>2.200</td>
<td>2.300</td>
<td>2.400</td>
<td>2.500</td>
<td>2.600</td>
<td>2.700</td>
<td>2.800</td>
</tr>
</tbody>
</table>

FIG(4-16) The Neutral-Format Intermediate Code Output
From the Prototype Flexible Post-Processor. For Details Refer [20].
since the off-line program may be stored also on floppy disk or hard disk media. The version of the robot program described here is the Restructured Software version 3 [176]. It is worth mentioning that the reference manual available is inadequately described and often erroneous. There was no resident expert at CML on the format of the robot program, and therefore, much of the work here has been based on a trial and error basis before arriving at the proper correct format of the robot program.

The robot program is divided into 3 main sections: a leader, a data section, and a trailer. The leader and trailer are defined as both 1000 bytes in length, each byte being a 80hex. In practice however, the trailer was found to be redundant.

The data section is initially arranged in a logical structure which consists of 2 main portions. The first portion, the data table section, is comprised of 9 fixed-length tables. Fig(4-17) shows the standard 9 tables used. The tables are present on every robot program, regardless of the option contents of the robot software system, and they always occur in the same order. This permits maximum compatibility of robot programs which may be used on robot systems having different software options. The contents of the individual data tables are described further below.

The second portion contains the total data points which are of a fixed-length fields of 16 bytes each. In general, each data point contains the tool dimension, velocity, robot function and coordinates of a taught point in the robot program. The length of this portion varies with the number of points taught and recorded with the robot program.

When the robot program is being made, the logical data section is then divided into fixed-length records of 46 bytes long and super-imposed with additional bytes of information. Only the last record may be less than 46 bytes long.

Each record starts with 3 bytes of FF (hex) followed by 2 bytes describing the length of the data field in this record. This length is usually 38 bytes, except for the last record. Next comes 2 bytes which
### TABLE NAME

<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>SIZE IN BYTES (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Identification</td>
<td>30</td>
</tr>
<tr>
<td>Function Existence</td>
<td>37</td>
</tr>
<tr>
<td>Tool Dimension</td>
<td>13</td>
</tr>
<tr>
<td>Velocity</td>
<td>21</td>
</tr>
<tr>
<td>Sequence Allocation</td>
<td>133</td>
</tr>
<tr>
<td>Variables</td>
<td>21</td>
</tr>
<tr>
<td>Flags</td>
<td>21</td>
</tr>
<tr>
<td>DAC</td>
<td>113</td>
</tr>
<tr>
<td>Weave</td>
<td>117</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>506</strong></td>
</tr>
</tbody>
</table>


### FIELD

<table>
<thead>
<tr>
<th>FIELD</th>
<th>SIZE IN BYTES (Decimal)</th>
<th>CONTENTS (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leader</td>
<td>1000</td>
<td>A1 80</td>
</tr>
<tr>
<td>Data Record :</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start code</td>
<td>3</td>
<td>FF FF FF FF</td>
</tr>
<tr>
<td>Record Length</td>
<td>2</td>
<td>2A</td>
</tr>
<tr>
<td>Load Address</td>
<td>2</td>
<td>7C04</td>
</tr>
<tr>
<td>Data</td>
<td>38</td>
<td>A11 00</td>
</tr>
<tr>
<td>Checksum</td>
<td>1</td>
<td>A6</td>
</tr>
<tr>
<td>Trailer</td>
<td>1000</td>
<td>A1 80</td>
</tr>
</tbody>
</table>

**FIG (4-18)** A Typical Format of a Data Record within the Second Portion of the Logical Structure of a Cincinnati Milacron T3 Robot Program Version 3.
contains the memory load address for this record at the time it was created in the robot controller memory. In actual practice, this memory address may be any value randomly assigned by the robot controller. However, on the WRAPS off-line robot program, this address is based on a logical incremental in the order the particular robot point was created off-line. There is no ill-effect induced in the program. The 2 bytes of memory address is followed immediately by the data field itself, finally followed by a checksum byte. In a similar format, each record is followed immediately by another record until the trailer is reached. Fig(4-18) shows a typical format of a data record which was created from memory address 7C04 (hex), and contains data of all zeroes.

The 1-byte checksum value is calculated following the procedure described in the manual for the T3 robot program.

4.4.1.3 The WRAPS Post-processor for the CML T3 Robot

The lack of a universal post-processor at this stage requires a dedicated post-processor to be developed for the CML T3 robot in order that current development on the WRAPS off-line programming capability could be tested. A dedicated post-processor takes considerably less time to produce.

As described in Sect 4-4-1-1, the data portion of the CML T3 robot program is arranged in a block by block fashion, and it is appropriate that the overall off-line robot program be constructed in a similar format. The WRAPS off-line robot program has already been described in Chapter Three Sect 3-5.

Analysis of the structure of the T3 robot program suggests that the robot program could be constructed by piecing separate items of data together. Each item of data may again be pieced together from individual bits or bytes of computer stored data. Following this approach, the following elementary software functions were identified and required to enable the building of the off-line T3 format robot program:

a) Convert a character value (1 byte) to a
b) Convert an integer value (2 bytes) to a hexadecimal format value.

c) Write a byte of data to a disk file (for storage).

d) Write an integer (2 bytes) data to a disk file.

e) An updating function to tally every 16 bytes of data.

f) A conversion function to change T3 robot units to user units (inches or mm).

Different computers have differing primary data type length, eg a normal integer (i.e. not short or long) on an IBM AT computer is 16 bits (2 bytes) while the length is 36 bits on a Honeywell 6000 computer. To achieve portability, the "C" sizeof() function is used initially to determine the size of the data type.

The post-processor is an overlay function which is part of the On-line Module. Its main task is to translate the format of the WRAPS off-line program into the internal format, so called the robot program that is transferred into the memory of the robot controller.

Robot function names in the WRAPS neutral format are translated to the specific equivalents of the T3 robot. Any un-recognised function names are replaced with "****" and a error message is reported to the user.

In a robotic arc welding system the welding equipment is controlled from the robot controller computer. Some robot systems, eg the FANUC series, accepts direct values of welding arc voltage and wire feed speed when programming the robot, but others use a conversion value which is related to the actual output of the welding equipment. The CML T3 robot output values to the welding equipment, which is an analog device, through a digital to analog convertor board (DAC).

The DAC board used by the CML T3 is a multi-channel digital-to-analog interface device. It provides 12 bits of analog resolution, yielding 4096 discrete possibilities between -9,000 to +9,000 V. Most analog equipment operates 0-10 V. The CML T3 DAC will
scale the 0-9 V signal from 0 - x units as defined in the robot system definition. One of the functions required during system definition [177] is to determine the maximum DAC value that could be used. For example, if the maximum wire feed speed specified on the wire feeder is 785 inch/min, then the maximum DAC value is:

\[
\begin{align*}
&= 0.9 \text{ (Rated Maximum Output)} \\
&= 0.9 \times 785 \\
&= 706.5
\end{align*}
\]

This maximum DAC value remains constant for the equipment in use. Any other required DAC values must be less than or equal to the maximum value.

The CML T3 system DAC function allows the specification of a DAC SCHEDULE where any 2 DACs have already been assigned values. In all 16 DAC SCHEDULES are allowed in the version used. The 2 DACs will output their assigned values for that schedule at the point the function is taught. The CML T3 robot system supports 2 DACs, 1 for the welding voltage and the other for the wire feed speed. These are referred to as DAC1 and DAC2 respectively.

Once the off-line robot program has been completely generated from the Programming Module, it is ready for post-processing by the On-line Module. The On-line Module may be called up in 2 ways in order to get to the post-processing facility. The first method is by "chaining" from the Programming Module, and the second method is to load On-line Module separately.

Chaining is a process of passing control from an executing "C" program to another "C" program [179]. This is achieved by the currently executing program through a function call to execfl(). Any number of "C" programs could be chained to but once control is passed on to the new program, it is impossible to return to the original program because it would have been overlayed by the new program. For WRAPS, this could mean proceeding directly from the Programming Module to the On-line Module. Before chaining is affected, the user is asked to confirm again in case of accidental erasure of incomplete work within the Programming Module.
Separate loading of the On-line Module would be ideal when there is a break in the use of the Programming and the On-line Module. Attempting to run the On-line Module from within the Programming without chaining would be impossible because of current memory limitations. Using new techniques to make use of expanded memory, i.e. memory beyond 640 Kbytes is possible and this is discussed in Chapter Seven for future development work.

In operation, the post-processing is straightforward. The user is asked for the filename, always of type *.OFF, of the WRAPS program to be processed. If the file specified is not found, the software request if another attempt is required otherwise the software is exited.

There are 2 stages involved in the post-processing. The first stage generates the logical data portion as described in Sect 4-4-1-2, and is stored in a file of type *.TMP. The second stage generates the actual off-line robot program and is stored in a file of type *.CT3, in this case for the CML T3 robot. Other robot systems are expected to have a different file type.

4.4.2 WRAPS Communications

The interface between the WRAPS host computer and other computers or robot systems, is usually through the serial RS232-C [38]. This was found to be a standard or additional option offered by all the robot systems investigated: CML T3, FANUC, KUKA, PROMETHEUS, SILVER-REED, RTX, PUMA, YAMAHA, and TOMKAT. In some cases, e.g. printing of some data from a file may be redirected to the parallel port of a computer but this is usually a function selectable via the host operating system. Hence it is not a facility required from WRAPS.

The communications facility available within WRAPS may be divided into 2 groups: simple communications and communications via a protocol.
(a) Simple Communications

The simple communications capability is to enable WRAPS to transmit and receive information with other computer systems or peripheral devices. The data being sent or to be received is controlled manually by the user. This capability consists of receiving or transmitting a single byte of information, e.g. for a control purpose, or the transmission/receipt of a complete file, e.g. an off-line program.

The software was written in a highly modular form but grouped under a single utility called SENDRECV. It is able to transmit/receive data ranging from a single byte to a complete file of any size stored on disk. Also, the data transmitted may be in 7 (true ASCII) or 8 (binary) data bits byte. SENDRECV has been arranged in 2 versions; it may be called up as an overlay via the On-line Module or used as a standalone module.

(b) Communications by Protocols

This method of communication enables the automatic exchange of data between 2 systems via a defined communication protocol. This protocol data exchange is necessary to achieve correct sequencing and data integrity and management of the physical channel, in an environment, e.g. to send or receive real-time robot data from a robot to an external computer. There is no universally accepted standard of communication protocol and for the WRAPS project, the DDCMP protocol [179] is used when required. This is a standard adopted by Digital Equipment Corp (USA) for intercomputer data communications. This protocol was selected because it is supported by several major robot manufacturers, for example CML, FANUC and UNIMATION, and is more widely used. Another protocol used is the ADLP-10 protocol [180] which is used in the ASEA robot systems. Most companies used their proprietary protocols, for example the RTX uses their IPC protocol [165] whilst SILVER-REED [181], SANDS [93] and FANUC [192] generally refer to theirs as communication protocols with no defined names.

All protocols of communication investigated conform to a basic
operating principle as described in Fig(4-19). The robot or computer whichever requires data to be sent to the other transmits a code to call the other. Transmission starts when the caller receives an acknowledgement indicating that the other could accept data.

Data transferred between a robot and external computer system is usually multi-byte strings of binary digits. These strings, which differ in length depending on the robot system, are given meaning to perform various functions by the software according to the robot-computer interface convention which is adopted by a manufacturer. These interpreting conventions vary from manufacturer to manufacturer. Because the basic software for communications are modular, they are capable of being linked up in the right order to provide data strings of the required protocol format and length. The messaging protocol currently written is only suitable for the CML T3 robot, but the flexibility to format the protocol is the topic of a separate research work [19,20] supervised by the author. Other communication protocols, eg MAP/TOP are discussed in Chapter Six.

4.4.2.1 Communication with CML T3 Robot

With the T3 hydraulic robot systems, there are 2 ways to send off-line data to the robot controller and 2 ways to communicate with the robot in real time.

(a) Off-line Communication

The first method to send off-line data uses a direct link between the WRAPS host computer and the cassette data tape port of the robot controller. The cassette port is a current loop interface and hence a RS232-to-Current-Loop convertor unit is required. The convertor unit is a through-link device [183] and sits between the RS232 port of the WRAPS computer and the Current loop cassette port.

The second method to send off-line data is via the STR-II cassette recorder unit, i.e. from WRAPS computer to cassette recorder using RS232, and between cassette recorder and robot controller cassette port.
Ro bot

REO. Send

Data

Computer

ACK

Send Data

ACK

Legend: REQ - Request  ACK - Acknowledge

FIG(4-19) The Request–Acknowledge Data Communication Protocol.

<table>
<thead>
<tr>
<th>Type of Parent Material to be Welded</th>
<th>Name of File Holding Suitable Welding Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET1.BST</td>
</tr>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET2.BST</td>
</tr>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET3.BST</td>
</tr>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET4.BST</td>
</tr>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET5.BST</td>
</tr>
<tr>
<td>Low &amp; Medium Alloy Steels (Example)</td>
<td>FILLET6.BST</td>
</tr>
</tbody>
</table>

FIG(4-20) An Example of the Historical Index File Linking Weld Conditions to Available Weld Procedures.
using Current Loop. The recorder has both RS232 and Current Loop interfaces.

(b) Real-time Communication

Communications with robot systems in real-time are usually carried out to transfer blocks of information, eg some robot axes data, status of controlling input/output registers monitoring some peripheral equipment, or readings from a sensor attached to the robot hand. The transfer of whole off-line robot programs are usually not possible with current robot systems, hence limiting the transmission of a program only when the robot is not executing a program already in memory. To quote an example, consider the FANUC robot system. If an external computer requests for on-line program transfer while the robot is executing a program, the robot responds to the external computer with an error number indicating that it is incapable of servicing that request.

On the CML T3 robot system used in this research, 2 means of interaction with the robot in real-time are possible by using the EXTERNAL and ADJUST function. The reader is referred to other functions available, for example REMOTE [184], on newer robot CML robot systems.

(a) EXTERNAL Function

The EXTERNAL function [185] permits an external device, such as an external computer, to transmit/receive entire sequences of information to/from the robot controller of the CML series of robots. Modification to the positional coordinates (X, Y, Z), orientation (D, E, R), and function (eg Delay, Wait) of each robot data point of a stored sequence is possible. This method of communication with the robot controller had previously been demonstrated, for example, by Queen's University (Belfast) in conjunction with Short Brothers Ltd [186].

Its successful operation involves the initial establishment of a communication link. Once a link is made between the robot controller and the external computer, it will be maintained until some error or malfunction on either side causes it to break. The communication is
performed under a subset of the DDCMP protocol.

The EXTERNAL function allows near real-time modification to a robot coordinate and function. The time taken to execute an EXTERNAL function may be estimated using a procedure suggested in the manual. As an example, to receive a 20 point sequence, it takes 353 msec or approximately \( \frac{1}{3} \) of a sec. The external device receives the sequence data from the robot when the EXTERNAL function is executed in the teach mode. The external device stores this data and may modify it at any time. In the auto mode when the EXTERNAL function is executed, the robot requests and receives the associated sequence data from the external device which replaces the existing sequence data residing in memory. This new sequence data is then executed.

This function is not ideally suited for the transfer of off-line programs for the following reasons:

(i) The main line sequence cannot be transmitted or modified. As a consequence, a dummy robot program, eg containing non-move commands and having the required number of robot points, is required in the robot controller memory to serve as a template.

(ii) Data for a particular sequence cannot be transmitted or received while the robot is within that sequence.

(iii) The number of data points within the sequence must remain constant. If there is a need to add or delete a point within the sequence, a new and longer dummy robot program containing the required number of points must be created and used.

(b) ADJUST Function

This CML T3 function extends better real-time control capability
for the robot to adjust its position based on information not directly available to the robot. The communication basis is the DDCMP protocol which is identical to the EXTERNAL function. The robot controller communicates with an external processor which has direct access to positioning information, e.g. weld seam position and orientation data from a sensor mounted at the wrist of a robot. The external processor evaluates and updates the coordinates for all 6 axes of the current position which are sent from the robot controller at the instant when the ADJUST function is executed. Similar to the EXTERNAL function, ADJUST requires a link to be made and will remain until some error causes it to break from either side of the link.

The ADJUST function allows only single point modification and has several other restrictions associated with it [187]. As an example of its transmission speed, it will take just under 1/10 sec at 9600 baud to complete a robot coordinate update from the external device.

The calibration of the modelled environment to the real world as described in Sect 4-3-1 is implemented using the ADJUST function because instant positional data of the robot could be received. This uses the the AMI RS232 port at the rear of the robot controller.

4.5 The Expert Module

This module provides the welding procedure, if available, for a particular application which are held in a database. The Programming Module accessed this database during off-line robot program generation. As a standalone module, the weld procedures managed may be interrogated and inspected. The use of KEB as the expert interface, enables the knowledge which is obtained to be formalised and implemented as rules.

This module provides weld procedure selection, but has provisions for extension to:

(a) Closed loop optimisation of procedures based on their measured performance in manufacturing operations.

Page 130
(b) Prediction of procedures where none previously exist.
(c) Real time process control.
(d) Adaptive control modelling.

4.5.1 Weld Procedure Search

A batch fabricator can expect to have a wide variety of components to be welded and each component itself may contain varying types of welds. In order to facilitate the selection of a suitable welding procedure, WRAPS performs a preliminary scan of the data banks by categories, to inspect for availability. This permits rapid retrieval of weld procedures if available, and is immediately embedded into the off-line robot program during generation. For this reason, the search routine is now embedded in the Programming Module Sect 4-3.

A variety of search techniques are available, and the technique(s) that can be adopted varies with the type of database design. Examples of database file design are sequential organisation, random organisation and list organisation. Examples of search techniques are sequential retrieval, binary search, direct-access, dictionary lookup, hashing, Indexed Sequential Access method (ISAM), Virtual Storage Access Method (VSAM) etc. Further information about search mechanisms and guidance for the selection of record addressing methods can be found in [188-191].

The search procedure designed for WRAPS is considered efficient. It incorporates the hashing search technique and a variation of the list-type organisation of the database information. The hashing technique is more suitable for the retrieval of a single record [188], such as a weld procedure, which could be arranged in a list organisation. List organisation of the database allows records to be placed anywhere within a file. This means that information to be updated can be appended to the end of the existing file, without any order sorting. This simplifies the interface to other software products, and this is important in consideration that for the WRAPS application, a weld procedure to be retrieved is expected to conform to a certain standard format of storage, eg according to BS4780. In this respect, it
is imperative that the embedded search facility (from within the Programming Module) would eventually be capable of extracting weld procedures which are stored, eg using the WELDSPECS software program (discussed further in Chapter Seven Sect 7-2) from the Welding Institute [192].

In a welding database search, to match a search parameter to a key in a record, it is expected that character strings comparison will be common occurrences. This is because there will be descriptions, eg relating to the type of welding process, type of parent material, or type of wire material. When stored in a computer memory, the descriptions are usually represented by a series of corresponding characters, commonly known as a character string. A terminating character, eg a '\0' in the "C" language, is used to denote the end of a string. Character strings, if they are lengthy, can occupy considerable amount of computer memory, and the outcome of a comparison (eg whether two strings are equal or not) usually takes a longer time than when compared, for instance with a number comparison. To improve on this, hash indices which uniquely identify a particular character string, are used. A comparison of two integer numbers is considerably quicker.

The hashing technique [193,194] involves taking each word of a string and add the values of the ASCII codes of the letters together. This produces a number called a hash index. This technique has a problem associated with collision where 2 different words hash to the same index. This however could be avoided by a technique called "probing". When a collision occurs, a new index table is created based on the original index table previously created. This new indices should be unique from each other otherwise the "probing" continues. As an example, Fig(4-20) shows the hash values for the list of parent material types which were available for consideration within the WRAPS programming module. The indices were created with an independent software program called 'MAKEHASH'. The hash indices for the procedure search in the Programming Module are read into the system from a file, hence any collision would have been eliminated prior to usage.

The prototype search routine for WRAPS, which is simple to implement and maintain, is as follows. Before a weld procedure is
selected for a particular weld, there are 7 parameters which must be provided:

(i) type of welding process (eg GMAW, TIG)
(ii) type of joint (eg filler, butt, lap)
(iii) welding position (eg horizontal, vertical)
(iv) parent material type
(v) parent material thickness
(vi) size of wire used
(vii) bead size required

The weld database is then scanned through to find a weld procedure that satisfy, or is close to, the above criteria. The search is performed first on a file that hold the history of the database. Any available weld procedure is entered into this file which is labelled SUITABLE.DBS. Fig(4-21) shows an example. The scan is carried out categorically according to the order of parameters entered. Initially it searched for the process, eg for GMAW. If GMAW is available it searches next for all files under GMAW for the type of joint. If the type of joint is available, it proceeds with the welding position category and so on. When a search fails at a category, the user is alerted to confirm if search is to proceed, hoping eventually to come up with the next best procedure. The next best procedure may be selected from a few relaxed parameters, eg for parent material thickness or bead size requirements but some parameters obviously may be unchangeable, eg the welding process. Fig(4-22) shows the hierarchy of search through the database and the information files accessed during the search.

In a simulated database containing over 1000 procedures, the software was made to search through, in the worst case situation, till the last item of every record and until the 1000th record is reached, therefore containing a total of 9001 searches, it took 25 +/- 1 secs to locate the required procedure. This represents an average of 0.025 sec per procedure search, and there is still scope in optimising the search software. The number 9001 is obtained by the following formula:

\[(\text{Number of items per record} - 1) \times \text{Number of Records} + 1]\
FIG(4-22) The Weld Procedure Database Search.
Chapter Four

In the prototype database, there are 10 items per record as was shown in Fig(4-21).

4.5.2 Extension to Generation of Welding Parameters

The automatic generation of welding procedures has been a research topic for many years. Various approaches have been analysed and reported [195]; without exception, the researchers employ formulae either experimentally or theoretically derived to relate the process variables. Whilst the experimental approach permits rapid generation of data and associated equations, it is constrained by its heavy commitment to experimentation. Indeed, the data for single sets of conditions cannot be generalised and the control algorithms generated were restricted to a narrow application area [196,197], using analysis which are based on certain conditions, for example, heat conduction analysis which is suitable for applications involving material with good heat conducting properties such as aluminium.

Conversely, the theoretical approach minimises experimentation and tends towards a generalised treatment of the problem. Such predictive methods have been published [198,199] and form the basis of BS 5135 (1974) for welding carbon and carbon manganese steels. These methods only provide general guidelines for a limited set of conditions. Considerable professional expertise is required to interpret and modify these guidelines appropriately. Moreover, recent findings [200,201] have raised serious concern about their intrinsic validity. Since this process planning function involves high levels of judgement and experience and in view of the need to justify a chosen procedure to the customer, an expert system solution shows promise. Furthermore, the penalty of high repair costs and time constraints discourage the welding engineer from investigating alternative procedures which may be equally acceptable but perhaps more suited to the company's production capability. The use of an expert system could alleviate this problem.
4.5.3 The Expert Knowledge Interface

Having decided on using an expert system shell as put forward in Chapter Two Sect 2.5, there was then the issue of deciding on the shell to be used. In early 1987, the author carried out a survey and there were over 20 expert systems shells available for use on microcomputers, compared to only 6 in the 1984/1985 period when the WRAPS research was started. At the time, most of the shells were at a very basic standard, for example, most were not capable of floating point arithmetic and most importantly, offered no external interfacing. The shell selected was KES [108] which stands for Knowledge Engineering System. The main contributing factors in selecting this shell are:

1) Interface capability of KES, both with WRAPS and other intended software additions in the future. WRAPS and KES are written in the language "C" and detailed interface information to embed KES into other software packages or to embed other "C" software into KES are provided. In the research into the welding process planning and the acquisition of welding knowledge, there is a need for the interfacing of expert system to external languages and databases, and particularly to real-time data acquisition systems.

2) Most expert shells in the same price bracket provide only the production-rule logic as their inference engine. This inference engine controls the use of the knowledge in the knowledge base, functioning the way an expert does when solving problems and making decisions. It is envisaged that in certain applications, the rules of welding are not fully known.

The Production-rule logic is not the most suitable in these cases. An alternative logic system is based on fuzzy theory [202-204]. Fuzzy logic is an extended form of boolean logic which allows logic
to be defined in probabilistic terms by using real rather than integer numbers. It allows for various degrees of truth as opposed to only true or false. There are various other forms of knowledge representation, each of which on its own or when used in combination provide the ideal situation, eg Triples, Frames, and Bayesian [202,205].

KES provides multiple inference engines because no single approach is well-suited to all expert system applications. KES provides:

(a) Production Rule - a modular knowledge structure representing a single chunk of knowledge, usually in "if..then" or Antecedent-consequent form.

(b) Hypothesize and Test - provides reasoning through hypothesis formulation and subsequent verification. This technique reflects abductive reasoning. It is suitable for minimal set covering, i.e. the inference engine determines the smallest number of causes, represented by frame-like descriptions in the knowledge base, that explain all known manifestations of the problem of interest.

(c) Statistical Reasoning (BAYES) - performs statistical pattern classification based on Bayes' theorem. This theorem relates the probability of a hypothesis being true before receipt of extra information, to the probability of the hypothesis being true after that information has been received.

The basic criteria in selecting the inference engine lies in the way the knowledge is represented and the way the information is
The appropriate environments in which each of the above mentioned inference engines are suitable for use are recommended in the KES Knowledge Base Author's manual [206] Chapter 5 "Designing a KES Expert System". In the welding process, incorporating process planning and process control, the available inputs and the characteristics of the desired outcomes favour the possible use of all 3 logic systems on different occasions. This is imminent because not all the knowledge could be or best implemented in an "if...then" approach.

