Model Based Design Environment for the Embedded System
Case Study: Inverter Power Supply

By

MONA ABD EL-BASET ABO EL-DAHB

A THESIS

Submitted to the Department of Production Science and Technology,

Graduate School of Engineering, Gunma University

in Partial Fulfillment of the Requirements for

the Degree of Doctor of Electrical and Computer Science Engineering

Under Supervision of

Prof. WEI SHU GANG

Prof. YOICHI SHIRAISHI

Department of Production Science and Technology

Graduate School of Engineering

Kiryu-shi, Gunma, 376-8515, JAPAN

September 2011

ACKNOWLEDGEMENTS

First of all, praise to ALLAH for all - it is the prime honor for me that ALLAH makes me Muslim and shows me the reality of this life. Also, I would like to declare my thanks for ALLAH for giving me the patience and perseverance to work through this research.

I would like to express my deepest appreciation and thanks to the great Professor **Yoichi Shiraishi** for his brilliant guidance, as well as his tremendous and earnest supervision, not only throughout the duration of this period of study, but also in all my general life matters. He always gives his support for my spirit and teaches me well. He has been a very caring and helpful supervisor throughout the time of my study in Gunma University and residence in Japan. There is no doubt this is all from ALLAH and I wish I keep my communication with Prof. Yoichi for my future research as well as social activities.

Special thanks extended to the Ministry of Higher Education in **Egypt**, Helwan University and all the staffs of Egyptian Culture Office in Tokyo to support me and my Ph D study in Japan.

Also, I would like to express my thanks to all the examiner committee of my thesis that includes Prof. Wei Shu Gang, Prof. Takeo Ishikawa, Prof. Kou Yamada and Prof. Toshikazu Matsui. Special thanks for Mis. Lida Tohidi, who is my family friend, for assisting in English revision of the written aspect of the thesis as well as my published papers.

I am also really grateful and highly appreciate for the support of all colleagues from Tokyo Sieden Company. A cordial thank you to everyone of Prof. Yoichi laboratory who gave me useful and helpful assistance, support, and cooperation in various ways during my study time. I would also like to express many thanks for all of the Gunma University staff for their help during my study.

I assign my profound and deepest thanks to my wonderful husband, Dr. Aly, for his devotion, support, patience, and encouragement. He provided the suitable atmosphere not only for my study but for my life. I would also like to thank my four wonderful kids, Fatama, Mohamed, Youssef and Ibrahim, who give me the happiness in this life.

EXECUTIVE SUMMARY

Nowadays, an embedded system is becoming a main solution to most specific tasks in industrial and business fields because of its high stability, economic power consumption and usefulness in many fields. Every year, billions of microprocessors are sold for use in embedded systems. This is in sharp contrast to a few hundred million desktop processors that are sold in the same timeframe. From automobiles to medical equipments, thermostats to space shuttles, embedded systems are all around us. Basically the embedded system can be simply defined as a combination of hardware and software that are built into a product for purposes such as control, monitoring and communication without human intervention. Hardware includes microprocessor with additional attached external memory, I/O, and other components such as sensors, keyboard, LEDs, LCDs, and any kind of actuators. Embedded software is the driving force of an embedded system when it is loaded on the system, it will never be changed unless it needs to be reloaded or replaced. Embedded systems are considered one of the most difficult technical and commercial environments because there are many critical systems controlled by embedded system. It includes communication applications, transportation navigation, medical systems and financial systems. Failure or compromise of such systems can have significant consequences including disruption of critical services, financial loss and sometimes loss of life. Subsequently, both quality and performance of such system are considering vital issues.

In developing embedded systems, the requirements for software design are completely different than the case of software design in the business application field. In the embedded software design, ultra high reliability, real time process and hardware/software co-design are required. However, in the development of an embedded system, the amount of software increases explosively in a very short period. The designers of the embedded system face ever increasing challenges in the design stage. One of the difficulties of the embedded system design is that the hardware and the software of an embedded system are developed simultaneously. In the business system development, the hardware on which the software should be executed is available. For example in the business software development, the platform for executing software is a computer incorporating Intel's microprocessor and Windows operating system. However, in the embedded software design even the specifications of hardware usually are not completed embedded software cannot be tested. The embedded system software designer should wait until the hardware

completed, thus software is tested in the actual prototypes. So, revealing the error in the embedded software is considered a critical step. Furthermore, the discovery of errors often resulted in production delay as well as additional expense can be added to the estimated product cost.

One of the major challenges in the design process of embedded systems is to accurately predict performance characteristics of the final system implementation in early design stages. Another challenge of the embedded system design is the system parameters optimization because embedded software controls hardware. Consequently, if the software is not accurately optimized then the embedded system may run out of control. Verification of the embedded system function, which is the process to verify that the system meet the required specification or not, is considered one of the most vital challenges of the embedded system design. In the traditional design method, it is reported that the verification period takes about 50% of the production time. In addition the complexity of the embedded system which arises due to the combination of more and more functions onto a single system. Building on the above descriptions, the complexity of the embedded system design in many applications has been dramatically increasing over the past few years. Therefore, most of researchers directed their attentions to solve some of those challenges. One of the leading techniques in this field is the Model Based Design method. In this method the model can be used to verify the system design and control algorithm. Furthermore, it can be used as executable specification for the designer before the actual implementation. The Model Based Design method is a new approach in the embedded system design and development. It only defines the design methodology and in actual applications, the problem is the definition and representation of each component of embedded system as a model. Once a model is defined in an application, the variations of the model can also be defined by the modification of the original model in other applications based on the reusability of object oriented programming. However, the usefulness of the Model Based Design method has not yet been verified in many applications. The objective of the study is to verify the usefulness, to find out the problems and to check the feasibility of the Model Based Design method.

This research contributes a novel technique that allows the full simulation of the embedded system in the virtual environment based on the Model Based Design method. We used the inverter power supply design as a case study of the application of the Model

Based Design method. Inverter power supplies are widely used in many industrial processes and applications, such as, uninterruptable power supplies, motor control, and electric vehicle applications. It can be defined as a device that converts DC (Direct Current) from sources such as batteries, solar panels, fuel cells, or wind generations to AC (Alternative Current). So, the output can be used in a wide range of AC applications. Traditionally, an analog control technique is used to control the inverter power supply. However, due to the fact that a digital control technique can provide the benefits which cannot be provided by an analog one, a digital control starts to be a viable candidate in the inverter power supply design. Actually, a digital controller can offer a programmable solution for the applications. Moreover, it also offers the flexibility in design and an advanced algorithm as well as additional features can be added to the controlling software instead of hardware. From the electrical point of view, a digital controller is less sensitive to the environmental conditions and shows precise behaviors compared to analog counterparts. In this study the SH7047 microprocessor is used to control the generation of the PWM signal which control the operation of the inverter power supply. To improve the performances of the digital PWM pulses, a digital control technique should be used. The main duties of the inverter control system are to regulate the output voltage against all possible existing disturbances. In this study, to an inverter power supply, a controlling algorithm is newly proposed, which is two layer controller that includes the feedforword and PI controller. To implement a digital control technique in the inverter power supply, an embedded system must be developed. Now, in general, this kind of embedded system is manually developed and the embedded software developing and testing time takes around 50% from the entire developing cycle. Therefore, the Model Based Design method is strong necessary in the design of inverter power supply. The entire inverter power supply system model is constructed under the name of Model In the Loop Simulation (MILS) environment. MILS environment is divided in two main parts which are the controlled part (electric DC/DC and DC/AC circuits) and the control part consisting of the microcontroller and the embedded software. A full simulator of the inverter power supply including electric circuit model and the functional model of the microprocessor is developed by using the MATLAB and Simulink environment. The electric part is developed using the Simulink power block set. The control part is modeled by simulating the function of the microprocessor unit using the S-Function technique and the embedded system software was described as C-MEX files. All parameters and the behavior of the model were tested, optimized and analyzed graphically as well as mathematically. The

validity and usefulness of the MILS environment is verified by comparing the simulated results with the experimental results in the actual device. The actual prototype of inverter power supply has been fabricated using Renesas SH7047 microprocessor and the embedded software is implemented manually. The input DC voltage is 24V and the target AC output is 100 RMSV with frequency of 60 Hz.

The results show that the application of Model Based Design method can contribute to system development period reduction and guarantee robust system design. The developed models helped us to optimize software parameters and to validate control algorithm in the virtual environment. Moreover, the models can be used to study and analyze the behaviors of the system which meet the challenges of designing the embedded control system. The results show that the suggested models are promising and the models can be useful for optimizing the performances in developing the embedded software. On top of that, the Model Based Design method saved the time and cost by large percentage comparing to the traditional method of design.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. GENERAL

This chapter offers a brief introduction of the embedded systems; the difference between the embedded system and the general propose computer, embedded systems classification and the design challenges. Every year, billions of microprocessors are sold for use in embedded systems [1]. This is in sharp contrast to a few hundred million desktop processors that are sold in the same timeframe. From automobiles to medical equipments, thermostats to space shuttles, embedded systems are all around us. If you count how many computers you own or use probably one or two at work and another one or two at home. Now count the number of embedded systems you own or use, a digital cellular telephone, a pager, microwave oven, washer, dryer, dishwasher, coffee maker, refrigerator, VCR, television, video-game console, stereo receiver, CD player, DVD player, portable Discman, remote control for the TV, remote for the VCR, remote for the stereo, garage-door opener, automatic sprinkler timer, fax machine, PDA, answering machine, and so on. The modern automobile has about 100 embedded systems on it. Consequently, that is why embedded system design is a very important research field.

1.2. DEFINITION OF EMBEDDED SYSTEMS

An embedded system can be simply defined as a combination of hardware (microprocessor) and software that is built into a product for purposes such as control, monitoring and communication without human intervention [2, 3]. By other way, embedded system is a special-purpose computing device designed to perform dedicated functions, which consists of hardware and software. The hardware includes a microprocessor or microcontroller with additional attached external memory, I/O, and other components such as sensors, keypad, LEDs, LCDs, and any kind of actuators. The embedded software is the driving force of an embedded system. Once it is loaded it will never be changed unless it needs to be reloaded or replaced [4, 5]. Traditionally most of these systems are used for control and process measurement, as a side-effect of higher integration of integrated circuits more complex applications can be solved by embedded systems. Nowadays embedded systems can be found in devices from digital watch to communication systems, transportation navigation systems, medical systems, and financial

systems. Figure 1.1 shows some examples of the embedded systems applications.

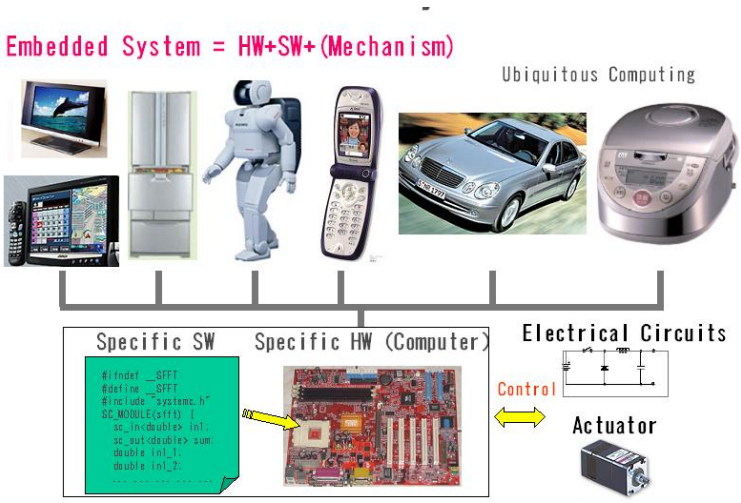


Fig.1. 1. Examples of embedded system

The number of the embedded systems increases rapidly in the last few decades to meet modern life demands. For example in Japan, which is considered one of the most advanced country in the world, there are many products including the embedded system act as main export products such as automobiles, office equipments, copying machines. Table 1.1 shows the main export products in Japan (Japanese Ministry of Economy, Trade and Industry (METI)).

Table 1. 1: Main export products of Japan

	Year	2000	2001	2002	2003
	Total Cost*	516,542	489,792	521,080	545,484
1	Products Amount Ratio %	Automobile 69,301 13.4	Automobile 72,108 14.7	Automobile 87,746 16.8	Automobile 88,756 16.3
2	Products Amount Ratio %	Elec. Devices 45,758 8.9	Elec. Devices 36,474 7.4	Elec. Devices 38,673 7.4	Elec. Devices 40,745 7.5
3	Products Amount Ratio	Office Equip 30,942 6.0	Office Equip 28,207 5.8	Office Equip 30,053 5.8	Office Equip 26,191 4.8
4	Products Amount Ratio	Copy Machine 26,256 5.1	Copy Machine 25,045 5.1	Copy Machine 21,171 4.1	Copy Machine 20,660 3.8

Note: * hundred million yen

However, in the development of an embedded system, the amount of software increases explosively in a very short period as shown in Figure 1.2 and this causes a very serious problem in the embedded system industry. Japanese METI (Ministry of Economy, Trade and Industry) continues the investigations of the embedded system industry from 2004 and published the reports. From the latest report, the deficiency of embedded software designers reaches to about 94,000 in 2007. In the software development, the increase of the amount of lines of codes in an automobiles, mobiles phones, and digital home appliances makes the complexity of software design very much increased and the supply of embedded software engineers can not satisfy the needs of the current deficiency. Therefore, many studies are needed to overcome such problems.

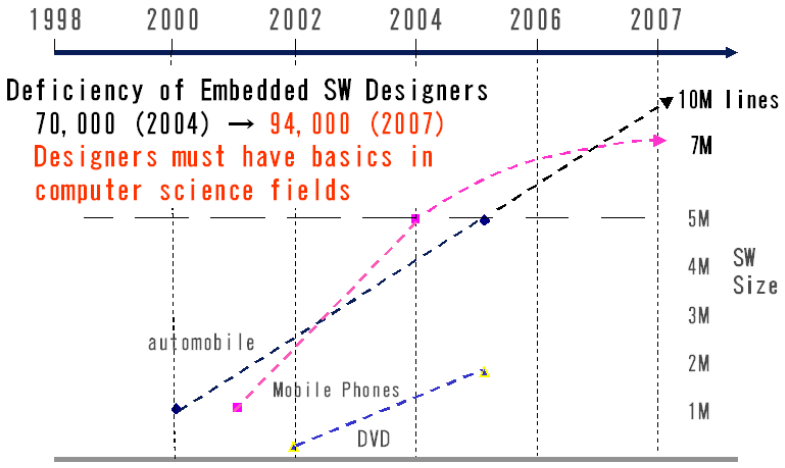


Fig.1. 2. Explosive increase in embedded software size

1.3. EMBEDDED SYSTEM VERSUS GERNAL PURPOSE SYSTEM

An embedded system is usually classified as a system that has a set of predefined, specific functions to be performed and in which the resources are constrained [6]. For example cellular phones, it is an embedded system and it has several readily apparent functions as follows: the main function is to call and receive phone call perhaps several functions such as clock, alarm, and camera and so on. It also has several resource constraints as follows: Firstly, the processor that is operating the mobile phone cannot be very large, or else no one would use it. Secondly, the power consumption must be minimal, only a small battery can be contained in that cellular phone. Finally, it must perform its Function accurately. Each embedded system design satisfies its own set of functions and constraints. By comparing this example with the general propose computer we can see the differences. In this section the main differences between both systems are described below:

1. Embedded systems are dedicated to specific tasks, where PCs are general computing platforms. Embedded system is programmed to perform specific tasks conversely. General computer is able to perform unlimited tasks or a general-purpose computer, for example, you can install any software to do all kinds of jobs such as word processing, data sheet, database management, and others depending on your purposes.
2. Embedded systems are usually cost sensitive because the embedded system is only part of the whole product. Subsequently, if the cost of the embedded system reduces you can potentially reduce the product cost.
3. The implication of software failure are much more severe in the embedded systems than the general propose computer. It is considered one of the most difficult technical and commercial environments because many critical systems are controlled by embedded computer system. These include communication systems, transportation navigation systems, medical systems, and financial systems. Failure or compromise of such system can have significant consequences including disruption of critical services, financial loss, and loss of life.
4. Embedded systems have power constrains. This is not practically serious constraint for the general computer system. However, consider an embedded system connected to medical system in the ambulance, so the system must work reliably and for long time in a set of small batteries. So, it is very important issue to keep the embedded system running on minute amount of power.
5. Embedded systems have real time constrains. Real time constrains generally are grouped into categories depending on the application as follows: the first category is time sensitive constraints and; the second category is time critical constraints. If the application has time critical constraints the task must take place within a set window of time, controlling the flight worthiness of an aircraft is a good example of this.
6. Embedded systems must operate under extreme environmental conditions. The embedded systems are everywhere, so the system must be designed to work well in different environment conditions and in the harsh environment as well.
7. Embedded systems microprocessors often have debugging circuitry.

1.4. EMEDDED SYSTEM DESGIN LIFE CYCLE

Aforementioned, the embedded system is a system which designed to perform a dedicated function, typically with tight real-time constraints, limited dimensions, and low cost and low-power consumption requirements. It is a combination of computer hardware and software and additional mechanical, optical, or other parts that are typically used in the specific role of actuators, sensors, and transducers [5, 7, 8]. In general the design life cycle of the embedded systems is unlike the design life cycle for the standard platform. The traditional design cycle of the embedded system can be summarized in the following phases [5].

1. Production specification,
2. Partitioning of the design into its software and hardware components,
3. Iteration and refinement of the partitioning,
4. Independent hardware and software design tasks,
5. Integration of the hardware and software components,
6. Product testing and release,
7. Ongoing maintenance and upgrading.

The percentage of the project time spent in each stage of the embedded system design life cycle is shown in Figure 1.3.

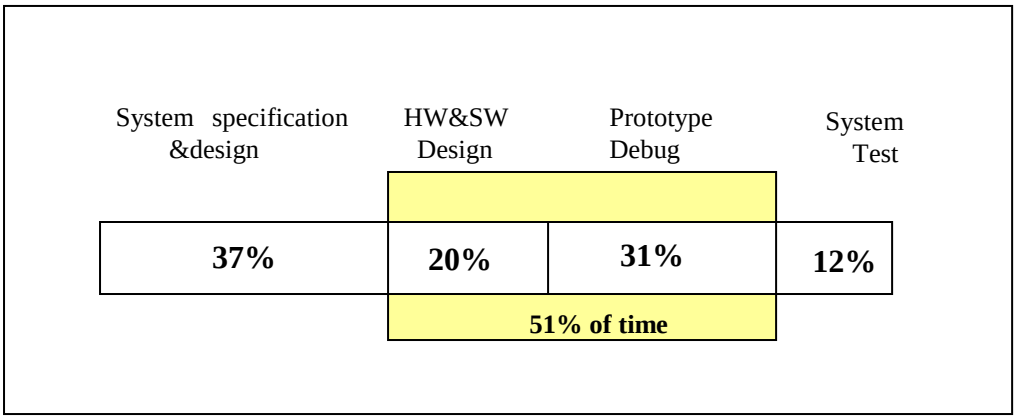


Fig.1. 3. Embedded system life cycle

In general, in the business software development, the platform for executing and testing the software computer is an easy task. However in the embedded system, an embedded system cannot be tested in the actual hardware. The software designer needs to wait until the hardware is completed then they can test the software which increases the product cost and time as well as it will affect the system quality. So, the developing cycle must have

special tools and method to manage the complexity of the design. Thus, there are many challenges face the embedded system design which are described in details in the next chapter.

1.5. CLASSIFICATION OF EMBEDED SYSTEMS

Embedded systems can be classified into different categories based on its Function, the required performance of the application and the size of the system. The classification presented below in details.

1.5.1 Classification Based of the Function and Required Performance

An embedded system is becoming a main solution to most specific tasks because of its high stability, economic power consumption and usefulness. It is widely used in a range of applications in our life. The function of the embedded system varies from application to other application. The functions of the embedded system can be classified as follows:

- Stand-alone embedded systems

  In this category the embedded system works by itself and it is a self-contained device. It takes either digital or analog inputs from its input ports then it processes this data to get specific output. Then it outputs the resulting data to its attached output device, which either displays data, or controls and drives the attached devices. Entertainment devices such as video game console and MP3 players, digital cameras, and microwaves are typical systems that fall into this category [9].

- Real-time embedded systems

  In this category, the time is considered as a critical factor. In other words, some specific functions must be done in particular period. There are two types of real-time embedded systems hard real-time and soft real-time embedded systems. In hard real-time systems if the task is completed after the deadline it may lead to critical failure and in some case it may result in loss of life. For example, a car airbag control system, if any delayed reaction in this system can cause severe results. The response time deadline for the hard real time system is very critical (in millisecond or even shorter). It is very important for such system to react to an event within a strict deadline, and missing a deadline will constitute failure of the system. The hardware and software of hard real-time systems must allow a worst case execution (WCET) analysis that guarantees the execution be completed within a strict deadline [10,11,12]. The second group for the real time embedded system is the soft real-time system, in this system it

can tolerate with some degrees of the time missing. In this system, if the task is completed after the deadline, the system can continue work but only the quality of the system will reduce. For example the microwaves and the washing machines, in this example, the time for completing the task can be delaying with second without causing any critical problems .

- Networked embedded systems

  The networked embedded system is considered one of the fastest growing areas in embedded systems applications. The networked embedded systems connect to a network with network interfaces to access resources. The connected network can be Local Area Network (LAN), Wide Area Network (WAN), or the Internet. The connection can be wired or wireless [13]. Home security systems are one example of this group.

1.5.2. Classification Based on the Size

For different applications the scale of the embedded systems are varied. We can classify the embedded system based on the size as follows:

- Small scale embedded system is designed using a single 8 or 16 bit microcontroller. Most of the small scale embedded systems are battery operated. Usually, 'C' language is used for developing these systems. 'C' program compilation is done into the assembly, and executable codes are then appropriately located in the system memory [10,11]. The software has to fit within the memory available and keep in view the need to limit power dissipation when the system is running continuously.
- Medium Scale Embedded Systems:

  These systems are usually designed with a single or few 16 or 32 bit microcontrollers or Digital Single Processors (DSPs) or Reduced Instruction Set Computers (RISCs). These have both hardware and software complexities. For complex software design, there are the following programming tools: Real Time Operating System (RTOS), Source code engineering tool, Simulator, Debugger and Integrated Development Environment (IDE). Software tools also provide the solutions to the hardware complexities. An assembler is of a little use as a programming tool.
- Sophisticated Embedded Systems:

  Sophisticated embedded systems have enormous hardware and software complexities and may need scalable processors or configurable processors and programmable logic arrays. They are used for cutting edge applications that need hardware and software

co-design and integration in the final system. However, they are constrained by the processing speeds available in their hardware units. Certain software functions such as encryption and deciphering algorithms, discrete cosine transformation and inverse transformation algorithms, TCP/IP protocol stacking and network driver functions are implemented in the hardware to obtain additional speeds by saving time. Some of the functions of the hardware resources in the system are also implemented by the software. Development tools for these systems may not be readily available at a reasonable cost or may not be available at all [10,11].

1.6. SCOPE OF RESEARCH

Embedded systems are considered one of the most difficult technical and commercial environments because many critical systems are controlled by embedded system including communication applications, transportation navigation, medical systems and financial systems. Failure or compromise of such systems can have significant consequences including disruption of critical services, financial loss and sometimes loss of life so the quality and the performance of such system are considering vital issues [8]. Building on this the focus of most researchers nowadays is directed to solve the challenges which face the embedded system. These challenges presented below:

1. The complexity of the embedded system, which arises due to the combination of more and more functions onto a single system.
2. The optimization: the software and the hardware parameter of the embedded system have to be optimized in very accurate way and also have to be on time.
3. Verification of the embedded system function, which is the process verify that the system meet the required specification or not. In the traditional design method it is reported that the verification period takes about 50% of the production time.

Many of researchers nowadays try to develop new technique to overcome such challenges. One of the leading techniques in this field is the Model Based Design method. This study will test how the model based design put a good solution for the embedded system challenges. In this research, one case study was taken as an embedded system example which is the inverter power supply. Due the fact that a digital controller can provide more benefits which cannot be provided by the analog one in the inverter power supply applications, the digital control starts to be used. However, developing the inverter power supply based on the digital control faces some challenges which listed below:

1. It is so difficult to monitor the control algorithm behavior before the actual implementation.
2. The embedded software which controls the operation of the inverter power supply is developed manually which takes much time and bugs in such program are inevitable.
3. Bugs discovery is considered a critical problem and it results in production delay time and additional cost will be added.
4. Within the scope of our investigation, all the previous study concentrated on the modeling of the analog part and the digital pulse was generated by pulse generator. So, in the actual implementation, the software is needed to be tested in the actual prototype which may lead to system failure as well as it takes so much time to obtain the optimum parameters. And when any change occurred in the embedded software, the verification process is needed to start from the beginning. It is takes long time to optimize the software parameters using the conventional design method.
5. It is reported that the verification of the system performance and the control algorithm for the embedded system takes from 50 to 70 % of the production time.

So, in this research, entire virtual environment for the inverter power supply was constructed based on Model Based Design technique. The entire embedded system model is constructed under the name of Model In the Loop Simulation (MILS) environment. The embedded software is designed and optimized in the virtual environment using the MATLAB and Simulink environment. Newly applied controlling algorithm is designed and tested in the virtual system; this controlling algorithm consists of PI controller and the feedforward controller. Then to verify the validity of the proposed MILS environment, and to verify the control algorithm, a comparison between the virtual environment and conventional design method will be presented.

1.7. PURPOSE OF RESEARCH

In developing embedded systems, the requirements for software design are completely different in the case of software design as in the business application fields. In the embedded software design, ultra high reliability, real time process and hardware/software

co-design are required. Because of embedded software controls hardware, if the software is not accurately optimized then the embedded system runs out of control. One of the difficulties of embedded system design is that the hardware and the software of an embedded system are developed simultaneously. In the business system development, the hardware on which the software should be executed is available as shown in Figure 1. 4. For example, in general, in the business software development, the platform for executing software is a computer incorporating Intel's microprocessor and Windows operating system. However, in the embedded software design, even the specifications of hardware usually are not completed. Therefore, embedded software cannot test on the actual platform until the completion of the hardware. This will cause the degradations of qualities of embedded software.

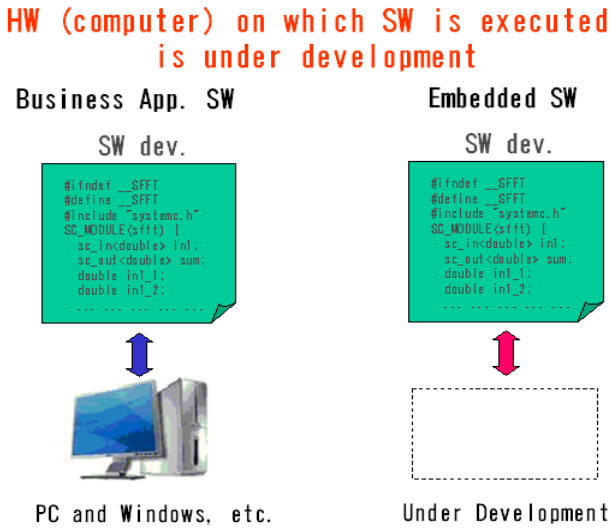


Fig.1. 4. Difficulty of embedded software design

In the Model Based Design (MBD) method the model acts as the heart of the developing process. This study aims to develop a virtual environment that can be used to test and optimize the embedded software parameters before the actual implementation. This environment will save the time and cost comparing to the traditional design method as well as reduces the possibility of any damage which can be occurred, when the software is tested and optimized in the actual prototype as in the traditional methods. We used the inverter power supply as case study of the embedded system. Traditionally, an analog control technique is used to control the inverter power supply. However, due to the fact that a digital control technique can provide the benefits which cannot be provided by an analog one, a digital control starts to be used [14, 15, 16, 17]. We used the SH microprocessor as a digital controller for the inverter power supply circuits.

This study proposes a new virtual environment to test and optimize the software parameters in the virtual environment before the actual implementation. This environment called the Model In the Loop Simulation (MILS). This virtual environment is divided into two main parts, the control part and the plant part. MATLAB and Simulink software package is used in the developing of this virtual environment. The functional model of the microprocessor is developed using the S-Function and C-MEX files. All the software parameters used in this model are optimized and tested. Such system needs control algorithm to be stabilized. So, this study proposes a newly applied two layer controlling algorithm including the PI controller plus feedforward control. The validity of the controlling algorithm is also tested in the virtual environment (MILS). Finally, the optimized software parameters and the proposed controlling algorithm are tested on the actual inverter power supply prototype which has been fabricated using the Renesas SH microcontroller.

1.8. THESIS OUTLINES

This thesis consisted of seven chapters as presented below:

Chapter 1 presents an introduction about the embedded system design and its challenges. This chapter also describes a brief background of the embedded system and the main difference between the embedded system design and the general propose computer system design. It also describes the design life cycle of the embedded system and finally some embedded system design challenges are discussed.

Chapter 2 presents the embedded system design tools. Model Based Design (MBD) technique and the advantages of using it in the early stage of design are described in this chapter. Also, MBD concept and methodology are presented in this chapter

Chapter 3 presents the description of the inverter power supply as an example of the embedded system and topologies of the inverter power supply design is presented. The digital control technique and its advantages and the methodology of design are also described in this chapter. As well as, the controlling algorithms which we proposed are described in this chapter.

Chapter 4 presents the description of the proposed environment of design which is the MILS of the inverter power supply under the MATLAB and Simulink environment.

Chapter 5 presents results of both experimental and simulation environments. The investigation of performances of inverter power supply as well as the verification of the proposed control algorithm is shown. Finally, the usefulness of the virtual environment model is verified.

Chapter 6 contains the main and specific conclusions obtained from this study and also highlights recommendations for future studies.

REFERENCES

[1] J. Turley. “Embedded processors by the numbers”, Website, Accessed in October, 2010, http://vault.embedded.com/1999/9905/9905turley.htm.

[2] T. Henzinger and J. Sifakis, “The Embedded System Design Challenge”, In the Proceedings of the 14th International Symposium on Formal Methods (FM), May 2006.

[3] W. Hongxing and W. Tianmiao, “Curriculum of Embedded System for Software Colleges”, In the Proceedings of the 2nd IEEE, August 2006.

[4] B. Thomas, “Embedded Robotics: Mobile Robot Design and Application with Embedded system”, Springer- Verlag Hiedelerg Ltd. Co., 2006.

[5] A. S. Berger, “Embedded Systems Design: An Introduction to Processes, Tools & Techniques”, Group West Publishers, US, 2002.

[6] D. Stepner , N. Rajan and D. Hui, “Embedded Application Design Using a Real-time OS”, In proceeding of 36th Design Automation Conference, New Orleans, USA, pp.151-156, 21-25 Jun, 1999.

[7] P. J. Mosterman, “Model Based Design for the Embedded System”, Taylor & Franicis Group, LLC, Ltd. Co., 2010.

[8] M. A. El Dahb, S. Iino, Y. Shiraishi and M. Tatsuno, “Model Based Design of the Inverter Power Supply”, In the Proceeding of ICCAS-SICE 2009 International Conference, August 17-21, 2009.

[9] K. Qian, D. D. Haring and L. Cao, “Embedded Software Development with C”, Springer Dardrecht, London New York, Ltd. Co., 2009.

[10] L. Brush, “Trends in Digital Power Management: Power Converter and System Demand Characteristics”, In the Proceedings of the Twentieth annual IEEE Applied Power Electronic Conference, 2005.

[11] T. Wikmshurst, “An Introduction to the Design of Small Scale Embedded systems”, Palgrave Macmillan Co., Ltd., 2001.

[12] Q. Li and C. Yao, “Real Time Concept for Embedded System Design”, CMO Media LLC, Group West Co., Ltd., 2003.

[13] J. Gregory, P. William and J. Kaiser, “Principle of Embedded Networks Systems Design”, ISBN Publisher Co., Ltd., California, US, January, 2009.

[14] M. Trigg, H. Dehbonei and C. V. Nayar, “Digital Sinusoidal PWMs for a Micro-Controller Based Single-phase Inverter, Part1: Principles of Digital Sinusoidal PWM

Generation", International Journal of Electronics, Vol.95, No.8. pp.819-840, August, 2008.

[15] Y. Xue, K. Baekhy and J. Bordonau, "Topologies of Single Phase Inverter for Small Distributed Power Generators: An Overview", IEEE Transactions, Vol.19, No.5, pp.1305-1314, Sept., 2004.

[16] O. Pop, G. Chindris and A. Dulf, "Using DSP Technology for True Sine PWM Generators for Power Inverters", In the Proceedings of the 27th International Spring Seminar, 13-16 May, 2004.

[17] C. Matthew, D. Hooman and C. V. Nayar, "Digital Sinusoidal PWM Generation Using a Low–Cost Micro-Controller Based Single–Phase Inverter", IEEE Transactions, Vol.1, pp.390-396, September, 2005.

CHAPTER 2
MODEL BASED DEVELOPMENT

2.1. INTRODUCTION

Given competitive temporal and cost constrains, developing a product on time and within budget requires a systematic approach to design and realization. A systematic approach ensures that the final products meet the initial requirements and let engineering teams with different specialization work together and communicate between the stages in the overall processes. In addition the approach also ensures that the design process and the final product are documented for maintenance and future development. In the traditional approach, engineering teams observe strict boundaries between their design activities. Furthermore, they transfer the data by passing design documents back and forth. This approach has the following drawbacks:

1. Documents can be unwieldy and unsuitable for recording functionality of the system.
2. It is difficult to keep the documentation synchronized with the current state of design.
3. Once the design is approved, coding the application becomes a separate, manual activity.
4. When the documents are used as deliverable and shared electronically, engineers often duplicate their efforts. It is difficult to trace the source of error along a paper trail. So it is considered as a waste of time and increases the product cost.

So, many researches now try to solve these problems by using computer aided design techniques which can be defined as the use of computer systems to assist in the creation, modification, analysis, and optimization of a design. There are many critical systems controlled by embedded system including communication applications, transportation navigation, medical systems and financial systems. Subsequently, embedded systems are considered one of the most difficult technical and commercial environments. Failure or error of such systems may lead to significant consequences including disruption of critical services, financial loss and sometimes loss of life. Thus, the quality and the performance of such systems are considering vital issues. So, the design method of the embedded system should be selected very carefully to meet the required specifications. But up to now, there are many challenges facing the embedded system development and some of these challenges are described in the next section.

