



Birzeit University
FACULTY OF INFORMATION TECHNOLOGY
Scientific Computing Master Program

**USING C LANGUAGE ON SCIENTIFIC COMPUTING
APPLICATIONS**

By

Mohammad Mahmoud Abu Omar

Supervisors

Dr.Hassan Shibly

Dr.Samir Matar

August, 2007
Birzeit, Palestine

“This thesis was submitted in partial fulfillment of the requirements for the Master Degree in Scientific Computing.

From The Faculty of Information Technology at Birzeit University, Palestine.”

Committee Members:	Signature
1- Dr.Hassan.Shibly / Chair.
2- Dr.Samir.Matar / Member.
3- Dr.Mohammad.Saleh / Member.
4- Dr.Yousef.Abu.Zir/ Member.

ACKNOWLEDGMENTS

I wish to express my sincere thanks to my supervisors Dr. H.S. Shibly and Dr. S.A. Matar and for their kind assistance and support while preparation of this thesis.

Also I would like to thank Dr.Mohammad.Saleh and Dr.Yousef Abu Zir for their rich and expensive help and guides, and to all lecturers in the Master of Scientific Computing courses at B.Z.U.

My thanks are due to my parents, wife for their financial and moral support.

ABSTRACT

This thesis uses the C programming language as a fast gate between a wide range of scientific application systems and a computer network.

It includes some comparisons between C, C++ and Assembly in this domain.

It implements the Intel programmable peripheral interface PPI 8255A and the corporation between PPI 8255A and the computer's CPU and how to manage the system.

The thesis presents the design and implementation for a general system to be used in the large wide of scientific applications. This system is controlled by the C language. C is used because time is critical in this system and the need for programming the central processing unit directly. The thesis evaluates the system's work by showing application example that can be applied to general systems.

TABLE OF CONTENTS

Chapter One	1
1.1 Introduction	1
1.2 Contributions of this work	3
1.3 An overview of C	5
1.4 The C compilations	7
Chapter Two: C for Scientific Computing Applications	12
2.1 Choosing C	13
2.2 Where C is poor	15
2.3 Using C in mathematical and scientific applications	18
2.4 Are C and C++ similar for scientific applications	20
2.5 Using C on microprocessor and firmware scientific Applications	23
2.6 Using C for interfacing via processors and computer systems	25
2.6. Using C for Programmable input-output devices	26
1	
Chapter Three: Designing A System for Multiple Scientific Applications	28
3.1 General design procedure	29
3.2 Designing procedure analysis	31
3.2. Interfacing	31
1	
Interface devices	33
3.2.	
2	
3.2.3 Device drivers	34
3.2.4 The main design control	37
Chapter Four: System Design Implementation	39

4.1	Introduction to the Intel programmable peripheral interface (PPI)	40
4.2	The Intel programmable peripheral interface (PPI) work	44
4.3	The operating modes of the PPI 8255A	50
4.4	Using the suitable mode operation for the system design	51
4.5	The corporation between PPI(8255A) and computer's CPU	52
4.6	How does the device driver manage the system	53
	Chapter Five: Scientific Computing Application Example	56
5.1	Application example idea	57
5.2	Applying the application example by the system design	59
5.2.	Stepper motors through application example	61
1		
5.2.	Power supply	64
2		
5.2.	Infrared sensor	65
3		
5.3	The expected model design for the application example	67
5.4	Analyzing the system management for parallel interfacing of	69
5.5	Controlling the stepper motor movement	73
5.6	Analysis of the software application example	75
	Conclusion	80
	Future Work	82
	References	83
	Appendix A	90
	Appendix B	95

LIST OF FIGURES

Figure	Page	Title
1.1	9	Compilation process.
3.1	29	General design procedure for parallel interface application.
4.1	45	Block diagram of The PPI 8255A.
4.2	45	The PPI 8255A interface.
4.3	48	Control register and their control functions.
5.1	60	Application example system design.
5.2	62	Excitation of stepper motor coils.
5.3	67	Controlling the movement by infrared sensors.
5.4	68	Application example model design.
5.5	69	Control register loaded by 90H.
5.6	71	Connecting the power supply and infrared sensors with Port A.
5.7	76	Analysis of power supply controlling.
5.8	77	Canceling the work of motor 2.
5.9	78	Canceling the work of motor 1.

LIST OF TABLES

Table	page	Title
5.1	64	Wire voltage of power supply.

Chapter One

1.1 Introduction

The C programming language can be used in a wide range of scientific computing applications. It is not easy to decide which applications will C suitable for. There are many important factors that must be taken into consideration, such as: time, speed, size, performance, development, maintenance, cost and economy. Some applications depend on one or more of these factors as primary requirements. The C language can satisfy some requirements for certain applications. The main question to ask if the C can satisfy one or more of these requirements on some applications, can this lead us for saying that C can satisfy these requirements in all other applications?

This thesis clarifies the applications that C is suitable and must be the first choice to use with, and also applications which C is poor with.

C has simple and small compilers which make it is able to execute programs with more speed and less time. C can work with digital and analog environments, it is able to portable between them, the versatility of C allows it to be run on the full range of processors, from 8-bit

microcontrollers up to the Cray series of super computers, according to these characteristics, the thesis shows that C is the best and the first choice to use in the processors and computer- based applications.

This thesis applies these C characteristics through designing a parallel interface system which thesis presents; it is designed and implemented not only for specific applications but also for multiple applications with a good level of efficiency.

The role of C through processors and computer-based applications and the importance of C speed in this type of applications are evaluated by showing an application example which is applied on the system design, and the results are analyzed throw the application example.

1.2Contributions of this work

This thesis has the following contributions to the scientific research:

- 1- C language must be the first choice to be used in the following scientific applications:
 - a- Applications concerned with speed, size and performance.
 - b- Applications that use compiler writers and interface device drivers.
 - c- General machine applications.
- 2- C language is a poor tool and must be avoided in the following scientific applications:
 - a- Applications concerned with development.
 - b- Applications that have a large-scale software.
 - c- Applications with portable software program.
 - d- Applications which depends on memory management.
- 3- Using the C language as the main tool or a tool for final production in the mathematical applications is not recommended and must be avoided.
- 4- Using C language is extremely different from C++ in the scientific applications. Object oriented features aren't enough to make

C++ better than C in general.

5. The theory for converting a system design of interface applications to a general system at a good level of efficiency, can be achieved by using:
- a- Parallel interfacing.
 - b- The programmable peripheral interface PPI(8255A) as interface device.
 - c- Programming language C as a device driver and a system software.
 - d- Computer central processing unit (CPU) and a network as main control of the system.

1.3An Overview of C

Dennis Ritchie initially [2] developed the C language at the AT & T Bell laboratories in 1972. His objective was to use a high/low-level language to implement the UNIX operating system in a way that was easy to maintain yet powerful and portable. To achieve this, he designed a language that had many effects. The most important was the language BCPL. The influence of this on C continued indirectly through language B. Therefore originally, the main usage of C was in system programming. Compilers, operating systems and utilities have all been written in C. It excels in these areas and is now applied to many other areas of difference types of computer systems. After its adoption, the C language, unlike other many languages, tended not to split into different dialects. By 1983, it had matured to the point where it required standardization. Most C products now comply with this standard and not with the original implementation [2].

Although it was originally intended to run under UNIX, there has been a great interest in running it under the MS-DOS operating system and specifically on the IBM personal computers and compatibles. It is an

excellent language for this environment because of the simplicity of expression, size, the compactness of the code, and the wide range of applicability.

It allows the programmer a wide range of operations from high level down to a very low level approaching the level of Assembly language.

A program written in C will have an increase in runtime and it is simple to write and easy to maintain more than the Assembly language [1].

1.5 The C compilations

The high-level language program in C is translated into a machine code that can be directly executed on the hardware. For doing this effectively, the compiler must analyze the source program as a larger unit, usually, the entire source file, before producing the translation.

A program needs only to be compiled once and if it has no errors, it can be executed many times. Many programming languages including C, C++, and FORTRAN, are typically compiled. The LCS-2 Assembler is an example of an elementary compiler. In contrast to an interpreter, the compiler processes the high-level language program and converts it into a machine code. We provide the compiler with a file, or possibly manipulate files, containing the program and it creates a new file containing a machine language version of the code. The compiler doesn't execute the program but rather only transforms it from a high-level language into the computer's native machine language [1].

The executable code is a machine language representation of a program that is ready to be loaded into memory and executed. The entire compilation process involves several components, only one of which is the compiler, because when using the C compiler, the processor and the

linker are often automatically invoked. Figure 1.1 shows how the compilation debugging, linking and executing processes are handled by these components. In this figure it shows the overall compilation process, the preprocessor, compiler, the linker. The inputs are C source and header files and library object files, the output is an executable image.

A C compiler often has additional capabilities that provide a user-friendly environment for implementing and testing C programs.

The C preprocessor scans through the source files (which contain the actual C program) looking for and acting upon C preprocessor directives.

The directives instruct the preprocessor for transforming the C source file in some controlled manner, for example, by substituting the character string `DAYS_THIS_MONTH` with the string `30` or by inserting the contents of file `stdio.h` into the source file at the current line.

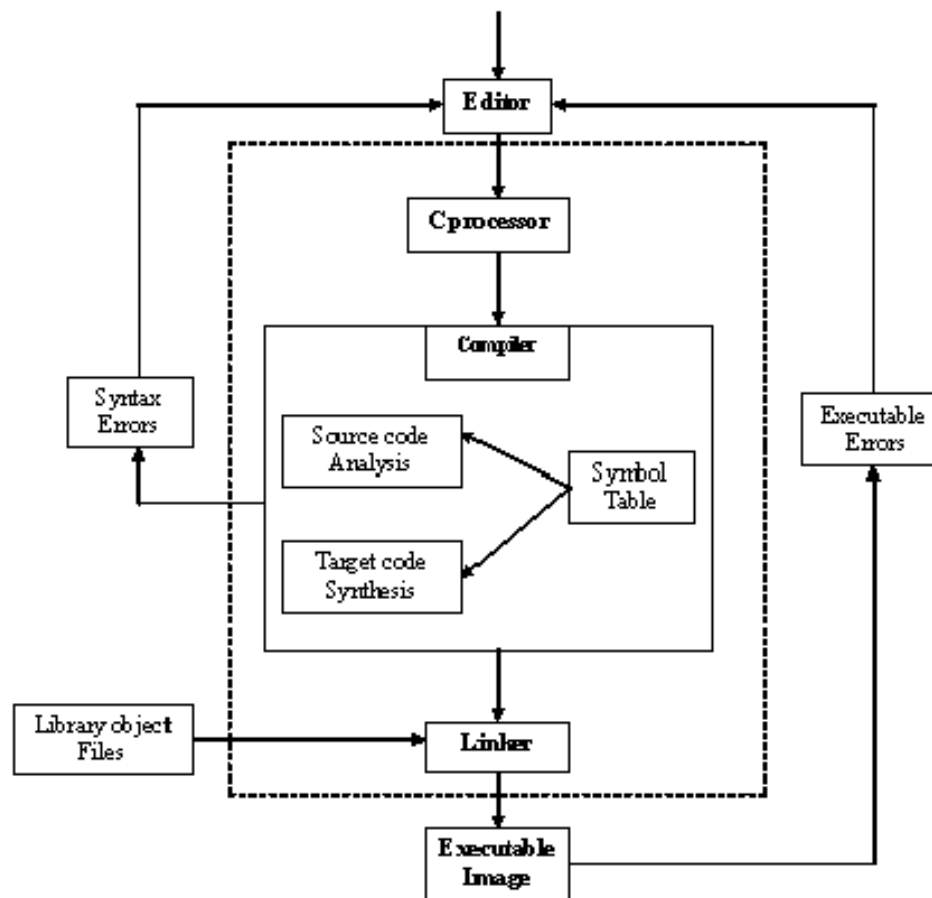


Figure 1.1: Compilation processes

All preprocessor directives (header files) have a hash sign # as the first character. All useful C programs rely on the preprocessor .After the preprocessor transforms the input source file, the program is ready to be handed over to the actual compiler. The compiler transforms the preprocessor program into object module, which is a machine code for one section of the program.

Compilation has two major phases:

- a- Analysis: where the source program is broken down into its constituent parts.
- b- Synthesis: where a machine code version of the program is generated.

Often, the two portions of compiler corresponding to these two phases are called the compiler's front end and the compiler's back end.

It is the job of the front end to read in, parse, and build an internal representation of original program.

The back end generates the machine code, if directed, attempts to optimize this code to run more quickly and efficiently on the particular computer for which it is generated. Each of these two phases is typically

divided into sub phases where a specific task such as parsing, register allocation, or instruction scheduling, is accomplished. Some compilers generate assembly code and use an assembler to complete the translation to machine code.

The linker takes over after the compiler has translated the source file into object code. It is the linker's job to link together all object modules and a library to form an executable image of the program.

The output of the linker is an executable image. The executable image is a version of the program that can be loaded into the memory and executed by the underlying hardware. When click on the icon for the web browser on your computer you are constructing the operating system to read the web browser executable image from your hard drive and load it into memory and start executing it [1], [3].

Chapter Two

C for Scientific Computing Applications

Scientific computing has a wide range of applications, and also uses a large several computing languages, software packages and tools. It is not easy to decide the most suitable language/tool for a specific application. C language is one of these computing languages that acts as a powerful tool on some applications, and a poor one on others.

According to the powerful properties and characteristics for the C language, this thesis discusses guides and analyses some scientific computing applications and solutions by applying the C language as powerful tool.

2.1 Choosing C

The thesis gives a specific and clear way for choosing C in scientific applications. C can be effectively used to achieve more performance in some systems and applications.

Speed, size and performance are three important elements in some scientific applications. Therefore C language will be the good choice for such applications. In this kind of applications, the C compiler is very mature and fit through experience [30], it is not complex compared to others, and also C programs will run a bit faster.

Also, C is the most suitable language for all applications that use the interface device drivers, digital-analog and for compiler writers [29]; this is for two reasons:

- 1- These scientific applications related to (time-critical applications) [28].
- 2- These applications need a tool for programming the central processing unit (CPU) directly [32].

Above all, C is rich for general machine applications; since it is a low level language with some high level features. In these applications C can

develop the following:

- 1- Increasing the implementation [28].
- 2- Debugging the time [28].

The C is a good implementation and efficient language in all applications which is required to replace the assembly language as well as greater readability and writ ability [15].

C is a fundamental language, especially if we look at the other important fields that C acts as an expensive tool for them:

- 1- Writing and building an entire operating systems, which is impossible to do by scripting languages, and more difficult in Java [27], [31].
- 2- Building embedded systems, libraries and components [27], [31].
- 3- Acting a considerable role in Computer graphics, for writing fast computer games [20], [27].
- 4- C is the most popular language in science and engineering graduates and amongst professional programmers.