4.5.3.1 Interface of Expert Shell to WRAPS

An investigation carried out by the author in 1986 on earlier expert system shells revealed that most shells have limited facility for external interfacing and communication with user's other application software. External interfacing capability expand the shell's role of being able to be participate in an integration exercise, eg a company may wish to link all of their software application packages and at the same time add an intelligent front-end, using an expert system shell, to the decision making process.

Some occasions where external communication is required are:

(a) Passing of information between 2 expert systems, eg when 2 expert systems are run concurrently but with different knowledge bases and it becomes necessary to exchange information or decisions. It may also be the case when the expert shell is expected to transfer some decisions and/or data to, eg a payroll program.

(b) Retrieval of existing data from a database program as input to the expert shell.

(c) Interacts with the host operating system for some special functions to be executed.
overcome this limitation to a certain extent. Three main methods have become possible, although not all 3 methods may be offered by a single shell. The 3 methods are (i) a read/write function available as a command within the shell, (ii) external function execution, and (iii) software embedding of the shell.

(i) Read/Write/Message Functions

There are command functions provided by the shell and are used when, eg data values are required to be retrieved or stored away in a file, appropriately termed a communication file. A communication file is usually a specially formatted file in ASCII or binary. For example, a man-power planning program writes an account of the past week’s production activities to a communication file, Fig(4-23/a). At the beginning of the next week, the foreman uses the expert shell to read the data stored in the communication file, analysed it and receives a recommendation as to how the coming week’s work load should be organised. He may also wish to use the shell to produce a status report, or write data to another communication file to be used by another application program, or another expert shell running a different knowledge base. For WRAPS, it may be necessary to use the read/write capability for retrieval and updating of weld procedures. The message command, Fig(4-23/b) is used to produce a file of the right format containing information which is to be used by an external program. Returning information from the external program is read in from a communication file.

(ii) External function execution

The expert shell uses the ‘externals’ facility to pass an instruction to the host operating system, eg to rename a file, to protect a file from deletion or to execute another program. Fig(4-24) shows the concept. Consider the following application for WRAPS.

The expert shell analysed a welding application and revealed that a particular welding procedure could be upgraded and stored away in a particular format, eg as dictated by BS4870. The expert shell calls up another program, in this case the WELDSPECS program of the Welding
FIG(4-23) Communication of Information Between the KES Expert System Shell and External Application Programs.

FIG(4-24) Use of the KES Expert System Shell "External" Function. To Execute Another Program While the Shell is still Resident and Functioning in Computer Memory.
Institute, to archive the welding procedure.

Another example of external program executing is when the shell has to infer a decision which is based on the evaluation of complex mathematical equations. This may be beyond the capability of the shell. So the shell instructs the operating system to execute the special mathematical program which returns the result in some defined expert shell variable or store it in a communication file and read into the shell when required.

(iii) Embedding Expert Shell

This is when conventional programs could access the expert system techniques for analysis of the problem. The conventional programs are normally required to be written in the same programming language as used for the development of the expert shell. For WRAPS, this is possible because KEB is also available in the "C" language as well as in LISP.

The embedding is usually achieved via a well defined set of functions or procedures and data structures which are linked to the user's application program. This method of external interfacing for expert shells is the most versatile as it provides limitless possibilities in which the shell could serve as an "intelligent" component of the application software. Fig(4-25) shows the embedding concept. It greatly simplifies the design of decision-making or advisory components of the application program. Three major advantages arises from this method of communication:

i) The user could effectively tailor the interaction between his programs and the knowledge base, eg the inclusion of graphics with the shell.

ii) Real-time control of process and instrumentation becomes plausible. Fast time response may be achieved via assembler routines which are linkable to, eg "C" or Pascal language programs.

iii) Response is quicker from the "intelligent"
An Application that is not Embedded

An Embedded Application

Integration of a "C" Language Application with KES Functions and Knowledge Base

FIG(4-25) The Embedding Concept of the KES Expert System Shell.
component of an expert system embedded into an application program compared with an application accessing response from a standalone expert system.

At present, microcomputers have limited RAM space for running large programs. A program which is already executed calls another program requires even more memory, a situation similar to multi-tasking or the running of several programs independently of each other at the same time. Embeddability within an application program partially overcome this memory limitation because the whole program could be organised into overlays. See Chapter Three Sect 3-3-5 for discussion of the overlay concept which is used for the WRAPS implementation.

All 3 methods of external interfacing are anticipated to be used for WRAPS development. The embedded method is used for the On-line module for process control and is discussed in Chapter Six Sect 6-6.
CHAPTER FIVE
5.1 Introduction to Chapter

This chapter covers testing and evaluation of the WRAPS package for off-line programming and simulation. Its practicality and benefits gained are assessed.

The next section covers the programming procedure used for WRAPS. It describes an example starting with the creation of the models, calibration of the modelled environment, programming and creating the off-line program, and the eventual post-processing, transmission of the robot program and its testing on the robot system.

The modelling and programming examples quoted in this chapter are by no means intended as a user’s reference manual. It is intended to demonstrate and explain to the reader some of the unique features, such as the utilisation of time-saving relocatable sequences, special task related macros, automatic weldgun orientation in relation to the weld seam, and inclusion of welding points and optimised welding parameters into the robot program.

5.2 The Programming Procedure of WRAPS

All the modules in WRAPS produce some output in the form of a file at the final stage of operation. From the Modelling Module, it produces a file containing geometrical data (and possibly welding points coordinate data) about the objects modelled. From the Programming Module, it produces the WRAPS off-line program, and from the On-line Module, the final usable off-line robot program. The Expert Module reads in data from a file and occasionally updates the same file.
Chapter Five

The output files follow a certain naming convention, which is dictated by its extensions. This is described in their respective modules and should be adhered to avoid confusion unless it is deemed essential to rename file extensions.

The modelling of the objects follows a minimal visualisation attitude, i.e. only those parts of objects which are deemed to affect the programing procedure are modelled. In particular, protruding parts which may involve collision with the weldgun or robot should be modelled. This will allow some visual checking since collision detection is not implemented at this stage, as was explained in Chapter Three Sect 3-3-7. All the other parts of the object are not necessary unless impressive visualisation is required. This reduces the memory requirements of the software and improves the speed of animation. It is expected however, that new more powerful microcomputers will overcome this limitation. This will be discussed further in Chapter Seven.

5.2.1 Users Assumption

As one of the primary aims, WRAPS is intended to be easy to use. The user is not expected to be computer literate or an experienced computer operator. WRAPS was designed to require the minimum of keyboard entry and provide assistance to the user whenever he needs it. This assistance is obtained by pressing a dedicated "Help" key which is defined as the key "h" in the current software version.

The user need not be a welding expert, save for the case when there is no known welding database to draw information from and manual weld data input is required, or when an improvement to a weld is sought.

The user is expected to have the knowledge of using the robot. This is important to avoid trying to program a robot to move to an impossible point, or to avoid collisions. Users and robot operators must read, understand and practice the safety procedures described in the accompanying manual of the robot system.

The user must be aware of the responsibility in generation of the
computer model since the actual robot system would take at face value all the geometrical data as stored in the off-line program. For actual industrial applications, it is expected that WRAPS would eventually have more built-in features for checking of, eg clashes between robot and equipment, to reduce the responsibility of the user. The research leading to the development of WRAPS as a viable off-line programming and simulation tool has made it essential to include appropriate interface points for additional software to facilitate improvements of such nature.

The quality of the geometrical data is also important and the models created from the measured dimensions should be based on some minimum modelling tolerances.

5.2.2 An Example from Modelling to Creation of a Robot Program

It is necessary first to define the setup of the welding cell, the component to be welded, and the task associated with it.

1) Work Cell Setup

The example described is based on a work environment in the Welding Automation Laboratory in the Department of Manufacturing Engineering of Loughborough University. The environment consists of a CML T3 Type 566 hydraulic robot arm [207], a Teledyne Readco work positioner [208], and a Transmig R500+ GMAW machine with AWP TF2-4R wire feeder [209].

The weldgun holder was specially designed in order that the tip of the welding wire, at which the robot TCP is set to be coincident, could be adjusted to lie parallel along the centre line of the roll axis. This straight-through design has the additional advantage in that if there is apparent collision of the weldgun with the component or nearby equipment, another joint configuration could be tried out just by rotating the weldgun about the roll axis. This ensures that the weldgun orientation angles relative to the weld seam is maintained as suggested from the Programming module. This is compared against the usual
swan-neck weldgun holder as shown in Fig(5-1). The swan-neck version enables easier access to weld seams, especially inside enclosed areas like box sections, but would be difficult to use in practice for a trial-and-error basis to re-configure the robot joints in order to obtain the right orientation angles relative to the seam. Furthermore, it is difficult or impossible to be sure that the weldgun-to-seam orientation angles are correct with the swan-neck design. Values of TCP orientation which are along the roll axis centre line are easier to obtain as they are directly shown on the robot display unit. However, both designs will gain, or lose some accessibility to certain weld points, depending on the component to be welded.

TCP offsets are presently not implemented in WRAPS as explained in Chapter Four Sect 4-2, but the holder will be an asset for future development work. It should be mentioned that the TCP is not required to be set coincident with the wire tip at all times. This is particularly so when the robot is merely manoeuvred in space to move from one robot point to another. Setting a TCP away from the wire tip can economise the number of robot movements involved.

**ii) The Component**

The component used throughout this example comprises of 2 plates mounted relative to each other as shown in Fig(5-2). The dimensions are measured relative to the ends marked "REF".

**iii) The task to be done**

The weld required is a simple fillet type at (1) and a series of tack welds along a line beginning at (2). The component is initially manually tacked at 2 places as marked in the figure to prevent buckling or shifting between the 2 plates during the welding. The essential dimensions of the robot, the weldgun, and the work positioner were obtained by direct measurements or from their respective engineering drawings.
Old Swan-neck Weldgun (a) New Design for Off-line Design Programming

Maintaining Weld Angle θ Adjusting from A-B is Difficult

A - Collision
B - This Orientation Normally Reduces Reach Capability of Robot Making It Impossible to Get Far Positions

Required Weld Angle θ is Maintained by Rotating Weldgun About Roll Axis.

FIG(5-1) Comparison of Swan-neck Weldgun with New Design and Showing an Example When the New Design is Preferred Over the Old.

Tack Weld Both Sides

Material - Mild Steel Dimensions in mm.

FIG(5-2) Component Used for Off-line Programming Exercise.

Page 148
Chapter Five

5.2.2.1 Using the Modelling Module

The module is called up by typing MODEL and the RETURN key. A main menu appears as shown in Fig(5-3). There are 7 options, viz:

(i) Create Segments - This option enables the user to create the individual segments and later combine them to form the overall object. There may be only 1 segment for a simple object, eg a box. To model a complicated object, the first step is to select a global reference point, usually X = 0.0, Y = 0.0 and Z = 0.0, to which all other segments are translated or rotated with reference to.

(ii) Show data - This option displays the coordinates information about the present segment or object in RAM or from a disk file.

(iii) Save Data - This option saves the current data of a segment or object to a disk file.

(iv) Transformation/Zero - This permits transforming a segment of object to a new position and orientation about 3D space. There is also an option for initialising the data area.

(v) Graphics Mode - Normally, the user creates the segments to form the object in alpha mode. The user then could switch over to the graphics mode to view the current segment/object in 3D wireframe from any angle, and adjust the view parameters if required. A copy of the graphics screen may be obtained at any time by pressing the "ESC" key. Fig(5-4) shows an impeller modelled and displayed in the graphics mode. Also in this mode, there are options to:

- make start and end welding points within the model.
FIG(5-3) Main Options of Modelling Module.

- inspect the coordinates of the objects, including the whereabouts of welding points if any. Fig(5-4) shows this particular option in use.
- retrieve a previously modelled object for further work.
- create special pre-defined objects, eg. a T-section, a square hollow section etc. Pipe intersections are objects that would be possible. Fig(5-5) shows this particular option in use.

(vi) Combine - This option enables the joining up of several segments previously created and stored on disk to be merged into one overall object.

(vii) Utilities - This option leads to some system utilities, eg. to delete, rename or print a file.

(viii) Exit - This is the exit point for the Module. The user is asked to confirm before all previously dynamically allocated computer memory is released for other applications.

The objects required in this exercise are modelled consecutively:

(a) Modelling the Robot Arm

Although the current software does not include kinematic analysis of the robot joints as already discussed in Chapter Three Sect 3-3-7, a model of the robot arm may be displayed while in the Programming Module when the robot is at the "Base" or "Home" position. This demonstrates the capability of the software to model a complete robot and to be expanded with robot control algorithms when required. The option is set by a "Toggle" in the utilities in the Programming Module. The T3 model shown in Fig(5-6) shows an example of a complicated object that could be achieved with the existing software. The arm consists of 11 segments,
FIG(5-5) Creation of Special Predefined Objects with the Modelling Module.

FIG(5-6) Display of the Robotic Welding System at Loughborough. Modelled with the Modelling Module and Displayed With the Programming Module. Picture Shows the Cincinnati Milacron T3 Robot, the Working Surface of the Work Positioner and a Component on it.
excluding the weldgun which is shown as a straight line, and more
details may be added if required.

(b) Modelling the Weldgun

The weldgun consists of 6 segments as shown in Fig(5-7), 4 of which
are cylinders, 1 a cuboid, and the last a line. Since the TCP is
required to coincide with the tip of the welding wire, the reference of
the weldgun model is set there. The Line is created first, followed by
segment 2, 3, 4, 5 and 6 which are all appropriately translated and/or
rotated relative to the reference line.

(c) Modelling the Work Positioner

The work positioner, Fig(5-8), may be modelled by a simple
cylinder to represent the top of the table. The rest of the parts of the
positioner are unnecessary unless there are potential collision points.

(d) Modelling the Component

The component may be created from 2 cuboids. First, plate "1" is
created and then plate "2". Plate "2" is translated to its position
relative to Plate "1" and then joined together as shown in Fig(5-9). The
reference point is maintained at the base point as was shown in
Fig(5-2).

If this component is to be loaded into the work cell by the
Programming module as the default component to be worked on, then the
option "Set Default" must be called from the UTILITY function to alter
or insert the filename of the component in the file SETWORLD.DAT. At the
first work session, the work cell has to be calibrated in the
Programming module and therefore the component is to be read in manually
only after the calibration process.
1. A straight line with the TCP as the origin.
2. A hexagonal cylinder transformed linearly behind the line.
3. Another cylinder, but rotated and moved behind part 2.
4. Similar to part 3, but moved further behind. Alternatively, can make parts 3 and 4 as a sub-unit rotated together and moved back together.
5. A cuboid transformed linearly into position.
6. A 12-cylinder transformed linearly into position.

FIG(5-7) Modelling Sequence for the Weldgun.
Note - Control of the Rotate and Tilt motion is obtained through the Programming Module.

FIG(5-8) A Simple Model of the Work Positioner.

When the cuboid for part 1 is created, its origin is at the centre as shown in (a). Its origin can be shifted to the corner (or any where) by linear transformation (b).

Part 2 is also transformed linearly relative to part 1.

FIG(5-9) Modelling of the Component for the Off-line Programming Exercise.
5.2.2.2 Using the Programming Module

This module is executed by typing PROGRAM followed by the RETURN key. There are 2 command line options implemented:

i) PROGRAM DATE - this will enable the user to enter a date and any other comments of up to 80 characters long. This is stored in a file called LASTDATE for future reference.

ii) PROGRAM ASK - the system will ask the user if the data, eg the individual coordinates, number of points, number of segments that form an object etc, of the various objects being read in is to be displayed.

For future security purposes, a PASSWORD option should be implemented for software modules that contain sensitive data so that only authorised personnel may have access to for modifying the databases and expert knowledge rules.

The maximum amount of RAM remaining in the computer is reserved for use with this module and displayed. As the component was not set as default from the Modelling module, only the weldgun (attached to the end of the robot wrist) and the work positioner are read into the environment. The component to be worked is obtained and positioned on top of the work positioner via the 'ADJUST ENVIRON' menu by selecting the "Obtain Component" option.

It may be necessary to adjust the view parameters, using options in "Adjust Views", until an acceptable view of the simulated work cell is achieved.

Positioning the Objects in the Work Environment

It is required that the work positioner be calibrated in relation to the robot before programming could proceed. The calibration method, which is applicable to any robot system, uses the square planar frame as
described in Chapter Four Sect 4-3-1. The measurements of the "actual" frame marked out on the top of the work positioner has been previously obtained and stored into a file called T3WP.REF.

The calibration sequence is as follows:

i) Go into "ADJUST ENVIRON".

ii) Go into "Calibration".

iii) Select "Square Planar Frame".

The user is then requested to supply the name of the file that contains the coordinates and mounting information of the reference planar frame. The mounting information, which is at the beginning of the file, informs the system where to locate the frame relative to the robot base. Once the planar frame data has been read, the system transforms the frame to its appropriate and display is on the screen.

The actual calibration routine is then carried out by going into "Do Calibrate". There are 2 alternatives to input the measured coordinates of the calibration frame - to read in from a file or input manually. In this exercise, the former method is used, therefore enter T3WP.MEA. The software would then perform the averaging procedure as was discussed in Chapter Four Sect 4-3-1, and eventually transfer the work positioner and component with the mounted reference frame aligned to the measured frame.

After the calibration procedure, it may still be required to move the component to a referenced position on the top of the work positioner. This is especially needed when there are no jigs on the positioner which have been previously calibrated to hold the component to within acceptable limits. When adjustments are required, the user goes into the 'ADJUST ENVIRON' and select the "Move Component" option. The component is then adjusted into position using the arrowed cursor keys to move it "in", "out", "left", "right", "up" or "down" relative to the robot's frame of reference. It may also be necessary to use the "Orientate Component" option.

In using this calibration technique, it was possible to achieve a
calibrated working area to within less than +/- 1 mm accuracy.

Creating the Off-line Program

As a first step, it is advisable to create a mental picture of the robot path and activities involved in the intended task. A proposed sequence of activities for the test piece is as follows:

(a) Arm the robot with the gripper that holds the weldgun. This is specific to the set-up at Loughborough University. Involves turning on the air supply to the solenoid valves to clamp the gripper to the robot face-plate.

(b) Turn on power to the welding machine.

(c) Turn on power to the work positioner.

(D) Warm up the robot.

(e) Move TCP to "Cycle Start" position, a convenient and safe position in space.

(f) Orientate the welding component to the required position with the work positioner.

(g) Select welding voltage and

(h) Select wire feed speed for the fillet weld.

(i) Move towards the start point of the fillet weld.

(j) Adjust orientation of weldgun relative to the seam.

(k) Start arc and wait for 1 second.

(l) Weld from the start to the end point of the seam.
(m) Stop arc.

(n) Move away to a suitable position for the start of spot welding sequence.

(o) Select welding parameters and carry out the spot welding sequence.

(p) Move away and return to cycle start.

(q) 1 Return work positioner to its "Base" position.

                   2 Switch off power to work positioner.

                   3 Switch off power to welding machine.

(r) Return robot to "Cycle Start" position.

Note that those labelled in capital alphabets are grouped conveniently as sub-sequences, i.e. D, O and Q. Each sub-sequence will contain more than 1 robot instruction.

The next step is to define the start and end points of the weld within the component. In this exercise, there will be 1 pair of each for the fillet and spot welds. To generate the weld points, go into 'MAKEWELD' and select "Offset Between Points". The software will guide the user through the weld points generation. Basically, the user is asked where the weld point is to be generated with reference to a component coordinate point and by how much. The 4 weld points are generated in the same way.

The corresponding off-line programming instructions with the Programming Module are as follows:

Fig(5-10) shows the robot functions which were available for the prototype Programming module. To Select/change robot functions, velocities, welding voltages, wire feed speeds, flags, and tool
FIG(5-10) Robot Function Names Available in the Programming Module.

dimensions go into 'EXAMINE'. To switch on/off status of welding equipment, work positioner, input/output signals, go into 'SETTOOLS' as shown in Fig(5-11). When a menu option is shown in capital letters, it means there are further options contained within it. If the user is uncertain about the usage of an option, press "h" for assistance.

The main sequence 0 begins automatically at the "base" point, 0.001. The next point 0.002 etc in the sequence follows:

(a) Select "ToolIB_On" from "Robot Functions" in 'EXAMINE'. This is linked to the air valve of the gripper for the Loughborough system. Go into 'PROGPTS' and save this point using "Teach".

(b) As in a) select and teach "Weld_Machine_On".

(c) As in a) select and teach "Positioner_On".

(D) Call up "Robot Functions" and select "Perform 2". The warm-up cycle is named sequence 2.

(e) Go into 'MOVEGUN' and use the "Shift TCP" option to move the TCP to a selected cycle start position. Also, use "Orientate TCP" to change orientation of weldgun if needed. Select robot function "Cycle Start", select velocity required to reach this point, then go into 'PROGPTS'. Use "Teach" to save this point.

If it is necessary to determine the distance from a given TCP location to a component or welding point, go into 'EXAMINE' and then select 'Examine Distance'. The user would then be asked whether it is required to determine the TCP distance from a component or welding point. The results are displayed for example as shown in Fig(5-12).
FIG (5-12) The Option 'Examine Distance' Used to Show the Distance From an Object's Coordinate to the Robot Tool Centre Point.

FIG (5-13) The Menu for Weld Procedure Search and Retrieval From a Database.
(f) Go into 'ADJUST ENVIRON' and select "Orientate Positioner". Use the designated "UP" or "DOWN" to tilt or "LEFT" or "RIGHT" to rotate the positioner to the desired orientation. In this case only a 90 degrees tilt "DOWN" is required. Note that if the robot function "Positioner_On" has not been used, the positioner will have to be turned on via one of the 'SETTOOLS' option to toggle the status of the positioner to ON. Otherwise, any attempt to orientate the positioner will result in instruction to switch it on first.

(g) As in a) select and teach "Volt_Select". The user will be asked to input the welding voltage required for this joint.

(h) As in g) for "Wfs_Select", and wire feed speed values requested to be entered by the user.

(i) Go back into 'MOVEGUN' and shift weldgun TCP near to the start of fillet weld. Select the robot function "Move" and suitable fast velocity. Program this point using "Teach".

It may be appropriate at this stage to "zoom in" at the component so that the details could be shown more clearly. There are 2 ways to achieve this. The first method is to go into "Adjust View" and select the "Zoom" option. The view may then be "zoomed in" or "zoomed out" at a fixed interval by using the designated 'IN' or 'OUT' key. The second method is to go into 'EXAMINE' and select "View Parameters". Five parameters are available to affect the view of the picture displayed. These are view distance, view reference point, centre of projection, centre of projection view-plane and view-plane normal vector [149]. In this case, to "zoom in", select option '1' which alters the view...
distance. A value for the view distance is to be entered. A greater value for the view distance than which is currently displayed, will "zoom out".

(j) Go into 'MOVEGUN' and select "Goto Weld Point". In response to the system request, answer the number of the weld point which was designated, in this case 0. If the user is unsure, go into 'UTILITY' and use "Show Weld Points" to display the coordinates and point number of the welding points previously created.

Once the TCP has been moved to the desired start of weld point, the weldgun need to be orientated to the appropriate weldgun angles relative to the direction of the weld seam. Go into 'WELD EXPERT' and first use "Search Database" to look for an appropriate welding procedure that will contain the optimal data available. The typical screen display is as shown in Fig(5-13). The user will be guided through a series of questions and answers. Fig(5-14) shows an example during a query on the type of parent material to be welded. Then the software will scan through the database and retrieve a suitable procedure, if not the nearest best procedure. If none is found, the user is asked if manual input of data is required. Assume that a qualified procedure is found. Then go into "Get Best" to retrieve and use "Set Best" to set the necessary weld parameters contained therein. At this stage, only the weldgun orientation angles data is used. Now use "Set Weld" to automatically adjust the orientation of the weldgun relative to the seam. The software will ask for the number of the start and end point of the weld and also a reference face to the seam in which to calculate the weldgun to vertical face angle. When a
FIG (5-14) A Typical Screen Display During an Interrogation Session for a Weld Procedure Database Search.

FIG (5-15) Options Menu for Simulation of Process Parameters of an Off-line Program Created in the Computer's Memory.
possible orientation is achieved or otherwise, this is messaged to the user.

Finally, select the robot function "Move" and an appropriate approach speed to this start point, before save this point with "Teach".

(k) Select robot function "Weld_Start" and saved this point. Also select function "Delay" and specify a hold of 1 second. Also save this point.

(l) Go into 'MOVEGUN' and use "Goto Point" to define a move to weld end point. Once the move is achieved, go into 'WELD EXPERT' and use "Get Best" to re-obtain the previously selected weld procedure since some parameters, in particular the robot move velocity, may have been altered since the first "Get Best" instruction was used. Use "Set Best" to set the parameters again. Note that the welding voltage, wire feed speed and robot velocity values will changed in their respective boxes to the prescribed values given in the weld procedure. However, in this case, the welding voltage and wire feed speed has no effect because the previous 2 commands "Volt_Select" and "Wfs_Select" in (g) and (h) respectively, are given priority so that those 2 commands could be demonstrated to function on their own.

Select a "Move" function now and then save this point.

(m) Select robot function "Weld_End" and save this point.

(n) Use "Shift TCP" in 'MOVEGUN' to move weldgun to a location close to the start of the first spot weld. Select function "Move" and a suitable
velocity, then "Teach" this point.

(0) Use the robot function "Perform 1" to signify a sub-sequence to be used as the spot welding routine.

(p) Move the weldgun away from the component to a safe location. Use a reasonably fast speed to save overall cycle time. Use "Move" and then "Teach" the point.

(Q) Use "Perform 3" and save this point to mean a sequence to turn off the welding and work positioning equipment and return the robot to its "Base" location.

(r) Finally the main sequence 0 has to be closed loop. Use the robot function "End_Path" and then teach this point. The software will ask for a point to close loop to. In this case, it is the "Cycle Start" point.