2.2. CHALLENGES OF THE EMBEDDED SYSTEM DESIGN

Embedded system development tools have traditionally lagged behind tools for the development of general systems [1]. Unlike general systems, the design space for embedded systems is extremely large, so it is difficult to contain all of the facilities to specify, design, and test embedded systems. The number of embedded systems increased rapidly year by year and then the designers of the embedded system face ever increasing challenges in the design stage. Some of these challenges are listed below:

1. In the traditional design method of the embedded system, the hardware and the software of an embedded system are developed simultaneously which is a sharp contrast of the general or business system. In the business system development, the hardware on which the software should be executed is available. For example, in general, in the business software development, the software can be tested and modified in the computer. However, in the embedded software design, even the specifications of hardware usually are not completed. Therefore, embedded software cannot test on the actual platform until the completion of the hardware. This will cause the degradations of qualities of embedded software as well as increasing the process time [2].
2. In traditional development method, design took places early in the development process, and the software designer should wait until late in the process. Then the embedded software is tested in the actual prototype. So revealing the errors in the embedded software is considered a critical step. Furthermore, the discovery of errors often resulted in production delay as well as additional expense can be added to the product cost [2, 3].
3. One of the major challenges in the design process of embedded systems is to accurately predict performance characteristics of the final system implementation in the early design stages.
4. The complexity of the embedded system arises due to the combination of more and more functions onto a single system [2, 4]. For example, luxury vehicles produced today contains more than 90 embedded electronic control units (ECU), which execute more than 10 million lines of computer codes which control many different functions in a car [5]. Increase of system complexity may lead to the increase of a project time and the system design cost.
5. As mentioned above, the main difference between the conventional computer

systems and the embedded computer systems is that the hardware on which the software should be executed is unavailable. Therefore, a simulator of the embedded hardware is very useful for the development of the embedded software but it becomes a difficult matter because of the length of time required to build a simulator [2-4].

6. Verification is the process of determining whether a system satisfies a given property of interest or not. It is considered one of the most difficult challenges of the embedded system. Due to the fact that the embedded systems deal with very critical applications, a designer has to make sure that the system meets the required specification perfectly. It is reported that the verification takes more than 50 to 70% of the project time and the cost for verification of the embedded system is increasing rapidly [1].
7. As a result, the competition between the companies to deliver the product to the market faster and with lower cost, in most cases, the embedded system is a part of much larger product. Thus, any delay in the embedded system development will causes overall delay in the project or products.

However, nowadays embedded systems have garnered more interest in the research community, as well as there being an increased need for those embedded systems. Increasing the embedded system challenges open a wide range of research in the developing tools. Most of researchers try to find design tools that can solve some of the embedded system challenges mentioned above. This chapter presents some of the popular embedded systems design tools.

2.3 HARDWARE / SOFWARE CO-DESGIN

One of the methodologies gained a wide acceptance in both the embedded world and the general purpose world is that of Hardware/Software Co-design. Figure 2.1 presents the description of the Hardware/Software co-design technique.

2.3.1 Hardware / Software Concept

In the traditional design method, the designers have to portion the system into hardware and software parts and each part is developed separately [6, 7]. The hardware designer usually makes the architecture based on the knowledge of the hardware requirements. While the software designer faces many difficulties to fix the software due to the shortage of knowledge and software understandings. The result is that often the software designers are forced to make up for problems in the hardware through additional work of the

software, often leading to a less than optimal overall design of the system. The concept of Hardware/Software Co-design is that of both hardware and software designers worked in parallel to develop a system [7]. The designers define requirements and create a working specification. Then the hardware and software designers work together to map this specification on hardware and software architectures as shown is Figure 2. 2.

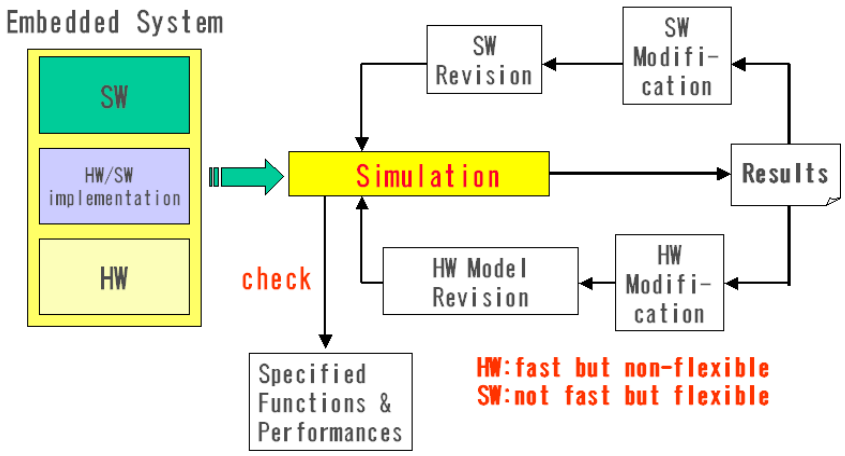


Fig.2. 1. Hardware/Software co-design technique

In this method the software design should wait until the hardware specification is fixed then they can test software. So, revealing the errors in the embedded software is considered critical step. In addition to the discovery of errors, it is often resulted in production delay as well as additional expense can be added to the product cost. Recently, many researchers try to find new methods that can reduce the suspension time for the software designer as shown in Figure 2. 2.

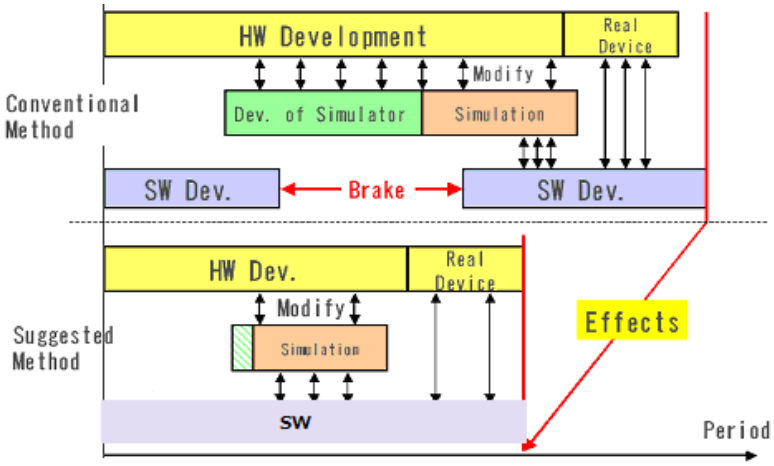


Fig.2. 2. Design processes of the Hardware/Software co-design

2.4. MODEL BASED DESGIN

Currently, many researches are focused on determining a good way to eliminate the challenges of the embedded systems design. Embedded systems have particularly tight performance, time to market, and cost constraint. To meet these constraints, researchers try to find solutions to efficiently design the systems with required performances. Recently, Model Based Design method is considered as one of the chief technique in this field. The Model Based Design method is considered one of the leading techniques as a solution of those challenges. In this method, the model can be used to verify the plant design and the control algorithm. MBD method puts a system model at the center of the development process, from requirement development through design implementation. Figure 2. 3 shows an example of the embedded control V Diagram which is often used to describe the development cycle of the embedded systems. Several versions of these diagrams can be found to describe variety of product design cycle. The figure from left to right describes the general propagation of the development steps. The main target of the Model Based Design method is to improve the development cycle by minimizing the iterations required for the design. If we consider the X axis of the diagram to represent the time, so the main goal is to narrow the V shape as much as possible by drawing the two legs of the diagram closer [8].

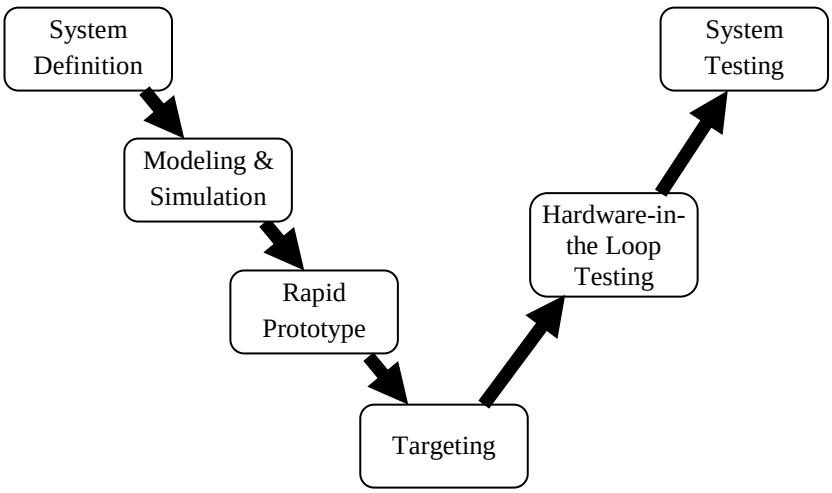


Fig.2. 3. Developing cycle of the embedded system

The system specifications phase is started by the analysis and the documentation of the system requirements. In the traditional method, this step is accomplished using paper based method which results in poor communication in the design process, errors in the design as well as limited traceability between the design and the requirements. In the

MBD method, the model can provide an excellent virtual environment for high level descriptions of the embedded system as shown in Figure 2. 4. This figure illustrates the main four elements of the MBD method. Description for elements of Model Based Design method is presented below.

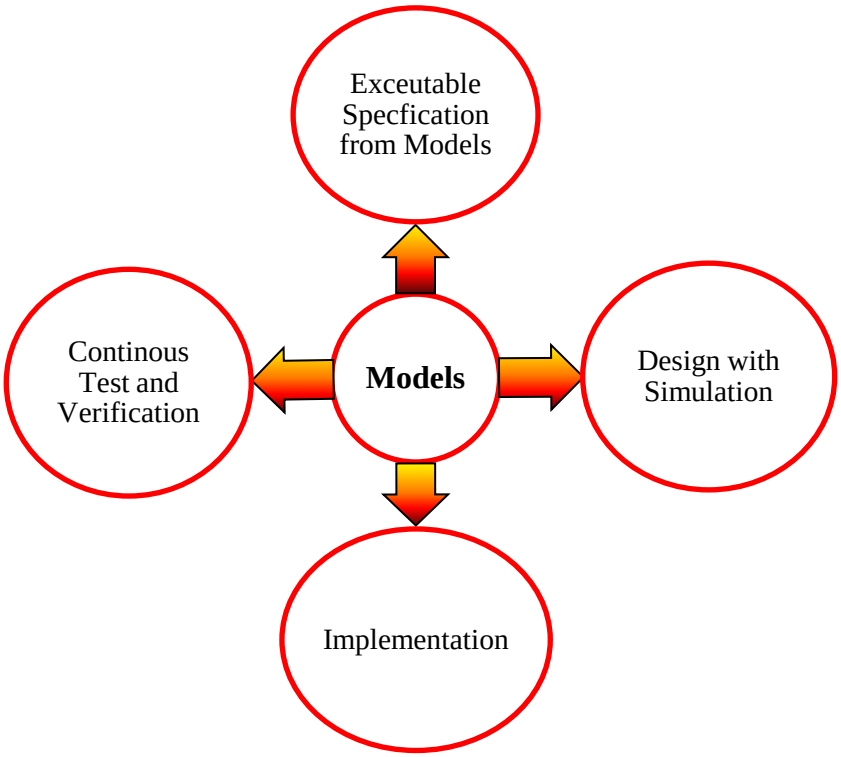


Fig.2. 4. Elements of Model Based Design

2.4.1. Executable Specifications

As designs become larger and more complicated, it becomes necessary to first describe them at a high level of abstraction. For example, Simulink can provide specific blocksets such as signal processing, communication, video and image processing block set to help the designer to build abstraction model. This model provides a documented method for verifying and validating of design prior to move the development in actual controllers and hardware [8,9]. System engineers usually develop this high-level description for several purposes as listed below:

1. It enables designers to perform simulations by directly executing the model.
2. It is used throughout the development process for testing, verification, and implementation.
3. It allows for developers to identify bugs early on and avoid costly bug discovery towards the end of development.

4. It eliminates the need for paper-based specifications, which is easily prone to misinterpretations, and replaces it with the executable specification.
5. Each member of a design team can understand and execute the model and can focus in developing parts of the main model [10,11].

A key point of shrinking the V embedded diagram by applying the MBD method is to begin developing the embedded control algorithm as early in the design cycle. In this stage, the model provides the ability to begin simulation of the control behaviors while the hardware prototype is still under development. In addition, the model can be reused for further modification of the same product which reduces the effort necessary to build the model again.

2.4.2. Design with Simulation

When designing the executable specification, the system engineer generally does not keep the implementation details in mind, but rather designs the algorithm to match the behavioral requirements for the system. Once the system engineer submits the executable specification to the development team, the team may need to make modifications to it in order to fit the design into a real world that may have limited resources, such as memory or processing power. These modifications may cause the output of the new design to deviate from the original design. Design engineers should decide if the deviation is acceptable. In this section, modifications to the algorithm will be done to make it suitable for hardware implementation and demonstrate how to continuously verify the design against the executable specifications. For example, if the designers need to change the controlling algorithm to meet the requirements, MBD method provides an environment where the designer can redesign the control algorithms and validate it in very short time comparing to traditional method of design [11].

2.4.3. Implementation and Testing

The modern Model Based Design tools provide automatic generation for both prototype and production codes directly from the model. So, all the design changes automatically flow through the final implementation. This process results in significant time and cost saving due to the inherent reproducibility and testability of the generated codes and elimination of communication errors [11,12]**.** In the Hardware in The Loop (HIL) Testing, the designer can test the real time behaviors and characteristics of the final system to verify the system control without the need for the physical hardware or operational

environment as shown in Figure 2. 5. HIL Testing can save the time with significant ratio comparing to the traditional design method. As well as, it is easy to implement comparing to physical prototype production. Up to this moment, the Model Based Design process does not completely eliminate the need for testing in the actual prototype, but it offers several opportunities to reduce the time needed to the testing stage [13,14,15,16].

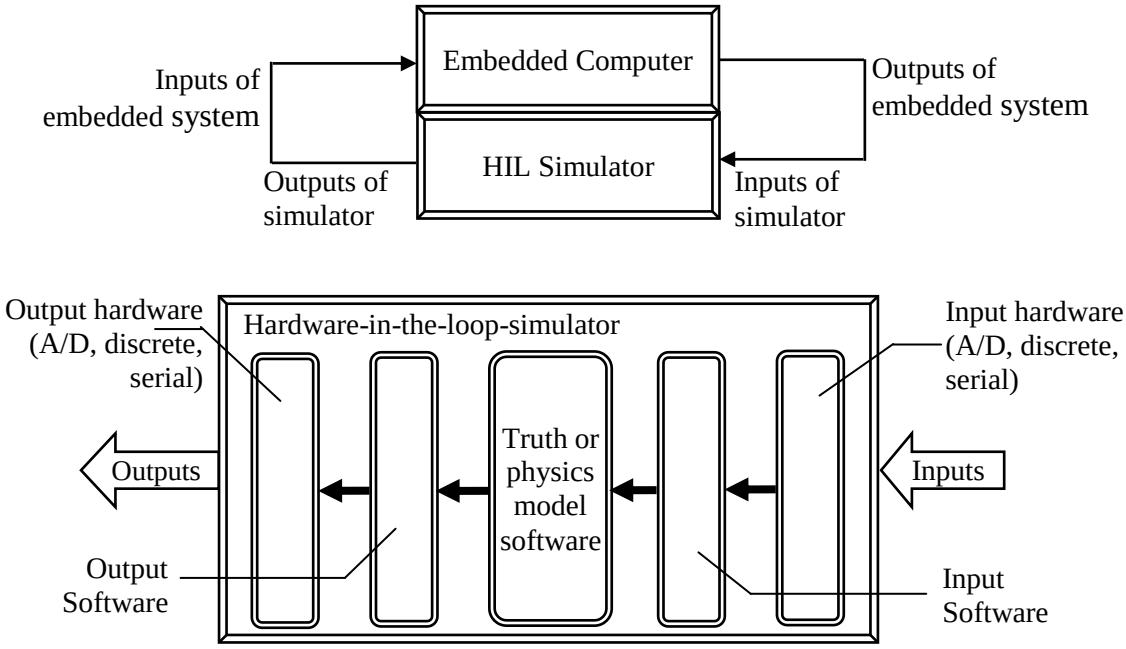


Fig.2. 5. Hardware In the Loop Testing

Generally, it can conclude that the Model Based Design method will reduce the number of the development stages by combining the design, implementation, and testing into one process. The reduction of the required step comparing to the traditional method of design will result in better project management and mitigate the system development risk. The system design using this approach will reach the market faster and the end up costing will less than that of the system designed using the traditional method. Subsequently, the use of the Model Based Design method can provide numerous advantages over the traditional design method. Therefore, this study investigates how the Model Based Design method can provide such advantages by applying and building a new virtual environment for the embedded system design. Inverter power supply is used as a case study of the embedded system in this research.

REFERENCES

[1] M. Charistopher, "An Evaluation of Embedded System Behavior". M.Sc Thesis, Department of Electrical and Computing Engineering, University of Maryland, 2000.

[2] V. Woodward and J. Mosterman, "Challenges for Embedded Software Development". In the Proceeding of Circuit and System, MWSCAS Conference, pp.630-633, 5-8 August, 2007.

[3] B. Manfred, "Challenges in Automotive Software Engineering", In the Proceedings of ICSE'06 Conference, May20-28th, 2006.

[4] A. Madhukar and I. Lee, "Challenges and Opportunities in Deeply Embedded System Security", ACM SIGBED Journal, New York, USA, Vol. 5, January, 2008.

[5] L. Ming-Shan, "Application of Embedded System in Construction Machinery", In the Proceedings of the 8th ACIS international Conference, 30th July-1st August, 2007.

[6] M. A. El Dahb, S. Iino, Y. Shiraishi and M. Tatsuno, "Model Based Design of Inverter Power Supply", In the Proceeding of ICCAS-SICE International Joint Conference, Fukouka, Japan, August 17-21, 2009.

[7] J. Hennessy and M. Heinrich, "Hardware/Software Co-design of Processors Concept Examples", Lecture Notes Presented at Advanced Study Institute (ASI), Temezzo, Italy, June, 1995.

[8] J. Rautio, "Shortening the Design Cycle", Microwave Magazine, IEEE, Vol.9, No.6, pp.86-96, Dec., 2008.

[9] D. Stepner, N. Rajac and D. Hui, "Embedded Application Design Using a Real-time OS", In proceeding of the 36th of Design Automation Conference, New Orleans UAS, pp. 151-156, 21-25 June 1999.

[10] P. J. Mosterman, "Model Based Design for Embedded System", Taylor &Francis Group Co., Ltd., New York, 2010.

[11] A. Behboodian, "Model Based Design", DSP Magazine, Vol.2, pp.52-56, May, 2006.

[12] C. Davey and J. Friedman, "Software Engineering with Model-Based Design", In the Proceedings of the Fourth International Workshop on Software Engineering for Automotive System, 2007.

[13] K. I. El-Far and J. A. Whittaker, "Model Based Software Testing", Encyclopedia on Software Engineering, 2001.

[14] T. Erkkinen, “Automatic Flight Code Generation with Integrated Static Run Time Error Checking and Code Analysis”, In the Proceedings of AIAA Modelling and Simulation Conference and Exhibit, Colorado, August 21-24th, 2006.

[15] J. Zhenhua, R. A. Dougl and R. Leonard, “Hardware In the Loop Testing of Digital Power Controllers”, In the Proceeding of Applied Power Electronic Conference, April 18th, 2006.

[16] X Wu, H. Fifueroa and A. Mont, “Testing of Digital Controller Using Real Time Hardware in the Loop Simulation”, In the Proceeding of Power Electronic Specialists Conference, IEEE, Vol.5 , pp.3622-3627, April, 2006.

CHAPTER 3
INVERTER POWER SUPPLY DESGIN

3.1. INTRODUCTION

The development of new useful energy sources is the main key to continued industrial progress, and the continual improvement in the world's standard of living. Discovering new sources of energy, obtaining an essentially inexhaustible energy source, making it available everywhere, and converting it from one form to another without polluting and destroying the environment are some of the great challenges in the world today. One of these energy sources is to produces AC (Alternative Current) from fuel cell or solar panels output this device is called the inverter power supply system. In general, power conversion is the process which converts the electric current from AC or DC (Direct Current) power to provide a different electrical waveform. The term "converter" denotes a mechanism for either processing AC power into DC power (rectifier) or deriving power with an AC waveform from DC (inverter). Some converters serve both functions, others only one. Converters are used for such applications as follows:

- Rectification from AC to supply electrochemical processes with large controlled levels of DC.
- Rectification of AC to DC followed by inversion to a controlled frequency of AC to supply variable-speed AC motors.
- Interfacing DC power sources (such as fuel cells and photoelectric devices) to AC distribution systems.
- Production of DC from AC power for subway and streetcar systems, and for controlled DC voltage for speed-control of DC motors in numerous industrial applications.
- Transmission of DC electric power between rectifier stations and inverter stations within AC generation and transmission networks.

Table 3.1 illustrates the available power conversion devices

Table 3. 1: Types of power conversion

Conversion work	Name of equipment
AC→DC	Rectifier
DC→AC	Inverter
DC→DC	Converter
AC→AC	Ac power regulator (transformer)

In this study DC/AC conversion is investigated through the inverter power supply. Inverter power supplies are widely used in various kinds of applications, such as motor driver controllers [1, 2, 3], uninterruptible power supplies [4, 5], audio power amplifiers [6], and many other applications. It is a device that converts DC from sources such as batteries, solar panels, fuel cells, or wind generations to AC and then the output can be used in a wide range of AC applications [7]. The first generation of the inverter power supplies invented to get a square wave output. The use of such kind of inverter leads to many problems involving the functionality of the device that were being powered because those devices were designed to work with a sine wave source. Furthermore, the harsh edge of the square ware can cause some disturbances in the devices. Some hardware modifications were done in the square wave inverter to produce the modified square wave inverter. The modified square wave inverter provides a cheap and easy solution to powering the AC devices. But it has some drawbacks as it reduces the problem of the harsh edge of the square wave and it is not eliminating it. Moreover, not all the AC devices can work properly with such kind of inverter. A quantum leap occurs in an inverter power supply when it becomes possible to use the Pulse Width Modulation (PWM) technique to get the pure sine wave output. The PWM method allows for filtering undesirable harmonics in the output signals that is not possible in either square wave output or modified square wave output. Figure 3.1 shows the output signal for the square, modified square and pure sine wave output. Usually sine wave inverters are more expensive then modified sine wave generators due to the added circuitry. However, this cost is made up for in its ability to provide power to all AC electronic devices, to allow inductive loads to run faster and quieter, and to reduce the audible and electric noise in audio equipments.

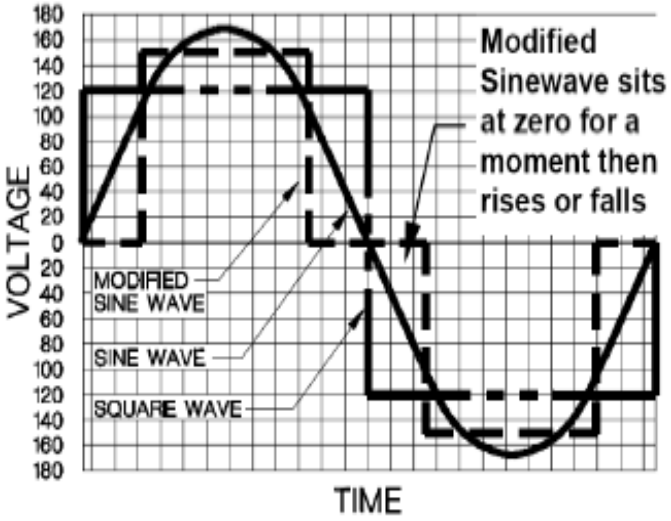


Fig.3. 1. Square, modified and pure sine wave inverter

3.2. TOPOLGIES OF THE INVERTER POWER SUPPLY

The inverter power supply can be classified into two main groups, single stage inverters and multi stages inverters. Each group has different topologies and multi stage inverters which will be described in this study. A multi stage inverter is defined as an inverter with more than one stage power conversion. There are many different topologies which are used to determine the parts of the inverter power supply which are listed below [8, 9]:

1. DC-DC-AC topology,
2. DC-AC-DC-AC topology,
3. DC-AC-AC topology.

The choice of such topology depends on many factors, such as size, cost, efficiency and capability. In this study DC-DC-AC inverter will be designed.

3.3. DIGITAL CONTROL FOR INVERTER POWER SUPPLY

Now, with the advent of high speed, lower cost digital signal processing (DSP) ICs and microprocessors, digital control has been one effective candidate in inverter power supply systems design [8]. Traditionally, the implementation of switching type inverter power supply has been accomplished by using analog technique, which is able to provide improved power factor. Analog control can provide continuous processing of signal, thus allowing very high bandwidth. It also gives infinite resolution of the signal measured. Analog control, however, also possesses some drawbacks such as a number of parts required in the system and their susceptibility to aging and environment variations, which

lead to high cost of maintenance. Further, analog control once designed is inflexible and performances cannot be optimized for various utility distortions. In the view of these, the digital controller started to be a valuable candidate in the inverter power supply control. Digital control provides advantages such as programmability, less susceptibility to environmental variations, and fewer part counts. It also reduces the size of the power supply by containing the complexity of control system within the software. Digital control is much flexible than analog control and its cost is becoming lower and applicable for intelligent control. However, the design for digital control inverter power supply faces many challenges which open a wide range of research. The developing cycle of the inverter power supply using traditional design method can be described in Figure 3.2.

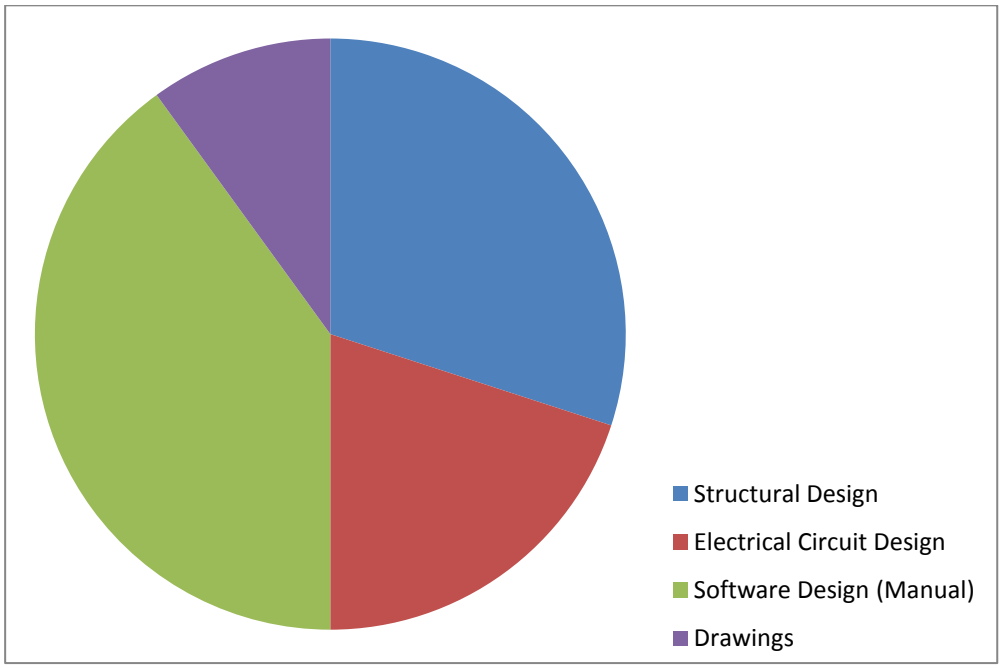


Fig.3. 2 Developing cycle of the inverter power supply

From Figure 3.2 it is clear that the software developing and testing time takes around 50% from the entire developing cycle as well as the software developing itself faces many challenges which are listed below:

1. The embedded software which controls the operation of the inverter power supply is developed manually which takes much time and bugs in such programs are inevitable.
2. The software engineers cannot test the software performance in early stage of design because it needs to be tested in the actual prototype which takes long time to develop. Each time the software is modified, it must be built, downloaded,

executed and evaluated in the actual prototype which takes so much time and may cause severe damage in the prototype. Moreover, it adds cost to the product as well as it increases the product time.

3. It is so difficult to monitor the control algorithm behaviors before the actual implementation.
4. Bug discovery is considered a critical problem and it results in production delay time and additional cost will be added.
5. Within the scope of our investigation, all the previous study concentrated on the modeling of the analog part and the digital pulse was generated by pulse generator. So in the actual implementation, the software is needed to be tested in the actual prototype which may lead to system failure as well as it takes so much time to obtain the optimum parameters. And when any change is occurred in the embedded software, the verification process is needed to start from the beginning. It takes long time to optimize the software parameters using the conventional design method, as shown in Figure 3.3.

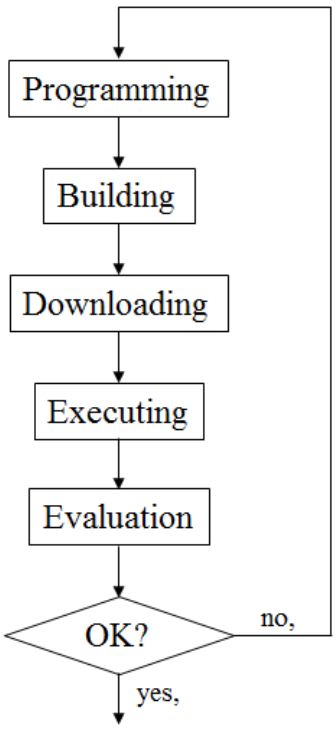


Fig.3. 3 Conventional program development process

So, in this research, we develop a virtual environment for the inverter power supply design to develop the embedded software and optimize its parameters. Then, we propose the new application of control algorithm and verity its performances in the virtual environment. All the details of the inverter power supply structure and the virtual environment will be presented in Chapter 3 and Chapter 4, respectively.

3.4. DESCRIPTION OF THE INVERTER POWER SUPPLY

As mentioned before, an inverter power supply is a device which can convert DC to AC that can be used in various AC applications. Many topologies are considered as a candidate of the inverter design and there are many factors affecting the choice of such topology, such as the size and the required efficiency as well as the cost of the inverter. In this study, an inverter power supply configuration is broken into two stages. The first stage is to step up the DC voltage level by using DC/DC converter and the second stage is to invert DC to AC through a DC/AC inverter.

The block diagram of the inverter power supply is shown in Figure 3. 4.

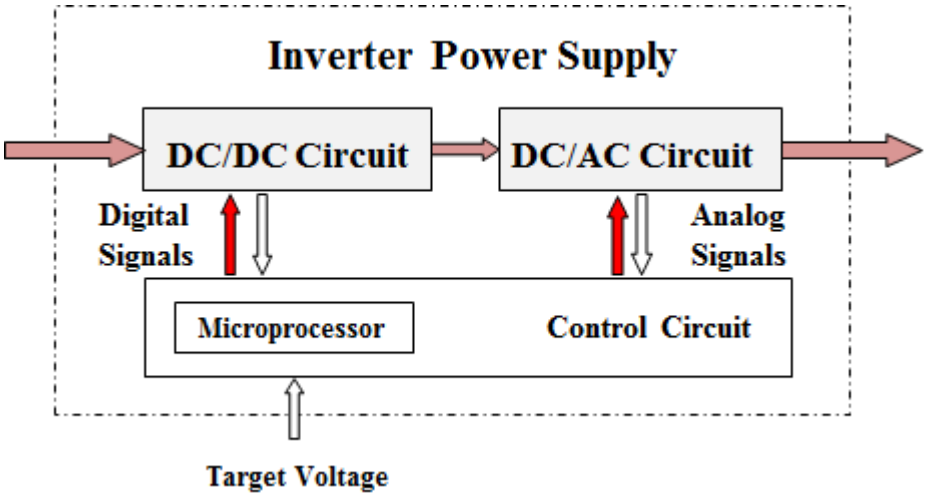


Fig.3. 4. Block diagram of inverter power supply

3.4.1. DC-DC Conversion

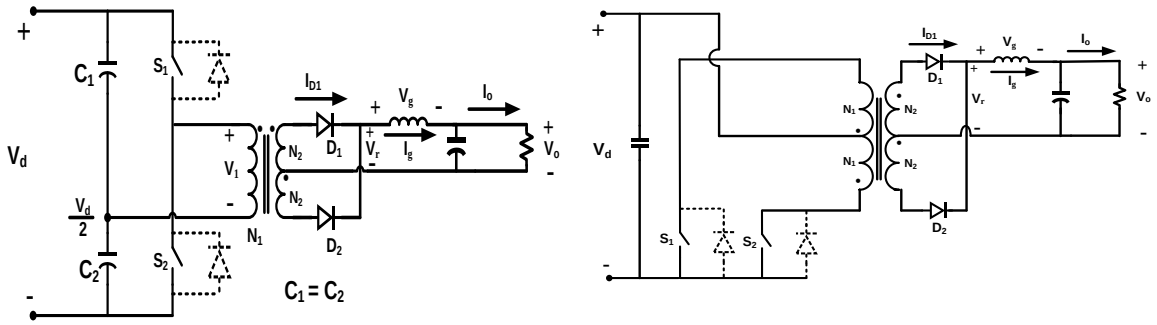
The first step of the inverter power supply is to step up the DC voltage level which comes from a battery to higher DC level by using the DC/DC converter. DC/DC conversion revolves around the conversion of DC voltage level which comes from sources as batteries,

solar panels, fuel cells, or wind generations to higher DC level. There are many different types of DC/DC converter, each of them tends to be more suitable for some types of application than for others. For conveniences, we can classify them into various groups. For example, some converters are only suitable for stepping down the voltage while others are only suitable for stepping it up; a third group can be used for both cases stepping up and stepping down. Another important distinction among converters is which one offers full dielectric isolation between their input and output circuits. Dielectric isolation behavior may be important for some applications, although it may not be important in many others. In this section we are going to look briefly at each of the main types of DC/DC converter in the current use as presented below:

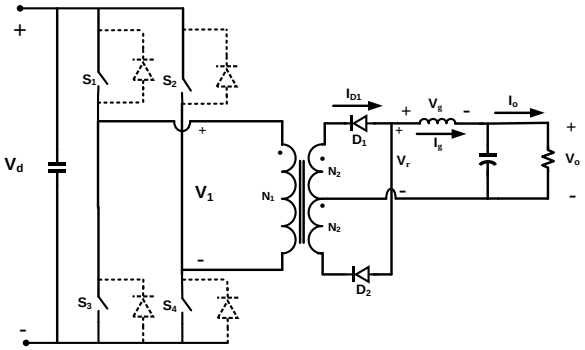
- **Non isolating converter**: The non-isolating type of converter is generally used where the voltage needs to be stepped up or down by relatively small value (less than 4:1), and there is no problem with the output and the input having dielectric isolation. Examples are 24V/12V reducer, 5V/3V reducer, and 1.5V/3V step up converter [7]. There are five main types
    1. Buck converter.
    2. Boost converter.
    3. Buck-boost converter.
    4. Cuk converter.
    5. Charge-pump converter

- **Isolating converters:** In many applications the non isolating converter is unsuitable where the output needs to be completely isolated from the input. Isolated converter topologies can provide advantages in applications which require large voltage conversion ratio. The transformer in the isolation type DC/DC converter can reduce switch and diode device stresses and allow multiple windings or taps to be used for multiple converter outputs [7]. Here is some of the isolating converter.
    a) Half Bridge
    b) Push-Pull
    c) Full Bridge DC-DC converter

In this study the isolation type DC/DC converter is used in the inverter power supply implementation. Figure 3.5 show examples of the isolated converter.



a) Half bridge (b) Push-Pull,



(c) Full bridge

Fig.3. 5. Examples of the isolated DC/DC converter

The full-bridge is a popular design for both buck and boost applications. It is one of the simplest and most effective cost configurations. Another advantage for using the full bridge converter is the fact that when higher power application are requested the full bridge converter can act as a modular block and that it is possible to stack up [5]. For this purpose, the chosen topology for the converter to be used in this application is a full bridge phase shifted PWM converter [10-11].

