Please note that fines are charged on ALL overdue items.
A Knowledge-based Microcontroller Software Development System

by

Chun Yu, Chen
(B. Eng., MSc)

A Doctoral Thesis

Submitted in partial fulfilment of the requirement for the award of Doctor of Philosophy of Loughborough University

December 2003

© by Chun Yu, Chen, 2003
Abstract

A large range of products are now designed based on the implementation of microcontrollers, as they lead to an overall system enhancement, such as a more compact system design and reduced costs. This can be credited to the peripheral functions embodied in the microcontroller and the flexibility offered by its software. Since the applications of microcontrollers are achieved through software, the implementation of the target controller is considered as an interdisciplinary process. Due to the capability of the current development tool, designers are required to input significant effort such that the development with the process is taking a long time to complete.

The purpose of this research is aimed at improving the current microcontroller software development process. Investigation of the existing development process revealed that the problems that impede the process were due to the lack of knowledge across the boundaries of different engineering disciplines. Therefore, it is desired to offer knowledge, experience and implementation rules to achieve the aim of software development improvement. Currently, knowledge-based systems are capable of offering specific knowledge and expertise to facilitate problem solving, and it is the intention of using such technologies to build solutions for improving current microcontroller software development.

A graphical knowledge-based microcontroller software development system was built according the needs of the current process. Under this system, the implementation rules to process the software development is handled, and supporting knowledge is offered. Also, designers are allowed to implement their designs by manipulating graphical objects rather than coding. Furthermore, this system is not merely supporting the configuration of peripheral functions, since the system composition and system behaviour settings are also included. Thus, the whole process of microcontroller implementation is covered and supported by this system.

When using this knowledge-based system to carry out microcontroller-based system design, the required time is shorter and the problems faced are fewer than using the traditional development tool. It can be concluded that a straightforward and consistent process is delivered by the combination of a knowledge-based system and the implementation rules of microcontroller software development. Furthermore, the implementation of the target controller is integrated with the intended system development. In addition, the fast system prototyping of microcontroller-based products can be achieved.
This thesis is dedicated to my parents.
Acknowledgements

In completing this thesis, I indebted to many people for their help, support, encouragement and advice.

My supervisors, Dr. Mike Jackson and Professor Rob Parkin, for their guidance throughout my doctoral training and generous encouragement and support in many of my academic endeavours and daily life.

Mr Dave Whitby, for sharing his experience in microcontroller software programming.

Dr Ching-Lin Wu, for his encouragement and support in many ways of house sharing and tennis coaching and through the route of PhD route together.

Jiun Keat Ong, Pavel Hynek, Luca Notini, Sebastien Salvan, for the effort to accompany with me for the PhD route.

Ping Chao Lee, and Tsai Li Li, for their encouragement to cheer me on.

Weber Lin, and Amjad Juna, for their effort to correct my writing.

Follow colleagues and friends in the Mechatronic research group for the comradeship and help.

Friends at Loughborough Taiwanese Society for their friendships and support.
List of Contents

Abstract .......................................................................................................................... i
Acknowledgements ...................................................................................................... iii
List of Contents ............................................................................................................ iv
List of Figures ................................................................................................................ x
List of Tables .............................................................................................................. xiv

Chapter 1 Introduction ................................................................................................. 1
  1.1. MOTIVATIONS AND AIMS .................................................................................. 2
  1.2. ENGINEERING CONTRIBUTION OF THE DISSERTATION ................................... 2
  1.3. SCOPE OF THE DISSERTATION ........................................................................... 3
  1.4. DISSERTATION OVERVIEW ................................................................................ 6

Chapter 2 The Development of Microcontroller Applications .................................... 9
  2.1. BENEFITS FROM DIGITAL COMPONENTS ............................................................... 9
    2.1.1. Era of mechatronics ...................................................................................... 10
    2.1.2. Microcontroller-based systems ....................................................................... 12
          Compact design with reduced cost ...................................................................... 14
          Improved system performance ............................................................................ 14
          Reliability ........................................................................................................... 14
  2.2. SOFTWARE DEVELOPMENT FOR MICROCONTROLLER ........................................ 15
    2.2.1. Fulfilling requirements - translating design into functions .............................. 16
    2.2.2. Realising functions - coding .......................................................................... 16
    2.2.3. Software debugging and testing ...................................................................... 17
  2.3. PROGRAMMING LANGUAGES FOR MICROCONTROLLERS .................................... 17
    2.3.1. C language ..................................................................................................... 18
          Increased software productivity ........................................................................... 18
          Improved software quality ................................................................................. 18
    2.3.2. Assembly language ........................................................................................ 19
          Speed of execution is critical .............................................................................. 20
          Memory size is a factor ....................................................................................... 21
  2.4. SUMMARY ............................................................................................................ 21

Chapter 3 Considerations of microcontroller software development ........................... 23
3.1. EFFORT REQUIRED WHILE COMPOSING MICROCONTROLLER-BASED SYSTEMS ... 23

3.1.1. composing software design ........................................................................ 24
   3.1.1.1. Understanding the microcontroller ..................................................... 24
   3.1.1.2. Understanding the relationship between the microcontroller and other components ................................................................. 26

3.1.2. Searching for related information .............................................................. 27

3.1.3. Completing software design by assembly language ................................... 29

3.1.4. program modification .............................................................................. 30
   3.1.4.1. Producing the intended machine code ................................................ 31
   3.1.4.2. Justifying the hardware function ...................................................... 32

3.2. DISADVANTAGES OF CURRENT DEVELOPMENT PROCESS ..................... 34
   3.2.1. Lack of information resources ............................................................... 34
   3.2.2. Insufficiency of software development experience .................................... 35
   3.2.3. limited assistance from development environment .................................... 36
   3.2.4. Unfamiliarity with coding instructions .................................................. 37

3.3. SUMMARY ....................................................................................................... 38

Chapter 4 Review of previous work ........................................................................ 39

4.1. ADVANCED SOLUTIONS FOR SYSTEM DEVELOPMENT ......................... 39

4.2. IMPROVEMENT FROM SYSTEM DEVELOPMENT METHODOLOGIES ........... 42
   4.2.1. Mechatronic system design ..................................................................... 42
   4.2.2. Hardware/Software codesign .................................................................. 44

4.3. IMPROVEMENT OF SOFTWARE DEVELOPMENT TOOLS ......................... 47
   4.3.1. Infineon- DAvE ...................................................................................... 48
   4.3.2. Lego- Mindstorm .................................................................................. 51

4.4. FEATURES OF THESE SYSTEMS .................................................................. 54
   4.4.1. Fast and quality software development ................................................... 54
   4.4.2. Graphical implementation interface (GII) ................................................. 56
   4.4.3. Limited information and experience inquiry .......................................... 58

4.5. LIMITATIONS OF THESE SYSTEMS .......................................................... 60
   4.5.1. Limited usage ......................................................................................... 60
   4.5.2. Performance justification .................................................................... 61

4.6. POSSIBLE METHODS TO IMPROVE THE CURRENT SOFTWARE DESIGN PROCESS .... 62
   4.6.1. Software reuse ...................................................................................... 63
4.6.2. Knowledge based system (expert system) .................................................. 64

4.7. PRINCIPLES FOR SELECTING THE PROPER METHOD ............................................. 66
4.7.1. Criteria........................................................................................................ 66
  4.7.1.1. Fulfilling the development needs of microcontroller applications ....67
  4.7.1.2. Releasing the potential of the target controller ................................. 67
  4.7.1.3. Memory space ..................................................................................... 67
  4.7.2. Selected solution...................................................................................... 68

4.8. PROPOSED TECHNOLOGY FOR THE DEVELOPMENT OF THIS KNOWLEDGE-BASED SYSTEM .................................................................................... 68
  4.8.1. Visual performance .................................................................................... 69
  4.8.2. Rule-based process .................................................................................... 69

4.8. SUMMARY ........................................................................................................ 70

Chapter 5 Design of the advanced microcontroller-based development system ...... 72
5.1. CONSIDERED ISSUES WHILE PROPOSING A KNOWLEDGE-BASED SOFTWARE DEVELOPMENT SYSTEM ............................................................................................. 73
  5.1.1. Usability ..................................................................................................... 73
  5.1.2. Functionality .............................................................................................. 73

5.2. CORE KNOWLEDGE BASES ........................................................................... 74
5.3. SYSTEM-LEVEL PROTOTYPING ................................................................... 75
5.4. PROFILING OF THE TARGET MICROCONTROLLER ................................................. 76
5.5. PERIPHERAL FUNCTION CONFIGURATION ........................................................... 77
5.6. SYSTEM CONTROL IMPLEMENTATION ................................................................. 79
5.7. EASY ACCESSING APPROACH .............................................................................. 80
5.8. SUMMARY ........................................................................................................ 81

Chapter 6 Implementation of the knowledge-based microcontroller software development system ..................................................................................................... 82

  6.1. SYSTEM OVERVIEW ............................................................................................ 83
     6.1.1. Development guiding strategy .................................................................... 85
     6.1.2. Structure of Knowledge-based system ......................................................... 87

  6.2. LINKING DATA STREAM AMONG THESE KNOWLEDGE BASES ......................... 90
     6.2.1. System composition .................................................................................... 91
     6.2.2. Microcontroller configuration ..................................................................... 92
     6.2.3. Implementation of system control strategy ................................................ 93
6.3. SYSTEM COMPOSITION ................................................................. 95
6.3.1. Integration with external components ........................................ 95
6.3.2. Interfacing methods with external components ............................ 97
6.4. APPLICATION PLAN OF THE TARGET MICROCONTROLLER ...... 98
6.4.1. Available resources and corresponding pins ............................... 99
6.5. PERFORMANCE-ORIENTED FUNCTION IMPLEMENTATION INTERFACE .......................................................... 102
6.5.1. Detailed function configuration ................................................... 104
6.5.2. Fast performance setting ............................................................... 105
6.5.3. Information provided to complete the setting ............................... 107
6.5.4. Result of simulation ................................................................. 110
6.6. FLOWCHART SOFTWARE CONTROL IMPLEMENTATION ......... 111
6.6.1. Events oriented factors ................................................................. 112
6.6.1.1. Major factors ......................................................................... 115
6.6.1.2. Sub-factors ........................................................................... 117
6.6.2. Subfunctions ............................................................................ 119
6.6.2.1. Pulse generator ..................................................................... 119
6.6.2.2. Time delay ........................................................................... 120
6.6.2.3. Encoder reading .................................................................... 120
6.7. CODE GENERATION ................................................................. 121
6.7.1. Function configuration ................................................................. 122
6.7.2. System control implementation .................................................. 123
6.8. TESTING PHASE ........................................................................ 124
6.8.1. PIC17C756 ............................................................................... 125
6.8.2. MPLAB .................................................................................... 125
6.8.3. Emulator .................................................................................. 126
6.8.4. Teaching kits ........................................................................... 126
6.8.5. Pilot study ................................................................................ 129
6.9. SUMMARY ................................................................................. 131

Chapter 7 Evaluation of the software development expert system with further discussion ................................................................. 133
7.1. TRADITIONAL SOFTWARE DEVELOPMENT PROCESS .......... 134
7.1.1. Fulfilling system needs by available functions ......................... 135
7.1.2. Seeking supporting information .................................................. 137
7.1.3. Implementing design aims through coding ................................. 139
7.1.4. Errors appearing and debugging .......................................................... 140
7.2. KNOWLEDGE-BASED SOFTWARE DEVELOPMENT PROCESS .......... 142
  7.2.1. Composing intended systems by available functions ...................... 143
  7.2.2. Initialising the intended peripheral functions ............................... 145
  7.2.3. Implementation of control strategy .............................................. 148
  7.2.4. Program testing .......................................................................... 151
7.3. COMPARISON OF THE TWO DEVELOPMENT PROCESSES .................. 152
  7.3.1. System composition ...................................................................... 152
     7.3.1.1. Development cycle ................................................................. 153
     7.3.1.2. External information accessing ............................................. 154
  7.3.2. Function configuration ................................................................. 156
  7.3.3. producing the corresponding machine code .................................... 159
  7.3.4. Program testing ........................................................................... 160
  7.3.5. Full process comparison .............................................................. 163
     Problems encountered ........................................................................ 164
     Information inquiry ........................................................................... 164
     Produced program ............................................................................ 165
     Development time ............................................................................ 166
7.4. DISCUSSION ...................................................................................... 167
  7.4.1. Effective development process ..................................................... 167
     7.4.1.1. Short development cycle ......................................................... 168
     7.4.1.2. Consistent development process ............................................. 168
     7.4.1.3. Organising available resources .............................................. 169
  7.4.2. User friendly implementation environment ...................................... 170
     7.4.2.1. Limited external Data accessing ............................................. 170
     7.4.2.2. Performance-oriented configuration ....................................... 171
     7.4.2.3. System behaviour for control strategy implementation ............ 172
  7.4.3. Quality program delivery .............................................................. 173
     7.4.3.1 Automatic code generation ...................................................... 173
     7.4.3.2. Software quality ..................................................................... 174
7.5. LIMITATIONS OF THE SOFTWARE DEVELOPMENT EXPERT SYSTEM ..... 175
  7.5.1. Multi-level condition setting ......................................................... 176
  7.5.2. Fixed mode of implementation ...................................................... 176
  7.5.3. Limited assembly programming experienced gained .................... 177
7.5.4. Adaptive learning is under establishing

7.6. SUMMARY

Chapter 8 Conclusion and further work

8.1. CONCLUSION

8.1.1. Summary of work undertaken

8.1.2. Contribution to knowledge

8.2. FURTHER WORK

8.2.1. Effective system simulation

8.2.2. Documentation for proposed systems

8.2.3. Tradeoffs for system implementation

8.2.4. Increasing the understanding of assembly language

8.2.5. Interactive implementation environment

8.2.6. Adaptive learning expert system

References

Appendices

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E
List of Figures

Figure 1- 1: Roadmap of this research work .................................................. 5

Figure 2- 1: Examples of microcontroller-based applications ....................... 10
Figure 2- 2: The integration of Mechatronics .............................................. 11
Figure 2- 3: Existing applications of microcontroller-based system ............. 13
Figure 2- 4: Software development process ................................................. 16
Figure 2- 5: An example of C type program ................................................ 19
Figure 2- 6: An example of Assembly program ........................................... 20

Figure 3- 1: A simplified scheme of peripheral functions with control mechanism inside a microcontroller chip (PIC17C756) ...................... 25
Figure 3- 2: Usages of the mechanisms inside the SFR ................................ 28
Figure 3- 3: The composition of a software-based hardware PWM waveform .................................................................................. 29
Figure 3- 4: Example of assembly instructions (Microchip P17C756) .......... 30
Figure 3- 5: The PIC In-Circuit Emulator with hardware testing components .................................................................................. 32
Figure 3- 6: The schematic diagram of emulator with peripheral boards ..... 33

Figure 4- 1: A modelling picture for component based system integration. 40
Figure 4- 2: The decomposition of Mechatronics system ............................... 43
Figure 4- 3: Hardware/Software Codesign .................................................... 45
Figure 4- 4: A smart search platform for designers to search suitable microcontroller .................................................................................. 48
Figure 4- 5: The architecture and available peripheral functions of the target microcontroller .................................................................................. 49
Figure 4- 6: An example of peripheral function setting in DAVeE .............. 50
Figure 4- 7: The created program according to designer's intention .......... 51
Figure 4- 8: Software development environment of RiS ............................. 52
Figure 4-9: The possible actions of Mindstorms are placed on the left hand side ................................................................. 53
Figure 4-10: A completed design of Mindstorms software .............. 54
Figure 4-11: The program developing progress of DAvE .................. 55
Figure 4-12: Drag and drop the intended puzzles to compose the proposed design ................................................................................................... 56
Figure 4-13: The switching of interfaces in DAvE .............................. 57
Figure 4-14: The condition and action setting in RIS .......................... 58
Figure 4-15: One of the provided information from DAvE ................. 59

Figure 5-1: The structure of intended knowledge-based software development system ........................................................................ 74

Figure 6-1: External effort required during traditional software development process ................................................................. 84
Figure 6-2: The intended software development supported by proposed KB development system ...................................................... 84
Figure 6-3: The sequence of development activities with required knowledge ...................................................................................... 85
Figure 6-4: The structure of the expert system and supporting databases with guiding strategy ................................................................. 88
Figure 6-5: Producing the intended systems with the required interface in KB1 ...................................................................................... 91
Figure 6-6: Configuring essential functions for the proposed system ...... 93
Figure 6-7: Composing the intended system performance through available system action ........................................................................ 94
Figure 6-8: The system composition tool inside the knowledge-based software development system ...................................................... 95
Figure 6-9: The auxiliary mechanism to prevent failure of microcontroller system design ...................................................................................... 96
Figure 6-10: The options displayed for DC-motor interfacing and signal feedback ...................................................................................... 97
Figure 6-11: The application purpose of the target microcontroller ...... 98
Figure 6-12: The available resources of microcontroller and its associated ports (featured from Microchip PIC17C756 microcontroller) ..........100
Figure 6-13: The composition of two PWM functions ..........................101
Figure 6-14: The completed mode of PWM function configuration .......105
Figure 6-15: The simplified mode of PWM configuration ..................107
Figure 6-16: Guiding message between system configuration and software development ..........................................................108
Figure 6-17: Indicating message for designers to complete their design ..108
Figure 6-18: Messages to indicate the missing part of function configuration ....................................................................................109
Figure 6-19: The simulation result of configured PWM waveform .......111
Figure 6-20: The setting process of a conditional action ......................113
Figure 6-21: Microcontroller-based ADC System diagram and control strategy .............................................................................115
Figure 6-22: The implementation process of control strategy by major factors ................................................................................116
Figure 6-23: The configuration of conditional actions .........................118
Figure 6-24: The location of function configuration in software program ........................................................................................................122
Figure 6-25: Program with comments to explain the instructions .........123
Figure 6-26: A label is applied for a conditional action .......................124
Figure 6-27: The emulator with teaching kits ....................................127
Figure 6-28: A microcontroller-based DC-motor system composed by emulator and teaching kits with system diagram ................128
Figure 6-29: The control strategy of these two DC-motors by the received data ..................................................................................130
Figure 6-30: The produced program for the intended motor control system ..........................................................................................131

Figure 7-1: A block system diagram of the target controller ............136
Figure 7-2: Mapping system configuration with an appropriate implementation solution .................................................................137
Figure 7-3: Required functions to compose the intended PWM waveform ..............................................................................................138
Figure 7- 4: Control mechanisms required to enable the interrupt function of timer3 ........................................................................................................ 141
Figure 7- 5: The first stage of the intended software development under knowledge-based system........................................................................ 143
Figure 7- 6: Information required to select the desired components .......... 144
Figure 7- 7: Time required to complete system composition .................... 144
Figure 7- 8: Problem faced during system composition ......................... 145
Figure 7- 9: Problems found offered information with this KB system .... 146
Figure 7- 10: External information accessed using KB system .............. 147
Figure 7- 11: The implementation tasks listed according to system composition..................................................................................................... 148
Figure 7- 12: Time required to complete software implementation .......... 149
Figure 7- 13: Problems faced during software implementation process.... 150
Figure 7- 14: An example of implementing the intended control strategy 150
Figure 7- 15: Errors found in the produced program by KB system ...... 151
Figure 7- 16: Performance of produced programs by KB system ........ 152
Figure 7- 17: Required time to complete system composition .................. 153
Figure 7- 18: Information acquisition during software implementation .... 154
Figure 7- 19: Problems faced during system composition ....................... 155
Figure 7- 20: Problems faced while dealing implementation tool .......... 157
Figure 7- 21: Problems found during implementation process ............... 158
Figure 7- 22: Time required to complete the implementation of design content ...................................................................................................... 159
Figure 7- 23: Errors of produced programs found during compilation ...... 160
Figure 7- 24: Debug process needed to correct the performance of produced programs.................................................................................................. 161
Figure 7- 25: Difference in information required during debugging ...... 162
Figure 7- 26: Difference in time required during debugging ................. 162
Figure 7- 27: Problems encountered during the development process .... 162
Figure 7- 28: The needs of vendor offered information during the development process ..................................................................................... 165
Figure 7- 29: Quality of the produced programs from different development processes ............................................................................................. 166
Figure 7- 30: The required time for software development ................. 166

xiii
List of Tables

Table 7-1: Indication factors of system composition during the traditional process

Table 7-2: Time required to complete the system composition

Table 7-3: Indication factors of software implementation during traditional process

Table 7-4: Special function registers required while configuring PWM1 function (PIC17C756)

Table 7-5: Indication factors of debugging during traditional process
Chapter I-Introduction

In order to reduce the cost and time for microcontroller software development, various strategies have been applied. These strategies were aimed at simplifying the development process and delivering corresponding programs as the final result. Somehow, the software development for microcontroller has not really benefited from these developed methodologies, as the development activities contained inside still require a lot of work, which was devoted by designers’ manual work.

1.1. Motivations and aims

The motivations for this thesis are to research, analyse and develop the technology required for a novel software development system for microcontrollers. During the building process of this software development system, advanced solutions were applied while the decided characteristics from various types of possible technologies and intelligent instrumentation are combined, e.g., a fast system composition environment and automatic code generation features. The aim is to deliver a prototype system with feasibility while developing microcontroller software. Research work for this thesis has been carried out as a hands-on study to present a practical engineering point of view on the subject.

The research is aimed at improving current microcontroller software development. By suggesting a novel solution, it is hoped that this will lead to a more efficient, economic and automated era of microcontroller-based product development. This project can be considered as the beginning of an ongoing research effort. It is meant to prove the suggested concept and investigate a few important criteria so as to allow further research to develop a system to suit future needs.

1.2. Engineering contribution of the dissertation

The engineering contributions of this dissertation consist of the following themes:

- A novel implementation of knowledge-based system for microcontroller software development has been introduced. The structure of the software development system has been described in detail including knowledge bases,
development guiding strategy, and the interfaces. Related technologies and potential applications have also been examined.

- The concept of a knowledge-based system as a software development environment has been formulated and presented from a practical oriented engineering point of view.

- The functioning of this knowledge-based software development system was analysed and tested with actual microcontroller-based system modules. The performance of the system was evaluated. A popular software development strategy was also modified to suit the development system with knowledge bases. As far as it is known, this was the first microcontroller software development environment embodied with knowledge bases.

- Attention was given to the practical application of this software development system. As a result of this research work, the prototype system has ideas on the trend of microcontroller-based applications and the possibility of expanding into a comprehensive development system for microcontroller-based systems in the future.

1.3. Scope of the dissertation

This thesis deals with the implementation of a knowledge-based microcontroller software development system, including fast system prototyping, function implementation and software program delivery. Two major subjects are treated in this research work: the new concept of microcontroller software development and concept development as an engineering framework. The majority of this thesis focuses on the problem of achieving and ensuring an efficient microcontroller software development and the delivery of a proper program through the implementation as the final result.

The research work first investigated the issues involved during microcontroller software development with the existing development tool. It should be mentioned that this work is not primarily based on developing an automatic program generation system. The approach is designed to achieve an efficient software development using a combination of user-friendly interfaces and supporting knowledge bases. The structure of this system has been designed for use in an advanced microcontroller
software development, where the applications of the target controllers are due to system configurations.

For the software development of microcontrollers, the whole process is divided into several activities which target individual purposes and can be supported by specific knowledge. Due to the sequence and characteristics of these development activities, a proper implementation strategy is deployed to link these activities. Automatic program generation is made available to this knowledge-based development system by the captured knowledge after the implementation process.

The conclusions suggest that this microcontroller software development process can benefit from the former experiences or expertise. Comparison between available technologies was done and a knowledge-based system is chosen. Hence, this work was an attempt to investigate these knowledge-based principles for microcontroller system development. Experiments were then carried out to test its feasibility and observe its limitations. The functions of monitoring designers' intentions and accumulating experiences are capable with knowledge-based systems, but it was not contained inside the software development system due to the limitation of time for system construction. The proposal aims of augmenting the functionality of this system will be described in further work.

A plan for achieving these goals is presented Figure 1-1. The experiments carried out in this framework are to support and verify the proposal aims. The experimental process includes the comparison of related issues during traditional and knowledge-based software development process.

In summary, the goal of this research is:

The demonstration of how using a combination of graphical interfaces and knowledge-based system can improve the efficiency of the microcontroller software development process. The addition of an automatic program generator further increases the usability of this knowledge-based system.
Chapter I - Introduction

Literature review

Gain experience related to μc software

Understand development environment

Identify problems

Set research directions, aims and objectives

Existing IDE

Guiding process

Options setting

Interface composing

Development of system structure

Testing of the interfaces

Ensure achievement in software development

Integrating interfaces into system

Created fast function configuration options

Knowledge-based microcontroller software development system

Supporting information offered

Graphic interface for implementation

Generate the corresponding codes

Apply of former experiences

Development of expert system

Figure 1-1: Roadmap of this research work
1.4. Dissertation overview

The research project is divided into the following four sections:

1. Literature review and research methodology
2. Design considerations and proposed features
3. Implementation and system and performance evaluation
4. Improvement and suggestions for further work

The tasks for the first phase are:

- To obtain information on current microcontroller applications and its software development, and consolidate fundamental knowledge needed to develop the advanced software development system.
- To analyse the difficulty of current microcontroller software development.
- To define the proposed performance for the intended microcontroller software development system.

After the first phase, the second and third phases form the critical part of the thesis. They are described as identifying problems encountered during the development stage and establish a novel approach to solve the problems. In this project, the research work focuses on the implementation of a knowledge-based system for microcontroller software development with automatic program generation. The developed result is confirmed by experiments to prove the ideas and performance.

The final phase describes further improvement and consolidates the work done, and presents the thesis detailing the reasons and future development processes.

The dissertation is organised as follows:

Chapter 2 provides background knowledge on microcontroller applications. The benefits offered by microcontrollers and the corresponding system development process is discussed. The key element in current microcontroller software development will be discussed. The predominance of assembly languages in microcontroller software development will be explained.
Chapter 3 describes the difficulties and problems encountered while developing microcontroller software and the required work. The thinking behind the microcontroller software development is discussed in detail towards the end of the chapter. This chapter also includes a list of problems found in current software development, which helps in the later development stage of the proposed knowledge-based system.

Chapter 4 presents the literature review about current methodologies and possible improvements related to the development of microcontroller-based systems. Existing solutions for microcontroller software development improvement are introduced. Additional features are suggested to improve the efficiency compared with traditional development process. It also demonstrates how the proposal can be applied to work for microcontroller software development. The possible technology to achieve this purpose is described, together with its characteristics.

Chapter 5 explains the proposed features of the intended knowledge-based microcontroller software development system. It is important to improve software development from every aspect, and produce quality programs as the final result. Since the implementation of the target controller was decided during the system configuration, it is essential to encompass system composition into the software development system. The features of the systems will be discussed in detail.

Chapter 6 contains the results of the implementation for the knowledge-based software development system. The structure of the knowledge-based system along with the implementation process is displayed. The methods used to simplify the software development process by graphical interfaces and achieve proposed performance through offered options are presented. Finally, the flowchart type software control strategy implementation environment is shown. All these sections will be discussed in detail.

Chapter 7 discusses the performance of the developed system. The evaluation was carried out through a comparison of development related issues during traditional and knowledge-based software development processes. The feasibility of software
Chapter I - Introduction

development through this knowledge-based system is analysed and explained. The advantages and limitations of the work are summarised.

Chapter 8 gives the discussions of the research work done. Future directions of the research to achieve a more efficient software development are proposed. Several possible areas of research are identified.

Five appendices follow the main content:
Appendix A is the testing module designed to test the capability of this knowledge-based microcontroller software development system.
Appendix B is the testing module to evaluate traditional microcontroller software development process.
Appendix C is the testing module to evaluate the knowledge-based microcontroller software development process.
Appendix D is the paper for International Conference of Mechatronics (ICOM).
Appendix E is the paper for International conference of Mechatronic Technology (ICMT)
The Development of Microcontroller Applications

A typical digital control system is usually composed of several units such as target controller, interfacing components and external devices. Each unit needs to be qualified to fit into the system through an evaluation process. Often, the interfacing design between the target controller and the external components takes a lot of the system design time. In order to accelerate the system development by reducing the workload for interfacing, microcontrollers are usually used as the target controller. This is because most of the interfacing circuitry has been integrated in the microcontrollers, and the development of such control systems can thus be simplified. This is also why the number of microcontroller applications grows.

The focus of this research was aimed at improving the current microcontroller software development process. It will begin by reviewing the applications of microcontrollers, the current software development method, and key element in microcontroller software development will be discussed.

2.1. Benefits from digital components

Traditionally, automatic control has been implemented by analogue components. There is a strong move towards digital control components because of the low cost of the components and the control abilities that they can perform. Another important reason why digital control units are so widely used is that they offer the possibility of improving the performance or realising complex systems. Currently, microcontrollers, microprocessors, or programmable logic devices are the most popular devices.
Digital control units could be found in both industrial automatic control and domestic electronic products (Figure 2-1), although these areas were previously dominated by analogue components. By the very nature of digital control units, they can be programmed to replace those conventional electrical logic components, and perform 'intelligent' functions that are expensive and difficult to implement by using conventional analogue components [1]. In addition, the costs of system implementation by applying digital control units have been reduced. More importantly, this leads to improved reliability due to the reduced number of components used [2]. Also performance modifications are considerably simplified, often requiring only changes to the software, not to the hardware. Due to these benefits, the number of digital applications is expanding.

Figure 2-1: Examples of microcontroller-based applications

2.1.1. ERA OF MECHATRONICS

Another important factor making applications of digital control units so widespread is the development of Mechatronics. Mechatronics is the synergistic integration of Mechanics, Electronics, Embedded Control and IT in the design and realisation of intelligent products, processes and systems (Figure 2-2). Mechatronic systems deal
with electronically controlled devices that are usually performing the distribution of functions between mechanical, electronic and software components. The performance of the mechatronic systems is the integration of each element [3]. In other words, the mechatronic systems attempt to take all the advantages from the components inside the system.

Figure 2-2: The integration of Mechatronics

There is a very close relationship between the areas of control and mechatronics. In control, a sensor and actuator are combined with a digital computer or analogue electronics to achieve effective control in a dynamic system. In mechatronics, the primary objective of system design can be summarised as sensing outside variables, computation of an appropriate output command, and using the command to exert influence on the outside world. The final product from the mechatronic design often has elements of control involved.

In the 1980s, significant advances in microcomputer and powerful electronic technology have made more complicated control algorithms realisable in mechatronic systems. Therefore, “Mechatronics” can be referred to as the use of hardware oriented apparatus with dedicated programmable control. Programmable devices nowadays vary in their speeds, capabilities, complexities and costs. In the past, emphasis was put on microprocessor-based equipments. The advance of microcontrollers and what
they offer combined with their speed made them more suitable for instruments. In the next section, the microcontroller and its applications will be discussed.

2.1.2. MICROCONTROLLER-BASED SYSTEMS

In the past, a microcomputer was composed of a microprocessor with external components such as I/O devices, and memory. As current technology advances, a much more comprehensive design of microcomputer has been created in a chip, called a microcontroller.

The basic characteristics of microcontrollers and microprocessors are very similar, since they both process a series of instructions that constitute to a program. Although they can both perform flexible functions due to programmability, there is a slight difference between these two. Microprocessors are oriented towards higher Million Instructions Per Second (MIPS), whereas microcontrollers are designed with peripheral circuitry on chips that provide I/O ports, timing capability, AD converter, serial communication, and various memory access capability. The availability of these peripheral functions on chip is an attractive feature as designs require fewer components outside the microcontroller and hence the result is a compact, less expensive product.

Microcontrollers are used extensively these days in a wide range of applications, such as process control units, communication systems, digital instruments, electronics and home products. Examples of microcontroller applications are given in Figure 2-3. This is because of the benefits provided by the microcontrollers, such as low cost, programmability and the readily availability of sophisticated peripherals. Accordingly, some products have been improved by applying microcontroller-based solutions. In addition, some systems which were not available or expensive with traditional electrical components have been realised by the appearance of microcontrollers [4]. [1, 5-11] can be used to illustrate the improvement that microcontroller-based systems can bring over traditional systems. Also the performance modification of a microcontroller-based system is considerably simplified, as it requires only software rather than hardware [12]. Later, the advantages that can be offered by microcontroller-based systems will be presented.
Figure 2-3: Existing applications of microcontroller-based system

- Automotive combustion control
- Photocopier
- Camera
- Toy
Compact design with reduced cost

For other types of programmable digital devices, all the functions for I/O data communication, timing and counting functions are achieved by adding external specialised devices. The microcontrollers were embodied with peripheral circuitry augmenting their capability, and this was one of the reasons why microcontrollers were so widely used. Due to these functions, the use of microcontrollers can lead to a compact design process and reduced system cost since relatively less interfacing design and components are required [13-18].