We have improved the speed of C, by making a comparison between two simple similar programs, the first is written by C, and the second by JAVA, the output of the two programs is printing "welcome" word. The comparison is done by using (Windows XP Performance Test); it is a Benchmark which shows the CPU Usage Time.

To see this comparison experiment and the results, see **Appendix A**.

2.2 Where C is poor?

Although, C is creative for some applications, it is poor for others.

To be at a good level of performance through applications, we mustn't use C on the following:

Firstly, applications that primary concern with development.

These applications need rapid development, which C can't achieve; development time is much longer in C; since bugs creep in more easily [29]. Hence cost is also higher for C application programs.

Java language with these applications will be a correct choice, especially the primary concern here is for development not for speed.

Secondly, applications which have a large-scale software.

C hasn't a direct support for modularity, its naming structure provides only two main levels: external (visible every where) and internal (within a single procedure) [15].

C doesn't support the Object Oriented (OO), so C isn't suitable language with large scale software applications, since the workers in this type of applications must write a large relations and use the internal relations instead of classes which C doesn't support, which is complex and hard [15].

Thirdly, applications with portable software programs.

In the developing of C, the portability wasn't an explicit goal in its design since it is actually a typed assembler aiming at machine level [22], [15]. In the underlying machine model assumed by the processors of C made us well aware that not all machine were the same. Many C types and operations are closely tied with the machines and make it difficult to port the software to other type of machine [15],[45].

Through these applications, software will move to new computing environments, so we see that portability in these applications is not

important but also much necessary. The benefits of portability are concluded on the following:

- 1- When software needs to move to new computing environments, there is much less effort than it would to rewrite them [22].
- 2- This leads to decrease the software maintenance and development [22].
- 3- And also the economic advantages for decreasing the cost.

C isn't a suitable in all applications which depends on memory management, because there isn't a good mechanism for garbage collection, and C doesn't support automatic garbage collection.

The objects will stay in the memory, and make it full, so the worker must delete every object when it is finished . [15], [18], [47].

Java language is suitable in this field because it has automatic garbage collection, There is a continuous test to delete every object in the memory when it is finished.

2.3 Using C on mathematical and scientific applications

There is a great tend to use the C language on mathematical and scientific applications; especially that C codes can manipulate mathematical operations, matrices, and vectors, by C structures functions and methods can manipulate.

This thesis main concern is to use the C language in some scientific computing where speed, performance, effort and cost are major concern. In Matlab, which is the rich mathematical and scientific package, the codes take a relatively long time to execute [19].

So, if more time for executing some applications in Matlab more than C leads to make C is more suitable than Matlab on these scientific applications? Also, Matlab was built on C language.

If you look at some expensive mathematical packages such as Matlab, Mathematica, Maple....etc. You will find the following:

- 1- Easy to use for mathematically.
- 2- Rapid prototyping.
- 3- Rich specialist libraries.
- 4- Include different features for numerical computing [25].

Two answers for choosing a tool of mathematical applications:

The first is not to use C, and instead, use the expensive mathematical packages such as Matlab, Mathematica, Maple....etc. Even if C has more execution speed.

The second is to combine between C and one of the expensive mathematical packages. On other hand, we can combine the creative points from each of them, for obtaining to the most performance. This can be done by using the expensive mathematical package for rapid prototyping and others then generate C codes for final production [25].

But this way may be correct in theory, but it is fail in practice, since development can be accomplished in different multiple environments.

Any change to machine generated C code has to be propagated from the code written in Matlab, Mathematica... [25], other problems will appear such as:

- 1- Increasing the cost for purchasing multiple software packages and compilers [25].
- 2- Developing and maintaining of codes will be difficult and also costly.

2.4 Are C and C++ similar for scientific applications?

C++ is an object oriented programming (OOP). This object oriented defines in addition to data type of a data structure the types of operations which can be applied to the data structure, so data structure will be an object that includes both data and data functions; further more, objects can inherit characteristics from other objects by creating relationships between objects.

No one can ignore the advantages and benefits for OO features in many applications, but this this will not lead for preferring using C++ to C at all. The C++ compiler is a lot more complex than a C compiler [30], so C++ programs may run a bit slower than C program.

In applications that primary concern with OO features, C++ is the suitable choice, but in other applications, OO features may not be as important as other factors such as time which is related to speed.

Therefore on some applications C is the suitable, not the C++ when the primary concern is time/speed more than OO features.

For example: formatting numerical output to line up in columns, which is very desirable in numerical computing, is slower and has less performance in C++ than C [26].

Some OO features such as a (polymorphism) increase the problem of speed and need for much time. In C there is a direct function call, this means that when calling a function, the compiler and linker will determine the address of the function to be called [17]. In C++, there is indirect function call. This means when calling a virtual function, linker will not know the address, it is bound at run time and stored in a virtual table, every object is linked with the virtual table, so calling a virtual function means getting the address of virtual table, then using an offset to access the called function's virtual table entry and calling that functions [17], [26].

Thus, the C++ indirect calling causes the following:

- 1- Increasing the execution time.
- 2- Needing to more bytes than needed in the C direct calling [17].

So, the thesis insists that we can't say: C for all or C++ for all, and we can't say that OO features always make C++ the first choice.

For obtaining the most performance scientific applications, we must consider that applications with primary concern to the OO and not critical time applications, C++ is the tool not C. At the same time applications with primary concern and critical to the time, more than OO features, the tool should be C not C++.

This work insists that in scientific computing, one should firstly look at the type of application before finding a tool that is more effective with this application, and shouldn't firstly search for the most powerful tool, since a powerful tool will be a poor tool with other applications.

2.5 Using C on microprocessor and firmware scientific

Applications

Microprocessors, digital signal processing devices (DSPs) and analog devices are programmed in the same languages as other scientific and engineering applications, usually there are two choices: C or Assembly language.

Applications that use assembly can execute their programs faster than C and also with less manufacturing cost, the C code usually requires a larger memory than assembly, resulting in more expensive hardware. But applications that use C are easier to develop and maintain. And C is much more efficient when there is a large, general purpose register set and a unified memory space. These future improvements will minimize the difference in execution time between C and assembly, and allow C to be used in more applications [38].

There are still many valid reasons to choose C as a programming language in this type of applications:

1- If there is a substantial amount of code written in C in the application, we obviously want to make full use of the analog devices' C compiler. If

we are concerned with the portability of the code between different platforms, we will want to consider writing the majority of your code in C. Analog devices' assembly language will only work on an analog devices' DSP and there are no efficient conversion programs available to translate code between different manufacturer's DSPs [36].

2- Using C will be faster in the applications that don't spend many hours optimizing the code to fit the memory and throughput requirements of the application [36].

3- There is also a way for getting to the best case: writing the program in C, and using assembly for the critical sections that must execute quickly. This is one reason that C is so popular in science and engineering. It operates as a high-level language, but also allows to directly manipulating the hardware [38].

2.6 Using C for interfacing via processors and computer systems

In interfacing via processors and computer systems the C language is a creative tool, this is because of C characteristics which are very suitable in this type of applications.

The C language can be run on the full range of processors from 8-bit microcontrollers up to Cray series of supercomputers, and it can be worked in analog and digital environments and it can be portable between them.

The C includes a rich set of operators to configure input-output devices and also flag testing. C can easily interface with I/O cards, It is able to program the CPU directly and manipulates the data through registers this will make the interfacing with the outside world in electronics and electrical projects with minimum of hardware.

So, using C is very close to the target machine, and leads to reach to the best design possible in this type of applications [39].

2.6.1 Using C for Programmable input-output devices

The parallel interfacing considered as the simplest method to communicate between the real world by computer systems.

This interfacing depends on the parallel ports of I/O cards and circuits, which include the integrated circuits and processors (integrated chips), such as Programmable Peripheral Interface (PPI) 8255A .

Although these chips are complex, but they are very easy to program, they include ports, registers and there are a data bus buffers.

The C can program the CPU directly, and the CPU can see all ports and addresses, so by C the CPU can write to and read from the ports.

The C is low- level enough to write the device drivers, this is because of C compilers which contain the library functions which make C able to manipulate the I/O port-mapped. For example the Microsoft C compiler version 5.1 uses *inp()* and *outp()* functions which are defined in the header file *conio.h*. The Borland Turbo C uses *inportb()* and *outportb()*.

The calling of these input- output functions in C is very easy , which is achieved by *#include<conio.h>* .

For example C can implement the Programmable Peripheral Interface PPI 8255A, in different applications, the input – output functions of C, can manipulate all ports and also the control register of the PPI [39].

On the next chapters of this thesis we can see the ability of C in designing the interfacing applications with computer systems, and how the C can manage these systems with high level of performance and efficiency.

Chapter Three

Designing a System for Multiple Scientific Applications

There are many different fields and systems in scientific computing applications. A system in our work has to analyze, design and manipulates some applications that are provided by the integration between software and hardware. Any applicable idea, related to this type of applications, can be achieved by this system with a good level of performance and development. Therefore our system design is to be considered as a general system. It can be applied not only for one specific application, but also for a multiple scientific applications. In this system design we want to see the effect and the power of using C as a tool through these types of systems.

3.1 General design procedure

The interface device driver applications in any system design may be considered as the following procedure:

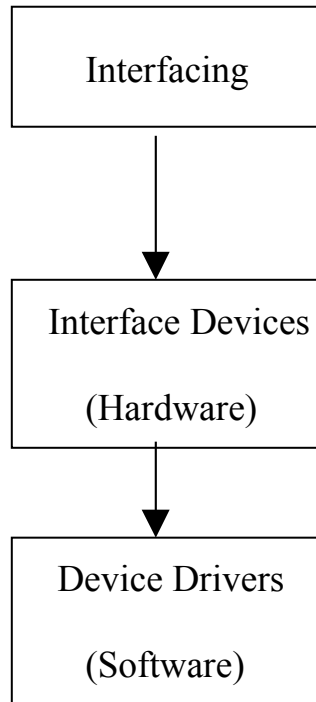


Figure 3.1: General design procedure for parallel interface applications

The need here is a system design and we want to reach to a general system for multiple applications

Our system design is achieved by the following ways:

Firstly:

In this procedure, there are general stages such as:

- 1- Interfacing.
- 2- Interface devices.
- 3- Device drivers.

And there are two types of interfacing and different large number of the interface devices, and also big choices for software tools/languages the correct choices for these stages are very important for achieving our design.

Secondly:

The control source/ type in this procedure is general, it is may be from integrated circuits, controllers, and computers.....etc. The suitable choice to the control source/ type is another factor to achieve our system design.

3.2 Design Procedure Analysis

According to the design procedure summary in previous section, the analysis for design procedure will be in the following:

- 1- Interfacing.
- 2- Interface devices.
- 3- Device drivers.
- 4- The main design control.

Now we explain each of them:

3.2.1 Interfacing:

The interfacing is a term for computer programs that accepts input from the user while they are running; for example, a game that waits for the user to take an action, then responds to that action. The interaction between computer and user may take place through typed commands, voice commands, mouse clicks, or other means of interfacing. The opposite of interactive processing is batch processing, where all the commands are given before the program starts to run.

There are two types of interfacing: parallel and serial. Parallel interfacing has a great advantage for using it in the design. This is because of the speed; all eight bits of an 8-bit data, one byte can be transferred simultaneously, but if we look at the serial interfacing, we find only 1 bit is transmitted at a time. So parallel interface provides a multi line data channel in which bits are sent across multiple lines simultaneously. The bits must stay in synchronization as they cross the wires, so the parallel interfaces are limited in distance. Parallel interfaces are usually associated with printer connections, but several technologies implement parallel interfaces, including:

- **HIPPI (High-Performance Parallel Interface)** HIPPI is a high-performance parallel interface that is used in data centers and supercomputer applications.
- **SCSI (Small Computer System Interface)** SCSI is a parallel interface for disk storage devices that is characterized by 50-pin or 68-pin connectors.
- **OFDM (Orthogonal Frequency Division Multiplexing)** OFDM is a multi carrier modulation (MCM) scheme in which many

parallel data streams are transmitted at the same time over a channel with each transmitting only a small part of the total data rate.

- **Computer's bus:** The computer bus is the peripheral interface bus inside desktop computers and servers. Which we used it in our design.

3.2.2 Interface devices:

We must find a parallel interface device for obtaining our general system; especially there are many different types of parallel interface devices from different companies such as: Intel and Motorola...

Our research work sees that Intel programmable peripheral interface (PPI) 8255A, can satisfy our general system. The PPI provides a parallel port to which external devices can interface easily. It is a bi-directional port that can receive or transmit up to 16 bits of data [35]. The PPI (8255A) is an important factor in the design, this is due to its characteristics, which make it the only parallel device at present that can perform the generality in the system with a considerable degree of performance.

These characteristics are:

1. Digital input and output interfacing systems.
2. Flexible enough to interface almost any input and output device without the need for additional external logic, and can permit easy implementation of parallel input and output in the processors.
3. Represents the optimum use of available pins.
4. Includes important features such as: single-bit, 4-bit, and byte-wide input and output ports; level sensitive inputs; latched outputs; strobed inputs or outputs; and strobed bidirectional input/outputs.

3.2.3 Device drivers:

A device driver is a tool/software used to control the interface device, and also control the system.

The correct choice for the system software (tool/language) makes a system in a good level of performance and speed, which are very important for our general system especially, if we have a general system with less performance and speed, this generality doesn't mean

anything, in this case it is more suitable to design a system for every application with a high level of performance and speed than design a general system for multiple applications with low level of performance and speed.

So, what is the suitable language in our system?

Our system acts as device driver system. The tool of this type of applications is discussed on chapter 2, and we find that C language acts as a powerful tool on these applications. Further more, Intel PPI (8255A) effectively works with C system software, this is because of the compatibility between C and Intel processors. If application software is implemented entirely in C code, the software can be easily ported for a processor based on Intel processors [33]. This is one of reasons in our research for choosing C through our system design.

The C programming language has found wide acceptance as the primary programming language for embedded microprocessor software and firmware development. The demonstrated utility of C in embedded applications development can be applied to microprocessors applications as well. The arguments for this are attracting.

1- Many engineers and scientists and most newly graduated engineers and scientists know C already, so the development of applications can start quickly, without the burden of learning the characteristics of a new assembly language and instruction set of the particular microprocessor chip the engineer may be using.

2- C allows for applications to be developed, prototyped, analyzed and tested in a workstation or PC environment using commercially available C compilers and tools. This prototyped code can be used, sometimes unaltered, as the basis for the application software running on the microprocessors target.

3- C is a well defined. Therefore, C code can become portable to any microprocessor chip for which there exists C language support. The stability of C in the future is insured with the adoption of the ANSI standard for C.