A schematic of DC/DC converter is shown in Figure 3.6. The major components are the four transistors (full bridge converter).

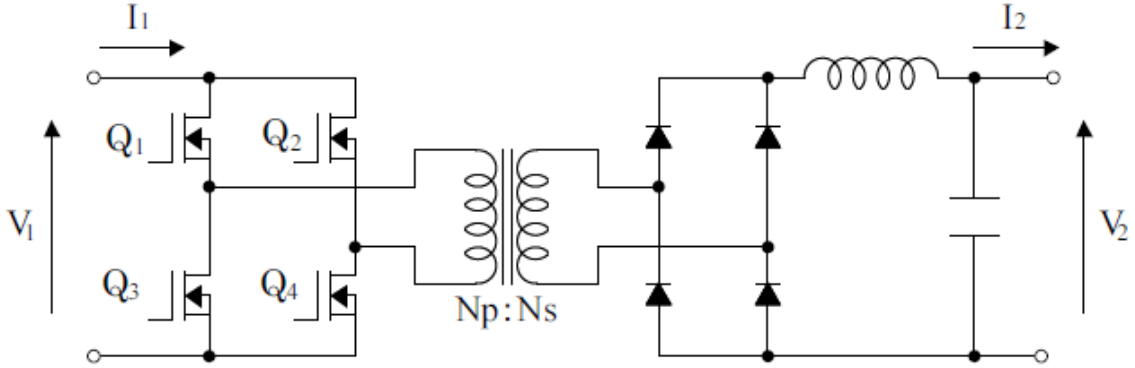


Fig.3. 6. Circuit schematic of DC/DC converter

The main purpose for this full bridge converter is to chop up the DC voltage so that AC is seen by the transformer. The current is forced across the primary side of the transformer when Q1 and Q4 are on and Q2 and Q3 are off, and the current in the primary of the transformer changes its polarity when Q2 and Q3 are on and Q1 and Q4 are off as shown in Figure 3.7.

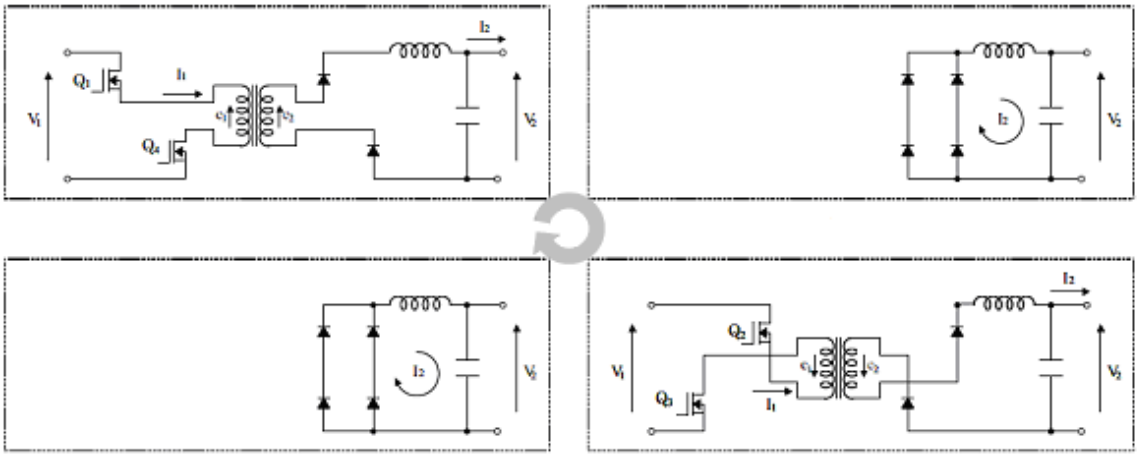


Fig.3. 7. Switching operation of DC/DC converter

The transformer is a part of the DC/DC circuit that is responsible for boosting the voltage V_1 by means of a ferrite core, a primary winding and a secondary winding. It is important to note that the transformer does not create any power, and it only transforms or transfers the voltage. The transformer operates by inducing a magnetic flux on the core from the current flowing through the primary winding. The flux passing through the core is induced onto the secondary winding and the current flows out of the device. The transformer output will apply to the full bridge rectifier and the low pass filter, respectively, to get the stepped up DC voltage V_2. The DC voltage is converted to a square wave signal due to the switching operation of the full bridge then the signal is stepped up the transformer as shown in Figure 3.8.

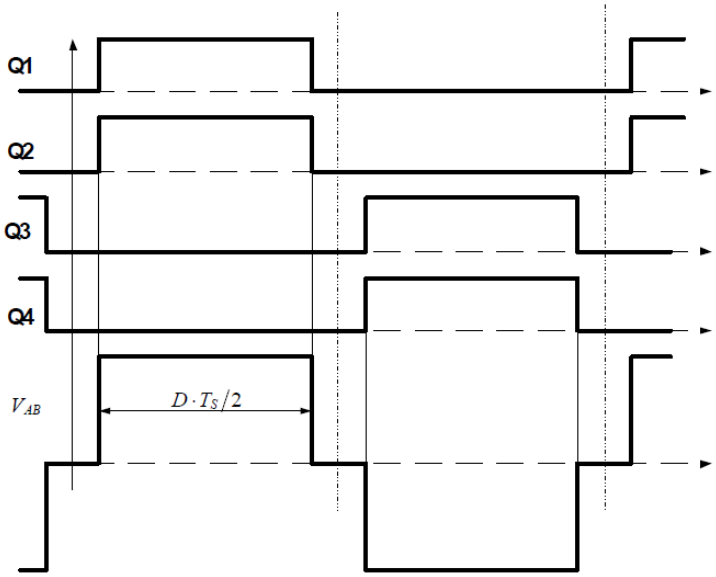


Fig.3. 8. DC/DC switching operation

The output of the transformer is rectified using the full bridge rectifier circuit and then filtered using low path filter all the signals are shown in Figure 3.9.

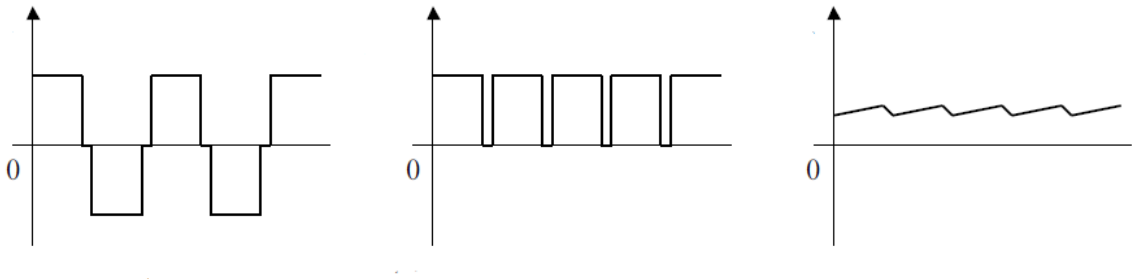


Fig.3. 9. DC/DC rectification and filtering operation

3.4.2. DC/AC Inverter

The second stage of the inverter power supply is to invert the new DC level into AC voltage through DC/AC inverter. There are many different topologies for a DC/AC inverter. The most common topology is the full bridge configuration because of its easy filtering [4, 10]. The full-bridge inverter was chosen as the inverting output stage for a number of reasons. It is preferred over a half-bridge inverter because with an equivalent input voltage, the full-bridge inverter can provide twice the output voltage. The full-bridge inverter is also significantly more controllable than other configurations. A single phase full bridge inverter is shown in Figure 3.10 and the function of the full bridge inverter is to convert the DC voltage supplied by DC/DC converter into a 100V, 60 Hz sine wave. The

most important part of the DC/AC conversion process is in the generation of the sinusoidal input signals to the gates of the MOSFETs. This will be covered in the next section which focuses on microprocessor control systems and PWM.

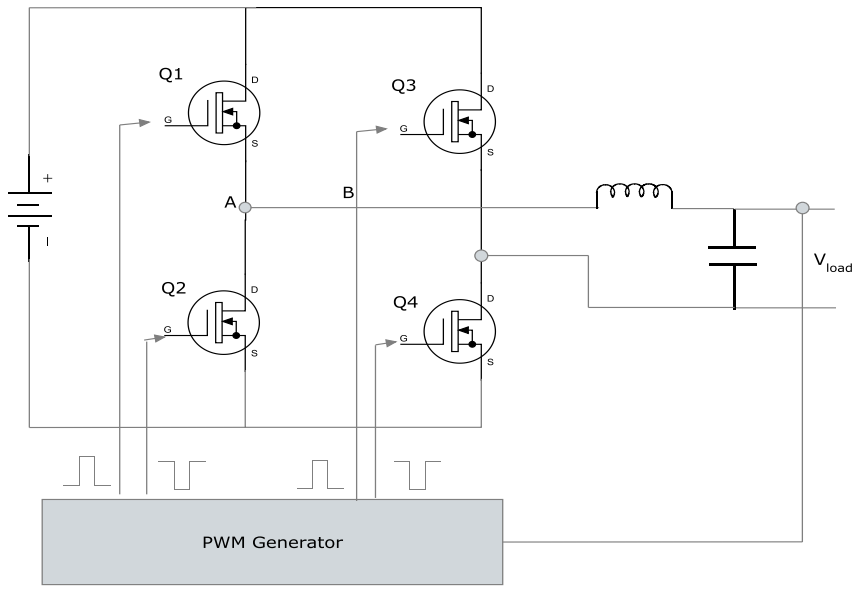


Fig.3. 10. Circuit schematic of DC/AC inverter

The PWM pulses which are generated by a microcontroller are fed into the gates of full bridge inverter. Programming the microcontroller allows the transistors Q1 and Q4 to be on while Q2 and Q3 to be off and vice versa. Due to the limited response time and delay time of the transistors, two switches in one leg may be switched on at the same time then shoot through will occur and the switches may be damaged due to high short circuit current then the dead time is introduced in order to avoid the occurrence of the short circuit.

Figure 3.11 shows the ideal switching patterns and the drive signals containing the dead time for the inverter leg. The S_p and S_n are the ideal switching pattern of the positive device and the negative device of the full bridge DC/AC inverter, respectively. As mentioned before, the short time delay is used to avoid shoot-through, the actual gate drive signals must be delayed by the dead time. The gate drive signals containing the dead time are denoted as S_{pd} and S_{nd} since the gate drive signals are shifted from the center of the sampling interval by the dead time. The generated phase voltage is also shifted as much as the delay time. It had been reported that the generated voltage pulses residing in the middle of the sampling interval contain the least amount of harmonics [4]. Although the

produced voltage pulses resulting from each of the gate drive signals during the sampling intervals are not much affected, the resultant voltage during an entire cycle is significantly reduced due to the dead time. In addition, those cumulated delays may distort the output waveform of the inverter.

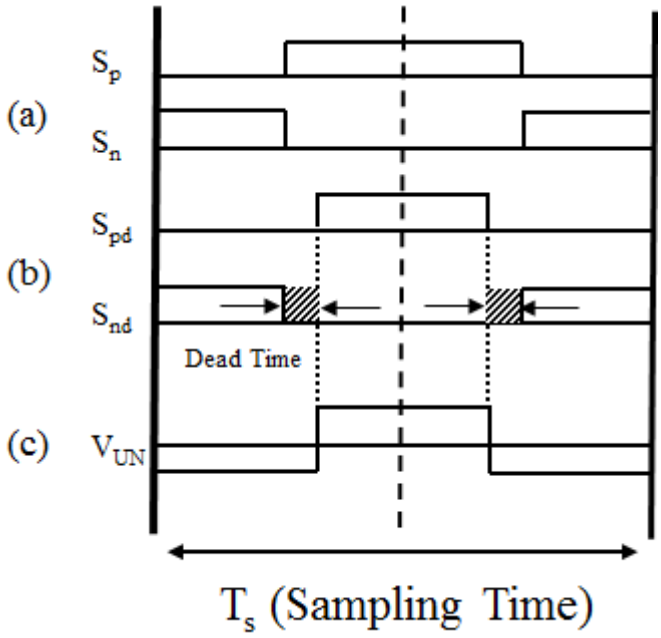


Fig.3. 11. Gate drive signals of the PWM inverters

In fact, the addition of the dead time can improve the performance of inverter power supply by preventing the short circuit current. However, the instability and harmonic distortion problem can be arising due to the miss selection of the sufficient dead time value [12]. Figure 3.12 shows the relationship between the voltage drop of inverter and the dead time value.

The relation between dead time and the voltage drop can be described as presented in Figure 3.13. It is considered that V_d is the voltage drop due to the dead time and the amplitude of the rectangular waveform indicates the mean voltage of each voltage drop during half cycle of the output voltage.

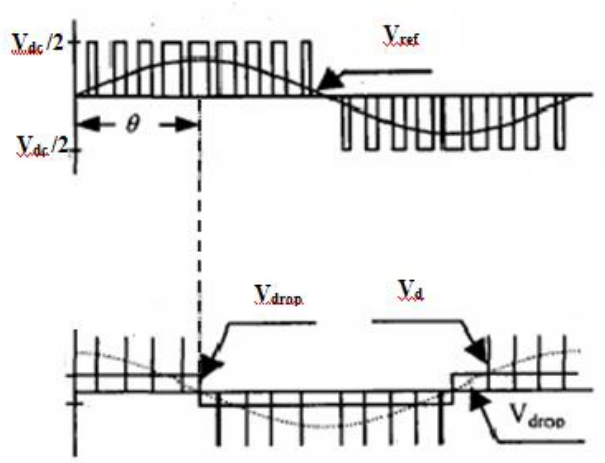


Fig.3. 12. Relation between dead time and the voltage drop

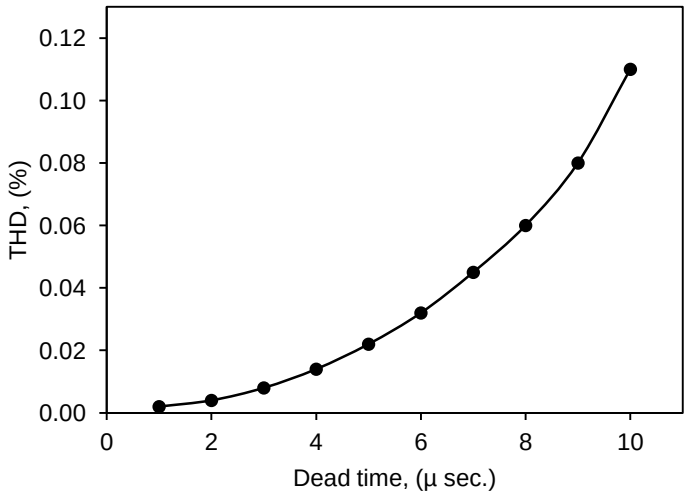


Fig.3. 13. The relation between dead time value and the total harmonic distortion (THD)

As described in the previous work [13], the amplitude of the rectangular waveform can be calculated by the following equation (3.1).

$$m = \frac{p}{2}(t_{dead} \times V_{dc}) \times 2 f_o = f_c \times t_{dead} \times V_{dc} \tag{3.1}$$

where:

$p=f_c/f_o$,

f_c : carrier frequency of the inverter power supply,

f_o : output frequency of the inverter,

t_{dead} : the dead time value of the inverter,

v_{dc} : DC input voltage.

$$v_{drop1} = \frac{4}{\pi} m \tag{3.2}$$

The reference voltage and the voltage drop can be described as in equations (3.3) and (3.4).

$$v_{ref} = \sqrt{2}V_1 \sin(wt) \tag{3.3}$$

$$v_d = \sqrt{2}V_{drop} \sin(wt) \tag{3.4}$$

where:

V_1 is the RMS value of the reference voltage

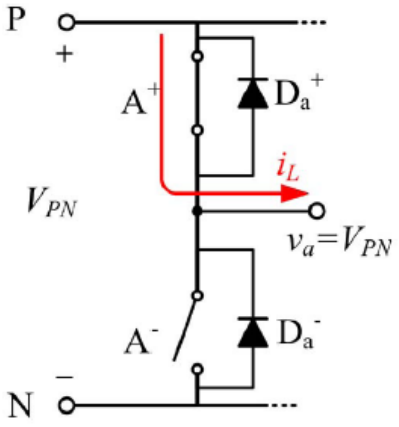
$$\begin{aligned} V_{out} &= \sqrt{2}V_1 \sin(wt) - \sqrt{2}V_{drop}(\sin(wt) - \theta) \\ &= \sqrt{2}\{V_1 \sin(wt) - V_{drop1}(\sin(wt)\cos\theta - \cos(wt)\sin\theta)\} \\ &= \sqrt{2}\sqrt{(V_1 - V_{drop1})^2 + 2V_1V_{drop1}(1-\cos\theta)}\sin(wt - \beta) \end{aligned} \tag{3.5}$$

Here:

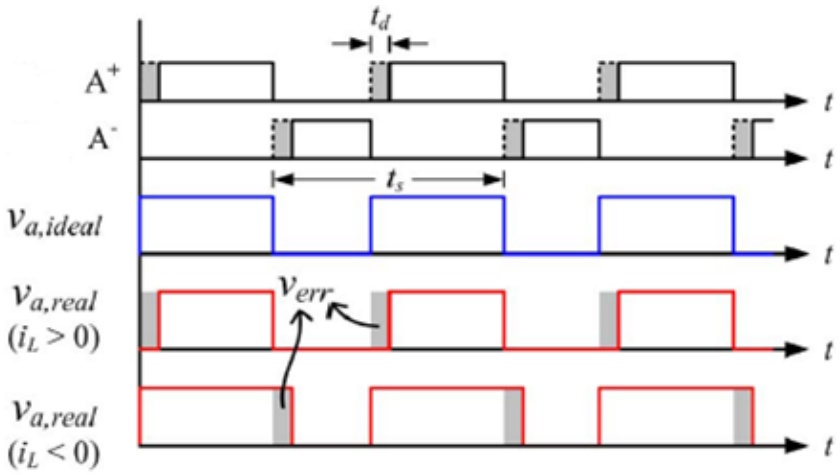
$$\beta = tan^{-1}\frac{V_{drop1}\, sin\,\theta}{V_1 - V_{drop1}\, cos\,\theta}$$

From the equations above, it is apparent that the output voltage of the inverter power supply contains voltage drop and phase delay and those values can affect the output if the dead time is not optimized correctly.

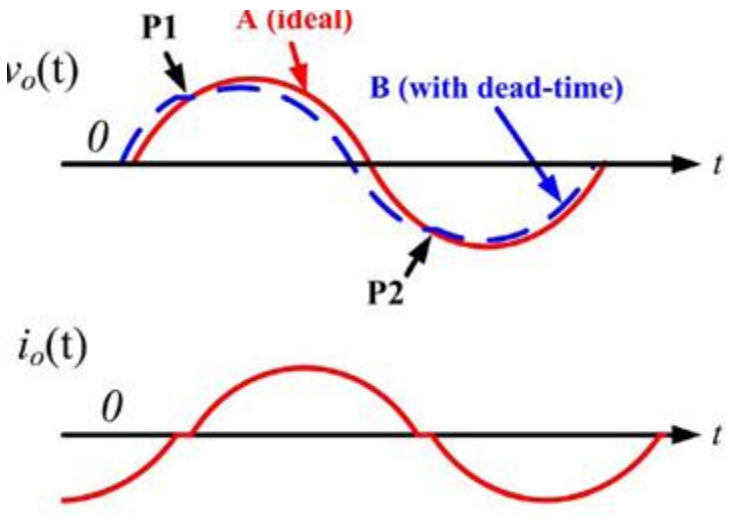
Figure 3.14 shows how the miss selection of the dead time can affect the performance of the inverter power supply. Figure 3.14 (a) shows the circuit of the inverter leg and figure 3.14 (b) presents the modified PWM signal by adding the dead time value.



(a) Inverter leg



(b) Pulse with the dead time.



(c) Current distortion

Fig.3. 14. Voltage and current distortion caused by the dead time

From the previous figures, the output distortion is defined by the difference between the ideal and the real voltage. The real output voltage is greater than the ideal voltage when the current is negative and the real output voltage is smaller than the ideal voltage when the current is positive as shown in Figure 3.14 (c). In this figure, the blue line (signal B) describes the output voltage with the dead time and the red line (signal A) describes the ideal output without dead time. When those two lines are intersected, the output current also can be distorted at those two zero crossing point P_1 and P_2.

So the selection of the dead time should be optimized very carefully. The shorter is this time, the better is the inverter performance. It is reported that the dead time should be usually around 1-5μsec [8]. In this research the value of the dead time is optimized in the virtual environment and then tested in the actual inverter power supply prototype. The details of the dead time value are described in Chapter 4.

3.4.3. SH microprocessor

In the inverter power supply application, the microprocessor is used to control the switching period of the transistor as a digital controller. However, due to the fact that a digital PWM technique can provide the benefits which cannot be given by an analog one, a digital PWM technique starts to be used [14,15]. The standard method for generating a PWM using a microcontroller or DSP is developed by using one of the built-in PWM modules. These modules operate by comparing a free running timer with a duty cycle and duty period register. When a matching occurs between the timer and duty cycle register, the corresponding pin is either set to “high” or “low”. The matching between the timer and the duty cycle register also causes the timer to reset to zero and then to restart counting [15]. Depending on the type of microcontroller or DSP, the PWM can be classified into “left-aligned”, “central-aligned” or “right-aligned”. In this study, Renesas SH microprocessor is used. It is a Reduced Instruction Set Computer (RISC) integrating a Renesas original RISC CPU core with peripheral functions required for a system configuration [14]. SH RISC is a microprocessor family and combines the computational ability of a high speed RISC core with embedded Multiply-Accumulate hardware and extensive on-board peripheral function to enable a virtual single chip PID controller [15].

In the inverter power supply application, two separate control units in the SH microcontroller are used to generate the PWM signal and control the system operation. These two units are Motor Management Timer unit (MMT) which controls the generation of the PWM pulse in the DC/DC stage and Multi Function Timer Pulse Unit (MTU) which

controls the generation of the PWM DC/AC stage. The block diagram of the entire microprocessor is shown in Figure 3.15 [16].

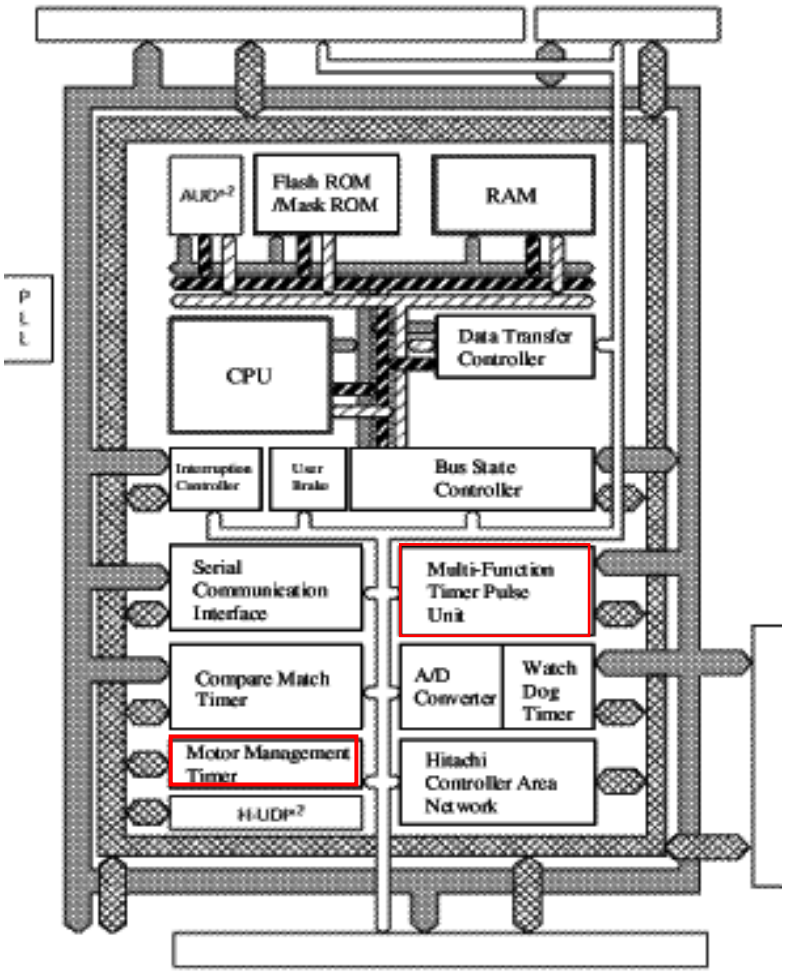


Fig.3. 15. Block diagram of SH microprocessor

3.4.3.1. Motor Management Timer (MMT)

Motor Management Timer (MMT) can output 6-phase PWM waveforms with non-overlap times. Figure 3.16 shows a block diagram of the MMT. In the inverter power supply application, the MMT unit is used to control the switching devices in the DC/DC stage by generating the PWM signal.

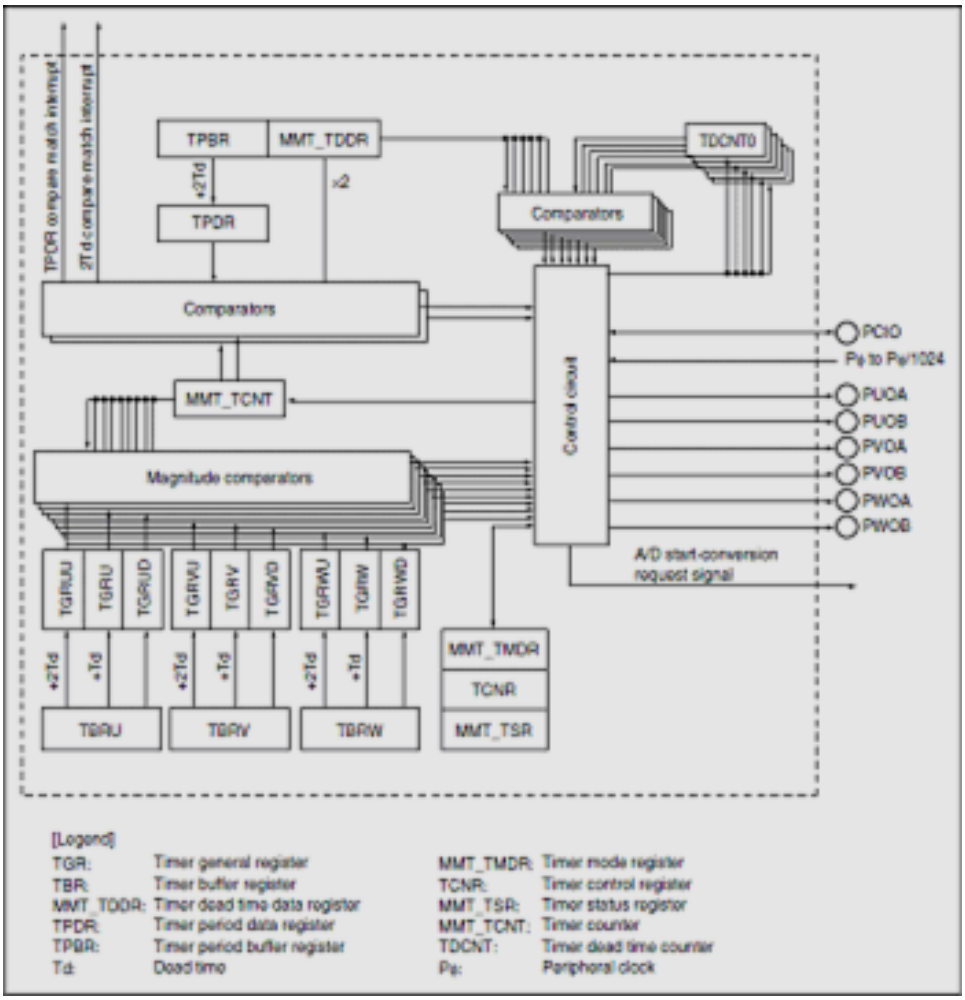


Fig.3. 16. Block diagram of MMT

Pin Configuration of the MMT unit in described in Table 3.2.

Table 3. 2: The pin configuration of the MMT

Name	I/O	Function
PCIO	Input/Output	Counter clear signal input when se as an input by PAIORL register: toggle output in synchronization with the PWM cycle when set as output by PAIORL register.
PUOA	Output	PWMU phase output (positive phase)
PUOB	Output	PWMU phase output (negative phase)
PVOA	Output	PWMV phase output (positive phase)
PVOB	Output	PWMV phase output (negative phase)
PWOA	Output	PWMW phase output (positive phase)
PWOB	Output	PWMW phase output (negative phase)

As shown in Table 3.2, the PUOA, PUOB, PVOA, PVOB, PWOA, and PWOB pins are PWM output pins [16].

Figure 3.17 illustrates an example of the PWM pulse which is generated from the MMT unit [16]. In this figure the PWM output waveform is generated by comparing the values in the TCNT counter and the TGR registers resulting in the compare output waveform. Then the dead time generation process is started using the TDCNT0 and TDCNT1 registers [16]. The output generation wave form is generated by adding the compare output waveforms with the dead time signal finally. The PWM waveform is generated by converting the output generation waveform to the output PWM pins. In the operating modes, PWM waveforms with any duty cycle from 0% to 100% can be generated.

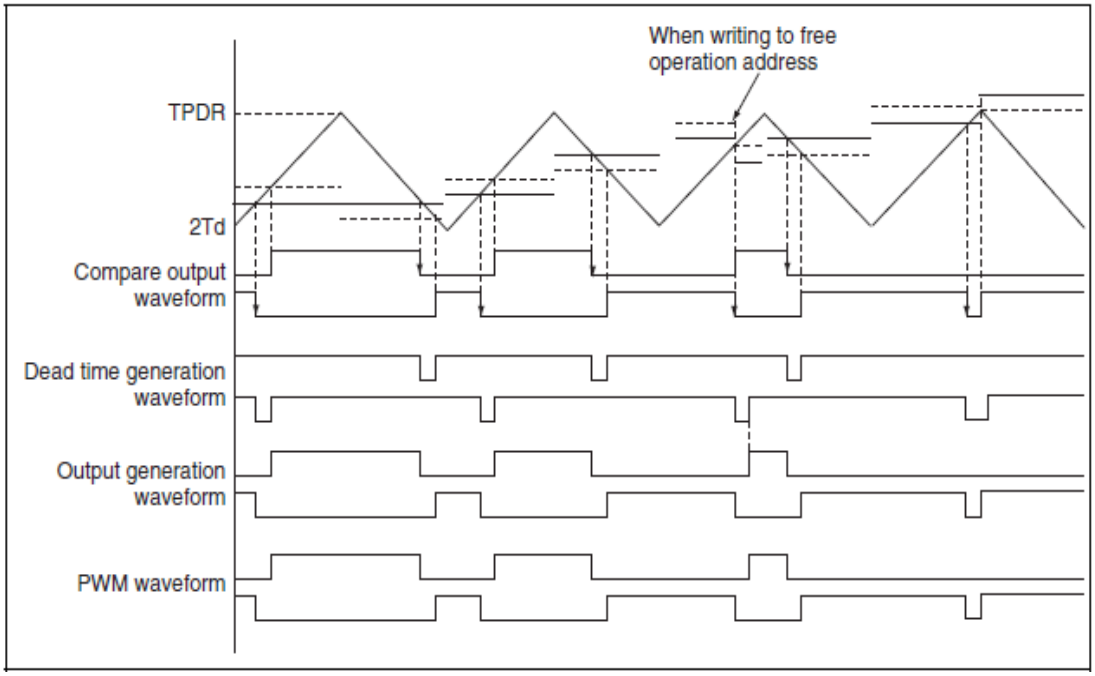


Fig.3. 17. Example of PWM waveform generation from MMT unit

3.4.3.2. Multi-Function Timer Pulse Unit (MTU)

MTU is the control unit which is used to generate the PWM pulses that control the operation of the DC/AC inverter stage. MTU comprises of five 16 bit timers. As shown in Figure 3.18, channels 3 and 4 of the MTU are used in complementary PWM mode with programmable dead time to generate the chopping waves for sinusoidal PWM.

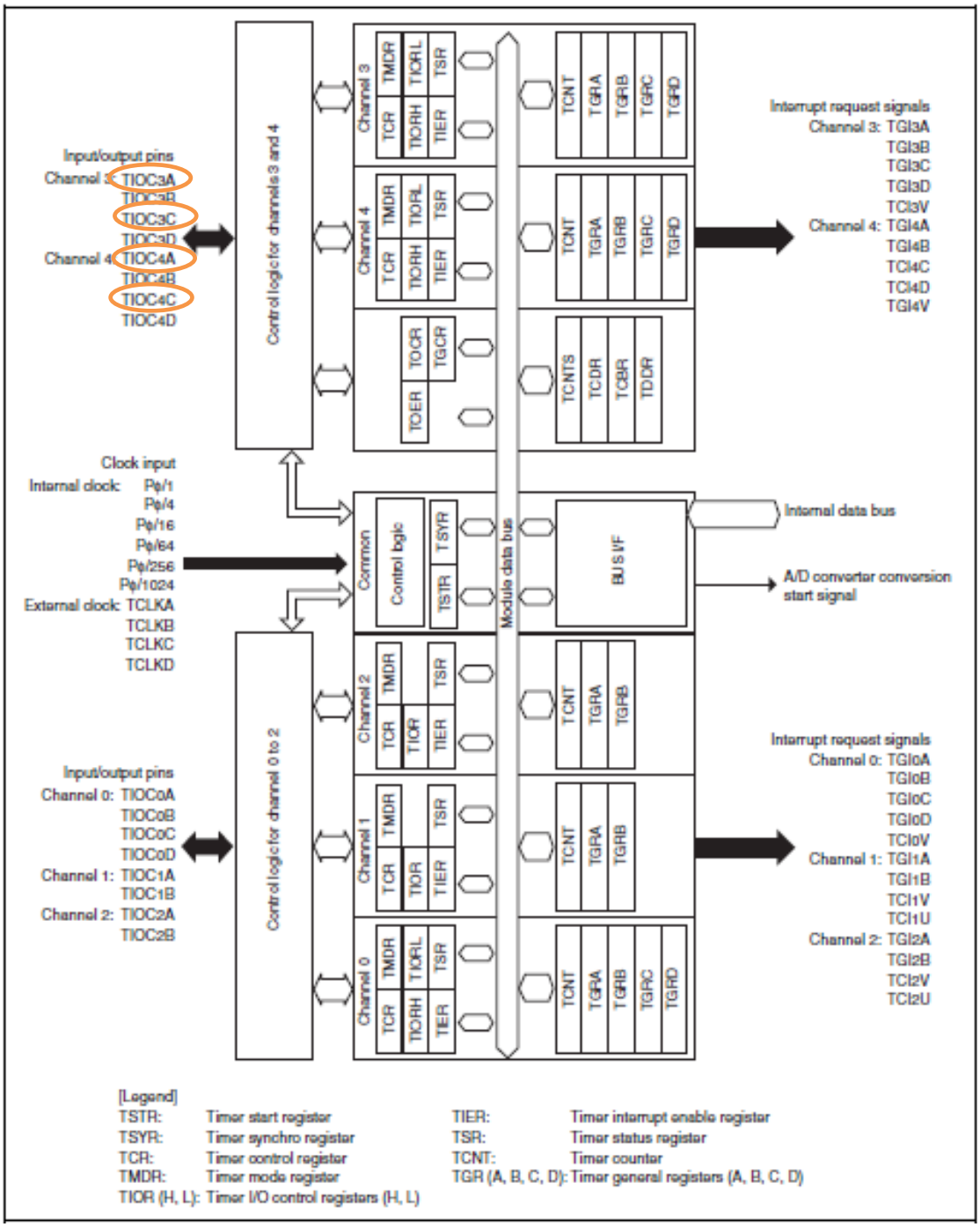


Fig.3. 18. Block diagram of MTU

In PWM mode, PWM waveforms can be generated from the output pins. The output level can be selected and TGR registers settings can be used to output a PWM waveform in the range of 0% to 100% duty. Designating TGR compare match as the counter clearing source enables the period to be set in that register. All channels can be designated for PWM mode independently. Synchronous operation is also possible [16].