Improved system performance

In the past, a control system was realised by analogue electronic components. However the control results varied as they were deeply affected by the tolerance of these components. Complex designs such as intelligent control are impossible to implement with the traditional control components, since the gap between design and implementation is too difficult to cross. Furthermore, the cost of implementation is usually high. With microcontrollers, complex designs can be implemented by software programs to achieve the desired goal [19-23].

Reliability

Before the digital era, system control was accomplished by many analogue components, such as resistors, capacitors and inductors. For such a system, the number of components required is high, and performance is affected by the quality of components used. For microcontroller-based systems, the control is implemented in software, and no hardware component is required. Accordingly, the performance of digital control systems is very stable and not critically affected by the components used. Accordingly, it leads to a reduced number of applied components. This increases the reliability of the intended systems [24, 25].

Since microcontroller-based systems can provide so many advantages, the applications of microcontrollers are emphasised, as well as the development process. The applications of the target controller are fully controlled by its corresponding software designs. As the capability of microcontrollers increases, the attention paid to
software development also grows among system development. Software development for microcontrollers will be explained below.

2.2. Software development for microcontroller

A general system design comes up with many specifications. The specifications require analysis to identify the actual requirements of the intended system. While implementing an intended system design, the considerations are on the part of fulfilling system requirements by functions. Afterwards, designers can develop a prototype system that contain all the functions required to meet the requirements. Hence, the system requirements are treated as the design principles for designers to evaluate the possible elements and to examine if they are qualified for the system. Then the designer can select the appropriate equipment to compose the desired system.

The same situation exists in microcontroller-based system development. The first action involves analysing and specifying the requirements from the proposed system. The intended solutions to meet system requirements can be either hardware-based or software-based functions. Afterwards, the intended functions performed by the microcontroller require corresponding software to activate it [26]. The hardware part of a microcontroller-based system is usually built before software development, as the hardware is used as the testing bed for software.

Software programs are the results of crossing conceptual design to solve the intended problems. For the development of software, it is not merely a thing; it is a process [27]. The process is composed of several activities, such as identifying system specifications, software requirements, software design, program coding, software testing and maintenance (Figure 2-4). Designers have to identify the requirements from system specifications and then convert these requirements into software programs. Knowledge and programming ability are both required in this process [28, 29]. Currently, the most frequently used methods for microcontroller software was mentioned in [30]. Later on, the activities involved in software development process will be introduced.
2.2.1. FULFILLING REQUIREMENTS – TRANSLATING DESIGN INTO FUNCTIONS

After system specification requirements, general schemes of the intended systems were developed. Designers have to decide whether to use hardware or software-based solutions during system implementation according to system design principles, such as performance/cost ratio [20], as the product cost increases if external components were used to implement the design. Therefore, the qualification of hardware and software solutions for system implementation is achieved based on these principles. On the basis of hardware, the functions performed by software can be defined. The contents of the proposed software include functions to perform and their corresponding performance.

2.2.2. REALISING FUNCTIONS - CODING

Once the desired functions performed by software have been designed in sufficient detail, these functions need to be realised by using a programming language which can be translated into machine code for the target microcontrollers [31]. Traditionally, assembler languages have dominated the software coding of microcontroller for a long time. Recently, the C language can also be used to program microcontrollers.
The source code programs require the assembling/compiling process to produce the machine codes which can then be executed by the target microcontroller. The produced programs required tests to evaluate their performance.

2.2.3. SOFTWARE DEBUGGING AND TESTING

In order to make sure the software program will perform the proposed functions, testing is required. The purpose of software testing is to examine the developed program for proposed performance. If the produced program did not perform its proposed performance, a debugging process is needed. Software debugging includes removing coding errors and logic errors from the programs.

The activities mentioned above are the core activities of software development, and the final result of software development is known as the application program. The most notable issues during microcontroller software development are programming language and development environment. The description of the current software development will commence in Chapter 3. The programming languages for microcontrollers will be described in the Section 2.3.

2.3. Programming languages for microcontrollers

Software designs eventually need to be coded into programs by programming languages, since the software programs need to be compiled/assembled into machine codes for the target microcontrollers. Currently, there are two types of programming language that can be used to program microcontrollers. Each programming language has its own instructions and rules, and the programming languages also vary in complexity. In the past, the programming language for microcontrollers was dominated by the assembler language, as it was the only available language. For the instructions of assembly languages, they are specific to microcontroller hardware architecture. Therefore, the programming of different microcontrollers may require the use of different assembly instructions. This increases the difficulty of microcontroller software development. As technology advances, a medial level language, C, can also be used to program microcontrollers. Due to the characteristics
of this language, different advantages are offered. The description of these two languages will be presented below.

2.3.1. C LANGUAGE

C, a kind of medial-level language, is currently available for developing microcontroller software, and its usage is increasing. The reason why designers are switched from assembly language to C is to avoid existing problems of the assembly languages, such as coding difficulties and takes up a lot of time. Generally, the medial language can offer a variety of logic control and data structures. For low level programming language, assembly language, the developed program can accept information in symbolic language and converted it into binary machine code. This procedure, of course, still requires many hours and is of little improvement over writing a program in machine language [32]. As a result, medial-level language is easier for designers to develop software program in a shorter time. Benefits offered by C language are:

**Increased software productivity**

Productivity in software refers to the amount of developing time and the cost for the development process. The medial level language programs can be completed sooner than the equivalent assembly language programs. This is a significant advantage in software development.

**Improved software quality**

Software quality refers to the design-related issues. In order to produce software programs which are modular and easy to test and maintain, it is more desirable to use programming languages supporting program modules and constructing structured control, since these issues aid the productivity. In addition, the developed programs should be understandable. Assembly language programs are very difficult to understand in terms of determining what the functions are supposed to be performed by the programs. On the other hand, medial-level languages support the writing of high quality, understandable software.
The aim of software development is partially to achieve easy-to-understand and easy-to-modify programs, and these goals can simply be achieved by using C (Figure 2-5). The C program is composed of syntax, and the usage of the syntax can be easily handled by designers. Therefore, the development of software while using C language could be achieved easier than assembly languages.

Figure 2-5: An example of C type program

```c
#include <P17C756.h>
#include <int16.h>
#include <timers16.h>
#include <pwm16.h>

#define Device_CLK 32000000  // set the clock for the microcontroller

void main(void)
{
    Disable();  // Disable all interrupts
    DDRB=0;     // Portb is all Output
    PORTB=0;    // Initialise outputting lows
    SetDCPWM1(512);  // Set the duty cycle for PWM1 as logic 1
    OpenPWM1(0xff);  // Set the PWM period to full scale:255
    OpenTimer1(TIMER_INT_OFF&T1_SOURCE_INT&T1_T2_8BIT);

    while(1){}  // Running forever
}
```

2.3.2. ASSEMBLY LANGUAGE

The reason why assembly language is called "low-level" language is because it is the closest language set to machine code specific to the microcontroller (Figure 2-6). Assembly language can give designers a large degree of control over the digital signal in the digital control systems. This is particularly useful where the software must interface directly with hardware and may require signal manipulation down to single bit level. The software, which is programmed in assembly language, can produce very fast and efficient operation code and save memory space. These factors could be critical in some microcontroller-based systems.
Chapter 2 - The Development of Microcontroller Applications

Figure 2-6: An example of Assembly program

```
LIST p=17C756
INCLUDE P17C756.INC
ORG H'00'
GOTO START
ORG H'30'

START
BSF CPUSTA, GLINTD ;DISABLE ALL INTERRUPTS
MOULB D'0'
CLRF PORTB,1
MOULW B'00000000'
MOUWF DDRB
BSF PORTB,1 ;SET UP THE DIRECTION
MOULB D'2'
MOULW H'FF'
MOUWF PR1
MOULW H'80'
MOULB D'3'
MOUWF P10CH
;ENABLE PWM MODE
BCF TCONH, T16 ;TIMER1 IS A 8-BIT TIMER
BCF TCONH, TMR1CS ;USE INTERNAL CLOCK
BSF TCONH, PWM1ON ;ENABLE TIMER1 AS PWM1'S TIME BASE
BSF TCONH, PWM1ON ;SWITCH PWM1 ON

;IDLE LOOP
REPEAT
GOTO REPEAT ;RUN FOREVER
```

However, there are some disadvantages with assembly language. While using assembly language to compose software designs, designers are required to have detailed understanding of the instructions, architecture design of the target controller, and possible knowledge of the interfacing devices and applied hardware devices. Moreover, assembly language is a very microcontroller-dependent language, so it is almost impossible to use a set of instructions to program different microcontrollers. Accordingly, it is much more difficult to structure, document and read an assembly program than medial level languages.

While a software design is implemented by assembly language, the common considerations of the intended system are:

**Speed of execution is critical**

As a result, the compiling process is required to create the machine codes for the programs programmed of medial-level language. However, the machine code
created by the compiler for medial-level language is usually bigger than the program produced from the equivalent assembly language. Since the program is bigger, it will run slower and take more memory space [32, 33].

**Memory size is a factor**

As mentioned above, the program created by the compiler for medial-level language is usually longer than the program composed by assembly language. Hence it will require more memory space when the program is loaded to the microcontroller [34].

Generally, designers can use the C language to improve the software quality due to these natural characteristics, such as readability and modification. However, these advantages can be overshadowed by assembly language as proper comments are added while developing the intended software. In addition, the machine code created from the C program needs more memory space due to the compiling process, and it also causes slower execution speed. Since the executive time and memory size are usually the critical conditions while composing a microcontroller-based system, this is why the current microcontroller software development is still dominated by assembly languages. For this reason, this research will focus on the use of assembly language and the following discussion of software development for microcontrollers will be based on the usage of assembly language.

**2.4. Summary**

The use of microcontrollers is becoming more popular due to the wide-spread applications of Mechatronics. In order to apply microcontrollers into systems, new design methods evolved from traditional engineering design were developed. For microcontroller-based system design, software design plays an important role, because most of the functions are implemented in software.

With increased capabilities, microcontrollers are able to perform larger and more complex functions. This means that software development becomes perhaps the most important activity in the microcontroller-based system development. Proper methods
aimed at reducing the complexity of software development and improving software quality were formed.

Two types of language, assembler and C, are available for microcontroller programming, and each of them provides different advantages. Currently assembly language still dominates microcontroller software programming as it offers the essential benefits. Microcontroller programming is accomplished in integrated development environment, which encompassed several tools. These tools allow software designers to compose, test, correct, modify and create the proper program to perform the desired functions. Generally, the development of microcontroller system has been introduced in this chapter. In the following chapter the issues relating to software design will be mentioned.
Considerations of microcontroller software development

Although the development process of the microcontroller-based systems was mentioned in the previous chapter, the required work has not yet been introduced. It is clear that most emphasis on microcontroller system development is placed on software, since the intended performance of the target controller is directly controlled by the software programs. In order to achieve quality software development for microcontroller-based system, different factors need to be taken into account. In this chapter, the required effort to compose a microcontroller system will be shown and shortcomings caused by current existing development tools will be revealed.

3.1. Effort required while composing microcontroller-based systems

The microcontroller-based system development process was presented in the previous chapter, section 2.2. Due to the possibility of trade-offs between software functions and hardware components, the software of microcontrollers is getting more complicated than before [35]. In order to mix up the hardware and software balance for the proposed systems, more effort is required. The inputting of effort takes up time in the development cycle and might affect the quality of design results [36]. The required effort to develop microcontroller-based systems will be introduced in the following sections. Since different effort is required at different stages of system development, the introduction of the effort will be shown in stages. Firstly, considerations during the system level design will be mentioned, followed by software design, and the debugging phase.
3.1.1. COMPOSING SOFTWARE DESIGN

External hardware components can be found in a microcontroller-based system, because the microcontroller-based system is not purely a software system. The purposes of these external components such as switches are aimed at linking the digital microcontroller to the real world [37]. In the past, peripheral functions can only be made available by adding hardware components. Currently, similar performance can be achieved through hardware-based software solutions. This is because of the advance of technology and more integral circuitry has been added onto the microcontroller chip. This means that more functions are now available with the microcontrollers through the software implementation.

Between the microcontroller and the external components, there is usually a need for interfacing devices. The interfacing devices used are dependent on the functions performed by the microcontroller and the external devices. A microcontroller-based DC-motor controller can be illustrated as mentioned in the above situation. Microcontrollers can achieve speed control by the variation of offered voltage through Digital Analogue Converter (DAC) or the power ON/OFF time period of motor through H-bridge with Pulse Width Modulation (PWM) function. Therefore, it is essential to understand the target microcontroller and the relationship between the controller and the external devices before composing the intended microcontroller-based system.

3.1.1.1. Understanding the microcontroller

The availability of microcontrollers is currently high. This means that many different types of microcontrollers are available to be adopted as the target controllers for the intended systems. Several criterions inferred from system requirements are used as development principles to evaluate the candidate microcontrollers. Often, the principles include cost, number of I/O pins, and required peripheral functions [38]. These basic principles help the designers to select the appropriate microcontroller for the proposed system.
Once the target microcontroller has been selected, the designer needs to devote a period of time to understand it, in order to gain full control of the microcontroller [39]. The current microcontrollers are usually combined with several options of integral circuitry capable of performing peripheral functions, and it is essential to be aware of the basic relationships between functions and their usage. Furthermore, the performance of these peripheral functions is directly controlled by specific mechanisms. Accordingly, the relationship between control mechanisms, function and performance are required to be understood (Figure 3-1).

Figure 3-1: A simplified scheme of peripheral functions with control mechanism inside a microcontroller chip (PIC17C756)

In order to optimise the potential of a microcontroller, designers also have to be aware of the available resources inside the microcontroller. The resources here include
peripheral functions and memory size. Peripheral function refers to the available options inside the microcontroller chip. Memory includes both read only memory (ROM) and random access memory (RAM). The ROM size means the possible length of the program, and the RAM size indicates the number of possible locations which allow designers to access through the execution of the program. RAM memory can be further distinguished into two types. The first type is the general purpose register (GPR), and the second type is the special function register (SFR). GPR means the free data space which allows designers to store and retrieve data while the program is running. SFR is usually associated with embodied peripheral functions or actions taken by the central processor unit (CPU). Since these two types of registers are the basic elements of a microcontroller software program, it is important for designers to handle the related registers for the proposed system.

3.1.1.2. Understanding the relationship between the microcontroller and other components

A special circuit that fits between a microcontroller and its external components for the integration of the whole system is called the interfacing device. For the output signals of the microcontroller, they are relatively weak, and can hardly drive any external device. In addition, the input signals from external devices are usually analogue and random, which might need to be rectified to be used by the controller. In order to match the microcontroller and the external devices, interfacing is required in the systems. The functions of interfacing devices are to allow the microcontroller to receive data contents from outside or control the external components through the output data or peripheral functions. Although I/O functions were already embodied in the microcontroller chips, it does not mean that the control of external equipment could be achieved by direct wire connection to the microcontroller chip. The interface used for the same external device might be varied. For example, the interfacing devices between a microcontroller and a DC-motor can either be a digital analogue converter (DAC) or a H-bridge, and the decision is usually made due to the specifications of the intended systems.

A hardware prototype is usually built after the intended system has been partitioned into the hardware part and software part. The prototype is then used as a testing bed
for the software design of the system. The hardware prototype includes external devices and interfacing circuits. The main purpose of interfacing is to produce and keep data flow heading to the right destinations [40-42].

Although interfacing mainly indicates the linkage between the microcontroller and external components, associated conditions along with the interfacing devices require the target controller to perform corresponding functions to fulfil these conditions. Therefore, the activation of the interfacing devices is also part of the microcontroller software. It is therefore important for designers to handle the required interfacing while composing the proposed system.

3.1.2. SEARCHING FOR RELATED INFORMATION

Software development is recognised as a very knowledge intense activity, and the required knowledge may come from multiple resources. Due to application purposes, a number of issues, such as function configuration and system control might be encompassed in a program. Although the part of system control is also a major issue in software development, it is, however, related more to the use of the instructions to achieve the intended control sequence. Therefore, the part of logic control will be discussed in the following section: assembly language.

A peripheral function is configured through the control of associated mechanisms located inside the special function registers (SFR). Therefore, the performance of the intended function is related to the data setting of these related SFR. As the data setting of these SFR is a sophisticated task, vendor offered information is frequently needed. The details of the control mechanism will be presented (Figure 3-2).
Chapter 3 - Considerations around software development for microcontroller

Figure 3-2: Usages of the mechanisms inside the SFR

<table>
<thead>
<tr>
<th>INTEDG</th>
<th>TOSSE</th>
<th>TCSC</th>
<th>TOPS3</th>
<th>TOPS2</th>
<th>TOPS1</th>
<th>TOPS0</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
<td>U = Unimplemented, Read as '0'</td>
<td>D = Default Value</td>
<td>P = Value at POR reset</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. INTEDG: RA0/INT Pin Interrupt Edge Select bit
   - This bit selects the edge upon which the interrupt is detected
   - '0' = Rising edge of RA0/INT pin generates interrupt
   - '1' = Falling edge of RA0/INT pin generates interrupt

2. TOSSE: Timer0 Clock Input Edge Select bit
   - This bit selects the edge upon which TMRO will increment when TCSC = 0 (External Clock)
   - '0' = Rising edge of RA1/TOCKI pin increments TMRO and/or sets the TOCKIF bit
   - '1' = Falling edge of RA1/TOCKI pin increments TMRO and/or sets the TOCKIF bit

3. TCSC: Timer0 Clock Source Select bit
   - This bit selects the clock source for TMRO.
   - '0' = Internal instruction clock cycle (TCY)
   - '1' = External Clock input on the TOCKI pin

4. TOPS3-TOPS0: Timer0 Prescale Selection bits
   - These bits select the prescale value for TMRO.

<table>
<thead>
<tr>
<th>TOPS3</th>
<th>TOPS2</th>
<th>TOPS1</th>
<th>TOPS0</th>
<th>Prescale Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td>1:1</td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td>1:2</td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td>1:4</td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td>1:8</td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
<td>1:16</td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td>1:32</td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
<td>1:64</td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
<td>1:128</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td>1:256</td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Unimplemented: Read as '0'

The setting of a peripheral function includes not only the enabling of the function but also the configuration of its corresponding performance. Very often, the performance of the function is controlled by several factors, and changing these factors can produce a different performance. This case can be exemplified by the manipulation of a PWM function. The waveform of PWM is decided by the ON/OFF ratio of a switching mechanism. The total length of a PWM waveform is controlled by the data setting of Period register (PR), and the ON period is controlled by the data contents loaded into the Duty Cycle register (DC). Both PR and DC count from zero to the number loaded inside the registers by specific pulses, which can be internal or external. For the counting numbers to reach the data inside DC, the PWM waveform is then turned off. A whole new PWM waveform will start after the counting number reaches the data inside PR (Figure 3-3). A proposed PWM waveform can then be produced as the factors mentioned above are manipulated.
Fulfilling the intended performance by proposed functions is a critical task during software development. To achieve this purpose, vendor offered information was usually required. However, the use of the information can be difficult if designers do not have enough experience. This is because the presented data focused on technical issues and a lot of very detailed information was listed. To be able to use the listed details, knowledge and experience is needed.

3.1.3. COMPLETING SOFTWARE DESIGN BY ASSEMBLY LANGUAGE

Assembly language is the closest language to machine code, and therefore provides better execution efficiency than other programming languages. The instructions of assembly language are usually direct actions to predefined data or the data contents inside special memory addresses. The actions can either be byte-oriented or bit-oriented. Furthermore, the use of assembly languages requires software designers to be familiar with the target microcontroller since the instructions are microcontroller-oriented. Overall, the function and related issues of the instructions need to be taken into account while using assembly language (Figure 3-4).
Each assembly language has its own limitations such as label length, and allowed character. Also the sets of instructions vary according to microcontrollers. Designers, therefore, need to understand the corresponding instructions of the target microcontroller. In addition, they also need to know how to realise the software design by these instructions. In other words, the designers have to know the development aims in terms of data and action.

Since the available instructions of microcontrollers vary in mnemonic, description, and function, it is therefore important for designers to search and handle information related to the instructions. Otherwise, designers will encounter problems such as invalid instructions or missing operands while composing their software programs.

### 3.1.4. PROGRAM MODIFICATION

The real program that can be executed in microcontrollers is a machine code program, and the machine code is the assembled result of the developed source program. Once the machine code has been built, designers can then simulate or run it on an in-circuit
emulator (ICE). The functionalities of simulator and emulator are similar, and their usages were mentioned in 3.1.2.1.3. Although these two tools are capable of imitating the performance of the target controller, the discovery of errors misleading the functions requires further work from designers. However, a slight difference between the functionalities of these two tools still exists.

Since a simulator is integrated within current software development environment, simulation costs are cheap. However, it also has its disadvantages. One of them is the speed of execution. Since simulation is achieved by the host CPU to simulate the actions of target microcontrollers, the speed is fairly slow. Another disadvantage is that the simulation cannot perform the peripheral functions. This is why emulators were introduced. The emulation is achieved by a replacement of target microcontroller and monitoring circuits that continuously feed signals back to the host machine. The programs can be run on the emulator's RAM as they are being developed and tested. Through the emulator, the host machine is able to capture and store the feeding-back information for display at a later time.

Both simulators and emulators will interpret the monitored machine-level information to assembly level. If the designers find that the programs did not perform as they were designed to, they need to modify the target program. This modification work for the target program can include producing the intended machine code and detailed function justification. These two issues will be explained below.

3.1.4.1. Producing the intended machine code

The first appearance of refining the instructions of the target program is usually due to failure of assembling process, since invalid instructions or data is discovered at this stage. The end of assembling means that only the corresponding machine code is produced. In other words, a successful assembling process does not mean that the desired functions can be performed by this program.

The refinement of the coded program relies heavily on the designers' experience, although hints of errors found were offered by the assembler. However, the simulation
or emulation was carried out based on the machine code. Therefore, producing the corresponding machine code is the first step of program modification.

3.1.4.2. Justifying the hardware function

Although the simulator and emulator can show the data flow of the target program for designers to debug, the required amount of work required for program modification is similar to the workload that designers devote to develop the software. During the program modification process, designers have to understand the performance of the target controller themselves from the sequence of the execution program as well as the data contents in special function registers. Since the simulation is only capable of presenting items such as symbolic names, variables, labels, or lines as well as the contents inside, it does not provide any advice for designers to modify the program. The work for refining the target program is done directly by the designers.

The intention of the microcontroller software development was not only for the control of sequential system events but also for the use of peripheral functions. The performance of a peripheral function is difficult to evaluate by simulators. Therefore, the functions are usually tested by an in-circuit emulator (Figure 3-5). The actual performance of the peripheral functions may need other equipment such as an oscilloscope to show.

Figure 3-5: The PIC In-Circuit Emulator with hardware testing components
The peripheral functions are performed by the integrated circuitry on the microcontroller chip. The sophisticated functions are configured through the setting of control factors in software. The functions are actually controlled by specific mechanisms, and these mechanisms are directly manipulated according to the data contents of the special function registers. In order to configure the functions successfully, designers have to know the related issues such as the names of special function registers, and the locations of the registers.

Figure 3-6: The schematic diagram of emulator with peripheral boards

If the target programs did not perform the desired functions, designers need to understand the problems, whether the problems are malfunctions or mismatched performance. No matter which problems are found, related information is required to accomplish the program modification, as the modification of the program is as complex as the implementation of the function. In order to check the actions taken by
the target controller, peripheral boards such as switches and LEDs are often used (Figure 3-6). These peripheral boards help designers while testing the performance of the developed software programs.

Both simulator and emulator are auxiliary tools for designers to examine the actual performance of the developed programs, but they cannot point out the problems while function mismatching is occurring. These problems need more direct action from designers. Designers need to obtain the experience to discover the causes of the function mismatching and then correct the programs.

3.2. Disadvantages of current development process

Generally, the design and implementation process for the microcontroller is a designer-oriented work, as most development work needs to be done manually by designers. The work done during the development process is aimed at producing the intended software program [43]. Therefore, the process control as well as design quality become difficult to manage. Later on, the disadvantages from the current software development process will be examined.

3.2.1. LACK OF INFORMATION RESOURCES

Adequate supporting information is required through the whole process of software development [27, 44, 45]. During the design stage, designers need the information to fulfil required specifications with available functions. Designers then use different information to produce the function in software by instructions. Information is still required for designers to debug, and modify the target program.

It can be said that the software development is a gathering and utilisation of domain-specific knowledge, and to fulfil different requirements, information is usually required. Information acts as a bridge, which allows designers to cross the gap between requirements and functions. As mentioned in section 3.1.2.2., the required information and knowledge during software development are usually domain-specific. In order to complete the process, multiple resources of supporting information should be included in the process.
Although vendor offered information is available, the needs for appropriate information still exist. This is because the offered information was aimed at offering technical details of the target controller, but the aspects of applications were hardly mentioned. Accordingly, the offered information could be difficult to use due to the presentation of information. The situation is as if there was no available information to designers. Furthermore, the required information for different microcontroller-based systems is different in detail. Hence, the complexity of current software development is high and also many problems are encountered.

### 3.2.2. INSUFFICIENCY OF SOFTWARE DEVELOPMENT EXPERIENCE

Experience was revealed as an important factor during system development. For experienced designers, they seem to fully use available knowledge to decompose and recompose the design problems, while less experienced designers might have problems in doing. In [46], it was indicated that there was an obvious difference between experts and novices in the ways that they approach their problems. In addition, the design experiences are able to increase the designer’s ability in handling the structure of the problems. The difference in time and decision-making can be seen in [47]. Due to these reasons, lack of experience can cause inefficiency in the development process.

Microcontroller software development is seen as a knowledge-intense activity, and the knowledge used is widespread [42, 48, 49]. It is indeed a problem for designers to handle the application, implementation and coding knowledge related to the target microcontroller unless the designer is well experienced. Experience is accumulated throughout many different aspects, such as system composition, software/hardware codesign, and software programming. For each aspect, it takes designers a huge amount of time to obtain the essential experience to manipulate the knowledge. Accordingly, the experience can be a key issue to a successful design.

Experience in software design refers to the capability to define problems and search the related information to solve these problems. For a good microcontroller-based system design, the microcontroller should be integrated into the design from the
beginning and not simply an after-thought add-on [50]. In order to do so, the engineers must be capable of addressing a variety of basic issues in electromechanical system design and implementation. Thus, many universities around the world have carried out a wide variety of curricula related to Mechatronics [3, 51-53]. All too often, the hands-on mechatronics, usually a microcontroller-based equipment, is incorporated to enhance the curricula.

Without experience, the designers will not even know how to start a design. Experience is used as a guideline in a microcontroller-based system development. Experience can guide designers to go through each development activity, although they may still need information to solve the problems encountered. The range of the required experience is not limited to a few areas, as the purposes of a microcontroller-based system do not always remain the same. Therefore, the efficiency and quality of current software development is difficult to control.

3.2.3. LIMITED ASSISTANCE FROM DEVELOPMENT ENVIRONMENT
Choosing a particular microcontroller with highly available software development tools is considered an important criterion [54], since the importance of software development among the whole system is increasing. However, designers have only received limited assistance while developing the intended systems with the current existing software development tools. The work of addressing the requirements from system development is usually done manually by designers.

Some software tools have been developed to assist the system implementation process, and the most important function of these tools is to guide designers to select the proper target microcontroller for the proposed systems [55]. With this software tool, proper microcontrollers will be listed after designers have set their requirements. However, it still cannot fully fulfil designers’ needs for the whole development process, as the selection of the target microcontroller is merely one activity during the process. Another software tool was described in [56], and its main functions targeted simulation. It allows designers to check the data contents of registers through a graphical interface during the simulation stage. However, it could only provide some
aids for simulating, which is just a part of the whole design and implementation process.

Due to the increasing complexity of microcontroller-based system, the range and capabilities of the required tools for software designers are rapidly changing. Although current available software tools allow designers to implement designs, there are a number of drawbacks, especially in terms of the effort required while composing and implementing system designs [57]. For existing software development environment, it provides enough tools for designers to compose their designs, but the usability is relatively low. The usability indicates the ease of use and the effort required to operate a system. As the offered help from current software development environment is very limited, the required activities were carried out by designers manually. Therefore, further workload was added to the process.

3.2.4. UNFAMILIARITY WITH CODING INSTRUCTIONS

It is usually difficult for novice assembly language programmers to complete development tasks, although they might have other high-level language programming experience [30]. The programmers need to achieve every design aim of the system through the manipulation of instructions. The programmers have to manage the resources such as special functions and memory for data storage. In addition, they also have to be aware of the required system control strategy for the system. It can be concluded that the designers have to be familiar with the details of the target microcontroller before composing the intended software.

The performance of a microcontroller software program was achieved by the configuration of proposed functions, which was done through the correct use of the instructions, as each instruction has its specific fields such as operation, and operand. To implement the desired design content, designers were required to manipulate resources such as special function registers and specified memory locations. Knowing and keeping track of the resources is mandatory for the programmers. Usually, different microcontrollers come with different architecture and require different instructions to program. Each microcontroller has its own specific limitations such as label lengths, allowed character, etc. The difficulty involved in the programming of
Chapter 3 - Considerations around software development for microcontroller

microcontrollers is further increased due to these factors. Accordingly, the effort needed to complete the intended software development was further increased.

It can be concluded that most of the effort required to complete the microcontroller software development was used to solve the problems appeared during the process rather than fulfill the implementation solutions. Therefore, the research was aimed at reducing the effort required for software development and achieving the improvement.

3.3. Summary

It was pointed out that a microcontroller should be considered at the beginning of a system design, rather than after. In order to achieve this aim, understanding of the target microcontroller is required, because it will allow designers to use the optimal potential of the controller. Also, the awareness of the interfacing between the microcontroller and external components is necessary, since the microcontroller software program is normally used to interact with and control external devices.

For a microcontroller-based system design, the part of software development plays an important role. Software development is for designers to bridge the gap between the system requirements and functions. The current software implementation is completed under the software development environment in a PC-based host machine. Designers have to be aware of all the essential issues contained inside the proposed software and then use the appropriate programming instructions to develop the program. Further development work such as simulation or emulation might be included inside the development process in order to refine or modify the created programs to achieve the proposed functions.

Although basic tools are already included in the software development environment, some shortcomings have appeared. These problems might be caused due to assembly languages, software development environment, supporting information and experience. Often, problems appearing during software development are caused by designers who do not have the ability to control these issues.
Chapter 4

Review of previous work

Software design is suffering a crisis in both quality and quantity[58]. Similar problems are also found in microcontroller software development, since the size and complexity of microcontroller software programs are continually increasing. In order to effectively relieve the pressure, new methods and technologies have been applied to improve current software development process. In the following sections of this chapter, the improvements made to the current system development process and how designers will benefit from them will be introduced. Although some improvement has been made by these solutions, most of the difficulty of microcontroller software development remains. Therefore, an advanced software development system is thus proposed.

In order to improve the microcontroller software development effectively, appropriate technologies aiming at augmenting the capabilities of software development tool was proposed. The proposed technology was selected after considering of the intended proposal, and it was the methodology adopted by the research to build the advanced system for microcontroller software development. The criteria and process of evaluating this technology will be revealed in this chapter.

4.1. Advanced solutions for system development

The problem of discarding the complete automatic systems is currently faced by many companies, because these systems are unable to cope with changes. This is a cause of big losses to these companies. A new component based (CB) approach trying to build machines was mentioned in the recent research. In this new approach, the various components of the production system, such as the actuators, sensors and control software are defined as reused, extended or reconfigured components (Figure 4-1).
Chapter 4 - Review of Previous Work

The components of reused control software can be reconfigured by process engineers directly without the need for conventional programming under the new system. The research aimed at developing a common model of the machine and its associated control logic which are based on investigation and evaluation of the reconfiguration and reuse of machines throughout their life cycle. Through the use of this model, the designers (machine builders, process engineers, programmers) will be able to consistently observe the evolution of automated systems at every stage, i.e. design, build, evaluation and diagnosis. The benefit of the research is to offer system designers solutions for integrated systems, and the reuse of the various components. However, the CB system focused on the system integration. The part of control software was achieved by reusing and reconfigure previous results rather than developing the proper software programs.