At any stage during the creation of the off-line program, the user could go into 'SIMULATION', see Fig(5-15), to determine any of the following:

Cycle distance - this is the total distance covered by the robot for the program points already created in computer memory.

Cycle time - the total time taken to 'execute' up to the most recent program point created.

Cycle arc-on time - the total amount of time when the arc is turned on for welding, up to the most recent time.

Cycle productivity - a measure of the productivity
performance of the off-line program, calculated as a ratio of arc-on time over the total time of the program.

The points for sub-sequence 2, the warm-up cycle, consists mainly of several "move" commands to move the robot arm about in space so that all the joints are used. This helps to remove air pockets in the hydraulic system (in the case of the CML T3 system at Loughborough) while warming up the oil to the optimal operating temperature.

It is assumed that the warm-up cycle has been previously programmed and it is desirable to re-use the same sequence. Furthermore, it is assumed that the warm-up cycle had been saved as a relocatable sequence as shown in Fig(5-16).

Once the a sequence has been programmed, and this case the main sequence, the software automatically search for any other sub-sequences that need to be programmed. The warm-up sequence, or any other sequence, can be inserted into the program when the system messages "Making New Sequence" to the screen. In this programming example, the next sub-sequence to be programmed is due to "Perform 2" which is next in line in the main sequence 0. The insertion of sequence is effected by using the "Insert Sequence" option in 'PROGPTS'. The relocatable sequence has no initial sequence number and so the user is asked to enter a number, in this case "2". When the insertion procedure encounters "End Path" in the relocatable sequence, the user is asked where the "End Path" point of this warm-up cycle is. In this case, this is the point at step (e), the "Cycle Start" position.

The next sequence to be programmed is sub-sequence 1, the spot welding routine. This sequence is available as a macro to the user. The macro used in this demonstration is suitable only for a series of the spot welds between 2 specified component or predefined welding coordinate points or any offset distance associated with those points. In addition, the TCP centre line of the weldgun is assumed perpendicular to the horizontal plate. The macros are accessed via the option 'MACROS' within 'MOVEGUN'. The 'MACROS' option opens up to a few more options...
# M.R.A.P.S. Relocatable OFFLINE PROGRAM *
PROGRAM NAME = WARMUP.REL *
FUNCTION = Warmup Cycle *
Num of Pts is *

7

#### SEQUENCE LOGIC ####

<table>
<thead>
<tr>
<th>Pt No</th>
<th>FUNCTION</th>
<th>STATE</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Ddxy</th>
<th>Edgy</th>
<th>Rdgy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Move</td>
<td>1218.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>Move</td>
<td>1218.0</td>
<td>500.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>Move</td>
<td>1218.0</td>
<td>500.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>Move</td>
<td>1218.0</td>
<td>-500.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>Move</td>
<td>1218.0</td>
<td>-500.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>Move</td>
<td>1218.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>End_Path</td>
<td>1218.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### PROCESS PARAMETERS ####

<table>
<thead>
<tr>
<th>Pt No</th>
<th>DAC1</th>
<th>DAC2</th>
<th>VELOC NO.</th>
<th>TOOL DIM</th>
<th>VARIABLE</th>
<th>FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FIG(5-16) A Relocatable Sequence Format Saved As a 'Warm Cycle' Routine.
which are all macro functions. In this case call up 'Spot Welding'. When the spot welding macro is executed, it prompts the user to enter parameters in an interactive mode until all the necessary information required is obtained. The following information is required:

- start and end points of the spot welding centre line as shown in Fig(5-2).
- offset distances from the above 2 points if required.
- number of intervals between the 2 points.
- a suitable approach and back-off robot velocity from the spot weld locations. A default velocity is given. The approach speed is typically slower than the back-off speed.
- Welding parameters for the spot weld, ie voltage and wire feed speed.
- END_PATH point for the macro.

The welding parameters required for the spot weld is determined via a special instruction "DAC Group" generated by the macro, and which is identical in operation with the CML T3 robot function called "DAC Schedule". A "DAC Group" function was provided so that both a welding voltage and a wire feed speed value could be called up for immediate use in a single instruction. This is compared with using "Volt_Select" and "Wfs_Select" previously. The "DAC Group" may therefore be only specific to the CML T3 robot system. Refer to Chapter Four Sect 4-4-1-3 for discussions about the CML T3 DACs.

For sub-sequence 1, the "End_Path" point for the spot welding routine is at 0.019, just before "Perform 3" which resets the work positioner and turns off power to the welding equipment and work positioner.

The sub-sequence 3 will be programmed next. Again the "Making New Seq" message will be sent to the screen. Go into 'ADJUST ENVIRON' and select "Orientate Positioner". Tilt the work positioner up back to zero. Go into "Robot Functions" in 'EXAMINE' and select "Positioner_Tilt" and "Teach" this point. Next go into "Robot Functions" and select
"Positioner_OFF", and then "Weld_Machine_OFF" and "Teach" both instructions. Finally "End_Path" to 0.006.

Once the programming is complete, i.e. when there is no more open-ended sequences and when the software has messaged "Program Complete", the user could then save a copy of the robot program onto disk. It is recommended that a file extension of type *.OFF is used. Fig(5-17) shows the complete off-line program for the exercise, saved as "T3DEMO.OFF". The coordinates shown, however, will vary if a different weldgun or weldgun setup is used. Fig(5-18) shows the real environment set up and ready for welding using the off-line program created.

In addition, a sub-sequence could also be saved as a relocatable sequence for future use. This is achieved by selecting "Save Relocatable Sequence" in "PROGUTIL", see Fig(5-19). The user will be asked which sequence is to be saved and a filename to save it to. It is recommended to use a filename extension of *.REL. It should be noted that a sub-routine once generated from a macro could also be saved and re-used repeatedly as a relocatable sequence.

**Useful Utilities**

The progress of the off-line program could be inspected by going into "PROGUTIL". A brief list of all the points, up to the last one entered could be shown by selecting "Show Sequence". The user will be asked which sequence is to be displayed. The "Show Current" displays only the last point saved. It is noted that whenever a called-up robot function has been programmed in, the word "saved" will be displayed in the box next to the function’s box. This word will only disappear when a new robot function is selected.

The "Program Status", Fig(5-20), will show how many sequences are involved so far, how many have been completely programmed, which is the sequence currently being programmed, and how many points are in each sequence. It will also show a list of the current status of the welding equipment, work positioner, and input/output signals.
The WRAPS Off-line Program for the Programming Exercise Continued Over the Next 2 Pages. Name of Off-line Program is T3DEMO.OFF.
<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.001</td>
<td>Move</td>
<td>1218.0</td>
<td>0.0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.002</td>
<td>Move</td>
<td>1218.0</td>
<td>500.0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.003</td>
<td>Move</td>
<td>1218.0</td>
<td>500.0</td>
<td>-433.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.004</td>
<td>Move</td>
<td>1218.0</td>
<td>-500.0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.005</td>
<td>Move</td>
<td>1218.0</td>
<td>-500.0</td>
<td>-433.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.006</td>
<td>Move</td>
<td>1218.0</td>
<td>0.0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.007</td>
<td>End Path</td>
<td>0.006</td>
<td>1218.0</td>
<td>0.0</td>
<td>-933.0</td>
<td>180.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### SUB SEQUENCE LOGIC 3

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.001</td>
<td>Posner_Tj</td>
<td>2033.0</td>
<td>580.0</td>
<td>-425.0</td>
<td>100.0</td>
<td>0.0</td>
<td>90.0</td>
</tr>
<tr>
<td>3.002</td>
<td>Posner_OFF</td>
<td>2033.0</td>
<td>580.0</td>
<td>-425.0</td>
<td>100.0</td>
<td>0.0</td>
<td>90.0</td>
</tr>
<tr>
<td>3.003</td>
<td>Weld_Mc_Off</td>
<td>2033.0</td>
<td>580.0</td>
<td>-425.0</td>
<td>100.0</td>
<td>0.0</td>
<td>90.0</td>
</tr>
<tr>
<td>3.004</td>
<td>End_Fath</td>
<td>0.006</td>
<td>2033.0</td>
<td>580.0</td>
<td>-425.0</td>
<td>100.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### PROCESS PARAMETERS

<table>
<thead>
<tr>
<th>#</th>
<th>PROG NO</th>
<th>DAC1</th>
<th>DAC2</th>
<th>VELOC NO</th>
<th>TOOL DIM</th>
<th>VARIABLE</th>
<th>FLAG</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.002</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.003</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.004</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.005</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.006</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.007</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.008</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.009</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.010</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.011</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.012</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.013</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.014</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.015</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.016</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.017</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.018</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.019</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.020</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.021</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

#### SUB SEQUENCE 1

<table>
<thead>
<tr>
<th>#</th>
<th>PROG NO</th>
<th>DAC1</th>
<th>DAC2</th>
<th>VELOC NO</th>
<th>TOOL DIM</th>
<th>VARIABLE</th>
<th>FLAG</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.001</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.002</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.003</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.004</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.005</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.006</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.007</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.008</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.009</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.010</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.011</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.012</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.013</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

FIG(S-17) WRAPS Off-line Program. Cont'd.
1.014  1  0  2  2  0  0
1.015  1  0  2  2  0  0
1.016  1  0  4  2  0  0
1.017  1  0  4  2  0  0

--- SUB SEQUENCE 2 ---
2.001  0  0  6  0  0  0
2.002  0  0  6  0  0  0
2.003  0  0  6  0  0  0
2.004  0  0  6  0  0  0
2.005  0  0  6  0  0  0
2.006  0  0  6  0  0  0
2.007  0  0  6  0  0  0

--- SUB SEQUENCE 3 ---
3.001  0  0  5  2  0  0
3.002  0  0  5  2  0  0
3.003  0  0  5  2  0  0
3.004  0  0  5  2  0  0

========== PROGRAM DATA ==========

--- DAC1 --------
--- Welding Volts ---
36  24  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0

--- DAC2 -------
--- WFS mm/s ---
50  60  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0

--- VELOCITIES ---
--- mm/s ---
153  420  1000  2000
8000  12000  15000  18000
12000  22667  10000  12000
14000  16000  18000  20000

--- TOOL DIMENSIONS---
--- mm ---
102  -500  570  0

--- VARIABLES ---
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0

--- FLAGS ---
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0

FIG (5-17) WRAPS Off-line Program. Cont’d.
FIG (5-18) The 'Real Environment' Ready to Commence Welding Using the Off-line Generated Robot Program.
FIG(5-19) Utility Options Available to Assist in the Generation of an Off-line Program. The Relocatable Sequence Option is Also Selected From This Menu.

FIG(5-20) The Option 'Program Status' Used to Display the States of the Off-line Program Being Created and of the Equipment and Associated Input/Output Signals Involved.
5.3 Post-processing of Robot Program

The post-processing is called up from within the On-line module. The name of the WRAPS off-line program to be processed is requested and if it is not found, the user is given another chance to submit another file before it exits the module. It assumes a default *.OFF. From the data contained in the WRAPS off-line program it creates a logical data file of type *.TMP. At this stage the software would have checked through the WRAP off-line program for errors, eg unrecognised robot function names, robot parameters, missing qualifiers for a robot function etc. Any error is messaged to the user together with a summary list.

If no errors were encountered at the logical data stage, the software prompts the user if the final off-line robot program is required. The off-line robot program which is then created is ready for transfer to the robot controller.

5.3.1 Inspection/ Editing a Robot Program

The final robot program may be inspected through a function called HEXDUMP, and certain contents modified if necessary through another called EDITREC. Both are intended to be called up from within the On-line Module, but are at the moment standalone software functions. HEXDUMP provides a hexadecimal listing of the robot program, T3DEMO.CT3. Fig(5-21) shows the listing for the programming exercise example, T3DEMO.OFF, given in this chapter. The individual bytes of the robot program are displayed in the order that they will be sent to the CML T3 robot controller. The first column of the listing is a line number. Each line has 16 bytes displayed in hexadecimal numbers. The last 16 characters along the line are their ASCII equivalent representation.

The EDITREC function enables individual bits or a collection of bits of information to be modified. The collection may also be any number of individual bits other than a standard grouping of bits to form a byte. The user can position the editing cursor to any bit place, or at
WRAPS UTILITIES
K.H.60H Copyright (C) 1987
HEXDUMP
Version 1.xxx

>>> Dump of t3demo.ct3.
>>> File type is BINARY.

1: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
2: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
4: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
5: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
6: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
7: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
8: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
9: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
10: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
11: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
12: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
13: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
14: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
15: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
16: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
17: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
18: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
19: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
20: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
21: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
22: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
23: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
24: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
25: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
26: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
27: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
28: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
29: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
30: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
31: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
32: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
33: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
35: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
36: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
37: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
38: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
39: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
40: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
41: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
42: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
43: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
44: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
45: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
46: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
47: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
48: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80

FIG(5-21) Hexadecimal Listing of T3 Robot Program, T3DEMO.CT3. This is the Post-processed Output of T3DEMO.OFF Shown in FIG(5-17). Cont'd Over the Next 3 Pages.

Page 178
FIG (5-21) Hexadecimal Listing of T3DEMO.CT3. Cont'd.
FIG(5-21) Hexadecimal Listing of T3DEMO.CT3. Cont'd.
FIG(5-21) Hexadecimal Listing of T3DEMO.CT3. Cont’d.
the beginning of a group of bits within the robot program which is displayed on the VDU.

5.4 Transfer of Post-processed Robot Program

The physical link-up is as shown in Fig(5-22). The off-line robot program is transferred via the cassette recorder to the robot controller. In the process, an actual T3 data tape is produced. The instructions for operation of the cassette recorder for receiving and transmitting of data are found in the manuals [210].

The transfer uses the SENDRECV standalone version. To load this communication software type 'SENDRECV' followed by the "RETURN" key. The RS232-C port on the host computer is set to the required to the following parameters which are required by the cassette recorder:

- baud rate : 1200 bits/sec
- data length : 8 bits
- start/stop bit : 1

To initiate the program transfer, call up option "4" which is "Transmit file". The software then requests for the name of the file, i.e. robot program to be transmitted. Hence type "T3DEMO.CT3". Each byte of data being transmitted may be echoed to the computer screen if required by toggling option "2" before calling option "4".

To transfer the robot program from the cassette recorder to the robot controller, follow the instructions as given in the manual. At this stage, errors may be detected by the robot controller. Some robot systems, eg the CML T3, remove the amount of improper commands/actions by halting the transfer of off-line data into the controller memory as soon as eg, an unknown command or an impossible reach point is encountered during program up-loading. Errors reported by the robot controller onto its screen are shown in the form of error numbers whose meanings are found in the T3 programming manual [177]. Typical errors encountered were:
FIG(5-22) The Offline Program to T3 Robot Transfer Route.

1) Program name mismatch - this is usually operator error rather than error in the off-line program or error induced during transmission. The name of the program entered at the robot controller is different to the off-line program name being transmitted.

ii) Unknown robot function name - the off-line program contains robot functions which are not supported by the version of the operating system used in the robot controller.

iii) Data transmission errors - this may occur due to a fault in the data tape or the recorder itself. It may also be a problem in the physical link between the recorder and the robot controller.

5.5 Testing the Robot Program

Misuse of a WRAPS robot program could incur severe damage to both robot arm and/or equipment and personnel. It is imperative that an off-line robot program be thoroughly checked for improper commands and actions.

After the off-line program is transferred across successfully, it is recommended that robot programs created externally be checked first in Dry Cycle mode, and then in a step-by-step fashion in Teach mode before running the system in Automatic mode. These 3 modes should be available on all good robot systems. The user should refer to the robot manuals to learn how to operate in these modes.

5.6 Evaluation of Test Results

The program for the demonstration ran satisfactorily. However, in other tests carried out, some of the off-line programs require occasional "fine-tuning" of robot points. Impossible robot joints
configurations were corrected by manipulating the robot in teach mode. This becomes minimal once the WRAPS operator gets accustomed to the system, and became more aware of the restraints of the robot from experience. The benefits of having a kinematic or indeed a dynamic model of the robot become obvious, as discussed in Chapter Three Sect 3-3-7.

The requirement for "fine-tuning" is attributed to (a) errors in measurement of the relative positions of the components of the work cell, and (b) the inaccuracy of the CML T3 robot. The amount of "fine-tuning" required for (a) varies with the magnitude of errors involved in measurements taken of the work cell components. The calibration method used earlier in this exercise had required a maximum of +/- 2 mm of robot points "fine-tuning". A more acceptable and easily-used technique must be developed. However, calibration of the workplace is a deep subject and should be a separate research topic on its own.

The amount of "fine-tuning" required due to (b) was only as much as the inaccuracy of the robot system. The coordinates from the off-line program and the robot controller display are identical except for digital "rounding-off" errors; hence a total probable error arising from the robot is its operating inaccuracy plus +/- 1 mm of rounding error. However, the setting of the TCP at the welding wire tip, if improperly done, can significantly affect the positional accuracy of the programmed task. If the TCP is set exactly at the wire tip, then an orientation of the weldgun will not alter the position of the TCP (at the wire tip), otherwise the wire tip will be swung round at an arc about the actual position of the TCP. With the weldgun attachment on the CML T3 robot, it was possible to set the concentricity of the wire coming out along the roll axis of the wrist and the TCP distance to +/- 1 mm in both cases. It thus become more obvious that a weldgun having a straight-through design is more practical for off-line programming purposes, than a swan-neck weldgun design as was discussed earlier in Sect 5-2-2. Inaccuracies due to the setting of the weldgun can also affect the errors due to (a) if the wire tip were to be used to carry out the calibration measurements as was so in this off-line programming exercise. Consequently, an associated problem with an off-line programming system is that whenever the weldgun is disturbed out of
position, the system have had to be re-calibrated. This could be avoided if a special weldgun alignment tool system is devised to re-locate not only the TCP but also the weldgun orientation. A disturbed weldgun, however, is a problem for on-line programming as well as off-line programming.

The initial repeatability of the robot was +/- 0.5mm but that has become impossible to achieve with the current state of the robot system. A test was made to assess the influence of the robot inaccuracies on off-line programming. The method used to measure the accuracy and repeatability of the CML T3 robot is a simple variation of one of several techniques [211-215] after consideration of the condition of the robot system and the equipment available. The equipment used comprises of an angle block and 3 dial test indicators (of resolution of +/- 0.001 inch (0.025 mm)) arranged orthogonally as shown in Fig(5-23).

The cube was held in the gripper and its top face was made level, by observing an engineer's level indicator and rotating the wrist of the robot. Next the robot was taught to move to a reference point where the 3 dial indicators were set to zeroes. Then the robot was made to move to the referenced point repeatedly, at various speeds, from a point which was about 45 degrees to the reference point from under. This was a point far enough so that the robot would have surpassed the initial acceleration to reach the required velocity. Readings of the deviations from the referenced first point were taken from the dial test indicators for a series of speeds.

The results show deviations of up to +/- 0.8 mm, compared to a specified repeatability of +/- 0.5 mm. Fig(5-24) shows the results for the range of robot velocities from 3001 mm/min (50 mm/s) to 15005 mm/min (250 mm/s). The maximum velocity of up to 20996 mm/min (350 mm/s) could not be tested because the repeatability results was unpredictable due to the amount of over-shoot which was affecting the reading of the dial test indicators. Also, the results were only plotted for the Z-axis as the effect on repeatability in the X and Y axes were negligible – the former deviated within a dead band of +/- 0.001 in (+/- 0.025 mm) while the latter was +/- 0.002 in (+/- 0.050 mm). The obvious effect in the repeatability in the Z-axis was that the robot had to move the weight of
FIG (5-24) Measured Deviations from a Zero Referenced Point, in the Direction of the T3 Robot "Z' Axis."
the arm against gravity.

The real concern is the amount of over-shoot of the robot before the TCP settled down to its targetted position. For the most affected axis, which was the vertical Z axis direction of approach, the over-shoot varied from +/- 1.3 mm for the lowest speed used of 3001 mm/min (50 mm/s). In the worst case, at the highest velocity used of 20996 mm/min (350mm/s), the over-shoot was +/- 5 mm. This was performed only with 1 approach direction and has not taken into consideration the effects of TCP orientation or location within the operating volume of the robot. Other research work [25,216] has proved that the positional repeatability of a robot is influenced by various approaches, positional reach, orientation, and load carried. The reader is referred to such work for full details. The simple test carried out here was sufficient to allow consideration to be given to the calibration of the real world to the modelled environment and subsequent off-line program generation.

The amount of over-shoot encountered suggests that low speeds should be used to approach locations near to a component to avoid crashing against it. For example, for the T3 robot, component vicinity approach speeds should not be over 4000 mm/min (about 67 mm/s). An inaccurate robot system, and especially one with poor repeatability, will also present problems to calibration of the work place. Such a system, however, can also affect the performance of the program taught by teach pendant. If the discrepancies due to off-line programming are significant, then "fine-tuning" of the robot points becomes necessary. The amount of corrective work performed at the workplace with the robot system must be minimised as far as possible, otherwise the potential advantages, as outlined in Chapter Two, of off-line programming will not be attained.

5.7 Economic Evaluation

Measured savings could only be determined for a task-for-task comparison basis, as each task is of a different nature. As such, only case studies would provide some form of feasible comparison. In carrying out the economic evaluation, the author had followed the techniques and
hints given by researchers [217,218] who have worked specifically in these or similar areas.

Considering the exercise given in this Chapter and in some other tests, eg programming the robot to carry out a warm-up cycle, and in a pick-and-place program, the results of time study confirm a programming time saving of up to 50 % better than the teach pendant. In cases, where an off-line program may take the equivalent time as the teach pendant, the advantage of having no down-time on the robot is sufficient to justify off-line programming. In an off-line application at General Motors (US), off-line programming offered the only viable method to avoid shutting down a 24-robot production line which was continuously performing critical resistance spot-welding operations on major structural components of each of the cars passing through the line at a rate of 75 cars per hour [219].

The programming time and effort saved in the use of relocatable sequences and macros for pre-determined cycles of tasks could be substantial as they need to be programmed once. Macros are particularly beneficial as the user only need to enter various parameters. However, the macros have to be written in the first place which depending on the job would take several days to write. The spot welding macro took 3 man-days to produce.

Other recent reports of time-related savings in the use of off-line programming suggests economies along the same line [220], but WRAPS provides additional (currently) intangible benefits:

i) direct interface to access optimal process parameters, hence better potential to produce higher quality welds or products.

ii) reduction of programming strain on the user because of its simple user interface and off-line programming approach.

iii) WRAPS is a system that is likely to be affordable by small firms interested in off-line programming and
tools to aid in the integration of their manufacturing facility. This is because the research into WRAPS has brought about a prototype system that could be used on low-cost microcomputers and have front-ends capability of interfacing with other off-the-shelf products, eg spreadsheets or database programs.
CHAPTER SIX
6.1 Introduction to Chapter

This chapter looks into various aspects in which WRAPS will be used to integrate the welding process and to provide operational control for a flexible robotic welding cell.

Sect 6-2 discusses the requirements of an integrating environment and some aspects which WRAPS should have to achieve it. Sect 6-2-1 discusses three network methods to link the various components of WRAPS. Part of the integration incorporates process planning, in particular with CAD input and design considerations. Sect 6-3 examines how this interface to other software or external packages can be achieved with WRAPS.

Sect 6-4 explains the implementation of the weld process integrating capability of WRAPS. A simulated demonstration is also discussed.

6.2 The Integrating Environment

Present technological developments dictate a natural progression towards fully automated flexible manufacturing systems. Although it is not expected that WRAPS should have every conceivable requirement of a CAD/CAM system, it is expected that WRAPS can provide many useful facilities on a low-cost basis to aid such developments, particularly in welding automation.

The evolutionary route to an automatic factory where CIM (Computer Integrated Manufacturing) prevails starts with CAD (Computer-Aided Design), followed by CAM (Computer-Aided Manufacture), FMS (Flexible
Manufacturing System), AMT (Advanced Manufacturing Technology) and CAE (Computer-Aided Engineering). Additionally there are other areas, eg CAPP (Computer-Aided Production Planning), and CAST (Computer-Aided Storage and Transportation), which are rather haphazard discontinuous additions to the progression towards the CIM objective. CIM and its predecessors are years apart [221]. The main aim of CIM is to provide a framework for a complete business strategy which is based on company-wide integration allowing inter-linkage of mutually supportive islands of information which are indivisible as far as inter-communication is concerned. CIM is intended to mesh together every department and every activity within departments. Information flow and its management are obviously of paramount importance. Clearly to achieve integration, the information must be accessible on-line and not stored away in filing cabinets or in off-line tape systems.

WRAPS fits into the flexible robot welding cell within the CIM environment as shown in Fig(6-1/a). The cell may be standalone as a welding unit or be part of a series of operations that make up the whole manufacturing process, eg where a component requires some parts to be cast, machined or fabricated.

6.2.1 Integrating WRAPS and Hardware Through A Network

An integrated environment comprises machinery and tools of various functional types. A robotic arc welding system includes the robot arm and controller, welding power source, wire feed system, weldgun, jigs and fixtures, gas/water solenoid valves with all the connecting cables and hoses. Also there will be safety features, such as weldgun collision detection and gate interlocks to prevent intrusion and injury to human workers.

An integrated welding cell can be managed by a cell controller, which is itself connected to an industrial or plant local area network as part of a CIM system. The cell controller attaches all the unit controllers that operate the robots, sensors and material handling equipment. Some form of communication of data between the various units within the manufacturing cell is possible and required since
(a) The WRAPS Welding Cell in a CIM Environment.

(b) A Weld Quality Control Cycle in a WRAPS Cell.

FIG(6-1) The WRAPS Flexible Welding Cell.
microprocessors and programmable logic controllers (PLCs) are built into almost every machine tool, process, robot, automated conveyor handling and storage system, and test and inspection equipment on the shop floor. It is possible that these microprocessors and PLCs are in turn directed and controlled by a hierarchy of computers acting as unit controllers, area controllers and finally plant controllers.