There are two PWM modes: in PWM mode 1, PWM output is generated from the TIOCA and TIOCC pins by pairing TGRA with TGRB and TGRC with TGRD. In PWM mode 2, PWM output is generated using one TGR as the cycle register and the others as duty registers. The output specified in TIOR is performed by means of compare matches. Upon counter clearing by a synchronization register compare match, the output value of each pin is the initial value set in TIOR. If the set values of the cycle and duty registers are identical, the output value does not change when a compare match occurs [16, 17].

3.5. PULSE WIDTH MODULATION

Nowadays, Pulse Width Modulation (PWM) plays a major role in the generation of pure sine waves. PWM can be defined as a powerful technique for controlling the analog circuits. The applications of PWM are widely used like ranging from measurement and communications to power control and conversion [8]. In the inverter power supply application, the PWM is a switching method which is used to control the operation of the DC/DC converter and DC/AC inverter stages, where the pulse width (duty cycle) at the switching gates of the transistors varies according to a sinusoidal reference signal (control signal). The generation of PWM patterns can be done by using two different techniques which are analog or digital techniques [7, 8, 18]. PWM provides a way to decrease the Total Harmonic Distortion (THD) of load current. Here, the total harmonic distortion, or THD, is defined as the ratio of the sum of the powers of all harmonic components to the power of the fundamental [8]. The THD requirement can be achieved more easily when the output of PWM inverter is filtering. The unfiltered PWM output will have a relatively high THD, but the harmonic will be at the much higher frequencies than for the required frequency that making filtering become easy [18,19]**.**

There are many different PWMs that have been proposed for single-phase inverters such as Bipolar, Unipolar [7], Space Vector Modulation (SVM) [18], the optimal PWM [20] and optimized PWM [21].

Analog PWM control requires the generation of both reference and carrier signals that fed into a comparator which creates output signals based on the difference between the signals. The reference signal is a sinusoidal signal at the frequency of the desired output signal, while the carrier signal is often either a sawtooth or a triangular wave at a frequency significantly greater than the reference. When the carrier signal exceeds the reference, the comparator output signal is at one state, and when the reference is at a higher voltage, the output is at its second state. This process is shown in Figure 3.19. At each point, where the

reference signal and the carrier signal intersect, the output of PWM toggles from a high state to a low.

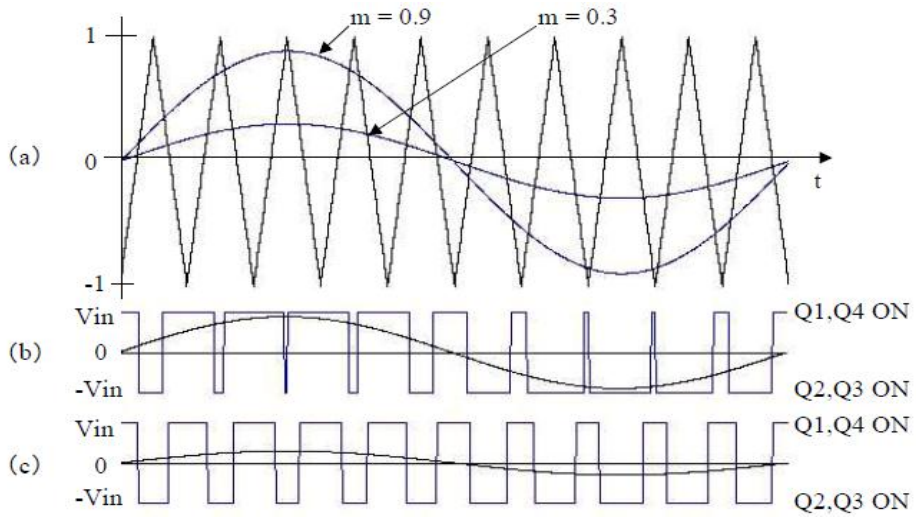


Fig.3. 19. Analog PWM generation

The frequency of the carrier signal is generally kept constant along with its values. The control signal is used to control the switching duty by changing its values as a factor, called modulation ratio "m", as described in equation (3.6).

$$m = V_{controller} / V_{carrier} \quad (3.6)$$

Here: $V_{controller}$ is the peak amplitude of reference sine wave and $V_{carrier}$ is the peak amplitude of sawtooth wave, respectively.

Figure 3.20(a) shows a PWM output at a 10% duty cycle. That is, the signal is on for 10% of the period and off the other 90%. Figure 3.20 (b) shows the PWM pluses when the duty equal to 50% and Figure 3.20 (c) represents the PWM output at duty cycle equal to 90%. These three PWM outputs encode three different analog signal values at 10%, 50%, and 90% of the full strength. If, for example, the supply is 5V and the duty cycle is 10%, a 0.5V analog signal can be generated.

(a) 10% of duty

(b) 50% duty

(c) 90% duty cycle

Fig.3. 20. PWM outputs

Due to the fact that a digital PWM technique can provide the benefits which cannot be provided by an analog one, a digital PWM technique starts to be used. Traditionally, the implementation of the switching of the inverter power supply has been accomplished by using an analog technique [3]. Analog PFC IC's which are manufactured by TI/Unitrode, Fairchild, and ST microelectronics are available and have been able to provide improved power factor. Analog control can provide continuous processing of signal, thus allowing very high bandwidth. It also gives infinite resolution of the signal measured. Analog control, however, also poses some drawbacks such as a number of parts required in the system and their susceptibility to aging and environment variations, which lead to high cost of maintenance. Further, analog control once designed is inflexible and the performances cannot be optimized for various utility distortions. On the other hand, the digital control can provide advantages such as programmability, less susceptibility to environmental variations, and fewer part counts [2]. It also reduces the size of the power supply by containing the complexity of control system within the software. Therefore, since digital control is much flexible than analog control, it is with lower cost, and applicable in many application [22].

Depend on the type of the microcontroller or the digital signal processor, the corresponding PWM is generated. The standard method for generating the PWM using the microcontroller or DSP is by using the built-in PWM modules. These can be done by comparing a free running timer with the duty cycle and duty period register. When the timer and duty cycle register are matched, the PWM output pins are either set to high or low. The timer is reset to zero and restart counting when a match between the timer and duty period register is occurred. The digital PWM generation can be classified as PWM can be left, center or right aligned as shown in Figure 3.21. The left aligned PWM is the most common PWM available on a low-cost microcontroller, as it only requires an up-counter to be generated. For left alignment, the PWM module automatically sets the PWM pin to high at the start of the switching period, and when a match occurs between the timer and duty cycle register, the PWM pin is set to low, and vice versa for the right aligned PWM. The centre aligned PWM is more difficult to generate [8].

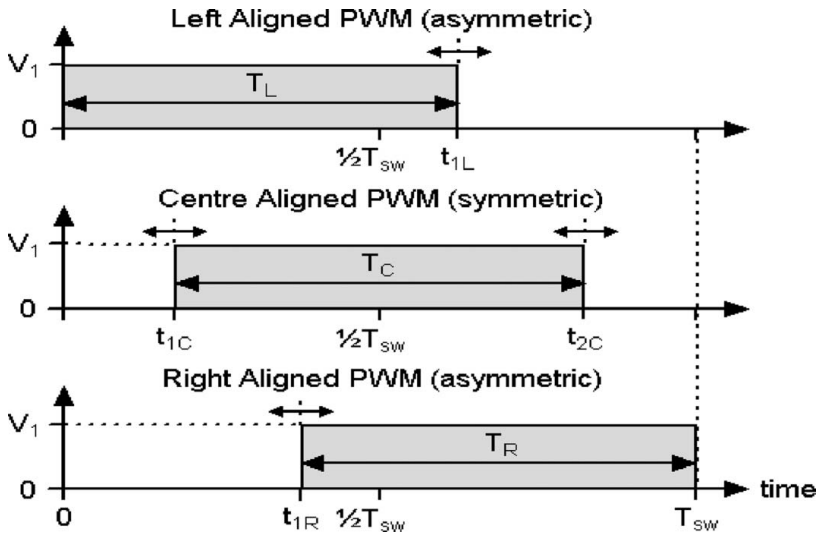


Fig.3. 21. Microcontroller based PWM alignments

3.6. PID Controller

For high performance inverter power supply, the requirement likely to be more stringent, such as fast response to load change in input voltage [23, 24, 25] and a system control technique should be used. The main duty of the inverter control system is to regulate the output voltage against all possible existing disturbances. Traditionally, the inverter power supply was controlled using analog control. But due to the fact that a digital control technique can provide the benefits which cannot be provided by an analog one, a digital control starts to be a good candidate in the inverter power supply design [23, 24, 25]. Today, various modern control techniques have been proposed to develop a sinusoidal

output which faces the requirements. These control techniques are classified into three groups which are presented below:

- Model based instantaneous feedback control: In these controllers, system variables such as the output voltage, load current and/or inductance/capacitor current are fed back to achieve good steady-state performances. The challenges of such controller technique are to make the closed loop operation robust to the system variations and also to eliminate the need for sensing inductor/capacitor current to reduce the cost [23].
- Feedforward learning control: In general, the load current of the inverter power supply is periodic at fundamental frequency. Due to this fact feedforward learning control is found to be attractive but such control has the ability to achieve excellent steady state performances but poor transient response when it is applied alone.
- Nonlinear control including sliding mode control (SMC), adaptive control and NN based control but such controllers are very complex to be applied [23].

To improve both the steady state performances and the transient performances of the inverter power supply, two layer control algorithm was newly applied. The control algorithm includes the feedback control and feedforward control. One of the powerful feedback controller techniques is the PID controller. The PID controller, which consists of proportional, integral and derivative elements, is widely used in feedback control of industrial processes. In applying PID controllers, engineers must design the control system. So, they must first decide which action mode to choose and then adjust the parameters of the controller so that their control problems are solved appropriately. To that end, they need to know the characteristics of the process. As the basis for the design procedure, they must have certain criteria to evaluate the performances of the control system. The basic knowledge about those topics is summarized in this section [26, 27, 28, 29].

The PID controller was first placed on the market in 1939 and has remained the most widely used controller in process control until today. An investigation performed in 1989 in Japan indicated that more than 90% of the controllers used in process industries are PID controllers and advanced versions of the PID controller. PID controller means proportional,

integral and derivative, it includes those three elements with three different functions as illustrated in Figure 3.22.

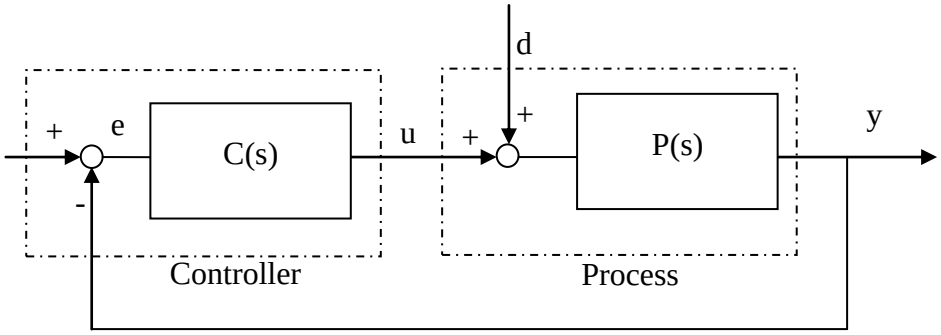


Fig.3. 22. Conventional feedback control system

The main three elements of the PID controller are:

- **P element:** Proportional to the error at the instant *t*, this is the present error.
- **I element:** Proportional to the integral of the error up to the instant *t*, which can be interpreted as the accumulation of the "past" errors.
- **D element:** Proportional to the derivative of the error at the instant *t*, which can be interpreted as the prediction of the "future" error.

The transfer function *C(s)* of the PID controller can be calculated from the following equation

$$C(s) = K_p\left\{1 + \frac{1}{T_1 s} + T_D D(s)\right\} \quad . \tag{3.7}$$

Where K_p, T_I and T_D are positive parameters, which are respectively referred to proportional gain, integral time and derivative time, and as a whole, PID parameters. *D(s)* is the transfer function given by the following equation:

$$D(s) = \frac{s}{1 + (T_D / \gamma)s} \tag{3.8}$$

Here *D(s)* is the approximate derivative, the approximate derivative *D(s)* is used in place of the pure derivatives, because the latter is impossible to realize physically. Further γ is a positive parameter, which is referred to derivative gain. It is reported that the pure derivative is not the ideal element to use in a practical situation. It is usual practice to use a fixed value of γ, which is typically chosen as 10 for most applications [26].

In the practical applications the designer can use different combination of the three functional elements which are appropriate to the applications. Theoretically, there are seven combinations but five of them are used in practical applications. Those combinations are called the action modes of the PID controller which are described in Table 3.3.

Table 3. 3: Action modes of the PID control

Action mode	**Elements**	**Transfer Function**
Proportional control	P element only	C(S)=Kp
Integral control	I element only	C(s)=Kp/s
PI control	P and I elements	$C(S) = K_p[1 + \frac{1}{T_1 s}]$
PD control	P and D elements	$C(S) = K_p\{1 + T_D(s)\}$
PID control	Three elements	$C(s) = K_p\left\{1 + \frac{1}{T_1 s} + T_D D(s)\right\}$

In the inverter power supply application we used the PI action mode because PI action mode has simple structure, easy to implement in the practical applications and its flexibility.

It is reported that the disadvantage of using the traditional PI controller is that the delay time caused by the analog to digital (A/D) conversion and computation time of the microprocessor reduce the maximum available pulse width. Hence this limitation can result in output voltage waveform disturbance [15]. Feedforward control combined with feedback control can significantly improve performances over simple feedback control whenever there is a major disturbance that can be measured before it affects the process output. In the most ideal situation, feedforward control can entirely eliminate the effect of the measured disturbance on the process output. Even when there are modeling errors, feedforward control can often reduce the effect of the measured disturbance on the output better than that achievable by feedback control alone. However, the decision as to whether or not to use feedforward control depends on whether the degree of improvement in the response to the measured disturbance justifies the added costs of implementation and maintenance. The economic benefits of feedforward control can come from lower

operating costs and/or increased scalability of the product due to its more consistent quality [23]. Building on the previous explanations, the two layer controller is applied as shown in Figure 3.23.

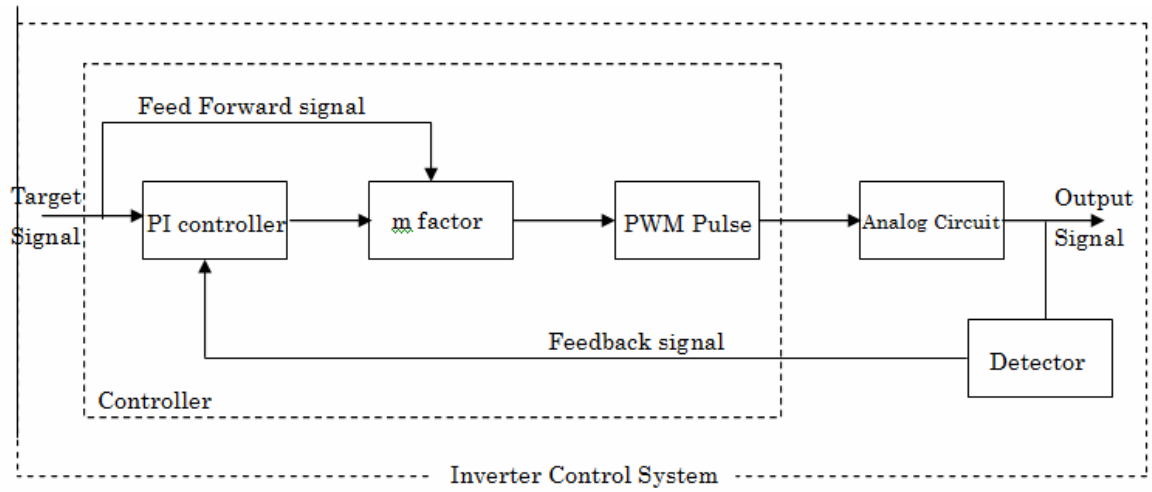


Fig.3. 23. Block diagram of newly applied control system

Figure 3.23 shows the block diagram of the inverter power supply control system. The microprocessor implementation of this controller can be described thought the equation (3.9)

$$u_{pi} = K_p e(K) + K_I T \sum_{n=1}^{K} e(n) + R(k) \qquad (3.9)$$

Where K_p and K_i, are the Proportional and Integral coefficients, respectively; u_{pi} is controller output and e (K) is the error signal and R(k) is the feedforward signal. The error value can be calculated as a difference between the target voltage and the actual output voltage. The value of the modulation ratio *m* will be changed depend on the present value of the error as well as the previous value. The evaluation of the control system will be explained in Chapter 5.

REFERENCES

[1] S. S. Wekhande, B. N. Chaudhari, S. V. Dhapte and R. Sharma, “Low cost Inverter Drive for Two Phase Induction Motor”, In Proceeding of International Conference on power electronic and Drive System , IEEE, Vol.1, pp.428-431, July, 1999.

[2] H. Kim, H. Lee and S. Sul, “A New PWM Strategy for Common Mode Voltage Reduction in Neutral Point Clamped Inverter Fed AC Motor Deriver”, Transactions on Industry application, IEEE, Vol.37, No.6, pp.1840-1845, November, 2001.

[3] T. F. Lowery and D. W. Petro, “Application for the PWM Inverter Fed Low Voltage Induction Motor”, Transactions on Industry application, IEEE, Vol. 30, No. 2, pp.286-296, March, 1994.

[4] J. Chen and C. Chu, “Combination Voltage Controlled and Current Controlled PWM Inverter for UPS Parallel Operation”, Transactions on Power Elec., IEEE, Vol.10, pp.547-558, September, 1995.

[5] A. N. Rahmin and J. E. Quaicoe, “Analysis and Design of Multiple Feedback Loop Control Strategy for A Single Phase Voltage Source UPS Inverters”, Transactions on Power Elec., IEEE, Vol.11, pp.532-541, July, 1996.

[6] A. E. Ginart, R. M. Bass, W. M. Leach and T. G. Habetler, “Analysis of the Class AD Audio Amplifier Including Hysteresis Effects”, Transactions, IEEE, Vol.18, No.2, pp.679-685, March, 2003.

[7] N. Mohan, T. M. Undeand and P. Robbins, "Power Electronics-Converters: Applications and Design", 2nd Edition, John Wiley& Sons, Inc. Ltd. Co., 1995.

[8] M. Trigg, H. Dehbonei and C. V. Nayar, "Digital sinusoidal PWMs for a Micro-Controller Based Single-Phase Inverter, Part1: Principles of Digital Sinusoidal PWM Generation", International Journal of Electronics, Vol.95, pp.819-840, August, 2008.

[9] Y. Xue, K. Baekhy and J. Bordonau, "Topologies of Single Phase Inverter for Small Distributed Power Generators: An Overview", Transactions, IEEE, Vol.19, No.5, September, 2004.

[10] S. Musumeci, A. Raciti, A. Testra, A. Galluzo, and M. Melito, “Switching Behavior Improvement of Insulated Gate-Controlled Devices", Transactions on Power Elec., IEEE, Vol.12, No.4, pp.645-653, 1997.

[11] S. Y. R. Hui, Y. K. E. Ho and H. Chung, “Modular Single-Stage, Three-Phase Full Bridge Converter with Inherent Power Factor Correction and Isolated Output”, Transactions on Power Electronic Application, IEEE, Vol.145, No.4, pp.407-414, July,

1999.

[12] W. C. M, W. H. Lau and H. Chung, “Analytical Technique for Calculation the Output Harmonic of H bridge Inverter with Dead Time”, Transactions on Fundamental Theory and Application , IEEE, Vol.26, No.5, pp.617-627, May, 1999.

[13] J. S. Choi, Y. Yong, W. Lim and S. Young, “A Novel Dead Time Minimization Algorithm on the PWM Inverter”, In the Proceeding of the Industrial Application Conference, IEEE, Vol.4, pp.2188-2193, 3-7 October, 1999.

[14] C. Matthew, D. Hooman and C. V. Nayar, "Digital Sinusoidal PWM Generation Using a Low–Cost Micro-Controller Based Single–Phase Inverter", IEEE Transactions, Vol.1, pp.390-396, September, 2005.

[15] C. Rech, H. Pinheiro, A. Grundling, H. Hey and J. Pinheiro, "Analysis and Design of a Repetitive Predictive–PID Controller for PWM Inverter", In the Proceedings of Power Electronics Specialists Conference, Vancouver, Canada,Vol.2, pp.986-991, 17-21 June, 2001.

[16] Sh-2Sh7047 Group Hardware Manual, Accessed in September 2009, http://www.renrsas.com.

[17] K. Schultz, “An Application of the Low Cost Hitachi SH-1 RISC Controller for PID Control of a Three-Phase Brushless DC Motor System", Hitachi Application Note, Preliminary V0.1, PMH11IA05D1, September 1997.

[18] T. Abeyasekera, C. M. Johnson, D. J. Atkinson and M. Armstrong, “Elimination of Sub harmonics in Direct Look-Up Table (DLT) Sine Wave Reference Generators for Low-Cost Microprocessor-Controlled Inverters”, Transactions on Power Electronics, IEEE, Vol.18, No.6, pp. 1315-1321, August, 2007.

[19] H. Dehbonei, L. Borle and C. V. Nayar, “Design and Implementation of a Low Cost Sine Wave Inverter”, In the Proceedings of IEEE International Symposium on Industrial Electronics, Vol.1, pp.280-285, 9-11 June, 2003.

[20] D. Czarkowski, D. V. Chudnovsky, G. V. Chudnovsky and I. W. Selesnick, “Solving the Optimal PWM Problem for Single-Phase Inverters”, Transaction on Circuits and Systems, IEEE, Vol.49, No.4, pp.465–475, April, 2002.

[21] C. Lin, F. Wang, B. Wang and Q. Zheng, “A New Method of Optimized PWM Based on 8XC196MC”, In the Proceedings of the Third International Power Electronics and Motion Control Conference, Beijing, China, Vol.3, pp.1266–1270, 16-18 August, 2000.

[22] T. L. Skvarenina, “The Power Electronics Handbook”, Industrial Electronics Series edition, J.D. Irwin Ltd. Co., New York: CRC., 2002.

[23] D. Heng, O. Ramesh and S. Dipti, “Modeling and Control of Single Phase UPS Inverters: A Survey”, In the Proceedings of Power Electronics and Drives System, Vol.2, pp.848-853, Oct. 28th - Nov. 1st, 2005.

[24] S. Ghazanfar, F. Jawad and S. Pegah, “Nonlinear Control Techniques in Uninterruptible Power Supply Inverter: A Review”, In the Proceedings of Second International Conference on Computer and Electrical Engineering, IEEE, Dubai, pp. 51-55, 28-30 December, 2009.

[25] R. Seungwan and C. Chulhyoi, “PI-PD Controller for Robust and Adaptive Queue Management for Supporting TCP Congestion Control”, In the Proceeding of the 37th Annual Simulation Symposium, 2004.

[26] W. K. Ho, T. H. Lee and Tay, “Knowledge-Based Multivariable PID Control”, Technical Notes, National University of Singapore, April, 1999.

[27] P. Seda, D. B. Emine and E. Kadir, “Temperature Control Using Autotuning PID Controller for Control Education”, In the Proceedings of the 5th WSEAS International Conference on Signal Processing, Robotics and Automation, Madrid, Spain, February 15-17th, pp. 131-134, 2006.

[28] M. Niroomand and H. R. Karshenas, “Review and Comparison of Control Methods for Uninterruptible Power Supplies”, In Proceeding of Power Electronic & Drive Systems & Technologies Conference (PEDSTC), IEEE, pp.18-23, May, 2010.

[29] H. Toshimasa, K. Atsuo and G. Richard, “Waveform Compensation of PWM Inverter with Cyclic Fluctuating Loads”, Transaction of Industrial Application, IEEE, Vol.24, No.00, pp.582-589, July, 1988.

CHAPTER 4
MODEL IN THE LOOP SIMULATION

4.1.INTRODUCTION

In developing embedded systems, the requirements for embedded software design are completely different in the case of software design in the business application field. In the embedded software design, ultra high reliability, real time process and hardware/software co-design are required. Because embedded software controls hardware, if the software is not accurate and the embedded system runs out of control, it may cause very serious accidents. For example, Toyota Prius, a hybrid car, had a sudden engine stall in the U.S.A. because of the bug of software, and many other troubles have already occurred in many products. As well as the real time characteristics of the embedded system which means that the process must be finished in the specified time period and almost all processes in an embedded system have such constraints in their executions. To satisfy these kinds requirements and, moreover, to minimize the cost of manufacturing of the embedded system, a hardware and a software must cooperate each other. In this chapter new design environment of the embedded software is proposed using the MATLAB and Simulink environment. This study applied the Model Based Design method for the development of the inverter power supply as a case study of the embedded system design. New virtual environment was used which is the Model In the Loop Simulation (MILS) environment. Both the hardware and software parts of the embedded system were modeled and tested using MATLAB/Simulink package. Detailed description of MILS environment is presented in this chapter.

4.2. MATLAB/SIMULINK ENVIRONMENT

To effectively design an embedded control system and accurately predict its performances, designers must understand the behaviors of the entire system in which control system will reside. It is attributed to the complexity of the embedded system design; the Model Based Design method is considered one of the leading solutions of the embedded system challenges. In this method, modeling and simulation are considered as vital parts of design. Particularly, the simulation is indispensable for developing a controller programs. It can lead to the performance improvement and the reduction of the cost and the time for development and production. There are varieties of software tools that are used for the

development of the embedded system. MATLAB and Simulink form the core environment for the Model Based Design method for creating accurate mathematical and functional model of the physical system behavior. The graphical block diagram paradigm of the MATLAB and Simulink environment lets the user to drag and drop predefined modeling elements, connect them together and create model of dynamic system. The dynamic system can be continuous time multi rate, discrete time or any combination of the three.

MATLAB and Simulink are software tools designed for modeling, simulating and analyzing the design of devices. It supports linear and nonlinear system models in continuous time, sampled time or combination of both. It consists of a set of blocks such as communications, controllers, power systems, neural networks, etc. There are a lot of available tools that can be used in the MATALB environment to design and optimize the performances in an effective and easy way. It is widely used in academic and industrial applications. In MATLAB and Simulink environment, the modeling environment can be hierarchical and self-documenting as presented in Figure 4.1 as well as the system structure and function can be expressed by grouping model [1, 2, 3].

4.3. MODEL IN THE LOOP SIMULATION, MILS

The topology of the inverter power supply is DC-DC-AC. MILS of the inverter process is done in two stages. The first stage is the DC/DC converter and in this stage, the input DC voltage is converted to higher level DC output. This new DC output acts as an input of the second stage which is the DC/AC inverter. The output of this stage can be used as AC power supply for any AC equipments. SH 7047 was used as a digital controller to control the generation of the PWM pulses and control the inverter power supply operation. Both the DC/DC stage and DC/AC stage is controlled individually as shown in Figure 4.2.

As mentioned before, the entire system was divided into two main parts; the controller part which is illustrated by the electric circuits and the control part or the microprocessor part. Most of previous works concentrated in the simulation of the controlled part and the software part is tested in the actual prototypes which can lead to system failures as well as it takes so much time and cost to obtain the optimum system behavior. Therefore, most of research attentions are concentrated on how to solve the above problems.

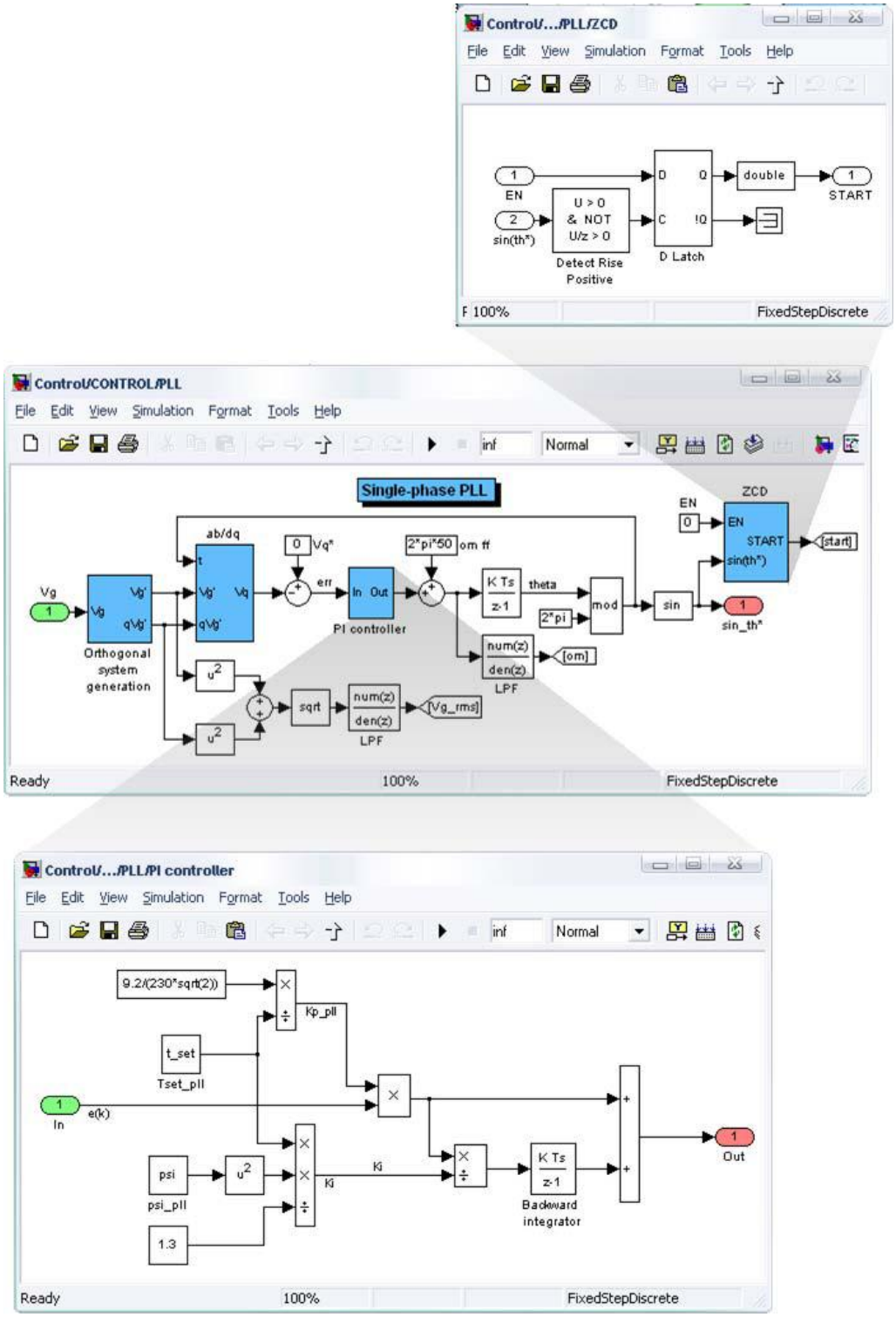


Fig.4. 1. Hierarchical models of complex control systems using Simulink

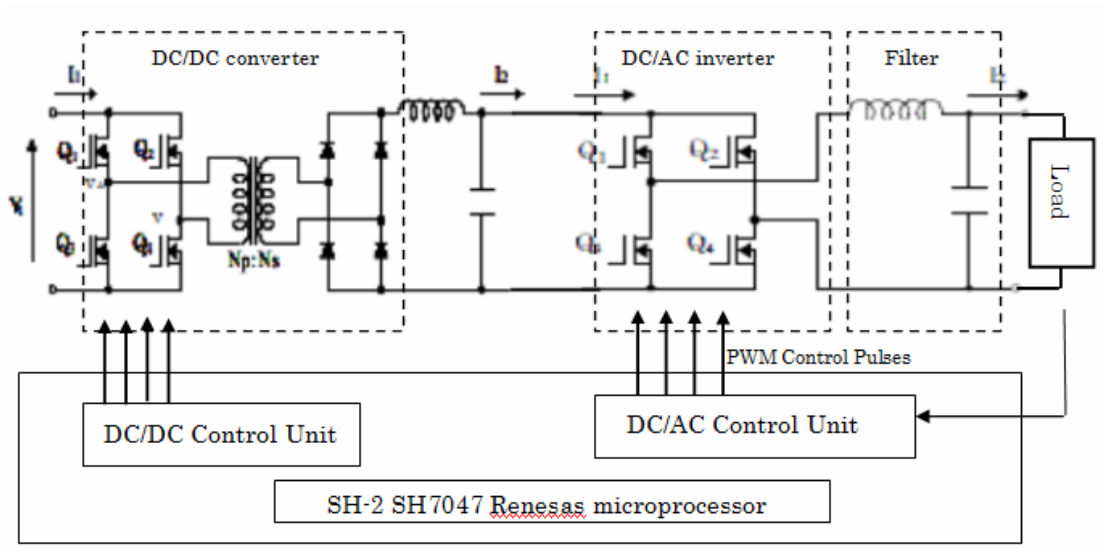


Fig.4. 2.Circuit schematic of inverter power supply

4.3.1 DC/DC and DC/AC Circuit Simulation

The hardware part or the electric circuits are modeled using Simulink power block set. The power block set consists of the power electronic elements as shown in Figure 4.3.