Figure 4-1: A modelling picture for component based system integration

![A modelling picture for component based system integration](http://www.lboro.ac.uk/departments/mm/research-groups/manufacturing-systems/index.html)

A system codesign environment was presented in [59], which allows designers to combine the performance and behaviour models during simulation. This codesign environment helps designers to implement design details for intended embedded systems, but the focus of this environment was on the development of the prototype of the proposed system, not the development of software to cooperate with the controllers.

MatLab is an interactive integrated environment for numerical computation and data visualization and it is used extensively by control engineers for analysis and design. This is because designers can use this tool to simulate and analyse the mode-based
design of control systems, and eventually achieve the aims of fast prototyping and code generation for embedded systems. During the analysis process, graphical elements are used to represent the required components of the system. In addition, the interactions of components are allowed to be set in detail. Therefore, MatLab is a useful tool for designers to design and implement their intended control systems. Although the applications of microcontrollers is frequently related to the implementation of control, the use of the embodied functions also plays an important part. For MatLab, the enabling of these functions and then using these functions as the solutions to achieve the system control aim is not actually emphasised by MatLab.

Microcontroller-based systems are usually treated as the integration of software and hardware, since both parts are required. From a system design's point of view, hardware and software need to be firstly partitioned and carried out independently, and then the integration work can commence. Hardware includes the microcontroller which is usually built as a prototype for software performance evaluation. In order to achieve fast prototyping for such a system, a platform based on microprocessor architecture for software part and a configurable FPGAs based board for the hardware part was developed in [60]. A similar idea was introduced in [61], as combining programmable hardware with programmable processors capitalises on the strength of both hardware and software. However, the focus of these systems was on the interfacing between the microcontroller and the external components. The part of microcontroller software design and implementation has not yet been emphasised. The other development environment was mentioned in [31] which targeted the development of microcontroller chips rather than the software to work with it.

The proposed software programs of the intended systems were built under specific development environments, and the purpose of these software development tools is to meet the needs of reliability and ease-of-use required by software designers [62]. Therefore, many solutions were then proposed to benefit the software development by improving the development tool. Thus, a graphical interface was frequently applied. In [63], a new system for real time embedded control system design was presented. The other design system to apply graphical interfaces was [64], which targeted mechanism design. Although the efficiency of these two development systems were
improved by applying graphical interfaces, both systems are not targeted on microcontroller software development.

Another improvement found in an integrated development environment for microcontroller-based systems was introduced in [56]. This IDE was aimed at improving the drawbacks of current simulator and command-line interface by implementing an easy-to-use graphical interface. This is because simulators are very useful tools for software debugging, and the system development is heavily dependent on the completion of the software. However, this IDE has similar drawbacks as other development environments, since they do not help designers through the microcontroller software design and implementation process.

4.2. Improvement from system development methodologies

An appropriate development strategy is an important factor in the success of a system development [65]. Two design methods are currently widely used for microcontroller-based systems; one is a mechatronic system design and the other is a hardware/software codesign. The main difference between these two is how they treat the target controller within the development process. The applications of microcontrollers are not considered but system control is finally achieved by the implementation of microcontrollers in mechatronic system design. In hardware/software codesign, microcontrollers are placed in the core of the proposed system, and the design method tends to balance hardware and software to meet the system requirements. More details will be given in the following sections.

4.2.1. MECHATRONIC SYSTEM DESIGN

In order to explore the implementation solutions for mechatronic systems, a system was usually decomposed into several subsystems before the actual implementation including the applications of the target controllers [66]. Each subsystem can represent either a function or an interaction. In addition to these subsystems, an architecture template is used to represent the whole system, and these subsystems are located inside this template. Therefore, the complexity of development process can be
Chapter 4 - review of previous work

reduced by choosing an appropriate system architecture or configuration to implement a given set of specifications [67].

Figure 4-2 explains the mechatronic system design for a motor control system. In order to achieve the purpose of motor speed control, a control strategy such as PID control is needed, and a proper solution for speed evaluation is also required. In addition, the proper method to drive the intended motor needs to be decided. The motor driving related issues include voltage control and interfacing devices. Eventually, the motor control system can be decomposed into four subsystems. A microcontroller can be chosen as the proper solution to implement some of the intended subsystems, for example the control strategy and voltage control. However, the rest of the two subsystems, encoder and interface, still require external components to implement them.

Figure 4-2: The decomposition of Mechatronics system

The method of mechatronic system design is applied in the early stage of system development, and the target microcontroller is not only aimed at replacing the control core. Therefore, the system requirements and implementation issues are fully addressed by this method, and some of the system needs can be fulfilled by the capability of the microcontroller [3, 38].

The applications of microcontrollers inside a mechatronic system can encompass several functions, which can eventually be performed using either software or hardware. The decision to implement the subsystem is dependent on the system requirements, such as cost-performance ratio, or performance estimation. Under the system architecture, a software component can simply be viewed as another hardware
component in this system. By treating software and hardware components as basic elements of the system, the complexity of system development could be effectively reduced. The complexity management is considered as an important task during system development [36]. In [68], it was believed that mechatronic system development method could supply flexible and low-cost design in a short design cycle.

Mechatronic system-level design method does not exactly fulfil designers' needs for microcontroller-based system development, as software development is rarely described. The focus of mechatronic system design is on finding the appropriate implementation solutions by decomposing the whole system into minor parts. Therefore, building a framework for implementation is the most important issue [69]. In fact, the development of a microcontroller-based system relies heavily on the software, as the final performance of the target controllers was fully controlled by the software programs. While considering mechatronic system development strategies, there is no widely accepted tool available to support the designers in the definition of a functional specification and then mapping the required elements onto a system architecture. Therefore, designers usually rely on manual techniques, relying on experience to allow them to explore alternative solutions between hardware and software. The development of the intended software for the target controller was not considered.

4.2.2. HARDWARE/SOFTWARE CODESIGN

The goal of hardware/software codesign is to develop systems containing an optimum balance of hardware and software components which work together to achieve desired performance and fulfil various design requirements [70-74]. This is because the design of a microcontroller-based system is related to the trade-offs of software and hardware. Such a system will be implemented in either software or hardware, which is determined by the system specifications such as cost/performance ratio. As the capability of microcontrollers increases, required functions are proposed to be implemented in software-based solutions as possible.
While composing functions in software increases development cost, moving functionality to the hardware increases product cost and these costs are incurred with every unit built. Today, the correct mix of embedded software and hardware determines the success or failure of products, because codesign can move hardware and software development closer together. Thus, this method can reduce the cost of the final integration of components. In order to achieve low-cost design, addition of any extra hardware can have a significant influence on product costs [33].

Hardware/Software codesign is the methodology developed for solving the problems of hardware/software trade-offs and it also attempts to achieve the aim of reducing design time and meeting the performance goals [75]. Basically, hardware provides the platform for sophisticated software to perform the desired functions [76].

Figure 4-3: Hardware/Software Codesign

Codesign is a concurrent and cooperative development approach that includes a fundamental component with the capability to explore hardware/software trade-offs [77] (Figure 4-3). When exploring the hardware/software trade-offs, the (hardware/software) alternative evaluation is critical. The evaluation principles can be changed according to the system requirements, such as cost or real time response. For codesign, the partition of hardware/software is considered as the most critical task.
Traditionally, the partition of Hardware/Software is finished manually and the decision is highly dependent on the designers' viewpoint. A relatively new approach to achieve the partition of Hardware/Software was described [78], and another system-level partition was proposed in [79]. In [80], a methodology was described to achieve codesign, and it adopted cosimulation and cosynthesis to solve problems of hardware/software partition and evaluation. For the traditional design method, it restricts the ability to explore hardware/software trade-offs, hence codesign can be said as the solution to break through the bottleneck of traditional design methods.

In hardware/software codesign, hardware/software partition and alternative evaluation are highly focused, since a microcontroller-based application can be expressed as a mix of hardware and software. Mapping any behaviour onto a microcontroller system involves hardware/software partition and the assignment of software and hardware tasks to microcontroller peripheral functions [81]. For this kind of system design, it is important to partition the system specifications into hardware and software implementations to fulfil the design requirements, such as cost, power consumption, and software acceleration [82-86]. Hardware provides the performance required by design constraints, and the software provides the flexibility of control. Currently, in order to improve the performance of the microcontroller-based system, the design principles were used to minimise the analogue part and maximise digital programming [37]. Since there is a wide range of design goals, hardware/software codesign can take in many forms.

Although these design methodologies help designers while developing a microcontroller-based system, the approach for the implementation of the target microcontrollers to achieve system requirements have not been advanced by using these methodologies. Thus, software development during hardware/software codesign encounters similar problems to mechatronic design, because the implementation of software was not focused in this methodology. In other words, designers can identify which system requirements needs to be fulfilled by software but these methods do not show designers how to achieve it.
Both system-level design and software/hardware codesign help designers to find the actual functions behind the system requirements with advantages such as short design cycle time. However, in the implementation stage, these design methods do not offer a great deal of help, because most effort is devoted by the designers themselves. In [87, 88], they both presented different system level design methods for microcontroller systems, but they focused on the design of microcontroller chip. As microcontrollers increase in capability, they can deal with larger and more complex tasks. This implies that software development is perhaps the most important part of microcontroller-based system development. Other possible improvement made to this field was found in [89], which highlighted the insufficiency of common software development methodology. However, the required work for software development was inputted manually by designers, and the purpose of reducing workload was not actually achieved.

It is clear that a lot of work has been devoted to make improvements for the system level development process from section 4.1. and 4.2. However, the current development process of microcontroller-based system had hardly benefited from these systems, as the software development was seldom targeted. Accordingly, demand for an advanced software development environment, which can help designers to deliver the program code for the microcontroller-based system, is increasing. Later on, relatively advanced microcontroller software development environments will be described, as well as its features and benefits.

4.3. Improvement of software development tools

Although several methods related to microcontroller-based system development have been developed and applied, software development as a whole was only slightly improved. These design methods help designers to specify the system requirements, and indicate to designers some solutions. However, the process of software implementation is not directly benefited due to the appearance of these design methods, since designers have to spend most of the development process on using programming language to implement the design. The problem was mostly due to currently used software development tools. In order to improve this software
development process, two types of software tools have been built, even though they might target other purposes. The two software tools include DAvE from Infineon\(^1\) and Mindstorms from Lego\(^2\).

### 4.3.1. INFINEON- DAVE

DAvE, the Digital Application virtual Engineer, was created by the company Infineon. The DAvE is a CD-based application support tool aimed at aiding embedded system designers while designing applications with Infineon’s range of microcontrollers. The tool allows the users to evaluate and select an appropriate microcontroller from the company’s portfolio, and then access information, configure the chip and generate code for its application in C language.

Figure 4-4: A smart search platform for designers to search suitable microcontroller

---

1 Dave, Infineon are all trademarks of Infineon company
2 Mindstorms, Lego, are both trademarks of the LEGO Group of companies
Chapter 4 - Review of Previous Work

The evaluation and selection of a microcontroller chip was traditionally done manually by designers to go through the related documents, such as product menus and data-sheets. In order to accelerate the process and relieve the workload of selecting a suitable microcontroller, a tool which can indicate to designers the proper chips according to the system requirements such as on-chip peripherals and I/O pins, was included (Figure 4-4). Thus, designers can soon move on to the next stage of system design.

Figure 4-5: The architecture and available peripheral functions of the target microcontroller

DAvE not only allows designers to select the appropriate microcontroller to implement the design, but also provides on-chip peripheral function configuration of the target microcontroller (Figure 4-5). After selecting the proposed peripheral function modules, designers can utilise these functions in the intended systems by setting the control issues (Figure 4-6).
Another advantage that DAvE has offered over traditional software development environment is code generation. Once designers complete the configuration of peripheral functions, the system will create the corresponding program according to designers' intention (Figure 4-7). In addition, some files, such as the documentation file, will also be produced.

Generally, DAvE helps designers to reduce the work required during current software development, such as data collection, progressing the software development time and build software program. These issues are all critical to software designers. In addition, DAvE was constructed with a user friendly interface.
4.3.2. LEGO- MINDSTORM

Mindstorms is the name of a commercial product developed by Lego. Although it is treated as a toy, it is a kind of microcontroller application. Users can program Mindstorms with the external kits such as sensors and motors to compose their intended system with the proposed functions. Since Mindstorms was developed as a toy, suitable for 12 years or older, a special software development environment, Robotic Invention System (RIS)\(^1\) (Figure 4-8), was also announced by Lego. Due to the potential users, the RIS was developed with several features which allows kids to be able to use it to program Mindstorms. Lego Mindstorms are also used as educational tools among places [90, 91], as they are suitable to teaching a variety of issues such as programming, and artificial intelligence.

\(^1\) RIS (Robotics Invention System) and Cybermaster are all trademarks of the LEGO Group of companies
For most of the software development environments, programming instructions are used as the basic bricks to build a program. In RIS, no text based programming language is used, as some puzzle-like pieces are used to compose the program. It means that the functionality of instructions is replaced by the puzzle pieces. Hence, a completed software program under RIS contains only puzzles without any coding.

Possible system actions and conditions of Mindstorms are well defined and placed aside under RIS (Figure 4-9). Users only need to drag and drop the puzzles into the programming area, and this is how users develop software program for Mindstorms. In other words, the users have to select the appropriate puzzles to describe the whole system design. There is no need for coding. The selected puzzle might require detailed configuration in order to perform the function expected by the users (see Figure 4-10).
After the design has been implemented fully, the proposed software program can then be generated and then loaded onto Mindstorms through infrared data communication. If the developed program did not perform the proposed function, users can refine the program by either changing the detailed setting of puzzles or replacing with new puzzle bricks. Users can simply drag out the incorrect puzzles which did not perform the proposed function and replace it with a new puzzle.

The Lego RIS can be treated as a special software development tool, although it is especially designed for Mindstorms only. However, it has brought about some important points to existing software development. The "drag and drop" process to implement the software in RIS is likely to express the design intention while using instructions in a traditional text-based programming environment. Furthermore, the RIS is a relatively easy to use software tool. No special skills and experience are required for general users to control the RIS to produce the software program for Mindstorms. Also, users understand the concepts of a microcontroller-based system from the development process of Mindstorms.
4.4. Features of these systems

Two relatively advanced microcontroller software development tools have been introduced in section 4.3. Although they targeted commercial purposes, they offer great benefits to microcontroller-based system designers. These benefits were actually offered due to the features performed by the integrated software development environments. These features include fast and quality software development process, easy-to-use interface and limited information and experience inquiry. In order to present the real features clearly, they will be discussed in the following sections.

4.4.1. FAST AND QUALITY SOFTWARE DEVELOPMENT

A microcontroller software program can encompass multiple issues, such as peripheral function setting and system control strategies. In order to achieve the design aims of the intended microcontroller-based systems, designers have to collect and work out related information and eventually code the design by programming
instructions. For current software development, the process of learning and practicing was hidden inside the development process, since the completion of function setting in coding require corresponding experience and knowledge. However, learning and practicing is a very time consuming process and causes difficulty in software design.

Improvements made to software development can be found in both tools. In DAvE, an interface displaying the available peripheral functions was presented after the target controller had been decided. Designers can continue peripheral function configuration directly by clicking on the proposed functions within the interface. Also, vendor offered information was captured inside the system, and designers can select and obtain the related part from the provided information. Afterwards, the corresponding software code for function configuration was created. It can be said that much of the workload has been reduced by using these software tools (Figure 4-11).

Figure 4-11: The program developing progress of DAvE

The traditional software development process is also reshaped in RIS. No traditional coding environment exists in the RIS and users only need to drag and drop the puzzles into the programming area. The part of function setting has been hidden behind the available programming puzzles. Users only need to express the proposed design by proper puzzles, and command these puzzles to perform the desired functions (Figure 4-
According to the software development process, most of the pressure of composing a software program for proposed performance has been alleviated by RIS.

**Figure 4-12**: Drag and drop the intended puzzles to compose the proposed design

Hence, it can be concluded that the software development process in these two environments is relatively simple compared to the implementation process within other traditional development environments. Designers can therefore complete the software design within a shorter time or work out better designs rather than spending time on coding.

### 4.4.2. GRAPHICAL IMPLEMENTATION INTERFACE (GII)

Software tools are considered with functionality and usability. Functionality refers to the ability to achieve the functions, and usability relates to the ease of use. For DAvE and RIS, they both allow users to develop software designs. Therefore, there are no doubts about the functionality of these two software development environments. Thus, the focus switches to usability.

It was noticed that graphical interfaces could lead designers to better understandings, and better usage than text-based interfaces [92]. This feature can also be observed from both software tools. In DAvE, the interface guides designers to complete most
procedures of a microcontroller software development from selecting the target controller to detailed function setting. In different stages of the software development, different interfaces are used. The switching of interfaces can lead designers to focus on current implementation issues (Figure 4-13). In addition, designers can make progress of the whole design process due to the change of interfaces.

Figure 4-13: The switching of interfaces in DAvE

Graphical interfaces can also be found in Lego RIS. In DAvE, different interfaces were designed for different peripheral functions. Once a function is selected, the corresponding interface appears. Designers can start to configure the function. A different approach is applied by Lego RIS. No peripheral functions are listed. Designers start software development from selecting the intended actions listed aside and then carry out the settings. In RIS, all available functions of the Mindstorms have been converted and hidden behind the puzzle-like pieces. Users can simply set the performance of the selected actions. Because of the interface, the progress of software development becomes very natural and straightforward. Although two different implementation strategies are applied in these two software tools, they both apply graphical interfaces to help designers through the software development process.
Chapter 4 - Review of Previous Work

The graphical interfaces provide benefits during the development process. As the interfaces change, users can focus on the current development issues of software implementation and work out the critical issues at the right time rather than just dealing with unorganised information and coding work (Figure 4-14).

Figure 4-14: The condition and action setting in RIS

4.4.3. LIMITED INFORMATION AND EXPERIENCE INQUIRY

As mentioned before, software development is a very knowledge intense activity. Knowledge and information are the fundamental equipments for designers to bridge the gap between the intended designs and the proposed software programs. However, the experience of obtaining information and applying the information effectively are both achieved by the learning process. Accordingly, a quality microcontroller software program is usually developed by experienced designers. The reason requiring software designers to be experienced is due to the limited supporting available from existing software development tool. Designers usually have to go through a time-consuming learning process in order to be experienced. The situation has been slightly changed in both DAvE and RJS.
These two tools applied different methods to reduce the effort such as data collection and the application of the related knowledge into software development. In DAvE, related information is collected from the inside (Figure 4-15). Designers can then easily discover the related information, when they read it. Later on, the issues related to function setting are sorted into groups, and designers can complete the configuration of function by setting those issues. Through this method, information handling work was reduced.

Figure 4-15: One of the provided information from DAvE

In RIS, the part of function configuration is hidden behind the puzzle-like pieces. It is not necessary for users to understand the actual performing function and the detailed settings for the design purposes, as the users only have to be aware of the intended performance acted. Further, the limitations and possibility of the actions are pre-defined, and users can apply the hand-on information directly onto the actions without obvious problems. Therefore, the work required to carry out proper performance is minimised by the RIS.
From the features of these two software development tools, certain improvements to the current software development process are achieved. The improvement came from features such as a user-friendly interface, quality program delivery, and shorter development time. Since the efficiency of current software development is restricted due to the usability of integrated software development environment, a new development environment is proposed to build to achieve the improvement. As most current software development environments are usually PC-based software, it is thought that applying advanced computer technology will make a difference. In order to make further improvements to current software development, the limitations of the previous mentioned software tools will be discussed later.

4.5. Limitations of these systems

It is clear that both DAvE and Mindstorms offer benefits to current microcontroller software development. However, some difficulty was still encountered during the software development such as producing the proper implementation solutions to fulfil the needs from the proposed systems. Further details related to the limitations of these two software development tools will be given in the following sections.

4.5.1. LIMITED USAGE

For DAvE, it is convenient to search suitable microcontrollers from the list of on-shelf products for the proposed systems. In addition, designers can start the development of the applications of the targeted microcontroller afterwards. The designers can obtain full control ability to configure the peripheral functions inside the microcontroller through the graphical interfaces, and each detailed control mechanisms related to the function was shown to the designers. However, there was no guideline for designers to organise the implementation solutions for the proposed system. Also, the captured information does not fulfil the users' needs while configuring the proposed functions for specific performance through the corresponding control mechanisms displayed. It meant that the gap between function performance and control mechanisms setting still exists. Supporting information was still required for the implementation of the proposed peripheral functions.
Even though the peripherals are emphasised on microcontroller-based software development, the configuration of the peripheral functions does not make up the entire microcontroller software. For the part of system control, designers still need to implement it by programming instructions. The process is achieved under the same way as the programming under traditional development environments. Therefore, it can be concluded that the major beneficial improvement of DAvE is fast peripheral function configuration. In other words, the development of microcontroller software was only partially improved over the traditional software development process.

In RIS, the limitations appear in different aspects. Although RIS is a relatively simple software development environment, it only allows designers to compose software design by the puzzle-like pieces and then creates ready-to-downloaded program. The goal, however, was achieved by reducing the capability of the Mindstorms. Furthermore, some peripheral functions of the microcontroller embodied inside the Mindstorms have been disabled or simplified in order to match the usage of RIS. Hence, the actual microcontroller inside the Mindstorms cannot perform its full potential under the RIS.

4.5.2. PERFORMANCE JUSTIFICATION

Once a software program has been developed, the following tasks are testing and evaluation. If the programs did not perform as the systems require, designers need to correct and modify the developed programs or even re-develop it.

For RIS, the developed program can be loaded into Mindstorms immediately through infrared data communication, and no source code will be shown. Therefore the justification of Mindstorms’ performance was achieved by either refining the performance of puzzle pieces of the original design or creating a new program. As no simulator or emulator is provided with the RIS, designers have to identify where the problems are inside the developed programs. Thus, the program justification might not be as easy as the development of the intended program.
While using DAvE to implement the proposed functions, a user-friendly interface was offered. Inside the interface, all the related factors were presented, and designers can manipulate these factors to produce the intended functions with their corresponding performance. The corresponding software program with comments will be delivered after the implementation process. If the produced performance does not meet the system requirements, further modification was needed for the delivered program. Although some comments were offered, only a limited amount of help could be offered for program modification. Therefore, designers still need to refine the source code themselves by experience or information.

Designers are offered with different benefits while composing software program with these two software tools. Although these benefits were delivered due to different features, the software development process was actually improved. It can be observed that the software design process was improved by handy software development tools. It is believed that the software design process can be improved further by adopting modern technology into the software development tool.

An advanced microcontroller software development system was thus intended to be developed, as it was proposed as the solution to improve the current microcontroller software development. The proposed software development system should perform similar features as DAvE and Mindstorms and also improve the drawbacks caused by these two software tools. Since the proposed aim of this research is to develop an advanced system to improve current microcontroller software development process, it is proposed to find a technology which is capable of performing the intended features to enhance the functionality of software development environment. The evaluation of the available technologies will be shown in the following section.

### 4.6. Possible methods to improve the current software design process

Microcontroller software development is suffering from a variety of problems related to quality and quantity. The current trend is to reduce the possible problems through well-defined systematic methodologies or advanced development systems. These solutions were described in detail in sections 4.1 to 4.3. However, the improvement
Chapter 4 - Review of Previous Work

found was very little, as the required effort from designers during the software development was still huge.

Due to recent advances in technology, mathematical methods could be applied to specify the question domain and infer the proper solutions due to the requirements [93], which aim at using mathematical methods to describe the process of software development. Indeed, this method offers a new solution to software engineering, but the effort required to carry out software development by using this method was similar to the amount needed while using the traditional methods. Hence, this method was not considered as the proposed solution for developing the intended software development environment.

The costs of developing software are growing. In order to improve the situation, alternative solutions are desired for the improvement of software development. During the software development process, knowledge and experience are considered important factors for quality improvement [27]. Hence, the main considered issue of selecting the proper technology for the implementation of the advanced software development system was the capability of offering knowledge and experience. The complexity of microcontroller software development is aimed to be reduced and quality software development achieved. Two technologies are currently widely used for this purpose as they are capable of distributing former experience. One method is software reuse, which is the reuse of previous software design components and adapt them to solve new problems; the other method is knowledge-based system, which allows designers to explore the possible solutions of the desired design.

4.6.1. SOFTWARE REUSE

There is a need for technologies which can significantly decrease the required work in developing software products, with the increasing quality of software products and reduced time to market [94]. Recently software engineering has been attracted by software reuse, as this method provides benefit to software development. Retrieving applicable past software design experiences to current circumstances is the general aim of software reuse [95]. Software reuse not only alleviates the difficulties of coding during the software development but also raises the quality of the developed
programs [96], since each reused component has been tested and validated. Thus, the often-encountered problems during the software development can be eliminated, and the aim of fast development and quality program delivery can also be achieved.

Composing a software program through Reuse Technology requires reuse architecture and components. By focusing on a particular problem, the desired architecture for implementation can be found [97-100]. The architecture here was treated as the implementation target. After the implementation architecture has been decided, the use of repository-based components is another important issue [101]. In order to effectively achieve the purpose of the reuse of existing knowledge, the components for reuse are treated as design patterns for easier composition of intended software [102-104].

With the aim of reducing the complexity of the software development process, reusing approved design components to solve new problems seems to be a good solution [105]. Often, a high performance reusable library containing sets of reusable artefacts is required in software reuse. For a reuse software library, the theory of retrieval facets focuses upon remembering past experience that is applicable to the current design. The designers can then select the suitable artefacts from the library to compose the intended design.

The method of software reuse started from the reuse of code-level entities such as subroutines and data structures. Object-oriented technology offers another solution of reuse, as the technology can improve the modularity and readability of software design [106-108]. Like a module, an object is a single instance representing an executable software artefact. Therefore, the structure of the software is easy to understand and maintain, modify and extend. By using software reuse, software development can be improved by short design life cycle and quality improvement.

4.6.2. KNOWLEDGE BASED SYSTEM (EXPERT SYSTEM)

A knowledge-based system (KBS) which comprises knowledge, information and expertise is capable of assisting its users in an interactive way to solve various problems or queries. KBS is a computer-based system, which attempts to represent
human knowledge or expertise in order to provide quick and easily accessible information in a practical and useful way. KBS has the ability to accomplish cognitive tasks, which currently require a human expert to complete. Therefore, KBS is applicable to a variety of areas where expertise or knowledge is needed.

The applications of KBS are software programs designed to capture and apply specific knowledge and expertise in order to facilitate problem solving. Designers can query different data or knowledge from KBS to cover their insufficiency in experience or knowledge. Examples of applications of KBS can be found in [109-116]. In these KBS applications, the captured expertise helps users to effectively solve encountered problems with improved results.

It is believed that applications of KBS can also shorten the large conceptual distance between the problem domain and design artefacts, since the recognised knowledge can improve subsequent problem-solving by avoiding previous failures in the future [117]. The applications of KBS used in the design field is not to replace designers but to assist by providing past experiences kept inside systems. Thus, designers can produce their intended results within a short design cycle and satisfactory results delivered [118-120].

The main merit of knowledge-based system is that it provides the required information that users want at the right time to facilitate the process. Hence, KBS is also applied to the field of system design [45, 121]. During the development process, designers need different kinds of information and knowledge to process the sequential development activities, and thus the organisation of the collected knowledge and information for KBS are as important as the infer logic.

Software development can be seen as a knowledge intensive activity in which such essence is initially missing or incomplete. Designers can use a variety of methods supported by knowledge and information to systematically fulfil these missing parts. Both quality and productivity may be significantly improved by the effectiveness and safety of the development software system, and effective decision support for software designers based on experts' experience may facilitate such improvement.
The organisation of expertise and knowledge is achieved by knowledge-based system (KBS). Both software reuse and KBS can provide benefits to software designers, and it is important to find the one which can benefit more to the microcontroller software design. In order to find the proper method for building this software design environment, an evaluation was carried out for finding the result.

4.7. Principles for selecting the proper method

Although both software reuse and KBS seem to be capable of solving the current problems in microcontroller software development, few criteria were used to evaluate these two technologies. The purpose of these criteria was to determine the appropriate solution which can then be applied to improve the process of current microcontroller software development. In this section, the principles used will be revealed and the final result of the decision will also be displayed.

4.7.1. CRITERIA

Experience and information play important roles in software development, since they are used to cross the gap caused by the missing part of the development tasks, such as defining the whole system needs, finding possible solutions, and preventing previous errors [122]. Therefore, the quality of software development could be improved by providing designers the expertise and information. Currently, both software reuse and knowledge based system can meet this need as they are both capable of applying former experience while carrying out new software design.

The main purpose of this research was to develop a system which is capable of simplifying the software development process and to help novice engineers to complete the corresponding software development of the target controller when they are developing microcontroller-based systems. For this purpose, few prerequisite conditions were set. These conditions were aimed at discovering the appropriate technology which can meet the needs of microcontroller software development, and then make improvement to the process.
4.7.1.1. Fulfilling the development needs of microcontroller applications

As mentioned in Chapter 2, microcontrollers can be implemented inside various products for different purposes. Therefore, the implementation of a target controller can be seen as application-oriented, same as the corresponding software. Accordingly, the functions performed by the microcontrollers are to meet the needs of the proposed systems. In order to fully support the implementation of software designs, the applied technology should be able to indicate and support the discovery of the possible solutions to fulfil the needs from the proposed systems. Due to the very nature of knowledge-based system, useful tips and hints can be offered to solve the faced problems. This characteristic fulfils the needs of microcontroller system development as the variation of applications is wide and the whole process is composed by multiple development activities. For software reuse, some of the system development issues might not be able to be supported by the available reuse components.

4.7.1.2. Releasing the potential of the target controller

Although the implementation of embedded peripheral functions is the key part of microcontroller software development, the applications of microcontroller are not merely that. In order to augment the performance of the target controller, integrated function were made available by the combination of software design and peripheral functions. These integrated functions are capable of performing functions which was performed by external hardware components. As compact systems can thus be produced and the cost reduced, the will to implement these integration functions is ever increasing. To design and implement these integrated functions, experience or expertise was required. It is easy for KBS to offer hints or guidelines to designers while they are carrying out the development. However, the function of indication can be a problem in reuse technology.

4.7.1.3. Memory space

In order to fit in the reuse criterion, the created programs composed by reused components will require more memory space as well as slow execution speed [33], since the reuse technology are only applicable for high level programming languages currently. This issue might not be a problem to other types of software designs, but it
can be a big problem to microcontroller software development. For microcontrollers, essential memory including ROM and RAM has been included. However, the size of the memory space might not be sufficiently high. Accordingly, the created programs from reuse technology for microcontrollers might not be always suitable, since the memory size is usually a major issue while selecting a proper microcontroller for the target controller.

4.7.2. SELECTED SOLUTION

It can be seen that both software reuse and KBS can benefit the software development process, although they provide different merits. Software reuse can help designers to deliver quality software program by using proven codes from the reuse library. The applications of KBS are capable of providing indication or the required information or knowledge to designers, and the information can guide the designers to the intended solutions. Due to current circumstance of microcontroller software development, an application of KBS can be more appropriate. This was due to the characteristics of microcontroller software development, as the development of microcontroller software was integrated using multiple activities and each activity focused on specific purposes. Helping designers to produce the required parts which were missing to complete the intended software development is the fundamental function of knowledge-based system. The use of the advanced computing technology was aimed at modifying the development process of microcontroller software to achieve the proposed improvement by the understanding of proposed applications.

4.8. Proposed technology for the development of this knowledge-based system

It is decided that the improvement of microcontroller software development through an application of knowledge-based system will be achieved. In order to give this knowledge-based software development system more capabilities, a visual-based programming language is proposed to build this system. This is because visual objects are easier for designers to accept than plain text.
Currently, there are several available visual based programming languages, such as Visual Prolog, Delphi, C and Visual Basic. It is common to use Prolog to build a Knowledge based system, since the language is famous for its logic ability. This is because Prolog can offer powerful logic operation, which can be a strong advantage while composing knowledge-based systems. Therefore, the first idea was to choose Visual Prolog to build the system. However, the final decision of the project was to use Visual basic to replace Visual Prolog. The reasons for this are given below.