4- The majority of the software developed for microprocessors applications is really control code requiring sophisticated data structures and complicated control flows. This control code typically represents a small fraction of application execution time, but a significant portion of

the software engineering effort. Control code is ideally suited for development in C. Also, C allows easy access to the underlying hardware. Code that needs to access hardware control and status registers and input and output (I/O) ports can be written in C. Using C is easier and will decrease the amount of time that is needed to develop an application [36].

So, our system tool and device driver will be the C language.

3.2.4 The main design control:

We use a central processing unit (CPU) of a computer for major system control, to achieve our system design. CPU can control and configure the interface device, and by reading from the device registers. CPU knows what happens on the interface device [13].

The CPU and interface device can exchange data by reading from or writing to the interface device data register.

In addition, using a computer for controlling systems will open the choices for scientific computing designers and workers to update a system software in the future. Since on computer you can run any tool

that you need, this can't be achieved when we use other sources of control such as the controllers, which work under a specific software. In our system design, using C as the system software and Computer's CPU, will give more flexibility and performance for the system; since that during our scientific research and analysis, we deduce that there is compatibility between them [37], [44].

This design procedure builds and creates our general system, for using it in multiple scientific applications; it can be used for any inputs and any outputs, with a high level of performance and speed, just we only determine what are the inputs and outputs of an application, and then we can apply it in our system for executing.

Chapter Four

System Design Implementation

In this chapter the system design implementation in both hardware and software will be discussed. And how the system design can be achieved and developed through implementation.

4.1 Introduction to the Intel Programmable Peripheral

Interface (PPI)

Computers are not just devices to calculate and perform mathematical operations. Computers should interact with the users and the programmers as well. For example the computer should accept input from the users through files, keyboard, mouse... On the other side the computer should show outputs either by the screen or by print out using the printers or other devices. The main problem in making the computers interact with such devices is the timing. To illustrate, the processor is much faster than these devices and should wait for them to perform those actions. There are two options: first, which is very inefficient and impossible, is to make the processor very slow to wait for the devices. Second way that is used now is to make the outer devices have some kind of conversation with the processor. That kind of conversation is called handshake as it makes the processor sends and receives signals from the devices and to the devices simultaneously. The processor starts sending a pulse to the device telling it that it will send some characters to the

device and the device should respond and tell its status whether it is busy or off line or it is ready to receive the characters.

***Handshaking**

There are two kinds of handshake operations the first is called simple handshake and does not need much effort to perform as it just needs one line of data to tell the processor that it is ready to and finished the previous order. That kind of simple handshake is used in the keyboard when we press a key. The second kind of handshake is the double handshake and it uses the same methodology to in performing the operation except that it needs to lines and two kinds of input signals; one from the processor and the second from the device itself. That kind is more complicated and is used in the operations of the printers and in some industrial applications.

*** Single handshake**

The problem of having the sent data faster than the ability of the receiving device led to make us use the handshake technology. That is clear in the times when we need more sensitivity of having and sending

data at specific time. In addition to the simple strobe way single strobe is used instead. In that method the peripheral gets the data whether input or output and send a strobe (STB) to the processor to inform it that it has valid data. Then the processor starts reading them and send an acknowledge line(ACK) or signal that it read the data and it is ready to receive the next line of data. The cycle goes on until they finish all the data. That operation can happen in both the input and the output operations. For example the printer can use that method to inform the processor that it has finished printing the character and ready to have the next character to be printed.

*** Double handshake**

When operations need to be more accurate in dealing with the time it is better to use double handshake. The same methodology used in single handshake can be used in the double handshake and with the same waveforms.

*** Wave forms and their relations with the handshake**

To illustrate the relationship between the wave of the timing diagrams and the handshake operations we can take the simple process of pressing a key on the keyboard as an example. The press of the key causes an interrupt to the processor to read the ASCII code coming from it which is putted on the parallel data lines of the peripheral input ports. After putting the data on the data lines the peripheral then sends an strobe STB to inform the processor that it has valid data on it and then the processor starts reading those data characters. That is a very simple process of the strobe and the communication between the processor and the peripheral and the timing waveforms. The STB or the ACK signals for the handshake transferees can be produced on a port pin by instructions in the program. That method is taking much time from the processor. So parallel port device is used. The 8255A is designed to automatically managing the handshake operation. The internal address for the device is port A 00; port B 01; port C 10; control 11. Asserting CS(chip select) input of the 8255 will enable it for reading or writing. The CS will be connected to the output of the decoder circuitry to select the device when

it addressed. The reset input of the 8255A is connected to the system reset input. So they are going to be reset in the same time to be initialized for input. When connecting the 8255A to 8086 processor for example. One of the 8255A is connected to the upper half of data bus of the 8086 system. Another 8255A are connected to the lower half of the 8086 bus system. So that we can enable one device or a word can be enabled by the same time according to the truth table for the I/O port. The base address of the first half of the data bus is going to be addressed by the base address of FFF8h and the second is of base address of FFF9h.

4.2 The Intel Programmable Peripheral Interface (PPI) work

The interface parallel device Intel programmable peripheral interface/PPI (8255A) was used. Its work during the system and through the computer system will be discussed.

Firstly, we show the block diagram of this PPI 8255A, see figure 4.1 and its pin layout in figure 4.2:

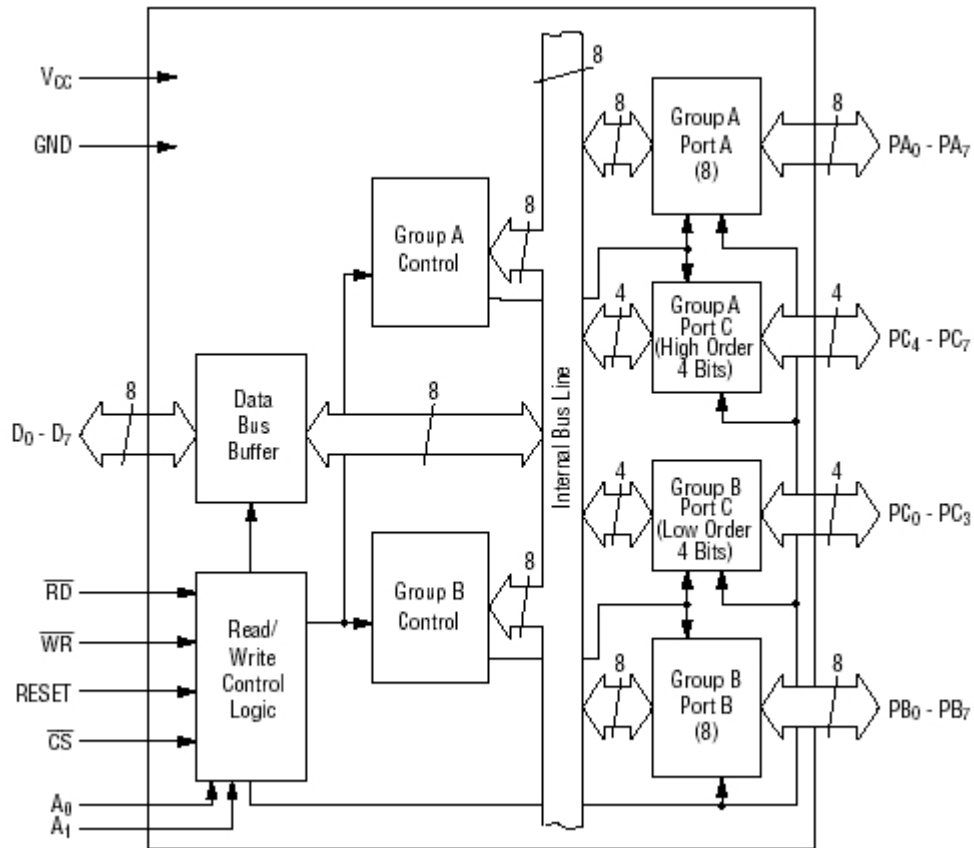


Figure 4.1 : Block diagram of The PPI 8255A [39]

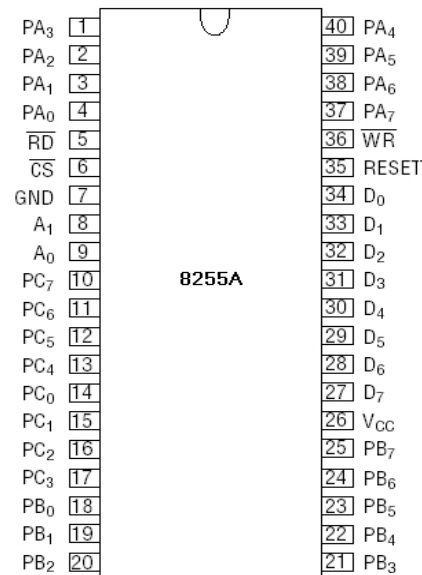


Figure 4.2: The PPI 8255A Pins Layout [36]

The left side of the block represents the microprocessor interface. It includes an 8-bit bidirectional data bus D0 through D7.

Over these lines, commands, status information, and data are transferred between microprocessor interface and 8255A.

These data are transferred whenever the microprocessor performs an input or output bus cycle to an address of a register within the device.

Timing of the data transfers to the PPI 8255A is controlled by the read/write (RD and WR) control signal. The source or destination register within the 8255A is selected by a 2-bit register selected code.

The source or destination register within the 8255A is selected by a 2-bit register selected code. The microprocessor must reply this code to the register select inputs, A0 and A1 of the 8255A.

The port A, port B and port C registers correspond to codes A1A0= 00, A1A0= 01, and A1A0 = 10, respectively.

Two other signals are shown on the microprocessor interface side of the block diagram; they are the reset (RESET) and chip select

(CS) inputs, CS must be logic 0 during all read or write operations to the 8255A, it enables the microprocessor interface circuitry for an input or output operation.

On the other hand, RESET is used to initialize the device. Switching it to logic 0 at power up causes the internal registers of the 8255A to be clear; Initialization configures all input and output ports for input mode of operation. The other side of the block corresponds to three byte- wide input and output ports. They are called **port A, port B, and port C** and represent Input and output lines PA0 through PA7, PB0 through PB7, and PC0 through PC7, respectively. These ports can be configured for input or output operation. This gives a total 24 input and output lines. As it mentioned the operating characteristics of the 8255A can be configured under software control. It contains an 8-bit internal control register for this purpose. This register is represented by the group A and group B control blocks in figure 4.1, logic 0 or 1 can be written to the bit positions in this register to configure the individual ports for input or output operation and to enable one of its three modes of operation. The control register is to write only and its contents are modified under

software control by initialing a write bus cycle to the 8255A with register select code A1A0 = 11. The bits of the control register and their control functions are shown in figure 4.3 bits D0 through D2 correspond to the group B control block in the diagram of figure 4.1.

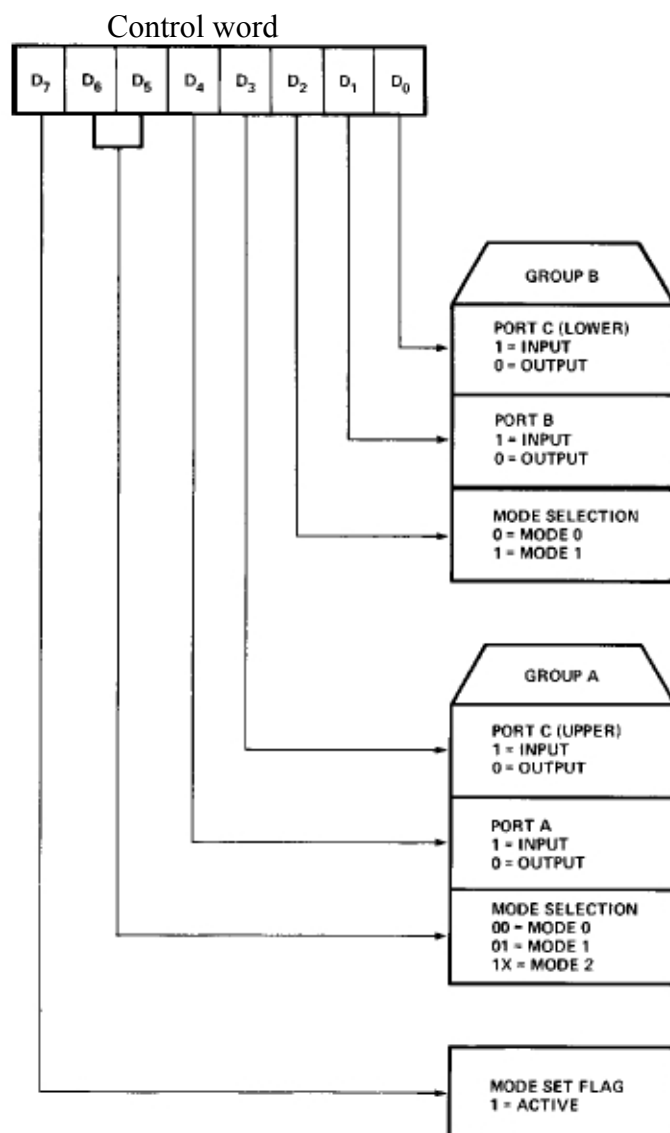


Figure 4.3: Control register and their control functions [39]

Bit D0 configures the lower four lines of port C for input or output operation. Notice that logic 1 at D0 selects input operation and logic 0 selects output operation. The next bit D1 configures port B as an 8-bit wide input or output port. Again, logic 1 selects input operation and logic 0 selects output operation. The D2 bit is the mode select bit for port B and the lower four bits of port C, it permits selection of one or two different modes of operation called (MODE 0) and (MODE 1). Logic 0 in bit D2 selects MODE 0, while logic 1 selects MODE 1. The next 4 bits in the control register, D3 through D6, correspond to the group A control block in figure 4.1. Bits D3 and D4 of the control register are used to configure the operation of the upper half of port C and all of port A, respectively. These bits work in the same way as D0 and D1 configure the lower half of port C and port B. There are now two mode select bits D5 and D6 instead of just 1, they are used to select between three modes of operation known as MODE 0, MODE 1, and MODE 2.

The last control register bit D7, is the mode set flag, it must be at logic 1(active) whenever the mode of operation is to be changed [7].

4.3 The operating modes of the PPI 8255A

As it mentioned in section 4.2, there are three operating modes for the 8255A PPI(programmable peripheral interface), which are MODE 0, MODE 1 and MODE 2.

According to the figure 4.3 in section 4.2, we notice that the all ports of the PPI 8255A can be set in any one of the three modes of operation.

To set all ports A, B and C in MODE 0 operation, load bit D7 of the control register with logic 1, bits D6D5 = 00, and D2 = 0, logic 1 at D7 represents an active mode set flag.

Thus the ports A, B and C are set on mode 0, and the ports A, B can be configured as 8-bit input or output ports, port C can be configured for operation as two independent 4-bit input or output ports, this is done by setting or resetting bits D4, D3, D1, and D0.