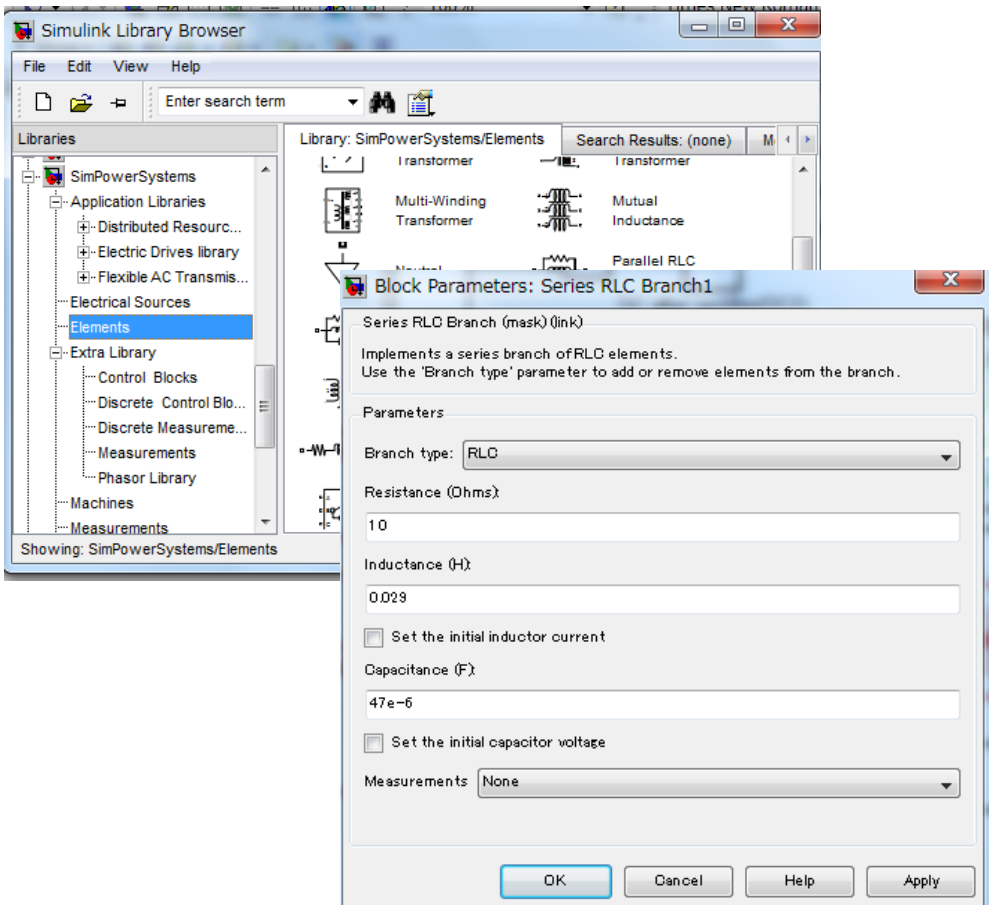


Fig.4. 3. SimPower system block set

Each element in this block set has its own block window which allows for the selection for the key parameters. At the bottom of the block parameter window is a pull down menu, which allows for the key voltage and current to be easily measured.

In the first stage, the electrical circuits was modeled in open loop system to determine the performance of each circuit separately. The PWM generation block was modeled for both DC/DC converter stage and DC/AC inverter stage. For example, the PWM model in the MTU microprocessor unit is illustrated in Figure 4.4. The PWM block compares both the sin wave signal with the required output frequency and the sawtooth signal with the carrier frequency equal to 10 kHz . Then the dead time is generated using the delay block and finally, the PWM pulse is generated and fed into the gates of the full bridge transistors.

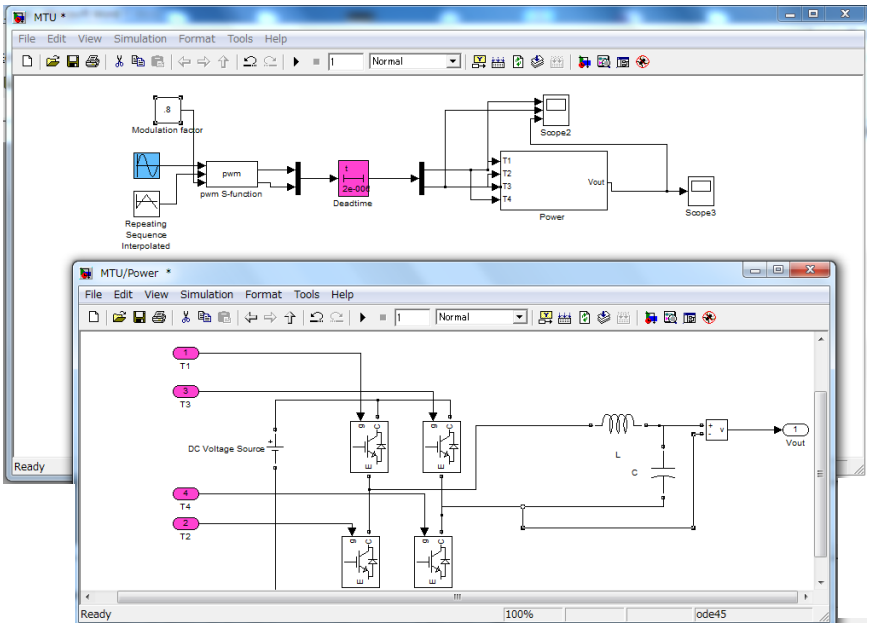


Fig.4. 4. PWM generator block

The modulation ratio, the carrier frequency, the reference frequency were changed manually. Equation 4.1 describes the relation among all three parameters.

$$m = V_C / V_{carrier} \quad (4.1)$$

Where:

m: modulation index,

V_C: amplitude of the reference voltage,

$V_{carrier}$: amplitude of the carrier signal.

In the DC/AC stage, the carrier frequency is set equal to 10kHz and the reference frequency equal to the desired output frequency of 60 Hz.

By changing the modulation factor, the duty of the PWM pulse is changed as described previously in Chapter 3. The electric circuit specifications of the two stages of the inverter power supply is described in Table 4.1

Table 4. 1: Specifications of the inverter circuits

Parameter	Value	Unit
Output frequency	60	Hz
Output Voltage	100	RMS
Battery Voltage	24	V
Switching frequency	10	kHz
Filter inductor DC/DC	29	mH
Filter capacitor DC/DC	47	µF
Transformer turns ratio	7.2	
Filter inductor DC/AC	3	mH
Filter capacitor DC/AC	25	µF

The circuit's parameters are modeled and tested in the open loop model without the controller as shown in Figure 4.5. The signal is monitored using the scope and display blocks which allow us to figure the circuit outputs as marked with dashed circles as well as the results can be sent to the MATLAB workspace for analysis.

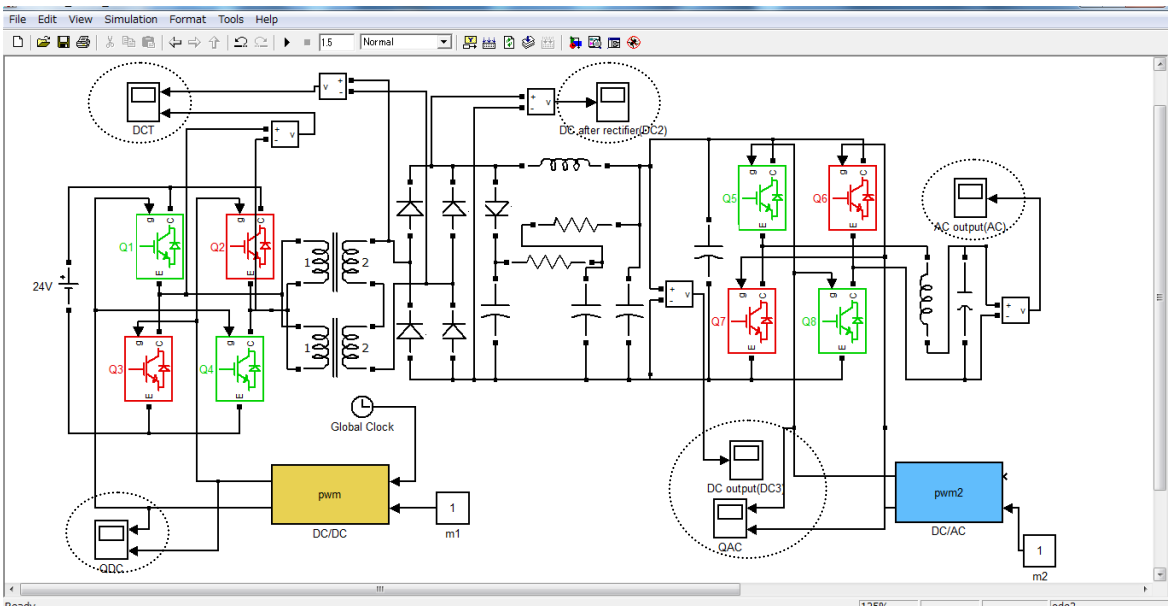


Fig.4. 5. MATLAB model of open loop system

4.4. MICROPROCESSOR SIMULATION

In the inverter power supply applications, the microprocessor is used to control the switching period of the transistors as a digital controller. As described in Chapter 3, SH7047 is used. This microprocessor has two main units which used to control both DC/DC and DC/AC stages of the inverter power supply. Each unit is controlled by a software program; this embedded software function can be divided into two main parts which are the control program (control algorithm) and the PWM generation program.

4.4.1. Control Algorithm

Traditionally, the implementation of switching type inverter power supply has been accomplished by using analog technique. However, the analog technique has some drawbacks such as a number of parts required in the system and its susceptibility to aging and environment variations, which lead to high cost of maintenance. Further, analog control once designed is inflexible and its performance cannot be optimized for various utility distortions. Now with the advent of high speed, lower cost digital signal processing (DSP) ICs, and microprocessors, digital control has been one effective candidate for inverter power supply.

Using a DSP or microprocessor has many benefits that make it attractive for use in control systems such as:

- Flexibility of Control: When using analog circuits to perform control, the control algorithm is fixed, and is not easily modified. Using a microprocessor allows the designer to change the control code very quickly. It is often helpful to implement simple, slow control algorithms first to verify that the hardware is functioning correctly before moving to a higher performance or complex control algorithm. If hardware were used to do this, this would mean separate hardware designs and implementations for each algorithm. With the use of software, modifying the control algorithm means changing several lines of codes, which will take only several minutes.
- Parameter Adjustment: Once the control algorithm is fixed, it is easy to modify the values of references and constants in the control code by directly modifying memory locations. This can be performed while the system is operational, allowing for quick adjustments to be made. If the control algorithm were implemented in analog hardware, this would not be as easy to do such process.

- Backtracking: The use of software allows for easy backtracking when the control algorithm is not working. If the control were implemented in analog circuits, the physical modifications would need to be reversed, which may also introduce additional errors in the process.

Many control techniques have been applied to the inverter power supply as mentioned in Chapter 3. In this study, we newly apply a controlling algorithm which is two layer control. This algorithm is a combined feedforward control and feedback control. This control algorithm can significantly improve performances over simple feedback control whenever there is a major disturbance that can be measured before it affects the process output. The basic configuration of the proposed controlling algorithm is shown in Figure 4.6.

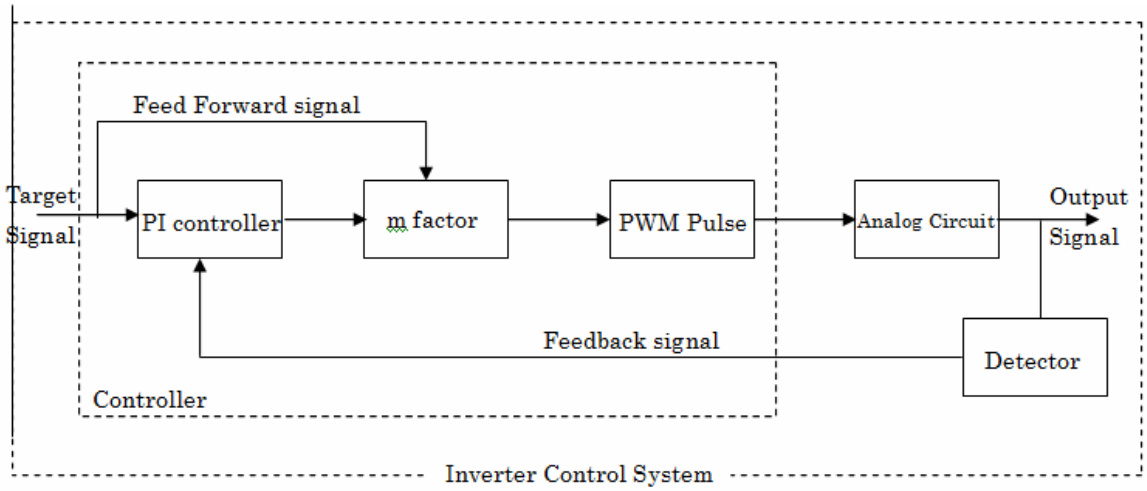


Fig.4. 6. Simplified block diagram of the proposed control system

PI (Proportional Integral controller) control algorithm has been one of the more utilized control techniques in the industry [4, 5]. It has been proved its wide range of applications and it is first introduced to the market in 1939 [6]. The main reasons for using PI controller are its simple structure, easy to implement in the practical applications and its flexibility.

To improve the accuracy of both the steady state response and transient response and to minimize the output disturbance, a feedforward control is added to the classical PI controller as shown in Figure 4.6. Then the controller equation can be described through equation (4.2).

$$u_{pi} = K_p e(k) + K_i T \sum_{n=1}^{k} e(n) + r(k) \tag{4.2}$$

Where: K_p and K_i, are the Proportional, Integral coefficient, respectively, u_{pi} is controller output and e(k) is the error signal and r(k) is the feedforward signal. The controlling algorithm is described in the flowchart which is presented in Figure 4.7.

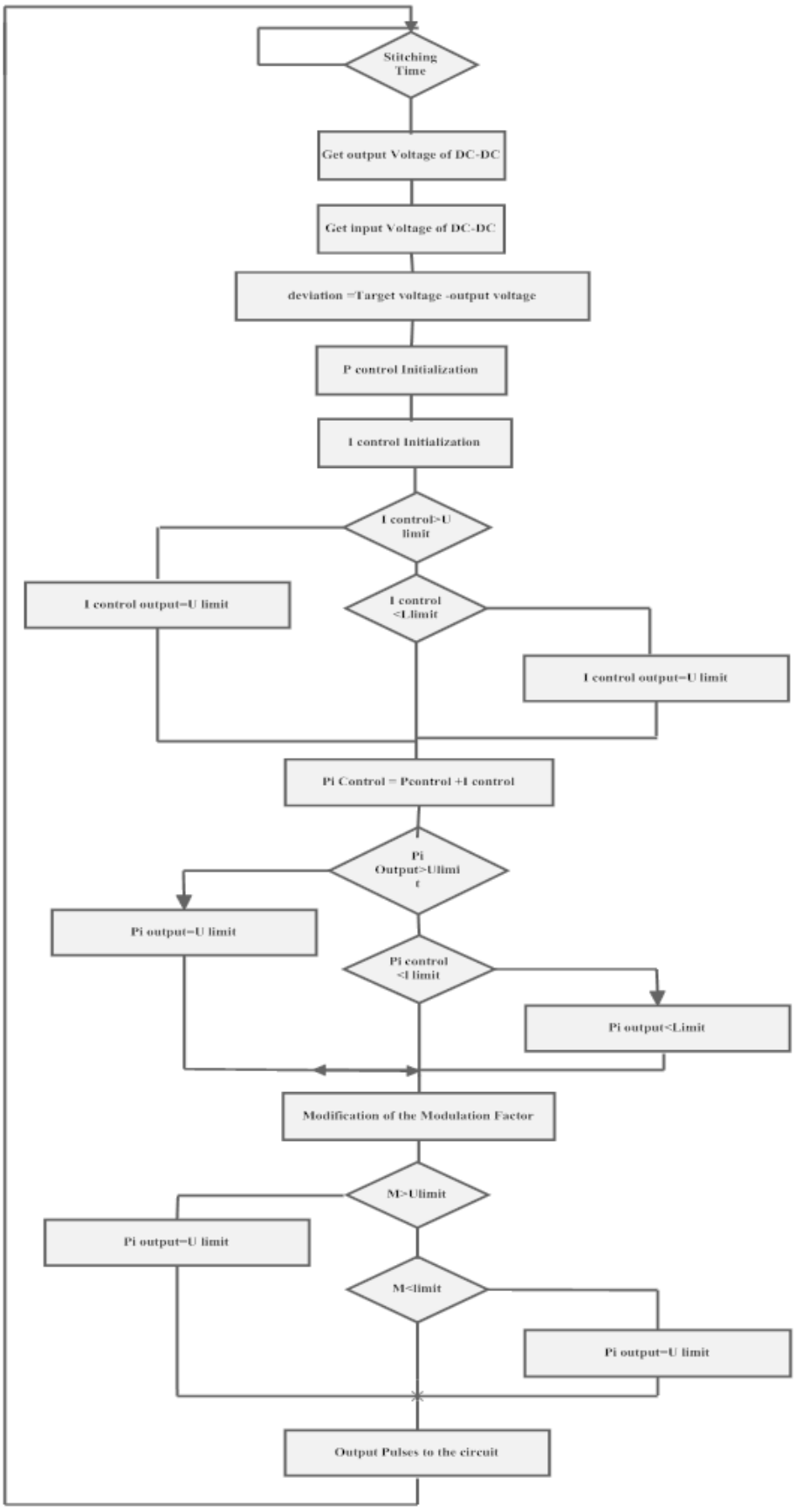


Fig.4. 7. PI controller algorithm

When the switching time is coming, the deviation between the target output and the actual output is calculated and this deviation is used to modify the controller output. The output of the PI controller must be within specific interval in order to protect the circuits. The modulation factor m is calculated with respect to the controller output.

The controller algorithm was applied to both the DC/DC converter and DC/AC inverter separately and it was implemented using the SH7047 microprocessor. In the MILS environment, S-Function Block is used as the interface block to test the embedded software and to optimize the parameters. Details of the S-Function will be described in the next section. Each of the inverter power supply stages has its own C-MEX file which describes part of the embedded software. This C-MEX file is compiled to the S-Function Block and then the output of the S-Function block acts as the simulation of the microprocessor unit.

4.4.2. PWM Generation Program

The standard method for generating PWM pulses by using a microprocessor or DSP is to utilize one of the built-in PWM modules. In our model, S-Function Block is used to compile the embedded system software to control DC/DC and DC/AC circuits. C programs implement the operations of the microcontroller's control units, that is, MMT which controls the generation of the PWM pulses in the DC/DC and MTU which controls the generation of the PWM DC/AC stage. These programs are embedded into the S-Function Block as shown in the lower part of Figure 4.10 (p.69).

4.5. S-FUNCTION

S-Function is the abbreviation of the System Function which provides a powerful mechanism for extending the capabilities of the Simulink environment. It is a computer language description of the Simulink block written in MATLAB, C, C++, and/or FORTRAN. S-Functions are compiled as MEX-files using the MEX utility [1]. S-Function uses a special calling syntax called the S-Function API (Application Program Interface) that enables the user to interact with the Simulink engine. The interaction is very similar to the interaction that takes place between the engine and the built-in Simulink blocks. S-Function follows a general form and it can accommodate continuous, discrete and hybrid systems. It allows the user to implement different algorithm and add them to the Simulnik model.

Two S-Function Blocks are designed: one to describe the DC/DC control unit which is the MMT unit and the other S-Function is used for the MTU unit which controls the second stage, DC/AC stage. The embedded C codes are compiled to the S-Function Block and the details of the embedded software are presented in Appendix I. In the first step, all the electrical circuits' parameters are tested and then, in the second step, the embedded software is tested using the S-Function.

Drag an S-Function Block from the user defined function block library into the model, then specify the name of the S-Function in the S-Function parameter block as illustrated in Figure 4.8.

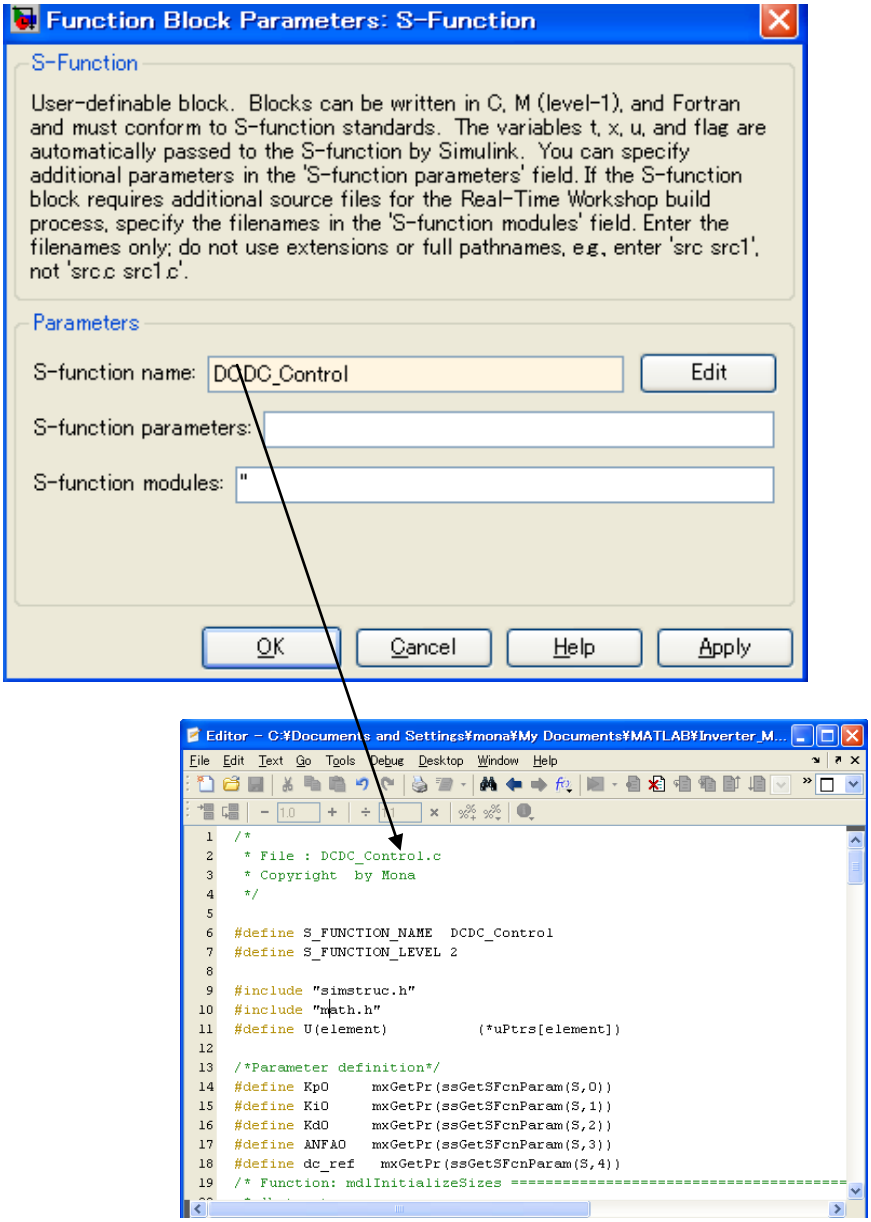


Fig.4. 8. Block parameter dialog box of DC/DC S-Function

The S-Function and the C-MEX file have the same name and if the MATLAB path includes a C-MEX file and M file having the same name, the S-Function uses the C-MEX file. After the S-Function name is set, then the S-Function parameters are defined. The parameters include the PI controller parameters which are the K_p and K_I, the proportional coefficient and the integral coefficient. Further the dead time value which is T_D is set. The parameters and the name of the S-Function is defined using the function block parameter then the main C code can be generated , by pressing the Edit button the user can modified the embedded software. The value of the S-Function parameters is initialized in the C-MEX file, and then the updated values can be tested using the S-Function Block parameters for both DC/DC S-Function and DC/AC S-Function as shown in Figure 4.9.

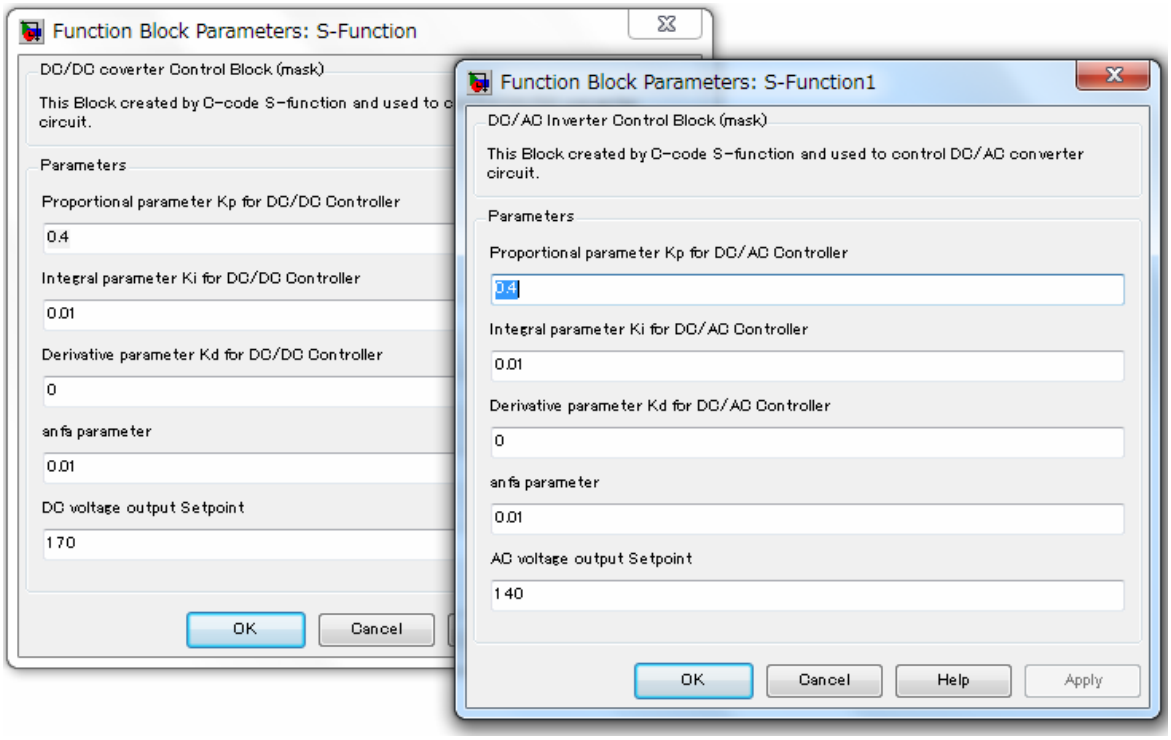


Fig.4. 9. S-Function parameter block

The data can be displayed and sent to the MATLAB workspace for further analysis by using the scope blocks. The scope block is used to measure and monitor the signal at each point in the model, which allows the user to check the model operation at each point as shown in Figure 4.10. This figure presents the entire inverter power supply model which is the MILS environment. The embedded software parameters are optimized and the system performance is tested as well as the verification of the control algorithm is done.

The C-MEX files and DC/DC controller which simulate the MMT function and the C-

MEX DC/AC which simulates the MTU function controlling the operation of the inverter stage are attached in Appendix I.

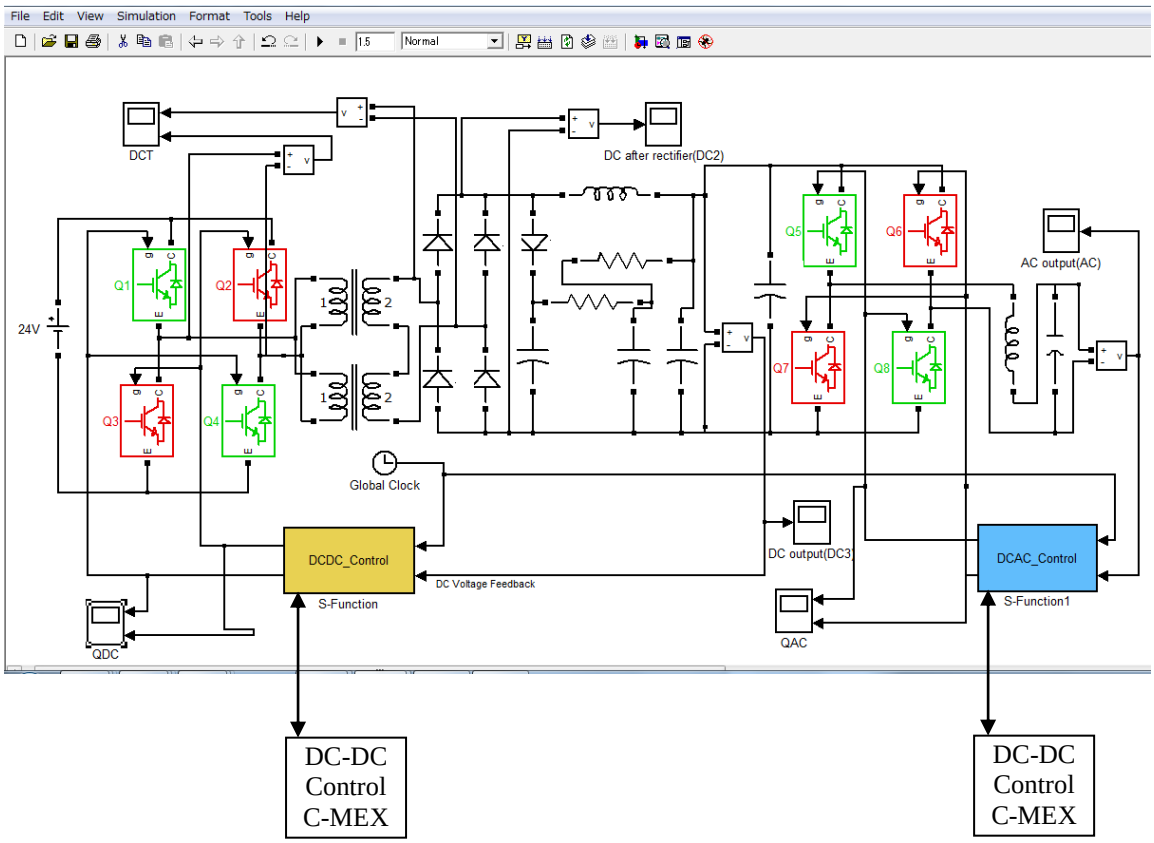


Fig.4. 10. Model In the Loop Simulation of inverter power supply

REFERENCES

[1] The MathWorks web site, Accessed in December, 2010. http://www.mathworks.com/.

[2] A. Tewari, "Modern Control Design With Matlab and Simulnik", John Wiley & Sons Ltd. Co., 2002.

[3] O. Beucher and M. Week, "Introduction to Matlab and Simulnik", Infinity Science Press LLc, Co., Ltd., New Delhi, 2006.

[4] M. A. El Dahb, Y. Shirashi and T. Shoji, "Simulation Based Design of The inverter Power Supply", Transactions on Mathematical Modeling and Application, IPSJ, Vol.3, pp.186-193, 2010.

[5] M. A. El Dahb, S. Iino, Y. Shirashi and M. Tatsumo, "Model Based Design of The inverter Power Supply", In the Proceeding of ICCAS-SICE International Joint Conference, Fukouka, Japan, August 17-21th, 2009.

[6] W. K. Ho, T. H. Lee and Tay, "Knowledge-Based Multivariable PID Control", Technical Notes , National University of Singapore, April, 1999.

CHAPTER 5
EXPIRAMINTAL RESULTS AND DISCUSSIONS

5.1. INTRODUCTION

The validity and usefulness of the MILS environment constructed is verified by comparing its results with the experimental results in the actual device. As shown in Figure 5.1, the actual prototype of the inverter power supply has been fabricated with the cooperation of Tokyo Seiden Co., Ltd. The full-bridge topology was chosen due to its power handling capabilities and the ability to provide PWM and thus effectively double the switching frequency of the output as mentioned previously in Chapter 3. Renesas SH7047 microprocessor was used as a controller that controls the operation of the electric circuits and the embedded software was implemented manually. The details of the actual embedded system software are described in Appendix II. The input DC voltage is 24V and the target AC output is 100 RMSV with frequency of 60 Hz. RMS is the Root Mean Square value of the output voltage.

Fig.5. 1. Actual inverter power supply device

In the MILS environment, we assumed that all components used are ideal components. So in the real implementation, some protection circuits were added to the inverter power supply design as well as monitors and switches. The actual circuit inverter power supply is described in Figure 5.2. The lower part describes the microprocessor units MMT and MTU which generate the PWM pulses to the MOSFET gates.

All the software parameters and the control algorithm are optimized in the virtual environment and then the entire software is tested in the actual prototype. Here, the software parameters include the PI coefficients, the value of modulation ratio, and the dead time value.

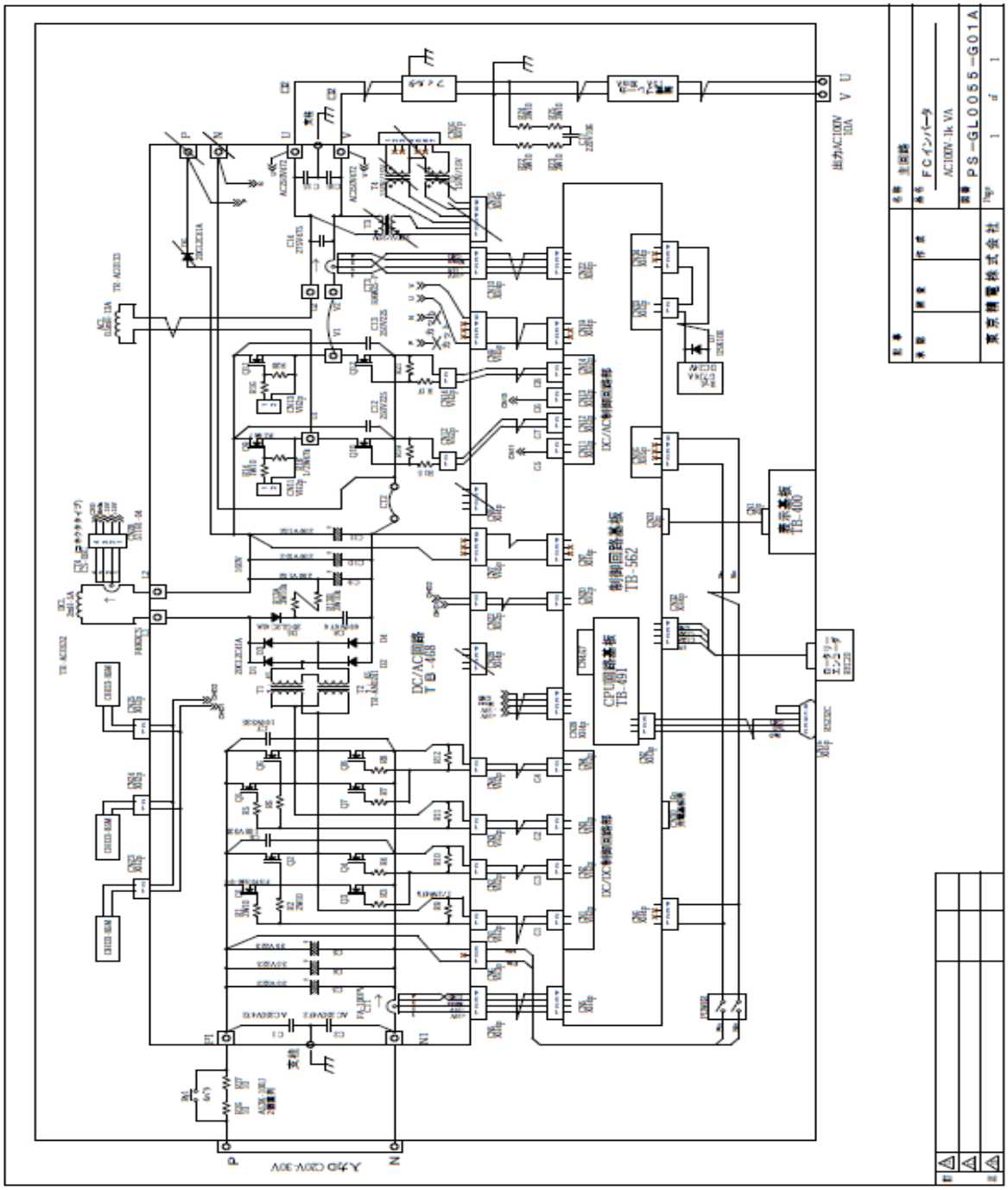


Fig.5. 2. Actual prototype circuits for the inverter power supply

5.2. DC/DC CONVERTER PERFORMANCES

The heart of this inverter power supply is the SH7047 microprocessor. This controller was utilized to perform the control algorithm to supervise the output voltage and to control the generation of the PWM pulses. After the parameters of the software are optimized, the performance is tested in the actual prototype. As mentioned before the SH7047 microprocessor has two units that produce PWM pulses which control the stages of the

inverter power supply. The output of each stage in the MILS is tested and compared with the actual output. DC/DC stage was controlled by the MMT output pulses. Figures 5.3 and 5.4 present the output of the MMT unit which controls the switching of the MOSFET in the DC/DC stage in the actual prototype and MILS environment, respectively.