4.8.1. VISUAL PERFORMANCE

For the desired software development environment, the software should be very user-friendly, as its true aim is for users to create their software programs. For these two languages (Visual Basic and Visual Prolog), they both offer similar visual components for users to create their programs. However, the actual problem comes later. As mentioned before, the purpose of this system is used to help designers to create their software programs for microcontroller. Therefore supporting information might be offered in different formats, either visual displays or text-based messages. For Visual Prolog, it is not a problem to create text format output, but the quality of the visual output can be less than when it was created in Visual Basic. This is because the capability of Prolog to create visual images is limited due to the available syntax. In Visual Basic, the chances of creating friendly image for designers to catch the idea is high, since the possible shapes that can be created are much more than the number in Visual Prolog.

4.8.2. RULE-BASED PROCESS

The creation of this knowledge-based system is to support designers to create software for microcontrollers. Generally, the usage of microcontrollers is the accordance result of the peripherals inferred from the used external devices. Hence, the microcontroller software development is highly related to data management, as the hardware function configuration is achieved by the manipulation of control mechanisms in terms of data setting in software, and different data setting leads to different performance. The part of peripheral function configuration in microcontroller software development can therefore be said as a rule-based process.
For a rule-based system, the consideration focuses more on producing the desired results from managing the available resources without interference of rules. Thus, the powerful logic operation ability provided by Visual Prolog did not take advantage over Visual Basic.

Due to the two main reasons mentioned above, it was then decided to adopt Visual Basic to implement the intended knowledge-based microcontroller software development system.

4.8. Summary

The improvement made to microcontroller-system development could be found in various types of development tools, as they attempt to reduce the complexity of system development by offering easy-to-use interface and achieve the aim of fast system prototyping. However, the software development for the microcontroller applied is not considered, as the controller is treated as a single components inside the system.

As the size and complexity of microcontroller software is currently still increasing, alternative development methods are required by the designers to improve the quality of the developed results. Recently, two new software development tools have been introduced, and they provide benefit to software designers. The benefit includes fast and quality software delivery, and user-friendly interface. Most of all, these two software tools do not require the designers to be very experienced.

Although software development can be improved with the using of these two development systems, the improvement is found limited. These limitations come from the commercial targets, as these systems cover only partial of the software development process or reduce the fully capability of the target controller. In order to improve microcontroller software development further, an advanced software development system was proposed. The implementation of this advanced software development system is designed to be realised by an application of knowledge-based system. This is because KBS can effectively reduce the complexity of microcontroller
software development process and also increase the designer's understandings of software development.
Chapter 5

Design of the advanced microcontroller-based development system

Improvements made to the current microcontroller software development have been found by applying graphical user interface, with simplified implementation procedures to the current development system. The software development is described as a very knowledge intense activity, and therefore expertise and information are usually very useful to such activities. Hence, it is believed that the microcontroller software development can also be benefited by former experience and supporting knowledge. The selected technology to implement former experience and knowledge to augment and improve the capability of microcontroller software development system was knowledge-based system (KBS).

During the software development process, designers usually need to carry out different development activities in different stages in order to produce the final results—software programs. These development activities require lots of work from designers to accomplish. With the support of activated knowledge, much of the required effort could be reduced. The knowledge offered was not only presented as text-based messages, as features capable of simplifying the development process can also be composed by knowledge. Therefore, the features of the intended system were proposed before carrying out the development work. Later on, the proposed features will be described in the following sections in this chapter.
Chapter 5 - Design of the advanced microcontroller-based development system

5.1. Considered issues while proposing a knowledge-based software development system

Software products are always considered with two issues, usability and functionality [57], and these two topics are also the major consideration while composing this proposed knowledge-based software development system. The usability for this software development system refers to its ease-of-use, and the functionality is the ability to produce the intended software program after the implementation process. To meet these two targets, corresponding features are also proposed while designing this knowledge-based software development system.

5.1.1. Usability

Existing microcontroller software programs are mostly built in programming instructions through coding. Accordingly, designers have to be able to use programming language and the development system to implement their intended designs. This method is currently popular, but inefficient as the implementation process, which is known as coding, is considered as time-consuming and requires lots of work to accomplish. A new implementation solution was thus applied to improve the software development process. This solution will require graphical user interface to support, as visual display can increase the accuracy of decision and lead to quality results [123].

5.1.2. Functionality

The most important function of a software development system is being able to produce the intended programs after the implementation process. Due to the proposed performance of the intended system, several different development tasks such as peripheral function setting and control strategy implementation could be encompassed within one program. In order to build a system capable of supporting designers to achieve those related tasks during microcontroller software development, corresponding features which support designers to accelerate the development process.
were also proposed. As the whole process of microcontroller software development was composed of several activities, different features will be proposed during different stages [124]. These features include intended system composition, target controller profiling, peripheral function configuration and control strategy implementation. All these development issues will be supported by captured knowledge. In addition, accurate programs should be delivered after the implementation process has been completed.

The intention of this knowledge-based development system is presented in Figure 5-1. All the features mentioned previously in both sections, 5.1.1 and 5.1.2, will be described in detail in the following sections.

Figure 5-1: The structure of intended knowledge-based software development system

5.2. Core knowledge bases

Knowledge is the bridge to cross the gap between concept and design. In order to improve the current software development process, application of former experience and existing knowledge is the proposed solution, and knowledge-based system is the intended technology. Therefore, the possible problems faced during software development need to be considered in order to organise the captured knowledge and collected information for problem-solving [125]. The collected knowledge and information are the essence of this proposed software development system [126].

As mentioned in chapter 2, the process of microcontroller software development is composed of several activities and each activity focuses on specific issues, which also
requirements corresponding knowledge and information to accomplish [127]. Accordingly, collected knowledge and information need to be categorised in proper knowledge bases in order to meet designers’ needs. The reason to categorise knowledge is to fully support different implementation purposes.

- Solving problems occurring in different development stages
- Supporting the development activities among the process
- Advancing the development process for the intended results

5.3. System-level prototyping

A microcontroller-based product is usually an integration system composed of external components and a target microcontroller, which was used for the intended peripheral functions and the implementation of control strategy. After system requirements were confirmed, designers need to decide the solutions which were used to fulfil these corresponding requirements, either by adding external devices or functions performed by the target microcontroller [128]. The applications of the target controllers will be decided until the external devices with interfacing methods are confirmed. Therefore, the development of software design for the target controller cannot start after system composition is completed.

In order to perform the full potential of the target controllers, the use of the target controllers needs to be considered from the beginning of system development rather than after [3]. This is because the applications of the target controllers will be affected by the methods used to compose the intended systems during the beginning of the system development. Accordingly, the implementation scheme of the target controller can be decided mostly after the system composition, excluding the control strategy. As the system composition is the key leading to software development for target controller, this procedure was aimed to be included and supported by this system.

Interfacing method was a frequently faced question during system composition. Although most devices can now connect to the controller directly, interfacing components are still needed [33]. Both the external components and interfacing
methods affect the application of the target microcontroller. In order to continue the following software development for the target controller, the solution for interfacing with external components needs to be decided during system composition. For developing successful software programs for the proposed systems, it is essential to implement proper solutions to meet the system needs. The knowledge captured aims at achieving the following purposes:

- Explanation of proposed functions
- Validation of implemented solution
- Comprehensive interfacing solution
- Displaying the required components of the proposed system

5.4. Profiling of the target microcontroller

The focus of microcontroller-based systems is on applying the available peripheral functions embodied rather than adding external components into the system, because it leads to great benefits such as low costs and high reliability of the systems [2]. The software for microcontroller can be described as "hardware-driven-software", since the performance of the peripheral hardware functions was directly controlled by the software. It is important for designers to have an overall concept of the peripheral functions inside the target microcontroller while developing applications of microcontrollers, as the total performance of the target microcontroller is the aggregation of the configured peripheral functions and control strategy.

Peripheral functions embodied in a microcontroller can be treated as software resources [129], as these peripherals are all directly controlled by software. In order to implement the proposed peripheral functions for the intended microcontroller-based system, handling these available resources during the software development process is an important task to designers [130]. According to the architecture design of microcontroller, the performance of some embodied peripheral functions can be based on other peripheral functions. This causes problems while organising the available resources of the target microcontroller to perform the intended functions. Furthermore, there are usually several options of a peripheral function available. Hence, the
decision making of a proper solution to perform the intended function is also considered. It can be concluded that the management of the available resources is an important task during software development.

From the exhibition of DAvE in section 4.3.1, available peripheral functions with possible options are presented to designers. In addition, designers are allowed to configure the proposed peripherals for intended functions. The performance of the target controller is the result of function accumulation. In order to perform the full potential of microcontrollers, good management of the available resources is proposed by designers during software development. Thus, the function of resource management for the target controller is targeted by this system. With the support of this function, designers should be able to choose a proper solution to implement the design according to the system configuration. The function of resource management is proposed as the microcontroller profiling inside this software development environment.

- Discovering the proposed functions performed by the target controller
- Offering guidelines to implementation solutions
- Resource management
- Capable of performing the potential of the target controller

5.5. Peripheral function configuration

The performance configuration of an intended peripheral function is associated with several factors, and these factors need to be configured in software. Due to the implementation method, these related factors need to be decided and then configured by loading proper data contents into corresponding special function registers during software development [34]. The data contents to special function registers here are to set the control mechanisms of the intended peripherals for the proposed performance. Therefore, function configuration can be described as a two-stage activity. Firstly, understanding the intended performance is required. Secondly, achieving the implementation purpose by setting the control mechanisms is also needed. In this
situation, the process of function configuration is a complicated task, since designers have to be aware of the related issues and then set the control mechanisms properly.

It can be a big problem for designers to make proper function settings in software without enough information such as vendor offer information, although it can still be a problem even with the information. This problem is caused by information editing and presentation. The data contained inside the vendor offered information is usually very technical and detailed, and this type of information might not be able to give designers direct help, as no indication is provided in the vendor offered information related to performance of the intended function. Thus, the problem of function setting is a mixture of several sequential issues.

In order to reduce the problems mentioned above, a different strategy was applied in DAvE. It can be seen from the exhibition of DAvE, related factors of a peripheral function are gathered inside on configuration panels. By this method, designers can produce the intended peripherals by manipulating the listed factors. In order to further simplify this stage of software development, it was proposed by this system to allow designers to set up the intended function in a straightforward manner, by offering designers the options of possible performance while configuring peripheral functions. This is because these functions are more well-known for their performance rather than the configuration process. Hence, common usage of the peripheral functions was investigated. The common usage will be presented as available options for designers to set up the intended functions. Accordingly, the development process can be directly benefited from this feature.

As the development of microcontroller software is to perform the proposed functions, a visual peripheral function simulation is also included. It allows designers to view the virtual created performance of the proposed functions. By offering straightforward implementation of the intended peripherals and virtual simulation of the performance, it can also increase users' understanding of possible usage of the intended functions.

The direct implementation process is aimed at shortening the distance between the function proposed and the intended performance. It is believed that a software
implementation tool which allows designers to compose their intended peripheral functions can actually benefit the microcontroller software development by:

- Straightforward procedure
- Performance oriented options
- Virtual performance simulation

5.6. *System control implementation*

Embodied peripheral function configuration is a very important issue in a microcontroller-based system development, so is the implementation of system control strategy. Usually the control strategy of a microcontroller-based system is different from one another. Logic or arithmetic operations are usually applied in order to implement such a software design. These sorts of operations are used to build the conditions required by the intended systems. Accordingly, the conditions must be built with available data which can be fetched by the microcontroller.

A control strategy of a microcontroller-based system usually refers to the behaviour that the system response to, such as external or internal event. A system control is often composed by two components, conditions and actions. The condition is related to the corresponding issue of the system. Once the condition is set, the following focus will be on the action. Of course, the actions of these systems must be functions performed by the microcontroller. There might be several conditions and actions inside a control strategy, and then designers need to organise the conditions and reactions in order, due to its sequence or priority inside the control strategy.

The problem frequently encountered by designers while implementing system control strategy is to link the system behaviour domain and the microcontroller function domain. Therefore, RIS is considered as a relatively easy implementation software tool, as there is no need to convert the intended system behaviour into functions performed by the target controller presented in 4.3.2. Furthermore, the control strategy has to be implemented by assembly language, which has only a few instructions for logic operation. To compose the intended system behaviour by
sequential conditions and actions, experience is required [131]. In order to reduce the difficulty of software development, a system-performance implementation tool is also proposed. The system performance implementation here aims at producing the intended system control by describing the desired performance acting by the system.

- System issue drive solution
- No coding implementation

5.7. Easy accessing approach

Functionality and usability are often considered while developing a software system. The features introduced in the previous six sections were more related to functionality. The feature that will be introduced in this section is usability. One obvious drawback of the current software development environment is due to the difficulty of implementing software design, as the only available method for implementation the design is using proper programming language to code. However, it is very time consuming, and requires lots of effort to complete.

The interface of a software system which allows interaction between end-users and the computer plays a vital role in the effectiveness of system functions [132, 133]. User interface generally consists of information displayed to the users and facilities which allow the user to enter information into the computer, in order to manipulate information displayed and to take control actions. It enables the end-user to access and make use of the facilities and functions which the system provides, and carry out the tasks for which it has been designed.

It can be concluded from section 4.4.2 that one of the major advantages of DAvE and Mindstorms is that no coding work is required while implemented the intended software, and this is the result of using graphical interface, since the required issues for software are presented as visual components for designers to decide. Therefore, the complexity of software development is reduced.
The usability often refers to the ease of use. It is thought important for designer to gain full control of the software tool. In order to make the system easy to use, a user-friendly interface was adopted in this system. The interface will be directly mounted with other features mentioned before. By integrating this feature, the current microcontroller software development process is proposed to be improved by reducing the input effort required currently.

5.8. Summary

In order to improve current microcontroller software development, the focus is on the design and implementation process. It was proposed to apply artificial intelligence (AI) into this area to improve current situations through a knowledge-based software development system. As the ease of use is often a consideration of software system design, a user-friendly interface was then proposed. It is aimed at augmenting the usability of the software development tool.

Although this desired system targeted the development of microcontroller software, the activity of system composition was also included, as it is the key leading to the discovery of the application of the target controller. Afterwards, guidelines to find the solutions for the composition of application are given. Furthermore, a simplified process for peripheral function configuration and control strategy implementation is also proposed. The major purpose of this software development system is to achieve the improvement of software development by processing the development activities in a relatively easy method than coding is.

The described features mentioned previously were aimed at improving the performance of the knowledge-based software development system. To achieve these features, some design strategies need to be applied while building the software development system. These issues of implementation will be introduced in the next chapter.
Implementation of the knowledge-based microcontroller software development system

The purpose of this research is to improve current microcontroller software development effectively through the application of knowledge-based system, which is capable of helping designers to accomplish the development process. The intended features of this proposed software development system were introduced in chapter 5 and the design and implementation of this system will be presented in this chapter. As a proposed feature might be the integration of several subfunctions, the main framework of the system, the knowledge base, will be introduced first, followed by all the subfunctions. The subfunctions of the knowledge-based software development system include system configuration, microcontroller setting, control strategy implementation and code generation. As these subfunctions target different activities of microcontroller-based system development, a guiding strategy was implemented to link these subfunctions and maintain the consistency of software development.

In order to implement the proposal for this desired knowledge-based software development system, an off-the-shelf microcontroller was selected as the core controller inside this system. The chosen microcontroller was Microchip Technology, PIC17C756. The reason that this microcontroller was chosen was due to its availability and functionality. In the following parts of this chapter, more details related to the implementation of the knowledge-based software development system will be presented.

---

4 PIC17C756 – PIC is registered trademarks of Microchip Technology Incorporated in the U.S.A and other countries.
6.1. System overview

The implementation of a microcontroller inside a well-integrated system needs to be considered from the beginning of the system development, since the application of the target microcontroller is not a simple after-design add-in [50]. The target microcontroller is aimed at performing the required functions according to system requirements. As the system implementation affects the software development of the target controller, the decisions related to system implementation need to be solved first. Afterwards, the development of software can then be started.

A microcontroller-based DC-motor controller is used as an example to explain the situation mentioned above. Due to different system specifications, the appropriate method to achieve the purpose of DC-motor control is not firmly fixed. The control of DC-motor can be achieved by changing either the input voltage or the percentage of power-on time period. The voltage change, which is popular in traditional analogue control systems, is achieved by outputting different parallel digital data from the target microcontroller to a Digital Analogue Converter (DAC) board to generate the intended voltages. The control of power-on percentage is achieved by the microcontroller to perform Pulse Width Modulation (PWM) function through an H-bridge. The output function of parallel data or PWM requires corresponding software to achieve, and these functions are the true development tasks.

The importance of supporting knowledge and information to microcontroller software development was described in Chapter 3. Accordingly, the lack of information causes problems while developing microcontroller-based systems. Although the existing integrated development system (IDS) allows designers to develop microcontroller software, it provides only a coding environment. The existing software development environment is not capable of offering enough supporting information that covers different aspects of the design information for designers to use during software development [57]. This is why a knowledge-based software development system was proposed.
Chapter 6 - Implementation of the knowledge-based software development system

Figure 6-1 is used to illustrate the current microcontroller software development. Currently, the related knowledge and information was obtained by designers. Due to the knowledge gaining process, the complexity of software development is therefore increasing. Furthermore, extra time and effort was required during software development [52, 134]. Therefore, the applications of knowledge-based systems have been reported in areas of mechanical engineering design, real-time software or hardware-software partitioning [121, 135, 136]. Whilst some commercial systems exist, e.g. DAvE, the underlying principles are not reported for obvious commercial reasons. Due to the competence of the knowledge-based system, an application of it can be helpful while designers are developing their software design. It was then that it was decided to create a new environment (Figure 6-2).

Figure 6-1: External effort required during traditional software development process

Figure 6-2: The intended software development supported by proposed KB development system
In order to effectively improve the development process, the application of a knowledge-based system was aimed at combining good quality knowledge and proper software development strategy to provide useful and related information for designers to process the development activities. Furthermore, the application of the microcontroller is not only to use the peripherals but also to implement the control strategies. Both peripheral function and system control of the target microcontroller will be covered inside the knowledge-based software development system. In order to give more details, the structure of this knowledge-based microcontroller software development system including supporting knowledge bases and guiding strategy will be explained later.

6.1.1. DEVELOPMENT GUIDING STRATEGY

The microcontroller software development process is made up of several sequential activities (Figure 6-3). Each activity needs support from specific knowledge and information, as each targets specific purposes. To compose an effective knowledge base containing useful knowledge which is capable of solving possible problems [137], a multiple-layer knowledge-based system was proposed.

**Figure 6-3: The sequence of development activities with required knowledge**

For these development activities, they need to be carried out in sequential order, and a guiding strategy is hence deployed in this knowledge-based system. The guiding
strategy not only links the development activities with the designated knowledge bases but also achieves data sharing between different stages for consistent development. It is aware that the application of the target controller was influenced by the used external components within the system. Therefore, finding proper components to fulfil system specifications is usually the first step of microcontroller-based system development, and the task is supported by system composition KB. Through proposed external components and interface devices, corresponding functions of the microcontroller can be inferred. After completing the composition of the intended system, the guiding strategy will advance the development process to the next stage.

The aim of microcontroller software development is to achieve the intended performance integrated by the corresponding functions, according to the intended system requirements. Since the proposed performance of a microcontroller is accumulated by functions, enabling proper functions to achieve the purposes is important. Configuring required peripherals will be carried out during the second stage of knowledge-based software development, function configuration KB. The guiding strategy will then move to the system control strategy implementation after issues of peripheral functions are set.

The final stage of the software development focuses on producing the intended system performance by those functions configured in the previous development stages. The implementation of system control strategy in microcontroller software development can also be expressed as the integration of system events, conditions and actions, and these events are covered by the third knowledge base, control strategy KB. After composing the intended systems, corresponding software programs will be delivered by this knowledge-based system. The delivered software programs are ready-to-assemble, and composed of assembly instructions. Due to the work of the guiding strategy, the essential development tasks for the intended system can be carried out in order, and these tasks can be supported by corresponding knowledge bases.
6.1.2. STRUCTURE OF KNOWLEDGE-BASED SYSTEM

Knowledge is identified as a useful element while developing software, but different domain knowledge is required during different development stages. Currently, most of the knowledge comes from designer’s experience. The purpose of this knowledge-based system is aimed at helping designers to go through these stages by offering supporting knowledge. In order to effectively increase the usage of the captured knowledge, knowledge is sorted into different categories. This is because useful knowledge needs to be provided at the right time in the right form; otherwise, inappropriate knowledge only serves to confuse the designers. This system is aimed at helping novice designers to go through the bottleneck of traditional software development. In addition, experience designers can also use this system to produce fast software prototyping.

As different knowledge is required during different activities, the organisation of the related development knowledge will therefore, be treated as the mainframe of this application of knowledge-based system [138]. In order to place the captured knowledge/information into proper categories for this knowledge-based system, suitable categorising principles inferred from the existing software development process were also needed [139]. These principles include interfacing, function configuration, performance setting, and intended system control strategy. According to these principles, captured knowledge can then be used to embody the intended knowledge-based development system.

The accomplishment of the software development is achieved after a sequence of activities, and each activity has it special aims. In order to advance to the following activity, designers have to understand the specifications of the proposed system and applied knowledge to need the needs. However, the applied knowledge is not random used, corresponding rules are required as they are like the interfaces between question domain and available solutions. In order to reduce the bottleneck of the traditional software development process, the implementation rules will be explained with Figure 6-4.
The software development for the target controller of a system actually starts from the beginning of the system composition, as the external devices affect the functions performed in accordance. This is because of the interfacing solutions used between the microcontroller and the devices. In order to discover the actual implementation solutions of the target controllers, the rules of interfacing are required (line 1 in Fig 6-4). Afterwards, the functions required to fulfil the system needs can be discovered by the rules of function usage (line 2 in Fig 6-4).

In order to well organise the resources of the controllers for producing the implementation solutions, the rules of implementation priority are involved (line 3 in Fig 6-4). These rules can prevent required functions being used for other purposes. Afterwards, the rules of function configuration (line 4 in Fig 6-4) will be applied during function implementation.

The software development of microcontroller is not merely for using the peripheral functions, as the part system control is usually involved. Often, the system behaviour as a whole is a sequence of system actions, which are performed by microcontroller functions. Therefore, there is a need of using the rules of producing system behaviour by function performance (line 5 in Fig 6-4). During the implementation process, pieces of codes are produced. However, it does not mean a program containing these pieces
codes randomly will work, and the rule of code organisation (line 6 in Fig 6-4) is also applied.

While handling all these rules, the software development process is more straightforward. There is still something missing between the changes of implementation rules - knowledge. The structure of the knowledge-base system will be explained below.

According to [126], a multiple-layer knowledge-base system was proposed to fit the characteristic of software development. This proposed knowledge base system was integrated by three knowledge bases which are named as KB1, KB2 and KB3. KB1 is the knowledge base for designers to discover the possible solutions for their intended systems, since the composition of the intended system was considered as the first activity of system development and an important clue to the following software development. The information inside the KB1 helps designers to select the necessary components to compose the intended system. KB2 is the knowledge base following KB1, and the information inside KB2 indicates the corresponding functions of the target microcontroller to fulfill the design purposes. This knowledge base also shows designers all the related issues of the selected peripheral functions as well as the configuration process. KB3 is the knowledge base supporting the implementation of the proposed control strategy of the intended system.

Under the knowledge-based system, these knowledge bases support designers to configure, enable and set the intended functions performed by the target microcontroller during different stages of the development process. The functions of these knowledge bases will be explained in the following paragraphs.

1. KB1 is the first activated knowledge base when the knowledge-based software development system is enabled. For the first stage of system development, the designers' aim is to discover the required components which can fulfill the system requirements. KB1 was designed to support this development activity. Messages and indications are the used tools to guide designers to find proper components in this knowledge base.
2. After composing the intended system, the development process will advance to the next stage: enabling the required peripheral functions for the fundamental part of the intended performance. For this purpose, it is essential to enable all the fundamental functions. In KB2, the required functions will be shown and the indications for proper implementation solutions will be offered. The whole development task might be split into several individual subtasks which can be implemented by a single function. In addition, setting the performance of these applied functions is also supported by this knowledge base. These required functions, the real development issues behind the system requirements, are covered in KB2.

3. Although the setting of corresponding functions was supported by KB2, the development process of the intended software program for the proposed microcontroller-based system has not yet been completed, because the part of system control strategy, the relationship between functions, has not been finished. The implementation of control strategy will be supported in KB3. From the selected peripherals and those proposed external components, the possible actions and conditions performed by the intended system can be pre-defined. For the purpose of implementing system control, related events, possible actions and conditions performed by peripheral functions, were the basic elements inside the KB3. These events will be left aside for designers to choose from while composing the system control logic. Therefore, designers can implement the control strategy by organising these related events in the proper sequence.

To enhance the performance of these knowledge bases and maintain the development consistence, a linkage strategy for data sharing was deployed inside the software development system. The linkage strategy was aimed at integrating these three knowledge bases.

6.2. Linking data stream among these knowledge bases

To advance microcontroller software development process from system configuration to function setting and control strategy implementation, there is usually a need for documentation. This is because these implementation issues inferred from the
requirements of the proposed systems are directly related to the software development for the target microcontrollers. In order to develop the intended system consistently, documentation usually solves the problem.

In order to reduce the work required for documentation, a data stream, which records the essential information from the system configuration, was designed to link the knowledge bases for different development procedures. The users can then directly continue their development process without the need for documentation.

### 6.2.1. SYSTEM COMPOSITION

During system composition, designers draw the blueprints of the intended systems. Generally, this development activity aims at drawing the outline of the intended system, and the software development for the implementation purposes can be proposed from this stage. With the help of data stream, the documentation work for further function setting and final control implementation can be reduced, as the details of system composition is kept within the data stream.

Figure 6-5: Producing the intended systems with the required interface in KB1

![Diagram](image-url)
The development process of a microcontroller-based DC-motor controller will be used to illustrate this (Figure 6-5). According to the application purpose, an interfacing device was needed between the target controller and the DC-motor. The options for interfacing and the feedback signal are offered. After selecting the intended solution, a simplified system diagram will be produced with the configuration conditions. Also, the intended configurations of the system will be sent to the next stage, KB2, so that the corresponding functions can be performed by the target controller.

This data captured at this stage of system composition has two main purposes: one is for designers to carry out the related function setting and the other is to infer the possible system events which can be performed by the configured functions. Since the recorded data is used for different purposes at different development stages, the data will appear in different formats when offered to designers. During the function setting stage, the data will be used to point out the needed functions. In the stage of control strategy implementation, the captured data presents the possible events which can be performed by the intended system to designers.

6.2.2. MICROCONTROLLER CONFIGURATION

With reference to the microcontroller software development process, the activity after system composition is the application of the target microcontroller- function configuration. With the data captured in the data stream, proper functions for the application purposes can be found in KB2. During the stage of microcontroller application, the data related is also recorded into the data stream for further use. This is because the application method of the target microcontroller is the key to achieving the fundamental control of the intended system, it is important to document data related to these functions for subsequent development stages.

The available peripheral functions of microcontrollers are usually embodied within multiple options. According to the decided option, the corresponding physical pin is different. In order to achieve proper applications of these intended functions, both the applied method and the corresponding pins needs to be documented. With the support of knowledge base, the coordination of the functions and pins are supported, and designers can continue software development directly. Accordingly, designers can
focus more on developing the intended performance of the system rather than on the fundamental settings of the microcontroller.

Figure 6-6: Configuring essential functions for the proposed system

In KB2, the function required due to system configuration will be used as the clues to infer the corresponding peripherals (Figure 6-6). These indications for functions will be presented aside for further function enabling and configuration. In addition, suggestions or direct solutions will be given when designers process this stage of development work. Further support was also given during the configuration of these functions. In the example here, the essential functions to fulfill the needs from system composition are offered, and the configuration of these functions are guided.

6.2.3. IMPLEMENTATION OF SYSTEM CONTROL STRATEGY

For this part of the system control strategy implementation, it is obvious that designers focus on achieving the intended system behaviour, which is performed by the enabled functions of the target microcontroller. Since the possible system events
have been inferred from the data saved in the data stream, designers are allowed to implement the system control strategy by manipulating the possible system events. The effort of KB3 was to offer the possible system events for designers to compose the intended system performance (Figure 6-7). Due to different configuration of the system and the setting of those functions, correct methods to produce this performance will be achieved by KB3 and direct options were offered to designers. Therefore, the complexity of system development can be reduced by the knowledge contained within KB3.

**Figure 6-7: Composing the intended system performance through available system action**

<table>
<thead>
<tr>
<th>Behaviour exhibited</th>
<th>Function performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor speed change through the inputted data from port C</td>
<td>Importing new data from port C</td>
</tr>
<tr>
<td></td>
<td>Loading the data to produce a new PWM wave from port C</td>
</tr>
</tbody>
</table>

When designers implement the intended control strategy, system events including actions and conditions are usually involved. With the recorded data, the possible actions and conditions can be inferred by this software development system. Designers can then implement their intended system performance by organising the
sequence of the intended system actions and conditions and no further effort is required. With the guiding strategy and linking data stream, this knowledge base can perform its full capability to help designers to implement their intended systems.

6.3. System composition

Although this knowledge-based system focused on software development for microcontrollers, system composition is also covered by this system. This is due to the composition solutions of the intended system affect the applications of the target controller, as well as the software development. In order to accelerate the following software development, a tool helping designers to generate the schemes of the intended systems is proposed. As the composition of the intended system contains two parts: applying external components and interfacing solutions, they will be discussed separately.

6.3.1. INTEGRATION WITH EXTERNAL COMPONENTS

Figure 6-8: The system composition tool inside the knowledge-based software development system

The available embodied peripheral functions of a microcontroller can be listed, after the target controller for the intended system has been decided. According to the
applied external components and the interfacing methods, the corresponding peripheral functions of the target controller can also be inferred. In order to benefit the subsequent software development and achieve a rapid system composition, common external components and available functions are presented for the designers to select (Figure 6-8). In addition, the embodied functions are shown with essential information to give designers a more detailed understanding.

In order to validate the proposed microcontroller-based system, an auxiliary mechanism to examine system composition was added into the knowledge-based software development system. This examination is achieved by managing the available resources embodied on the microcontrollers, such as I/O pins, options of PWM. The idea of this mechanism is to ensure that the target microcontroller is capable of performing all the proposed functions. The mechanism of system composition will give a clear indication to let the designers know that their intended designs are not possible for the target microcontroller to implement (Figure 6-9).

Figure 6-9: The auxiliary mechanism to prevent failure of microcontroller system design
6.3.2. INTERFACING METHODS WITH EXTERNAL COMPONENTS

While developing a microcontroller-based system, some design issues such as interfacing methods need to be considered while composing the intended systems. The connection of the target microcontroller and external components is usually achieved through suitable wiring due to the availability of the general binary input and output functions. However, the connection between the microcontroller and external components is not always this simple and some interfacing devices might be needed while connecting with some external components, e.g., DC-motor. Due to different implementation methods and constraints of the proposed systems, different interfacing devices are required even though the same external components are needed. For example, the target microcontroller can achieve the control of DC-motors by performing PWM function through an H-bridge, or by performing binary output function with the use of Digital Analogue Converter. The decision of interfacing method is usually made according to system specifications.

Figure 6-10: The options displayed for DC-motor interfacing and signal feedback

The problems of interfacing design are often faced by designers during system composition, but only specific knowledge can be used to indicate designers towards proper solutions. In order to reduce the workload of information handling, offering interfacing methods with the proposed external components to designers is proposed. Designers can decide the intended external components, and the required interfacing method will then be displayed for designers to apply (Figure 6-10). Accordingly, the
development activity of system composition can be accomplished within a short time, and designers can then start the implementation of the target controller.