Similarly, for setting all of the ports in MODE 1 operation, load D6D5 = 01, and D2 = 1, bit D7 = 1 to activate the mode set flag.

In this way the ports A, B are configured as two independent byte-wide input or output ports, each of which has a 4-bit control data port associated with it. The control data ports are formed from the lower and upper nibbles of port C, respectively.

On the MODE 2, the setting of the ports in this operation mode will be done by loading bit D6 = 1, bit D7 = 1 to activate the mode set flag, other bits haven't an effect for the setting of this mode operation, so it can be loaded by either 0 or 1 [7], [8],[42].

4.4 Using the suitable mode operation for the system design

Mode 0 is the only mode operation from the PPI modes is a suitable for our system, our research work sees that for the following reasons:

- 1- Data is simply written to read from a specific port, it provides simple input and output operation for each of the three ports, and it is a bidirectional mode.
- 2- In this mode there is no "hand shaking" signal as in MODE 1, and MODE 2.

So, this operation mode is important to achieve the idea of our design, and increase the performance.

4.5 The corporation between PPI (8255A) and Computer's CPU

The interfacing of the 8255A programmable peripheral interface is done by the using of the PPI 8255A interface card or circuit, which plugs into any variable 8 or 16-bit slot (AT-slot) on the computer's motherboard for doing both digital input and output to the computer. This card is simple to use and powerful! With this card you can interface and control almost any device such as DC (direct current) and stepper motors, relays, transistors, LCDs, keypads, DAC (digital to analogue) and ADC (analogue to digital) converters. This card is compatible with all computers. The included 34-pin connector cable, brings all 24 digital lines, and +12, -12, +5, -5 V lines to the Terminal Expansion Board. The included Terminal Expansion Board has standard header posts so that you can easily hook-up your sensors, motors, and engineering creations. It even has a 152-hole prototyping area so that you can build any of your additional circuits on if you wish. Assigning an address to the card can be made physically by using jumpers on the card, and software can tell the computer's CPU (Central processing Unit) what the address is.

The three ports 8-bit data bus buffer is used to interface the PPI 8255A to the system data bus, these ports are: A, B and C.

Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.

Control word and status information are also transferred through the data bus buffer [43].

4.6 How does the device driver manage the system

On the system device drive, parallel interface device and the computer processor controlling all are connected on one system environment, where the device driver manages the working environment of the system as we see in the following:

1- C codes can define and address the three ports of the PPI (8255A),

Which are (port A, port B, and port C) and also the control register (Cr), as we see in the flowing codes:

```
# define pa -----  
# define pb -----  
# define pc -----  
# define cr -----
```

} Addresses

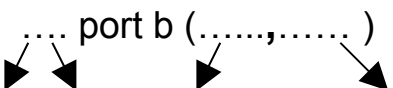
See Appendix for details.

2- Loading the control register (Cr) by the control word, which provides the main management for the device driver; since it contains information about the configuration of the ports to be input or output, and it determines the modes operation for setting it in 0, 1 or 2, and the bits setting or resetting.

The loading of a control word X to the control register can be achieved by the following code:

out port b (cr , X)

3- Manipulating any other port in the PPI (8255A) in both input or output cases, as we see in the following codes:



In out port name hexadecimal address

These codes can configure the PPI (8255A) through the system; ports defined and addressed and configured to be input or output, the operation mode selected and any input or output port can also be manipulated through the system.

Scientific Application

Chapter Five

Scientific Computing Application Example

On this chapter, we have an idea of a scientific application related for device driver applications; we use our general system that we have designed and implemented on this thesis, and then we can see how this application idea can be achieved and executed through our system design, and we clarify the ability of C characteristics through the system and through this type of applications, this evaluations and all the results are discussed and analyzed scientifically.

5.1 Application example idea

The application is (Electronics Security System); it is an electronic and mechanical system used to observe and guard a specific area of interest to the security system.

This model depends basically on observing the area that guarded electronically by using cameras.

These cameras are sat on stepper motors for photographing any moving bodies entering this area.

If a moving body enters from one side, camera in that side will start moving and photographing the object while the other cameras are still.

Observers in observing room are supplied with information from computer screen about the side which is entered.

In the case of entering more than one moving bodies from different sides at the same time, then the system will supply the security man in his room with information about the sides which were entered and which camera. The camera will start moving and photographing to detect the intruder. The model supplies other different security services such as:-

1- Control handling to move camera or cameras to a place or aside that we want to observe it.

2- Observing a specific place or establishment in the guarded area; the names of establishments and other places are showed on computer screen. Thus the security man in his room can move a camera to a place or establishment named (x) for example, just by choosing it from computer screen, in the model the establishments in the guarded area could be colleges, gardens and research centers.

In the application only two cameras are enough to observe the four sides of guarded area where each of them has the responsibility of photographing two perpendicular sides.

5.2 Applying the application example by the system design

If we look to the application example idea, we notice that the main input and output parts are:

- Stepper motors and its driving circuits for moving cameras.
- Power supply.
- Infrared sensors.

Power supply and infrared sensors have to be input parts, and stepper motors and its driving circuits are output parts.

So, power supply and infrared sensors attached on input port of 8255A (assuming to be port A). Stepper motors and its driving circuits attached on output port (assuming to be port B), just by determining the input and output parts of the application example, our system can achieve the application; and gives the following design procedure for this application:

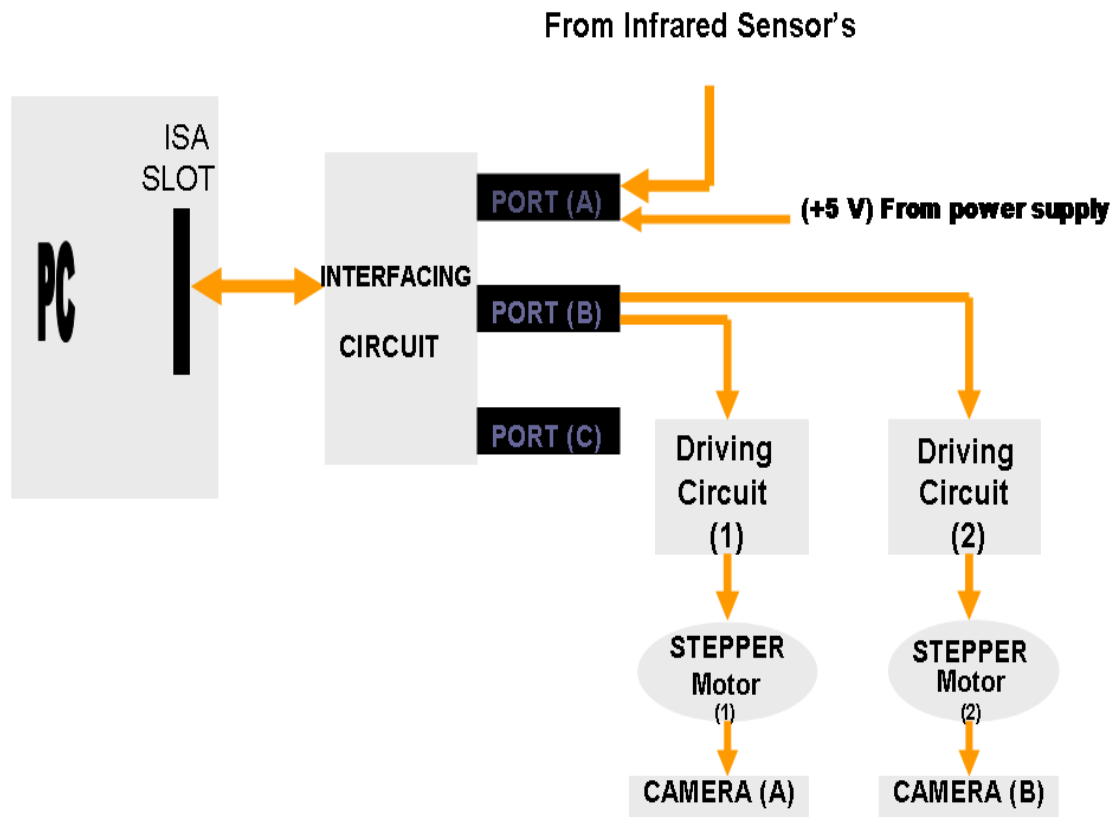


Figure5.1: Application example system design

This figure presents the design procedure of the application example, through our system design.

After running the C software to the design, according to main C control and management codes that related to our general system design, which

we discussed in chapter 4, the application example will be ready to run and execute.

When we want to talk about the external hardware side in application example, we will talk about:

- Stepper motors and its driving circuits for moving cameras.
- Power supply.
- Infrared sensors.

5.2.1: Stepper motors through application example

The goal of putting a stepper motor is to drive the camera which protects the area, and because it's not moving continuously. It can be derived by a pulses from a driving circuit to limit it's rotation upon the needed in the applications need.

Actually in the application each camera will protect and scan 90° so the motor will rotate maximum 90° .

The stepper motor is a synchronous motor, and typically has three-phase or four-phase windings on the stator and when a pulse is given to one of the phases of the stator the rotor tends to align with the MMF(magnetic

force) axis of the stator coil and the coils switched and the rotor follows the stator MMF in sequence.

The stepper motor used is $1.8^\circ/\text{step}$, which indicates that to rotate the motor 90° we need 50 pulses (steps). And also it has four phases, which means to have four coils each coil has two wires. Which means four wires will be an input pulse for the steps and the other wires will be put to a power supply with +5V.

The excitation of the motors will happen when two adjacent coils will be excited which means that the motor is Two-Coil Excitation.

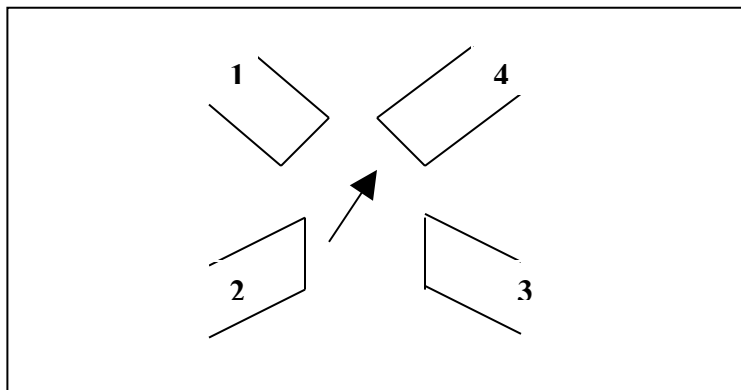


Figure 5.2: Excitation of stepper motor coils

The following sequences are supposed:

Step A: coil1 input (1), coil2 input (1), coil3 input (0), coil4 input (0)

Step B: coil1 input (0), coil2 input (1), coil3 input (1), and coil4 input (0)

Step C: coil1 input (0), coil2 input (0), coil3 input (1), and coil4 input (1)

Step D: coil1 input (1), coil2 input (0), coil3 input (0), and coil4 input (1)

Because of using 1.8° stepper motor each step of these means the motor will rotate 1.8° , and after these four steps the motor rotated $1.8^\circ \times 4 = 7.2^\circ$.

After this sequence of inputs the motor will rotate and this process is repeated to derive the motor to one direction, and if there is a needing to reverse the motor this sequence must be in the opposite direction as follows:

Step A: coil1 input (0), coil2 input (0), coil3 input (1), coil4 input (1)

Step B: coil1 input (0), coil2 input (1), coil3 input (1), and coil4 input (0)

Step C: coil1 input (1), coil2 input (1), coil3 input (0), and coil4 input (0)

Step D: coil1 input (1), coil2 input (0), coil3 input (0), and coil4 input (1)

5.2.2 Power supply

Supplies power throughout the computer, convert potentially lethal 110-115 or 220-230 volt alternating current (AC) into a steady low-voltage direct current (DC) usable by the computer. A power supply is rated by the number of watts it generates.

The output lines of power supply have different voltages according to the color of the line, as we see in the following table:

Color of power supply line	Value of voltage
Red	+5V
Yellow	+12V
Black	Ground
Blue	-12V

Table 5.1: Wire voltage of power supply

In the application a power supply used from a PC which has four main lines as we mentioned that needed for the application, which are:

- YELLOW Line: +12V and supplies 5A.
 - 1- Used to supply Driving circuit.
 - 2- Used to supply the electronic Sensor.
- BLACK Line: Ground
- RED Line: +5V and supplies 20A.

- 1- Used to supply the motor.
- 2- It is used as input to port A to indicate a signal when the power is off, then stopping the pulses until returning the power again.

Note that when the power is OFF there will be an indication to the processor to wait the power until return. And that happen by using the RED line as input to port A.

5.2.3 Infrared Sensor

It is a passive infrared intrusion detector for electronic security system. It detects intrusion by determining changes in infrared energy patterns. It emits no radiation and is harmless to humans and animals.

The passive infrared sensor employs variable pulse-width adjustment and a highly accurate range control to virtually eliminate false alarms without sacrificing detection ability, this sensor performs better when provided with a constant and stable foreground. The protected area should not exceed the detector's selected detection range.

The maximum horizontally range the sensor protect is 120° and Vertically the range depends on a calibration screw put on the sensor electronic board; this screw can be celebrated to protect maximum 5

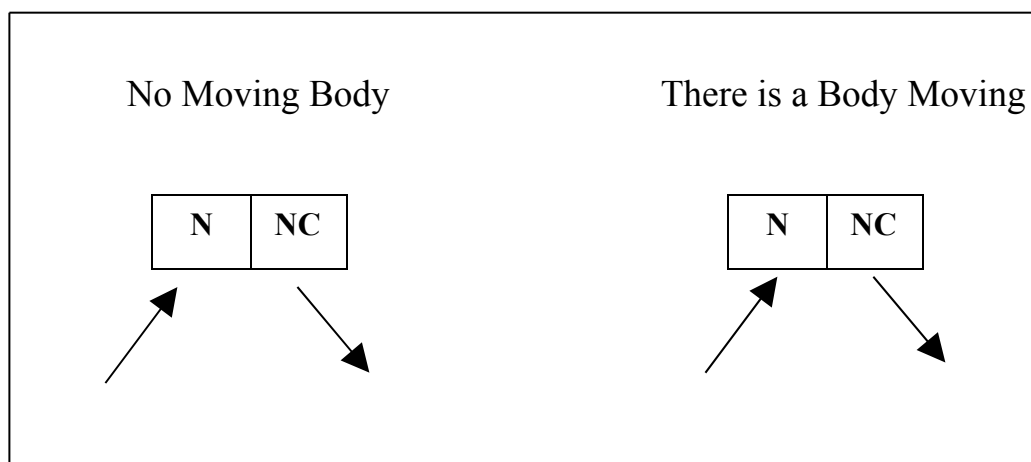
Meters up and 3 meters down with angle 120° which is enough to good protection also from humans and small animals.