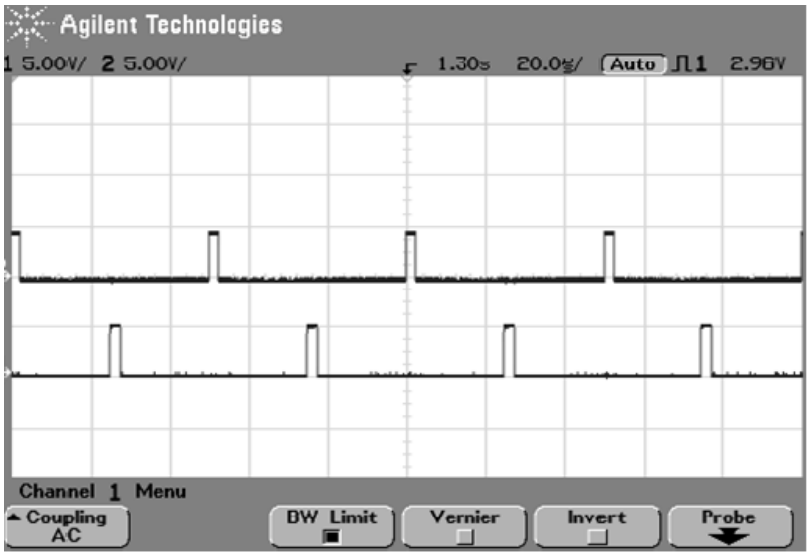


Fig.5. 3. Actual control pulse in the DC/DC

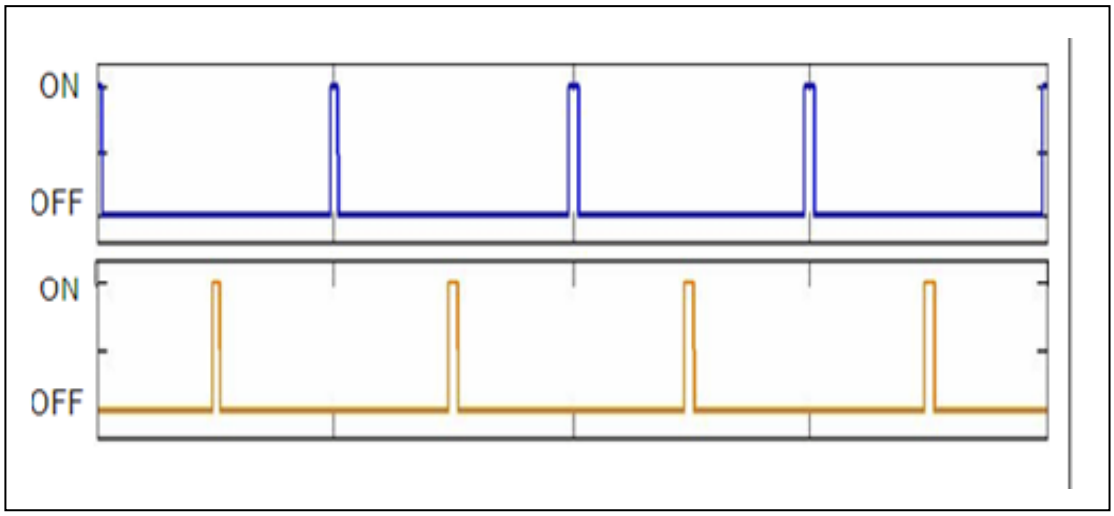


Fig.5. 4. Simulated control pulses in DC/DC stage

These pulses are applied to the gates of the MOSFETs of the full bridge converter and then the output of the full bridge is applied to the transformer to step up the voltage as shown in Figure 5.5. After the transformer, the output is applied to the full bridge rectifier and then low path filter. The output of the DC/DC stage was tested in the MILS

environment for the range of input voltage from 24 to 30 DCV. It is shown that steady no load voltage about 165 V as shown in Figure 5.6. DC/DC stage is performed well and it can be seen that the results obtained by the developed simulator are in a good agreement with the actual results. So the developed model can accurately model the actual prototype. Moreover, the developed MILS environment is good to optimize the parameters of embedded software programs. The output of the DC/DC stage cannot be tested in the actual prototype because of the circuit complications. Therefore, we tested the DC/DC performance stage in the virtual environment only and its performances in the actual prototype can be judged due to the entire output of the inverter power supply.

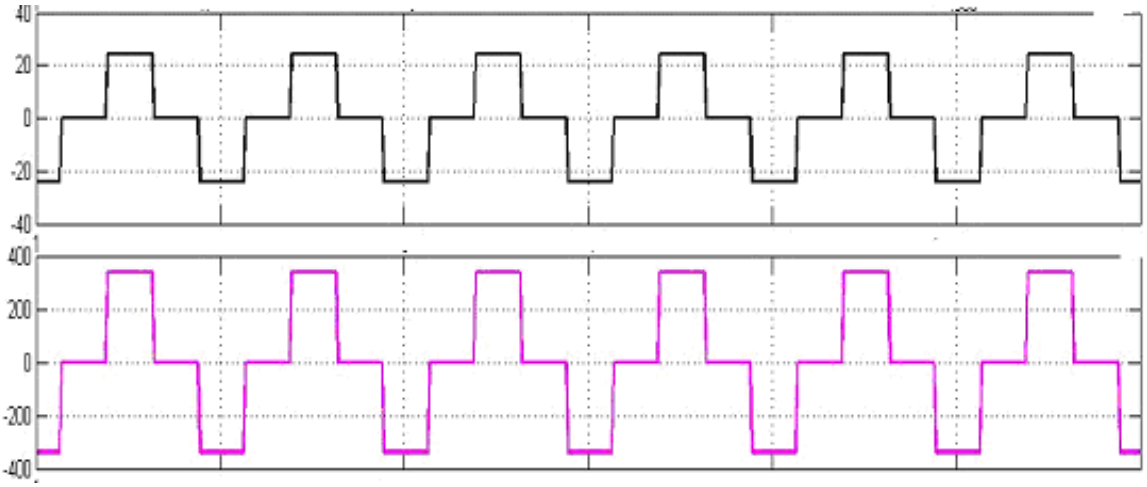


Fig.5. 5. Principal of operation in DC/DC converter

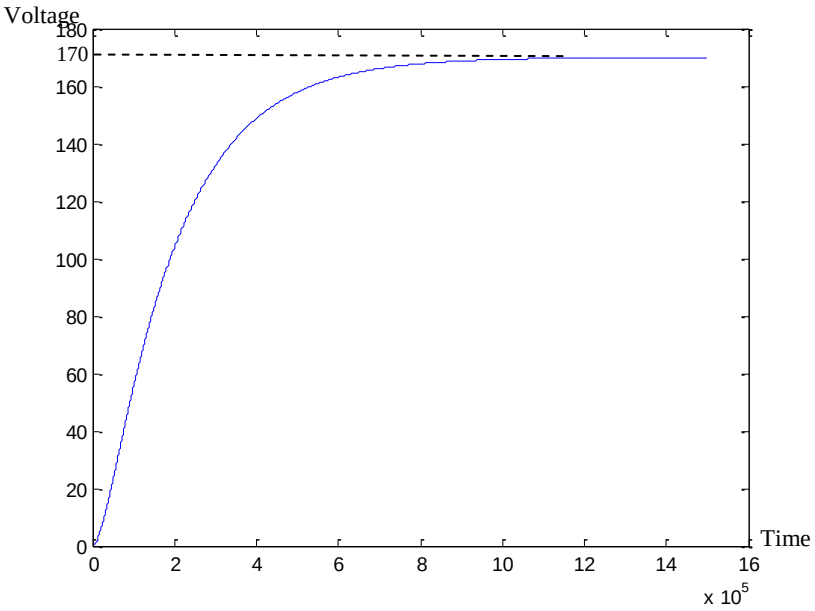


Fig.5. 6. Simulation result of DC/DC converter

5.3. DC/AC INVERTER PERFORMANCES

The function of this stage is to invert the new DC level which comes from the DC/DC converter stage to the desired AC voltage. The output of this stage is tested considering many factors.

5.3.1. The Time for Reaching the Stationary Voltage

Many trials were done to get the optimum values of the PI coefficient and some of the trials are described below. Table 5.1 shows three cases for different values of the controlling parameters.

Table 5. 1: The three different cases a, b and c of the controlling parameters

PI Parameter	A	B	c
K_p	0.4	0.5	0.4
K_I	0.02	0.01	0.01

The time needed for the inverter power supply to reach its stationary voltage is a very important factor in the inverter power supply design. By optimizing the software parameters accurately, this time can be decreased. Figure 5.7 describes the AC output of the inverter power supply in the simulation environment case (a) and Figure 5.8 shows the output of the same case in the actual prototype. In this case the inverter power supply reaches its stationary values at about 900 msec. With the same value of the software as in case (a), the actual prototype is tested as shown in Figure 5.8.

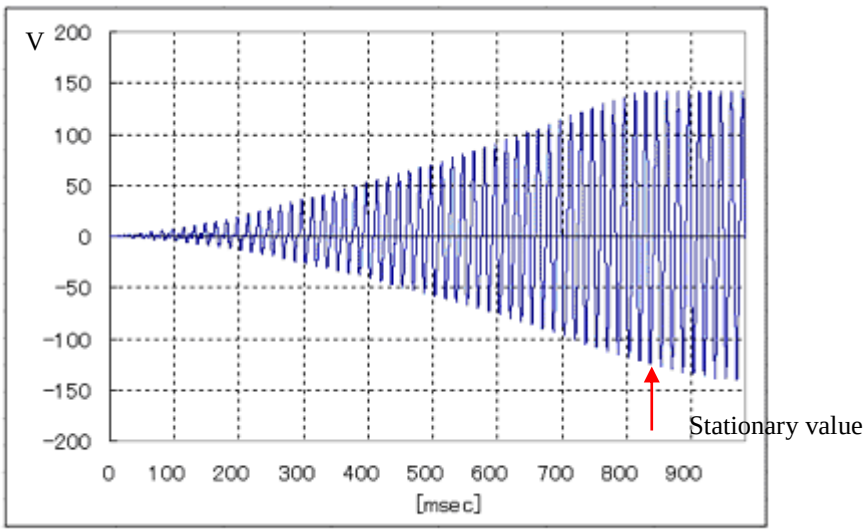


Fig.5. 7. Simulated AC output in case (a)

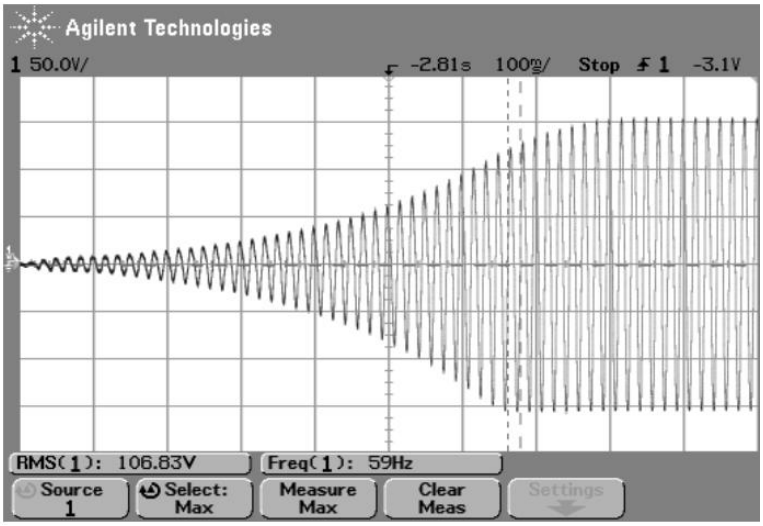


Fig.5. 8. Actual AC output in case (a)

The simulated and actual outputs in case (b) are presented in Figures 5.9 and 5.10, respectively.

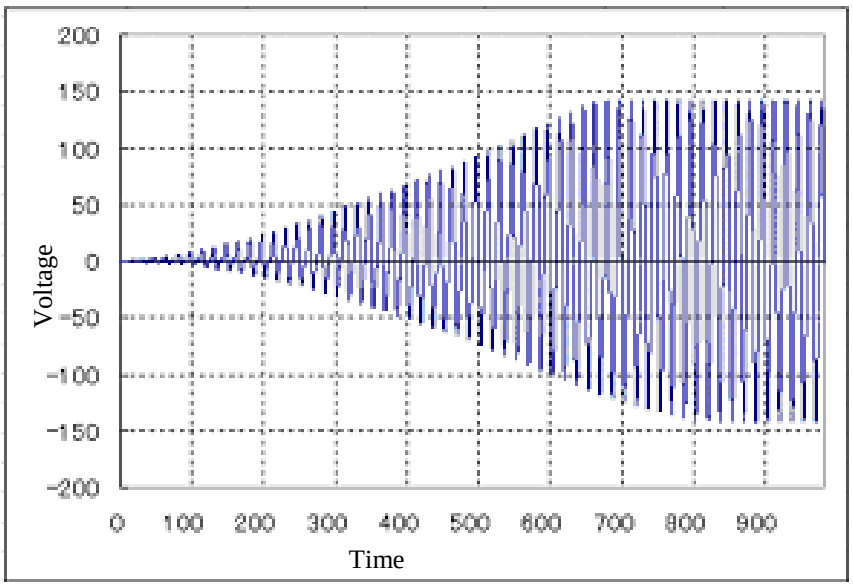


Fig.5. 9. Simulated output in case (b)

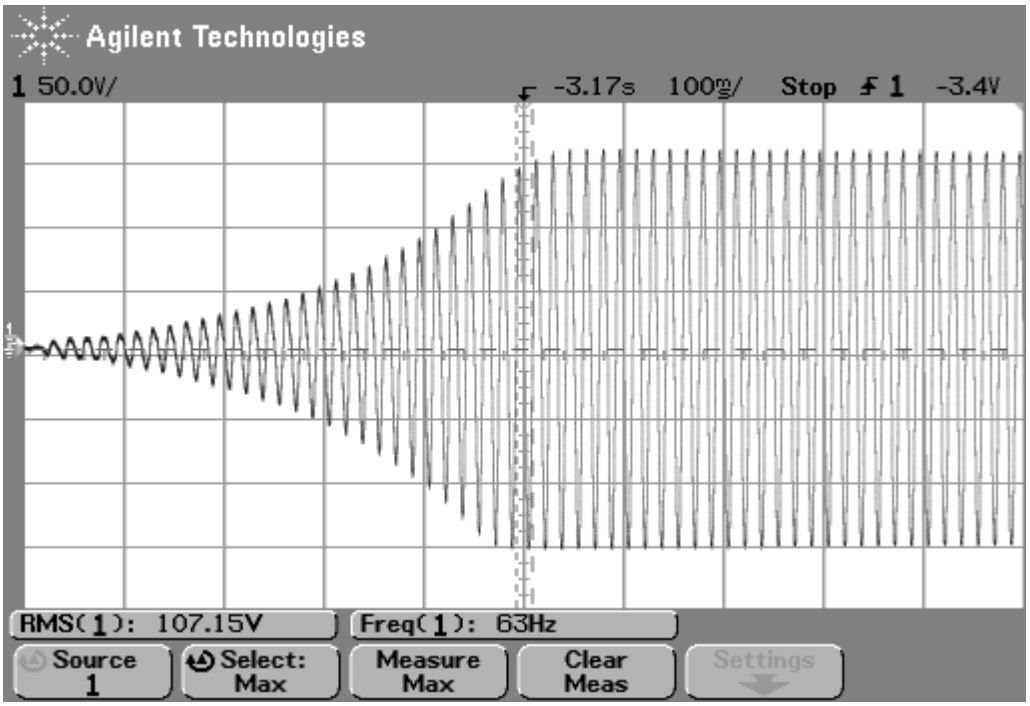


Fig.5. 10. The actual output in case (b)

Here is both the virtual environment and the actual prototype when applying in case (c) as shown in Figures 5.11 and 5.12, respectively.

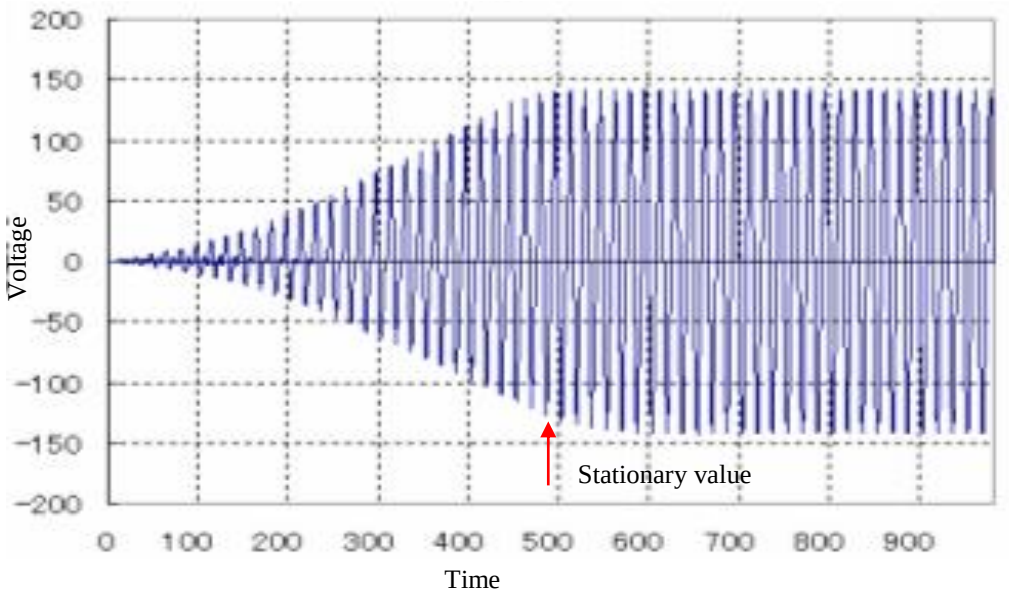


Fig.5. 11. The simulated result in case (c)

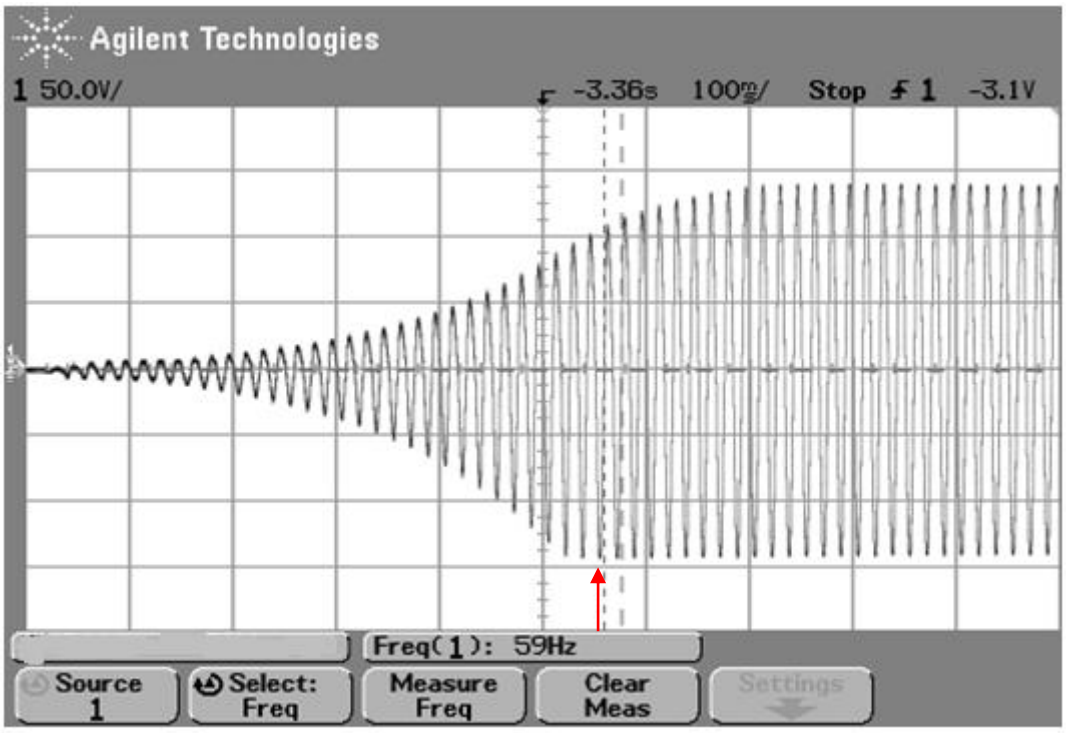


Fig.5. 12. The actual output in case (c)

By comparing the results of the MILS environment with the actual results, we can conclude that the values of the PI control parameters are optimum as shown in case (c). From these figures, it is clear that the shapes of the wave are visually identical. The actual device reaches the desired stationary voltage after 550 msec., while the MILS environment reaches the desired stationary AC voltage after 522 msec. It means that the simulation time is less than the actual time by about 5.4%. We can conclude that the developed model can simulate the actual prototype in a good way within very small difference comparing to the traditional method of design. Further, the MILS environment is a good tool that can be used to optimize the software parameters.

5.3.2. Pure Sine Wave Output Voltage and Frequency

All the software parameters which are optimized in MILS environment are described in Table 5.2.

Table 5. 2: Inverter power supply parameters

Parameter	Kp	KI	Dead time	Carrier frequency	Required frequency	RMS
Value	0.4	0.1	2µ sec	10 K Hz	60 Hz	100 V

Those parameters are tested in the actual prototype. Figures 5.13 and 5.14 shows the actual sin wave output and the simulated sin wave output, respectively. By comparing the two waveforms, it can be seen that by applying the proposed control algorithm, the efficiency of the inverter power supply output reached can be improved. The output voltage meets the standard requirement of the inverter power supply. Moreover, the efficiency of the proposed model is 99% which is calculated by comparing the difference between the actual and simulated peak voltage. So, the developed model can accurately model the actual prototype. As well as, the developed MILS environment is good to optimize the parameters of embedded programs.

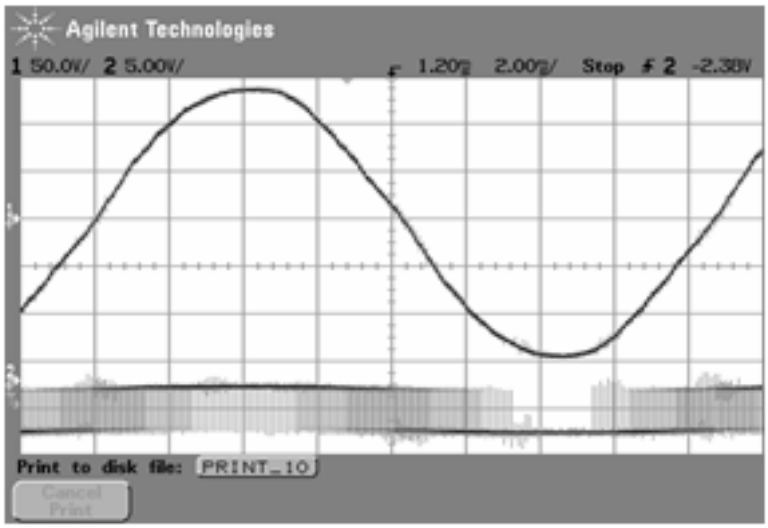


Fig.5. 13. Actual sin wave output of inverter power supply

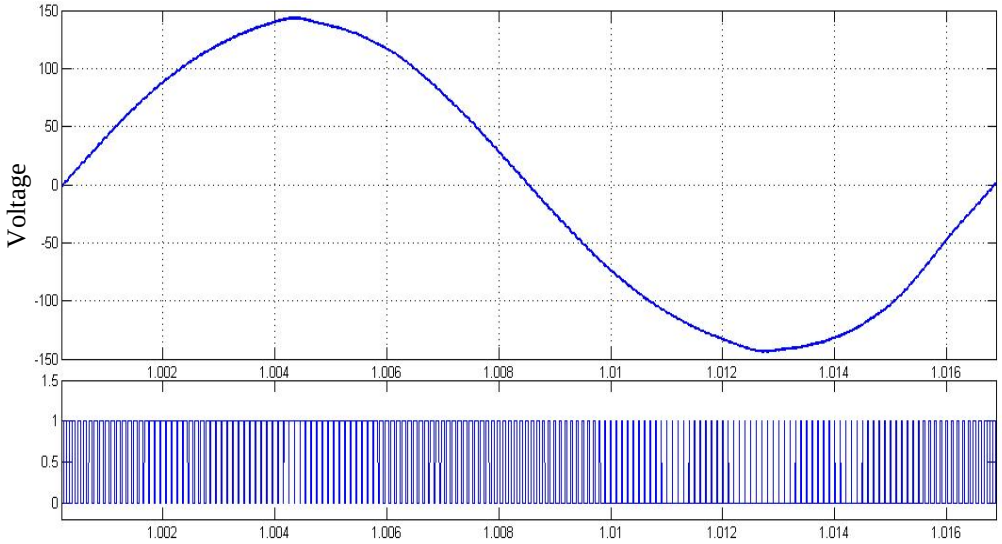


Fig.5. 14. The simulated sin wave output of inverter power supply

From Figure 5.15, it is apparent that the narrow pulse is generated when the modulating signal is at its maximum or minimum values. This notification can be explained by mathematical analysis of the PWM process as presented below.

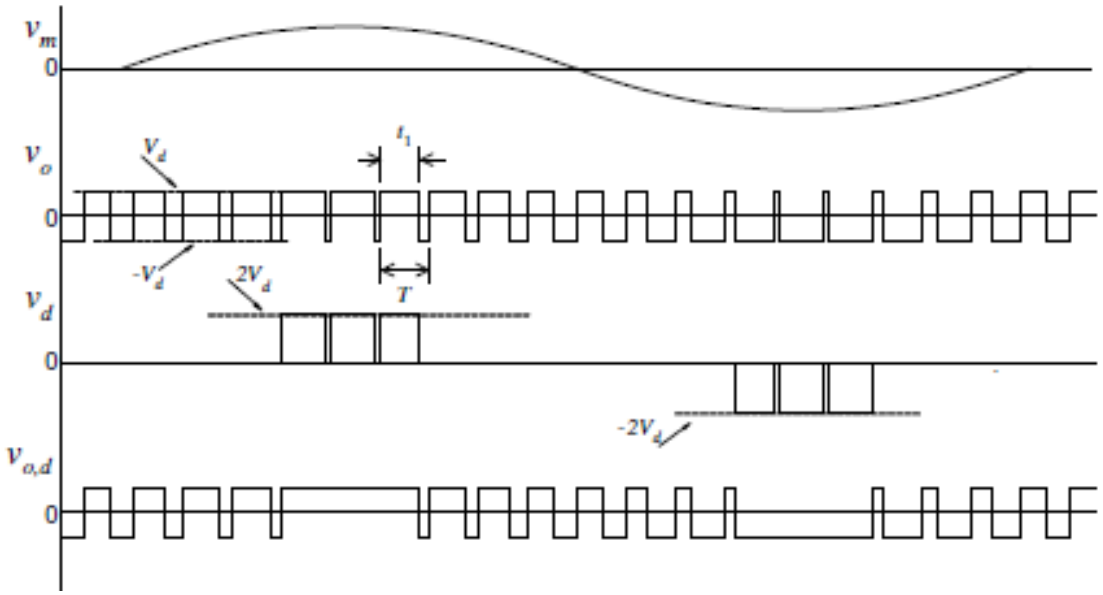


Fig.5. 15. Relation between the modulating signal and the generated pulses

Consider a switching cycle shown in Figure 5.15, when v_m is at the maximum value the average of V_o of the pulse is equal to:

$$V_o = (2d-1)V_d \Rightarrow d = \frac{1}{2}(1+m) \tag{5.1}$$

Where: d is the ratio of the duration of the pulse at positive value (i.e., t_1) to the period T, V_d is the DC voltage and m is the modulation ratio as defined in Chapter 3.

The minimum pulse duration t_{min} can be calculated from the following equation:

$$t_{\min} = [1-\frac{1}{2}(1+m)]T = \frac{1}{2}(1-m)T \tag{5.2}$$

From this equation, it is apparent that t_{min} decreased as m increased. We can conclude that, when the modeled signal reached the maximum value the narrow pulse is produced. Figures 5.16 to 5.18 illustrate the actual pulses taken from SH microprocessor output and the MTU output pulse from the MILS environment, respectively.

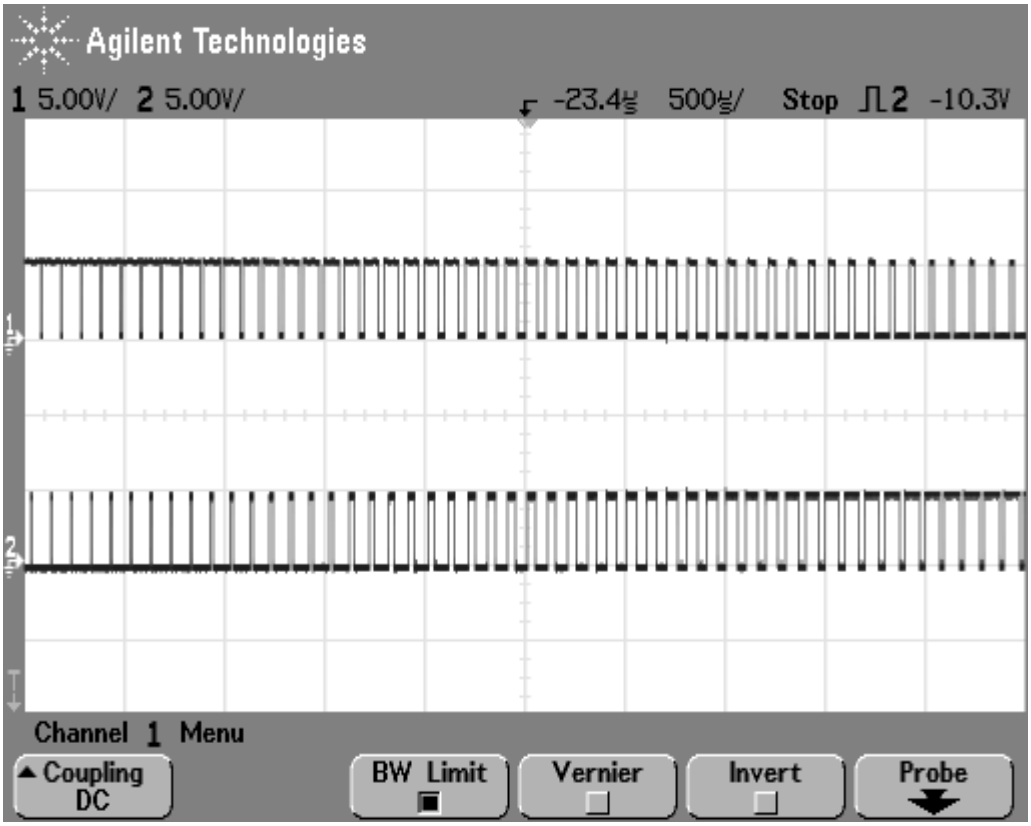


Fig.5. 16. The actual control MTU pulses

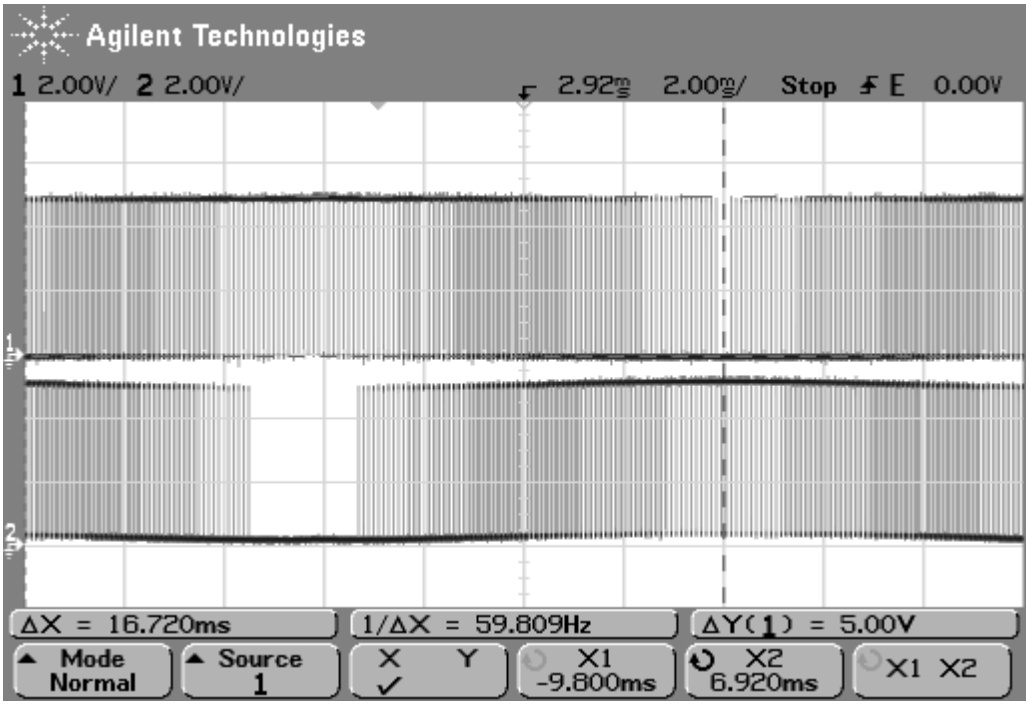


Fig.5. 17. The actual MUT microprocessor output pulses

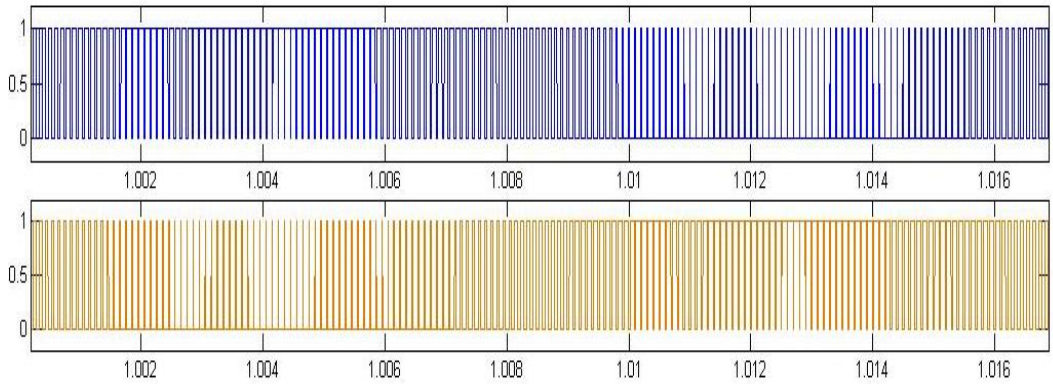


Fig.5. 18. The simulated MTU microprocessor unit pulses

Initial testing showed that the microprocessor functioned as required to produce the pulses which controlled the operation of the inverter power supply.