6.4. Application plan of the target microcontroller

After the composition of the intended microcontroller-based systems has been decided, the required peripheral functions can then be discovered (Figure 6-11). However, the implementation solutions of these peripheral functions have not been decided. To achieve the intended implementation purposes, the management of microcontroller peripherals is emphasised, since the available number for peripheral functions is limited and the availability of these resources might be affected due to the use of other resources. In addition, the peripheral functions often interact with external components, and the corresponding pins of the peripheral functions also need to be considered while composing the intended implementation solution. To avoid the function corruption with other peripheral functions through the same pin, the implementation solutions for the proposed functions need to be considered as a whole. Due to the complexity of producing proper implementation solutions, a method of resource management was applied during the stage of peripheral function configuration.

Figure 6-11: The application purpose of the target microcontroller
One major attraction offered by microcontrollers is the availability of embodied peripheral functions, since these functions enhance the capability of microcontrollers. According to different microcontrollers, different chips may provide different peripheral functions. Therefore, producing the implementation solutions by organising available peripherals to achieve the intended performance is an important task in microcontroller software development. Also, multiple options of a peripheral function might be found available in a microcontroller, and the possible solutions for implementation are therefore increased. On the other hand, the complexity of software development grows due to the decision making of the implementation solutions. Accordingly, a resource management tool was needed to handle the situation. Hence, fulfilling system requirements with available resources is the most important task of resource management. In other words, performing the full potential of the target controller to match the implementation plan is the true purpose of resource management.

The applications of the peripheral functions are usually associated with external devices through specific pins. However, it causes further difficulty of resource management during software development, as the specific pin which can serve other purposes, may be used for other purposes due to the use of other peripherals. To achieve an effective implementation plan, a resource management scheme was applied inside the KB development system. The resource management including solution offering and corruption prevention is covered by the captured knowledge of the development system during target controller profiling. Resource management will be explained in the following section.

6.4.1. AVAILABLE RESOURCES AND CORRESPONDING PINS

A microcontroller can offer several options of a peripheral function, and these are usually associated with the same or related circuitry. Therefore, the first configured option of a function might affect the subsequent use of different options of the same function. Furthermore, the pins of a microcontroller are usually dual functional. This meant that these pins are capable of performing different functions according to the implementation purposes of the microcontroller. However, they can be used to
perform only a function at a time. Therefore, the implementation plan of microcontrollers can be affected by the usage of some specific pins. This is because of the possibility of implementation solutions might be just affected due to the use of the designated pins. Understanding the relationship between the functions, pins and resources is an important task for microcontroller system designers. In Figure 6-12, the relationship between functions and associated resources are presented below.

Figure 6-12: The available resources of microcontroller and its associated ports (featured from Microchip PIC17C756 microcontroller)

As stated previously, the pins of a microcontroller are usually multiple functional. Usually, the pins can be used as I/O pins or the extension of other peripheral functions. Designers have to avoid function corruption which was caused by a pin is applied by both the I/O function and other peripheral functions with different purposes during software development. In other words, when I/O and the other peripherals are both...
needed for the intended system with specific pins, the designers have to use other alternative I/O pins or choose another option of the peripheral. In order to prevent this situation from happening, the implementation solution for the intended system needs to be organised.

The peripheral functions are performed by the integral circuitry inside a microcontroller chip. Often, a peripheral function can be performed by several options within a microcontroller. Although it is not firmly fixed, some limitations appear when a system design requires more than one of several configuration options of a function. This is caused by the sharing the same resources between these options, and therefore the use of the second option of the intended function might have to follow the configuration of the shared resources made from the first option.

Figure 6-13: The composition of two PWM functions

The configuration of two options of the PWM function will be used to illustrate the situation mentioned above (Figure 6-13). During the configuration of the first option of PWM function, both the shape and frequency of the waveform are required. The frequency of the first PWM option is mainly decided by the application of the designated base timer. For the shape of second PWM waveform, it was controlled by independent issues from the first option. However, the frequency of the second PWM option could be influenced by the first configured option if the same base timer was adopted. Therefore, designers have to be aware of the influence that shared resources can do to the subsequent options of a function to avoid problems caused by the inconsistencies of software implementation.
In order to prevent failure from happening to a validated implementation plan, a resource management is thus needed. This resource management will save the required resources as well as the associated pins for the required functions. The implementation of a PWM function will be used to exemplify the situation here. All associated pins of available PWM functions could be used for other purposes under the implementation of software design. However, if this does happen, the intended PWM function will not be able to be produced. To avoid this, an option of PWM with its associated resources and pin will be saved. As the solutions were used to fulfil the implementation plan oriented from the system composition, the resource management is therefore considered as an important factor to microcontroller software development.

The committed resources will be served as an implementation solution offered to designers during function implementation. Applying peripherals of a microcontroller is not only to enable them, but also to configure the appropriate performance. The performance of these peripheral functions is influenced by several factors, and these factors are directly affected by these control mechanisms which need to be configured in software. Also, the ability to complete the implementation plans are offered by this knowledge based development system. In the following section, a new configuration environment for peripheral functions offered by this KB development system will be introduced.

6.5. Performance-oriented function implementation interface

Although these peripheral functions are performed by the embodied circuitry, the control of these functions is achieved through software implementation. Hence, controlling the related resources/mechanisms to perform the intended functions is a critical issue during microcontroller software development. Later on, the development process of peripheral function configuration will be introduced.

The enabling of each peripheral is related to several detailed issues, and these issues are directly controlled by mechanisms. Without proper setting of these detailed issues, the peripherals cannot perform the intended functions with the proposed performance.
Information or knowledge is usually required during the implementation of peripheral functions during software development. Therefore, it means that further work is required for producing the intended performance by mechanism setting during the software development. To effectively reduce this work, a new implementation environment was proposed.

For application purposes, the intended functions of target microcontrollers can then be decided, and a function means a development task. Designers have to consider each related issue of the development task consequentially with the proposed performance aimed at deciding the setting of those control mechanisms. Therefore, function setting is a time-consuming procedure, and often requires experience to handle due to the complexity of control mechanism manipulation. In addition, further attention is needed for composing the intended performance of the function. Offering the related knowledge and expertise to designers was proposed as the solution that could effectively reduce the workload required for function implementation during software development.

The implementation solution of an intended function was explained in section 6.4.1. Traditionally, designers have to discover the available options and decide which options to use as well as configuration. Although the control method of the decided option is similar to others, the corresponding control mechanisms are totally different. The configuration process of different options of a function needs to be considered and handled by the designers.

With the aim of improving current implementation of function setting in software development, a simplified implementation environment, a comprehensive function configuration panel, was created. It was aimed at the enabling of the available peripheral functions by these panels, where all configuration issues of the target functions are listed inside. As the key factors of different peripherals are not the same, each panel targeted one function. While activating a peripheral function, it is very difficult to separate function configuration away from the performance setting. Therefore, the issues of performance setting are also contained in these configuration panels. In order to fulfil different needs during function configuration, two kinds of
panels were designed; one is detailed configuration and the other is fast performance setting. To clearly present these designs, they will be described separately.

6.5.1. DETAILED FUNCTION CONFIGURATION

The traditional solution to achieve the implementation of peripheral functions inside the microcontroller was described in Chapter 3, and the key to peripheral configuration is the control of those corresponding mechanisms. During the process, it is important to handle the information related to these control mechanisms as designers can then be aware of the locations of these control mechanisms and their influence for the intended performance.

The intended performance of a peripheral function might be based on the support of other peripheral functions, and the based peripheral function also needs to be configured while enabling the intended peripherals. It means that the completed configuration of a peripheral function might involve more factors than those direct issues. Since the based peripheral function provides the fundamental basis for the intended function as well as the restrictions, the use of the based peripheral requires more detailed consideration during implementation process. In order to make proper aggregation of functions, the proper method used to produce function cascading is important. For the desired performance, these related issues also targeted during the peripheral function configuration. It was the purpose of creating a comprehensive peripheral function implementation tool for designers.

The work required to make the peripheral function configuration to fulfil the system needs is not only to manage the available peripherals of the target controller but also to compose the proposed performance by manipulating the control mechanisms properly. This development work relies on knowledge and experience heavily. In order to reduce the complexity of microcontroller software development, it was aimed at integrating the microcontroller handbook knowledge with comprehensive design experience into this software development environment. Although it is useful to provide messages when designers face problems, the software development system uses another method to help designers. For the part of peripheral function configuration, direct implementation environments were designed.
implementation environments allow designers to configure the intended functions and
the corresponding performance. In the implementation environment, all related factors
were listed with explanations, and designers can manipulate them to produce the
proposed performance (Figure 6-14).

Figure 6-14: The completed mode of PWM function configuration

![Completed PWM Configuration](image_url)

In order to give designers full control of these functions, the performance of the
intended functions remains undecided. The performance of the intended function will
be decided after the configuration of these related factors is set. The other type of
configuration panel is also available within this knowledge-based software
development system. This part aims at a different purpose: a fast function
configuration process. Performance setting will be mentioned in the following section.

6.5.2. FAST PERFORMANCE SETTING

Although a function is influenced by several factors, these factors are not important
concerns during the intended software development. Therefore, the focus of software
development is now on how to achieve the required performance by the function
rather than controlling the factors. Designers often focus on one or two major factors
among these factors in order to generate the proposed performance. After the major
factors have been decided, the rest of factors can be decided. A DC-motor control
system through PWM function will be used to illustrate. In order to achieve the
control aims properly, the frequency of output PWM waveform needs to be modified as the internal clock source is too fast. Frequently, an embodied timer is selected to produce the external pulses for PWM function. However, it is not actually a problem to designers if there are few options to choose from.

Since different peripherals are controlled by different factors, the corresponding simplified implementation environment of each peripheral function was proposed individually. In order to present the significant factors and hide the rest to the designers, an alternative solution of function configuration which was made available by offering options of possible performance was proposed, since the significant factors are the keys to make such performance. Designers only have to decide the major factors with the proposed performance during the function configuration process. The other factors will be completed by this knowledge-based development system.

A decisive factor could be the integration of two or more minor factors or the most significant characteristic of the intended peripheral function. Since the performance of a function is heavily influenced by these major factors, the implementation process can be simplified if designers can decide only these factors rather than consider all the related issues during software development. Although the performance of a peripheral function might be the integration of two or more functions, the integration will not affect the role of the main factor during function setting but the full performance. The rest of the factors can then be decided according to the major factor, and this part of development work can be fully supported by the captured knowledge and experience.

To achieve rapid implementation for this intended design, fast performance-oriented configuration panels were created for the knowledge-based software development system (Figure 6-15). These panels were aimed at giving designers advantages over traditional implementation methods when designers are composing peripheral functions. Design purposes including function enabling and performance setting can be done by deciding the major factors of the intended functions rather than considering all the control mechanisms and their influence.
For the intended performance of a proposed function, additional peripheral functions might be required. In order to perform the capability of the target controller and make a compact system design, the software-based solutions, which can be achieved by the target controller, were suggested. The use of these additional peripherals is also managed by this knowledge-based system. Even though the implementation plan of the target controller is expanded, it is still under its capability, as the availability of multiple options of peripherals. For the example of function aggregation, more details will be shown in Section 6.6.2.1 as the pre-designed functions inside the knowledge-based development system.

6.5.3. INFORMATION PROVIDED TO COMPLETE THE SETTING

Information is usually needed when designers develop software through different stages. For this knowledge-based system, it aims at providing information to guide designers to complete the design purposes. According to different development activities, different kinds of information are provided. In many chances of this software development system, information is offered in the format of text messages [140].

The provided messages can be used to link between different stages of the software development process. Since this knowledge-based system is an innovative design,
users might not be aware of the sequence of the procedure to complete the software development. In order to reduce this problem, pop-up messages are provided to allow the designers to continue their development work (Figure 6-16).

Figure 6-16: Guiding message between system configuration and software development

Figure 6-17: Indicating message for designers to complete their design
Chapter 6 - Implementation of the knowledge-based software development system

Messages can also show designers the options to accomplish their implementation plans. In this knowledge-based system, two available options of peripheral configuration were offered. The intended functions can be set through detailed configuration process of manipulating control mechanisms. Also, users can decide the desired performance of the proposed function by choosing the options of pre-set functions. In order to satisfy the designers' needs, two types of function configuration panels are prepared. An indication message will appear before the system provides the suitable configuration panel for designers (Figure 6-17).

It can be seen that there are two different modes of function configuration, one is the advanced mode, and the other is the completed mode. Although these two modes can both be used to achieve the function configuration, there are still some differences. For the advanced mode, performance options for the intended functions are offered. Designers can complete the function configuration by setting its intended performance. However, if designers want to configure the functions with specific performance, which might not be available from advanced mode, they might go for the completed mode. Designers gain full control of the proposed function by manipulating the control factors.

Figure 6-18: Messages to indicate the missing part of function configuration
During the configuration process of a peripheral function, designers need to consider all the related factors properly for the function and its performance. In the detailed configuration panel, designers have to decide each of them to enable the function for proper performance. While any factor is missing from the setting, the configuration of the peripheral function is not completed. Messages are provided to remind designers of the missing factors of the function configuration (Figure 6-18). This situation is similar to the performance-oriented configuration panel. To prevent designers from forgetting the setting of factors, reminding messages are also provided. For different situations, different messages with different text are used to remind the designers.

According to different situations, different messages are provided by this knowledge-based development system. The messages lead designers to complete the missing issues from function configuration, or guide designers to advance to the next development stage. Sometimes, the knowledge-based system will require the designers to input data to complete the basic requirements for the intended design.

6.5.4. RESULT OF SIMULATION

As mentioned before, the intended performance of the target controller is considered by designers [141]. Although the control mechanisms have been set in software, the final result can only be seen after performing in accordance with hardware. Therefore, it takes extra time and workload to adjust the control factors if the proposed performance is not fulfilled. It is therefore proposed to allow users to preview the produced performance during the stage of function configuration.

Due to the characteristics of each function, individual performance simulation tool was then proposed. After finishing the configuration process of a peripheral function, the simulation result in accordance with the setting function will be displayed (Figure 6-19). Using the simulation result, designers can decide whether to adjust or save the configuration. Accordingly, an acceptable function performance can be produced after the configuration process rather than modifying the control factors after running the program on the emulator.
6.6. Flowchart software control implementation

Microcontroller software development is not all about peripheral function configuration, as the implementation of system control strategies also plays an important part. The relationship between a microcontroller-based system and a control system were described in Chapter 2, and microcontroller-based systems can be said to be the implementation of control systems. In addition, the system control of the microcontroller-based system focuses on performing the interactions between conditions and actions. The conditions might be caused by internal issues or triggered by external events, and the actions are the possible functions performed by the target microcontroller. In order to implement the control strategies, it is important to organise the sequence of conditions and the following actions.

The whole development process of microcontroller software was targeted by this knowledge-based development system. This meant that the knowledge-based system was aimed at solving problems not only during the stage of peripheral function configuration, but also during the stage of control strategy implementation. As designers often face problems while turning their proposed system performance into
software, a new approach was applied by this knowledge-based system. It is aimed at producing the proposed system behaviour by organising system conditions and actions, which were inferred from the data stored during the stage of system composition and function configuration.

To describe a design of control strategy, flowchart is a very popular method for software designers to draw down their designs. This is because a flowchart displays the sequence of system events, which include conditions and following actions. After the flowchart has been built, the relationship between events are clear, and the implementation process is easier to achieve. As flowcharts can provide the actual benefits to fulfil the designer’s need, the method was adopted by the knowledge-based system to help designers while implementing control strategies.

### 6.6.1. EVENTS ORIENTED FACTORS

According to system specifications, the components required to compose a microcontroller-based system can be decided. The purpose of the target microcontroller is to produce the intended performance by enabling peripheral functions and implementing system control strategy. In spite of the setting of peripheral functions, the control strategy of the system is to perform the proposed functions/actions under specific conditions. In order to produce the intended system performance, the intended actions and conditions should be organised in correct sequence while implementing control strategy. The actions involved in system control strategy can be either unconditional actions or conditional actions. As the control strategy is encompassed as a part of the microcontroller software development, the actions need to be functions performed by the microcontroller, no matter what kind of functions they are. While implementing system control strategy, the critical issues are to produce the proposed system behaviour by microcontroller functions and the sequential order. Accordingly, the implementation of the control strategy can be expressed as the system behaviour as a whole, in terms of the sequence of system events, which may include conditional checking and function performing.

The composition of a microcontroller-based system leads to a profiling of the target microcontroller. The profiling of the target controller provides the basic clues to
identify the possible actions and conditions of the system. A microcontroller-based DC-Motor system with a counting encoder, a HCTL chip in this case, is used as the example here. The possible actions of this system are rotating forward, rotating backward, motor start, motor stop and speed change, and these actions can be defined after system composition. For this system, the possible events that can be used to form system control might be based on the encoder reading or other internal functions performed by the microcontroller itself. Therefore, the possible conditions of the system control strategy are also found (Figure 6-20).

Figure 6-20: The setting process of a conditional action

Since the system control strategy can be expressed as the organisation of system events, conditions and actions, the focus of this part of software development was to achieve the required system behaviour by using these events. However, the system behaviour needs to be realised into function-related events that can be performed by the target microcontroller, and the microcontroller software development can then be continued in the control strategy implementation. The complexity of software development was high, as designers have to gain the ability to convert the intended system behaviour into functions performed by the target microcontroller. In order to break through this bottleneck, the knowledge-based software development system was designed to allow designers to use system events to continue their implementation of control strategies. Accordingly, designers can simply organise the sequence of the intended system events to implement the design purposes. In addition, it is thought helpful if possible system events are offered to designers to manipulate when
designers are implementing their designs. Therefore, it is proposed to present the possible system conditions and actions in the knowledge-based software development system.

From the viewpoint of microcontroller software development, the consideration of system actions are different from microcontroller actions, as system actions emphasise on the results, and the microcontroller actions focus on how to achieve the results. A system action can be the result of a performing function, which can be either input or output function. The consideration of input type function is different from output function. For output functions, the key interest during control strategy implementation is to achieve the intended system performance by loading proper data contents into the control register of the function, as the intended function was enabled in the previous development stage. It means that the data contents are the main consideration while composing the intended output system performance, although the data contents might be from fixed resources or designated value. Therefore, the expected output performance was directly controlled by the loading data contents. The action of data input is similar to output function, however the contents of input data are, however, in different situations, as the contents of input data are unknown. Therefore, different considerations will apply to the essential components to compose microcontroller behaviour. Although the results of input function can be obtained, they may require operations for further applications. Hence, a function output action will be completed after loading the data contents to achieve the intended performance. However, the input function action will require further actions to complete, as the input function will be meaningless without further use of the input results.

The characteristics of the system events will be defined by the knowledge-based software system as conditions or actions. In a microcontroller-based DC-motor system with an encoder, the actions of DC-Motor are defined as action only, because the control of the DC-Motor is a result of function output. The encoder applied in the system is able to form some control conditions for the intended system, as the received data from the encoder can be used to compose conditions. Although the possible actions and conditions of proposed systems can be defined, the control strategy has not yet been implemented. Since the details of the functions and
Chapter 6 - Implementation of the knowledge-based software development system

conditions remain uncompleted, the implementation of the control strategy needs these issues to be set. The formation of actions and conditions will be presented in the following section.

6.6.1.1. Major factors

Figure 6-21: Microcontroller-based ADC System diagram and control strategy

System diagram

![System diagram](image)

Control strategy

![Control strategy](image)

While implementing a control strategy of a microcontroller software design, it is important to realise the actual behaviour that the system should exhibit. The system
behaviour includes all the actions and conditions performed by the microcontroller. Designers have to compose the intended behaviour by using these actions and conditions during software development. A microcontroller-based ADC data displayer was used to exemplify here. The intention of this system is to convert an analogue signal into binary format and then display the data on flashing LED (Figure 6-21).

According to the system specifications, two intended actions need to be performed by the microcontroller: Analogue/Digital conversion and data output, in accordance the actual tasks performed by the microcontroller are converting the analogue voltage into digital data and sending the data through an output port. If there are no further condition applied in front of the events, the control strategy should be composed by these two actions. These two actions can be treated as the major events in software design. Since the analogue voltage sampling and conversion is set to be performed routinely while configuring the ADC functions, the only action which needs to be configured is data output (Figure 6-22).

Figure 6-22: The implementation process of control strategy by major factors

For a simple system control strategy, the implementation of control strategy can be composed by these major events. However, not all the control strategies can be
expressed in such way. In order to be able to implement complex system control strategies in this knowledge-based software development system, sub-events are also added into the software implementation environment.

6.6.1.2. Sub-factors

Actions performed by the target microcontroller can be generally described as identified events, and they might be through serial, parallel binary output or input or peripheral functions. In order to complete the configuration of intended actions and conditions for the implementation of system control, further information is required to accomplish it. The information is used to fulfil the missing parts of the proposed actions or conditions. This is because the possible actions can be pre-defined, but the intended performance remains undecided. Since the performance is directly related to the output data of the functions, it is necessary for designers to load the required data to achieve the proposed aims. Again, a microcontroller-based DC-Motor is used as the example here. Indeed, the possible actions of the DC-motor can be pre-defined. For the option of speed change, although it can be achieved by the target microcontroller, the new response speed needs to be configured by designers. This part of data needs to be set by designers.

In order to respond to a specific event properly, condition checking is often applied in a control system to examine the current situation. To implement this method, condition checking needs to be able to be carried out by the microcontroller, and this means that the condition must be digitalised in software. To set a condition in software program, it is essential to understand the contents of data for comparison and reference. Usually, the comparing data is received externally, and the reference data is assigned by designers. For example, the serial input data and the capture number can both be used as the comparing data. The other essential factor for condition composition, reference data, needs to be assigned by designers while composing the condition.

While composing the intended system control strategy with events, these events can be unconditional actions performed by the peripheral functions, or conditional actions, which requires conditions to accomplish. In the case of unconditional function output,
designers need to be aware of the proper data setting required for the target microcontroller to achieve the exact performance required by the system. For conditional actions, designers need to know which conditions to fulfil system requirements as well as the essential data to compose the conditions.

For conditional actions, it is important for designers to know both the applied conditions and the following actions. To implement this part of design in software, designers need to be aware of the available resources of issues for condition setting. In a microcontroller software design, the issue is usually either a piece of input data, or the completion of peripheral functions. After the condition is designated, the intended actions follow. In order to maintain the consistency of software development, the setting of the following action is achieved within the procedure while designers are configuring the proposed conditions (Figure 6-23).

Figure 6-23: The configuration of conditional actions

In order to implement the idea of sub-event into the flowchart-like software implementation environment, each system event selected by designers was allowed to make further configurations for detailed implementation. The reason for allowing designers to make further configuration of these system events is to produce the
intended performance for the proposed systems, since system events can be inferred by the knowledge-based system, but performance cannot.

6.6.2. SUBFUNCTIONS

The range of microcontroller-based application is very wide. In order to fulfil system requirements by the performance of the target microcontroller, microcontroller development was emphasised on producing the intended functions. Although the implementation purposes of microcontrollers within different systems might be different, the functions performed by the microcontrollers could be similar. It is believed that designers will have benefited from the development process if frequently used functions have been prepared inside the microcontroller software development system.

6.6.2.1. Pulse generator

Several peripheral functions embodied inside microcontrollers are time-based functions, and the internal clock is usually used to run these base timers. As technologies advance, the executing speed of microcontroller can be really fast. Fast execution speed makes microcontroller more suitable for real-time systems than it has been. However, such a fast internal clock is not always good for application purposes. In order to slow down the pace of these timers, regular, stable and slow pulses are needed in order to achieve a large scale of time. In the past, slow frequency pulses required external components to produce, but now it can be achieved without adding any external components due to the availability of multiple options of timer functions.

For the timers embodied in the microcontroller, they can perform interrupt functions regularly after counting a designated number of instructions. If a pulse can be generated once during the interruption of a timer, the timer can be used as a pulse generator. Due to this characteristic, it allows the microcontroller to produce the effect of time cascade. As a result, it makes the embodied function more applicable to the real world without adding other external components. Producing the pulse generator requires one timer and an available output pin, and the pin is used to generate the
required pulses during interrupts. Therefore, the pulse generator can be described as an example of function aggregation.

6.6.2.2. Time delay

Another often-used sub-function is time delay. Although it looks very similar to the timer, their compositions are different. For timer, the microcontroller counts the number of internal clocks by peripheral circuitry without wasting the central processing ability. It means that the microcontroller can still carry out other tasks as the timer is counting. However, time delay is different. Time delay is a stack of codes that require microcontrollers to execute a number of instructions to achieve the aim of time delay. While executing the stack of codes for time delay, the microcontroller cannot carry out other tasks.

The target microcontroller is unable to carry out other tasks during the time delay. The characteristic of time delay can be a unique attraction to system design, as there is no need of configuring timers. In this case, time delay is very suitable for this purpose.

6.6.2.3. Encoder reading

For the design of a close loop control system, feedback signals are needed. Frequently, the encoder is used for the production of feedback signals. To use the signals from the encoders as feedback, there is a need for signal conversion, which is turning the changes of the encoders' status to meaningful data. Currently, hardware and software solutions are both available to achieve the conversion of signals into counting.

For the hardware solution, an incremental encoder chip is often used. In order to get the counting data, the encoder chip needs to be set in proper status. Both the actions of data reading and status control need to be performed by the microcontroller. As the setting of these corresponding functions and proper status control requires specific knowledge during software development, a pre-set function of encoder reading is prepared inside the knowledge-based development system. The requirements of this hardware solution such as data reading and status control need a number of I/O pins
from the microcontroller, and these needs are included inside the target controller configuration. After deciding the proper solutions to connect with the encoder chip, the status control and data reading will be completed by a pre-set function. Another software-based solution for encoder reading is also available within this KB development system. By implementing proper software detecting functions inside the target controller, the microcontroller is capable of counting the changes of encoder status through direct wiring. The status detecting function is prepared as a pre-set function inside the development system too. The enabling of this pre-set function requires the decision of corresponding pins for signal input which will also be supported.

6.7. Code generation

Regardless of what the contents of the microcontroller software are, the design content eventually needs to be converted into programs. As mentioned earlier, the contents of microcontroller software design encompass few parts, such as function composition, performance configuration and implementation of control strategy. It meant that all the corresponding codes of the parts are included inside the proposed programs, and the codes need to be organised in appropriate locations due to the use of memory space.

It is known that the configuration of peripheral functions is achieved by loading appropriate data contents into the associated special function registers. The initialisation of function performance was achieved in the process similar to function setting. For both function configuration and performance initialisation, the corresponding codes need to be firstly executed by the target controller in order to accompany with further function adjustment according to system control. The case of implementing control strategy is a little bit different from previous ones. Since the control strategy is more related to the interaction with the external components with those configured functions, this part of the code may need to be executed routinely. Also, the sequence and priorities of the occurring events need to be considered in order to implement the system control strategy. Furthermore, the execution sequence will be directed to specific vectors while interrupts appear. Therefore, how to locate
Chapter 6 - Implementation of the knowledge-based software development system

The codes appropriately with the right order is important. Figure 6-24 displays a frequently seen solution to place the code. The part of codes for peripheral function initialisation was placed in the beginning of the program, followed by the part of codes for control strategy. The solution was also adopted to place the generated code.

**Figure 6-24: The location of function configuration in software program**

The codes appropriately with the right order is important. Figure 6-24 displays a frequently seen solution to place the code. The part of codes for peripheral function initialisation was placed in the beginning of the program, followed by the part of codes for control strategy. The solution was also adopted to place the generated code.

**Figure 6-24: The location of function configuration in software program**

### 6.7.1. FUNCTION CONFIGURATION

The configuration of peripheral functions is related to the setting of the control mechanisms in terms of data contents of special function registers. In order to produce the intended system performance, required functions need to be initialised. Although the performance of the functions may need to be adjusted from time to time due to system control strategy, function initialisation needs to be completed before the applications of the function.
Generally, initialisation of these peripheral functions is only part of microcontroller software development. This part of the code is usually placed in the beginning of the program, since designers can easily find the enabled peripheral functions for further software maintenance. Also, the used peripheral functions can be involved in system control strategy, and it is better to set the required functions in advance rather than just before further applications. In addition, the result of software design can be managed in a good order. Comments are also needed for users to find the part of code for function configuration (Figure 6-25).

Figure 6-25: Program with comments to explain the instructions

6.7.2. SYSTEM CONTROL IMPLEMENTATION

The implementation of microcontroller-based system software strategy can be expressed as the sequence of system events that need to be executed by the target controller. In order to achieve this aim, each function related to the proposed event needs to be configured during function configuration. Also, the sequence of events might need to be executed routinely, and therefore the code of system control needs to be placed in a separated area different from system configuration.
In order to effectively produce the effort of code separation, labels are used inside software programs. The used labels can be used not only to create routinely executed events but also to indicate the correct events to follow conditions. For routine events configured during system control implementation, the corresponding codes will be placed into an area specially created by the knowledge-based development system. These events performed by the microcontroller can be either conditional or unconditional actions. For conditional actions, specific labels were used to indicate the correct actions to follow the conditions. The sequence of the corresponding code produced for the intended system behaviour configured during the flowchart implementation process will also follow the order implemented (Figure 6-26).

**Figure 6-26: A label is applied for a conditional action**

---

**6.8. Testing phase**

Using an emulator to run the developed software programs with the hardware prototype is a common method for evaluation. If the program can perform how it should, the program can be said to be a quality program. In order to reduce the development cost and time, the software program was often run on an emulator instead of being burnt into the microcontroller chip. However, using only the emulator
Chapter 6 - Implementation of the knowledge-based software development system

is not good enough for the aim of evaluation, since some other hardware components are needed to show the performance of the produced programs. These hardware components are aimed at working with the emulator as a testing bed for the software programs. For the evaluation purpose, an evaluator with hardware components was involved in the knowledge-based software development system. More details of the included hardware will be revealed later.

6.8.1. PIC17C756

The target microcontroller selected to be placed in the centre of the knowledge-based software development system is the Microchip PIC17C756 microcontroller. This is due to not only its availability but also its popularity. Inside this microcontroller, many sophisticated peripheral functions introduced before were embodied. According to these functions, the applications of microcontrollers were benefited.

Inside the controller, almost each peripheral function is embodied with multiple options. It increases the capability of the microcontroller to implement complicated systems. Therefore, it was believed that this microcontroller can be applied in various applications and provide a satisfactory performance. Therefore, it was decided to choose this microcontroller to be the target microcontroller, and to demonstrate that the development process of microcontroller software can be improved by this advanced software development environment.

6.8.2. MPLAB

The produced results of designers' intended designs are source-code programs. Although it is the most recognised results of software development, the performance is difficult to examine by the source code programs only. The source code needs to be assembled and then further examination can be commenced. For PIC\textsuperscript{5} series microcontroller, an existing development tool - MPLAB\textsuperscript{6} is used.

\textsuperscript{5} PIC is registered trademark of Microchip Incorporated in the U.S.A. and other countries.

\textsuperscript{6} MPLAB is trademark of Microchip in the U.S.A. and other countries.
In order to make the software development system more function comprehensive, MPLAB was also involved as a part of this system. With MPLAB, program simulation and emulation can be linked to work together. Under such circumstance, the performance of the software design can then be evaluated.

6.8.3. EMULATOR
A software development is usually finished in the form of a source code. After the assembling process, the final machine code can then be produced. The machine code needs to be burned into the microcontroller chip, and then the performance of the software program can be examined. For a fast examination of the produced program, there is an alternative solution, an emulator, as the performance of the produced program, such as accuracy and functionality could be tested. It was aimed at adding the circuit emulator (ICE) in this software development system, since the created software design can then be tested immediately without burning the program into the chip. Furthermore, it allows users to see the real performance rather than some numerical number changing among registers.

6.8.4. TEACHING KITS
In order to allow users to observe the performance of the created software programs, few circuit boards with special purposes are used. These special function circuit boards were developed for teaching purposes in the Department of Mechanical Engineering, Wolfson School, Loughborough University. Although these boards are aimed at performing simple functions, they are very suitable for result evaluation and microcontroller-based system composition (Figure 6-27).