Connect the power inputs (8.2 TO 16Vdc) to the 12v + and – terminals.

We connected these terminals to 24 hour normally closed circuit protective in the control unit and if the front cover of the detector is opened an immediately alarm signal will be sent and open the circuit.

We connected the terminals N and NC which is in the board of the sensor with VCC (Voltage Control Source) and one of the port A inputs, normally when no body is moving this circuit will be closed and the VCC will still arrive to Port A input, but as soon as any body moves around a signal appears and the circuit will be open and the VCC will stop flowing to port A input so the signal which arrives to Port A input is 0.

No problem where we will make the input from the sensor to Port A, we decided to put on bit 7.



5v

5v

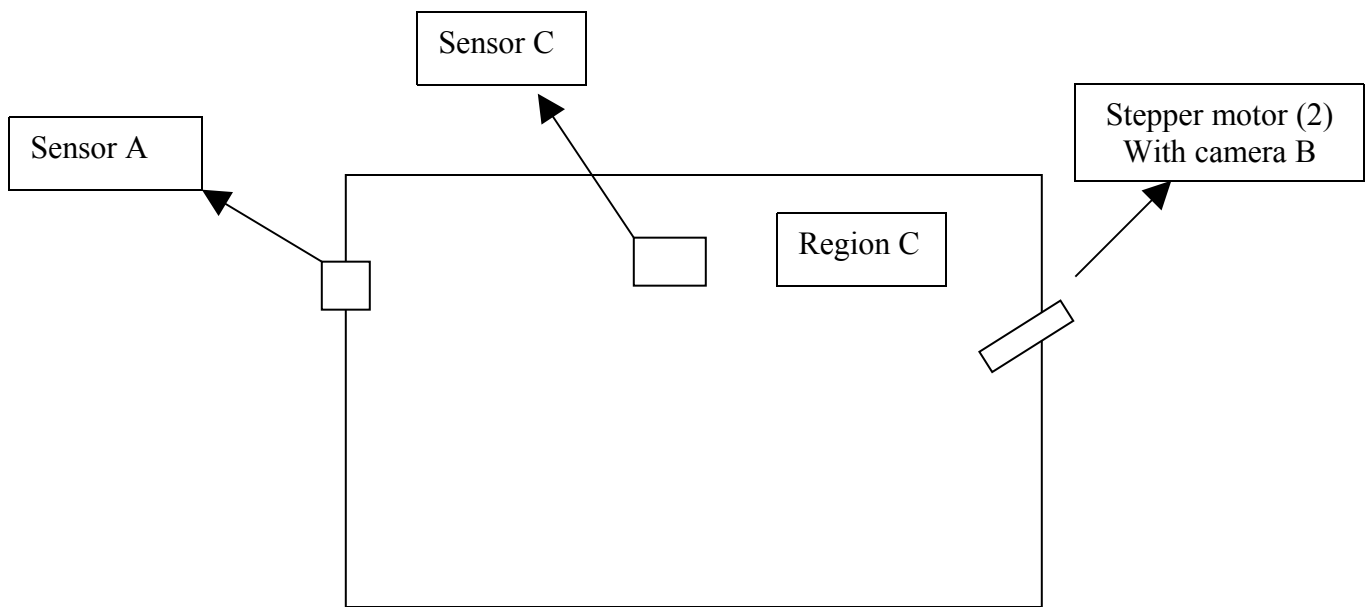
5v

0v

Figure 5.3: Controlling the movement by infrared sensors

5.3 The expected model design for the application example

The following figure appears the form of model design for the application example "Electronic security system", its appear the location of stepper motors, sensors and the guarded area:



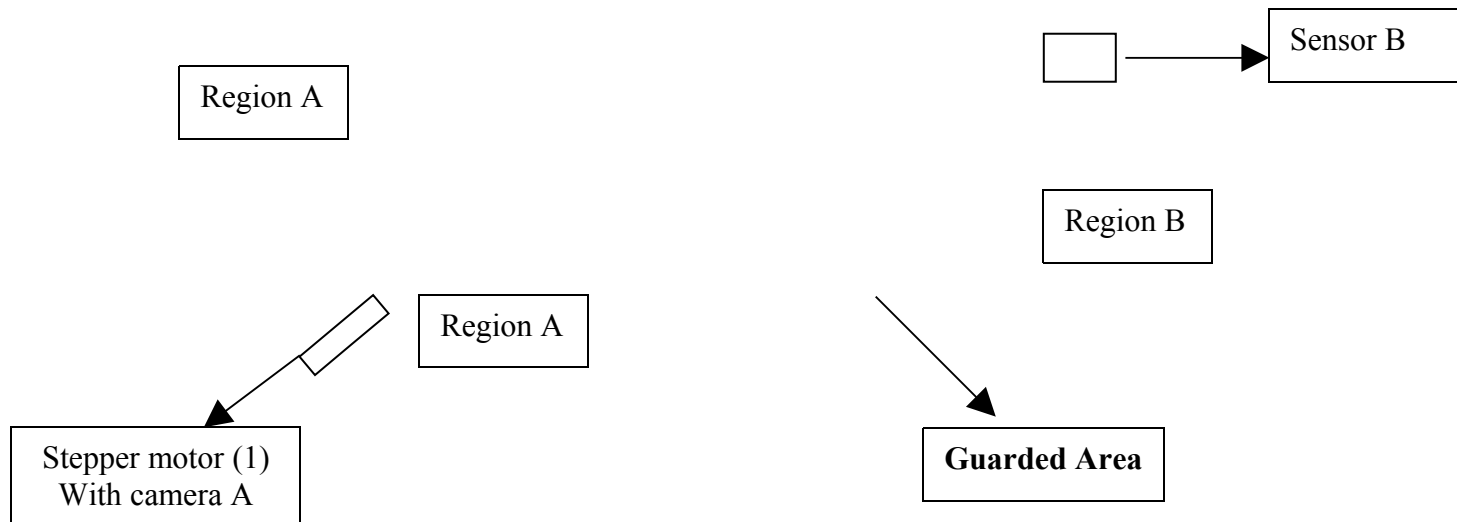


Figure 5.4: Application example model design

5.4 Analysis the system management for parallel interfacing

System design provides a parallel interfacing by the Intel PPI (8255A), the circuit of the device driver can connect to Industry Standard Architecture (ISA) slot on a computer.

On application example we want to make port A input, and ports B, C outputs; this is will be done by returning to the main control C codes on our software system design and using the following code :

`out port b (cr , X)`

Where X is a control word that will be loaded to the control register.

By choosing X to be 90H, the code will be :

`out port b (cr , 0x90)`

Which makes port A input, and ports B, C outputs, we can see the control register after loaded by 90H, if we return to chapter 4, figure: 4.3.

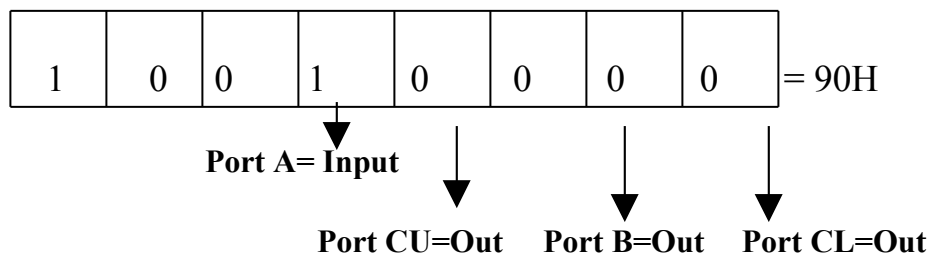


Figure 5.5: Control register loaded by 90H

To give an address for the interfacing card the two decoders located in the interface card are used which is 74LS154 (4*16 decoder), each decoder have 10 lines (A0-A9), one of the decoders uses A6-A9 lines, and the other uses A2-A5 lines, and to select the ports (Port A, Port B, Port C, And the control register) we use the lines A0 and A1.

Now, application example can use the ports as following:

1- Port A:

The port A which has 8 pins used as input ports from signals come through the following:

- **Infrared Sensor A:** this sensor protects region (side) of the model which is A, the signal from the sensor is connected to D7, and it acts as active low signal which means that it's normally 1 if there is no moving body, and as soon as something moves the signal will be 0 (opens the circuit).
- **Infrared sensor B:** this sensor will protect Region B in the model, and it's connected to D6, the signal is normally 1, and when a body enters the region the signal will be 0.

- **Infrared Sensor C:** this sensor will protect region C, and its signal connected to D4, and acts as sensor B (active high).
- **Power Supply** (Red line = 5v): this line is connected to D5, normally when the power supply is ON there will be a signal on D5 and the pulses will continue,
- But when the power supply get OFF the signal will be Zero and the Loops in the software will stop until returning the power supply being ON, so it's active low line.

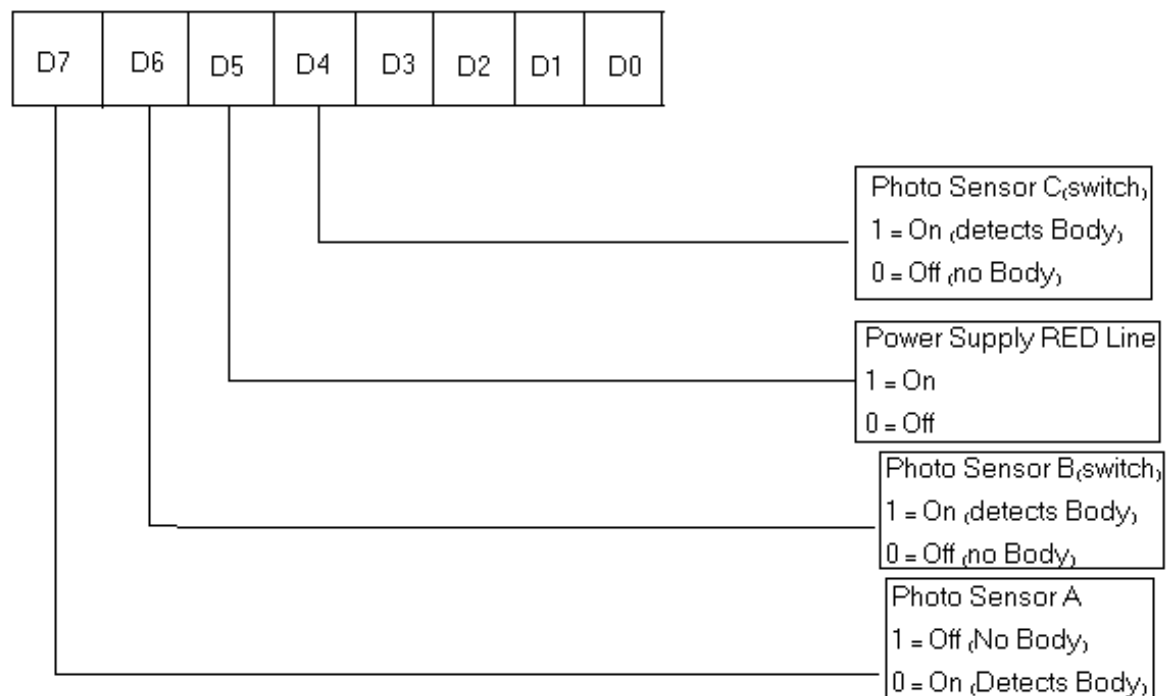


Figure 5.6 : Connecting the power supply and infrared sensors with Port A.

The Main Input Words on Port A is:

05H: There is no Moving Bodies, Power Supply is ON

04H: a body Enters in Region A, Power is on.

07H: a body Enters in Region B, Power is on.

0DH: a body Enters in Region C, Power is on.

0FH: a body Enters in Region B, C Power is on.

0EH: a body enters in Region A, B, C Power is on.

2- **Port B:**

This port is used an output Port to supply Pulses to the driving circuit of the motors, so we loaded this Port initially with an integer $x = 6666H$ to get a sequence of Pair of ones in each four bits, we have used The Port as Follows:

- B0-B3: used to drive the motor A.
- B4-B7: used to drive the motor B.

To prevent motor B to rotate in case (A0-A3) we ANDED X by 000F then A3-A7 will be Zeroes, and when driving motor B we ANDED x by 00F0, So A0-A3 will be zeroes.

5.5 Controlling the stepper motors movement

As mentioned on this chapter, port B of the Intel PPI (8255A), is satisfied to be output port for stepper motors, for controlling stepper motor movement we use the driving circuit.

Driving circuit consists the following:

1. IC L293NE: supports four channels with 1 A of current maximum, with 2 enable lines, each one enables two of the four channels. The four inputs are connected to driving sequence lines four lines from port B. when the input of the IC is high, the output of the corresponding line is 12V, and when the input is low the output is 1.5V.
2. Two clamping diodes: these diodes protect the IC from back currents which may burn the IC.
3. 200Ω resistor: this resistor helps in biasing the TIP31C transistor and make its operation in the saturation region.

The base current when high input will be:

$$I_B = \frac{V_{ss} - V_{BE}}{200\ \Omega} = \frac{12\text{V} - 1.8\text{V}}{200\ \Omega} = 50\text{mA}$$

And the saturation collector current will be ($V_{ce(sat)} = 1.2\text{V}$):

$$I_{c(sat)} = \frac{V_{cc} - V_{cE(sat)}}{R_{coil} + R_{cE}} = \frac{5V - 1.2V}{1.4 + 0.4} = 2.10A$$

When there is no input (OFF) the coil resistance is 1.4Ω , and when ON it will increase depending on magnetic field generated, and collector current will be $1.70 A$

5.6 Analysis of the software application example

1. Power supply controlling analysis:

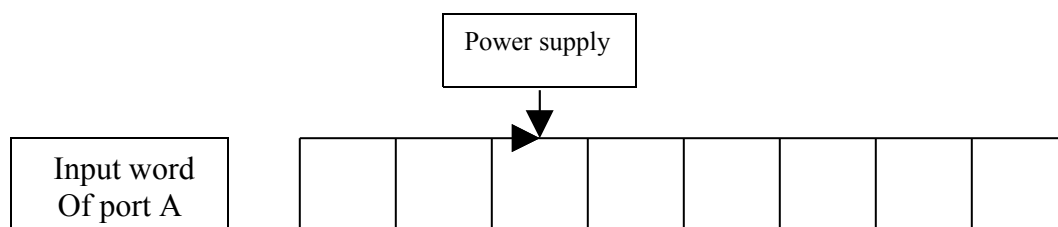
The power supply must be ON while running the software, and as we mentioned before, the power supply is connected on D5 in the input word of port A (see figure 5.6).

If D5 is one "1" this means power supply is ON, and if D5 is zero "0" the power supply will be Off.

To achieve that power supply is (ON) during all the software, the input word of port A is logically AND'ed with (04 H) by the command

if((inportb(pa)&0x04)==0x04) during all the software, which makes D5 is always one "1".