As a low-cost task dedicated system, WRAPS is expected, at the minimum, to provide some form of control of such equipment serving as a welding task supervisor. WRAPS should provide programmable functions whereby certain physical tools may be activated or de-activated. Some examples of such applications are to attach a gripper mechanism for the weldgun holder, or to start a conveyor system once a particular job has been completed. It may even be to initiate the start of another task on another robot system. Similarly, WRAPS should also be able to accept an input signal from another system, eg a robot loader, to indicate that the next job in hand is ready. Thus in a process integrated environment, WRAPS is required to have messaging and input/output capabilities with external devices.

When installing a range of computer equipment from a single supplier, it is generally designed and built on a basis of a single communication architecture, eg for IBM it is the Systems Network Architecture (SNA) [222] or the evolving Systems Applications Architecture (SAA) [223]. This enables the computers to exchange data usefully with one another. Similarly, other suppliers may have their own propriety communication architectures, which allows each product range to be interactive to some extent. This inadvertently creates considerable difficulties in inter-connection between computers as protocol differences prevail. Furthermore, different computers use different data type representations internally and different applications use different file formats.

In order to have useful communication between equipment from different suppliers, a worldwide set of communication architecture standard is required. The International Standards Organisation (ISO) aims to provide this standard by the development of the Open Systems Interconnection (OSI) [224,225] which attempts to overcome the various
computer communication architectures.

In the United States, two major companies have inspired international standards for communication within the factory - General Motors, the Manufacturing Automation Protocol (MAP) [221,225-229] and Boeing, the Technical and Office Protocol (TOP) [225,227-229]. They have demonstrated the space-age factory to be a technically and financially viable possibility. In the United Kingdom, sixty companies collaborated to show that a mock-up factory of fifteen production cells could be integrated under a "ring main" CIM system at the CIMAP event in Dec 1986 [230]. There is increased worldwide interest [231] through work promoted by, eg the Europe MAP User's Group, and other events, eg the Communication Network for Manufacturing Automation (CNMA) in Hannover Fair (April 1987, Germany) and the Enterprise Networking Event (to be held in June 1988 Baltimore, USA).

It is inevitable that the total WRAPS concept is based upon a network system, in view of MAP and forthcoming standards. Batch fabricators who are component suppliers to such big corporations will eventually be required to be able to communicate directly with their computer systems, to receive orders or for transmission of components data. WRAPS should therefore be extended to enable modern communication by such protocols. However, emerging MAP specifications, the latest being Version 3.0 at the time of writing this section, requires costly hardware and software implementation. A MAP starter kit can cost between £20,000 to £50,000 [232]. MAP has also been criticised for the slow data transfer because of the 7-layer route, see Appendix(A10).

For time critical data transfer, such as will be in the case of the adaptive aspect of the WRAPS cell, it will be necessary to use a scaled-down implementation of MAP. Two systems are available at the moment, one is called MAP-EPA (EPA - Enhanced Performance Architecture) and the other Mini-MAP. Both are comparatively less costly than the full MAP [221,232-234]. More importantly, it is intended for use at the cell (rather than plant) level of sub-networks where short messages must be communicated quickly, such as in real-time communication between a robot and a welding system.
MAP-EPA is a dual architecture device incorporating a full MAP version on one side and a so-called "collapsed architecture" on the other. The 7-layer side allows it to communicate as usual with other full MAP nodes whilst the other 3-layer side provides rapid access to like nodes.

Mini-MAP only has the 3-layer architecture and therefore is not an OSI compatible device. However, it appears to be the least expensive at the moment, and it can gain access to other nodes of a full MAP network through the MAP-EPA if there is a future requirement [221]. An example of Mini-MAP is the GEnet carrierband local area networking interface from GE FANUC [232,235] which can provide support from 6 to 20 nodes. Mini-MAP reduces the 7 layers to 3. Mini-MAP costs about £2000 per node (for the plug-in boards for PLC or a personal computer) and another £200-£300 for the software.

MAP may be required for WRAPS from a global point of view, however, from within WRAPS, only some form of networking system shall suffice to allow local communication among the various sources of information. It is only a "bridge" that would be required to interface the WRAPS local area network (LAN) to the wider-based MAP plant network. To this effect, the concept of a distributed system of standalone modules for the data acquisition and the data-flow management within WRAPS is discussed below.

The reader who is unfamiliar with network standards and jargon should refer to [236-238] to obtain first time user information. Three possible ways of creating a networked system are suggested, starting with a proposed system based on (a) the IBM Token Ring network system, (b) the IEEE-488 and (c) proprietary controller systems. A proposal for the WRAPS network system is made in (d).

(a) The Token Ring

The Token Ring (IEEE 802.5) [239,240] works on the token-passing principle. Each node has a predetermined window of opportunity within which to initiate transmission of data. A 24-bit packet of data, called
the token, is passed around the system. If a message is to be sent, the node waits until the token comes round and then transmits its packet of data. No other transmission can occur until the packet has reached its destination and its data unloaded, at which point the packet is re-transmitted by the receiving node.

The Token Ring is suited to the WRAPS cell integration application, than for example compared with a linear bus network system, because it does not have the risk of a complete network stoppage due to a failure in one of its cables, and priorities can be allocated to the nodes, giving those at the top of the hierarchy faster access to resources than those at the bottom. It provides better control of access to the information being passed round the ring, through the scheme which is known as "Deterministic access" [237]. For example, it is necessary to acquire data about the process prior to accessing the node that contains the expert system that is used to interpret the data. Also the token-ring topology avoids the contention problems associated with other systems, eg Ethernet, where collision detection is needed. The reader may find more information about other network topologies in [236-239], eg the distributed star and the star-wired ring topology.

A network system based on a widely used topology such as the token-ring offers scope for expansion and flexibility in the control of welding cell. The main drawback is the quantity of wiring required, as traffic gets heavier and the network increased in size, thicker and longer cables are required, hence incurring high expenses. Installing and maintaining a network system, including the token ring, may not be a straightforward exercise and can be plagued with problems in certain cases as has been reported [241,242].

(b) IEEE-488

An alternative to the LAN for WRAPS for control of the flow of information and the process itself is the use of the IEEE-488 interface Bus [243]. It is also known as GPIB, HPIB and IEC625. This system enables the connection of a number of instruments (which may be sensors and/or computers) to a common input/output port of a supervisory
computer, which also controls the other software programs running from other computers.

In the standard IEEE-488 interface definition, an 8-line bus carries addresses, data and commands between instruments and the supervisory computer. Each instrument is referred to by the supervisor by an address assigned to it. After activation, the addressed instrument may either send data (talk) or receive data (listen) before being de-activated (un-talk, un-listen) and the next instrument activated. When an instrument has data which it wishes to send, it interrupts the supervisor which in the meantime continuously poll the instruments it controls.

The IEEE-488 is an established technology for linking with programmable devices to computers, and for serving as a network interface, usually over a short distance, for computer peripheral devices such as printers, plotters, disk systems and tape systems. Its role in a network system, especially in the MAP environment, has yet to be demonstrated. The main advantages are its low cost and ease of implementation.

(c) Propriety Control Systems

Another alternative is to use the PLC which has already been described above, to interconnect the various aspects of the cell. The use of PLCs to serve as cell controllers are increasing [244-247]. Before using PLCs in a cell, it is important to consider several functions, for example, performance of hardware and the overall cell supervisor operating system. Some pointers to bear in mind before designing a real-time system is given in [248].

However, an improved solution, based on the second alternative, is to use one of the numerous flexible microcomputer based data-logging systems in conjunction with WRAPS, albeit some do not follow the directions of the MAP/TOP for standardisation. Some of these systems offer the capability to acquire data like a logging machine, controls a process like a PLC, and displays and archive data like a graphical
spreadsheet package. They may be configured to interface to a wide variety of digital and analogue input/output devices, and with intelligent equipment and computers with serial communication facilities. A detailed survey on distributed control systems in [249] lists a selection of these systems. Most have proliferated out of the need for integration and flexible control purposes. They are obvious immediate benefits:

- uses for data logging as well as process control.
- simple to use low-cost turnkey system.
- acquired data is linked to database management system such as spreadsheets and special data analysis programs.
- easily re-configurable to meet application changes.
- some have capability to network through MAP.

The obvious disadvantage with these systems is that a startup integrated control and data acquisition costs from £10000 onwards. Also some systems will limit communications within the system to an RS232-C line.

(d) Proposal

Three possible methods of linking the individual components of WRAPS have been proposed, but the author propose that the method based on the token-ring (IEEE 802.5) be used primarily because:

- this topology has been proved viable as a data highway system for several distributed control systems in industry [250]. In addition, the interface to a MAP environment through a bridge has been demonstrated [251]. Immediate implementation of MAP is not required as MAP is still being continuously upgraded and is currently not the most cost effective approach. The token-ring is a widely supported standard with a published stable architecture.
- flexibility to optimise control strategies within the networked system. A distributed control system can be implemented, which is preferable than, eg the usual method of using a central computer to control every required aspects of the welding process. Here, each entity within the network can be considered as an intelligent node by using a computer dedicated to look after one or more manageable tasks, eg one computer to control the data acquisition relating to the robot and welding equipment, another computer to control the motion of a dynamic sensor (such as the laser triangulation sensor) to acquire dimensional data of workpieces, and yet another to make decisions based on the data acquired. This scheme will ensure that only the necessary information is transmitted to the supervisory computer.

- WRAPS is considered as a low-cost integration tool and therefore low-cost network and interfaces to equipment and transducers are essential.

- affordable within the financial restraints of the WRAPS project.

Fig(6-2) shows the proposed token-ring based system for WRAPS.

6.2.2 Integrating WRAPS with other Software Packages

The WRAPS package on its own may be inadequate to some users who require more than just off-line programming. Compatibility with other software packages that they use may be required. For example, a user may need to use some special database stored in a particular format together with WRAPS. There are 2 feasible approaches to import/export information from WRAPS and the appropriate approach will be adopted when the need arises.
FIG (6-2) Token-Ring Topology as LAN for WRAPS Flexible Welding Cell
1) Through Files
Passing of information through files on a storage medium. The files maybe in ASCII or binary format. The ASCII format is slightly is usually slower in terms of access time, but instant readability of its contents is possible by listing it to a computer screen or printer. On the other hand, binary format files are indirectly readable and the exact structure of its contents must be known prior to reading and interpreting the information. But binary format files are more efficient in terms of storage and access time.

This approach has the main advantage that the third party software remains un-altered. This should avoid modifications to the standard packages, requires less development time for interfacing, and ensures easier upgrading to newer versions of the packages. Many widely used packages, eg LOTUS 1-2-3, dBASE III, Framework, etc, could be accessed via this approach. The welding procedures produced by the WELDSPECS program as discussed in Chapter Four Sect 4-5-1 can also be accessed similarly from the Programming Module of WRAPS. However this would also mean that a 2-way data format translator is required.

This should avoid modifications to the standard packages, requires less development time for interfacing, and ensures easier upgrading to newer versions of the packages.

2) Direct interface
This involves the embedding of the third party software into the WRAPS package or vice versa. This is possible in a manner similar to the
embedding of KES expert shell into WRAPS which is discussed further below in Sect 6-4-5-2. Where both the packages are written in the "C" and assembler functions, this approach requires the supplier of the third party software to divulge considerable interfacing information and perhaps "bridging" functions which WRAPS can call directly. In these cases, it would only be practical if the third party software is also written in the "C" language, or in relocatable Assembly language. The main advantage with this method is faster interaction speeds, and if a "standard" third party software package does not perform some particularly desired function, eg lack of graphical display or a sorting routine to make an ordered list for interpretation, then some specialised additional software could be written and linked together with the rest of the combined packages by the user.

6.3 The Process Planning Interface

Process planning is important in contributing improved efficiency in a welding environment. It could remove welding bottlenecks, reduce work handling, and achieving the maximum possible arc-on time. It can decide in advance during the planning stage whether a component is indeed suitable for robotic welding or normal manual welding. WRAPS should therefore provide such decision-making to be possible, with the "intelligence" built-in or assessed or consulted via external packages.

There are several total integrated process management software packages with which WRAPS can be interfaced. Two examples are quoted:

(a) ActionFile Software [252]
Comprises of a range of microcomputer software which provides a tool for the management of complete business operations, encompassing the
Chapter Six

Accounting, commercial, distribution, Purchasing and Production functions. It allows, eg for sales order processing, production documentation, parts explosion, material requirements, planning and purchasing control, standard costing and job costing.

(b) Hornet Project Control [253]
It is a self contained project/process management system for microcomputers. Bar charts, histograms and schedules could be prepared to show the current status of project/process. Can evaluate multi project/process time scheduling, resource smoothing and project/process costing.

It is desirable that, eg a bill of materials is available for a batch job. This bill can include items like type and amount of materials used, time required to complete the job, unit costs, labour requirements, job preparations required, etc. The efficiency of job production can thus be optimised through careful utilisation of such information. See [254] for a survey of production management systems. Some CAD packages incorporate some of these facilities, eg VersaCAD [255]. Other specialised software packages, available from the Welding Institute, that also help in decision-making may be used in conjunction with WRAPS. For example, it is intended to interface the following packages to WRAPS:

i) WELDCOST - for the calculation and analysis of welding costs for arc-welded fabrications.

ii) PREHEAT - for the calculation of "safe" combinations of preheat temperature, arc energy, carbon equivalent and weld hydrogen level for the prevention of heat affected zone hydrogen cracking.

iii) WELDVOL - for the calculation of weld sizes and consumable requirements for a particular weld application to assist cost estimating and
consumables ordering.

To avoid unnecessary modifications to standard vendor packages, as already discussed in Sect 6-2, the best method of interaction with WRAPS is via binary or ASCII data files of a well defined format. This method is suitable as the external data format of both the above packages are publicised.

6.4 Weld Process Integrating Capability of WRAPS

The following section discusses the concept of the integrating capability of WRAPS for the GMAW process. The current implementation work, based on this concept, will be discussed in Chapter Seven. The aim is to demonstrate how a component for welding is subjected to pre- and post-welding geometrical inspection to enable a total closed loop feedback of the welding process to be achieved. It assumes that for a particular component, an initial weld procedure, has been selected either by manual input or an automatic search by WRAPS. The latter method will produce a weld procedure which is optimised if the weld database is maintained up to date. For discussions here, a scenario is presented in which the task is to obtain successive upgrading of the initial weld procedure until a minimum quality acceptance is attained.

A complete WRAPS closed-loop welding system is shown in Fig(6-1/b). The stages involved is categorised as Pre-weld Inspection, Welding and Inspection, Post-Weld Inspection.

In addition, time measurements are required to enable a process planning strategy to be implemented. These are:

i) Robot cycle time.
ii) Arc-on time.
iii) Total throughput time, including time for handling, transportation and idling.

Component and batch counts may also be necessary.
6.4.1 Pre-Weld Inspection

A welding procedure assumes a certain quality of assembly and fit-up of the components to be welded. The parts to be welded are fitted up and an inspection is done to determine the following:

i) Type of joint identifier - this may be a manual input by the operator at the computer keyboard, or an automatic recognition system. The latter may be, eg a bar code or proximity sensor system which is directly linked to the supervisor computer.

ii) Joint Data - this is basically dimensions of the component parts and its relative fit-up relationships.

iii) Joint Orientation - relative to the global reference axes system.

The latter two classes of information can be acquired by existing technology, eg using laser sensor based on the triangulation [24] or scanning [256] or stripe [37, 257] principle, or electronic touch trigger probes [24, 258, 259], which is attached to the end of a manipulator system. The sensor may be carried by a robot or an automated coordinate measuring machine. The data deriving from these sensors may be transformed to binary coded digital data which is easily interpreted by a control computer. The principle and operation of the sensors are covered in the background study of Chapter Two.

6.4.2 Welding and Inspection

Certain features occurring during the welding process give positive indications of weld quality, eg penetration and bead width. These can be monitored in-process to regulate the manufacturing of defectives by shutting down the system, and also provide further data feedback for procedure optimisation and process control. Two main areas of information may be elicited during the welding process. The first is from the robot welder system and the second is from the weld itself when
it is being generated.

The information obtainable are:

(a) The robot welder

i) Robot position points; its coordinates and orientations.
ii) Weldgun standoff distance
iii) Voltage and amperage across the arc.
iv) Wire feed speed.
v) Gas flow rate.
vi) Corrections made to path of robot during welding as determined via the seam tracking capability of the robot.

(b) The weld

i) Penetration of the weld, possibly by using a frontface or backface penetration sensor [260]. This will involve conversion of light intensity to analog and finally to digital signals for the computer.
ii) Dimensions of the bead.
iii) Width of weld pool.

6.4.3 Post-weld Inspection

The inspection falls broadly into two main areas — automatic or manual inspection.

(a) Automatic

The type of data required is basically the same as for pre-inspection, i.e. mainly dimensions and geometry eg bead width, bead reinforcement height, but which may be more readily acquired than by
in-process. This requirement is similar to the work reported by Terry [261].

(b) Manual

Most of the inspection data required here are off-line and some may take longer to access and assess than others. In addition the data is expected to be manually incorporated into the system database for interpretation. These may include visual inspection of weld appearance, surface finish to provide subjective results or may involve non-destructive testing methods to identify cracks, porosity or to determine its macro structure.

6.4.4 Data Acquisition

The automatic inspection of the components as discussed above will not be purposeful if the data is not harnessed in an appropriate and interpretable form for immediate or delayed analysis, and stored for future references. The need for computerised data measurement and acquisition has been covered to a certain extent in Chapter Two Sect 2-2-4.

The advancement of low-cost precision analog/digital interface systems, which could effectively perform all the functions of larger dedicated instrumentation and control systems but at a fraction of its former cost, has enabled a microcomputer interface which could be easily configured and expanded to satisfy a variety of instrumentation and control applications.

"Data Acquisition" is the collecting of information that describes a given situation. The data typically reflecting what was happening when a given condition was satisfied [262].

There are now sensors/transducers which can provide data logging
for the following welding parameters:

i) voltage - direct connection between the contact tube of the weldgun and the work positioner via heat resistant leads. This will not give the actual arc voltage because the ohmic resistance between the workpiece and the table is not taken into consideration but will be accurate enough as initial implementation. An alternative method is to tap the voltage (and current) feedback from the welding machine to the robot controller and/or vice versa. These two-way systems are becoming common with the new generation of robots, e.g. the FANUC ARCMATE welding robots. Older systems have to rely on the welding equipment to provide the parameters information.

ii) current - using current shunts or Hall-effect probes, or similarly tap off directly from robot as in voltage above.

iii) weldgun to seam angles - vision systems, or from the robot control system.

iv) wire feed speed - tachometers, with solid state speed control with external output capability.

v) gas flow - pressure differential gauges.

vi) weldgun traverse speed - obtained directly from the robot control system.

vii) arc time - digital clock with external output, or from the robot control system.

viii) temperature - pyrometer or contact probe with external output.
ix) standoff - vision systems.

x) weave pattern - robot control system.

xi) shift in component - vision or robot control system.

An article by Kaufman [263] explains how some of the above parameters, in particular the voltage, current, and wire feed speed, may be interfaced in a robot welding system. In addition, the following are good sources of information (publishers in parenthesis) about transducers and instrumentation for data acquisition:

- Control and Instrumentation (Morgan-Grampian)
- Industrial and Scientific Instruments (Hanover Press)
- Transducer Technology (Industrial Media)
- What's New in Electronics (Morgan-Grampian)
- What's New in Processing (Morgan-Grampian)

Some of the other parameters, eg wire consumed, arc energy, heat input etc, can be derived from the basic parameters. Most of the sensors produce analogue signals which is conditioned and converted if needed, to digital signals. There are available devices which offer multi-channel monitor for the arc welding process, eg the WI multi-channel digital printing monitor [264] and the OIS portable arc monitoring PAMS-IV system [265].

In the total adaptive control of the welding process, where all relevant data has to be collected and interpreted immediately within a time window. This necessitates a very short time in the "sensor/transducer-feedback" to "decision-making" to "onward-device-control" loop. The author's opinion is that the time should be short enough to typically enable adaptive control over a 0.1mm interval of travel along a weld seam to achieve a sound weld. A super microcomputer or mainframe computer in conjunction with fast-response sensors and data conditioning equipment will be required.
A plausible alternative is to have a less ambitious environment. There will be selective monitoring of the parameters and the adaptive control occurs over a longer period, e.g., after every segment of a weld of say 5mm long. In addition, use a distributed low-cost microcomputer arrangement where each in the system is assigned to a certain task, e.g., one to act as the overall controller, another for driving and controlling geometrical data acquiring sensors, and yet another to collect incoming data for storage and interpretation. This would require some form of a network interlinking all the microcomputers and peripherals was as already discussed in Sect 6-2-1 above.

Some aspects of the control or data collection and interpretation may require fast information exchange, e.g., short sampling rates over a short distance or tracking and adaptive modification of robot path. These fast response would require Direct Memory Access (DMA) techniques ([64,266]). Commercially available add-on boards or standalone data acquisition units make the concept feasible.

6.4.5 Simulated WRAPS Flexible Robotic Welding Cell

To test the whole concept of WRAPS in an integrated environment would require robots, automatic conveyor system and data acquisition equipment, that were interfaced to a supervisory computer setup and controlled by appropriate software. It would be impossible at this stage to use actual manufacturing equipment because it would require up to another 1 to 2 man-years to provide the necessary facilities, and some of the equipment, and especially the data acquisition equipment and sensors, were not available. Instead, a simulation system was devised to prove the concept of WRAPS. As will be explained in Chapter Seven, this concept is being implemented at Loughborough.

The demonstration of the WRAPS concept is based on an integrated robotic welding system that has digital and analogue signals which can be classified as follows:

1) RS232-C serial communication to/from a Programmable
Logic Controller, eg to control a task sequence, or to/from a robot controller, eg to activate a robot program, up- or down-loading of an off-line program, acquiring robot coordinates data from the controller etc. It may also in some cases, require a parallel port interface, eg to a printer having a corresponding port.

ii) Digital ON/OFF signals, eg for controlling the on/off status of equipment, eg through relay switches.

iii) Analogue signals, mostly from output of sensors. Some sensor systems would have analogue-to-digital conversion units built-in to facilitate interfacing and pre-conditioning of signals prior to transfer of data to computer systems. On the contrary, some sensors or other control equipment may require analogue signals to activate it, eg to drive servo-motors, temperature controller, AC Power controller etc.

It can be shown that the WRAPS software is capable of:

i) providing the necessary communication capability and for control for information gathering.

ii) being used as a supervisory cell controller, incorporating expert knowledge-based adaptive control of the welding process.

It was anticipated that a considerable amount of software interfacing and coding would be required, and this was becoming over-demanding in terms of man-power. As a consequence, the author supervised a MSc project [267] to enlist additional assistance in developing some of the WRAPS software for concept demonstration and in
building a simulator for input/output control. The author was able to specify the functions, type of boards and associated equipment required, but was not experienced enough to design electronic circuits. However, the concept of operation is still the author's original, and so is the software written for this demonstration except for those acknowledged in the software.

6.4.5.1 Equipment Used and Setup

The setup for the WRAPS demonstration comprises the following equipment:

- an IBM ATx microcomputer [268] with 1.5 Mbytes RAM, 30 Mbytes hard disk and floppy disks, 2 serial and 2 parallel ports, and etc.

- a Burr-Brown PCI20000 Data Acquisition and Control board [269].

- a custom-designed simulator console.

The scheme of the demonstration is shown in Fig(6-3). The ATx serves as the supervisory controller of the simulated WRAPS cell.

The PCI20000 is comprised of a Carrier board which is piggy-mounted with the required instrumentation modules for the data acquisition or control. The Carrier board, of the type PCI-20041C High Performance with Direct Memory Access (DMA) and a 8Mhz programmable pacer clock for timing use in DMA data transfer, is connected to the ATx via one of the computer long expansion slots. For proving of the WRAPS concept, the instrument used was a 32 digital channel I/O module.

The simulator is connected to the PCI board via a PCI-20058T High Density Termination Panel and suitable cables. The Panel can accommodate up to 48 channels of digital input or output which are divided into groups labelled as GROUP 1, GROUP 2 and GROUP 3. There are 16 channels per group so only 2 groups are used since the instrument module has 32
I/O channels.

To simulate the communication with a robot system which has a typical RS232-C interface, a serial link was made between the Future FX20 and the IBM ATx. The FX20 is assumed to be a robot system. The second serial port of the ATx was made to a DEC LA20 printer, which served as a storage/output device. The parallel port on the IBM ATx was connected to a Kaga Taxan printer, considered as another storage/output device. Fig(6-3) shows the equipment used.

The 32 digital channels from the instrument module was arranged inside the simulator console as follows:

- 8 channels were used as 8 digital input lines. Each line is assumed linked to some peripheral equipment. In this demonstration, the lines were used and configured as follows:
  
  line 1 - status of welding machine power.
  line 2 - status of mains power to robot.
  line 3 - status of shielding gas flow availability.
  line 4 - status of coolant flow to weldgun.
  line 5 - status of robot safety interlock.
  line 6 - availability of component.
  line 7 - status of robot ready to weld.
  line 8 - status of auxiliary device.

- 8 channels as 8 digital output lines. The lines were assumed linked to perform the following functions:
  
  line 1 - on/off control of work positioner.
  line 2 - start/end current data logging.
  line 3 - start/end voltage data logging.
  line 4 - start/end welding speed data logging.
  line 5 - start/end wire feed speed data logging.
  line 6 - turn on/off weld penetration sensor.
<table>
<thead>
<tr>
<th>IBM PC ATx</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line Cell Control Program</td>
<td></td>
</tr>
<tr>
<td>Main Application Code and Data</td>
<td>Knowledge Base</td>
</tr>
<tr>
<td>I/O Ports</td>
<td></td>
</tr>
<tr>
<td>Parallel</td>
<td>RS232C</td>
</tr>
<tr>
<td>Storage Device 1 (Taxan/Kaga Printer)</td>
<td>Storage Device 2 (DEC LA120 Printer)</td>
</tr>
<tr>
<td>Digital I/O</td>
<td>Binary Input</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG(6-3) Configuration of WRAPS Integrated Welding Cell Demonstration.
line 7 - hold/activate robot to stop/perform welding.
line 8 - control of auxiliary device.