These boards were designed with specific purposes, such as LEDs, or switches. Furthermore, these boards can be connected with the adopter of emulator with simple wiring. An example of the microcontroller-based system was composed by these teaching kits in Figure 6-28. Therefore, these teaching kits are really suitable for result examination while testing the developed program.
Figure 6-27: The emulator with teaching kits

Adapt board & Emulator

Step motor and FET drivers board

Input board

Output board
Figure 6-28: A microcontroller-based DC-motor system composed by emulator and teaching kits with system diagram.
6.8.5. PILOT STUDY

During the development stage of this system, the available peripheral functions inside the microcontroller were built as individual modules. Therefore, it was examined to see if this system could manage to produce the software which contains more than one option of a function together with the other function within a program. Therefore, a control module was proposed (Appendix A), which attempted to achieve two DC-motor control through serial communication. The operation of these two DC-motors was designed to be controlled by the received commands.

To fulfill the requirements of this system, two DC-motor control modules and one serial communication module (USART) were needed. Each module needed to be configured to fulfill the intended aims, such as the operation frequencies of the two options of PWM waveforms and the data transmitting speed (baud rate) of serial communication. After the configuration of these required peripherals was set, the intended system control also needs to be implemented from the available system actions.

The control of the two DC-motors depends on the commands received from serial communication. In order to perform the intended actions after the commands were received, it is important to identify the contents of the received data and act accordingly (Figure 6-29). As the motor action will not be changed unless new commands are received, the new motor action was taken only after identifying the new commands. Also, the commands for motor control originate from the received data. Therefore, three conditions need to be set along with its corresponding actions.
Figure 6-29: The control strategy of these two DC-motors by the received data

**Loading received data:**
If received data equals H'F0', then Motor 1 stop

**Loading received data:**
If received data equals H'0F', then Motor 2 stop

**Loading received data:**
If received data equals H'FF', then Motor 2 starts and runs forward

Actions related to received data
Chapter6-Implementation of the knowledge-based software development system

Figure 6-30: The produced program for the intended motor control system

```
MOULB D'1' ;MOVE TO BANK 1
BTFSS PIR1,1 ;WAIT FOR TX BUFFER EMPTY
GOTO WAIT_T

;LOAD THE SEND DATA INTO WREG
WAIT_T
MOULB D'1'
BTFSS PIR1,0
GOTO WAIT_R
MOULB D'0'
MOULP RCREG1,WREG

;LOAD DATA FROM WREG
MOULP WREG,RC1DATA
MOVUL D'240'
CPPSEQ RC1DATA ;IF THE DATA EQUALS 240, MOTOR 1 STOP
GOTO RC_A1
MOULB D'3'

RC_A1
MOULB D'15'
CPPSEQ RC1DATA ;IF THE DATA EQUALS 15, MOTOR 2 STOP
GOTO RC_A2
MOULB D'3'

RC_A2
MOULB D'255'
CPPSEQ RC1DATA ;IF THE DATA EQUALS 255, MOTOR 2 START
GOTO RC_A3
MOULB D'3'

RC_A3
BSF TCON2,PWM2ON

GOTO IODL
```

The implementation activities of this system were processed with graphical interfaces, and there was no need to use the assembly instructions. From the developed program, it can be seen that the motor action was taken after the received command was identified. Examining the received data was done via sequential checking with the use of instruction "goto" and labels (Figure6-30). No motor control action will be taken if no commands are identified.

6.9. Summary

It is believed that information and knowledge are needed during the process of microcontroller-based system development. However, multiple types of knowledge and information will be required, since different development stages require different information. In order to fully support the microcontroller software development, a
knowledge-based system containing three knowledge bases were built to fulfil the situation.

A microcontroller software development starts from system composition, since the devices applied will affect the application of the microcontroller. In order to improve the efficiency and reduce the difficulty of the software development process, the part of system composition is also encompassed inside the software development system. Also, the application of the target microcontroller, function setting and performance configuration, is also fully supported. Furthermore, the implementation of the control strategy is also covered by this system.

In order to produce a good quality software program after the development process, good management of the available microcontroller resources is needed. This was aimed at applying the target microcontroller to achieve the development purposes. For application purposes, straightforward configuration panels are created. These panels allow designers to set the intended functions within a short design cycle by providing designers the related information and options of available performance. Also, the implementation of the system control strategy is simplified by this system with a flowchart type software design panel. To augment the functionality of the software development system, some frequently used subfunctions are prepared for designers to choose from and use.

In order to maintain the consistency of software development, it is important for the knowledge-based development system to keep the fundamental data of the system development, as the information is critical for further application of the target microcontroller. After the development of the software design, the final source-code program will be delivered. This program can be tested directly after compiling.
Evaluation of the software development expert system with further discussion

After completing the design and implementation of the knowledge-based microcontroller software development system, the attention now aims at evaluating the performance of this system. The performance evaluation will be presented by comparing the difference between the software development process by using a traditional development system and this knowledge based development environment. For this purpose, two case studies were carried out. The first case study exhibited the development process based on a traditional software development system, which was achieved by inviting a number of subjects to develop the intended microcontroller-based systems with a traditional software development system. The second case study which focused on using the knowledge-based software development system will also be presented. While subjects were carrying out intended system development in both environments, factors related to software development such as time and external information accession were recorded. These factors are used to address the details of both software development processes. Furthermore, these factors will be used as index factors during the comparison. The case studies and comparison will be presented in this chapter.

After examining the performance of this knowledge-based software development system, a discussion section targeting the knowledge-based development system will commence. The discussion will cover the advantages and limitations of this knowledge-based software development system.
Chapter 7 - Evaluation of the software development expert system and discussion

7.1. Traditional software development process

Traditional microcontroller software development is described as a complex and time-consuming process, and the essential activities of the process were mentioned in chapter 3. In order to present detailed features and the difficulty of traditional software development, a case study was carried out to indicate the characteristics of the process. The results of this case study will be used to explain the current problems faced by designers. In addition, these results will be used to compare with those found from the development process with the knowledge-based software development system.

Traditional software development is highly dependent on experience, and some index factors were therefore aimed to be recorded to show the features of the process. The characteristics of traditional software development were proposed to be shown by presenting and analysing these index factors. For this case study, three subjects were invited to develop a microcontroller-based system by using a traditional development environment with vendor offered information. One microcontroller-based dc-motor control module was thus designed for this case study (see Appendix B). Among these three subjects, two were new to microcontroller software development, and the other had some experience, which was estimated around 20 hours in training and practicing assembly coding. The intended index factors were recorded while the subjects were developing the microcontroller system, and they will be used to address the details of current software development.

In order to show quantitative characteristics of traditional microcontroller software development, a different form of description will be presented. The software development process was decomposed into a number of activities, and characteristics of each activity were described sequentially. Related examples to explain the situation in development activities will also be presented. The presentation of traditional software development will be shown in the following sections.
7.1.1. FULFILLING SYSTEM NEEDS BY AVAILABLE FUNCTIONS

The specifications of the intended microcontroller-based systems need to be fulfilled by functions, and the solutions to these functions may be hardware or software-based. For those software-based functions, they are often performed by the target controller. Enabling these software-based functions with intended performance requires corresponding software programs and producing these functions are the implementation aims of these programs. Hence, designers need to manage available resources of the target controller for these implementation aims. To achieve this purpose, information and expertise are identified as key factors, as they have been used as supporting tools for decision making in this procedure.

Table 7-1: Indication factors of system composition during the traditional process

<table>
<thead>
<tr>
<th>During system composition</th>
<th>Always</th>
<th>Many</th>
<th>Some</th>
<th>Few</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem faced</td>
<td>33%</td>
<td></td>
<td>33%</td>
<td></td>
<td>67%</td>
</tr>
<tr>
<td>Information accessed</td>
<td></td>
<td>67%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7-2: Time required to complete the system composition

<table>
<thead>
<tr>
<th>Time required to complete</th>
<th>Exceed 5 minutes</th>
<th>Less than 5 minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>67%</td>
<td>33%</td>
</tr>
</tbody>
</table>

While subjects were considering the implementation plan by managing available resources inside target microcontrollers, mistakes were found from their decisions related to resource management (Table 7-1). Two out of three subjects made mistakes during this stage of system composition. The discovered problems could be related to the confictions of intended functions or due to the selection of an inappropriate function which does not meet system needs. As subjects did not have enough experience or knowledge to handle the development work, subjects relied on the vendor offered information to make or confirm the decisions for proposed solutions to their intended systems. Also, the subjects used the vendor offered information to correct the mistakes that they made. From Table 7-1, it can be seen that subjects heavily relied on the vendor offered information to complete this stage of development work.
Chapter 7 - Evaluation of the software development expert system and discussion

As the subjects did not have enough experience or could not handle vendor offered information to continue this development activity, system composition, extra time for reading and understanding the available information is needed to complete this stage of the development task. Another reason that extra time was required is due to the correction of mistakes. Since subjects might fail to produce a proper implementation solution for the intended system, an iterating process to produce an appropriate implementation plan was required. Accordingly, more time is thus needed for designers to accomplish this procedure. Therefore, a longer development cycle is needed to produce the implementation plan for the target microcontroller (Table 7-2).

Figure 7-1: A block system diagram of the target controller

The problem may be illustrated by considering the system diagram in Figure 7-1. The system comprises of a target microcontroller, input switches, and an H-bridge board and a DC-motor. The purpose of this system is to achieve speed control through manipulation of the input data from switches. While considering using an I/O port to connect with this 8-bit switch input board, two factors need to be taken into account. The first one is the number of available pins of the proposed port, and the second is the availability of PWM output. There are seven I/O ports in the target microcontroller, but not each port could fulfil this need, as some of them are not 8-bit ports. Further, the appropriate port should avoid confliction with the intended PWM pin. Therefore, an iterating process for correcting mistakes might be needed (Figure 7-2).
Chapter 7- Evaluation of the software development expert system and discussion

Figure 7-2: Mapping system configuration with an appropriate implementation solution

---

The solution contains PWM function

---

The selected solution does not meet system needs, as they are not 8-bit port

7.1.2. SEEKING SUPPORTING INFORMATION

After deciding the implementation plan, the intended functions performed by the target microcontroller need to be developed in the corresponding software. One major issue of microcontroller software development is to enable the required peripheral functions. The enabling of these functions is achieved by setting related control mechanisms properly, and information often plays an important role in this procedure, as the manipulation of these mechanisms also requires the understanding of the effects caused by these mechanisms. Therefore, seeking supporting information during this stage of software development is very important. Through observation, subjects were looking for information through their development process (Table 7-3).

<table>
<thead>
<tr>
<th>During software implementation</th>
<th>Always</th>
<th>Many</th>
<th>Some</th>
<th>Few</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information accessed</td>
<td>67%</td>
<td>33%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem with the information</td>
<td></td>
<td></td>
<td>67%</td>
<td>33%</td>
<td></td>
</tr>
<tr>
<td>Problem with assembly language</td>
<td>34%</td>
<td>33%</td>
<td>33%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The search for supporting information also implies that designers have to be aware of how to use the vendor offered information. The vendor offered information aims at covering most of the technical aspects. Accordingly, the information might not be
perfectly met the application purposes. Hence, extra effort is needed to filter the information to get the essential part related to the subjects' application aims. Although the subjects could find information for intended functions, they had problems with using the vendor offered information while carrying out the intended system development with traditional solutions (Table 7-3). This is because too many detailed issues were contained inside the vendor offered information. In addition, completing a function setting might need information among several sections. It further increases the difficulty of using this information.

Producing an intended PWM waveform captured from the testing module is the example. In order to drive a DC-motor properly, the frequency of created PWM cannot be too fast. Therefore, a stable resource of counting pulses for the aim of slowing down PWM frequency was needed, and a software-based hardware timer is often the solution, since no external components will be needed in this application. Figure 7-3 displays the relationships of function aggregation for the aim of producing the intended PWM waveform.

Figure 7-3: Required functions to compose the intended PWM waveform

![Diagram showing the relationships of function aggregation for producing the intended PWM waveform.](image)

However, to produce such a PWM waveform with designated frequency and ON/OFF ratio, it is critical to integrate these essential elements, e.g. internal clock, base-timer,
corresponding interrupt, and PWM function properly. Usually information is the tool for designers to achieve such a goal, since all the basic elements need to be enabled and structured in the right order.

7.1.3. IMPLEMENTING DESIGN AIMS THROUGH CODING

Information acquisition is also beneficial to the following coding process, since the implementation of software design is achieved by coding. To accomplish this task, designers need to be able to use assembly languages to achieve the setting of corresponding control mechanisms. Unlike high level programming languages, assembly instructions are keen on directing data operation. Due to the characteristic of direct data operation, assembly instructions allow designers to access and control data in different formats, either by bit or byte. However, it also causes problems when designers aim at manipulating data contents of special function registers to enable functions (Table 7-3).

Table 7-4: Special function registers required while configuring PWM function (PIC17C756)

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit7</th>
<th>Bit6</th>
<th>Bit5</th>
<th>Bit4</th>
<th>Bit3</th>
<th>Bit2</th>
<th>Bit1</th>
<th>Bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16h, Bank3</td>
<td>TCON</td>
<td>CA2ED1</td>
<td>CA2ED0</td>
<td>CA1ED1</td>
<td>CA1ED0</td>
<td>TI6</td>
<td>TMRI3CS</td>
<td>TMRI2CS</td>
<td>TMRI1CS</td>
</tr>
<tr>
<td>17h, Bank3</td>
<td>TCON2</td>
<td>CA2OVF</td>
<td>CA1OVF</td>
<td>PWM2ON</td>
<td>PWM1ON</td>
<td>CAP1</td>
<td>TMRI3ON</td>
<td>TMRI2ON</td>
<td>TMRI1ON</td>
</tr>
<tr>
<td>16h, Bank2</td>
<td>TMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16h, Bank1</td>
<td>PIE1</td>
<td>RBIE</td>
<td>TMR3IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>CA2IF</td>
<td>CALIF</td>
<td>TX1IF</td>
<td>RC1IF</td>
</tr>
<tr>
<td>17h, Bank1</td>
<td>INTSTA</td>
<td>PEIF</td>
<td>TOCKIF</td>
<td>TOIF</td>
<td>INTF</td>
<td>PEIF</td>
<td>TOCKIE</td>
<td>TOIE</td>
<td>INTE</td>
</tr>
<tr>
<td>Unbanked</td>
<td>CPCAPTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unbanked</td>
<td>PW1DCL</td>
<td>DC1</td>
<td>DC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10h, Bank3</td>
<td>PW1DCH</td>
<td>DC9</td>
<td>DC8</td>
<td>DC7</td>
<td>DC6</td>
<td>DC5</td>
<td>DC4</td>
<td>DC3</td>
<td>DC2</td>
</tr>
</tbody>
</table>

--- Means unrelated to the proposed function

Subjects were required to set the corresponding control mechanisms properly during the peripheral function configuration. The control mechanism can be just one specific bit of a special function register. Therefore, achieving the proper setting of all necessary control mechanisms through available instructions is important. In addition, these special function registers were located in different memory locations (Table 7-4). Designers then have to change to the correct directory in order to access the data contents of the special function registers. During data setting, designers also have to avoid overlapping previous configuration made by other functions. Accordingly,
designers have to consider which instruction to use for the data operation to meet the configuration aims.

7.1.4. ERRORS APPEARING AND DEBUGGING

Software programs composed of assembly instructions need to be assembled to produce the machine codes, which can be executed by the target controllers. During the assembling process, errors from the coding stage will be found and displayed. Unless the programs are error-free in coding, corresponding machine codes cannot be produced.

Table 7-5: Indication factors of debugging during traditional process

<table>
<thead>
<tr>
<th>Index factors during debugging</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error found</td>
<td>100% (more than 5)</td>
<td></td>
</tr>
<tr>
<td>Information required</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Problem found</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

The errors displayed in the assembling stage are purely coding errors. This means that either the instruction or register are not recognised. During the case study, mistakes were found in each case (Table 7-5). The problems found here can soon be solved by the clues offered by the information of compilation.

After correcting these errors, the corresponding machine codes of software development can be produced. However, this does not mean that the machine codes are capable of performing the intended functions. That is why a debugging process is often needed. An emulator is used for the purpose of debugging in the case study. Unlike program correction, there is no obvious clue or information for subjects to start debugging, and it is the reason that subjects were looking for expertise or information to achieve the debugging process. In this case, all subjects required vendor offered information to start their debugging (Figure 7-11).

Since subjects were not aware of the exact reasons causing the performance mismatch (Table 7-5), they checked through the data setting for function configuration
sequentially to identify the problems of the program and then amend them. On average, fifty-one minutes were needed for each subject to debug and correct the mistakes causing function mismatch.

The production of a corresponding interrupt function for a timer is the example here. It was mentioned that a software-based timer was often used as the solution to produce the intended external pulses for the base timer of a PWM function. By its very nature, corresponding interrupts of timer appear regularly. The appearance of the interrupt function is due to the detection of interrupt flags by the CPU, and this requires corresponding interrupt enable bits to be set. There can be a problem with the interrupt, if any of the control bit is not set properly. Due to the design of the target controller, the enabling of any peripheral function needs to be completed with the enabling of the control mechanism for peripheral interrupts. Furthermore, the control bit for global interrupt disable (GLINTD) needs to be set clear. Without setting these two mechanisms properly, the interruption function of timer cannot perform properly (Figure 7-4).

Figure 7-4: Control mechanisms required to enable the interrupt function of timer3

<table>
<thead>
<tr>
<th>TMR3IF (clear)</th>
<th>TMR3IE (enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PEIF (clear)</td>
<td>PEIE (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>GLINTD (clear)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt to CPU</td>
<td></td>
</tr>
</tbody>
</table>

After the debugging process, the developed programs can then perform the intended functions. Generally, traditional software development required a huge amount of
Chapter 7 - Evaluation of the software development expert system and discussion

manual workload from designers. The amount of manual work can be described as the results of problems faced by designers when they carry out microcontroller software development. This acts as a barrier to multidisciplinary engineers – many do not have the time to gather the experience necessary to master the traditional development process.

7.2. Knowledge-based software development process

In this section, the software development process with the researched and implemented knowledge-based software system will be shown. The knowledge-based software development process is also proposed to be addressed by the factors recorded during the evaluation process. In order to make the comparison of traditional and knowledge-based software development process, the presentation of this section will be similar to the previous one – by discussing the sequential development procedures with the factors recorded during the process. In order to collect more data for analysis, five modules were designed and twelve subjects were invited for the evaluation process.

These five testing modules which targeted five different implementation topics were designed and twelve subjects, most of them were novice microcontroller software designers, were invited for the evaluation of this knowledge-based software development system. The topics of these five modules include serial communication, analogue signal converter, system timing and signal capturing, step motor control and DC-motor driving module (Appendix C). Since most of the subjects were not familiar with the microcontroller-based system development, corresponding system diagrams and design flowcharts were also provided with the testing modules. Each of these testing modules was aimed at applying the target microcontroller into the proposed systems by performing different functions. In addition, different control strategies were also proposed in different modules. The purpose of these testing module is to evaluate the capability of the knowledge-based software development system and its actual performance. Later on, the knowledge-based software development process will start.
7.2.1. COMPOSING INTENDED SYSTEMS BY AVAILABLE FUNCTIONS

Composing a microcontroller-based system can be described as finding elements including software-based functions and external components to fulfil the intended system requirements. The software-based functions which need to be performed by the target controller are the purposes of software development. Inside this knowledge-based software development system, frequently used components and available functions of the target controller are listed inside the system composition environment (Figure 7-5). Furthermore, information is offered for designers to understand the usages of these available functions. It aims at reducing the problems faced by designers during the stage of system composition.

Figure 7-5: The first stage of the intended software development under knowledge-based system

During the evaluation process of the knowledge-based software development system, it can be said that the subjects were able to identify the required elements to compose the intended microcontroller-based systems from the system composition environment (Figure 7-6). Especially, subjects could achieve this aim within a short period of time among most cases. This leads to a fast development process.
As there is no need for designers to infer the required peripheral functions of the target controller, subjects could, accordingly, accomplish the development task of system composition right after they found the necessary elements (Figure 7-7). It can be said that this knowledge-based software development system helps designers to process the frequently-seen issues during system composition.

Although there were some problems faced by subjects during the system composition process, the problems were likely to be due to the unfamiliarity with the knowledge-based system, since these problems were found mostly while subjects were using the knowledge-based system for the first trial of the studies. Once subjects got some experience about the system composition environment, they could achieve system composition without facing any problem. It can be found that most of the system composition cases were achieved without facing problems (Figure 7-8).
Therefore, the need of supporting information is not as strong as using traditional development environment.

Figure 7- 8: Problem faced during system composition

The illustration here is a microcontroller-based dc-motor controller. Designers can directly find dc-motor control module from the system composition environment. After selecting it, further options related to the motor control will be offered. The options include interfacing and feedback methods. Designers only need to make the decisions to meet the system needs and the initialisation of the target controller will be organised by the knowledge-based system. Therefore, not many problems were reported during the system composition while using knowledge-based development system. In addition, the procedure can be finished within a short time.

7.2.2. INITIALISING THE INTENDED PERIPHERAL FUNCTIONS

This knowledge-based development system will list out the corresponding functions which need to be performed by the target controller according to the elements proposed during system composition. Designers can select a function as the design task to implement once at a time, and the knowledge-based system will help designers to enable the proposed function. Designers will be offered implementation solutions, and these solutions will then direct designers to configure the selected function. During the function configuration process, there is no need for designers to use assembly instructions to achieve the implementation purposes, although they still need to complete the function configuration by either deciding the settings of detailed control mechanisms or selecting pre-designed performance options for the intended functions.
Chapter 7 - Evaluation of the software development expert system and discussion

The implementation process, from searching the intended solutions to configuring the proposed functions, is supported by several sequential graphical interfaces within the knowledge-based system. These interfaces are aimed at helping designers to focus on current implementation issues and then complete the task gradually. In addition, the interfaces can present useful knowledge captured, which is considered to support the implementation purposes. Hence, knowledge and information can be offered in a better format and designers can accomplish function configuration and the corresponding performance easily. During the evaluation process, most cases were finished without problems reported (Figure 7-9). It can be said that the captured knowledge and information are really useful in supporting the software development process.

Figure 7-9: Problems found offered information with this KB system

<table>
<thead>
<tr>
<th>Problems with using the offered information</th>
</tr>
</thead>
<tbody>
<tr>
<td>10% None</td>
</tr>
<tr>
<td>7% Few</td>
</tr>
<tr>
<td>2% Some</td>
</tr>
<tr>
<td>81% Many</td>
</tr>
</tbody>
</table>

The validation of appropriate implementation plans for the intended systems requires information to support, as it is highly related to the architecture design of the target controller. In order to implement the intended functions through the available resources, the proposed solutions need to avoid all kinds of function overlapping. In order to clear out the relations between proposed functions, supporting information was expected to confirm the solutions. This is thus why information was needed during the development of microcontroller-based systems. However, using vendor offered information further increases the workload of software development as the editing and presentation of information. For a compact software development process, this part of workload was aimed to be reduced. It can be seen that workload required during the traditional software development was eliminated by this knowledge-based system as less problems were faced.
The implementation of a peripheral function requires not only function enabling but also performance configuration. This is also a process which requires knowledge and information to support. This is because many detailed control mechanisms are involved. From Figure 7-10, most of the cases can be completed by subjects without accessing external information. This meant that the function of solution guiding and simplified function implementation process are actually helpful during the development process.

The dc-motor controller is again used as the example. After deciding the interfacing and feedback method, H-bridge with feedback, corresponding functions performed by the target controller will be indicated – one PWM function, one binary output pin for direction control, and one 8-bit binary input port and three output pins for encoder chip status control. The knowledge-based development system will guide designers to appropriate solutions for these required functions (Figure 7-11). A configuration environment for the selected function will then appear, and designers can complete their performance setting by deciding the offered conditions. While testing, subjects could continue software development by shifting interfacing panels, although they were not familiar with microcontroller software development. Accordingly, they could also complete function configuration due to the provided options without facing problems.
7.2.3. IMPLEMENTATION OF CONTROL STRATEGY

System control strategy is an important part of microcontroller software development. Somehow, the focus of system control strategy in microcontroller-based systems is to perform the proposed functions under specific circumstances. Designers have to be aware of the control issues and then realise these issues by assembly instructions to achieve condition settings and perform the intended system actions. In this knowledge-based software development system, the available functions performed by the system will be organised and listed. Designers are allowed to implement the control strategy by organising these available actions and conditions.
Chapter 7 - Evaluation of the software development expert system and discussion

A flowchart type of implementation environment is used in this knowledge-based software development system for designers to compose their control strategies. The available system issues including possible conditions and actions are listed aside as basic elements for designers to select to compose the flowchart for proposed system control. Designers are allowed to make further detailed settings of these selected system control events. This is because not all control strategies are simply composed of pure actions, and conditions may be applied. The completion of a control strategy can be right after designers have finished the required detailed setting of these system issues. From the observation (Figure 7-12), most of the cases were finished this stage of development within a minute. The short development cycle can be said as the result of a simplified implementation process.

Figure 7-12: Time required to complete software implementation

Using the flowchart environment, subjects were capable of composing the intended system control strategy from the listed elements of system issues. Subjects might take some more time to understand the method to manipulate these elements for strategy implementation as they use it for the first time. Afterwards, they can use the implementation environment to structure their control strategy by organising these elements. No special problems were reported while they were using this flowchart implementation environment (Figure 7-13). This implies that the control strategy implementation environment is easier to use and the intended control strategy can actually be composed by those listed system elements.
Figure 7-13: Problems faced during software implementation process

In Figure 7-14, a case of system control implementation is displayed. In spite of the required functions performed by the target controller, the knowledge-based system offers system events for users to decide. Therefore, very few problems were reported by subjects while composing the intended control strategies for their proposed systems.

Figure 7-14: An example of implementing the intended control strategy
Chapter 7 - Evaluation of the software development expert system and discussion

7.2.4. PROGRAM TESTING

After completing the required development procedures, a corresponding assembly program will be delivered by this knowledge-based system to designers. The programs which were composed of assembly instructions can directly be assembled to produce the machine codes for the target controller. The performance of the developed programs were also aimed to be evaluated to ensure its functional correction.

The delivered programs by the knowledge-based software development system are composed by pieces of codes according to designers’ configuration. The delivered programs were aimed at reducing designers’ workload during the implementation process and to cut the effort required for debugging. Most of all, performing the intended functions by the target controller is the focus. From the observation results, limited errors were found in few cases during assembling process (Figure 7-15), and they were caused by mistakes during the operation of the knowledge-based system.

Figure 7-15: Errors found in the produced program by KB system

![Errors found during compilation](image)

After assembling, the performance of the delivered programs was tested by an emulator with supporting circuit boards. From the recorded results, the delivered programs can accurately perform the proposed functions, with very few mistakes found during the evaluation process (Figure 7-16). These mistakes were caused by wiring problems rather than the mistake of codes. The results show that the delivered programs are capable of performing the functions configured by designers during the development process.

151
7.3. Comparison of the two development processes

The main purpose of this knowledge-based software development system is to improve the current microcontroller software development process. In order to show the improvement properly, a comparison between the traditional and the knowledge-based development process was thus targeted. The comparison will show the differences found at the different stages of software development with these two development systems. The compared factors were the development-related issues recorded during both development processes, and these factors are used as indications of the differences.

Software development is a very designer-oriented process, since different designers have different views about software development. Therefore, it is important to find general characteristics from the development process e.g., software quality as the comparison issues. Since the quality of software development is related to factors such as, development cycle and errors found, these factors recorded during both case studies were used to compare these processes.

7.3.1. SYSTEM COMPOSITION

In order to handle the capability of the target controller and compose the implementation plan to fulfil the requirements of the proposed systems, information was needed for the decision making process. Further, the use of information implied
Chapter 7 - Evaluation of the software development expert system and discussion

that corresponding manual work was also required. Therefore, the activity of system composition might require a longer period of time while composing the intended system with the traditional development system due to the lack of appropriate supporting information. However, this situation was changed while using the knowledge-based development system. Three factors will be discusses for the development activity of system composition: development cycle, external information accessing, and problems faced.

7.3.1.1. Development cycle

From the observation results, it can be seen that the system composition, in most cases, with the knowledge-based system was completed within a minute. The other cases which took longer time can be due to the fact that subjects were not familiar with the interface used for system composition, as most of the cases which exceeded two minutes were the first module of each subject. The required time for system composition with the traditional development system varies due to a subjects' experience. Often, designers require more time in solution finding, decision making and mistake correction with supporting information while using the traditional development environment.

Figure 7-17: Required time to complete system composition

![Figure 7-17: Required time to complete system composition](image-url)
Chapter 7- Evaluation of the software development expert system and discussion

From Figure 7-17, it can be seen that the development cycle needed while using the knowledge-based system is shorter than using the traditional system. This implies that an easier composition process is offered to designers by the knowledge-based system.

7.3.1.2. External information accessing

Overall, over two-third of cases could complete the system composition without accessing external information while using this knowledge-based system (Figure 7-18). It can be said that most system development information is covered by this knowledge-based system. This can be used to explain why system composition in most of the cases with the knowledge-based development system could be achieved in a short cycle. Although few subjects still wanted to have extra information from external resources during the test, they were aimed at understanding the selected peripheral functions with inner control components. However, the need of external information while composing the intended systems with the target controllers in the knowledge-based system is relatively low compared to the need in traditional systems, since most information required was offered. While implementing software designs in a traditional development tool, the need of using external information is heavy, as designers rely on the information to discover the available solutions to implement the proposed systems.

Figure 7-18: Information acquisition during software implementation
Since the traditional development system are not capable of offering designers appropriate supporting information while composing the intended microcontroller-based systems, the use of vendor offered information is necessary. Accordingly, more time and endeavours are thus required to complete this development activity. This is also why this knowledge-based system was aimed at offering supporting information during this stage of development, and the offered information is actually helpful.

Reducing the effort required for software development can be achieved by preventing errors appearing, since the correction of errors requires additional effort which can be the same amount as the original development work itself. As the results displayed, problems appeared only in a few cases while using the knowledge-based software development system (Figure 7-19). However, the reason causing these problems was more related to the unfamiliarity with the knowledge-based software development system since subjects used the knowledge-based system for the first time. While using the traditional environment, designers have to handle their system composition plan as well as the vendor offered information, and mistakes related to “unable to meet system needs” might be made by designers, as they were unable to manage the available resources of the target controller to meet the intended implementation plan. Due to these mistakes, more time was therefore needed for correction to complete this activity, since effort to understand information for proper solution is needed.

Figure 7-19: Problems faced during system composition
By offering a simplified environment combining with supporting information for system composition to designers, subjects were able to complete this development activity easily. From the observed results, the knowledge-based system did achieve the purpose of improving microcontroller software development, as the system composition is the fundamental part of software development. The improvement can be achieved by the need for external information has been minimised, the development cycle has been shorten, and problems happening has been reduced.

After completing system composition, designers traditionally need to make documents related to their composed systems for the following development work, since the selected functions performed by the target controller have to be implemented in software. However, the part of documentation work is not required with the knowledge-based software development system, since the required functions of the target controller will be indicated after the designers finish the system composition. This can save designers time for this part of manual work and also maintain the consistency of the system development. Therefore, it can be said that this knowledge-based development system fully supports subjects while they are composing their intended systems.

7.3.2. FUNCTION CONFIGURATION

After deciding the implementation plan of the target controller, the intended performance needs to be configured in software. Traditionally, designers have to use assembly instructions to implement these proposed solutions. As software programs are treated as the final result of software development, coding these functions is highly considered, and a huge amount of manual work and time are needed to accomplish this activity. Manual work including information using and coding was considered as time consuming and requires skill to accomplish. In order to relieve designers' workload, supporting knowledge captured was transferred into guiding information as a part of the knowledge-based software development system for users to use. The overall performance of both development systems will be compared later.
While using the traditional development system, a common problem faced by designers is related to the implementation tool-assembly instruction. In order to use these instructions to build the implementation aims, extra learning time may be required, as the characteristics of the instructions are different from the syntax used in other programming languages. In the knowledge-based development system, the implementation tool is replaced by graphical interfaces. It can be seen that only a few subjects reported problems when they used the knowledge-based system. However, the number of problems found with the traditional development system varies due to the designer’s experience. Based on Figure 7-20, the graphical interface combined with the supporting knowledge can be said as a better implementation tool for software development.