So the power supply will be ON and the following figure explains more about this operation:



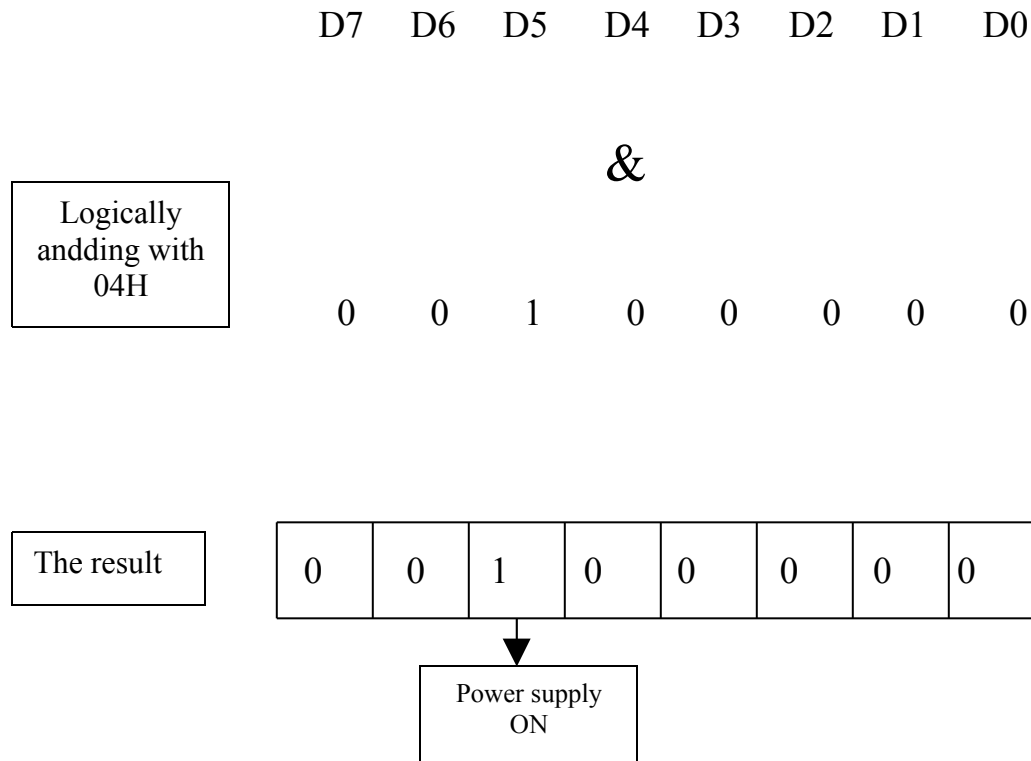


Figure 5.7: Analysis of power supply controlling

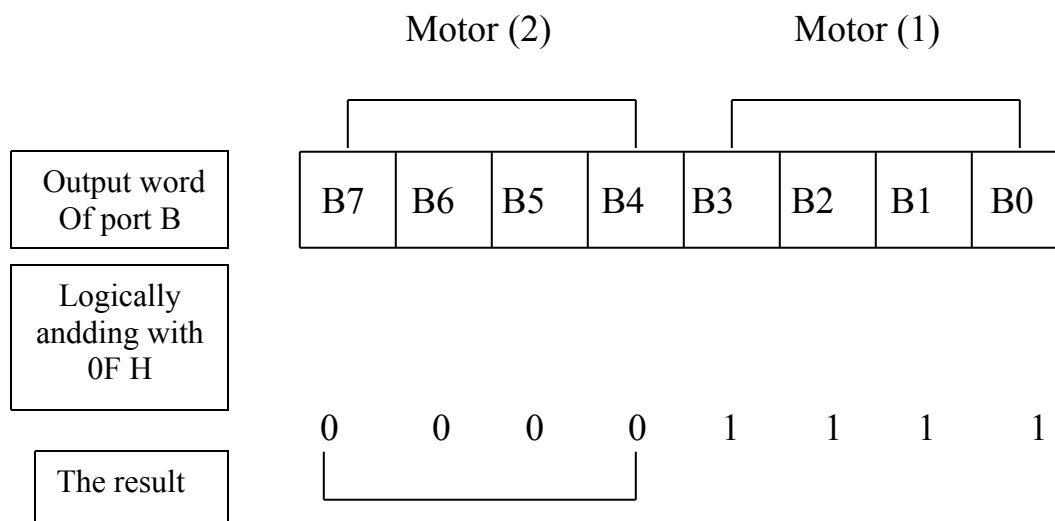
2. Stepper motors controlling analysis:

The stepper motor will be moved depending on the words loaded to

port B. In the software some cases need to move motor (1) only, or moving motor (2) only, and sometimes there is a needing to move both of them at the same time, so port B are divided into two parts:

Part 1(B0-B3): to drive stepper motor (1) so when using this part B4-B7 is canceled by the logically ANDING. The word loaded to Port B with 0F. This operation achieved by the command: **int y=0x000f&x.**

The following figure shows the operation in detail:



Motor (2) canceled

Figure 5.8: Canceling the work of motor 2

Part 2(B0-B3): to drive stepper motor (2) so when using this part B0-B3 is canceled by logical ANDING the word loaded to Port B with F0, this operation achieved by the command: **int y=0x00f0&x.**

The following figure shows the operation in details:

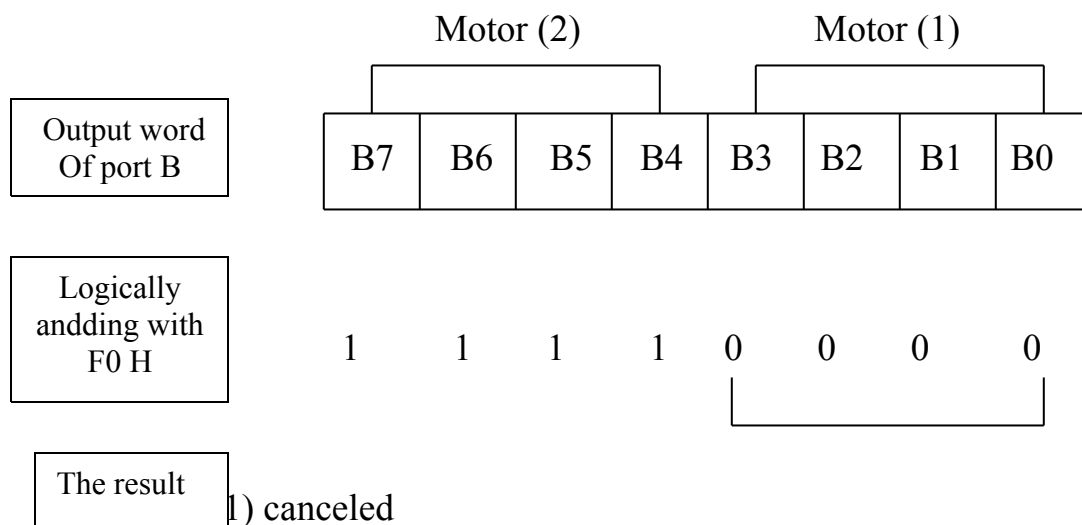


Figure 5.9: Canceling the work of motor 1

In the case of moving the two stepper motors at the same time, the two commands: **int y=0x000f&x**, **int y=0x00f0&x** are used in different two loops of the same program.

The command "rot" used in two forms there are "rotr" for rotating the motor to the right side, and "rotl " for left side, the delay time must be suitable for stepper motors movements.

For all application example software in C, see **Appendix B**.

Conclusion

We can't say that C doesn't excel in scientific computing applications, at the same time; C doesn't always succeed in this domain,

It is said that [15] "there are many fields C doesn't excel, such as scientific computing", but this may be right on some applications but not at all. C is not the problem, but how to use C is the problem. There is wide range of applications which C isn't s only relevant but it also powerful, on the contrary C will be a poor tool on other applications.

Although mathematical applications are a primary concern to speed which C satisfies, but this work recommends not to use C on this domain.

Using C is too different from C++ through scientific applications. The advantages features for object oriented (OO) doesn't make C++ better than C at all. On some applications the OO features may not be a primary concern as other factors which C achieves and C++ doesn't.

All of these ideas, problems, and results are manipulated scientifically by both: research and analysis.

The idea of converting a system design of parallel interface application into a general system can be achieved if we use parallel interfacing with

a parallel peripheral interface PPI (8255A) as interface device, and computer's CPU for main system controlling, and a C as a device driver and system software. This design system will be general for using it for multiple applications with a good level of efficiency and performance. The full design and implementation of the system was scientifically discussed and analyzed through this work. We have evaluated the performance of designed and the efficiency of C through analyzing the example which was demonstrated in chapter five.

Future Work

Try to design a parallel interface system with C management, which Using a PCI (Peripheral Component Interconnect) computer's slot Instead of ISA (Industry Standard Architecture) slot, to plug the PPI 8255A circuit, this is will increase the speed of data transfer while interfacing and controlling the input/ output devices of the system, especially that PCI can transfer the data at least with 132 MB per second, while ISA with 16 MB per second.

References

- 1- Patt, yale N, 2001, *Introduction to computing systems*, Mc Graw Hill.
- 2- Buchanan, William, 1995, *C for electronic engineering*, Prentice Hall.
- 3- Etter, Delores M, 1995, *Engineering problems solving with ANSI C*, prentice Hall.
- 4- Ferraris, Guido Buzzi, 1993, *Scientific C++*, Addison-Wesley publishing company.
- 5- Ege, Raimund K, 1994, *Object-Oriented programming with C++*, Ap Professional.
- 6- Groth, David, 2003, *A+ Complete*, San Francisco. London.
- 7- Singh, Avtar and Triebel, Walter A, 1991, *16-Bit and 32-Bit Microprocessors*, Prentice Hall.
- 8- Division, R&D, 1991, *Experimental book of MTS-88.C the 16 bits. 8088 CPU*, K&H MFG. CO. LTD.

- 9- Gibson, Gleen A and Liu, Yu-Cheng, 1986, *Microcomputer systems the 8086/8088 family*, Prentice Hall.
- 10- Tanenbaum, Andrew S, 1986, *Structured Computer Organization*, Prentice Hall.
- 11- Press, William H Teukolsky Saul A Flannery Brian p and Vetterling William T, 1993, *Numerical Recipes in C*, Cambridge University Press.
- 12- Bilson, Richard C.2003. Implementing overloading and polymorephism in C for all. M.Sc diss., University of Waterloo.
- 13- Fu, Guangrui. 1999.Design and implementation of an operating system in standard ML. M.Sc diss., University of Hawaii.
- 14- Jye Chang, yuh.1999. Architecture of fine-grained data flow network programming environments. PhD diss., University of Syracuse.
- 15- Zhong, Weilin.2001. Why C is not my favorite language.
<http://www.cs.virginia.edu/~wz5r/cs655/whycnot.htm>
- 16- Fateman, Richard.2000. Why C is not my favorite language.
<http://www.cs.berkeley.edu/~fateman/papers/software.pdf>

- 17- Grenning, James.2003. Why are you still using C.
<http://www.objectmentor.com/resources/articles/WhyAreYouStillUsingC.pdf>
- 18- Ricdude.2001. Athread on the C programming language.
<http://www.advogato.org/article/230.html>
- 19- Whiteley, Jonathan.2005. C++ for scientific computing.
<http://web.comlab.ox.ac.uk/oucl/courses/topics04-05/cs>
- 20- Wilson, Kyle.2006. Why C++.
<http://gamearchitect.net/Articles/WhyC++.html>
- 21- Zolman, Leor.2006. Why Java should be your first programming language.
http://www.course.com/techtrends/java_0899.cfm
- 22- Johnson, s.c and Ritchie, D.M.1978. Portability of C programming and the Unix system, Bell System Technical Journal.
- 23- Krahn, Joe.2006. Scientific programming: perl or python.
<http://www.niehs.nih.gov/Connections/2002/sept/python.htm>
- 24- Coombs, M.J and Alty, J.L, 1981, *Computer skills and the user interface*, Academic Press.
- 25- Heller, Martin.2001.Solutions for scientific computing in engineering and science.

- 26- Kernighan and Ritchie.2006.Introduction to programming for scientific computing.<http://www.math.nyu.edu>.
- 27- Why learn C.2006.<http://www.cprogramming.com>.
- 28- Why not C.2003.<http://www.catb.org>.
- 29- Why C and C++ are awful programming languages.2006.
<http://www.cs.rice.edu>.
30. (Alavoor Va sudevau), AI Dev.2002.C++ programming How-To.
<http://www.linuxarkivet.se>.
- 31- Ztf.2001.Sounds like the goals of java.
<http://www.advogato.org/article/230.html>
- 32- Moshez.2001.What is C.
<http://www.advogato.org/article/230.html>
- 33- Intel.2004.Motorola- to- Intel IXP4XX Product Line and IXC1100 Control Plane Processors.
<http://download.intel.com/design/network/swsup/25323202.pdf>
- 34- What is C/C++.2005.[http:// www.techiwarehouse.com/cms/articles.php](http://www.techiwarehouse.com/cms/articles.php).

- 35- Lam, C.2004. Engineer to Engineer Note/ Interfacing TFT LCD Panels to ADSP-BF533 Blackfin processors via PPI.
[http://www .analog.com/ee-notes](http://www.analog.com/ee-notes)
- 36- ADI's DSP Applications and Development Tools Groups.1998.Engineer to Engineer Note/ADSP-2100 Family.
<http://www.analog.com/dsp>
- 37- Bjarne Stroustrup.2000. Appendix B/ Compatibility.
[http:// www.research.att.com/~bs/3rd_compat.pdf](http://www.research.att.com/~bs/3rd_compat.pdf)
- 38- Smith, Steven W, 1997, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing.
- 39- An Introduction to C.2000.
<http://www.apnet.com/bookscat/samples/9780750648318/9780750648318.pdf>
- 40- Ziring, Neal.1999. Dictionary of programming languages.
http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl?_alpha=c
- 41- Hughey, Richard, IEEE Computer Society,1992.Programming Systolic Arrays,Computer Engineering Board-University of California.
<http://www.soe.ucsc.edu/research/kestrel/papers/asap92.pdf>

- 42- T.D.Burd and R.W. Broderson. Processor design for portable system.
Journal of VLSI Signal processing, 13(2/3):203-222, August/
September 1996.
- 43- N.Morgan, J. Beck, P. Khon, j. Bilmes, E. Allman, and J. Beer.
The Ring Array Processor(RAP): A multiprocessing peripheral
for connectionist applications. Journal of Parallel and Distributed
Computing, 14:248-259, April, 1992.
- 44- M.Bass, P.Knebel,D.W.Quint, and W.L.Walker.The PA 7100LC
Microprocessor: a case study of IC design in a competitive
environment. Hewlett-Packard journal, 46(2):12-22, April, 1995.
- 45- Dennis M.Ritchie. The development of the C language.In the
Second ACM SIGPLAN conference on History of programming
Languages, pages 201-208.ACM Press, April,1993.
- 46- Bjarne Stroustrup. C and C++: a case for compatibility.
The C/C++ Users Journal, July, 2002.