- 8 channels were divided into 2 groups of 4-bits binary input values. One group was used to indicate the feedback obtained from a voltage sensor, and another from a current sensor. It is expected that an analogue-to-digital convertor, which is a common device, will be required at the sensor output to produce a binary value for external interpretation. The 4 bits, referenced as BCD-IN, will therefore give a maximum input value of 15. Some sensors, eg the Optocator triangulation laser sensor [24,270] offers both type of control signals, analogue and digital, as standard.

- 8 channels similarly grouped as above but for output binary values to control some external equipment, for the example here, is linked back to control the voltage and current of the arc welding machine. If the equipment is incapable of receiving digital signal, a digital-to-analogue convertor would then be necessary. The maximum output value, for the 4-bits channel or BCD-OUT, will be 15.

These input/output channels are linked to their associated "equipment" as shown in Fig(6-4). The simulator console has Light Emitting Diodes (LEDs) to display the status or binary value of the respective channels or groups. At the moment, input to the system, whether on/off or a binary sequence, are achieved manually via the rocker switches provided. In addition, each input line has a 3-way rotary switch, denoting, "off", "manual" and "system", and therefore it is assumed that the console could be switched over to the "system" mode where the signal status would come directly from the real equipment used.
FIG (6-4) The WRAPS Cell Simulation Console and its Associated Feedback and Control Signals.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single-bit Digital I/O Groups</th>
<th>BCD-In</th>
<th>BCD-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bits Voltage Feedback</td>
<td>Voltage Control</td>
<td></td>
</tr>
<tr>
<td>4-bits Current Feedback</td>
<td>Current Control</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Weld Machine Power Status</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Robot Power Status</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Shield Gas Status</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Weldgun Coolant Status</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Robot Safety Inter-lock</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Component Availability</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Robot Readiness to Weld</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Auxiliary Device Status</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single-bit Digital Outputs</th>
<th>BCD-In</th>
<th>BCD-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digital OUT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Digital IN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BCD OUT</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BCD IN</td>
<td></td>
</tr>
</tbody>
</table>

Simulation Console
To determine the quality of penetration at the joint, a "penetration value" is obtained from a penetration sensor. In this example, the sensor value is requested from the user. In the real situation, the interface would be similar to the voltage and current as described above. This is justified because such sensors capable of producing digital signals for external usage are available, e.g., the weld-pool vibration sensor [260, 271].

6.4.5.2 Software Building for the System

The PCI board system must be initialised before further use and this is performed by software which was supplied PCI-200466_2) and must be linked with the WRAPS application program. In addition, the board device driver must have been previously loaded permanently into resident memory before the functions called will perform the action desired through the PCI board.

Apart from interfacing with the data acquisition and other hardware devices, WRAPS is required to be linked with the KES expert system shell. Apart from some assembler routines, e.g., for the device driver and communication software, all the programming, including those library functions provided for the PCI board and KES, were written in the Lattice "C" language [272]. The device driver for the PCI board could only work in MS-DOS. Hence because of this, the development has to be undertaken with Lattice "C" and in a different operating system, namely MS-DOS version 3.2, as used for the other WRAPS module, i.e., CP/M.

There are 3 main software components:

i) Process monitoring and control, which are in Assembler and "C".

ii) Expert system shell interface and its associated library, written in "C".

iii) Control software for the PCI-20000 board, written in Assembler.
The intention was to develop a WRAPS supervisory system in a multi-tasking environment using Concurrent-DOS-XM. However, the device driver supplied for the PCI-20000 board could not be used without extensive re-write or modifications which even the UK supplier could not contemplate doing in the very near future. Hence, the multi-tasking environment had to be abandoned at this stage.

The embedded expert system with the "C" WRAPS application software was achieved in 3 stages - (i) creating the knowledge base, (ii) writing the WRAPS application program, and (iii) linking the previous 2 together in conjunction with the device drivers for the PCI20000 board.

The simulation demonstration program has to be developed with the "C" large memory model because KES was available only in that model. The PCI20000 software had both the small and large memory models.

(i) Creating the Knowledge Base

The creation of the knowledge base follows the convention as dictated by the KES expert system shell. The knowledge base is in 2 sections, the first contains the attributes and the second the rules extracted from the experts.

The attributes represent the characteristics or features of a problem domain that are important to the decision-making. It is important because the expert system uses them to distinguish one from another and generate the appropriate inferences. There are 2 types of attributes: asserted and inferred. The asserted attributes may be values input by the user or defined explicitly in the "C" program. The inferred attributes are determined with inference to the rules embodied in the knowledge base.

The rules are logical statements that link attributes to each other and enable the expert system to infer a value for an attribute based on the values of other attributes. In the Production Rule inference system, this is in the form of:

Page 219
If antecedent than consequent

The knowledge base may comprise of up to 9 non-mandatory sections of related information which constraints or manipulates domain knowledge. The sections may be of any of:

- constants, text, patterns, types, attributes,
- classes, externals, rules, actions.

Following this, the knowledge base is parsed using "kesp" to turn the base into a format recognised and used by the run-time module called "kesr" for the inferencing and decision-making. Once the knowledge base is tested to be functional, the "actions" section in the base is removed as it is not required in the embedded environment.

(ii) Interface with KES

The application program is written in the conventional way except that it must include the following items to enable it to communicate and use the KES inferencing capabilities:

(a) Header file definition which holds KES function names and data types. The "C" program must include "kes.h".

(b) KES library files which contain the object codes to be linked with the "C" program. Hence, during linking, specify to include "kes.lib".

The "C" program makes appropriate calls to KES to perform specific requirements, eg the parsed knowledge base must initially be loaded to gain access to the knowledge using the KES function labelled as KES_load_kb(), or to obtain an inferred value for an attribute, use KES_obtain(). At the end of the consultation, the memory holding the knowledge base should be released to the computer system, using KES_free_kb(), to be used for other purposes. All the "C" callable KES
functions are found in the KES manuals.

(iii) Linking "C", KES and the Device Drivers

The demonstration program is made by linking the "C" application program, and the object library functions code for KES which are stored in \textit{kes.lib}, and for the PCI20000 in \textit{PCI-20046S_2}. In addition, the library functions for the PCI board also requires \textit{C46_LL.OBJ} (large memory model) to be linked with the application program.

The application program must initialise the PCI hardware using the `sysinit()` and `init()` function. There are arguments required for these functions and their respective usage are described in the PCI20000 User's manual [273]. The library functions requires constants definitions and global data declarations which are stored in \textit{PCIHEAD.H}. This file must be included in the usual "C" compiler directive \texttt{#include} in the application program.

6.4.5.3 Demonstration of the Simulated Cell

It is assumed that a component to be welded has gone through the process planning stages and the conclusion is that the component is robot weldable. A production batch of a certain size, say 50, is to be fabricated. Also it is known that an initial weld procedure is available to start off the fabrication. To demonstrate the WRAPS concept of on-line expert-based control and integration, it is necessary to take a simplistic view that for the particular component to be welded, the only criteria of weld quality is weld penetration. The acceptance level of the weld procedure is at the moment 60% and hence there is scope for the weld procedure to be further enhanced. Any other additional parameters that required to be monitored is considered as an expansion to the basic WRAPS demonstration.

This demonstration merely shows the adaptive change of the welding procedures once every weld has been carried out. It does not propose an on-line self-adaptive system where a shorter all-round feedback and forward control loop is necessary. This will be possible to a certain
extent if the use of state-of-the-art sensors, eg the Odelft Seampilot [256] or Metatorch [257] optical sensors. Also, the expert system must be capable of fast inferencing and provided with real-time input/output control utilities. For real-time adaptivity, the WRAPS cell controller must also be capable of:

i) event-driven information transfer - the state of a weld is affected by several factors. A change of state, for example a sudden lack of weld fusion, may become significant to require that certain control parameters be communicated to control and adjust the process equipment affecting the weld.

ii) task synchronisation - the state of the weld, for example may require a modification to the welding procedure in use. This will require some other task, in this case, the expert system, to supply new information. Control of other tasks, for example, data capture via sensors or weldgun orientation, will be required.

iii) priority interrupts - certain aspects of control of the welding process may be more critical than others. For example, process monitoring equipment, may detect 2 defects - 1 of weld penetration and the other of weld bead centralisation. There should be a facility which will respond to such alerts raised by the monitoring system and suspend or abort task activities in favour of higher priority activities.

A suggestion for future work in Chapter Seven (Sect 7-1 Immediate Future Work) leads the development to this possibility.

For the WRAPS cell demonstration, the expert rules used for the GMAW process, expressed in KES language format, are shown in Fig(6-5). The goal is to achieve, say a set target of minimum 80% weld penetration. This amounts to a minimum binary input value from the "sensor" of (0.8 * 15) or 12.
EXPERIMENTAL Online Control KNOWLEDGE BASE

Author: Kim H Soh 1987

This Knowledge Base creation is based on the KES2.3 Production Rule sub-system inference mechanism available for the IBM-PC.

attributes:

Control parameter: sgl (peaked bead, weld penetration) (question: "Which parameter is used to affect", "the outcome of the weld quality").

Alarm status: sgl (normal, alarm).

Working status: sgl (begin, end).

Volt limit: sgl (not_max, not_min).

Amp limit: sgl (not_max, not_min).

Do_increase_actions: sgl (inc_voltage, inc_current, no_change).

Do_decrease_actions: sgl (dec_voltage, dec_current, no_change).

Weld penetration: sgl (good, fair, bad).

Weld quality: sgl (good, poor, fair, bad).

- The penetration sensor value will be requested from the user.
- Present current is linked as the parameter that controls the operation of the WFS in this example, or in general the apps could be used to control any other equipment. The value is obtained from the BCD input of the simulator console.

\# Present current: real [constraint: Present current ge 0 and Present current le 151.]
If voltage may be an increment or decrement, value based on the Present voltage:

Present voltage: real (constraints: Present voltage \( \geq 0 \) and Present voltage \( \leq 15 \))

Peaky bead: truth (question: "Is appearance of bead peaky?")

Switch1: sgl (on, off) (question: "Is Welding mains").

Switch2: sgl (on, off) (question: "Is power to robot").

Switch3: sgl (on (question: "Yes"), off (question: "No")) (question: "Is shielding gas available").

Switch4: sgl (on (question: "Yes"), off (question: "No")) (question: "Is weld gun coolant available").

Switch5: sgl (on, off) (question: "Is robot interlock").

Switch6: sgl (on (question: "Yes"), off (question: "No")) (question: "Is component ready").

Switch7: sgl (on (question: "Yes"), off (question: "No")) (question: "Is robot ready to weld").

Switch8: sgl (on, off).

All output to equipment are initially OFF:

Out Equipn1: real.
Out Equipn2: real.
Out Equipn3: real.
Out Equipn4: real.
Out Equipn5: real.
Out Equipn6: real.
Out Equipn7: real.
Out Equipn8: real.

Rules:

checkrule1: if Penetration sensor value lt 5 then Weld penetration = bad. endif.

checkrule2: if Penetration sensor value lt 12 and Penetration sensor value ge 5 then Weld penetration = fair. endif.

checkrule3: if Penetration sensor value ge 12 and Penetration sensor value le 15 then Weld penetration = good. endif.

checkrule4: if Weld penetration = fair and Peaky bead = true

FIG(6-5) Cont'd
then Weld quality = bad.
endif.

checkrule5: if Weld penetration = good
   and Peaky bead = false
   then Weld quality = good.
endif.

checkrule6: if Weld penetration = good
   and Peaky bead = true
   then Weld quality = poor.
endif.

checkrule7: if Weld penetration = fair
   and Peaky bead = false
   then Weld quality = fair.
endif.

checkrule8: if Weld penetration = bad
   and Peaky bead = false
   then Weld quality = bad.
endif.

\------------------------------
\* CONTROL RULES *
\------------------------------

cntrlrule1: if Weld quality = good
   then Do_increase_actions = no_change.
      Do_decrease_actions = no_change.
endif.

cntrlrule2: if Present voltage < 15
   then Volt limit = not_maximum.
endif.

cntrlrule3: if Present voltage ≥ 5
   then Volt limit = not_minimum.
endif.

cntrlrule4: if Present current < 15
   then Amp limit = not_maximum.
endif.

cntrlrule5: if Present current ≥ 5
   then Amp limit = not_minimum.
endif.

cntrlrule6: if Weld quality ≠ good
   and Control parameter = peakybead
   and Peaky bead = true
   and Volt limit = not_maximum
   then Do_increase_actions = inc_voltage.
      Do_decrease_actions = no_change.
endif.

cntrlrule7: if Weld quality ≠ good
   and Control parameter = peakybead
   and Peaky bead = false
   and Volt limit = not_minimum
   then Do_increase_actions = inc_voltage.
      Do_decrease_actions = no_change.
endif.
then 
\begin{align*}
\text{Do\_increase\_actions} &= \text{no\_change}, \\
\text{Do\_decrease\_actions} &= \text{dec\_voltage}.
\end{align*}
endif.

\begin{verbatim}
\# First preferred course of action \#
\# when weld penetration is not good \#
\text{cntrlrule1}: \text{if} \\
\begin{align*}
\text{Weld quality} &= \text{good} \\
\text{and Control parameter} &= \text{weld penetration} \\
\text{and Weld penetration} &= \text{good} \\
\text{and Amp limit} &= \text{not maximum}
\end{align*}
\text{then} \quad \text{Do\_increase\_actions} &= \text{inc\_current}, \\
\text{Do\_decrease\_actions} &= \text{no\_change}.
endif.
\end{verbatim}

\begin{verbatim}
\# Second course of preferred action \#
\text{cntrlrule2}: \text{if} \\
\begin{align*}
\text{Weld quality} &= \text{good} \\
\text{and Control parameter} &= \text{weld penetration} \\
\text{and Weld penetration} &= \text{good} \\
\text{and Volt limit} &= \text{not maximum}
\end{align*}
\text{then} \quad \text{Do\_increase\_actions} &= \text{inc\_voltage}, \\
\text{Do\_decrease\_actions} &= \text{no\_change}.
endif.
\end{verbatim}

\begin{verbatim}
\# Peripheral equipment control rules \#
\#***********************
\text{equip\_rule1}: \text{if} \\
\begin{align*}
\text{Switch5} &= \text{on} \\
\text{or Switch1} &= \text{off} \\
\text{or Switch2} &= \text{off} \\
\text{or Switch3} &= \text{off} \\
\text{or Switch4} &= \text{off} \\
\text{or Switch6} &= \text{off} \\
\text{or Switch7} &= \text{off}
\end{align*}
\text{then} \quad \text{Alarm\_status} &= \text{alarm}, \\
\text{Out\_equipn1} &= \text{1}, \\
\text{Working\_status} &= \text{end}.
endif.
\end{verbatim}

\begin{verbatim}
\text{equip\_rule2}: \text{if} \\
\begin{align*}
\text{Switch5} &= \text{off} \\
\text{and Switch1} &= \text{on} \\
\text{and Switch2} &= \text{on} \\
\text{and Switch3} &= \text{on} \\
\text{and Switch4} &= \text{on} \\
\text{and Switch6} &= \text{on} \\
\text{and Switch7} &= \text{on}
\end{align*}
\text{then} \quad \text{Alarm\_status} &= \text{normal}, \\
\text{Working\_status} &= \text{begin}.
endif.
\end{verbatim}

\begin{verbatim}
\# If work is to begin and welding power is turned ON \#
\# then turn the following devices ON : \\
\# Out\_equipn1 = work positioner \# \\
\# Out\_equipn2 = current data logging \#
\end{verbatim}

FIG(6-5) Cont'd
\* Out_equipn3 = voltage data logging *\\
\* Out_equipn4 = wfs data logging  *\\
\* Out_equipn5 = robot weld speed log  *\\
\* Out_equipn6 = penetration sensor  *\\
\* Out_equipn7 = activate robot  *\\
equip_rule_3: if Working_status = begin and Switch1 = on
then
Out_equipn1 = 1.
Out_equipn2 = 1.
Out_equipn3 = 1.
Out_equipn4 = 1.
Out_equipn5 = 1.
Out_equipn6 = 1.
Out_equipn7 = 0.
endif.

Auxiliary device

auxEquip_rule_1: if SwitchB = on
then
Out_equip8 = 1.
endif.
auxEquip_rule_2: if SwitchB = off
then
Out_equip8 = 0.
endif.

FIG(6-5) Cont'd
When the embedded KES system is executed, the system checks if the cell is to start welding. The first rule in the knowledge is worked on, moving in a top-down fashion to achieve the various attribute values as required. First, the interlock to the robot is checked. If it is indicated that the door into the robot work area is opened, i.e. switch 5 on the simulator console is "open", a warning message is sent to the screen and the program remains in an infinite loop, or until the user prefers to quit the program by selecting "Exit" or "close" the safety door using the input switch 5. Similarly, it also checks for the availability of shielding gas, water coolant to the weldgun, the existence of a component for welding, and the readiness of the robot for welding.

Next, the robot is to commence welding. The welding machine mains power is checked first, if it has been turned on manually, it turns all the other necessary devices, viz the output lines 1, 2, 3, 4, 5, 6 and 7 to initiate the required data logging and signals to the robot and welding equipment to begin welding. In the real situation, there will have to be a defined sequence of switching all these signals ON, e.g. the data logging equipment would not start recording until there is welding in operation and also to be turned them off when the welding stops. During welding, it continuously "monitor" the state of the welding arc, gas and coolant flow. If any of these fail, the welding process is interrupted and alarm is sent out. The amount of penetration is dependent on the rule of voltage regulation, as laid out in the rules. In the demonstration, its value is entered by the user via the keyboard giving a penetration representation along the weld. From this value, a representative amount of penetration percentage is evaluated. If the value is less than 12, then the voltage to the welding machine is increased via BCD-OUT. In an actual situation, the voltage is not the only parameter that could be modified to control the penetration.

The monitoring continues until a penetration value of 12 or greater is reached. Thereafter, the user is informed, and the updated set of weld procedure may be saved for future use.
CHAPTER SEVEN
7.1 Introduction to Chapter

This chapter examines further research and development work on WRAPS to make it a more usable software tool for laboratory and commercial applications. It also describes the implementation of the integrated welding cell which was simulated in Chapter Six. Some of the recommendations to improve the usability of the WRAPS software described herein are being implemented at time of writing.

7.2 Further Research and Development Work

Further research and development work suggested for the continuation of the initial WRAPS concept are divided into 3 areas: immediate, short-term and long-term. Each area is described in the following sections.

Immediate Work

The "Perform" instruction for WRAPS corresponds to execution of a block of instructions, and was un-conditional, i.e. the sub-sequence was always executed. It should now be extended so that the execution of the instruction is conditional on (i) an mathematical evaluation, (ii) external signals or interrupts, and (iii) internal flags.

In the programming exercise of Chapter Five, the spot welding macro used was inflexible and caters only for a specific application. Different joints will require different welding patterns. Macros could be created with some "intelligence" built-in, eg self-adaptive macros that could accommodate a wide range of spot welding patterns or indeed macros for any other application tasks, eg multi-pass welding. Macros
have the potential to save substantial labour and time in the generation of repeated and complicated robot tasks, eg welding along a contoured surface or water-jet pattern-cutting on plates.

In future, a decision-making mechanism, possibly involving an expert system, to assist in the selection of a particular application macro from a macro library, should be made available. A special software module should also be devised to assist in the generation of additional programs.

Kinematic and preferably dynamic simulation should also now be implemented. And so is the replay of an off-line program by simulation. The simulation carried out in Chapter Six is not only a demonstration of the WRAPS integrating concept. A simulator of that nature is in fact important for the quasi real-time testing of the off-line program. The off-line program should not only be statically simulated, i.e. carried out by replaying the program through by graphical simulation on a computer screen. It should be dynamically simulated to include tests on its links with peripheral devices. Robotic systems presently used for manufacturing are frequently equipped with some sensors for feedback control, and used to control other equipment and processes. WRAPS with a more developed simulator console will play an important role in checking this dynamism.

A new research grant (SERC Research Contract No OR/E/03946) has been awarded to commercialise WRAPS and to further implement the concept of WRAPS, in collaboration with the Welding Institute (WI) and the Marchwood Research Laboratory of the Central Electricity Generating Board (Southampton, UK). This should lead to the research and development of WRAPS into a controller of an adaptable flexible integrated robotic welding cell. The ideal cell being one when (i) welding products from the computer-aided design screen are transferred to the fabrication line without delay and (ii) the welding fabrication could be turned on or off in relation with other production facilities from a remote terminal in an office, and being able to adaptively select production procedures to fabricate economically a batch component quantity as low as 1.
Such a cell is being developed at Loughborough University. The cell, shown in Fig(7-1), comprises of 3 robot systems - a CML T3, a SILVER-REED ARY4, and a FANUC S100, which are interlinked to a custom-modified WESTWOOD automatic conveyor system [274] and a custom-designed sensor manipulator. Functions of the individual components of the cell are as follows:

The basis of the WESTWOOD automatic conveyor system is a 750 x 200 mm pallet which has a matrix of 7.94 mm diameter holes at 25.4 mm centres and accurate to within 0.051 mm. The pallet has 4 shot bolt locations on the underside, and is individually identified by a binary number system, using 7 aperture holes on the side of the pallet. The binary identification is read by a series of proximity switches. The matrix of holes on the pallet provide the location and clamping of jigs/fixtures which holds the component to be presented to the welding robot.

The conveyor is of a 2-tier roller design and which is divided into sections designated as follows:

i) Component Loading/Unloading Station
This is where the component to be welded is loaded/unloaded by the CML T3 robot which is equipped with grippers. The T3 receives a signal from this station as to when to perform, and an identification code as to which pallet (and hence component) is involved. This indirectly selects the appropriate T3 program to be executed.

ii) Pre-welding Inspection Station
The component to be welded is first inspected for geometrical data. Based on this data, the necessary welding conditions are selected. The supervisory computer creates a circular-linked list, as shown in Fig(7-2). This is similar to the flow of pallets on the conveyor system, and the necessary welding data is stored onto an appropriate slot on the linked list so that, when the particular pallet appears at
FIG(7-1) The WRAPS Flexible Integrated Robotic Welding Cell.
FIG(7-2) Organisation of Welding Control Data at the Welding Station.
the welding station, the right set of welding conditions would be used.

The geometrical data is to be collected via a triangulation laser sensor, called the Optocator, which is mounted as the end-effector on one of the manipulators. The suitability of the Optocator for inspection work has been covered by the author in a MPhil thesis [24].

iii) Welding Station

This is where the component is welded by the FANUC S100 robot. In-process data is gathered here, eg (1) robot path and welding arc information via the robot controller system through external interfacing and interrogation, (2) shielding gas flow rate via flowmeter, and (3) weld penetration information via a front-face penetration sensor. The penetration sensor which may be implemented is described in an article [260] for TIG welding applications.

iv) Post-welding Inspection Station

The finished weld will be inspected for quality, manually or automatically. Manual inspection will, eg refer to weld appearance and surface finish or deferred non-destructive testing to inspect for porosity or cracks.

Automatic inspection will be performed, eg to obtain information of the bead geometry, using a stripe laser sensor mounted at the end of another manipulator system. The concept and importance in having an automatic inspection system within a cell of an integrated entity is now supported in a recently published article [275].
Chapter Seven

At each of the above 4 stations are 4 pneumatically controlled shot bolts that raise and accurately locate each pallet at their respective stations.

WRAPS with enhanced facilities as described in this Chapter, and in conjunction with the KES expert system, is to be the supervisory and control system of the cell. An immediate objective of the cell is the closed-loop optimisation of welding procedures based on their measured performance. This entails selection of welding procedures to be incorporated within the WRAPS off-line program, monitor the performance of the procedure via sensors and data acquisition facilities, and then consequentially using expert knowledge, improving the procedure based on this evidence. This flow of activity which is to be achieved via a cell being set up at Loughborough University is illustrated in Fig(7-3). The cell will automatically shutdown if it detects out-of-control conditions.

It has already been described in Chapter Six Sect 6-4-5-3 during the simulated WRAPS cell control demonstration that an expert system task controller will require improved input/output control and fast rule inferencing capability. First and foremost, the controller should be further developed in order that real-time data be directly fed into WRAPS, and conversely, output data from WRAPS be communicated directly to physical devices. Programmable Logic Controllers (PLC) emulation and actual PLC programming capability, used in the logical control of process events, should also be incorporated into WRAPS. The state-of-the-art PLC, or Programmable Controller as it is also known, is a flexible device-integration system which enables reception and transmission of operational signals required to operate a variety of interlinked devices to direct the logic and flow of a process. A PLC system also offers simple adjustment of programming routines and communication data and status reports to higher level controllers such as the microcomputer [276,277].

In an embedded "C" language environment, the expert system shell could be easily provided with real-time input/output facilities via assembly code routines. To achieve fast rule inferencing will require considerably more software development, possible only in association
FIG(7-3) Flow of Activities of WRAPS Flexible Integrated Robotic Welding Cell.
Chapter Seven

with the expert system shell developer. However, an alternative is available through the use of another expert system called STIMULUS [278]. Its design features were aimed at providing real-time processing for engineering control, monitoring, and diagnosis. The system is an embedded construction within a "C" language program, similar to the KES expert system shell but lacking some of its features, for example multiple inferring engines, and ease of use and of rulebase construction. STIMULUS at the time of writing was only capable of supporting production rules. In theory, because of the flexibility offered by KES and STIMULUS due to the use of "C" and embedding, it would be possible to combine the 2 systems together to provide an extremely powerful expert system based real-time task supervisor. It has been demonstrated to the author that STIMULUS is capable of arriving at a decision based on some 10000 if-else rules in about 10 seconds in the SUN workstation UNIX environment.