The output voltage and frequency are tested as shown in Figures 5.19 and 5.20 which describe the output of the inverter power supply with no load in both the actual implementation and in the simulation environment, respectively.

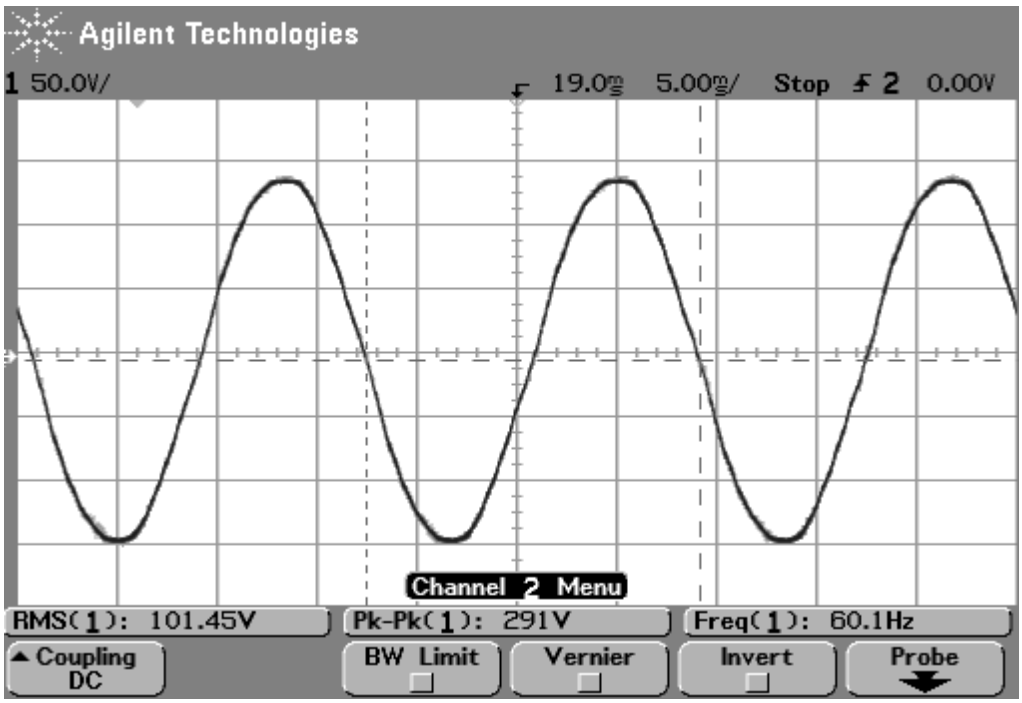


Fig.5. 19. Actual sin wave output

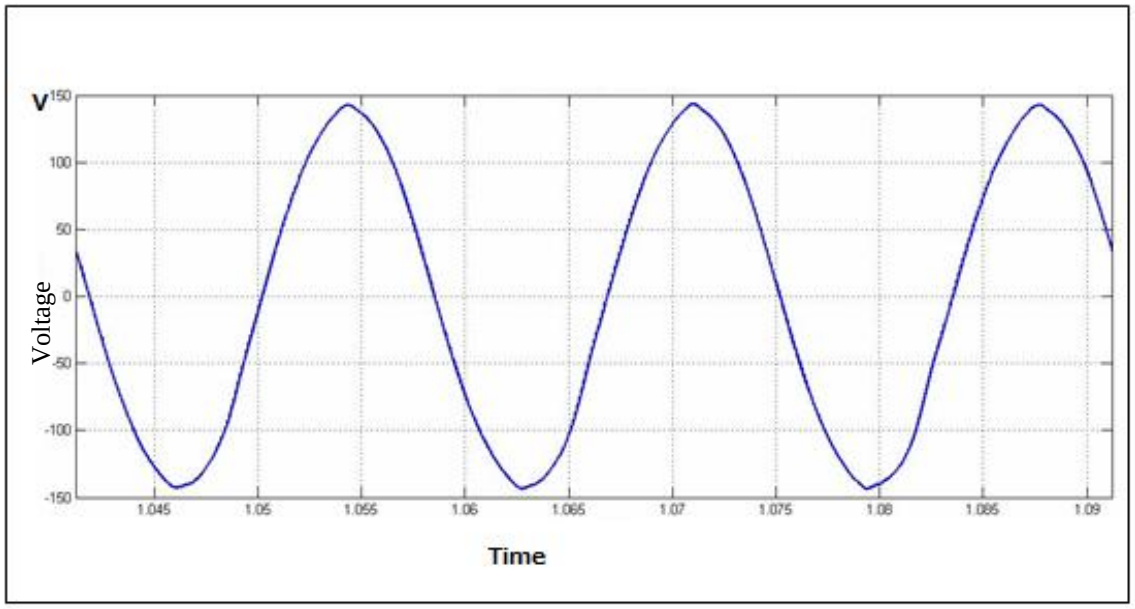


Fig.5. 20. Simulated sin wave output

Comparing the two waveforms, it can be seen that the result obtained by the developed model is in a good agreement with the actual result in both frequency and voltage amplitude as shown in Figure 5.19. The output of the inverter power supply is 101.45 RMSV value and frequency equals to 60.1 Hz which meets the required specifications.

One of the most important parameters in the inverter power supply design is the selection of the dead time value. It is a short time delay between turning on and off of the MOSFET to prevent the short circuiting that may occur. The dead time value is optimized in the virtual environment (MILS environment) and this value is used in the actual implementation. Theoretically, the value of the dead time can be selected between 1 to 5 µsec. The shorter is the dead time is, the better is the inverter power supply response. Any miss selection of the dead time value can lead to reduction in the desired voltage amplitude due to the accumulated delay during an entire cycle. As well as, it can cause distortion in the voltage output. Trial and error technique is used to optimize the dead time value which is 2 µsec and this value can be tested in both the actual and simulated environment as shown in Figures 5.21 and 5.22, respectively.

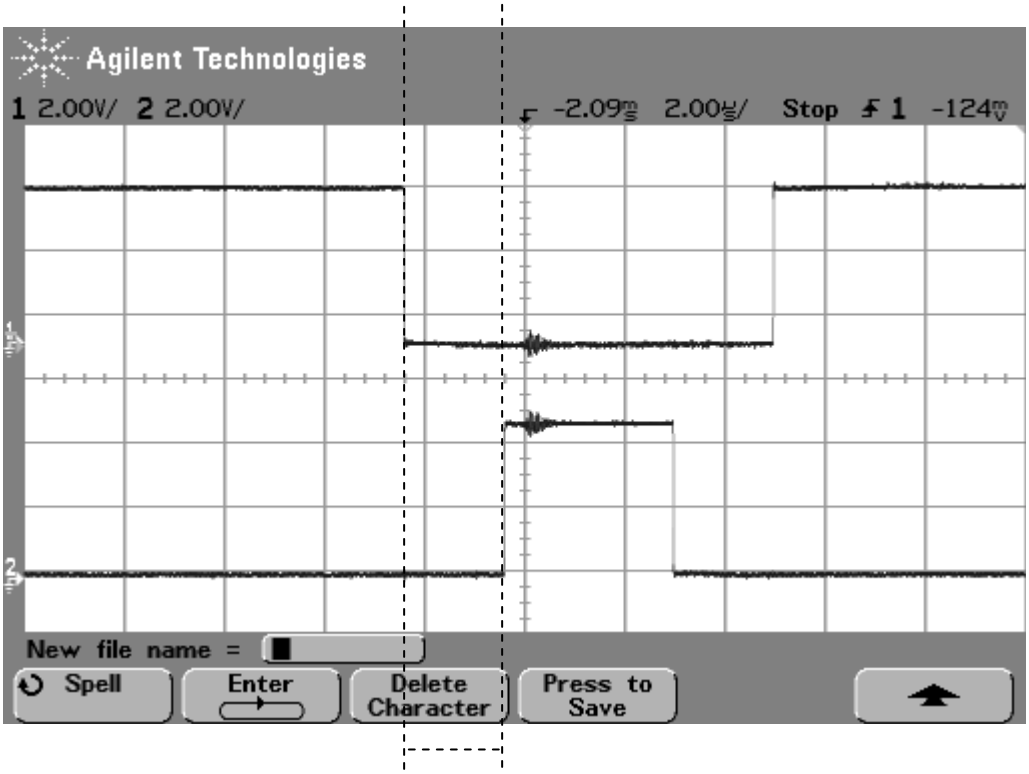


Fig.5. 21. The actual dead time value Td = 0.002ms

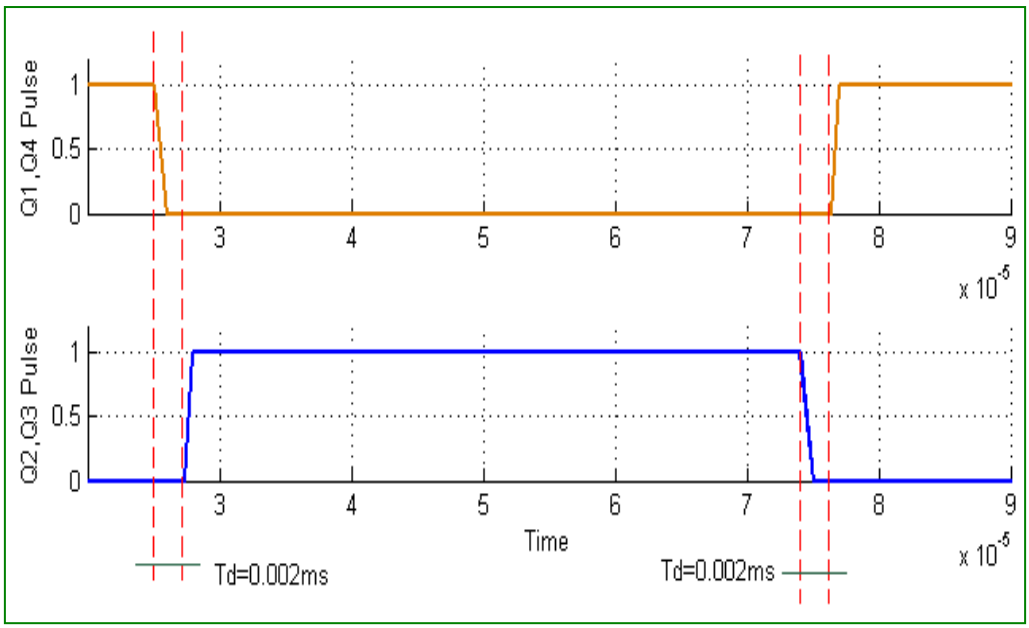


Fig.5. 22. The simulated value of the dead time

5.4. RESPONSE WITH LINEAR LOAD

The inverter power supply output distortion or the output voltage drops is considered a very important issue in the design of the inverter power supply. There are many

regulations regarding the allowable voltage drop in the inverter power supply. For example, based on the standard (IEC686) [1, 2], the allowed voltage drop is no more than 5% voltage against single parameter. Various techniques are used to compensate the effect of the voltage drop in the inverter power supply design. One of the main techniques is to apply the control system. The main duty of the inverter control system is to regulate the output voltage against the entire possible disturbance and the load variations. As mentioned before, the two layer control algorithm which consists of the feedback PI controller plus the feedforward controller is proposed to control the inverter power supply. The validity and usefulness of this controlling algorithm is tested in the existence of linear load. Figure 5.23 and Figure 5.24 show the output of the inverter power supply with no load in both the actual prototype and in the MILS environment, respectively. While Figure 5.25 and Figure 5.26 show, the inverter power supply output after connecting the load resistance Rl = 60Ω. From Figure 5.25, we can see the voltage reduction at the moment when the resistance is connected but due to the control algorithm the voltage can return to its value within a very short time. We can conclude that the proposed control algorithm is able to modify the distortion occurred due to the load in a very short time. So, using the control algorithm can improve the inverter power supply's peformances.

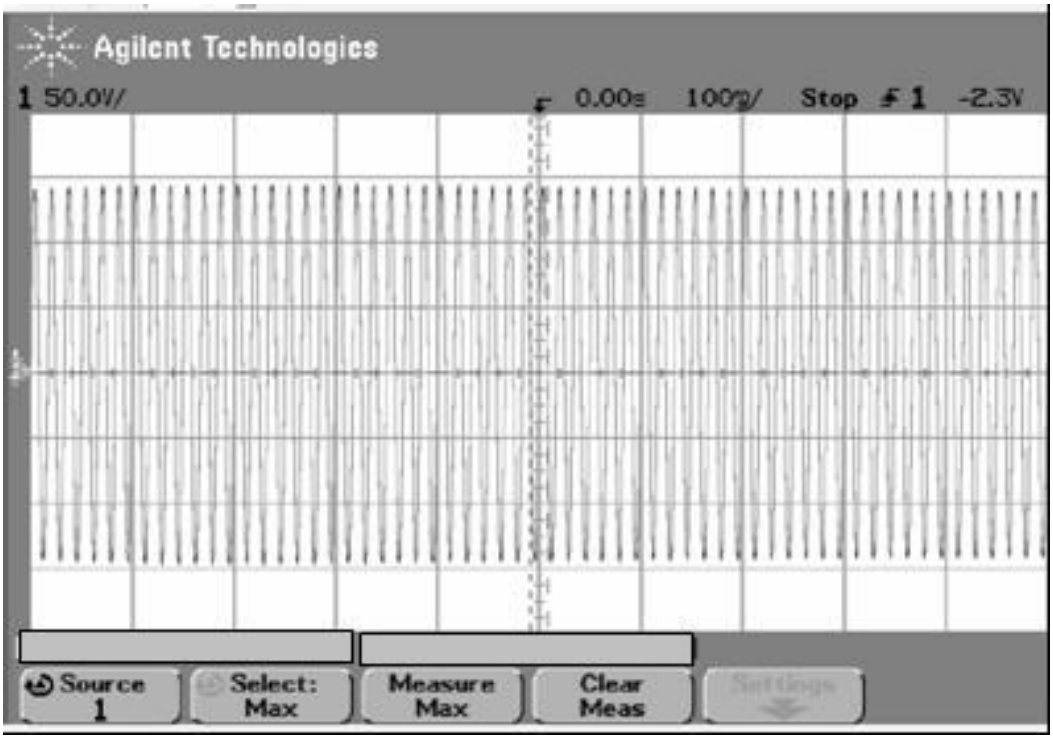


Fig.5. 23. Inverter power supply output with no load

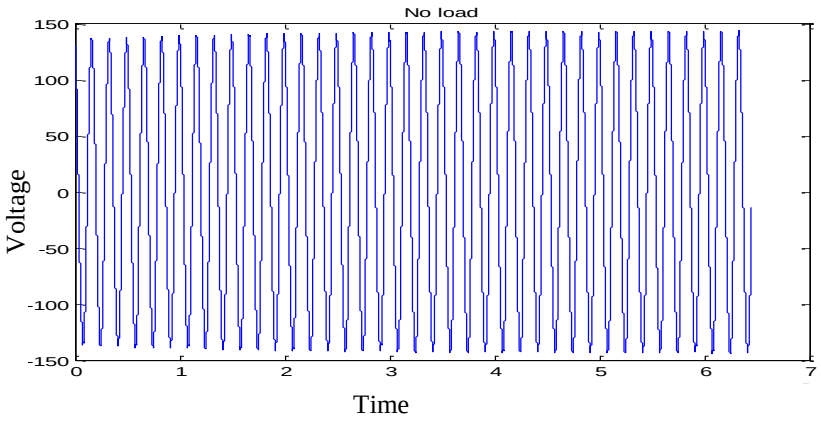


Fig.5. 24. Inverter power supply output with no load in the MILS

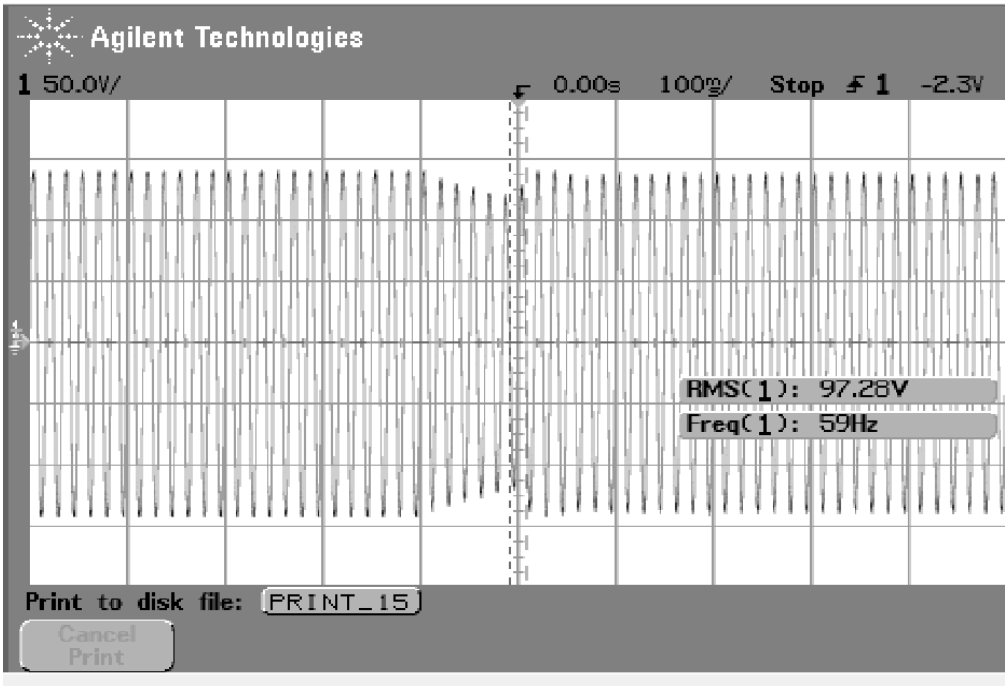


Fig.5. 25. The inverter power supply output with the linear load

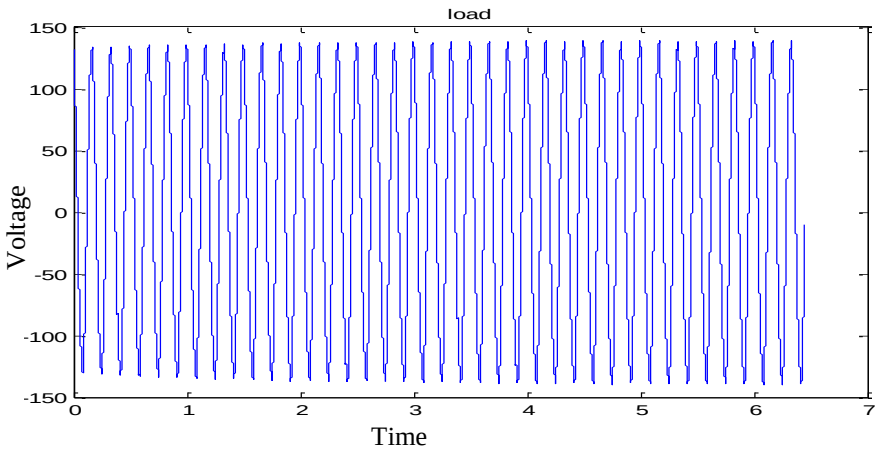


Fig.5. 26. The inverter power supply output with the linear load in MILS

Comparing the result with no load and the result with load, it seems that the shapes of the wave forms are almost identical except in the moment that we connect the load resistance. The voltage decreased by about 2% of no load voltage for a very short time around 100 msec. And then it returns to its normal value. So, we can conclude that the proposed controlling algorithm is working well and can maintain the disturbance occurred due to the load connection.

The inverter output with no load is illustrated in Figure 5.27 while the inverter output with the linear load is illustrated in Figure 5.28. By comparing both figures, it is clear that there is a very small voltage drop due to the wiring and the internal component resistance in Figure 5. 26. The voltage drop is less than 1% which falls within acceptable range.

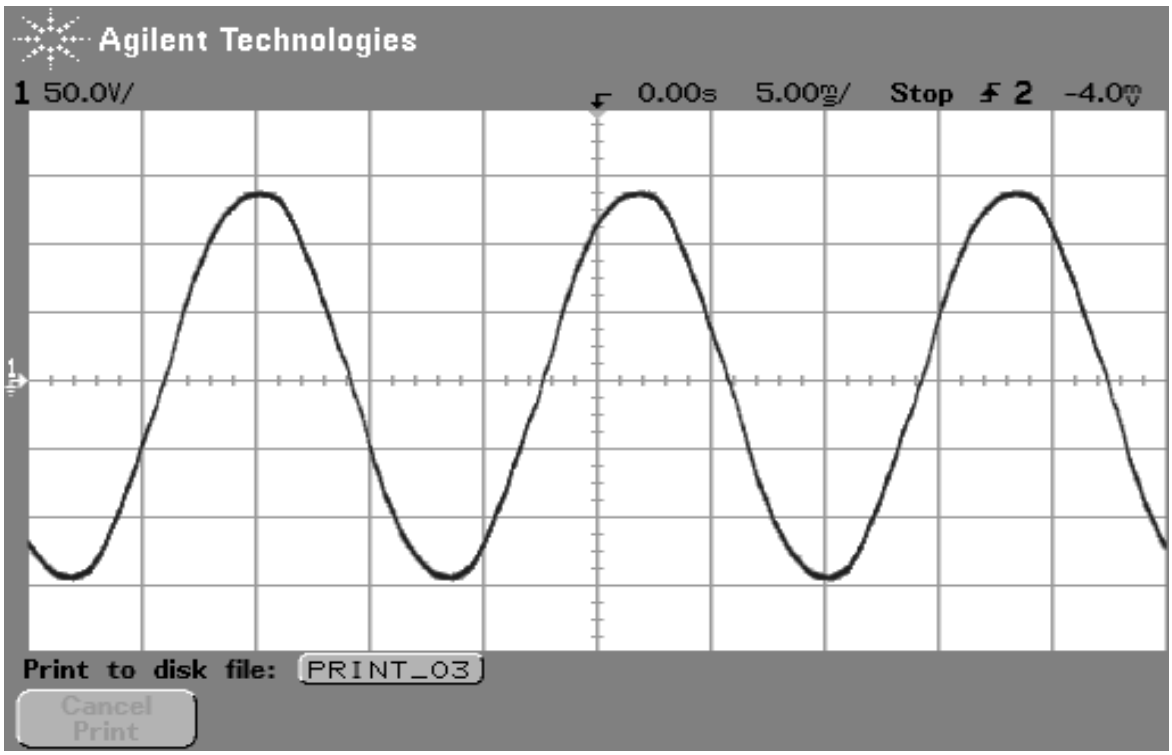


Fig.5. 27. Output of the inverter power supply with no load

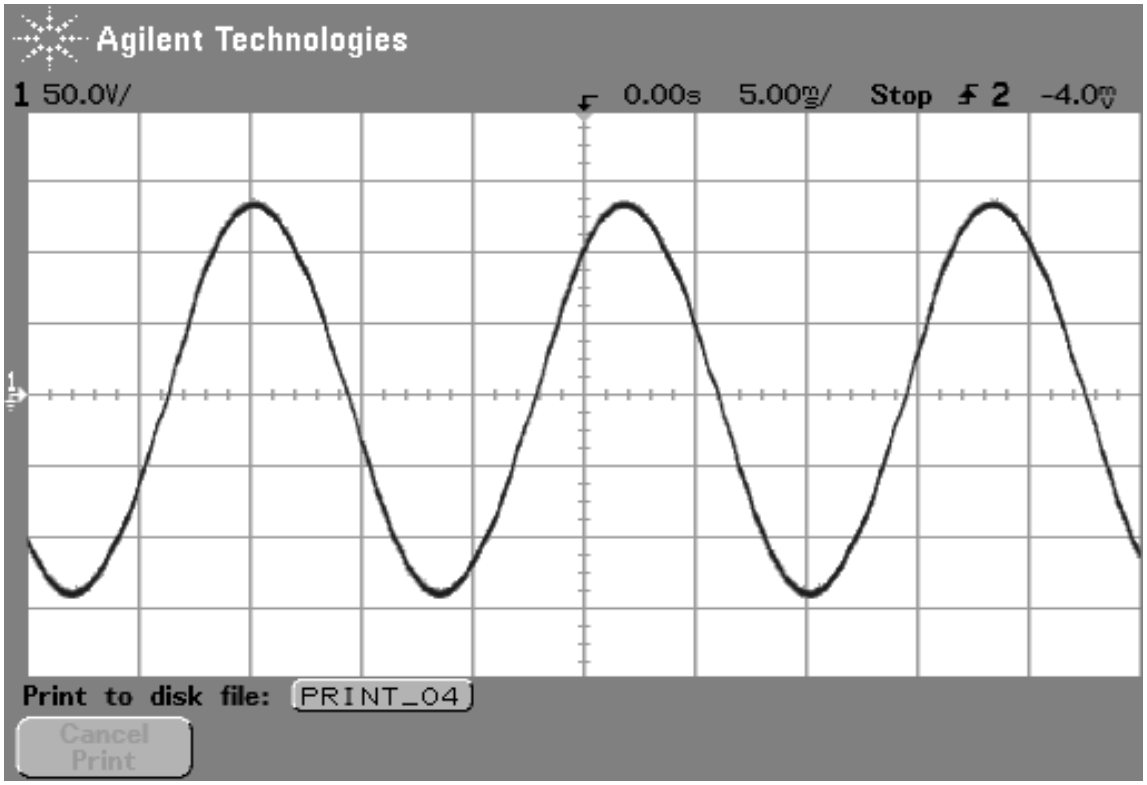
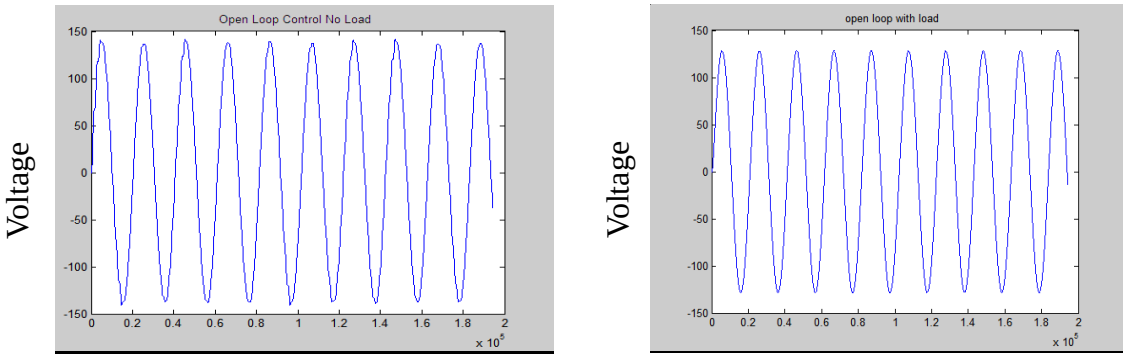


Fig.5. 28. Output with the linear load

One of the Model Based Design method's merits is the ability to test and check the control system freely. In the conventional design method, the control algorithm is tested in the actual prototype so stopping the control algorithm can lead to severe damage in the device. Here, in this section, the inverter power supply was tested with the open loop case and with the controlled case.



(a) No load (b) linear (resistive) load

Fig.5. 29. Inverter power supply output with open loop

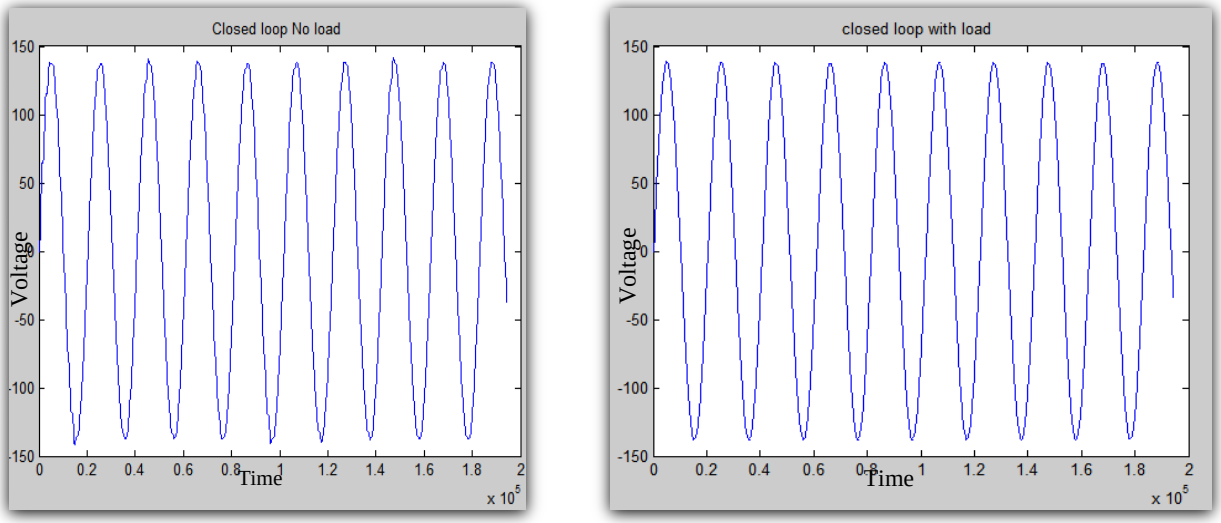
Figure 5.29 (a) describes the inverter power supply output with open loop control and with no load in the MILS environment. The output is 100 RMSV. Then, the resistive load was connected and the output is tested as in figure 5.29 (b). The output equals 90.9 RMSV. So, it can be seen that the load voltage has reduced by 10% which results in reduced load power as confirmed by the following equation:

$$P_{load} = V_{load}^2 / R_L \qquad (5.3)$$

To show the effectiveness of the proposed control algorithm, the output of the inverter power supply is tested as shown in Figure 5.30 (a) and 5.30 (b). These results were compared with the results shown in Figure 5.29 (a) and 5.29 (b).

The output of the inverter power supply with the proposed control algorithm under no load condition is shown in Figure 5.30 (a). The linear load voltage is presented in Figure 5.30 (b). The voltage reduction is less than 2% from the case with no load. Comparing these results, it can be seen that the shape of the waveforms are almost identical. The main differences are in the magnitudes of the waveforms. In case of the proposed controller, the

load voltage distortion is decreased and this can improve the performances of the inverter power supply.



(a) No load (b) linear (resistive) load

Fig.5. 30. Inverter power supply output with control algorithm

5.5. SUGGESTION OF ARTIFICAL NEURAL NETWOK APPLICATION

In the recent years, due to the pervasive development in computer software, the Artificial Neural Networks (ANN) is introduced in many applications [3, 4]. The Artificial Neural Networks (ANN) will be used to optimize the software parameters and to predict the software error before the actual implementation. ANNs can be applied to the global parameter optimization of both of the plant and the control part of embedded system. ANN can be defined as a massively parallel distributed processor which is constructed from interconnected processing elements called neurons. These neurons are simple emulation of the biological neurons [5]. In the future work, we will use the ANN in the system controller. It can be trained either online or off line. ANN is adaptive enough for the environment changes. Furthermore, it has excellent merits for nonlinear behaviors. Building on that, ANN will be used instead of the S-Function for the parameter optimization. In the application of ANN to the actual problems, the learning speed and accuracy are very important issues. The ANN with new learning algorithm based on the simulated evolution is proposed in the inverter power supply application. The performances of the proposed algorithm, called SimE-ANN, are checked comparing to the conventional back propagation algorithm using the landslide materials as example data [3, 4]. So SimE-ANN will be applied to optimize the embedded software of inverter power

supply as shown in Figure 5.31. By using this SimE-ANN, the learning time was decreased by about 99% comparing to the traditional BP algorithm.

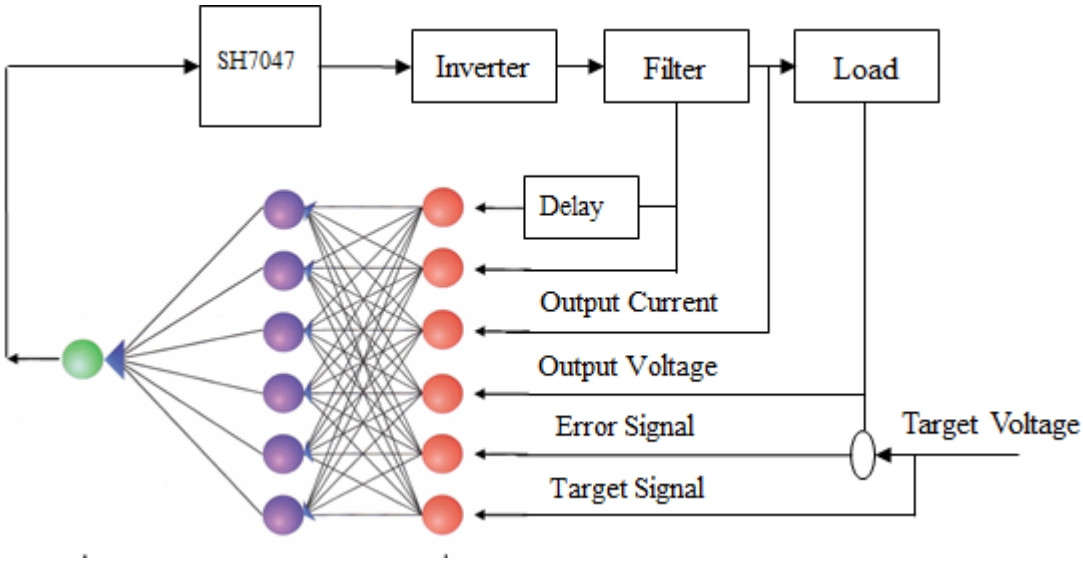


Fig.5. 31. Proposed ANN control for the inverter power supply

In this Figure, the input data in the input layer are the actual output voltage and current, the target voltage, the error signal and the delay time. The ANN model will be trained off line to optimize the embedded software parameters. Then the ANN model will be connected to the inverter power supply circuits as a controller, then the performance of the ANN model will be tested compared to the actual prototype implementations. The proposed ANN control scheme must be evaluated as the future work.