- 47- Andrew Tolmach. Tag-free garbage collection using explicit type Parameters. In Proceedings of the 1994 ACM conference on LISP And functional programming, pages 1-11. ACM Press, 1994.
- 48- Trevor Jim, Greg Morrisett, Dan Grossman, Michael Hicks, Yanling Wang, and James Cheney. Cyclone: A safe dialect of C. In USENIX Annual Technical Conference, June 2002.
- 49- [A History of C++: 1979-1991](#). Proc ACM History of Programming Languages conference (HOPL-2). ACM Sigplan Notices. Vol 28 No3, pp 271-298. March 1993. Also, History of Programming languages (editors T.J. Begin and R.G. Gibson) Addison-Wesley, ISBN 1-201-89502-1. 1996.

Appendix A

In this appendix we use the (Windows XP Performance Test) to improve the speed of C, by making a comparison between two simple similar programs, the first is written by C, and the second by JAVA.

Running the (Windows XP Performance Test):

- 1- Press Ctrl + Alt + Delete on the windows desktop.
- 2- Choose the (View Kernel Time) from View List.
- 3- Choose the (High Speed) from View List.
- 4- Choose (Performance).

The Comparison Experiment:

The C Program:

```
#include<stdio.h.>

main()

{

printf("welcome);

return 0;

}
```

The JAVA Program:

```
import java.io.*;

class hello{

public static void main(String[] args){

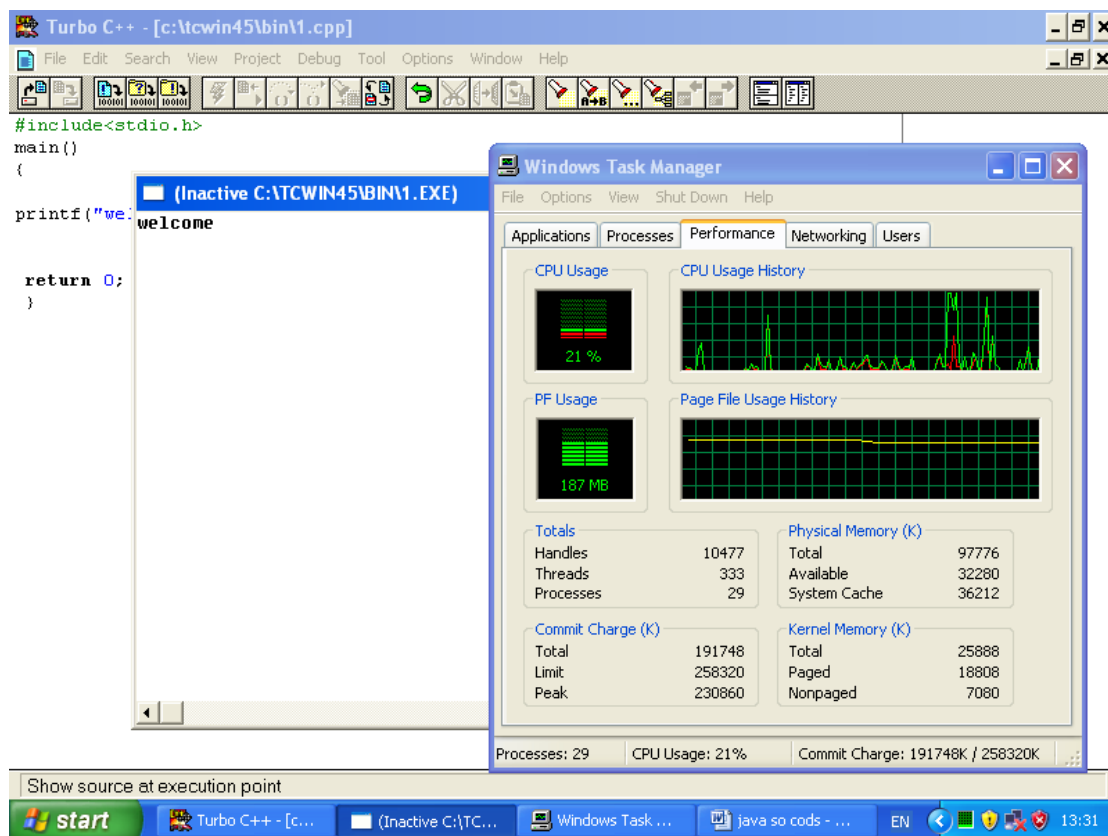
System.out.print("welcome");

}

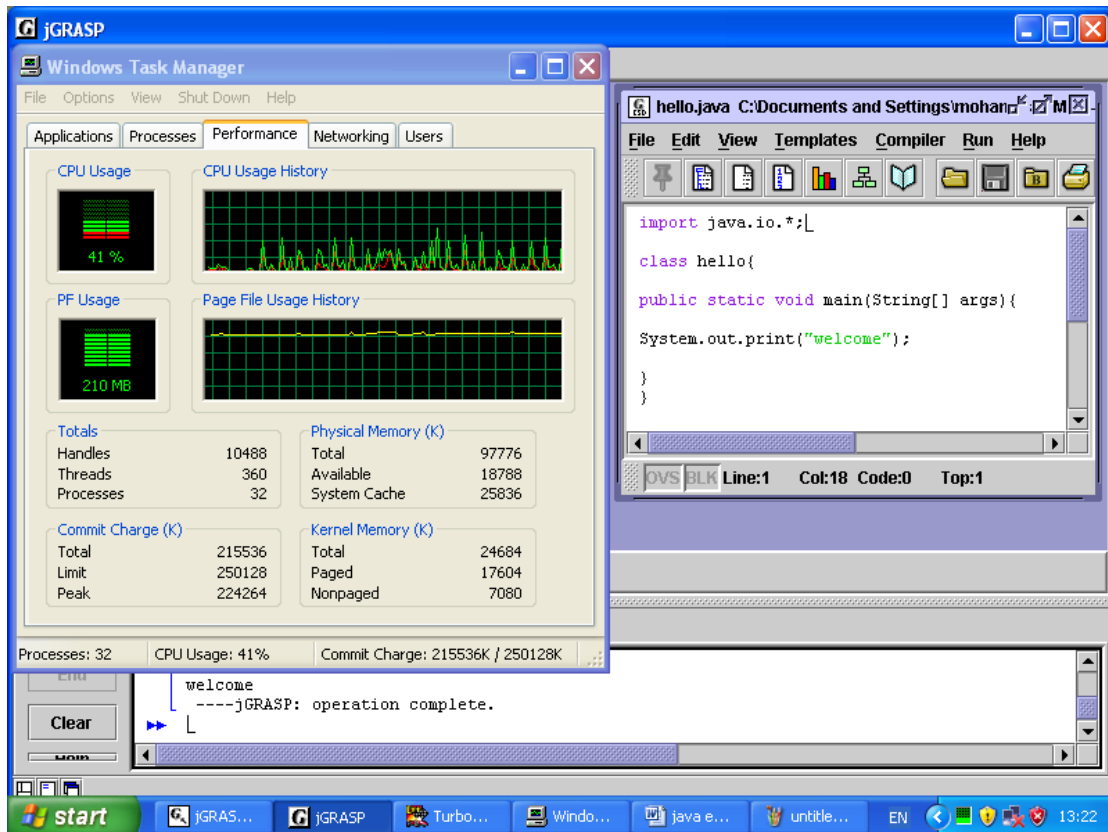
}
```

The output of the two programs is printing (welcome word).

After running the C program, The CPU Usage reaches to **21%**, as we see:

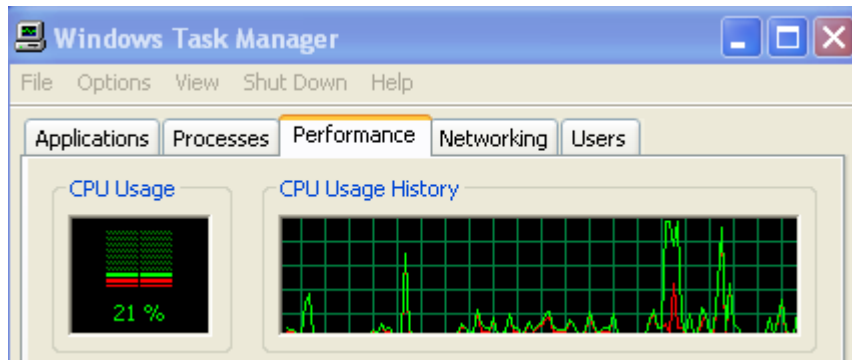


After running the JAVA program, The CPU Usage reaches to 41%, as we see:

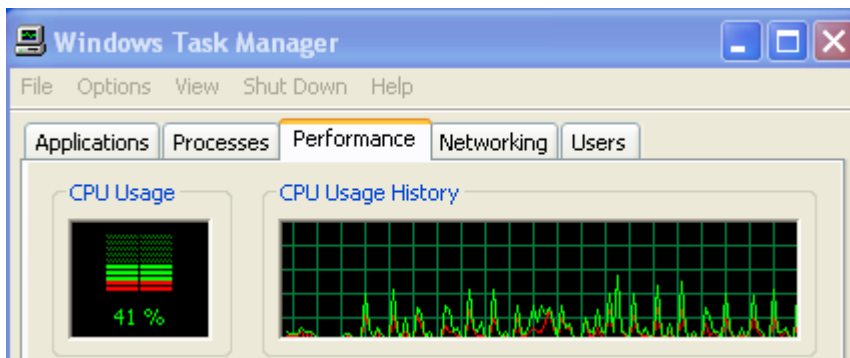


Conclusion of the results:

The CPU Usage by **C** program is **21%**



The CPU Usage by **JAVA** program is **41%**



Appendix B

Through this appendix, we can see the full software of application example:

```
/* wrtiten by Mohammad Abu Omar*/
/* M.Sc. thesis 2005-2006*/
#include<graphics.h>
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
#include<conio.h>
#define pa 0x300/* giving 300H as address to port A in the PPI 8255A*/
#define pb 0x301/* giving 301H as address to port B in the PPI 8255A*/
#define pc 0x302/* giving 302H as address to port C in the PPI 8255A*/
#define cr 0x303/* giving 300H as address to control register Cr in the
PPI 8255A*/

main()
{
int x=0x6666,    j=0, i=0, r=10, c=10;
/* loading the control register Cr in the 8255A by 80H to make port A as input
port and ports B, C as output ports*/

    outportb(cr,0x80);
    while((r!=0)&&(c!=0)){
        clrscr();

        printf("No body is moving now");
        printf("what is the camera you want\n1- Camera
A\n2- Camera B\n3- Both\n Enter your choice:");
        scanf("%d",&c);
        printf("\n where do you want the camera to
move:\n1-To College\n2-To Garden\n3-To the
Research Center\n4-Hand free moving\n5-Auto
protection\nEnter your choice:");
        scanf("%d",&r);
```



```

/* Moving Camera A to the College */

    if ((c==1) && (r==1)) {

        while(i<10) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */

            if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work  
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper  
motor1 and camera A will be drive */
                int y=0x000f&x;
                outportb(pb, y);
                clrscr();
/* Rotating the stepper motor 1 with camera A to the right side*/
                x=_rotr(x, 1);
                delay(35);
                i++;    }    }
                while(i>10) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                    if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work  
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper  
motor1 and camera A will be drive */
                        int y=0x000f&x;
                        outportb(pb, y);
                        clrscr();
/* Rotating the stepper motor 1 with camera A to the left side*/
                        x=_rotl(x, 1);
                        delay(35);
                        i--;    }    }
                    }

```

```

/* Moving Camera B to the College */
    if ((c==2) && (r==1)) {
        while (j<45) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the left side*/
                x=_rotr(x,1);
                j++;
                delay(35);
            }
            while (j>45) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                    int y=0x00f0&x;
                    outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the right side*/
                    x=_rotr(x,1);
                    j--;
                    delay(35);
                }
            }
        }
    }

```

/* Moving Camera A to the Garden */

```
    if ((c==1) && (r==2)) {  
  
        while (i<30) {  
  
            /* Checking if power supply is on or not, by andding port A Logically with 04H */  
            if ((inportb(pa) & 0x04) == 0x04) {  
  
                /* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work  
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper  
motor1 and camera A will be drive */  
                int y=0x000f&x;  
                outportb(pb,y);  
                clrscr();  
  
                /* Rotating the stepper motor 1 with camera A to the right side*/  
                x=_rotr(x,1);  
                delay(35);  
                i++; } }  
                while (i>30) {  
  
                    /* Checking if power supply is on or not, by andding port A Logically with 04H */  
                    if ((inportb(pa) & 0x04) == 0x04) {  
  
                        /* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work  
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper  
motor1 and camera A will be drive */  
                        int y=0x000f&x;  
                        outportb(pb,y);  
  
                        /* Rotating the stepper motor 1 with camera A to the left side*/  
                        x=_rotl(x,1);  
                        delay(35);  
                        i--; } }  
  
            }  
        }
```

```

/* Moving Camera B to the Garden */

    if ((c==2) && (r==2)) {

        while(j<20) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb, y);
/* Rotating the stepper motor 2 with camera A to the left side*/
                x=_rotl(x, 1);
                delay(35);
                j++;    }    }

            while(j>20) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                    int y=0x00f0&x;
                    outportb(pb, y);
/* Rotating the stepper motor 2 with camera A to the right side*/
                    x=_rotr(x, 1);
                    delay(35);
                    j--;    }    }

        }

```

```

/* Moving Camera A and B to the College */

    if((c==3) && (r==1)) {
        while(i<10) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa) &0x04)==0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                int y=0x000f&x;
                outportb(pb,y);
                clrscr();
/* Rotating the stepper motor 1 with camera A to the right side*/
                x=_rotr(x,1);
                delay(35);
                i++;    }    }

            while(i>10) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if((inportb(pa) &0x04)==0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                    int y=0x000f&x;
                    outportb(pb,y);
                    clrscr();
/* Rotating the stepper motor 1 with camera A to the left side*/
                    x=_rotl(x,1);
                    delay(35);
                    i--;    }    }

```

```

    while(j<45) {
        /* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa)&0x04)==0x04) {
            /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
            of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
            motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
            /* Rotating the stepper motor 2 with camera B to the left side*/
            x=_rotr(x,1);
            j++;
            delay(35);
        }

        while(j>45) {
            /* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa)&0x04)==0x04) {
                /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
                of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
                motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb,y);
                /* Rotating the stepper motor 2 with camera B to the left side*/
                x=_rotr(x,1);
                j--;
                delay(35);
            }
        }
    }
}