The expert system approach adopted for WRAPS is plausible, as recent published articles and reports of new products associated with real-time control of processes have shown a similar trend. It has been reported [279,280] that expert systems are being developed and used for process control, CAD, industrial logistics and a host of other areas of prime interest to system integrators. The use of KES in such areas is also reported.

In one of the German national research programme reported [280], there is an application involving expert system techniques for the control of power distribution. Management of load involves monitoring and understanding the prevailing load demand conditions and prediction of future trends. This prediction is based on the operator's experience and intuition.

In another industrial application, an application program called VIOLET [281] is coupled with an expert system to capture the expertise of maintenance engineers and facilitate the automation of the interpretation of vibration data which is obtained directly from any machine that could be connected to a vibration measurement system. From the same company there is also SYNERGIST and ANNIE. The former is a fault diagnosis and schematic capture which provides automated test
procedures for the manual testing of board-level circuits. SYNERGIST uses the electronically captured schematic of a circuit to derive and perform test procedures automatically. ANNIE is specially designed for engineering applications, such as mechanical health-monitoring, control and alarm monitoring. It can read values directly into the expert system for automatic interpretation of the rules. Last but not least, there are 5 products called PICON, M1, K1, S1 and MAINTEX, being developed and marketed by Scientific Computers [282] which are designed for various on-line monitoring on various makes of computers, and in the case of PICON it is claimed it can be used to communicate with process control systems and PLCs via almost any protocol.

In a UK market survey of expert system shells carried out by the author in mid-1987, it was found that most were used for advisory or consultative work, thereby limiting its potential on the factory floor. The author ascertained that there is a high potential for expert system shells to be used widely in industry if they were provided with fast-response input/output capability to enable the on-line control of process equipment, logic, and database storage, updating and retrieval. It is possible therefore that the WRAPS On-line Module be further developed for commercial exploitation as a standalone package to be used in conjunction with commercial data acquisition and control boards.

In view of WRAPS commercial potential, it is important to further enhance the ease-of-use and user-friendliness of the system. These 2 aspects are important, as is also pointed out in a recent report [283] that 55% of the life cycle costs of setting up and maintaining a computer system is attributed to the non-productive time spent on learning and managing the system.

**Short-term Work**

WRAPS should be developed continuously and transferred to emerging microcomputers to take advantage of improved facilities. Extra on-board addressable RAM is one and concurrent task execution is another. The former is particularly of importance because there was just sufficient RAM available to create the 49-points off-line robot program for the
programming example in Chapter Five. Extra RAM and task concurrency is becoming possible through new hardware processors, eg Intel 80386 and Motorola 68030 microchips, and new operating systems [284] that make full use of such powerful chips, eg Digital Research Concurrent DOS-XM [285,286] and Microsoft OS2 [287,288]. On existing microcomputers that uses IBM PC/DS and MS-DOS there is a 640 kbytes barrier for available RAM on the IBM PC architecture. The actual directly addressable memory is 1 Mbytes, but only 640 kbytes is allocated to RAM and the other 360 kbytes are for read-only memory, video buffers, and permanent storage of programs and device drivers. At the time of writing, Concurrent DOS-XM (expanded memory) was already available whilst OS2 was being developed for the new generation of IBM Personal Systems (PS/2) computers. Concurrent DOS-XM, for version 5.1 at time of writing, supports up to 8 Mbytes of RAM, multi-tasking, multi-user access, and permits most existing IBM PC/DS, MS-DOS and CP/M-86 applications to run without modifications. The applications that do not work are those which uses direct addressing to the microcomputer hardware rather than going through the operating system.

An alternative method to access more RAM is to use a memory paging technique (or bank switching as it is also called) that allows a microcomputer to access memory outside the 1 Mbyte. This additional memory is known as expanded memory which requires a specially designed plug-in card (called the expanded memory card) [289] and a software device driver (called the expanded memory manager) which is loaded into a protected area of memory and acts as the interface between applications, the host computer operating system, and the expanded memory card.

Essentially, the expanded memory system functions in a way similar to the reading of a book by a human. "Pages" of memory, defined in 16 kbytes size and up to a maximum of 8 Mbytes, are switched into view by manipulating hardware registers on the card in the same way that humans read a book by turning to the appropriate page. The paging technique follows a procedure defined by the Lotus/Intel/Microsoft (LIM) called Expanded Memory Specification (EMS), Fig(7-4), or by another superset defined by AST/Quadram/Ashton-Tate called the Enhanced Expanded Memory Specifications (EEMS) [290-293]. However, the latest development is the
FIG(7-4) Principle of Expanded Memory in an IBM PC-AT.

FIG(7-5) Use of Expanded Memory [E.M.S. 4.0] to Multi-task and Share Data in a WRAPS Application.
EMS 4.0 [294-296] and was being released in the US at the time of writing this thesis. EMS 4.0 is intended as the unifying specification which supercedes all previous expanded memory specifications. In addition to an enhanced expanded memory addressable size of 32 Mbytes, EMS 4.0 offers multi-tasking. These 2 features are very important to the WRAPS concept because larger and more complicated objects could be modelled and with increased simulation capability. Multi-tasking in expanded memory would ensure viable real-time control of the flow of welding automation data.

This approach, to take advantage of EMS 4.0, is illustrated in Fig(7-5). Process data is obtained from sensors and continuously updated in a buffer area located in expanded memory. Several concurrently executed programs could access the information for various purposes which is based on the information received.

Floppy disk storage media is not reliable in industrial environment. Alternative program and data storage media, eg bubble memory cassettes, compact laser discs, should be considered.

WRAPS is expandable to a wider involvement in CIM and a larger computer memory and faster computational speed is required. Enhanced graphics over the current Future FX20 is highly desirable. As the power of microcomputing has increased, it has become possible to house the equivalent of a minicomputer on a desktop. And the trend continues with more and more power to boost the graphics and processing capabilities, occupying less and less space and of lower cost [297,298].

Workstations provide a possible upgrade path for WRAPS as their costs are dropping rapidly, eg consider the lower end of the range offered by APOLLO, Whitechapel, Apple MacIntosh II, and SUN [299-301]. The Atari Abaq standalone workstation [302] perhaps might be the lowest yet in terms of costs. In applications involving high-grade interactive graphics, such as for drafting, finite-element modelling, solid modelling, CAM, AI and robotics, workstations were identified as the least costly way of providing the greatest amount of processing power [303,304].
Chapter Seven

If WRAPS is used in a situation where it serves as a welding cell controller, concurrent multi-tasking becomes a necessity to facilitate simultaneous execution of cell functions. An example of a concurrent multi-tasking operating system is the OS-9 operating system [305,306]. This will be aided by real-time computing power, for instance through the use of Reduced Instruction Set Computers (RISC) [307,308] or parallel-processing computers called transputers [309]. Current transputers are available as add-on development boards [310-311] at reasonably low cost but is expected to be available as stand-alone units in the near future.

An alternative is to use a real-time multi-tasking operating system kernel. A real-time kernel manages the concurrent execution of various functions, or tasks, that the software must perform and it provides synchronisation and communication between tasks in the system. Some examples of real-time kernels are AMX [312], FlexOS [313], and VTRX [314]. The adoption of a standard operating system kernel is essential for effectiveness as a development tool, and the Japanese Sigma Project is leading towards such a system through the TRON kernel [314]. The TRON kernel comprises two primary kernel standards, an ITRON kernel targeted for industrial applications, and a BTRON kernel for business machines. TRON kernels pre-package essential software functions, particularly those needed for real-time operations.

The weld procedure selection built into WRAPS has scope for further development. It could be linked to interrogate the specialised databases held at the WI. WI has produced a software package called Weldquest [315] which is a computerised method for setting up or selecting weld procedures. It allows flexible searching and selective display of stores data in verbal or mnemonic form. Weldquest is suitable only for mainframe computers but WI has also produced a PC version called WELDSPECS which is more suitably integrated with WRAPS. WELDSPECS was formerly called Micro-Weld-Arc-Quest and was developed by Powers [316] at Liverpool University.

A modem link via the telephone to access the databases is technically feasible. This is a facility which the WI should expand so that its subscribers could dial direct to consult and retrieve...
appropriate welding procedures if available. This means that a weld procedure transmission is required electronically. It is essential that a standard messaging protocol be implemented, eg following the formats of the standardisation initiative being achieved by MAP. MAP, which is a 7-layer, broad-band, token bus based communication specification for the factory environment (see Chapter Six Sect 6-2-1), is concerned with enabling dissimilar industrial computers to communicate with each other. The transmission of the weld procedure, in particular, should be based around the application layer called Manufacturing Message Service or MMS [317,318]. MAP is becoming more and more acceptable in industry and a similar protocol which is devised for the transmission of weld procedure would find itself more readily acceptable to the intended users.

The importance of a network system, and in particular a system with the capability of MAP or MiniMAP, within the WRAPS concept as was discussed in Chapter Six Sect 6-2-1 should be implemented.

**Long-term Work**

Presently, WRAPS is virtually restricted to a search of the best procedure from those available in a database, but takes no account of the eventual performance of the procedure. WRAPS should be able to apply expert rules and heuristics to both procedure and performance data to provide continuous optimisation of procedures held in the database, based on the latest state of knowledge of the procedures. It should also be able to predict procedures for joints or categories of joints for which no empirical data is available.

The parameters used in the weld procedure search, i.e. the welding process, wire size, joint type, welding position, parent material type and thickness range, and fillet size requirements, should be enhanced to include the root run process. Further, the search should be made flexible so that the user could define preferred search parameter(s). One possibility is to link the weld procedure database storage/retrieval with software packages such as WELDSPECS or WELDERQUAL from the WI, the former is a planned WRAPS development.
WRAPS should further be developed to apply such procedure optimisation in "real-time" during fabrication of components in batches. The low cost of WRAPS will make it readily viable for many potential batch fabricators to employ the system for effective computer control of the welding system. The measured performance of the system with the selected welding procedure can provide expert modification of the procedure for each successive assembly to be welded.

In turn, if suitable data can be collected in real-time, on-line control of the process may be achieved. This requires that the robot program held in the robot controller memory can be modified in real-time, a facility now available on most commercial robot systems. It is also necessary to develop an intelligent interface to the robot welding system based on artificial intelligence techniques for real-time data monitoring and adjustment. A conceptual structure of the advance WRAPS system is shown in Fig(7-6).

All the information about the weld joint variability, together with procedure and performance measures could be applied to statistical modelling. Expert statistical modelling rules and heuristics applied to the collected data for categories of welding procedure will enable automatic generation of process control algorithms. These models could be enhanced as more related data is collected until an acceptance algorithm is achieved.

The use of expert systems in the welding cell could eventually lead to a system which can self-learn and perform self-modifying functions to update and expand on the welding algorithms used. This could largely be achievable once some of the limitations on current expert system shells are removed. These basic restrictions are (a) the lack of external input/output recognition and control, (b) inability to update and/or acquire additional knowledge interactively in order that the knowledge base could be re-organised automatically to accommodate the dynamic requirements of the application domain. As already pointed out in the previous section, there are now indications that the state-of-the-art expert system technology are moving towards this trend.

Although expert systems have now been used by a growing number of
FIG(7-6) The Total Advanced WRAPS Concept.
firms, much of its application success depends to a very large extent on the human factor. This is supported by an article [319] reporting that several companies which have to use appropriate approaches to overcome any reticence on the part of the human experts. There are also problems when an attempt is made to seek knowledge from a "non-expert" person. Knowledge acquisition is seen as the biggest bottleneck in the development of expert systems. The interested reader should refer to [320] for a review, criteria for use, and literature sources for all the principle methods employed for knowledge acquisition. Some useful tools to assist the knowledge engineer to perform its knowledge elicitation task hence become desirable. In the UK, one such tool called the Liverpool University Knowledge Elicitation System (LUKES) [321] is being developed.

On the CAD front, an interface should be made to other CAD packages to allow external data to be imported into WRAPS. On a one-to-one data transfer between WRAPS and another CAD package, a dedicated two-way format processor, similar to the CML T3 robot program post-processor described in Chapter Four Sect 4.4, should suffice. On a wide applications area, it is essential that the route adopted adheres to some standard of data exchange. In relation to WRAPS, most of these data exchange is geometry data. At present, there is no universal standard in the definition of geometry data, but the Initial Graphics Exchange Specification (IGES) [322,323] appears to be widely accepted and some form of standardisation is inevitable. Looking further into the 1990s, it is expected that the Product Data Exchange Specifications (PDES) or some other similar specification, will eventually supersedes IGES. In addition to geometry data, PDES [324] is planned to support the entire life cycle requirements of a component, including non-geometry information such as manufacturing features, tolerance specifications, material properties and surface finish specifications.
8.1 Introduction to Chapter

This chapter discusses and summarises the conclusions and results of the research.

8.2 Conclusion

This research can be considered as successful and has yielded the following achievements:

(i) Identified the requirement of a low-cost easy-to-use off-line programming system for robotic arc welding and developed a working prototype WRAPS which is of practical usage to the batch welding fabrication industry.

If the requirement is for off-line programming, on comparison of software costs, WRAPS is expected to have a cost advantage of 2 : 1 with ROPS (Cincinnati Milacron Ltd) and 10 : 1 with GRASP (BYG Systems Ltd).

The core of WRAPS is "neutral" and may also be applied to other processes such as assembly and material handling.

(ii) Identified the integrating concept for the welding process and expansion of a flexible robotic welding cell into the realms of CIM, and has incorporated the various state-of-the-art software applications, eg expert systems,
graphics programming, etc. into the integration.

Some simulation packages, as was reported in Chapter Two, can offer off-line robot programming but have no process data integration. WRAPS can incorporate process data directly from a database.

Usability to the initial expert system shell KES, in terms of user interface and input/output communication with the real world, has been enhanced through embedded software and external functions specially written to this effect.

(iii) WRAPS software is easy to use through the development of a WIMP-type user interface at a time when similar interfaces such as GEM or WINDOWS on microcomputers were not available.

(iv) The research is continuing and the initial work forms the basis of future work which is directed towards the integration of the various aspects of the welding process as indicated in the discussions of Chapter Seven. This is possible through a new funding from the SERC (Research Contract OR/E/05285) and is in collaboration with the Welding Institute (Abington, UK) and the Marchwood Research Laboratory of the Central Generating Electricity Board (Southampton, UK).

The WRAPS off-line programming concept has been proved to be practical in Chapter Five, and the low-cost equipment involved will make the package a viable proposition to many batch fabrication manufacturers, or other manufacturers intending to use off-line programming for their robots since WRAPS can be adapted for other applications. Also, many of the WRAPS integration concepts, including the use of expert knowledge through KES to control the welding cell,
have been tested and validated using the simulator in Chapter Six.

Discussions are in progress to put WRAPS into industrial trials for off-line programming in a batch robotic welding environment. The involvement of the Welding Institute in this respect will bring about rapid introduction of low-cost off-line programming and integration, and it will also enable many of the integrating features of WRAPS to be possible, eg through the interface with their software products, such as WELDSPECS, and their immense database of welding procedures and welding knowledge. The expert system part of the WRAPS package will enable this knowledge to be formalised and implemented for the most effective utilisation.

Finally, the author is encouraged to mention that this research has provided the backbone to assist in the expansion of robotic welding, and particularly in robotic off-line programming where more companies have since been reported to have demonstrated their systems, for example from HITACHI, FANUC and NACHI [325]. The very concept of WRAPS is the key to a cost effective implementation of future robotic welding systems, where the whole welding process, including planning and production, need to be supervised and controlled. WRAPS is an implemented prototype of a process control system (called SCADA) based on the current industrial trend in the manufacturing industries [326,327]. SCADA is an abbreviation for Supervisory Control and Data Acquisition. Off-line programming and especially graphical off-line programming, complete adaptive control and interface of the welding robot (or indeed any other applications robot) to a CIM system represent the leading edge of technology. This view is supported by Castner (Edison Welding Institute) in his article on the current and future states of robotic welding in the USA [328] and also by Christensen (Danish Welding Institute) in the Danish welding industry [220].
REFERENCES


References


22 Braggins D: "In the Kingdom of the Blind" The Production Engineer Apr 1984 pp45-46.


References


37 Clarke S et al: "Low Cost Seam Tracker for TIG Welding" WI Seminar Exploiting Robots in Arc Welded Fabrication 6-7 Nov 1984 Coventry, UK.


41 Williams E: "Sure Grasp of Robot Design" Financial Times Mon 17 Sep 1984 Technology Section.


45 McAuto Simulation/Off-line Programming System - PLACE, BUILD, COMMAND and ADJUST Modules. Product of McDonnel Douglas Automation Co, St Louis, MO, USA.


49 GRASP. Product of BYG Systems Ltd, Unit 1, Highfields Science Park, University Boulevard, Nottingham NG7 20P England.

Page R-3
References


52 "Intelligent Workcells Expand with Software Advances" Robotics World Mar 1986 Vol 4 No 3 pp30-33.


54 Lawrence A: "Does CADCAM Pay?" CADCAM International Aug 1987 Vol 6 No 8 pp14-16.

55 "Bureau Houses CAD/CAM" CADCAM International Sep 1984


60 Wyss C R: "Planning a Computerized Measurement System" Byte Apr 1984 pp114-123.


References


74 "Robot Off-line Programming for Cincinnati Milacron Industrial Robot with Acrumatic Version 4.0 Robot Control" Pub No 10-IR-84204 Cincinnati Milacron Ltd See address in Appendix (A1).


81 "YAMAHA Robot YK8000 Series Users Manual" Yamaha Motor Co Ltd. See Address in Appendix (A1).

82 "MSP FORTH Teach and Replay" Issue No 1 1987, Universal Machines Intelligence Ltd. See address in Appendix (A1).


References


References


114 "If It Moves - Sell It" CADCAM International Feb 1988 Vol 7 No 2 pp34-36.


121 Encotel Systems Ltd, 7 Imperial Way, Croydon Airport Industrial Estate, Croydon, CR0 4RR, England.

125 Flum T : "Learning to Program in C" Prentice-Hill New Jersey, USA 1983.
References


Page R-9
References


163 EIA RS-232/20mA Current Loop Interface (101-4Q), Product of Black Box Corp, Box 12800 Pittsburgh PA 15241 USA.


168 "ASEA System Communication Language - ASCOL" ASEA Doc YL 710-121E.


172 Computer Aided Manufacturing-International Inc, 611 Ryan Plaza Drive, Suite 1107, Arlington, Texas 76011 USA.

173 "IRDATA - General Structure and Type of Records" Draft VDI2863 W Germany 1982.


References


181. "Model-B3KA Robot Controller Introduction Doc No M100-B3KA-ED" and "Instructions for Connection and Operation of Robot Controller Model 33 KA with Host Computer" Silver-Reed / Silver-Selko Ltd. See address in Appendix(A1).


183. "EIA/Current Loop Interface 101-40" A product of BLACKBOX Corp Pittsburgh, USA.


References


207 "Robot systems. Product of Cincinnati Milacron Ltd. See address in Appendix(A1).

208 Teledyne Readco UPC25 Positioner. Product of Teledyne Readco, York, Pennsylvania, USA.

209 Transmig R500+ MIG welding machine with AWP TF2-4R wire feeder. Product of BOC Arc Equipment, UK and Export Dept, Patricia Way, Pysons Rd, Broadstairs, Kent CT10 2LE UK.

210 "STR-LINK IIA Data Cassette Recorder (unbuffered) - Users Manual" Doc No 207-DMS-040 Electronic Processors Inc Colorado, USA.
References


References


231 Helis T : "Europe Demonstrates MAP Commitment" Automation Apr 1987 Vol 23 No 3 pp31-34.


235 GEnet Local Area Network System. Product of GE Fanuc Automation Ltd (Also General Electric Automation Europe), Low March, London Road, Daventry, Northamptonshire NN11 4B3 England. See brochure "Industrial Communications" Doc No GEI-1118. See also article in [247].


246 Several articles about Programmable Controllers in Mechanical Engineering Vol 108 No 4 Apr 1986, and in Automation Jan 1987 Vol 23 No 1.


253 ActionFile Software. Product of The Microsolution Ltd Park Farm House, Heythrop, Chipping Norton, Oxon OX7 5TW UK.

254 Hornet Project Control Software. Product of Claremont Controls Ltd Albert House, Rothbury, Morpeth, Northumberland NE65 7SR UK.


256 VersaCAD, a CAD product of VersaCAD Corporation 7372 Prince Drive, Huntington Beach, California 92647 USA.

257 Seampilot Optical Profile Sensor System, a Product of Oldelft, BV Optische Industrie ‘De Oude Delft’, P.O. Box 72, 2600 MD Delft, Holland.

258 MetaTorch Sensors, Product of Meta Machines Ltd. 9 Blacklands Way, Abingdon Business Park, Abingdon, Oxford OX14 1DY, UK.

259 Roe J : "An Introduction to In-cycle Gauging" HELEX 1982 Metrology Conference.


261 Salter R J and Dearn R T : "A Practical Front Face Penetration Control System for TIG Welding" Paper 38 Preprints 2nd Int Conf on Developments In Automated and Robotic Welding 17-19 Nov 1987 London UK. The CEBG Marchwood Research laboratory (Southampton, UK) have been actively involved in penetration sensors for TIG and MIG welding.

References


265 Portable Arc Monitoring System - PAMS-VI. Product of OIS Engineering Ltd, Bessemer Way, Hartreys Ind Est, St Yarnemouth, Norfolk NR31 OLX, UK.


268 IBM ATx microcomputer. IBM (UK) Ltd, National Enquiry Centre, 414, Chiswick High Road, London W4 5TF, UK.

269 Burr-Brown PCI-20000 Data Acquisition Board. Product of Burr-Brown Corp Tuscon, Arizona, USA.

270 Optocator Sensor. Product of Selective Electronics Inc (Belcom), P.O. Box 250 9-433 25 Partille, Sweden. Distributed in the UK by Planer Products Ltd, Windmill Road, Sunbury-on-Thames, Middlesex TW16 7HD England.


272 Lattice MS-DOS "C" Compiler Version 3. Product of Lattice Inc, 22W600 Butterfield Road, Glen Ellyn, IL 60137, USA.


275 "Gauging the Future for Inspection" Engineers' Digest Apr 1987 Vol 48 No 4 pp12-17.

276 "Intelligent Conveyors - With Brains to Get from A to B" Material Handling News Feb 1988 No 362 pp54-60.


Page R-16
References


282 PICON, M1, K1, S1 and MAINTEX. Products being developed/marketed by Scientific Computers Ltd, Victoria Road, Burgess Hill, W Sussex RH15 9LW England.


290 "Enhanced Expanded Memory Specifications (Enhanced EMS) Software Interface Version 3.2" ASI Research, Quadram Inc and Ashton-Tate, USA Jul 1986.


292 Brzeski M: "Writing Big Programs with the RAMpage!" .EXE May 1986 Vol 1 Issue 3 pp22-24.


References


299 Reed D: "War of the Workstations" Informatics Dec 1987 Vol 8 No 12 pp44-49.


312 ANX Multitasking Executive, Product of KADAK Products Ltd 206-1845 W Broadway, Vancouver, BC Canada V6J 1Y5.


References


323 Neill T; "IGES: Breaking Down the Comms Barrier" CADCAM International Dec 1985 Vol 4 No 12 pp41-44.


## APPENDIX (A1)

### Robot Manufacturers/Suppliers Mentioned In Thesis

<table>
<thead>
<tr>
<th>Robot Name</th>
<th>UK Supplier</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CML T3</td>
<td>Cincinnati Milacron (UK) Technologies Ltd, P0 Box 505, Kingsbury Road, Birmingham B24 0GU, England.</td>
<td>Cincinnati Milacron, 4701 Marburg Avenue, Cincinnati, Ohio 45209, USA.</td>
</tr>
<tr>
<td>FANUC</td>
<td>600 FANUC Robotics Ltd, Hythe Station Road, Colchester, Essex CO2 8JP, England.</td>
<td>FANUC Ltd, Oshino-mura, Yamanashi Prefecture 401-05, Japan.</td>
</tr>
<tr>
<td>ASEA</td>
<td>ESAB-LINCOLN Automation Ltd, Gunnel Wood Road, Stevenage, Herts SG1 2B4 England.</td>
<td>ESAB AB, Box 106, S-695 01 Laxa, Sweden.</td>
</tr>
<tr>
<td>PROMETHEUS</td>
<td>Sands Technology, 22 Cheddars Lane, Cambridge CB5 BLD, England.</td>
<td>-ditto-</td>
</tr>
<tr>
<td>TOMKAT</td>
<td>Unimation (Europe) Ltd, Unit C, Stafford Park 18, Telford, Shropshire TF3 3AX, England.</td>
<td>Unimation Incorp, Shelter Rock Lane, Danbury, Connecticut 06810, USA.</td>
</tr>
<tr>
<td>PUMA</td>
<td>UMI Ltd, UMI House, 9-15 St James Road, Surbiton, Surrey KT6 4QN, England.</td>
<td>-ditto-</td>
</tr>
</tbody>
</table>
An Example of a Search Operation for a Horizontal Fillet Weld Using a TOMKAT Robot

When the weldgun moves to point 'P', the search routine is initiated.

1. The wire touch sensor starts, and the weldgun lowers from point 'P' until it comes in contact with the lower base metal. The sensor takes as the position data at the point where the wire is in contact with the base metal.

2. The weldgun moves up X mm, in this case 10 mm, from the wire contact point at fast speed. Various of X values are used via a subroutine to meet different joint types and dispersion of dimensions.

3. The wire touch sensor is started again to move the weldgun from step 2 above, for detecting the upper base material. The weldgun moves along the positive Y axis direction until the wire comes into contact with the upper base metal. At this point, the system records the position data.