The implementation tool, graphical interfaces offered by the knowledge-based system targeted not only on reducing coding problems, but also offering related information to help designers to achieve the implementation purposes. As for the implementation purposes, the intended functions performed by the target microcontroller might be controlled by several factors. Producing the proposed performance through manipulation of these control factors is a complex task. As these control factors targeting different aspects of performance, information is usually used to complete the implementation. This is also why problems are often found with the microcontroller software development work. While using this knowledge-based system, over 80% of cases could complete the function configuration without problems reported (Figure 7-
The improvement to software development can be the result of offering
supporting information in terms of implementation options and solutions to designers.
In addition, the offered information was organised in proper order, and it can be
delivered to users at suitable situations. This knowledge-based system does reduce the
number of problems compared to the traditional development process.

By offering the graphical interfaces to replace coding, and listing the related
configuration factors for function setting, function implementation in the
knowledge-based system is easier than manipulating the control mechanisms by using
coding instructions in the traditional development system. This viewpoint was
supported by the short period of time required to complete the implementation activity.
While using the knowledge-based software development system, 85% of the cases
were completed within a minute, and only 3% required more than 2 minutes (Figure 7-
22). In the traditional development process, the time period required is dependent on
the designer's experience, and usually the process requires 10 minutes or even longer.
It can be said that a compact implementation process was offered by the
knowledge-based system.
7.3.3. PRODUCING THE CORRESPONDING MACHINE CODE

After completing function implementation, the next stage is to produce the corresponding machine codes for the target controller. There is no coding process required in the knowledge-based software development system, as a source code program will be delivered after the implementation process. Now, the difference between the produced programs by the knowledge-based development system and traditional development environment will be presented.

As mentioned before, coding is the most considered work during the software development process, and problems occurred are originated from or related to this development work. This is because coding is a mixture of processes with several factors, e.g., the use of assembly instructions and data manipulation, and each factor has its difficulty. In order to produce correct results for implementation purposes, effort is needed to overcome the difficulty.

A frequently seen problem while producing corresponding machine code is that errors were found during the assembling process. These errors are related to wrong spellings or unidentifiable items (register or data type) appearing in the program. While errors were found, the machine code cannot be produced, and correction work was thus needed. Accordingly, the total workload increases as correction is needed. 97% of the
cases with the knowledge-based system produced error-free programs (Figure 7-23), and the corresponding machine codes can accordingly be created. The programs developed from the traditional environment all found errors during the assembling. Therefore, rectification work was needed for those errors found, and the corresponding machine code will be produced if all the errors have been corrected. Hence, the knowledge-based software development process is more straightforward than the traditional process.

Figure 7-23: Errors of produced programs found during compilation

7.3.4. PROGRAM TESTING

Errors causing the performance incorrectness exist in different formats inside the developed programs. These errors can be due to the incompletion of function setting, or wrong configuration of the control mechanisms. The assembling process is not capable of discovering these errors, and this is why a debugging process is needed. Since these errors can result from various reasons, the effort required in debugging is as much as implementation work itself in order to locate the errors and the correction work.

While testing the first produced programs using the traditional development environment, none of them could perform the intended functions. It meant that each of these programs required a further debugging process to amend the mistakes. 98% of the programs developed using the knowledge-based system could produce the
correct result directly (Figure 7-24). It can be said that the delivered codes by this knowledge-based system is accurate, and further debugging process can thus be reduced.

Figure 7-24: Debug process needed to correct the performance of produced programs

Information is thought as an important tool while debugging, as errors can exist in various formats. In order to correct these errors, designers require information to locate them. Since problems were found in all programs produced with the traditional software environment, all of the subjects required information. However, the situation of programs delivered by the knowledge-based development system is different. For about 92% cases with knowledge-based software development system, there was no need of using information (Figure 7-25), as the correct performance are found with the produced programs. Although the other 8% of cases asked information for debugging, it did not mean that the debugging processes were actually carried out in these cases. This is because the subjects considered that information was very necessary for the debugging process.
Because of the needs of debugging, more time is required to amend the problems found in programs to perform the intended functions. Therefore, the programs produced by the traditional development environment required more time and workload to perform the intended functions. The situation changed while using the knowledge-based software system. Since the delivered programs from the knowledge-based development system can perform the intended function directly, there is no need to debug (Figure 7-26). Thus the effort and time for debugging can be saved. There really is a huge difference between using the knowledge-based system and the traditional development environment.
The knowledge-based software development system aiming at solving the frequently seen problems during traditional development process was the desired solution to achieve the purpose of software development improvement. The required development tasks, which are inferred from system specifications, will be indicated by the knowledge-based system. Designers only have to complete the required setting for proposed functions to fulfil the needs of the intended systems, and no coding work was required for the implementation of these functions. Therefore, the need of the supporting information and the period of development cycle was reduced. Because of this characteristic, the difficulty of software coding can even be reduced and a compact and simplified software development process was delivered by this knowledge-based system.

The knowledge-based development system aims not only at helping designers to implement their intended designs, but also delivering accurate programs after the process. To produce correct programs, it is essential to ensure that the configuration factors were set by designers. Therefore, multiple checking mechanisms were deployed inside the knowledge-based software development system. They targeted on the completion of the function configuration process. Otherwise, designers will not be allowed to continue software development if any factors was found missing in the current development stage. Due to this characteristic, errors caused by the designers’ ignorance can be prevented. In addition, the performance of the intended functions will be simulated and presented. Designers can adjust function performance during the development process rather than change it after seeing emulation results. The composing codes for the intended software programs have been checked in advance, and therefore, very limited errors were expected to be found.

7.3.5. FULL PROCESS COMPARISON

After comparing the difference between the development activities of the existing and the knowledge-based software development process in stages, it was now aimed at examining the overall performance of this knowledge-based software development process. The factors used for comparison include problem encountered, time spent, required information, and the debugging process.
Problems encountered

The microcontroller-based system development was integrated of several activities, and each activity has a specific purpose to achieve. As different activity focuses on different aspects, various kinds of knowledge and information are thus required. Due to designers' knowledge and experience, problems might be faced during these stages. As microcontrollers were applied into more sophisticated systems due to their increasing capabilities, more problems might be faced during the development process, and the complexity of development thus grows. It means that more time and effort will be needed to solve these problems. Reducing the complexity of software development by compensating the inadequacy of knowledge access and manipulation is the main aim of this knowledge-based system. From the result in Figure 7-27, the encountered problems were reduced while using the knowledge-based system in each development stage. It can be said that the complexity of microcontroller software development has been effectively simplified by the use of this knowledge-based software development system.

Figure 7-27: Problems encountered during the development process

Information inquiry

To implement the intended software, information was often used to confirm the development related decisions as well as the details. While using the existing software development tool, vendor offered information was needed to confirm the implementation solutions for the proposed systems. Afterwards, information will then
be needed during the programming process for the use of coding instructions and implementation details. Also, the information will be used again while debugging the programs. It can be said that vendor offered information was highly required along with the traditional software development process all the way (Figure 7-28).

Figure 7-28: The needs of vendor offered information during the development process

The requirement of vendor offered information with the knowledge-based software development system was not high, as the essential information for software development is contained inside the knowledge-based system. Due to the offered information, software development has become more straightforward. Therefore, it is easier to complete the process with fewer distractions with this knowledge-based software development system.

Produced program

The performance of the produced program was highly considered. It is critical to make sure that the performance of the produced programs fulfilled the needs of the proposed system; otherwise the software development process is meaningless. The debugging process is required if the produced program could not perform its intended functions. As the reasons causing the misleading of the functions are random, detailed examination of the coding program is required and the effort required to debug can be as much as the software development itself. From the cases, the debugging process for existing software development system exceeds 20 minutes. As correct result can be
produced by the knowledge-based system, the debugging process was not required (Figure 7-29). Hence, the amount of time and effort for debugging could be reduced.

Figure 7-29: Quality of the produced programs from different development processes

![Quality of the produced programs](image)

**Development time**

Figure 7-30: The required time for software development

![Development time](image)

The average time for using the existing software development tool to build the intended software took more than 2 hours on average. The period of time includes information finding, software coding, and debugging. For the knowledge-based
software development, the average time to complete it was about 20 minutes. A large difference is found in the amount of time for software development (Figure 7-30). The amount of time saved comes from the reduction of coding work, information accessory and debugging.

7.4. Discussion

From the comparison between the traditional and the knowledge-based software development process, it can be seen that an improved software development process was offered to designers while they are using this knowledge-based software development system. The improved development process is the result of characteristics shown by the knowledge-based system. These characteristics include an effective development process, user-friendly implementation environment, and quality program delivery. Later on, these characteristics with the offered benefits will be discussed.

7.4.1. EFFECTIVE DEVELOPMENT PROCESS

During the traditional development process, most of the time is used to understand the system development issues and the implementation of the solutions. In addition, further workload is required to accomplish those development tasks. As the amount of work required is a lot, it is thus considered that the traditional software development is inefficient.

Using this knowledge-based software development system, the required software programs of the target controller can be developed according to the system requirements. During this process, development activities were fully supported by this knowledge-based development system with the offered guidelines to complete these activities. Therefore, less effort will be required to complete the software development process. After the implementation process, accurate source code programs will be delivered to designers. Accordingly, the development cycle can be reduced and the efficiency of software development can be improved as the result.
7.4.1.1. Short development cycle

The development cycle time is an important index factor while considering software quality, since a longer development cycle means more time and higher cost are required to accomplish the intended software development. For quality software development, it was aimed at completing the software development within a short cycle. However, using the traditional software development environment, the development cycle is difficult to control, since several factors, e.g., information handling and assembly coding, can deeply influence the result. Therefore, the software development process can be costly and low in efficiency due to the large amount of manual work.

In order to effectively reduce the software development cycle, the effects of personal factors were aimed to be offset by knowledge offering. The knowledge is converted into different formats to help designers to implement their design. With the support from offered knowledge, the software development process becomes simpler than it was. As the result, the required development cycle had been obviously shortened while using this knowledge-based system.

The shorter development cycle is because of the following reasons: a straightforward development process, graphical performance-oriented implementation panels, and automatic program generation. Due to the new method for software development, each required development activity is simplified, and especially, the traditional implementation method, coding, is not required. Hence, the coding related problems were being reduced. Accordingly, the cost for software development can also be reduced.

7.4.1.2. Consistent development process

The performance of a target controller can be the integration of several individual functions. In order to perform the proposed aims, the configuration of the required peripheral functions and the implementation of intended control strategy are both essential. By leading designers through all the settings of fundamental peripheral functions for the proposed systems, the knowledge-based system can then offer
system-based events for designers to compose their intended system behaviour with control strategy. Therefore, software development is completed in an organised order.

The software development, beginning from system composition to control strategy implementation, was guided by the knowledge-based system. In addition, the required tasks will be indicated by the knowledge-based system sequentially. During different stages of the development process, the completion of every intended implementation task was required, as each function was the fundamental basis of system performance. By splitting the whole implementation task into individual parts properly, each part can focus on a single purpose. During the process, the knowledge-based development system indicated each part of the implementation task and offered guidelines to complete these tasks. In this process, designers have to only focus on one implementation task without spending time and effort to manage the development process. Therefore, the complexity of software development was reduced.

As the required functions were oriented from system composition, and the control strategy was implemented in system-based events, the development of the intended software is accomplished in a coordinated order. In addition, the required tasks were completed in a sequential order. Therefore, software development was carried out consistently.

7.4.1.3. Organising available resources

During software development, the proposed performance is composed by functions, and function configuration is achieved through the arrangement of available resources and the setting of related factors. To successfully achieve the purpose of function configuration, intended solutions of the proposed functions and corresponding factors need to be set properly. As the available resources of a microcontroller is fixed, the organisation of the available resources for the proposed functions is an essential task.

Peripheral functions are often used as the fundamental core of a microcontroller-based system. In order to meet the needs of the proposed systems, the ability of organising the available resources is required. In other words, the skill to produce the proposed performance through the manipulation of these available resources is needed.
However, one of the parts of microcontroller software development is likely to be the system resource management. To fulfill the system needs, a proper management method was integrated inside a microcontroller software development process to reduce the interference between proposed functions.

For peripheral functions, control factors need to be set to produce the required performance, as well as the corresponding pins. If the corresponding pins did not configure properly, the proposed functions might not be performed. Therefore, mechanisms aiming at checking the missing issues from the configuration process were also provided. Furthermore, the mechanism also can reserve the corresponding pins from other usage as the peripheral functions were selected. Accordingly, the possibility of function corruptions happening could be effectively reduced. By proper organisation of the available resources, the required functions could be performed by the target controller to fulfill the needs of the proposed systems. In addition, the full potential of the target controller can also be realised.

7.4.2. USER FRIENDLY IMPLEMENTATION ENVIRONMENT

As the traditional software development method, coding, is considered as an inefficient solution, a new method of software implementation is offered to designers within this knowledge-based software development system. In order to further simplify the microcontroller software development, effective factors to enhance the capability of implementation tools were also applied. These were the essential factors while developing the knowledge-based software development system.

7.4.2.1. Limited external Data accessing

The occasions of applying information appear from the beginning to the end of the microcontroller software development process. Due to the difference of the required knowledge, a three stage software development process was implemented inside the knowledge-based system. By this solution, captured knowledge inside the system can be effectively offered to designers while carrying out the intended development activities.
Information is considered to be an important factor during microcontroller software development. This is because many detailed issues for peripheral function configuration are contained inside a software program, and the setting of these details requires information to support in order to deliver the correct results. Unfortunately, the presentation of current available vendor offered information could cause problems whilst being applied, as information presenting was keen on displaying the technical factors of the target microcontroller and other prospects such as application and performance were hardly mentioned. To apply the vendor offered information for proper usage, experience or expertise were often needed. However, the gaining of experience needs some learning and practice which takes time to achieve. In order to relieve the workload during software development, direct help coming from the supporting information was offered through the development process.

From the starting of system composition to function configuration and control strategy, information was offered in different formats by this knowledge-based software development system. Also, information was offered to meet designers’ immediate needs. As the implementation process was carried out in a sequential order, designers do not have to handle too much information at anyone time. Therefore, the offered information can actually be used effectively and the need to use external information was reduced.

7.4.2.2. Performance-oriented configuration

Instead of the coding process, the knowledge-based software development system offers designers an alternative solution for software implementation: performance-oriented function configuration interfaces. For different peripherals, different interfaces will be used. These interfaces can help designers to focus on the essential issues for the intended functions, and then convert these issues into corresponding codes in the produced program. Therefore, designers only need to decide the performance or the setting of control mechanisms rather than assembly coding.

Besides offering the essential factors for function configuration, direct performance setting options were also available. The proposed peripheral functions with
corresponding performance can be produced through direct manipulation of the offered options. With the implementation of graphical interface, designers were allowed to implement their software designs in a simplified environment with the visual components. Accordingly, the difficulty of function implementation is reduced by replacing of the traditional coding process. Due to this transformation, software development can be processed in a simple way.

With the performance-oriented configuration interfaces, function configuration was achieved requiring only a little amount of effort, since most of the effort such as information handling and manual coding needed has been removed. Therefore, a fast function configuration process with accurate results is achieved by this performance oriented configuration interfaces.

7.4.2.3. System behaviour for control strategy implementation

The proposed system behaviour can be expressed as the sequence of system events. To accurately implement system performance in microcontroller software, the relations between system events and functions performed and the sequence of system events was considered. In addition to improving the consistency of software implementation, a direct performance setting from the interrelation of system actions and the intended control strategy was offered to designers by this knowledge-based system.

To compose the control strategy in microcontroller software, it is essential to understand the relationship between functions performed and system events. Thus, system performance can fully be implemented by the applications of the target microcontroller. To reduce the workload at this stage, a direct method for system control strategy implementation was applied – using system events to compose the intended control strategy. Therefore, the implementation of the intended system control strategy has become more straightforward than it was.

As the required peripheral functions have been configured in earlier stages of software development, it meant that possible system events performed by the microcontroller could also be inferred, as well as possible conditions. In spite of the required functions
performed by the target controller, using possible system actions to compose the intended system performance is easier and more consistent.

7.4.3. QUALITY PROGRAM DELIVERY

Software quality considers not only the implementation process but also the developed results. The improvement to the implementation process was described in the previous sections. Later on, the factors making the improvements to the produced results will be described.

7.4.3.1 Automatic code generation

The main aim of the microcontroller software development is to build a program which is capable of performing the intended functions. It can be said that all the input effort in software development was used to produce the result that can fulfil system needs. Traditionally, the developed result is through the coding process of the software implementation. Due to the coding process, software development is considered as a complex and time-consuming process.

The traditional microcontroller software development which refers to the use of coding instructions to program is complicated work for microcontroller-based system designers. Firstly, designers have to figure out the related registers and corresponding control mechanisms for the intended functions and performance. Then, the use of available programming instructions to achieve the intended setting was considered. Afterwards, the designers need to compose the intended function performance through the control of related registers. Also, designers also need to be aware of the relations between the intended system events and the corresponding functions performed by the microcontrollers. After these procedures, designers can develop a software program, which can perform the proposed behaviour to meet the system requirements by the available instructions.

In spite of the issues such as software maintenance, the developed programs were proposed to perform the intended functions. Accordingly, the actual performance of the developed results was important, as further work will be required if the first
produced result does not meet the system needs. The work to amend it can be as much as the development work itself. To avoid this, the developed programs from the knowledge-based system were delivered directly from the issues decided during the implementation process, and no coding was required. As the delivered programs were composed by pieces of codes, which were checked through properly before being captured into the knowledge-based system, it was expected that no problems would be found inside the delivered programs.

As there is no need of coding, most of the workload of traditional microcontroller software development process has been reduced. Furthermore, the delivered programs are capable of performing the intended functions. These characteristics of the knowledge-based system are due to the automatic code generation.

7.4.3.2. Software quality

The quality of software programs was important in the current software development, as the cost of software development and maintenance is increasing. Performance of the developed programs is considered as the first issues of software quality, since a program that cannot perform its intended functions cannot be said to be a quality program. Therefore, delivering error-free and performance correct programs was the basic aim of the knowledge-based development system. Furthermore, in addition to make the produced software program applicable for further implementation, program structure and available comments of the developed programs play important roles. This is because the purpose of an assembly code program is too difficult to understand through the instructions. Therefore, an appropriate program structure and enough comments are essential to build quality microcontroller software programs.

During the development process, there is no fixed rule to compose software programs. Hence, different designers can manage their programs in different structures. Although the structure of software program does not affect performance, a disorganised program can be very difficult to understand as well as to maintain. Currently, the most common method of making assembly programs easier to access is through comments and proper program structure. With the listing of comments, designers can obtain more accessibility and awareness of the program. Therefore, comments were found
inside the programs produced by this knowledge-based software development system, and they were aimed at raising the quality of the produced programs.

The demand of amending software programs was also faced by designers as errors were found during assembling or the performance does not fulfil system needs. Traditionally, designers have to go through the configuration of detailed control mechanisms in software. Due to the limitations of current available software development tool, however, designers cannot check the possible performance of the intended function until it was finished. The performance checking iteration which includes changing data setting, assembling and running on emulator is a very time consuming process. In order to improve this drawback, a performance simulation tool was provided within this knowledge-based software development system. This tool will allow designers to see the possible performance of the intended functions during function configuration. Therefore, the amending of function performance can be achieved before producing the final result. It saves time for designers and also delivers a reliable result. Furthermore, the piece of codes for the delivered results was examined before being embedded into the knowledge-based software development system. Accordingly, very limited errors were expected to be found in the program produced by using the knowledge-based development system.

Another method used by this knowledge-based development system for further raising software quality is through proper structure. The produced programs are composed of blocks of codes, and each block targets on a single function. The full performance of the intended software programs was achieved by accumulating these functions. By separating a whole program into blocks, the program is easier to understand along with the provided comments. This was the solution used by the knowledge-based software development system to produce quality software programs.

7.5. Limitations of the software development expert system

The development of this knowledge-based software development system aimed at helping designers, however, still has some limitations. These limitations might come
Chapter 7 - Evaluation of the software development expert system and discussion

from the structure of knowledge bases or current technologies. These limitations will be examined thoroughly.

7.5.1. MULTI-LEVEL CONDITION SETTING

While implementing system control strategies with this knowledge-based development system, proposed conditions are allowed to set before the intended system actions. Condition setting can help designers to realise their control strategies properly. In order to produce required actions with assembly instructions, labels, indications to appropriate actions, are needed in codes. Inside this knowledge-based development system, labels were used to produce the proposed sequence of execution codes for the implementation of system control strategy. This is why labels can be found after each condition.

To set a complicated conditional action, a sequence of conditions may be required and then followed by an action. As a condition can be set at a time by this knowledge-based development system, it therefore takes a few steps to complete the whole condition setting for a complex conditional action. Hence, designers might need to re-structure their conditions in different order to build the intended control strategy using this knowledge-based system.

7.5.2. FIXED MODE OF IMPLEMENTATION

The produced programs through the software system were composed from the data stored inside the knowledge bases. In order to meet most users’ needs, a general platform for pieces of codes to load from different knowledge bases was applied in this software knowledge-based software development system, as a proper program structure can lead to ease-to-understand and benefits following the maintenance issue. Although this platform can manage to structure intended programs, this platform also leads to a fixed type of implementation.

While fulfilling system needs by functions, the knowledge-based software development system will offer designers indications. This can help designers to discover the proper implementation plan. Since this expert system will offer the same
options while similar cases are faced by designers, a fixed mode of implementation plan might be found. This was due to the same solutions which could be found among similar applications. However, these functions can still meet designers' needs and fulfil system requirements.

7.5.3. LIMITED ASSEMBLY PROGRAMMING EXPERIENCED GAINED

By using this knowledge-based software development system, it was believed that the efficiency of microcontroller software development could be improved, and ready-to-assembler programs were delivered afterwards. Therefore, designers need only to consider the design content during the development process, as there was no need for them to use assembly languages. Accordingly, designers can avoid problems related to using assembly instructions. However, this characteristic can also lead to a disadvantage: learning very limited experience about assembly programming.

To be familiar with assembly language, a learning and practice process is needed. Often, designers can gain experience while they are building programs for microcontrollers. Since no coding process was needed with this knowledge-based software system, the users had no chance of gaining experience about assembly language. For inexperienced designers, maintaining and accessing the built programs were very difficult tasks for designers to achieve. In order to increase the accessibility and ease-of-maintenance for novice designers, comments were added inside the delivered programs. Although the comments can increase the understanding, the comments help very little in programming in assembler for the coming development tasks.

7.5.4. ADAPTIVE LEARNING IS UNDER ESTABLISHING

Although the main aim was to improve current microcontroller software development by simplifying the implementation process, the quality of the delivered results was also emphasised. The quality of the delivered results can be considered from several different aspects. The produced program which should be error-free and the performance of the intended functions was treated as the bottom line. This is because much extra effort will be required for software correction to achieve the aim of system
modification. To reduce the need of further effort, a functional correct software program was proposed as the developed result. A checking mechanism has been deployed for this purpose inside the knowledge-based development system. For an advanced solution, a wrongly composed design should be noticed and refined with the help of the knowledge-based system during the earlier stages. In order to achieve this aim, an adaptive learning function was required.

The purpose of the adaptive learning function is to learn from the mistakes which users made previously and then prevent similar mistakes from happening. This can further increase the capability of this knowledge-based system as this system can diagnose the larger mistakes and apply proper solutions to implement the intended designs.

The capability of a knowledge-based software development system is not only to detect mistakes made during the configuration process, but also to prevent the same mistakes from happening twice. In order to achieve this aim, it is critical to get the function of adaptive learning developed inside the system.

7.6. Summary

After the implementation of the development of the knowledge-based software development system, the functionality and performance of the system was evaluated. The evaluation was done through a comparison of different development process by using traditional software and knowledge-based systems. Two case studies were then carried out to observe these two software development processes.

It was observed that a longer development cycle was needed during the traditional software development, and the reasons came from external information accessing, design refinement, coding and program debugging. Hence, it can be said that the traditional microcontroller software development is a challenging task to novice designers. With the knowledge-based development system, development tasks were supported by knowledge and offered with solutions. The software development
process with knowledge-based system was easier compared with the traditional process.

Through comparison, benefits offered by the knowledge-based system can be found from several aspects, for example, the user-friendly interfaces, and automatic code generation. These benefits allow designers to complete their intended software development with a short cycle and quality results were delivered afterwards. In addition to the implementation process, there was no need to code. Thus, an effective microcontroller software development was presented by this knowledge-based system.

Due to the automatic code generation, additional benefits were also delivered such as the accuracy of results. However, some limitations might follow. The limitations include fixed mode of implementation and complex condition setting, but these limitations will not affect the system to produce suitable software programs. An issue related to this knowledge-based development system was experience gaining. Since no coding work was required, only very limited assembly coding experience will be gained by the designers.
Conclusion and further work

8.1. Conclusion

The thesis presents an advanced development system for software development of microcontrollers. The key idea of the advance development system is to support designers to produce the intended software programs after the applications of target controllers have been decided. The support for microcontroller implementation tasks was achieved by continuously delivering the missing information during every stage of software development. Specified information was offered to designers aiming at accomplishing the current development task. The offered information acts to bridge the gap between the intended performance and available functions. The experimental test that validated information delivery for the software development process showed positive results. From these results, it was concluded that the knowledge-based system is a potential alternative for a fast system prototyping and an advanced microcontroller software development.

The thesis recommended that the graphical interfaces and the automatic program generation were also beneficial to the implementation of microcontroller software development. The software implementation was achieved within the application of graphical interfaces and its feasibility verified, and the traditional coding process was considered as a challenging task. This is because the graphical interfaces not only achieve the functionality of software implementation, but also organise the sequence of development activities. However, the graphical interface feature offers a user-friendly implementation environment, a consistent development process and the tool to deliver information to support development activities. No coding problems were faced within the knowledge-based software development system, but accurate, and ready-to-assemble programs were delivered after the implementation process. It extends the usability of the system.
Chapter 8 - Conclusion and further work

8.1.1. SUMMARY OF WORK UNDERTAKEN

During the course of this research, various issues related to microcontroller software development have been identified. These include the applications of microcontroller, system development methodologies, software development process and the improvement made to current development systems. Their abilities as well as limitations were noted. In addition, microcontroller software programming had been undertaken to be familiar with the software development process where a new development system will be appreciated. The literature survey has thus enabled the author to propose a viable solution to the existing limitations of current software development environments as well as the current software designers' wishes and demands.

The proposed solution consists of a knowledge-based software development system with graphical interfaces and automatic code generation. The knowledge-based system was specially designed for microcontroller-based system implementation as well as useful for designers to generate programs within a short development cycle.

The software development system used to demonstrate a knowledge supporting strategy is useful in software development. The strategy uses a combination of supporting information and possible solutions. The knowledge was categorised to enable support in different development stages. The knowledge was captured from available vendor offered information and solutions to possible problems were formed. The solutions were structured to form knowledge bases by using Visual Basic 6, and corresponding tests have been carried out to examine the correctness of the offered solutions. The tests involved case studies of using the knowledge-based system to produce microcontroller-based systems. The tests had been useful in the tuning of organised knowledge as well as observation of the produced programs according to the conditions from system compositions.

The implementation of graphical interfaces was integrated and programmed by using Visual Basic 6. It was necessary to use visual based programming languages due to the poor interfacing capabilities of other types of languages. Practical studies by inviting novice designers to use this knowledge-based system to build
Chapter 8 - Conclusion and further work

Microcontroller-based modules were carried out. The responses from the invited subjects and the processes of development were recorded and the produced results from this knowledge-based system were observed. These results are then used to compare with another study focused on using the traditional development environment to build software programs. The comparison has proved that knowledge-based system was able to produce the correct results (programs) after system configuration and simplify the software development process.

In addition to the simplification of software development, an automatic code generation was proposed in order to increase the functionality of this knowledge-based system. A function performance simulation was also offered during function setting, and it helps designers to realise the created performance before testing on board. The produced results were ready-to-assemble programs. Evaluation of these produced programs was also carried out during the case studies. The results from evaluation can prove the correctness of the created programs and feasibility of the system. Observations were also carried out to discover limitations of the generated code, such as fixed program structure, and unchanged implementation solution.

As a conclusion, an engineering prototype was designed, built and tested. The knowledge-based system unit had demonstrated technical feasibility, including a compact development process for microcontroller software, and graphical interfaces for system design implementation and automatic code generation to facilitate software implementation.

8.1.2. CONTRIBUTION TO KNOWLEDGE

The main contribution of the thesis lies in the development of a knowledge-based system for a compact and improved microcontroller software development process. A new presentation for system composition and software implementation has been identified. The research work was indeed a demonstration of mechatronics software development system where integration of engineering based on a coherent fusion of mechanical and electronic engineering and software science. The results of this effort indicated that the development of such system is feasible, and that such a system
potentially offered significant benefits to system designers. The contribution of this research will be listed as follow:

1. The combination of knowledge-based systems with microcontroller software development
   1.1 Offering information for system composition
   1.2 Solution guiding for function configuration
   1.3 Automatic program generation
2. The achievement of fast system prototyping
   2.1 Time-taking and effort-demanding software coding is reduced
   2.2 A consistent and compact system development process
3. An integrated system development process
   3.1 Bring software implementation close to system development
   3.2 Interfacing is included as part of the development process
   3.3 System control is implemented based on sequential system actions

Part of this research work was presented in the following publications:


8.2. Further work

The research work done has achieved an efficient microcontroller software development and delivered quality programs after the implementation by the application of knowledge-based systems. The thesis marks the beginning of research on the conceptual proposal of an advanced software development system for applications of microcontrollers. Further work of this research is aimed at increasing
the capability of the knowledge-based system and offering more benefits to software designers.

Enrolling more specific components can provide great benefits to microcontroller-based system development. While considering the trade-offs of cost and performance during system development, available options or possible solutions to reveal the availability of target controllers were proposed by designers. In addition to the target controllers, the external components were proposed to be configurable with more specifications. As the purpose of this development system is to support the software development process, corresponding knowledge bases related to these issues will be needed. Since these knowledge bases do not stand alone, it means that cooperation of the knowledge bases is necessary, due to the implementation targets related to the target controllers need to be kept for further applications. Accordingly the complexity management of the system development needs to be organised in good order. Furthermore, the function of performance simulation was also proposed in order to achieve accurate system development. Hence, system performance modification can be done before the whole system was completed. It was believed that the cost of system development could be reduced by effective simulation.

8.2.1. EFFECTIVE SYSTEM SIMULATION

A strong relationship was realised between the applications of microcontrollers and control systems. As the control effort was emphasised in such applications, the design of the intended system was aimed to be simulated before implementation. Due to the function of simulation, the possibility of design failure could be avoided. Accordingly, the cost of system development could be reduced and quality raised.

Microcontrollers usually act as the core of systems. Therefore, coordinating other external components to produce the intended performance becomes a major implementation purpose of target microcontrollers. For this purpose, the use of embodied peripheral functions with control strategies could be needed. However, the final delivered results from the functions and control strategies remains unclear. In the past, a prototype system composed by both hardware and software was used to exhibit the system performance, but the testing process was cost and time consuming. It was
thus aimed to reduce this part of cost by offering the simulation function. The simulation is not only about to profile the target controllers, but also to achieve the simulation of the whole system. In order to achieve this aim, detailed information related to components applied in systems was required for precise simulation results. More pre-set functions will surely be needed in order to produce fast simulation.

8.2.2. DOCUMENTATION FOR PROPOSED SYSTEMS

Documentation was considered as an important issue during software development, since the information recorded was useful for further maintenance and usage. Often, documentation was done manually by designers, and therefore the recorded information was not standardised. From the viewpoint of software development, poor documentation can cause further costs later. A documentation tool was thus proposed to be encompassed inside this software development system.

Documentation includes not only the implementation solutions but also the listed comments inside the software development. This is because the programming syntax might not be easy to understand. In order to increase the readability of developed programs, a proper text for comments is thus needed. These further comments were proposed for good understanding of the programs as well as ease of maintenance. Also, a proper document to record system implementation was required. Therefore, more details of the implementation plan could be presented. It was aimed at increasing the understanding of the produced software programs further and extending the maintainability.