```

```

/* Moving Camera A and B to the Garden */

    if ((c==3) && (r==2)) {

        while (j<20) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb, y);
/* Rotating the stepper motor 2 with camera B to the left side*/
                x=_rotl(x, 1);
                delay(35);
                j++;    }    }

            while (j>20) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */
                    int y=0x00f0&x;
                    outportb(pb, y);
/* Rotating the stepper motor 2 with camera B to the right side*/
                    x=_rotr(x, 1);
                    delay(35);
                    j--;    }    }

            while (i<30) {

```

```

/* Checking if power supply is on or not, by andding port A Logically with 04H */
    if ( (inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
        int y=0x000f&x;
        outportb(pb, y);
        clrscr();
/* Rotating the stepper motor 1 with camera A to the right side*/
        x=_rotr(x, 1);
        delay(35);
        i++;    }    }

        while(i>30){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if ( (inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                int y=0x000f&x;
                outportb(pb, y);
/* Rotating the stepper motor 1 with camera A to the left side*/
                x=_rotr(x, 1);
                delay(35);
                i--;    }    }
        }

/* Moving Camera A to the research center */

        if ( (c==1) && (r==3) ) {

            while(i<44){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if ( (inportb(pa) & 0x04) == 0x04) {

```


/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper motor1 and camera A will be drive */

```
int y=0x000f&x;
outportb(pb,y);
clrscr();
```

/* Rotating the stepper motor 1 with camera A to the right side*/

```
x=_rotr(x,1);
delay(35);
i++; } }
```

```
while(i>44){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa)&0x04)==0x04){
```

/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper motor1 and camera A will be drive */

```
int y=0x000f&x;
outportb(pb,y);
```

/* Rotating the stepper motor 1 with camera A to the left side*/

```
x=_rotr(x,1);
delay(35);
i--; } }
```

```
}
```

/* Moving Camera B to the research center */

```
if((c==2)&&(r==3)){
```

```
while(j<15){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa)&0x04)==0x04){
```

/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */

```
int y=0x00f0&x;
outportb(pb,y);
```

/* Rotating the stepper motor 2 with camera B to the left side*/

```
x=_rotl(x,1);
delay(35);
j++; } }
```

```
while(j>15){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa)&0x04)==0x04){
```

/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */

```
int y=0x00f0&x;
outportb(pb,y);
```

/* Rotating the stepper motor 2 with camera B to the right side*/

```
x=_rotr(x,1);
delay(35);
j--; } }
```

/* Moving Camera A and B to the garden */

```
if((c==3)&&(r==2)){
```

```
while(j<20){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa)&0x04)==0x04){
```

/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */

```
int y=0x00f0&x;
outportb(pb,y);
```

/* Rotating the stepper motor 2 with camera B to the left side*/

```
x=_rotl(x,1);
delay(35);
j++; } }
```

```
while(j>20){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa) &0x04)==0x04) {
```

/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper motor2 and camera B will be drive */

```
int y=0x00f0&x;
outportb(pb,y);
```

/* Rotating the stepper motor 2 with camera B to the right side*/

```
x=_rotr(x,1);
delay(35);
j--; } }
```

```
while(i<30){
```

/* Checking if power supply is on or not, by andding port A Logically with 04H */

```
if((inportb(pa) &0x04)==0x04) {
```

/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper motor1 and camera A will be drive */

```
int y=0x000f&x;
outportb(pb,y);
clrscr();
```

```

    /* Rotating the stepper motor 1 with camera A to the right side*/
    x=_rotr(x,1);
    delay(35);
    i++; } }

    while(i>30){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa)&0x04)==0x04){
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
            int y=0x000f&x;
            outportb(pb,y);
/* Rotating the stepper motor 1 with camera A to the left side*/
            x=_rotl(x,1);
            delay(35);
            i--; } }
    }

/* Moving Camera A and B to the research center */
    if((c==3)&&(r==3)){

        while(j<15){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa)&0x04)==0x04){
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the left side*/
                x=_rotl(x,1);
                delay(35);
                j++; } }
    }

```

```

    while(j>15) {
        /* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) &0x04)==0x04) {
            /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
            of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
            motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
            /* Rotating the stepper motor 2 with camera B to the right side*/
            x=_rotr(x,1);
            delay(35);
            j--; }

    while(i<44) {
        /* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) &0x04)==0x04) {
            /* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
            of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
            motor1 and camera A will be drive */
            int y=0x000f&x;
            outportb(pb,y);
            clrscr();
            /* Rotating the stepper motor 1 with camera A to the right side*/
            x=_rotr(x,1);
            delay(35);
            i++; }
        while(i>44) {
            /* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa) &0x04)==0x04) {
                /* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
                of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
                motor1 and camera A will be drive */
                int y=0x000f&x;

```

```

        outportb(pb, y) ;
/* Rotating the stepper motor 1 with camera A to the left side*/
        x=_rotl(x, 1) ;
        delay(35) ;
        i--;    }    }
    }

/* Hand Moving Camera A in */

    if ((c==1) && (r==4)) {
        while(inportb(0x60) !=0x01) {
            while((inportb(0x60)==77) && (i<50)) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if ((inportb(pa) &0x04)==0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                    int y=0x000f&x;
                    outportb(pb, y) ;
/* Rotating the stepper motor 1 with camera A to the right side*/
                    x=_rotr(x, 1) ;
                    i++;
                    delay(15) ;
                }}
                while((inportb(0x60)==75) && (i>1)) {

/* Checking if power supply is on or not, by andding port A Logically with 04H */
                    if ((inportb(pa) &0x04)==0x04) {

/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                        int y=0x000f&x;
                        outportb(pb, y) ;

```

```

/* Rotating the stepper motor 1 with camera A to the right side*/
    x=_rotr(x,1);
    i--;
    delay(15);
    }}
    }

/* Hand Moving Camera B in */
    if((c==2) && (r==4)) {
/* Checking if pin A7 in port A is loaded by 0H, which makes sensor A ON
which means there is a body in region A*/
        while(inportb(0x60)!=0x01) {
            while((inportb(0x60)==30) && (j<50)) {
/* Checking if power supply is on or not, by adding port A Logically with 04H */
                if((inportb(pa)&0x04)==0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
                    int y=0x00f0&x;
                    outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the left side*/
                    x=_rotr(x,1);
                    j++;
                    delay(15);
                    }}

                    while((inportb(0x60)==31) && (j>1)) {
/* Checking if power supply is on or not, by adding port A Logically with 04H */
                        if((inportb(pa)&0x04)==0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
                            int y=0x00f0&x;
                            outportb(pb,y);

```

```

/* Rotating the stepper motor 2 with camera B to the right side*/
    x=_rotr(x,1);
    j--;
    delay(15);
    }}
    }

/* Hand Moving Camera A and B in */

    if((c==3)&&(r==4)){

        while(inportb(0x60)!=0x01){

            while((inportb(0x60)==77)&&(i<50)){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if((inportb(pa)&0x04)==0x04){
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                    int y=0x000f&x;
                    outportb(pb,y);
/* Rotating the stepper motor 1 with camera A to the right side*/
                    x=_rotr(x,1);
                    i++;
                    delay(15);
                    }}
                    while((inportb(0x60)==75)&&(i>1)){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                        if((inportb(pa)&0x04)==0x04){
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                            int y=0x000f&x;
                            outportb(pb,y);

```



```

/* Rotating the stepper motor 1 with camera A to the left side*/
    x=_rotl(x,1);
    i--;
    delay(15);
    }}

    while((inportb(0x60)==30)&&(j<50)){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa)&0x04)==0x04){
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the left side*/
            x=_rotl(x,1);
            j++;
            delay(15);
            }}

            while((inportb(0x60)==31)&&(j>1)){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                if((inportb(pa)&0x04)==0x04){
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
                    int y=0x00f0&x;
                    outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the right side*/
                    x=_rotr(x,1);
                    j--;
                    delay(15);
                    }}
            }
    }

```

```

        if ((c==3) && (r==5)) {
            clrscr();
/* Checking if pin A7 in port A loaded by 0H, which makes sensor A on which
Means there is a body in region A*/
            while (inportb(0x60) != 0x01) {
/* Checking if power supply is on or not, by checking if port A loaded with 04H */
                while (inportb(pa) == 0x04) {

                    while (i < 50) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                        if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                            int y = 0x000f & x;
                            outportb(pb, y);
                            clrscr();
                            gotoxy(10, 10);
                            printf("security system detects abody In region  
A be carefull!!!!!!!!!!!!!!");
                            textcolor(2009);
                            textbackground(120);
/* Rotating the stepper motor 1 with camera A to the right side*/
                            x = _rotr(x, 1);
                            delay(35);
                            i++;    }
                        }
                    while (i > 1) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
                        if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                            int y = 0x000f & x;

```

```

        outport(pb, y);
    /* Rotating the stepper motor 1 with camera A to the left side*/
        x=_rotr(x, 1);
        delay(35);
        i--; }
    }

/* Checking if pins A5-A7 in port A are loaded by 1H, and pin A4 is loaded by 0H,
which makes sensor B ON, which means there is a body in region B*/
        while (inportb(pa) == 0x07) {

            while(j < 50) {
/* Checking if power supply is on or not, by adding port A Logically with 04H */
                if ((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
                    int y = 0x00f0 & x;
                    outportb(pb, y);
                    clrscr();
                    gotoxy(10, 10);
                    printf("security system detects a body from  
region B be carefull!!!!!!!!!!!!!!");
                    textcolor(2005);
                    textbackground(120);
/* Rotating the stepper motor 2 with camera B to the left side*/
                    x=_rotr(x, 1);
                    delay(35);

                j++; }
            }

```

```

    while(j>1){
        /* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) & 0x04) == 0x04) {
            /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
            of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
            motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
            /* Rotating the stepper motor 2 with camera B to the left side*/
            x=_rotr(x,1);
            delay(35);

            j--; } }
    }

    /* Checking if pins A4-A7 in port A are loaded by 1,1,0,1H,
    which make sensor C ON, which means there is a body in region C*/
    while(inportb(pa) == 0x0d) {

        while(j<50) {
            /* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa) & 0x04) == 0x04) {
                /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the
                work of stepper motor 1 and camera A. B4-B7 in port B will be active and
                stepper motor2 and camera B will be drive */
                int y=0x00f0&x;
                outportb(pb,y);
                clrscr();
                gotoxy(10,10);
                printf("security system detects abody from
                Region C and right be carefull!!!!!!!!!!!!!!");
                textcolor(2011);
                textbackground(122);

                /* Rotating the stepper motor 2 with camera B to the left side*/
                x=_rotl(x,1);

```

```

    delay(35);
    j++; }
    }

    while(j>1) {
        /* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) & 0x04) == 0x04) {
            /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
            of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
            motor2 and camera B will be drive */
            int y = 0x00f0 & x;
            outportb(pb, y);
            /* Rotating the stepper motor 2 with camera B to the right side*/
            x = _rotr(x, 1);
            delay(35);
            j--; }
        }

        /* Checking if pins A4-A7 in port A are loaded by 1,1,1,1H,
        which make sensors B, C ON, which means there are bodies in regions B and C*/
        while(inportb(pa) == 0x0f) {

            while(j<50) {
                /* Checking if power supply is on or not, by andding port A Logically with 04H */
                if((inportb(pa) & 0x04) == 0x04) {
                    /* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
                    of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
                    motor2 and camera B will be drive */
                    int y = 0x00f0 & x;
                    outportb(pb, y);
                    clrscr();
                    gotoxy(10, 10);
                    printf("security system detects abody from
                    Regions B & C and right be
                    carefull!!!!!!!!!!!!!!");

```

```

    textcolor(2011);
    textbackground(122);
/* Rotating the stepper motor 2 with camera B to the left side*/
    x=_rotr(x,1);
    delay(35);
    j++;}
    }

    while(j>1){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the right side*/
            x=_rotr(x,1);
            delay(35);
            j--;}
        }}

/* Checking if pins A4-A7 in port A are loaded by 1,1,1,0H,
which make sensors A,B, C ON, which means there are bodies in regions A, B
and C*/
    while(inportb(pa) == 0x0e) {
        while(i<50) {
/* Checking if power supply is on or not, by andding port A Logically with 04H */
            if((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
                int y=0x000f&x;
                outportb(pb,y);

```

```

clrscr();
gotoxy(10,10);
printf("security system detects abody from All
Regions and right be carefull!!!!!!!!!!!!!!");
textcolor(2011);
textbackground(122);
/* Rotating the stepper motor 1 with camera A to the right side*/
    x=_rotr(x,1);
    delay(35);
    i++;}

    }

    while(i>1){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) &0x04)==0x04){
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
            int y=0x000f&x;
            outport(pb,y);
/* Rotating the stepper motor 1 with camera A to the left side*/
            x=_rotl(x,1);
            delay(35);
            i--;}
        }

while(j<50){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
    if((inportb(pa) &0x04)==0x04){
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
        int y=0x00f0&x;
        outportb(pb,y);

```

```

    clrscr();
    gotoxy(10,10);
    printf("security system detects abody from All
Regions    and right be carefull!!!!!!!!!!!!!!");
    textcolor(2011);
    textbackground(122);
/* Rotating the stepper motor 2 with camera B to the left side*/
    x=_rotr(x,1);
    delay(35);
    j++;}
    }
    while(j>1){
/* Checking if power supply is on or not, by andding port A Logically with 04H */
        if((inportb(pa) & 0x04) == 0x04) {
/* Loading port B by F0H, to cancel the B0-B3 in port B and also cancel the work
of stepper motor 1 and camera A. B4-B7 in port B will be active and stepper
motor2 and camera B will be drive */
            int y=0x00f0&x;
            outportb(pb,y);
/* Rotating the stepper motor 2 with camera B to the right side*/
            x=_rotr(x,1);
            delay(35);
            j--;}
        }}
    while((inportb(0x60) == 77) && (i <= 50)) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
        int y=0x000f&x;
        outportb(pb,y);
/* Rotating the stepper motor 1 with camera A to the right side*/
        x=_rotr(x,1);
        i++;
        delay(15);

```



```

    }
    while( (inportb(0x60)==75) && (i>=1) ) {
/* Loading port B by 0FH, to cancel the B4-B7 in port B and also cancel the work
of stepper motor 2 and camera B. B0-B3 in port B will be active and stepper
motor1 and camera A will be drive */
        int y=0x000f&x;
        outportb(pb,y);
/* Rotating the stepper motor 1 with camera A to the left side*/
        x=_rotr(x,1);
        i--;
        delay(15);
    }
    clrscr();
    gotoxy(15,15);
    printf("%d",inportb(0x60));

    delay(100);

}

    delay(35);
}

}

return 0;
}

```