4. Finally the weldgun is moved backwards in the negative Y axis direction and downwards towards start point 'Ps'. The distance moved is an optimum value specified in the subroutine.
EXSYS APPLICATIONS

The following is a partial list of applications that have been created with EXSYS. This list is based on a survey conducted of registered EXSYS users. The list of registered users represents only a fraction of the total number of EXSYS users. In addition, only a small percentage of the users contacted returned the questionnaire and many of those that did return the questionnaire asked that their application be kept confidential. Based on these factors, we believe that well over 3000 applications have been developed with EXSYS or are in the process of being developed using EXSYS.

Diagnose telecommunication difficulties
Personal tax advisor
Telephone system configurator
Select/recommend library reference materials
Commercial loan credit analysis
Weed identification
Gas Turbine troubleshooting
Training in gas turbines
Material Selection by engineers
Recommend documentation to computer users
Product selection system
Chemical process diagnosing and troubleshooting
Structural damage assessment
Analysis of simulation results in bank product planning
Medical diagnosis
Diagnostic advisor for pulp bleaching
Choosing an executor for trusts
Choosing a living or testamentary trust
Evaluation of stock purchases
Evaluation of commodities purchases
Industrial training
Equipment troubleshooting
Real time process control
Manufacturing resource planning aid
Production scheduling
Service network assistant
Troubleshooting airplane starting systems
Cost/benefit assistant
Forecast snowfall accumulation
Forecast severe convecting weather
Customer water quality analysis
Invention patentability expert
Solid waste disposal management assistant
Power plant boiler tube failure identification
Radar mode design workstations
Assist service desk in troubleshooting applications problems
Machining advisor for grinding, milling, turning
Advice on single family home purchase
Advice on stock and commodities trading
Analysis of X-rays
Training new financial planners
Advertising copy development
Market segmentation and positioning
Marketing advisor for process control systems
Computer system configurator
Quick proposal estimator
Hardware and software selection
Customer assistance in selecting types of investments
Soil acidity analysis
Space shuttle payload on-orbit analysis
Futures, stocks, and options trading
Credit control system
MIS decision support system
DP Production support system
Product development support system
Advising on choices for new technology
Determine correct mixture of propellant ingredients
Aid for isolating failing chips
Finding phases present in super alloys
Submarine approach officer training
Fertilizer, climate, and soil interaction
Fertilizer recommendations
Analysis of soil site characteristics
Psychiatric interview
Real time troubleshooting for wastewater process control
Alarm management system
Conveyor for selecting, sizing, and writing parts list
Call screening to interview users with application problems
Detailed analysis of hardware and software problems
System to identify feasible rehabilitation strategies
Detailed design for asphalt concrete pavement
Implementation planning assistant
Hardware sizing assistant
Career development
Hardware failure analysis
Classification of software programs
Selection of non-materials in aerospace applications
Relay diagnosis
Federal contract management
Irrigation and pest control management
Conservation tillage equipment selector
Broker syndication planner
Estimating construction costs
Determining best shipping documentation and routes
Grading of graft vs. host disease
Failure analysis of a component
Materials selection for specialized component parts
Rating for substandard life insurance
When to perform physical audit
Decision support for correct testing by auditing
Problem diagnosis for local area networks
Problem diagnosis for printers on a SNA network
Create standard loan documents based on characteristics
Commercial loan documentation check list
Software development risk analysis
Software vendor risk analysis
Evaluation of multi-family housing projects
Real estate appraisal
Mortgage credit analysis
Underwriting assistance
Closing and issuance assistance
Classification of data from satellites
Student financial aid eligibility
Estimate employee's potential retirement salary
Sales order analysis
Equipment fault diagnosis
Assistance in search for part numbers
Soil characterization and utilization
Advise nursing students on care of patients
Salary planning
Application sizing based on similar existing applications
Computer modeling support
Advisor on choosing soy bean varieties
Pest management and soil organism interaction analysis
Assist in identification of rare antibodies
Diagnosis of sports-related injuries
Epidemiology expert system
Toxicity of laboratory chemicals

Augment expertise of resource manager
Crop management and irrigation simulation
Advisor on design of new magnetic components
Assist in compiling tax planning ideas
Account business assessment
Strategic marketing and planning aid
Causal mode of account marketing
Local area network selection
System to prepare process estimates
Strategic alternatives for a fragmented industry
Portfolio construction
Teaching mineral and rock identification
PC hardware and software configurator
Aid in salmon stocking rates, species selection
Salmon disease diagnosis and treatment
Correct selection of cost codes
Pavement rehabilitation
Pavement performance diagnosis
Fault diagnosis for electronic hardware
Medical decision making
Underwriting guidance for line underwriters
Aid for financial futures traders
PC configuration
Interpretation of statistical quality control data
Real estate site selection
Product performance troubleshooting for salesmen
Select pension types
Network operations system diagnosis
Risk assessment of error or fraud in financial statements
Selection of solvents for chemical compounds
Customer service advisor for problem resolution
Lime recommendation system
Papayas management system
Hazardous chemical ranking
Aid in diagnosis of computer console messages
Line diagnosis and fault detection
Data communications troubleshooting
Perform hematological diagnosis
Assist new users of DOS
Software system diagnosis model
Geographic information system analysis aid
Tactical battle management

california
intelligence
knowledge engineering
products and services
912 Powell Street #3
San Francisco, Calif. 94108
(415) 391-4846

Page A3-2
APPENDIX (A4)

First Estimate of Computer Memory Requirements to run a Basic WRAPS Program

The initial estimation of the off-line robot program size was based on the CML T3 robot because it was the only one available in the welding laboratory for off-line programming tests. The estimate was approximated as follows:

(a) Estimated Size of a Robot Program Point in Memory

The robot program shall consist of:

<table>
<thead>
<tr>
<th>Description</th>
<th>Bytes Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>function name, character strings of up to 20 long (plus &quot;C&quot; string terminator)</td>
<td>21</td>
</tr>
<tr>
<td>function name qualifier, eg ON or OFF</td>
<td>4</td>
</tr>
<tr>
<td>robot positional data (X,Y,Z,D,E,R)</td>
<td>6 x 8 = 48</td>
</tr>
<tr>
<td>allow a comment line per point (max 80 chars)</td>
<td>81</td>
</tr>
</tbody>
</table>

Bytes per robot point = 154

It was felt that most robotic welding applications in practice will not be required to use up to the maximum of 999 points per sequence or the maximum of 64 sequences allowed per robot program for the T3 robot. A reasonable number was to have the maximum points per sequence (J E Middle experience) of 150 and the number of sequences of 10.

Hence, approximate bytes required for prototype robot program = 150 x 154 x 10 = 231 Kbytes

(b) Memory Space to Hold Model of the Environment

At this stage, assume same number of points per object for robot arm, weldgun and component allocated. A number to permit reasonably complicated objects to be modelled was 300 per object. It was expected to be working with double precision for calculations, so there...
Appendix (A4)

are 24 bytes (3 x 8) per coordinate point of X, Y and Z.

Hence, bytes required for the 3 objects =

\[ 24 \times 300 \times 3 = 21.6 \text{ Kbytes} \]

Two sets of data would be required, one set to hold the actual dimensions of an object, and another to hold its transformed data for display on the screen. Therefore, for modelling purposes the bytes required was:

\[ 21.6 \times 2 = 43.2 \text{ Kbytes} \]

(c) Memory Space for Welding Procedure

It was expected that a weld procedure will remain in memory at any one time. Others required will be called up from the floppy disk. It was expected that the weld procedure will occupy up to a full A4 page of characters. Allowing for a 80 columns and 66 lines down, the amount of information approximate to \( 80 \times 66 = 5.3 \text{ Kbytes} \). But only about a fifth of the information need to be stored directly into memory since the rest can be treated as comments. Hence, memory space required to store 1 weld procedure is approximate 1 Kbytes.

(d) Code for the Main Program

Most microcomputers applications in 1983/4/5 rarely exceeded the 64Kbytes barrier. Because the author had seen WRAPS as a potential commercial software package, it was expected that the program size will approach this barrier. So assume maximum required memory for code = 64 Kbytes.

(e) Total Approximate Size Required for WRAPS Development

Total approximate number of bytes required:

\[ 231 + 43.2 + 1 + 64 = \text{ approximately } 339 \text{ Kbytes.} \]
Appendix (A5)

APPENDIX (A5)

Descriptions of software functions for the Modelling Module
(In alphabetical order)

ADJUSVIEW
Primary purpose to provide the following options for controlling
view of object using the cursor keys. To change view parameters by user
input see VIEWPARM.
a) to zoom in, zoom out, shift left, shift right, shift
up, shift down, the object in the view-port.
b) to rotate or tilt the modelled object about the view
centre point. Enable viewing of object from top, bottom, front, back or from any intermediate view
angle.

CHANGSIZ
Function to change the size of a segment/object. The user is asked
to input a size reduction/enlargement factor. The data to be affected on
may be in current computer memory or from a disk file. If it is from the
latter, the data is read in via READSEQS and the transformed data is
then saved via SAVEDATA.

COMBINE
Contains the options for combining segments created into a complete
object. Only the option for joining end-to-end is available. Calls
COMJOIN and SHOWMENU.

COMJOIN
Function for combining several segments into an overall object. The
user is asked how many segments are to be combined consecutively. The
segments data are read from files via READSEQS and the combined object
is saved to a defined filename via SAVEDATA.

COMPOS
Contains future options for the creation of complicated composite
objects, eg objects of intersection, pre-defined weld joints etc. At the
moment only calls SHOWMENU.

CREATCONE
Function to create a conic object, all the coordinates are created
here.

CREATCUB
Creates the coordinates of two faces of a cuboid. Calls MAKSOLID to
form link up the coordinates in a defined sequence to form the overall
object.

CREATCYL
Function to create a cylinder. Only the coordinates of the end
faces are created. Calls MAKSOLID to link up points to form the
cylinder.

CREATLIN
Function to create two coordinate points to form a straight line.
The user is asked for the origin and length of the required line.

CREATSPH
Function for creating a 3D sphere. The user is asked for the origin
and radius of the sphere, and also the amount of data points that is
used to form the sphere. A sphere with a lot of data points will look
more realistic, but requires more computer memory.

CREATSPN
Contains options for creating lines and splines. Only straight line
option is used at the moment. Calls CREATLIN and SHOWMENU.

DELFILE
Function to delete a disk file.
FILEUSE
Contains options for disk file utilities, eg to delete or rename a file etc. Calls RENFILE, DELFILE, PRNFILE and SHOWMENU.

FRAME
Display module title, static labels, and border frame in the alpha mode.

GENSETUP
1) Prints title of module
2) Install dynamic memory for message strings, graphics arrays and menus.
3) Install general items, eg memory space for icon arrays. Also initialises data, eg the maximum number of coordinate points and maximum number of definable welding location points for each object model allowed in the Module.
4) Reads in the addresses that points to individual blocks of text information in the 'Help' file.

GETDATA
Reads in data of an object from a file. Calls READOBJ that actually reads in the data, and then perform the necessary transformation to display it on the view-port.

GRAPHAI
A blank overlay module to allow for future expansion.

GRAPHICM
This is the main module which the supervisory MODEL will call to enter graphics mode from the alpha mode. Holds all the options which are available in the graphics mode. Calls ADJUSTVIEW, GETDATA, SHOWPTS, VIEWFARM, MAKEWELD, MAKEFACE, GRAPHAI, PVEWDAT and REDRAN.

IREQFACE
Function to create a polyhedra with a irregularly shaped cross-section. One 'face' is first defined by the user who enters the individual initialises to form the 'face' from a reference point. A second 'face' is projected over by the required length to form the other end of the polyhedra. Calls MAKSOLID to link up points of the two 'faces' to form an object.

MAKEFACE
Contains routines for creating 2D 'faces' that could be extended to a 3D object. Also contains functions to create pre-defined constant cross-sections, eg T-sections, L-sections etc.

MAKESEG
Contains the options for creating the various types of segments: polyhedra, polyprism, lines, etc. Calls POLYHED, POLYPRSM, CREATSPN, REVSOLID, COMPOSIT and SHOWMENU.

MAKEWELD
This function enables welding points to be created within the modelled object. The weld points may be defined (i) at a coordinate point, (ii) mid-point between two specified coordinates, and (iii) off-set between two specified coordinate points.

MAKSOLID
Links up the coordinates of two 'faces' in a pre-determined sequence to form an overall object. The coordinates for the two 'faces' are created by their appropriate routine, eg CREATCYL for a cylinder, or CREATCUB for a cuboid.

MODEL
1) Grabs all the available memory in the computer for use with the Modelling module.
2) Acts as the supervisory program of the module, enabling entry to the various optional functions, and exit from the module. It resides permanently in memory whilst the other are overlays. Calls GENSETUP, SCRSTAT, SHOWMENU, MAKESEG, SHOWNDATA, SAVEDATA, TRANSFORM, GRAPHICM, COMBINE, UTILITY.
On sign-on, the module will be in alpha mode. To enter graphics mode, the overlay GRAPHICM is called.

OBJNAME
Function to give a name to a segment/object that is currently in memory.

POLYHEX
Contains options for creating polyhedras which may be of regular or irregular cross-section. Regular cross-section polyhedras is called within POLYHEX which in turn calls CREATCUB, CREATECYL and CREATECNE. For irregular cross-section, it calls IREGFACE. Also calls SHOWMENU.

POLYPRXM
Function to create a polyprism. The user is asked to input all the required coordinate values, and in the desired sequence, to form the polyprism.

PRNFILE
Function to list a file to a printer.

PRVEWDAT
List all the current view control parameters to a printer.

READOBJ
Function to read in the individual segments of an object from a disk file for transformation or combining purposes.

READSEQS
Function to read in the actual coordinate data of an object but does not perform any transformation on the data. Also the number of segments used to form the object is also determined from the file. Individual segments of an object are read in from another function, READSEQS.

REDRAW
Clears the view-port and re-draws the current work scene.

RENFILE
Function to rename a disk file name.

REVSOHLD
Contains options for creating spheroids and solids of revolution. Only spheroids is implemented at the moment. Calls CREATSPH and SHOWMENU.

ROTATE
Function to transform an object by rotation about the x, y or z axis. The user is asked for the axis and the number of degrees the object is to be rotated about. The data to be rotated may be in current memory or be read from a file. In the latter case, the segment/object data is read in via READSEQS and when the data is transformed, saved back to file via SAVEDATA. In either case, the transformed data becomes the current object in memory and is displayed in the view-port.

SAVEDATA
Function to save coordinate data in computer memory to a disk file. It also save information such as disk filename, number of segment units contained in the file and any defined welding points in the modelled segment/object.

SCRSTAT
Displays the graphics screen layout, draws border lines, and display sign-on message.

SHOWDATA
Display the coordinate data of the current object in memory on the screen. Also can read and display data from a disk file.

SHOWMENU
Displays options pertinent to the respective overlay modules that calls it. Calls FRAME to display module name and border frame. Also displays name of current segment/object in memory if any. Returns user’s selected option to the supervisory module for action.
SHOWPTS
This overlay function is called to display two types of coordinate data; i) object coordinates and ii) welding points coordinates. A number is shown at the point whose coordinates are displayed. The number of the coordinates shown may be stepped through one at a time, or jump through by an interval, decreasing or increasing in value, until the last point. When a carriage return is pressed, this function is immediately exited to the caller function.

TRANSFOM
Contains the options for transformation routines. The transformations available are translation, rotation, and changing of size of an object. There is also an option to initialise the working memory area that is used to create the objects. Calls TRANSLAT, ROTATE, CHANGSIZ, ZEROASK and SHOWMENU.

TRANSLAT
Function to translate an object in 3D space. The user is asked for the amount of translation in the X, Y and Z directions. The data to be translated may be in current memory or called up from a file. In the latter case, READSEQS will be called to read in the segment/object to be translated. When done calls SAVEDATA to write new transformed data back into file. In either case, the new transformed data becomes the current segment/object in memory and will be displayed on the view-port.

UTILITY
Contains options for utilities; Calls FILEUTIL and OBJNAME.

VIEWPARM
Contains routines for changing view parameters by user input of values. Enables view distance, view reference point, centre of projection, view-plane centre of projection, and the view-plane normal vector to be changed. There is an option that displays all the parameters.

ZEROASK
Function to initialise the global memory storage area for the object, eg to clear memory to load a new object into the system. This function may be called with a 'flag' to ask user to confirm initialisation before proceeding. Use in some modules initialise memory straight away.
APPENDIX (Ab)

Descriptions of Software Functions for the Programming Module
(In alphabetical order)

ACTCALIB
Function to perform the actual calibration procedure as discussed in the main text of the thesis. It aligns the work positioner (or other objects to be calibrated in the workplace) according to the averaged measured values of the reference object.

ADJUSTSUN
This function provides the control to orientate the weldgun to a required angle in terms of the robot pitch, yaw and roll definitions. The orientation increments and total amount orientated are displayed continuously. The actual robot orientation angles are also automatically updated. A <RETURN> key is expected to terminate this function.

ADJUSTPIC
Function to provide the following options for controlling the view of the work environment by using the keyboard cursor keys. To change view parameters through user input see option under EXAMINE.
(a) Zoom in/out, and shift left/right/up/down.
(b) Tilt or rotate the objects about the view center point.
(c) Change the perspective of the picture.

ADJUSTVIEW
Primary purpose to provide the following options for controlling view of work scene:
(a) Calls ADJUSTPIC to zoom in, zoom out, shift left, shift right, shift up, shift down, or change the perspective of the picture in the view-port.
(b) Reads in view parameters saved in a disk file, eg to restore to initial startup view from ORIGVIEW.DAT, or to any other desired view previously saved using SAVEPARM.
(c) Calls REVOLVIEW to rotate or tilt the whole work environment about the view centre point. Enable to view work environment from top, bottom, front, back or from any intermediate view angle.
(d) Calls SAVEPARM to save a particular view to file.

ANIMATE
Contains future expansion options to provide graphical animation (play-back) of the off-line program.

ARCTIME
Function to sum up the total arc-on time encountered, up to the very last point taught in the off-line program.

BESTDATA
Function to read and display the optimal or next best weld procedure that was retrieved via WELDTYPE.

CALIBRAT
Contains various options used in the calibration of the work cell environment. The options are:
(a) Loads or frees a reference object used for the calibration purposes.
(b) Calls ACTCALIB to perform the actual calibration procedure as discussed in the main text of thesis.

CLEXPERT
Contains options in retrieving and setting welding data into off-line program. It does the following functions:
(a) Get empirical welding parameters from the user.
(b) Calls WELDTYPE to select the type of weld joint for the database search.
(c) Calls MAKEWELD to create weld points on the object.
model.

(d) Calls SETWELD to set the welding parameters into the off-line program. It automatically sets the weldgun orientation as specified in a weld procedure retrieved by the search operation in calling WELDTYPE. There is also an option to set the voltage and wire feed speed rate to be used in the off-line program.

(e) Save the current weld procedure in computer memory into a disk file.

(f) Calls BESTDATA to display the current best weld procedure that was retrieved from the database search.

CREATNEW

Future expansion options for the creation of new objects to be added to the default work environment which comprises of a robot, a work positioner and a component. For example, a unit vector used in displaying the orientation of the weldgun could be 'cloned' and the new vector object used for other purposes, such as targets for robot homing in.

CYCDIST

Function to sum up the total distance travelled by the robot, up to the very last point taught in the off-line program.

CYCTIME

Function to sum up the total cycle time, up to the very last point taught in the off-line program.

DACSTAB

Contains routines for inspecting and/or changing of robot digital-to-analogue-convertors (DAC) schedules. This contains the welding voltage and wire feed speed for the welding process.

DRAWARM

Displays the robot arm when the flag is set within the software. The status of this flag is toggled via UTILITY.

ENviron

Contains options for the adjustments of objects displayed in the work environment. It has the following options:

(a) Calls CREATNEW to load new objects in addition to the default members (the T3 robot, work positioner and a component).

(b) Rotate and tilt control of the work surface of the work positioner. The work positioner must be set 'ON' first using the appropriate option within SETTOOLS.

(c) Translation and rotation of the work positioner relative to the robot model. This usually affects the component which may be mounted on it.

(d) Translation and rotation of the component that is located on top of the work positioner.

(e) Replacing the work positioner or component being worked on with another. This calls GETWORK.

(f) Contains the option to enable work cell calibration to be performed. Calls CALIBRAT to achieve this.

EXAMDIST

Contains routines to evaluate the distance between two component coordinate points or two weld points, or the distance from a component or weld point to the robot TCP.

EXAMINE

Contains options for examining various parameters. Calls ROBOFUNC, VELOCTAB, DACSTAB, VIEWPARM, EXAMDIST, FLAGSTAB, VARSTAB, and TOOLDIM. There is also an option to display the current status of the work positioner.

FILEUTIL

Provide facilities to interact with the operating system of the host computer, eg to delete a disk file, rename a disk file, and show the directories of files. Calls SHOWFILE.

FLAGSTAB

Contains routines for the inspection and changing of 'flag' status. A 'flag' is a condition variable that has two states: ON or OFF.
GENSETUP

1) Prints title of module
2) Install dynamic memory for message strings, graphics arrays and menus.
3) Install general items, eg memory space for icon arrays. Also initialise data, eg the maximum number of coordinate points and maximum number of definable welding location points for each object model allowed in the Module.

GETDATA

Function to read an object from a disk file into the system. It frees the memory held for the previous object before re-allocating new memory for the new object.

GETWORK

Function to read an object into the Programming Module. It calls GETDATA to read in the coordinate and segment data and then perform the necessary transformation to render the object suitable for display in the view-port.

GODIRECT

This function moves the weldgun directly to a specified coordinate given by its relative number. The argument passed to it determines whether it is going to move the weldgun to an object point or to a welding point previously defined within the object in question. The software will open a "Ask" window to request the user to input the point number. The move to coordinate point may be an object point or welding point defined in the object. The orientation of the weldgun is preserved, only the coordinates are shifted. The robot actual coordinates are also automatically updated.

GOHOME

Returns the weldgun to the robot base location and default orientation. Robot point data automatically updated.

INITSYS

Primary purpose to initialise work scene from data stored in a file called SETWORLD.DAT.

a) Locate equipment relative to each other.
b) Set up robot initial base position.
c) Assign default "incremental" values for view control and equipment adjustment.
d) Assign initial process data for the following items: robot function, robot velocity, wire feed speed, welding voltage, tool dimension, variables, flags, output and input status.
e) Initial view system parameters: view distance, view reference point, view up directions, centre of projection, view plane normal vectors, and centre of projection view plane.
f) Robot data tables: velocity, welding voltage, wire feed speed, tool dimension, variables, flags.
g) Transforms and position all equipment to their respective defined positions in the work scene.
h) Keep a copy of the original view parameters in ORIGVIEW.DAT.

MACROS

Called via MOVEGUN where sub-sequences are created by macro software functions. A macro is different from a relocatable sub-sequence created by READSEQ although they produces sub-sequences as a result of their use. See main text for explanation of differences. The sub-sequence is created interactively with the user.

MAKEWELD

This function enables welding points to be created within the modelled object. The weld points may be defined (i) at a coordinate point, (ii) mid-point between two specified coordinates, and (iii) off-set between two specified coordinate points.
MOVEGUN

This overlay serves to control the motion of the weldgun. From a set of menu options, it calls one of the following overlays, depending on the option selected: SHIFTGUN, GODIRECT, ADJUSTGUN, GOHOME and MACROS. GODIRECT is called with an entry argument to determine whether it is to go directly to an object point or a welding point defined in the object.

PRINTTAB

List out to printer the robot data tables, which includes: robot functions, welding voltages, wire speed feeds, tool dimensions, variables, and flags. Used for quick reference of parameters available in module.

PROGPTS

Programming of the robot points in the off-line program. It provides the following functions:

a) Save each robot point in a tree-like structure by dynamic memory allocation. When a point is required, a new record of the "C" structure is created. Then all the conditions pertaining to this point (eg point number, TCP) are saved into this new record. The address of this new record is assigned to the future pointer of the previous robot point saved. Checks for out of memory and for non-programmable robot functions, eg BASE and PERFORM sequence numbers which have already been used.

If robot function is a close loop instruction, the software asks for the point to end loop to. It then searches through the previously saved tree-like program to see if there is such a point point to close loop to otherwise it echo error message. An efficient recursive "C" search function is used.

b) Show robot points currently resident in RAM. Can show last point saved or show all the points in a sequence.

c) Checks if a particular sequence has been completely programmed. Automatically move on to program a sequence defined in a PERFORM function but which has not been programmed.

d) Calls PROGUTIL.

PROGRAM

1) Grabs all the available memory in the computer for use with the Programming module.

2) Acts as the supervisory program of the module, enabling entry to the various optional functions, and exit from the module. It resides permanently in memory whilst the others are overlays. Calls GENSETUP, READALL, INITSYS, REDRAW, SETTOOLS, SIMULATE, ADJUSTVIEW, PROGPTS, UTILITY, MOVEGUN, CLEXPERT, EXAMINE, ENVIRON and REALTIME.

PROGUTIL

Provides the utility functions to PROGPTS:

a) inspect status of work equipment, input/output signals, and robot sequences being programmed. If a particular sequence has been completed, it is shown "OK" and if it is being worked on "CURRENT".

b) Initialise and free all robot program points held in RAM.

c) Calls PRINTTAB, SAVEPROG, SAVESEQ, and READSEQ.

PRVIEWDAT

List all the current view control parameters to a printer.

READALL

1) Reads in coordinates data of objects from files previously created with the Modelling Module. There is one default item each of the weldgun, work manipulator and a component. Also read in are data for the robot axis symbol and a vector symbol. The former is a
Appendix (A6)

3D representation of the robot axis system which is continuously display at the top right hand corner of the view port, whilst the latter is a vector which may be called up to show the location and orientation of the weldgun. If the number of objects data points exceeds the maximum specified in GENSETUP the Module is exited and error messaged. Similarly, the required size of memory to hold definable welding points are also installed.