8.2.3. TRADEOFFS FOR SYSTEM IMPLEMENTATION

The trade-off between cost and performance was frequently treated as the principle for an implementation plan, as reducing product cost was considered to be an important issue during system development. Thus, the solutions to implement the intended functions were important. Often the trade-off lies in the implementation plans of hardware or software.
Due to the experience of system development or awareness of possible solutions, designers might not be able to distinguish the difference between implementation based on hardware and software. Accordingly, the possible options might be ignored and cause increases in cost of total system development. A knowledge base which contains the information related to trade-off solutions was proposed to be encompassed in the development system. During the procedure of system configuration, available options aiming at offering the implementation trade-offs can be presented for designers to choose. A system which meets all its requirements can then be composed and delivered with low cost.

8.2.4. INCREASING THE UNDERSTANDING OF ASSEMBLY LANGUAGE

The capability of using assembly programming language was through a learning and practice process. For such a process, it is usually time consuming. Although this software development can deliver assembly programs for designers to compile, it does not improve the designer's ability in assembly coding. The function of educating designers was also proposed in the further development of this system.

Assembly instructions are microcontroller-dependent programming languages. It meant that different target controllers need different instructions to achieve the program. Although implementation plans are required while using this knowledge-based system, coding work was not required. In order to develop the designers to be more experienced along with the implementation process, an educational method was aimed to be embedded with the development system. Proper coding instructions will be presented while implementing designs. It was hoped that designers could gain useful experience from this process.

8.2.5. INTERACTIVE IMPLEMENTATION ENVIRONMENT

The software implementation process was usually processed under a designer controlled environment, where no interaction exists between the designers and the development tool. Without interactions, the development tool cannot perform its full potential to help designers while implementing the design.
Thus, an interactive implementation interface was aimed to be applied for the development system. With this interface, system simulation, programming experience gaining and trade-off of components were proposed to be delivered. Therefore, this feature can be treated as the major improvement for future work of this knowledge-based development system.

8.2.6. ADAPTIVE LEARNING EXPERT SYSTEM

Applications of knowledge-based system are capable of offering options formed by information and knowledge to solve problems faced. Therefore, completing the intended tasks is relatively easy with the support of knowledge-based system. As the offered options are specified for problems, more information will be needed if the knowledge-based system aims at wide area.

For this knowledge-based software development system, knowledge and information for microcontroller system development was captured inside the system. However, to cooperate with this system in microcontroller-based system development, it will be critical to deploy the function of adaptive learning. This function aims at gaining experience from mistakes made during the development process and then preventing similar situation from happening in the future. Hence, the performance of this system can be further improved.
References


[136] Kevin L. Mills and H. Gomaa, "A knowledge-based approach for automating a design method for concurrent and real-time systems," SEKE '96:


Appendix A

A two DC-motor control module through serial data communication

Introduction
Microcontrollers are popular with those embedded sophisticated peripheral functions. In order to understand the capability of this developed knowledge-based system, a complex control system was proposed. This control module contains of two DC motors and the use of serial communication. Through the received data, the intended control order can be given to these two motors.

Objective
Serial communication is widely applied in digital systems, as the required number of wires for data communication is significantly reduced. Therefore, it is an attempt to compose a two DC-motor control module, and the control command is sent through the serial communication. As two motors are composed within this system, the received data needs to be identified to give the correct commands to achieve the intended control.

Tasks
Data receiving through serial communication is needed in this module, as well two PWM functions. The serial communication is for the purpose of command receiving, and these two PWM are to achieve the aim of motor speed control. For these two PWM functions, they can be based on the same frequency. Hence, while the received data is H'00', Motor 1 should stop and Motor 2 should stop while receiving H'0F'. While receiving H'FF', motor 2 should start.

Functions performed by the target controller
Two PWM functions and the designated frequency is 200 Hz, initialise with 50 percent power on.
Two output pins, and they will be used for the direction control of these two motors.
Serial communication (USART), which is operating in asynchronous mode, and proposed baudrate is 9600 and the data format is 8-bit.
Task scheme

Enable Asyn. USART

Enable first PWM

Enable second PWM

Enable two output pins

Load received data:
If received data equals H'F0', then Motor 1 stops

Load received data:
If received data equals H'OF', then Motor 2 stops

Load received data:
If received data equals H'FF', then Motor 2 starts

Configure the required functions for the system

Step 1: checking the received command, if the data equals H'F0', motor 1 stops

Step 1: checking the received command, if the data equals H'OF', motor 2 stops

Step 3: checking the received command, if the data equals H'FF', motor 2 starts
Appendix B

DC motor driving module

Introduction
Pulse width modulation (PWM) is often the control method inside a microcontroller-based DC-motor control system. To achieve this control aim, a proper frequency for the PWM waveform is very important; as a too fast or too slow frequency is not able to drive the DC-motor well. The control of motor speed is achieved by the change of ON/OFF percentage of the waveform.

Objective
Apply PWM to drive a DC-motor through H-bridge, and the motor speed can be changed due to the data manipulating from switch input board. To create a 40 Hz PWM waveform to achieve this control aim, and this is because a too fast or too slow frequency is not able to drive the DC-motor properly.

Function performed by microcontroller
1. Enable an eight-bit port as an input port.
2. Enable two output pins; one for direction control and the other for pulse generation.
3. Enable timer with its corresponding interrupts. (Generate regular pulse every 0.1 msec. The purpose will be accomplished by a timer with its interrupt function and an output pin. By toggling the value of the output pin every interrupt; it is possible to generate regular pulses.)
4. Enable PWM function. Set PR value to 255 and Duty Cycle to 255 as the initial setting. Select the standard resolution. The base timer of PWM will count input pulses generated.
5. Load input data into DC register as new PWM value.
Appendix B

DC motor driving module

1. System diagram

```
Adaptor board

Main adaptor board

PWM

H-Bridge

DC motor
```

2. Task scheme

Enable required functions

- Enable an input port (8-bits)
- Enable two output pins
- Enable timer and its interrupt
- Enable PWM

Motor action (speed change)
Load the data content from the input board for new PWM value

Connect with the switch input board

- One for direction control, the other for pulse generation
- The timer interval = 0.1ms, Generate pulses in interrupts
- \( P_r = 255, \quad d_c = 255 \)
- Load input data into DC as new PWM value
Appendix C

Appendix C

Serial communication

Introduction
Serial communication is an important issue because it has been used as a fundamental function in microcontroller-based systems. "Parallel in, parallel out" is the common method used for communication, but it requires as many wires as the data width. In order to solve the problem of wiring, serial communication is then applied.

Objective
The objective here is to read 8-bit data from the input board then display the data in the output board. The data transmission applied in between is Asynchronous mode serial communication. This design demonstrates the idea of "parallel in, serial out" and "serial in, parallel out".

Task
The data communication through serial receiver and transmitter is required by this task. In order to examine the data sending and receiving is successful, one switch board and one LED board are required.

1. The input board needs to be wired with an I/O port (suggestion-PortC).
2. The output board need to be wired with an I/O port (suggestion-PortD).
3. Select USART for serial communication (wired Tx and Rx pin)

Function performed by microcontroller
In order to achieve this task, two I/O ports and USART function will be required.
Select PortC as an input port.
Select PortD as an output port.
Configure the USART for serial communication (Asynchronous mode is the proper mode. Baud rate can be 9600. Tx and Rx are both in 8-bit mode, and no interrupt functions are required).

Software design (Actions to take after peripheral function configuration)
Tx action- Load the data contents of the selected input port (the input data is located inside the register of data memory).

Output data via port –display the received data (the received data RC1data is also located inside the register of data memory)

(To fetch the data from the switch board and to receive the transmitted data are achieved by the microcontroller automatically.)
Appendix C

Serial communication

1. System diagram

2. Task scheme

Enable required functions

- Enable one binary input
- Enable one binary output
- Enable Asyn. USART
- Getting data from Input
- Sending the data via Tx
- Loading data from Rx
- Showing the data in Out

Load the data from input port into the serial transmission
Load the received data into the input port
Appendix C

Analogue signal converter

Introduction
Converting analogue signal is an important issue in microcontroller systems, because the real world is mostly composed by analogue. Therefore, the use of the Analogue Digital Converter (ADC) in a microcontroller system is essential if the system deals with real world issues. To apply analogue sensors in microcontroller systems, it is important to convert the output value (voltage) into digital form.

Objective
To convert a random analogue signal (voltage) by the A/D converter embodied on the microcontroller. Display the converted result on a LED board.

Task
Analogue signal (voltage) needs to be converted into digital form which can be achieved by microcontroller. To view the result, it is aimed to wire an output board to the microcontroller for displaying the result.

1. Wired the output board with an I/O port (suggestion-PortD).
2. Select ADC for analogue voltage conversion. (wired the analogue voltage with AN0 pin in [PortG, bit3])

Function performed by microcontroller
One I/O and ADC function are required in order to achieve this task. Select portD as an output port.
Configure ADC function (AN0 [PortG, bit3] for getting the analogue signal, select the main part of the converted result.)

Actions to take after peripheral function configuration
Output data via port - Output the A/D converted result (ADCxHI, which should be located inside registers of data memory)

Analogue voltage sampling- it is set automatically through the ADC configuration.
A/D conversion- the action commences automatically
Appendix C

Analogue signal converter

1. System diagram

![System diagram]

2. Task scheme

Enable required functions

- Enable one Analogue input port
- Enable one binary output port
- Enable ADC function

- Sample the voltage from DAC
- Start the A/D conversion
- Showing the data in out. board

- Get data from the proposed ADC pin
- Start the conversion

Showing the data in out. board
Appendix C

**System timing and signal capturing**

**Introduction**
For real-time control, timing is highly considered. In a microcontroller chip there are two functions to perform the timing function; one is timer and the other is capture, since they are both timer-based function. Interrupts will be generated once the counting period of the timer is completed. According this character, timer can be used to create regular events, such as generate pulses regularly. The focus of capture on the specific events appearing on capture pin, and a corresponding interrupt can also appear due to the capturing of the specific event. Since the capture is also based on timer, the specific event is appearing or not during a specific period of time can be detected.

**Objective**
Create a 10 KHz pulse generator by timer0, and the pulse is sent out through PorB, bit0. The pulse output pin is directly wired to capture1 (PortB, bit1). Capture is aimed to capture every rising edge of the generated pulses. While the 1st rising edge is captured, the capture interrupt is aimed to display the input data on the output board.

Task
1. Wired the input board with PortC
2. Wired the output board with PortD
3. Select Timing
4. Select Capture.

**Function performed by microcontroller**
Two I/O ports, one timer and one capture function are required.

1. For timer, Select Timer0 is aimed to act as a pulse generator. In order to create a 10 KHz pulse generator, the time set for the timer should be around 0.1 msec. In addition, a pulse should be generated once the interrupt appears by selecting the regular pulse generator during interrupts. PortB, bit0 (RB0) is the pin used to generate the pulse.

2. For Capture, select Capture2 is aimed to catch every 1st rising edge. The proper mode for capture to set is three capture mode. The corresponding timer (timer 3) is set to 1 msec.

Configure portC as input board and configure portD as output board.

**Actions to take after peripheral function configuration**
While timer’s interrupts occurs, generate pulses.
Apply capture 2—the output data via portD occurs while the 1st rising edge is captured. Select PORTC from the register list in order to display the data contents on switch board.
Appendix C

System timing and signal capturing

1. System diagram

![System diagram diagram]

2. Task scheme

Enable required functions

- Enable one binary input port
- Enable one binary output port
- Enable Timer function
- Enable Capture function

When TMR interrupts happen:
1. Create pulses for capture
2. Get the input data

When CAP interrupts happen:
1. Output the input data in output board
Appendix C

Step motor control module

Introduction
Step motor has been getting popular now as it provides excellent position control. The focus of this module is to drive the step motor via the FET board. By switching the FET in correct order, it is possible to drive a step motor. Before changing the status of the FET, it also requires a time delay since switching FET too fast is not possible to drive the step motor.

Objective
Try to drive the motor at 30 rpm. (The time delay in between different FET status is about 2 msec.)

Task
Selecting a step motor module. Select 4 phases, 4 steps and no feedback component is required.
Switching the FET boarding in proper order to turn the step motor on. The switching of the FET can be simply done by output data to the FET driver board. However, the switch of the FET must be in right order, otherwise, the motor will not work.

Time delay is also required. The time delay is to provide the enough time to activate the rotator inside the step motor.

Function performed by microcontroller
4 I/O pins and software time delay.
Set 4 output pins and wired them with FET boards.

Actions to take after peripheral function configuration
While setting the output pins for the control of FET board, the Software Expert System will also generate the switching sequence of the FET. In addition, a fixed period of time delay will be applied in between the FET status change.
Time delay – set to 2 msec
Appendix C

Step motor control module

1. System diagram

![System diagram]

2. Task scheme

Enable required functions

- Enable one binary output port
- Set the first status of FET
  - Time delay
- Set the second status of FET
- Set the third status of FET
  - Time delay
- Set the fourth status of FET
  - Time delay
Appendix C

DC motor driving module

Introduction
Traditionally, the DC motor is controlled by changing the applied voltage to the motor by various of DAC and Linear amplifiers. Another method is PWM. Here the voltage is a constant level when switched on. The motor speed is controlled by switching the applied voltage on/off at a desired frequency and wave/space (on/off time) ratio. The focus of the module is to perform motor control through PWM with required interface board.

Objective
Apply PWM to drive a DC-motor. The motor speed is able to change due to the switching of ON/OFF percentage. Since a too fast PWM frequency is unable to drive the DC motor properly, it is often to generate a slower frequency by the timer cascade. Change the motor speed via am input board.

Function performed by microcontroller
Enable a DC-motor module. Select H-bridge to interface the DC-motor, no feedback component is required. Configure PWM function (select the advanced mode. Timer1 is the base timer, 75 percent, and PWM frequency is 800 Hz [while choose this option, an external timer is required-Timer0, and portB, bit0 is the choice for pulse]) Configure one input board (PortC) Configure another pin for direction control.

Actions to take after peripheral function configuration
Choose motor action. The Motor speed change is one possibility of motor actions. The change of motor speed can be done by load the data contents of input port for the new speed.
Appendix C

DC motor driving module

1. System diagram

![System diagram of DC motor driving module]

2. Task scheme

Enable required functions

- Enable one binary input port
- Enable one binary output pin
- Enable PWM function
- Read the input board
- Load into PWM

DC motor
Appendix D

The implementation of a Microcontroller Software Development Expert System

C.Y. Chen, M. Jackson, R. Parkin
Mechatronics Research Centre, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, United Kingdom.

ABSTRACT

The ability of microcontrollers in problem solving has been well known. By their very nature, microcontrollers can perform different functions according to software designs. Therefore, the software development has been highly considered in microcontroller-based system design. The software design process has been described as a very knowledge-intense activity. In order to deliver quality software design results, comprehensive knowledge is usually the fundamental basis. The performance of a microcontroller is the integration of hardware peripheral functions and software control strategies. However, all the proposed functions need to be coded in software eventually. Currently, the programming of microcontrollers is still dominated by assembly languages. Due to the hardware issues and the programming languages, the complexity of microcontroller software design is ever increasing. Modern technologies are thus aimed to apply onto the design process to improve software development. Due to the characteristics of software design, the applied technology is artificial intelligence, knowledge-based system (KBS). This expert system was proposed to help designers to deliver their design results for their target systems according to the configuration of the microcontroller. To effectively help designers during the software design phase, a graphic user interface was applied onto the system. Further detailed functions such as simulation were also included inside the expert system.

1. INTRODUCTION

Microcontrollers are used extensively these days in a wide range of applications, such as process control units, communication systems, digital instruments, electronics and home products. This is because of the benefits provided by the microcontrollers, such as low costs, programmability and readily availability of supporting peripherals [1, 2]. Accordingly, some products have been improved by the integration of microcontrollers, and some systems which
were not possible with traditional electrical components have been realised by the appearance of microcontrollers. To apply the embodied functions inside microcontrollers, it requires corresponding implementation in software. As more peripherals are now found available in microcontroller chips, the microcontrollers are now capable to implement in complicated systems. Accordingly, the current microcontroller software design process is getting more difficulty than ever [3].

The difficulty of microcontroller software development is due to several issues, since the development process is accumulated from several different stages. In different design stages, different knowledge is required to handle the corresponding design tasks. The required information may be located in different sources, and it takes time and effort to discover that. Afterwards, experiences or skills might be needed while using the found information [4]. Eventually, the implementation of the target microcontroller design needs to be coded into a program. Designers need some experience in order to express the design by coding language. Since the software development is usually surrounding the software development system, it was then aimed to build an advanced software development system to improve current situations.

In the remainder of this paper, the current software development systems and previous work devoted for system improvement will also be introduced. The difficulty of using current software development will be introduced in the next issue. Afterwards, the proposed solution will be described as well as the advantages that the system can offer to software designers.

2. EXISTING SOFTWARE DEVELOPMENT SYSTEM
The current microcontroller software development is usually carried out surrounding an integral development environment (IDE), which is provided by the microcontroller manufacturer. The IDE encompasses several tools such as editor, compiler and simulator, and these tools are for designers to build their intended software design [5]. Although the IDE allows designers to develop their intended software designs, it provides only basic functions. The designers have to input huge endeavours in order to turn design into software program due to the limited functionality of those tools offered by the IDE.

2.1 The endeavours to build software design by using existing development systems
To implement a microcontroller software design, designers need to infer the required functions from the proposed system. The inferred functions will be part of the design contents. Designers then have to handle the issues to set up the function with proper performance. Eventually, the function-related issues such as configuration and setting have to be transferred into coding. From software design to program, every stage needs some effort such as information and programming experience to break through. Therefore, the software implementation might be kept away from the designers due to their lack of effort.

In order to build intended software programs with the software development system, information and knowledge are acquired to accomplish the design tasks. Since information acquisition can be said as additional issues, it further increases the total workload of the software designers. In addition, software implementation requires coding, which is highly motivated by experience.
3. IMPROVEMENTS TO CURRENT DEVELOPMENT SYSTEM

Microcontroller-based systems are usually treated as the integration of software and hardware, since both parts are required. In order to achieve fast prototyping of such a system, a platform based on microprocessor architecture for software part and a configurable FPGAs based board for hardware part was developed in [6]. Similar idea was introduced in [7]. However, the focus of these systems was on the interfacing between the microcontroller and the external components, rather than software development. Another improved integrated development environment for microcontroller systems was introduced in [8]. This IDE was aimed at improving the drawbacks of current simulator and command-line interface by implementing an easy-to-use graphic interface, but this design did not help designers through the implementation process.

There are two more software development systems, which are keen on improving the software development process. They are DAvE and Mindstroms. Although these two systems were built for different purposes, they somehow offer benefits to software designers while using these systems to compose software design. These two systems do bring some useful features, such as fast software development, graphic user interface, and limited information inquiry. These features did improve the development process. However, it can also cause problems such as limited usage of the target microcontroller, or difficulties in function justification. In order to further improve software development process, a new software development system was proposed.

4. PROPOSED SOFTWARE DEVELOPMENT EXPERT SYSTEM

The purpose of the intended system is to fulfil the gap between the microcontroller software design and the implementation program. In addition, the intended system aims to simplify the software design process and offer good quality software program. Since software design and development process is described as a knowledge-intense activity, it is proposed to provide designers the required knowledge to solve the problems. The required knowledge will be embodied inside the software development system.

4.1. Available technologies

Applying former experiences and knowledge is concluded as the intended solution to reduce the complexity of software development. There are two current technologies available for applying former experiences in software development: one is software reuse, and the other is knowledge-based system. However, software-reuse focuses on producing new software programs by integrating reusable software components rather than processing the design demands. Therefore, software reuse might be helpful to the implementation part [9, 10], but software development is not really benefited. The final decision for the applied method is knowledge-based system, since it can actually support designers to develop their intended systems [11, 12].

4.2 Structure of the development system

The purpose of this system is to support designers through the microcontroller-based development process. Hence, the mighty problems faced by designers during the design process are targeted by this system. In order to provide proper knowledge to software designers, a simplified design method was also integrated into the software development system. This design method was implemented as the platform for the collected knowledge to be planted (Fig. 1). Therefore, the collected knowledge needs to be categorised since different
the implementation of control strategy is also covered inside this development system. A ready-to-compile program will be built after completing the software design, and it is the final result of the software design (fig. 2).

- Fig. 2 The software development process under expert system

5.2 Characteristics and benefits of the software development system

Some unique characteristics were shown while building software program through this software development system. Firstly, software development process is very straightforward. Designers can build the final software program from composing the intended system. Secondly, a convenient peripheral performance setting process is offered. The function setting is usually done through detailed configuration of related control mechanisms. This software development system offers designers options of possible performance instead of complicated control mechanisms. Thirdly, a ready-to-compile program is delivered after finishing the design. The result of software design will be developed after designers input the design issues, rather than coding.

Due to these characteristics performed by this software development system, some advantages were then provided. Since software development will start from system composition, the whole process is very compact. During the process, essential design issues for the intended systems will be guided by the software development system. Furthermore, related information will be offered while configuring these design issues. During the development process, no coding procedure is required. Most of all, designers are asked to described their intended systems, and the design of the system will be interpreted into assembly programs by the software development system. It is believed that designers can develop their intended software design within a shorter design cycle. In addition, quality result will be delivered after this efficient development process.

6. Discussion and conclusion
The complexity of microcontroller software development is getting higher due to the rich availability of peripheral functions and programmability. Currently, microcontroller software design is implemented by designers through coding. Since designers are demanded to input so much effort to achieve their software design, software design process is far away from efficiency. Since software design is recognised as a knowledge-intense activity, it was then proposed to build a software development expert system aiming at offering knowledge to support designers through the development process. In order to further improve software development, traditional data setting for function configuration was replaced by performance options. Eventually, a ready-to-compile program will be delivered as the software design result.

By simplifying software development process and offering designers required knowledge and information during the software development, this new software development system is more efficient than traditional development systems. The improvement can be found in system configuration, function setting, control strategy implementation and the save of coding work.

REFERENCES


Appendix E

An expert system approach to microcontroller software development

C. Y. Chen, M. R. Jackson, R. M. Parkin
Mechatronic research centre, Wolfson school of Mechanical and Manufacturing engineering, Loughborough University, UK.

Abstract
Microcontrollers are applied widely in products, since they can offer these products with improved performance and reduced cost. As the technology advance, microcontrollers are capable of performing peripheral functions which need to be initialised in software. Since the software controls the implementation of microcontrollers, the software development for microcontrollers is highly emphasised during the product development process. During the traditional software development process, designers can only rely on their own experience, and vendor provided information to accomplish the design tasks manually. Since the size and complexity of microcontroller software program is increasing, it means that a longer development cycle will be needed in order to produce the required result. Accordingly, the quality control of the software development result is difficult to achieve. It is thus aimed to improve current microcontroller software development. As software development is considered as a knowledge intense activity, a software development expert system which integrates expertises and knowledge was proposed for this purpose. In this paper, the design, implementation and performance of this expert system will be presented.

Keywords: microcontroller, software development, knowledge-based system (KBS, expert system)

1. Introduction
Microcontrollers are applied widely in modern products, and they can be found from a wide variety of products from industrial automation control systems to domestic products. The popularity of microcontrollers is continuously increasing, since it can offer benefits such as improved performance and low cost to products [1-4]. Due to the increasing number of microcontroller-based systems, the development of microcontroller-based system is getting more attention. To achieve the intended functions performed by target controllers, corresponding software programs are needed to be developed.

The main objectives of microcontroller software development are to deliver programs, which are capable of performing intended functions and last, but not least, to improve the quality of the development process. From the viewpoint of software implementation, experiences or information affects the process deeply, as designers' understanding affects the result of software development [5, 6]. In addition, the microcontroller software development is highly dominated by assembly languages, which increase further problems during implementation due to its characteristics such as direct data operation, and chip-dependent instructions [7]. By the appearance of knowledge-based system, possibilities exist to improve the implementation process and increase the quality of software development. This is because knowledge-based
systems are famous for offering appropriate solutions while facing difficult situations [8-10].

Therefore, the fundamental idea of this knowledge-based system is to find a way to improve the efficiency of microcontroller software development. To achieve this goal, related issues will be considered and components will be composed to build the knowledge-based system.

2 Software knowledge-based development system
Software design and development suffers problems from quality and quantity, and the amount of problems are due to the designer's experience and available information to achieve the complexity management [11]. Similar situation also appears in microcontroller software development. To avoid these problems, a knowledge-based (KB) development system is proposed, as the experience and knowledge needed during traditional development process will not be required from designers. Instead, these issues will be covered by KB development system. This KB development system delivers corresponding programs that are achieved without going through the manual coding process.

Microcontroller software development can be said as an integration of several procedures [12], and each procedure aims at specific purposes. Achieving these individual purposes is the basis to complete the proposed software development. Due to different purposes, different types of knowledge and information are thus needed. Therefore, a multiple layer knowledge based system is needed, as it aims at supporting the whole software development process.

2.1 Knowledge bases
The required functions performed by the target microcontroller inside a system are affected by the application tasks, and these functions are the purposes of software development. Reducing the distance between system composition and the required functions, knowledge and experience are often used to match each other. To accelerate this procedure, a knowledge base, named system configuration KB, aiming at finding out the required functions from intended system composition, is proposed.

After the required functions of target controllers are decided, the following development task is to turn them into codes. However, available peripheral functions embodied in microcontroller need corresponding control mechanisms to be set through software coding. In addition, the intended performance also needs to be achieved through the same method. Accordingly, the configurations of peripheral functions are proposed to be supported by another knowledge base, microcontroller profiling KB.

Microcontroller software development is not only about peripheral function configuration, but also control strategy implementation. Often the control strategies are proposed to produce responses according to events, and the required condition checking and following actions need to be organised in proper order to generate the intended control strategies. The third knowledge base, control strategy KB, targets system action composition to achieve the intended system control.
With the design of these three knowledge bases, the most often seen scenarios during microcontroller-based system design are supported. Information and expertises will be offered to software designers during the procedures of microcontroller software development.

Figure 1: The location of Knowledge based inside the KB system

2.2 System guiding process
Although individual knowledge bases might still be helpful to designers in different development procedures, each of them can only cover a part of the development process. In order to link these KB to perform optimal ability of these knowledge bases, a development guiding strategy, which comes from a classic software development strategy [13], is proposed to link these three knowledge bases. Accordingly, the development process will be supported under consistent knowledge resources.

3. Features of the KB development system
The main elements of the intended knowledge-based system were introduced in the previous section. In order to further improve software development, some more features are proposed onto this KBS. These features are mainly used to overcome the inconvenience caused by the traditional coding process. Later on, these proposed features will be presented.

3.1 Easy implementation environment
Programs are often recognised as the results of software development. The programs are produced by using traditional development environment, which is often a text-editing tool only. Designers have to be aware of all the issues including manipulating the development tool to compose programs. Thus, graphic interfaces are proposed to replace the existing coding environment.

3.2 Fast configuration process
In addition to the existing software development tool, detailed function settings, both in peripheral functions and system control function, are needed. Actually, most problems found during the software development are related to function setting. It is
then intended to compose a tool to allow designers to compose their intended designs without facing information overload during the development process.

3.3 Non-coding process
Producing intended programs is considered as the most important issue in software development. In the real world, designers have to not only compose the intended designs, but also implement them in software programs. Since the coding work is done manually, the function of automation code generation is proposed to be found in this KB development system.

4. Implementation of the proposed expert system
Intended features of the knowledge based development system were mentioned above, and it is thus needed to build a system, which can perform corresponding functions to fulfill these needs. Therefore, a few functions are proposed in order to meet these system needs.

4.1 Implementation of the knowledge bases
For the development of the backbone of the knowledge-based system, it is important to collect knowledge and expertises related to micro controller software development. Afterwards, the collected issues should be categorised into corresponding knowledge base. In between these knowledge bases, hidden routes for data passing exist. These routes are used to pass critical information decided in early development stages to later usage. To organise these knowledge bases inside the new development system, a software development methodology is applied as the guiding process. Accordingly, the prepared knowledge can be served as the supporting tools during different stages of software development.

4.2 Graphic interfaces
Existing microcontroller software development tool simply offer a coding environment for designers to build their designs. However, this environment causes designers some problems while coding. Therefore, it is intended to cover the new software development system with graphic interfaces. Using these graphic interfaces, the functions of knowledge offering and software development can both exist. In addition, the offered knowledge can directly be used to achieve current intended development purposes.

4.3 Function setting through performance options
Several different graphic interfaces are applied, and each of them targets different purposes. In each interface, all issues related to the function are listed as options for designers to decide. Designers are allowed to configure functions by setting the options directly. As the available peripheral functions are famous for their performance, the other set of options is therefore designed. This set of options allows designers to configure the intended performances to proposed functions. It is aimed at achieving fast function configuration through this type of implementation.

4.4 Automation code generation
The ultimate aim of software development is to build the corresponding programs, which are capable of performing the intended functions. Since the essential implementation details have been defined, the corresponding codes can then be produced from the knowledge base according to the information. Therefore, no
coding work is needed during the process. The produced programs are coded in assembly instructions, as current microcontroller software development is still mainly dominated by the assembler. The produced programs can then be tested after simple compilation.

5. Performance of this expert system
The design and implementation work of the knowledge-based software development system was presented previously. After finishing the development work, the evaluation of the performance of this knowledge-based system is aimed. In order to show the capability of this system, factors considered with software quality such as development time and produced results are recorded while a same system development was carried out in both traditional and this knowledge-based development system. The evaluation is achieved by the comparison of the recorded data in both development processes within different systems. In the following sections, the emphasised issues will be mentioned and comparison results will be presented.

5.1. Development cycle
While using existing microcontroller software development tool, designers have to identify the implementation aims, of the target controller and then carry out the required coding work. The work mentioned above needs to be done manually, and it is therefore time consuming. The situation is improved in this new development system, as the work is supported by knowledge, and no coding process is needed. From the comparison of over a range of peripheral function software writing challenges, longer development cycle is needed for using traditional system to achieve this purpose. The difference in development cycle can be seen in Figure 2. This is because much of the development time was used to read to understand what to do and correct the mistakes or errors made during the process while using traditional development system.

Figure 2: Required time to complete implementation
The implications of this comparison are: an effective software development was provided by this expert system, and little effort is required from designers to achieve the implementation purposes. These results came from the combination of knowledge, performance-oriented function implementation environment, straightforward development process, and easy-to-access interfaces offered by the expert system.

5.2. Information access
Information is needed during the software development process, no matter what the purpose is, e.g. ensuring the implementation or the correction of previous mistakes. During different stages of the development process, different information is usually needed. However, the current available information is often obtained from vendors, which is not edited to fit designers' needs. Therefore, designers have to go for several different parts in order to integrate a piece of intended information. Accordingly, the frequency of external information access is higher. Within the new system, the most often required information is offered, as well as error checking messages. Thus, little external information was required, and this is because designers aim to get more details related to their designs. The difference of information needed during implementation can be seen from Figure3.

Figure3: Information needed during implementation

Due to the knowledge offered by the exert system in different system development activities, the complexity of software development is reduced. Accordingly, the additional effort to handle information and code program is minimised. Therefore, designers can concentrate on the implementation of design contents rather than the implementation method.

5.3. Accuracy of the produced programs
Performances of the developed results are highly emphasised, and they needed to be tested to be ensured. As the produced program did not perform the intended functions at once, a debug process is often needed. However, the effort required for debugging can be as much as implementation itself. Therefore, extra time and effort are needed to achieve the debugging issues. The codes used to produce programs have been
tested, and thus the produced program can directly perform the proposed functions. As the results displayed in the case study, the produced programs can perform their intended functions without debugging. It can be said that the accuracy of the delivered programs is high. Figure 4 shows the performance of the developed program from different systems.

Figure 4: Performance of the produced programs

![Performance of the produced programs](image)

Software quality refers to several factors, and function correction is one of them. As the workload for debugging may require as much effort as the development process need, it is thus aimed to produced the function correct program after the implementation process.

6. Discussion and conclusion
From the comparison, the knowledge-based software development system offered more advantages than the traditional system. These advantages come from the characteristics of the system, such as the information and solution provided, the compact and consistent development process, the user-friendly interface, and the automation code generation. By the combination of these characteristics, a fast system prototyping can be achieved through easy implementation environment, and no coding work is required.

The new development system demonstrated that knowledge can effectively support microcontroller software development. In addition to the provided knowledge, graphic interface is feasible for design implementation. The automation code generation can further reduce problems encountered while coding. Accordingly, the microcontroller software development can effectively be improved due to these factors.

References