REFERENCES

[1] M. Niroomand and H. R. Karshenas, "Review and Comparison of Control Methods for Uninterruptible Power Supplies", In Proceeding of Power Electronic & Drive Systems & Technologies Conference (PEDSTC), IEEE, pp.18-23, May, 2010.

[2] D. Heng, O. Ramesh and S. Dipti, "Modeling and Control of Single Phase UPS Inverters: A Survey", In the Proceedings of Power Electronics and Drives System, Vol.2, pp.848-853, Oct. 28th - Nov. 1st, 2005.

[3] M. A. El Dahb, Y. Zhou, U. F. Siddiqi and Y. Shiraishi, "Artificial Neural Network based on Simulated Evolution and its application to Estimation of landslide", In the Proceeding of Mathematical Modeling and Problem Solving MPS Meeting, Okayama, May 17, 2011.

[4] M. A. El Dahb, Y. Zhou, U. F. Siddiqi and Y. Shiraishi, "Artificial Neural Network based on Simulated Evolution and its application to Estimation of landslide", IPSJ Transactions on Mathematical Modeling and Problem Solving, September, 2011 (to appear).

[5] Mona A. El-Dahb, Yao Zhou and Yoich Shiraishi, "The Application Simulated Evolution and Neural Network to Estimate of Ground Sliding". In the Proceedings of ICICIS International Conference 2011, Ain Shams Univeristy, Cairo, Egypt, June 30-July 3, 2011.

CHAPTER 6
CONCLUSIONS AND RECOMMENDATIONS

6.1. CONCLUSIONS

In general, this research meets the challenges of the embedded system design by applying the Model Based Design (MBD) method in the early design stage. Furthermore, the main contribution of this research is to prove the potential of using the MBD method in the optimization and verification of the embedded system software development. This study proposed functional model of microprocessor in the early stage of the inverter power supply development, which allows the full simulation of the system in the virtual environment including the electrical circuit and software implementation.

Inverter power supply was taken as a case study of the embedded system design. We developed entire model of the system including the electrical circuit and the microprocessor using the MATLAB and Simulink package. All the software parameters were optimized as well as newly applied controlling algorithm was evaluated using the Model In the Loop Simulation (MILS) environment. The validity and the usefulness of the proposed model were tested comparing with the actual prototype of the inverter power supply. The developed model was shown to be well in agreement with the experimental results. This method can be used to optimize the software parameters before the actual design. Moreover, the models can be used to study and analyze the behaviors of the system which meet the challenges of designing the digital control system. The results show that the suggested models are promising and the models can be useful for optimizing the performances in developing the embedded software.

Based on experimental and theoretical results, the advantages of the proposed environment in the embedded system development can be drawn as follows:

1. Using the MBD method reduces the number of the development stages by combining the design, implementation and testing in one process.
2. The reduction of the required steps will result in better project managements and mitigation of the project risk.
3. The system which is designed with this approach may be able to reach the market faster and reduce the end cost less than the system developed using the conventional method.
4. This method shows a significant improvement in the software parameter optimization.

5. The proposed method can be used to study and analyze the behaviors of the system before the actual implementation which was considered very important in the embedded system development.
6. The use of MBD method makes it easy to modify the system in the future.
7. In the virtual environment, the user can control the system freely without side effects. For example stopping the entire control system, modifying the control algorithm while if the real control system stops, this may result in a very serious situation.
8. The control system designers can validate their control algorithm as well as optimization and validation of the embedded software by using the MBD method in early stage of design.

6.2. RECOMMENDATION FOR FUTURE STUDY

6.2.1. MILS Quality Improvement

In this study, we modeled the basic blocks in the microprocessor which controls the operation of the inverter power supply. More complex models should be considered to improve the accuracy of the environment. Analog to digital (A/D) and digital to analog (D/A) converters of the microprocessor will be added to the MILS. As well as, the performance of the inverter power supply will be tested with nonlinear loads.

6.2.2. Optimization of Software Parameters Based on Neural Network

The Artificial Neural Networks (ANN) will be used to optimize the software parameters and to predict the software errors before the actual implementation. ANNs have been employed in many applications in recent years. ANN is a massively parallel distributed processor which is constructed from interconnected processing elements called neurons. These neurons are simple emulation of the biological neurons. In the future work, we will use simulated Evolution algorithm (SimE) to improve the performance of the conventional back propagation algorithm then we will use this proposed SimE-ANN in the system controller of the inverter power supply. It can be trained either online or off line. ANN is adaptive enough for the environment changes. Furthermore, it has excellent merits for nonlinear behavior. Building on that, ANN will be used instead of the S-Function for the parameter optimization.

APPENDIX I

DC-DC Converter

* SH7047 C-MEX File code for the MMT description

* Function: Produce PWM pulse to drive the full bridge converter

*Also implement the control algorithm

==========

```
 #define S_FUNCTION_NAME  DCDC_Control
#define S_FUNCTION_LEVEL 2
 #include "simstruc.h"
#include "math.h"
#define U(element)          (*uPtrs[element])
 /*Parameter definition*/
#define Kp0     mxGetPr(ssGetSFcnParam(S,0))
#define Ki0     mxGetPr(ssGetSFcnParam(S,1))
#define Kd0     mxGetPr(ssGetSFcnParam(S,2))
#define ANFA0   mxGetPr(ssGetSFcnParam(S,3))
#define dc_ref   mxGetPr(ssGetSFcnParam(S,4))
/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *   Setup sizes of the various vectors.
 static void mdlInitializeSizes(SimStruct *S)
{  int ninputs=2;
   int noutputs=2;
   int nstates=4;
   ssSetNumSFcnParams(    S, 5);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      return;
   }
   ssSetNumContStates(    S, nstates);
   ssSetNumDiscStates(    S, 0);
   if (!ssSetNumInputPorts(S, 2)){
   return;
   }
   ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
   ssSetInputPortWidth(S, 1, DYNAMICALLY_SIZED);
   ssSetInputPortDirectFeedThrough(S, 0, 1);
   ssSetInputPortDirectFeedThrough(S, 1, 1);
   if (!ssSetNumOutputPorts(S, 2)){
    return;    }
```

```
    /*ssSetOutputPortWidth(S, 0, noutputs);     // Width is set to private port */
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
    ssSetNumSampleTimes(   S, 1);            // sampling time
    ssSetNumRWork(         S, 0);   /* number of real work vector elements    */
    ssSetNumIWork(         S, 0);   /* number of integer work vector elements */
    ssSetNumPWork(         S, 0);   /* number of pointer work vector elements */
    ssSetNumModes(         S, 0);
    ssSetNumNonsampledZCs( S, 0); }
Function: mdlInitializeSampleTimes ========================================
static void mdlInitializeSampleTimes(SimStruct *S)
{ ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}
 #define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S){
    int i,nstates=4;
    real_T *x0 = ssGetContStates(S);
    for(i=0;i<nstates;i++){
        x0[i] = 0;
    }}
/* Function: mdlOutputs =======================================================
 static void mdlOutputs(SimStruct *S, int_T tid)
{ InputRealPtrsType    time = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType Vdc_feedback =ssGetInputPortRealSignalPtrs(S,1);
    real_T           *pulse1     = ssGetOutputPortRealSignal(S,0);
    real_T           *pulse2     = ssGetOutputPortRealSignal(S,1);
    real_T           *x=ssGetContStates(S);
   double
sawtooth,PI,kp,ki,kd,anfa,m,Vdc_ref,sawtooth_period,k_trans,dc_error,dc_deadtime,sawtooth_freq,Vdc_in;
    int n_sawtooth_halfperiod;
    Vdc_ref=dc_ref[0];
    pi=3.1415926535898;
    AC_freq=60.0;
    sawtooth_freq = 10000.0;
    sine=sin(2*AC_freq*pi*(*time[0]));
    sawtooth_period=1/sawtooth_freq;
    dc_error=Vdc_ref-*Vdc_feedback[0];
```

```
    n_sawtooth_halfperiod=floor ((*time[0])/(sawtooth_period/2));
    dc_deadtime=0.000002;
    Vdc_in=24.0;
    k_trans=2*7.08;
    kp=Kp0[0];
    ki=Ki0[0];
    kd=Kd0[0];
    anfa=ANFA0[0];
    PI=ki/anfa*x[0]+(ki-kd/anfa/anfa-kp/anfa)*x[1]+(kd/anfa+kp)*dc_error;
    m=1-(Vdc_ref+PI)/(k_trans*Vdc_in);
    if ((n_sawtooth_halfperiod%2)==0)
    {
    sawtooth=((2/(sawtooth_period/2))*((*time[0])-(sawtooth_period/2)*n_sawtooth_halfperiod))-1;
    }
    else
    { sawtooth=1-((2/(sawtooth_period/2))*((*time[0])-((sawtooth_period/2))*n_sawtooth_halfperiod));}
       if(sawtooth>m){
          pulse1[0]=0;
          pulse2[0]=1;}
      else
      {if((0-m)<sawtooth)
      {   pulse1[0]=0;
          pulse2[0]=0;}
      else
      {if(sawtooth<(0-m)){
          pulse1[0]=1;
          pulse2[0]=0;}}} }
#define MDL_DERIVATIVES
static void mdlDerivatives(SimStruct *S){
    double Vdc_ref,dc_error;
    real_T *dx = ssGetdX(S);
    real_T *x = ssGetContStates(S);
    InputRealPtrsType Vdc_feedback =ssGetInputPortRealSignalPtrs(S,1);
    //InputRealPtrsType    clock = ssGetInputPortRealSignalPtrs(S,0);
  //  long double sine=sin(2*50*3.14*(*clock[0]));
    double anfa=ANFA0[0];
    Vdc_ref=dc_ref[0];
    dc_error=Vdc_ref-*Vdc_feedback[0];
    dx[0]=x[1];
    dx[1]= -1/anfa*x[1]+dc_error;
```

```
}
 /* Function: mdlTerminate =================================================
static void mdlTerminate(SimStruct *S)
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

DC-AC Inverter

*** SH 7047 C-MEX File code for the MTU description**

*** Function: Produce PWM pulse to drive the full bridge Inverter**

***Also implement the control algorithm**

```
*/================================================
#define S_FUNCTION_NAME  DCAC_Control
#define S_FUNCTION_LEVEL 2
 #include "simstruc.h"
#include "math.h"
#define U(element)          (*uPtrs[element])
/*Parameter Defination */
#define Kp     mxGetPr(ssGetSFcnParam(S,0))
#define Ki     mxGetPr(ssGetSFcnParam(S,1))
#define Kd     mxGetPr(ssGetSFcnParam(S,2))
#define ANFA   mxGetPr(ssGetSFcnParam(S,3))
#define ac_ref   mxGetPr(ssGetSFcnParam(S,4))
/* Function: mdlInitializeSizes ===============================================
static void mdlInitializeSizes(SimStruct *S)
{  int ninputs=2; // Num of input
   int noutputs=2;// Num of output
   int nstates=4; //
   ssSetNumSFcnParams(    S, 5);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      return; }
   ssSetNumContStates(    S, nstates);  // Number of instance
   ssSetNumDiscStates(    S, 0);        //
   if (!ssSetNumInputPorts(S, 2)){
      return; }//
   ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
   ssSetInputPortWidth(S, 1, DYNAMICALLY_SIZED);  // provide port
   ssSetInputPortDirectFeedThrough(S, 0, 1); //
   ssSetInputPortDirectFeedThrough(S, 1, 1);
   if (!ssSetNumOutputPorts(S, 2)){
      return;//
   }
   /*ssSetOutputPortWidth(S, 0, noutputs);     // */
   ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
   ssSetOutputPortWidth(S, 1, DYNAMICALLY_SIZED);
   ssSetNumSampleTimes(   S, 1);             //
```

```
    ssSetNumRWork(        S, 0);   /* number of real work vector elements    */
    ssSetNumIWork(        S, 0);   /* number of integer work vector elements */
    ssSetNumPWork(        S, 0);   /* number of pointer work vector elements */
    ssSetNumModes(        S, 0);
    ssSetNumNonsampledZCs( S, 0);
    }
/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
static void mdlInitializeSampleTimes(SimStruct *S)
{  ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDisallowInheritance(S);
}
 /*initialize the initial conditions*/
 #define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S){
    int i,nstates=4;
    real_T *x0 = ssGetContStates(S);
    for(i=0;i<nstates;i++){
        x0[i] = 0;
    }}
/* Function: mdlOutputs =======================================================
 * Abstract:
static void mdlOutputs(SimStruct *S, int_T tid)
{  InputRealPtrsType    time = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType Vac_feedback =ssGetInputPortRealSignalPtrs(S,1);
    real_T            *pulse1     = ssGetOutputPortRealSignal(S,0);
    real_T            *pulse2     = ssGetOutputPortRealSignal(S,1);
    real_T            *x=ssGetContStates(S);
    double
sine,sawtooth,PI,kp,ki,kd,anfa,m,Vac_ref,sawtooth_period,pi,ac_error,ac_deadtime,AC_freq,sawtooth_freq,
Vdc_out;
    int n_sawtooth_halfperiod;
    Vac_ref=ac_ref[0];
    pi=3.1415926535898;
    AC_freq=60.0;
    sawtooth_freq = 10000.0;
    sine=sin(2*AC_freq*pi*(*time[0]));
    sawtooth_period=1/sawtooth_freq;
```

```
    ac_error=Vac_ref*sine-*Vac_feedback[0];
    n_sawtooth_halfperiod=floor((*time[0])/(sawtooth_period/2));
    ac_deadtime=0.000002;
    Vdc_out=170.0;
    kp=Kp[0];
    ki=Ki[0];
    kd=Kd[0];
    anfa=ANFA[0];
    PI=ki/anfa*x[0]+(ki-kd/anfa/anfa-kp/anfa)*x[1]+(kd/anfa+kp)*ac_error;
    m=(Vac_ref+PI)/Vdc_out;
    if ((n_sawtooth_halfperiod%2)==0)
    {
    sawtooth=((2/(sawtooth_period/2))*((*time[0])-(sawtooth_period/2)*n_sawtooth_halfperiod))-1;
    }
    else
    { sawtooth=1-((2/(sawtooth_period/2))*((*time[0])-((sawtooth_period/2))*n_sawtooth_halfperiod));}
    if(sawtooth<m*sine){
          pulse2[0]=0;
    if ((sawtooth+((2*ac_deadtime)/(sawtooth_period/2)))<m*sine)
     {
            pulse1[0]=1;
      }
          else pulse1[0]=0;
    }
    else
       { pulse1[0]=0;
     if ((sawtooth-((2*ac_deadtime)/(sawtooth_period/2)))>m*sine)
     {
           pulse2[0]=1;
    }
      else pulse2[0]=0;
    }}
#define MDL_DERIVATIVES
static void mdlDerivatives(SimStruct *S){
    double Vac_ref,ac_error;
    real_T *dx = ssGetdX(S);
    real_T *x = ssGetContStates(S);
    InputRealPtrsType Vac_feedback =ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType    time = ssGetInputPortRealSignalPtrs(S,0);
    double sine=sin(2*60*3.14159*(*time[0]));
```

```
    double anfa=ANFA[0];
    Vac_ref=ac_ref[0];
    ac_error=Vac_ref*sine-*Vac_feedback[0];
    dx[0]=x[1];
    dx[1]= -1/anfa*x[1]+ac_error;
}
 /* Function: mdlTerminate =================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
static void mdlTerminate(SimStruct *S)
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

APPENDIX II

Description of the actual embedded program for the inverter power supply

File name	Type	Description
Init.C	C code file	Starting CPU operation
Initsct.C	C code file	Initializing the variables
ADRS.C	C code file	This file used for the symbolic debugging.
rtl.C	C code file	Low level interface
Invc.C	C code file	Combined AD control and PWM
bkup.C	C code file	the module necessary for using the flash memory
boot.C	C code file	This file used for re-writing programs when they are executing
dbg.C	C code file	a debugger
vect.C	C code file	Interruption
MAIN.C	C code file	Main program
Sys.h	C header file	Including Definitions of IO , some values for constants or variables

Inverter Power Supply Main Embedded File

*** Function: Produce PWM pulse to drive inverter power supply**

***Also implement the control algorithm**

```
INT pwm_sw;                         /* Control switch */
FIXED Klpf;
FIXED Klpf2;                /* for the current lag */
FIXED maxaveV;    /* Maximum voltage */
FIXED maxD;                 /* Maximum modulation rate */
FIXED Ksft;         /* Modulation rate coefficients to convert from a value outside the specified */
FIXED Csft;                 /* constant */
int sft_goal;         /*  Goals and soft-start voltage */
int now_V;                  /* Current voltage  */
FIXED sft_m;        /* Voltage modulation rate for current issue */
int up_rate;          /* Voltage rise */
FIXED D_min;
FIXED fp_ten;
/* -- PIControl --*/
FIXED Kp;                                   /* Proportional gain           */
FIXED Ki;                                   /* Integral gain               */
FIXED Vout_max;   /*   Output voltage to the maximum AD conversion to a numeric value*/
FIXED Vin_max;                      /* Input voltage to the maximum AD conversion to a numeric*/
FIXED TR_n2n1;                      /* transformer ratio   */
FIXED lim_f_on;                     /* When feedforward limiter ON          */
FIXED lim_f_off;                    /* when feedforward limiter OFF                 */
FIXED m_min,m_max;                  /*          Minimum and maximum amount of operation*/
FIXED lpf_V;                        /* Filtered voltage            */
FIXED fp_V_ref;                     /* Value of fixed-point directive */
FIXED fp_ad_V;              /* Small number of points obtained voltage clamp AD converter */
FIXED dv;                   /*Command value and control deviations   */
FIXED P_ctrl_val;           /*          P value control            */
FIXED I_ctrl_val;           /*           I value control           */
FIXED PI_ctrl_val;          /* PI control values           */
FIXED Vout;         /* DC / DC output voltage (0 ~ 332V)       */
FIXED Vin;          /*  Input voltage (0 ~ 59.3V)              */
Int  forward;               /*          Feed-forward                        */
FIXED pi_m;                 /*          Modulation rate
// DCAC
FIXED Kp1;                  /* Proportional gain   */
FIXED Ki1;                  /* Integral gain       */
```

```
FIXED Kd1;
/*For debugging */
int Vout_int,Vin_int,dv_int,P_int,I_int  ;
int pi_m_int;        /*Check for pulse*/
FIXED fp_test1;
FIXED fp_test2;
FIXED fp_test3;
FIXED fp_test4;                 /* Analog Data */
FIXED prim_Vin;                 /* Input Voltage */
FIXED dcdc_Vout;                /* DC / DC output voltage */
int dcdc_Vref;                  /* DC / DC reference value）  */
FIXED dcac_Vout;      /* DC / AC output voltage (analog value converted to FIXED）  */
FIXED dcac_Iout = 0;
int now_V1 = 0;
FIXED ACMAX, ACMIN;
/*Sequence*/
enum {RUSH_PREVENTION=0,
        ERROR_RESET,            /* Error reset */
        SOFT_START,                     /* Soft Start */
        MAIN_LOOP                       /* The main loop (PI control)*/
sequence; }
/* test variable*/
int che[10];
FIXED def [10];
extern int INTE[3];
FIXED tempD;
int TEMP_I;
int prim_Vin_chk;
int chk_sft_m;
/* ********************************************************** */
/* Inverter operation MMT*/
/* ********************************************************** */
#define PI          3.1415927
#define F1          60                  /* Output Frequency             */
#define FS              20000           /* carrier frequency for the MMT    */
#define TD              2000                    /* deadtime ns]             */
#define FS1             10000           /* carrier frequency for the MTU        */
#define VMAX_AC 100                     /* DCAC output maximum*/
extern  FIXED   fp_zero;
extern  FIXED   fp_one;
```

```
extern      FIXED     fp_twopi;
FIXED       fp_m,fp_u;
FIXED       fp_dwt,fp_wt;
FIXED       fp_m1, fp_u1;                          // MTU
FIXED       fp_dwt1,fp_wt1;                        // MTU
FIXED       fp_k,fp_rt2,fp_05;
INT tdcin, tdcout, tacout;                  // for test
/*          MMT Start inverter operation
void        inv0_start(void)
{ inv0_set_deadtime(TD);                           /* Dead time */
            inv0_init(0,1,FS);                     /* Modulation Mode */
            fp_dwt = fixedp(2.0*PI/FS*F1);         /*      Δωt          */
            fp_wt = fp_zero;                       /* Default           */
            fp_m = fp_zero;                       /* Modulation index starts at zero   */
            fp_u = fp_zero;                                    /* Default          */
            fp_k = fixedp(0.05);                          /* Filter factor       */
            fp_rt2 =fixedp(1.41421356);                   /* √2                  */
            fp_05 = fixedp(0.5);                          /* Voltage rise coefficient*/
            //inv0_set_uvw(fp_zero,fp_zero,fp_zero);  /* Initial value setting */
            inv0_set_uvw(fp_zero,fixedp(1.1),-fixedp(1.1));
            wait(100);
            inv0_start_int();}
/*          MTU inverter start operation     */
            void        inv1_start(void)
{inv1_set_deadtime(TD);                                /* Dead time setting */
            inv1_init(0,1,FS1);                    /* The first argument: modulation mode *
            fp_dwt1 = fixedp(2.0*PI/FS1*F1);       /* Δωt               */
            fp_wt1 = fp_zero;                      /* Initial value             */
            fp_m1 = fp_zero;                       /* Modulation rate from 0    */
            fp_u1 = fp_zero;                       /* Initial value             */
            //inv1_set_uvw(fp_zero,fp_zero,fp_zero);  /* Initialization */
            inv1_set_uvw(fp_zero,fixedp(1.1),-fixedp(1.1));
            wait(100);          /* To enable the initial value over a period to wait for carrier */
            inv1_start_pwm();
            inv1_start_int();                                  /* start interrupt     */
}
/**********************************
/*  PI control
/*                  V_ref      : Target voltage）
/*                  ad_Vin     : After filtering the input voltage
```

```
/*                  ad_Vout   : Output voltage after filtering
/***********************************/
void PIctrl(INT V_ref, FIXED ad_Vin, FIXED ad_Vout){
        fp_V_ref = TOFIX(V_ref);                /* FIXED command value into */
        Vout = ad_val(ad_Vout,fp_zero,Vout_max);        /* AD converted value in volts (however,
FIXED-type) AD 322 were the 1023 [V]  */
        Vin = ad_val(ad_Vin,fp_zero,Vin_max);           /* AD converted value in volts */
        dv = fp_V_ref - Vout;                           /* Deviation */
        /*For debugging*/
        Vout_int=TOINT(fpmuls(Vout,TOFIX(1000)));
        //Vout_double=Vout_int / 1000.0;
        Vin_int=TOINT(fpmuls(Vin,TOFIX(1000)));
        //Vin_double=Vin_int / 1000.0;
        dv_int=TOINT(dv);
        if(pwm_sw){
                P_ctrl_val = fpmuls(Kp,dv);
        /* P(Proportional) control                 */
        I_ctrl_val = I_ctrl_val + fpmuls(Ki,dv);    /* I(integral) control                 */
        if(forward){                                /* Presence of feed-forward            */
        LIMITER(&I_ctrl_val,-lim_f_on,lim_f_on);        /* ON-time limiting integral term*/
                }else{
        LIMITER(&I_ctrl_val,-lim_f_off,lim_f_off); /* OFF integral term at the limiter     */
                }
                PI_ctrl_val = P_ctrl_val + I_ctrl_val;/* Control the value of PI = P + I   */
                if(forward){            /* Presence of feed-forward                */
LIMITER(&PI_ctrl_val,-lim_f_on,lim_f_on);/* PI limiter controls the ON time                 */
                }else{
                LIMITER(&PI_ctrl_val,-lim_f_off,lim_f_off); /* OFF-time PI control value limit */
                }
                /* For debugging */
                P_int = TOINT(fpmuls(P_ctrl_val,TOFIX(1000)));
                I_int = TOINT(fpmuls(I_ctrl_val,TOFIX(1000)));
                /* feed-forward element is now required */
                //nextV = Vout + PI_ctrl_val;           /* Next target output voltage*/
                if(Vin > 0){
                pi_m=fpdivs(PI_ctrl_val+fpmuls(TOFIX(forward),fp_V_ref),fpmuls(Vin,TR_n2n1));}
                //pi_m = pi_m + fpdivs(fp_V_ref,fpmuls(Vin,TR_n2n1));
                pi_m = fp_one - pi_m;
                LIMITER(&pi_m,m_min,m_max);
                PE.DRL.BYTE.L = (pi_m >> 8) & 0xFF;
```

```
                inv0_set_uvw(0,pi_m,-pi_m);
        }else{I_ctrl_val = 0; /*Integral term initialization */
        }}
/*MTU PI Control*/
FIXED pi_m1, dv1;
FIXED fp_V_ref1, Vout1, Vin1;
FIXED P_ctrl_val1, I_ctrl_val1, PI_ctrl_val1;
FIXED D_ctrl_val1, PID_ctrl_val1;
FIXED dvp = 0;                                  // Last Deviation
FIXED ACSUM = 0;
void PIctrl1(INT V_ref, FIXED ad_Vin, FIXED ad_Vout){
        fp_V_ref1 = TOFIX(V_ref);
        ACSUM = -ACMAX;
        Vout1 = ad_val(ad_Vout,-ACMAX,ACMAX);       /* converted the analog value to Digital   */
        Vin1 = ad_val(ad_Vin,fp_zero,Vout_max);
        dv1 = fpmuls(fp_V_ref1, fpsin(fp_wt1)) - Vout1;                 /* Deviation */
        if(pwm_sw){
                P_ctrl_val1 = fpmuls(Kp1,dv1);          /* P (proportional) control*/
                // feed forward
                I_ctrl_val1 = I_ctrl_val1 + fpmuls(Ki1,dv1);        /* I (integral) control */
                if(forward){            /* Present value of feed-forward           */
        LIMITER(&I_ctrl_val1,-lim_f_on,lim_f_on); }/* limites of integral term    */
else
{LIMITER(&I_ctrl_val1,-lim_f_off,lim_f_off);/* OFF-time limiting integral term     */
                }
PI_ctrl_val1 = P_ctrl_val1 + I_ctrl_val1;/*the value of PI = P + I control value control*/
        if(forward){
LIMITER(&PI_ctrl_val1,-lim_f_on,lim_f_on); }    /* ON time limites for PI          */
else{
LIMITER(&PI_ctrl_val1,-lim_f_off,lim_f_off);    /* OF limites for PI   */           }
nextV = Vout + PI_ctrl_val;                     /* Next target output voltage */
                if(Vin > 0){                    /*Avoid negative division */
                        pi_m1 = fpdivs(PI_ctrl_val1, Vin1);
                }
pi_m1 = pi_m1 + fpdivs(fp_V_ref,fpmuls(Vin,TR_n2n1));
        pi_m1 = fp_one - pi_m1;
        LIMITER(&pi_m1,-fp_one,fp_one);/* Modulation rate limiter (-1 to 0 duty ⇒ 1% ~ 100%) */
                fp_m1 = pi_m1;
        inv1_set_uvw(0,pi_m1,-pi_m1);
        }else{I_ctrl_val1 = 0; /* Integral term initialization */
```

```
        }}
void parameter_init(void){
        //Kad = fpdivs(fp_3_1,fp200);          /* Analog */
        //Kvacc = fixedp(204.8);                        /* 1024/5 (5V→1024) */
        //Kq16 = fpdivs(fp_1024,fp_5);
        ACMAX = TOFIX(170);                           // Maximum DC Output
        ACMIN = fixedp(0);          // debug MTU
        dcdc_Vout = 0;
        prim_Vin = 0;
        sequence=RUSH_PREVENTION;       /* The first sequence set */
        dcac_Vout = 0;
       Vset = 100;
        now_D = TOFIX(1);
        pwm_sw=0;
        lpf_V = 0;
        if(Vset1 > VMAX_AC)
                Vset1 = VMAX_AC;
        else if(Vset1 < 0)
                Vset1 = 0
     /* --- Filter-- */
        Klpf = fixedp(0.1);
        Klpf2 = TOFIX(1)-Klpf;
        maxaveV = fixedp(3.8);                  /* The maximum voltage */
        maxD = fixedp(0.475);                   /* The maximum modulation rate */
        Ksft = fpdivs(TOFIX(-1),TOFIX(170));
        Csft = fixedp(1.1);
        sft_goal = 170;                         /* Target Voltage      */
        now_V = 0;                              /* Current Voltage     */
        sft_m = TOFIX(1);                       /* Modulation rates*/
        up_rate = 100;
     D_min = fixedp(0.1);
        fp_ten = TOFIX(10);
    /* --- PIControl ……. */
        Kp = fixedp(0.1);           /* Proportional */
        Ki = fixedp(0.01);          /* Integral */
        Vout_max = TOFIX(322);
        Vin_max = fixedp(59.3);
        lim_f_on = TOFIX(20);
        lim_f_off = TOFIX(200);                 /* Integral limiter*/
        TR_n2n1 = fixedp(7.2);      /*transformer turns ratio*/
```

```
m_min = fixedp(0.1);/* The minimum value of modulation index = 0.1 (equivalent to 45% duty）  */
        m_max = TOFIX(1);/* The maximum value of modulation index = 1.0 (equivalent to 0% duty) */
        forward = 0;                    /* Feedforward ON (0 = OFF) */
        /*- DC/AC -*/
        dcdc_Vref = 100;                /* Target Voltage */
        Kp1 = fixedp(0.1);              /* P control */
        Ki1 = fixedp(0.01);             /* I control */
        Kd1 = fixedp(0.01);
        fp_test1 = fixedp(0.5);
        fp_test2 = fixedp(0.25);
        fp_test3 = TOFIX(0);
        fp_test4 = TOFIX(1);}
/*      Voltage setting                                         */
static  void    chg_volt(FIXED _vset, FIXED _dcv)
{static FIXED   _setv;
                FIXED   _delta;
        if(!Output){
                fp_m = fp_zero;
                _setv= fp_zero;
                return;                                         }
//      _delta = fp_one;
        _delta = fp_05;
        if (_vset>(_setv+_delta)){              /* Voltage Change   */
                _setv += _delta;
        }else if (_vset<(_setv+_delta)) /* Voltage change of less than 1V*/
                {_setv -= _delta;
        }else{  _setv = _vset;          /* The required voltage         */
        }
        _vset = fpmuls(_vset,fp_rt2);   /* √2times              */
        fp_m = fpdivs(_vset,_dcv); }            /* Modulation factor*/
/*      Frequency setting                                       */
static  void    chg_freq(FIXED _fset)
{static FIXED   _setf;
                FIXED   _delta;
                INT             setf;
        if(_fset==_setf){               /* If the current frequency     */
                return;     }
        _delta = TOFIX(10);             /* 10Hz/10msec (800Hz->2KHz:1.2sec)   */
        if(!Output){
                _setf = _fset;                  /* Direct Frequency Change*/
```

```
        }else if (_fset>(_setf+_delta)){ /* Changing the frequency of 10Hz or higher/
                _setf += _delta;
        }else if (_fset<(_setf-_delta)){
                _setf -= _delta;
        }else{_setf = _fset;                                    }
        setf = TOINT(_setf);
        fp_dwt = fixedp(2.0*PI/FS*setf);        /* Δωt*/
        return;}
/*              error JOB                                       */
static  WORD    err_job(WORD err)
{static WORD    timer,cntr,cntend;
        if(err){    /* If Error */
                cntend = err<<1;    /* Double */
                cntr=0;
                timer = 25;
                LED0 = 1;                   /* LED OFF          */
                LED1 = 1;                   /* LED Off          */
                inv0_stop_pwm();    /* Inverter will stop   */
//              SET = 0;                    /* Output Suspended */
//              PFCL = 0;                   /* DC/DC OFF        */
                return (ST_ERR);}
        timer--;
        if(!timer){                 /* Time             */
                LED1 ^= 1;          /*ON / OFF LED      */
                cntr++;
                if(cntr==cntend){
                        cntr=0;
                        timer = 75;         /*Interval level */
                }else
                        timer = 25;         /* Flashing period*/
                        //if(Tmp_disp[0]>0x0f){Tmp_disp[0]=0;}
                        //else{Tmp_disp[0]++;}}
        return(0);  }
/*              Display decimal                                 */
static  INT     syousu2(FIXED fp)
{       fp &= 0xffff;               /* Lower 16 bits    */
        fp *= 100;
        fp /= 65536;
        return((INT)fp); }
/* main program
```

```
void    main(BOOL ramerr)
{       INT     i=0;
        WORD    msts=0,mtimer;
        WORD    stssv=0xff;
        FIXED   _fsav,_vsav;
        static int r_cnt;
        static int prim_Vin2;
        int Vset_cntr = 0;
        if (ramerr){
                msts=err_job(RAM_ERR); }            /* RAM error        */
        //if(!rom_sum_chk()){                       /* ROM Check sum error*/
        //      msts=err_job(ROM_ERR);              /* ROM error        */
        //} init_cpu();                             /*Initialize CPU registers*/
          initial();                    /* Initialization WORK        */
//      dispini();                      /* View: Initialization       */
        dbg_init();
        parameter_init();               /* Variable Initialization */
 {              WORD    k;
                int             j;
                for(j=0,k=1;k;k<<=1,j++){
                        if(dbgsw & k){
                        Dpsw[j]=TRUE;
                        }}}
        stt_timer();                    /* Start the timer interrupt-based    */
        sttsci1();              /* SCI interrupt*/
        if(!msts){
                ad_init(0);
                inv0_start();           }
        inv1_start();                           /* MTU Inverter operation starts    */
        mtimer =10;                             /* Time to be confirmed AD  */
        for(;;){if (Base_10m){                  /* 10msec Timer     */
                        continue;               }
                        Base_10m = 1;
                WDT_tcnt = 0x5a00;
                if (dbg_task()){
                        continue;
                }switch(sequence){
                        case RUSH_PREVENTION :
                                r_cnt++;
                                if(r_cnt >= 200){
```

```
                        r_cnt--;
                        //break;
                        if(r1_st()){
                        sequence++;}}
                break;
        /* --- Error reset--- */
        case ERROR_RESET :
                PA.DRL.BIT.B4 = 1;
                PA.DRL.BIT.B11 = 1;
                wait(100);
                PA.DRL.BIT.B11 = 0;
                sequence++;                     /* Move to the next sequence */
                break;
        /* --- Soft Start--- */
        case SOFT_START :
                if(pwm_sw){                     /* Switch  control *ON */
        if(now_V/100 <= sft_goal){      /* Is less than the target value */
                                        sft_m = fpmuls(Ksft,TOFIX(now_V/100)) ;
chk_sft_m = TOINT(fpmuls(sft_m,TOFIX(100)));
                                        now_V = now_V + up_rate;
                                        inv0_start_pwm();     /* PWM generator */
                        }else if(now_V/100 > sft_goal){
                                        pi_m = sft_m;
                                        sequence++;
                }else{      now_V = 0;
                                        sft_m = TOFIX(1);
                break;
        case MAIN_LOOP:
                break;
        default:
                break;}
#endif
```