Two sets of memory are installed for each modelled object. One set is to store the actual coordinates of the object while the other is to hold the transformed version of the data for display purposes.

If there are welding points defined in the component data, these are also read in once after the arrays to hold them are installed.

READPARM
Read in view parameters from a disk file that was saved earlier using SAVEPARM, to restore the displayed image to previous view settings.

READSEQ
Called via PROGUTIL when a relocatable sub-sequence is to be inserted into a main sequence. It asks for the name of the disk file that contains the relocatable sub-sequence and flags an error if none is found. Once a sub-sequence has been inserted, it asks the user where the "End_Path" point is to be for the inserted sequence.

REALTIME
Contains future options to link up directly with peripheral devices. At the moment, this interface is achieved via the On-line Module.

REDRAW
Clears the view-port and re-draws the current work scene.

REVOLVEW
Contains options to rotate or tilt the work environment about the view centre point in order to obtain a view from any angle. These are all performed via pre-defined cursor keys. The user can specify a default rotate or tilt interval per key stroke. In addition, the whole working environment could be shifted left, right, up or down to obtain the best view conditions.

ROBOFUNC
Contains routines to inspect or change the robot function that is to be used currently for the off-line program.

SAVEPARM
Saves all the view control parameters to a disk file. It asks for the name of the file to be saved. The parameters are view distance, view reference point, view up directions, centre of projection, view plane normal vectors, and centre of projection view plane.

SAVEPROG
Called via PROGUTIL to save the RAM resident robot program to a disk file. It checks if program is complete before saving to disk.

SAVESEQ
Called via PROGUTIL to save a sub-sequence as a relocatable sub-sequence to disk. It asks for the sub-sequence number to be saved and flags an error if such a sequence does not been programmed.

SCRSTAT
a) Displays the graphics screen layout; draws border lines, view-port, command menu area, display area for robot TCP coordinates and weld parameters; display icon of current type of weld to be performed.

b) Displays values of all current parameters specified above.
SETTOOLS

a) Display current status of tools: welding machine power, wire feed (to initiate arc ON), table positioner, input/output status.
b) Permits setting of the tools status i.e. to ON or OFF.

SETWELD

Function to automatically set the weldgun orientation relative to the weld seam as specified in the weld procedure that was retrieved via WELDTYPE.

SHIFTGUN

This overlay is called to control the 3-D space movement of the weldgun. It actually moves the TCP, shown as a flashing target, in the X, Y and Z directions of the robot. The current move increment and the total moved since, are shown for each axis. The actual robot coordinates are also updated. When destination is achieved, pressing the <RETURN> key will terminate this function.

SHOWFILE

Main purpose is to list an ASCII file on disk to the computer screen or to the printer. When this overlay function is called, the Programming Module exits to the non-graphics mode where the menu options are displayed.

SHOWMARK

When called, this overlay function displays the origin of the robot base point TCP or the global base reference point. It draws a 3-D vector at the location.

SHOWPTS

This overlay function is called display two types of coordinate data: i) object coordinates and ii) welding points coordinates. A number is shown at the point whose coordinates are displayed. The number of the coordinates shown may be stepped through one at a time, or jump by an interval either decreasing or increasing in value until the last point. When a carriage return is pressed, this function is immediately exited to the caller function.

SIMULATE

a) Display options available for use in simulation. This is only conceptually implemented to show completeness. Has "options" for eg playing a simulated program sequence backwards or forwards, one step at a time or continuously from the beginning till the end.
b) Evaluates program cycle efficiency from the programmed sequence(s). Calls CYCTIME and ARCTIME.
c) Evaluates robot cycle times from the programmed sequence(s). Calls CYCTIME.
d) Evaluates robot distances moved. Calls CYCDIST.

TOOLDIM

Contains routines to inspect and/or change the values of tool dimensions used.

UTILITY

Contains options for utilities:

a) Calls PREWDAT.
b) Toggle functions - for changing the status of various things, eg to switch on or off the display the position and orientation of the welding gun vector. Others sub-options are not yet implemented.
c) Calls FILEUTIL.
d) Restart the current session of work and frees all dynamically allocated memory. Calls READALL and INITSYS.
e) Calls SHOWPTS and SHOWMARK.

VARSTAB

Contains routines to inspect and/or changing the values of variables used.
VELOCTAB
Contains routines for inspecting and/or changing the values of robot velocities used.

VIEWPARM
Contains routines for changing view parameters by user input of values. Enables modification of view distance, view reference point, centre of projection, view-plane centre of projection, and the view-plane normal vector. There is an option that displays all the view parameters at a glance.

WELDTYPE
Displays types of welding joints as candidates for the database search. Performs search through history files to determine availability of weld procedure for the type of joint specified. If it finds a suitable procedure, it assigns the name of the disk file containing the procedure to the system which could then be displayed via BESTDATA. If no procedure is available, it calls WHATNEXT to determine the next action required from the user.

WHATNEXT
Function to prompt the user on the next action to be taken if no weld procedure was located using WELDTYPE. The user could choose (i) Reject job for manual welding, (ii) load establish empirical procedure, (iii) predict weld procedure by using manual input parameters, or (iv) do nothing.
The Cincinnati Milacron Limited T3 Robot Character Set

The supported character set is derived by adding 80 (Hex) to the standard ASCII set.

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII (Hex)</th>
<th>Character</th>
<th>ASCII (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>B0</td>
<td>K</td>
<td>C8</td>
</tr>
<tr>
<td>1</td>
<td>B1</td>
<td>L</td>
<td>CC</td>
</tr>
<tr>
<td>2</td>
<td>B2</td>
<td>M</td>
<td>CD</td>
</tr>
<tr>
<td>3</td>
<td>B3</td>
<td>N</td>
<td>CE</td>
</tr>
<tr>
<td>4</td>
<td>B4</td>
<td>O</td>
<td>CF</td>
</tr>
<tr>
<td>5</td>
<td>B5</td>
<td>P</td>
<td>D0</td>
</tr>
<tr>
<td>6</td>
<td>B6</td>
<td>Q</td>
<td>D1</td>
</tr>
<tr>
<td>7</td>
<td>B7</td>
<td>R</td>
<td>D2</td>
</tr>
<tr>
<td>8</td>
<td>B8</td>
<td>S</td>
<td>D3</td>
</tr>
<tr>
<td>9</td>
<td>B9</td>
<td>T</td>
<td>D4</td>
</tr>
<tr>
<td>A</td>
<td>C1</td>
<td>U</td>
<td>D5</td>
</tr>
<tr>
<td>B</td>
<td>C2</td>
<td>V</td>
<td>D6</td>
</tr>
<tr>
<td>C</td>
<td>C3</td>
<td>W</td>
<td>D7</td>
</tr>
<tr>
<td>D</td>
<td>C4</td>
<td>X</td>
<td>D8</td>
</tr>
<tr>
<td>E</td>
<td>C5</td>
<td>Y</td>
<td>D9</td>
</tr>
<tr>
<td>F</td>
<td>C6</td>
<td>Z</td>
<td>DA</td>
</tr>
<tr>
<td>G</td>
<td>C7</td>
<td>SPACE</td>
<td>AO</td>
</tr>
<tr>
<td>H</td>
<td>C8</td>
<td></td>
<td>AE</td>
</tr>
<tr>
<td>I</td>
<td>C9</td>
<td>+</td>
<td>AB</td>
</tr>
<tr>
<td>J</td>
<td>CA</td>
<td>-</td>
<td>AD</td>
</tr>
</tbody>
</table>
Work on standardization of the interface for offline-programming of industrial robots in Germany began approximately four years ago. The work is based on the existing standards in the field of NC-technology. These are:

- ISO 840/DIN 66024 NC-Code
- ISO 1056-1058/DIN 66025
- ISO 4343/DIN 66215 CLDATA
- DIN 66267 DNC Protocol

An interface similar to the NC-CLDATA-code has been defined in the draft standard VDI 2863. The IRDATA-interface (Industrial Robot-DATA) is defined open. Extensions to the existing instruction set for user specific requirements are possible.

In VDI 2863 part 1, the instruction and program structure had been defined, as well as the representation of IRDATA-code and data transfer between programming system and robot control based on the DIN 66267 DNC-proposal. An alphanumeric representation has been chosen. These standard should enable the robot user to easily combine his specific robot systems with available programming systems.

4 General Structure of IRDATA-Code

This standard does not focus to a particular type of controller (e.g. 16 bit). If there are any restrictions related to the possible range of values (Sects. 4.1.3, 4.2.2 and 4.2.3), they apply only to the implementation purposes. This increases the portability of IRDATA-code but should not imply a particular data format.

4.1 Structure of IRDATA-Code

4.1.1 IRDATA-Code

Each IRDATA-coded program consists of a sequence of records, terminated by semicolon. The records are specified in Chap. 6.

4.1.2 Record

Each record consists of a sequence of words. These words are separated by commas.

* Extract from “VDI-Standard 2863, Part 1 (1963)” reproduced by kind permission of the publisher, VDI-Verlag, Düsseldorf. For applications, the latest version of the standard is essential and obtainable from Beuth-Verlag, Burggrafen-Strasse 4-10, 1000 Berlin 30.
The controller independent programming system as shown in Fig. 1 is able to compile user programs into IRDATA-code. In addition, the interchange of test information in IRDATA-code is possible.

4.1.3 Word
Each word consists of a sequence of characters. It can be of the types
- INTEGER
- UNSIGNED INTEGER
- REAL
- STRING
The range of integer values according to 32 bits is:
- 2147483647 ... + 2147483647
The range of unsigned integer is:
0 ... 4294967295
The range of real values according to 32 bits is:
1.7E38 ... + 1.7E38
These ranges are only minimum values for each IRDATA implementation but there are no regulations according to the precision of real values.

4.2 Composition of Records
4.2.1 Representation of Words
The representation of the words W1 to Wn of a specific record is as follows:
W1 serial record number (unsigned integer)
W2 instruction code, composed of type and code number (integer)
W3 to Wn arguments according to the instruction code
4.2.2 Serial Record Number
The word W1 of a record is representing the serial record number. Data type of W1 is unsigned integer ranging from 1 to 65535. Within IRDATA-coded programs records are numbered consecutively.

4.2.3 Instruction Code
The instruction code W2 of each record is composed of the type of record and a code number. The type of record is representing a specific task oriented group of IRDATA-instructions. The code number classifies the particular instruction. Data type of W2 is unsigned integer ranging from 1000 to 32767.

\[
\text{type of record} = \frac{(\text{Word W2}) \div 1000 \times 1000}{\text{code number}} = \text{Word W2} \mod 1000
\]

Instruction code = (type of record) + (code number)

\[
\text{DIV} = \text{Integer division} \\
\text{MOD} = \text{Modulo division}
\]

Examples:
W2 = 5022
- type of record = 5000
- code number = 22

4.2.4 Arguments
The words W3 to Wn are representing arguments according to the instruction code W2. If not specified, the data type of W3 to Wn is always real.

4.3 Structure of Programs in IRDATA-Code
The structure of an executable IRDATA-code program is as follows:
1) Instruction code for the begin of program (PBEG, type of record = 22000).
2) Declaration part.
3) Executable part of the main program. It is included within a block.
4) Instruction code for the end of program (PEND, type of record = 22000)

4.3.1 Declaration Part
The declaration part consists of procedure- and task-declarations (see also type of record = 22000) as well as lists of data (type of record = 14000). This part could be empty. The sequence of declarations is not specified.

4.3.2 Executable Part
The executable part consists of a set of IRDATA instructions without declarations or lists of data. This part could be empty.

4.3.3 Declaration of Procedures and Tasks
A declaration of a procedure or task is composed of
1) the instruction code for the begin of a procedure or task (PRBEG, TSKBEG, type of record = 22000)
2) the executable part, which is part of a block, if the procedure or task has local variables or parameters
3) the instruction code for the end of a procedure or task (RETURN, TSKEND, type of record = 22000).
4.4 Type of Records

*Record 1000*
This record contains the linenumber and the designation of the user program statements together with additional user information.

*Record 2000*
This record contains technical specifications for the robot motion, e.g. speed, acceleration etc.

*Record 5000*
This record contains motion statements.

*Record 9000*
This record contains definitions of units of measurements, length and angles as well as scaling factors.

*Record 14000*
This record contains statements to transfer lists of data.

*Record 17000*
This record contains definitions of tool dimensions.

*Record 19000*
This record contains definitions of robot kinematic and its workspace.

*Record 21000*
This record contains statements for mathematical operations with reference to the data types described in Chap. 5.1.

*Record 22000*
This record contains statements for program structure, program flow and memory organization.

*Record 23000*
This record contains statements for communication with peripheral devices, operator panels and robot controllers.

*Record 24000*
This record contains definitions and instructions for tracking (see part 2).

*Record 25000-27000*
Reserved for future extensions.

*Record 28000-32000*
Reserved for user requirements and application specific statements.

5 Data Types and Adressing Methods

5.1 Data Types
According to this standard, the following data types as shown in Tab. 1 can be used in IRDATA-code. In addition, any new data types can be specified by user specific commands. The representation of these data types depends on the implementation and the control architecture and is not part of this standard. The specific codes to represent data types are unsigned integer.
<table>
<thead>
<tr>
<th>Data type</th>
<th>Abbreviation</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>B</td>
<td>129</td>
<td>A data type covering the logical values &quot;TRUE&quot; and &quot;FALSE&quot;. These logic values are defined as follows: TRUE = 1, FALSE = 0</td>
</tr>
<tr>
<td>INTEGER</td>
<td>I</td>
<td>130</td>
<td>A data type covering a range of nonfractional numbers as defined in Chap. 4.1.3.</td>
</tr>
<tr>
<td>REAL</td>
<td>R</td>
<td>132</td>
<td>A data type covering a range of fractional numbers as defined in Chap. 4.1.3.</td>
</tr>
<tr>
<td>VECTOR</td>
<td>VEC</td>
<td>4</td>
<td>A data type composed of 3 real numbers describing a position by 3 cartesian coordinates as defined in Chap. 5.2, using the current unit of measurement. These sub data types are defined to handle with particular vector components.</td>
</tr>
<tr>
<td></td>
<td>VECX</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VECY</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VECZ</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>ORI</td>
<td>8</td>
<td>A data type composed of 3 real numbers describing the orientation of end effectors by 3 revolving angles relative to a cartesian coordinate system as defined in Chap. 5.2, using the current unit of measurement. These sub data types are defined to handle with the orientation components.</td>
</tr>
<tr>
<td></td>
<td>ORIO</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORIA</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORIT</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>ADDITIONAL AXIS</td>
<td>ADX</td>
<td>32</td>
<td>A data type composed of up to 31 real numbers describing the position of any additional axis by joint coordinates. These sub data types are defined to handle with the particular ADX components.</td>
</tr>
<tr>
<td></td>
<td>ADX1</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADX2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADX31</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>WORLD</td>
<td>WLD</td>
<td>16</td>
<td>A data type composed of VECOR and ORIENTATION. They define the position and orientation of end effectors with reference to a cartesian coordinate system of Chap. 5.2, using the current unit of measurement. These data types are defined to handle with the WORLD components.</td>
</tr>
<tr>
<td></td>
<td>WLD POS</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD POSX</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD POSY</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD POSZ</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD ORI</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD ORIO</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD ORIA</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLD ORIT</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>JOINT</td>
<td>JOINT</td>
<td>64</td>
<td>A data type of up to 31 real numbers describing a position with reference to a joint coordinate system. The kinematic structure is defined by a record with the instruction code 19010. These sub data types are defined to handle with the JOINT components.</td>
</tr>
<tr>
<td></td>
<td>JOINT1</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JOINT2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JOINT31</td>
<td>95</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1. (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Abbreviation</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>C</td>
<td>136</td>
<td>A data type ranging from 0...255. The first 128 values are representing the ASCII code (Chap. 7.1).</td>
</tr>
<tr>
<td>STRING</td>
<td>STR</td>
<td>144</td>
<td>A data type composed of an array of character. The index starts with 1, the possible length should be not less than 255.</td>
</tr>
<tr>
<td>POINTER</td>
<td>P</td>
<td>160</td>
<td>A data type for addressing memory locations. Only unsigned integers are allowed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Up to</th>
<th>4095</th>
<th>Reserved for extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>From up to</td>
<td>65535</td>
<td>User specified data types.</td>
</tr>
</tbody>
</table>

### 5.3 Adressing Methods

The following adressing methods are defined to handle with any kind of data in IRDATA records.

<table>
<thead>
<tr>
<th>Argument Wi</th>
<th>Argument Wi + 1</th>
<th>Addressing method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTEGER REAL STRING</td>
<td>Boolean or integer constant real-constant character- or stringconstant</td>
</tr>
<tr>
<td>2</td>
<td>bst*s + index</td>
<td>block relative addressing (unsigned integer)</td>
</tr>
<tr>
<td>4</td>
<td>adress</td>
<td>absolute addressing (unsigned integer)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>constant on stack</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>block relative address on stack</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>absolute address on stack (POINTER)</td>
</tr>
<tr>
<td>16</td>
<td>STRING</td>
<td>symbol (name)</td>
</tr>
</tbody>
</table>

Explanations

a) Blockrelative addressing method:

bst = block indenting, unsigned integer 0...65535
index = index with reference to the address table, unsigned integer 0...65535
s = 65536 (2^16)

b) Absolute addresses are calculated by means of instruction ABSADR (type of record 22000), which enables a general indirect addressing method.

c) The elements of the symbol character set is limited to ASCII-code greater than 31. The first element of a symbol is always a letter.

d) The string constant can be empty. The character constant is represented by a string with one element.

e) The organization of blockrelative addresses is performed by using a memory space independent address table as shown in Fig. 6. Several instruction codes are
Appendix (AB)

responsible for the reservation of memory space (BLBEG, DEFVAR and GENARR).

f) Addressing variables by symbolic names are considered as an extension to all other addressing methods and therefore not a minimum requirement.

Fig. 6

<table>
<thead>
<tr>
<th>Addressable</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address variable 1</td>
<td>Value variable 1</td>
</tr>
<tr>
<td>Address variable 2</td>
<td>Value variable 2 3 real values of a vector</td>
</tr>
<tr>
<td>Address variable n</td>
<td>Value variable n</td>
</tr>
</tbody>
</table>
APPENDIX(A9)

Description of the LSS Table and the Intermediate Code Format of the Flexible Post-processor Concept

(a) LSS Table

The structure of the LSS table is divided into 3 blocks, see Fig(1). The first block contains the name of the file and any other comments. The second block contains only 2 numbers which are used when the LSS table is read by the software function. The first number is the maximum number of blocks in the off-line data file. The second number is the maximum number of columns in the last block in the off-line data file. Note that if a block consists of items of the same parameter, eg a block of a number of velocity values arranged in columns, it is possible to consider this as 1 block. The third block is divided into sections as shown in the figure, and it contains details of the logic, semantic and syntactic codes of the off-line program.

Each section represents the data structure of per column of the off-line program. The first line in each section contains 4 numbers. The first number is the block number while the second number is the column number within that block. The third number represents the mode of separation between the present block/column and the previous block/column. The numbers which denotes the mode of separation are shown in Fig(2). The fourth number represents the type of data, those which are currently supported by Law's software [20] are listed in Fig(3). The second line in the section stores all the names used by the data type. This line is ended with a character '*' . If the data type is of the default type used in the software, a character ' & ' is used instead to represent the data type names. After each section, a character '! ' is inserted to enhance readability.

(b) Intermediate Code Data File

This file is divided into 3 blocks. The first block consists of comments, name of the file and any other remarks. The second block contains the robot program data. Each section is again separated by a '! ' and it comprises of 2 lines of information. The first line contains 10 variables which are representation of:

- the robot point number.
- the number of robot points associated with the current robot function. This was based on the initial WRAPS concept that if the robot function is to execute a sub-routine, this number then represents the number of robot points within the sub-routine, otherwise for any other robot command, it will be zero.
- robot function code.
- qualifying status for the robot function if necessary. For example, if function is "TOOL 1" than status is ON or OFF.
- first DAC code.
- second DAC code.
- robot velocity code.
- tool dimension code.
- variable code.
- flag code.

The second line stores the TCP coordinates as X, Y, Z and TCP.
orientation as D, E and R.

The third block comprises of 6 sections and they contain operating system data for the robot. They are grouped as follows: first DAC, second DAC, robot velocities, tool dimensions, variables and flags. Each section holds 16 values of data and is read by the software from left to right in a top-bottom sequence. If a particular section has less than 16 values, the rest will be padded with zeroes.

(c) Format of the code dictionary

The dictionary consists of comments and numbers only. These number codings are the same as used for the LSS table. The first number is the data type code. The second number represents the total number of data code that follows. The sequence '999' is used to signify the end of the data block.

\[\begin{array}{|c|}
\hline
\text{BLOCK} \\
\hline
1 \hline
2 \hline
3 \hline
\text{SECTION} \\
\hline
\end{array}\]

File name is : offlog.dat

```
1 0 2 1 & *
0 1 1 2 & *
0 2 1 3
Home No Op - Tool_1 - Sel_DAC1 Sel_DAC2 Cyc_Start End_Path *
0 3 1 4
ON OFF *
1 0 2 1 *
```

Fig(1) Example of a LSS Table Data file.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Separation by white spaces.</td>
</tr>
<tr>
<td>2</td>
<td>Separation by white spaces or comment lines begin and end with a ' *'.</td>
</tr>
<tr>
<td>3</td>
<td>Separation by white spaces or comment lines begin with a '!'.</td>
</tr>
</tbody>
</table>

Fig(2) Meaning of Code of Separation.
<table>
<thead>
<tr>
<th>Code</th>
<th>Type of Data</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number</td>
<td>Robot program statement or program number.</td>
</tr>
<tr>
<td>2</td>
<td>Number of Points</td>
<td>Number of points in the subroutine if the robot function is a branching function.</td>
</tr>
<tr>
<td>3</td>
<td>Function name</td>
<td>Name of the robot command.</td>
</tr>
<tr>
<td>4</td>
<td>State of Function</td>
<td>Qualifying status to robot function.</td>
</tr>
<tr>
<td>5</td>
<td>TCP (X)</td>
<td>X coordinates of TCP.</td>
</tr>
<tr>
<td>6</td>
<td>TCP (Y)</td>
<td>Y coordinates of TCP.</td>
</tr>
<tr>
<td>7</td>
<td>TCP (Z)</td>
<td>Z coordinates of TCP.</td>
</tr>
<tr>
<td>8</td>
<td>TCP (D)</td>
<td>D pitch angle of TCP.</td>
</tr>
<tr>
<td>9</td>
<td>TCP (E)</td>
<td>E yaw angle of TCP.</td>
</tr>
<tr>
<td>10</td>
<td>TCP (R)</td>
<td>R roll angle of TCP.</td>
</tr>
<tr>
<td>11</td>
<td>Speed</td>
<td>Speed of the robot movement.</td>
</tr>
<tr>
<td>12</td>
<td>First DAC</td>
<td>First Digital-to-Analogue Convertor data.</td>
</tr>
<tr>
<td>13</td>
<td>Second DAC</td>
<td>Second Digital-to-Analogue Convertor data.</td>
</tr>
<tr>
<td>14</td>
<td>Tool Dimension</td>
<td>Tool dimension data type.</td>
</tr>
<tr>
<td>15</td>
<td>Variable</td>
<td>Variable data type.</td>
</tr>
<tr>
<td>16</td>
<td>Flag</td>
<td>Flag data type.</td>
</tr>
<tr>
<td>17</td>
<td>Function Name and State</td>
<td>Function name and status together.</td>
</tr>
<tr>
<td>18</td>
<td>Other</td>
<td>Other types of data.</td>
</tr>
</tbody>
</table>

Fig(3) Robot Function and Data Types Currently Supported, by Law's [20] software.
### Layers of the ISO Open Systems Interconnection Standards

#### OPEN SYSTEMS INTERCONNECTION (OSI) STANDARDS

The International Standards Organisation open systems Interconnection (ISO/OSI) model describes the communication process as a hierarchy of layers, each dependent on the layer directly beneath it. Each layer has a defined interface with the layer above and the layer below, this interface is made flexible so that designers can implement various communications protocols and still follow the standard.

The fundamental purpose of the OSI model is to provide a standard for data communications systems, that will provide the basis for machines to communicate over Local and Wide Area Networks. The seven layers of the OSI Model are:

1. **The Physical Layer:**
   - This layer defines the physical connection between the computer and the network, including the mechanical aspects of the connection and the electrical aspects. This layer also defines the topology.

2. **The Data Link Layer:**
   - This layer defines the protocol that computers must follow to access the network for transmitting and receiving messages. These messages are sent onto the network as specially formatted discrete frames of information rather than being continuously broadcast. If data input to the layer is large enough, the data link layer will break it up into several frames. This layer also specializes handling of frame receipt acknowledgement if required.

3. **The Network Layer:**
   - This layer adds destination switching, routing, and relaying functions, and presents these in a manner which is independent of the actual network in use.
   - This provides interconnection across a variety of inter-linked networks.

4. **The Transport Layer:**
   - The transport layer defines how you address the physical location/devices on the network, how connections between nodes can be made or broken, what the protocol is for guaranteed message delivery and how to handle the internetwork routing of messages.

5. **The Session Layer:**
   - This layer manages the conceptual interface to the transport layer for applications. For example, it is this layer that lets you refer to devices by name rather than by their network address. This lets you write software that will run on any given kind of network. Layers three, four and five are frequently described as the network's subnet level.

6. **The Presentation Layer:**
   - This layer defines how applications can enter the network, and it translates the format and syntax of the data they produce and consume for its transmission on the network.

7. **The Application Layer:**
   - This uppermost layer simply defines the network applications that support file serving. Conceptually, this is where electronic mail and other network utility software exists.