APPENDIX-A DEVELOPMENT TOOLS

This chapter describes a comprehensive study of supporting tools and the work done by the researchers using soft computing techniques and algorithm development with the use of MATLAB Simulink , real time workshop toolbox ,code composer studio (CCS) for the controller design of motors & drives system.

A.1 MATLAB-SIMULINK

MATLAB is matrix laboratory software which provides design, analysis and data visualization tools for system design. Simulink is a collection of different tool boxes of MATLAB used for the system modeling, simulation and validation. Real Time Workshop is used for code generation from simulink model and provides framework for running the code in real time. It is possible to perform Continuous time Simulation, discrete time simulation as well as combined continuous and discrete time simulation as well as linear and non linear system simulations. Simulink library contents different readymade block sets. User may also modify the parameters of readymade block sets .It also allows user to create simulation blocks based on mathematical modeling of the system. It also provides different sources and connectors to build the system model using the block sets. In simulink model simulation parameters may configured using simulation parameter dialog box. Simulation can be “start” and “stop” using start/Stop button in tool bar. It is also possible to pause the simulation to observe the results of particular operating point. To see the output results it also provides different types of scopes and display blocks. The output results may be sent to workspace or to a particular file for further analysis of the results.

A.2 MATLAB Real Time Workshop and Embedded Coder

The Real time workshop requires installation of MATLAB, Simulink and C/C+ compiler software. The Real-Time Workshop generates code from MATLAB, Simulink, and Stateflow models. The Real-Time Workshop along with MATLAB and Simulink and code composer studio generates assembly language code for DSP based real-time systems. Real-Time Workshop also can be used for hardware-in-the-loop (HIL) simulations. The real time workshop provides flexibility of code generation of specific target hardware implementation. It is used for code generation of real time control system design, real time signal processing and for prototype testing applications. The code generated using real time workshop may be in C language code, DSP assembly language code

or in.m file code format. Real time workshop used to generate code form MATLAB- Simulink compatible with the target hardware. It is possible to change the parameters of the algorithm or model even during the real time running of algorithm on hardware using Simulink's external mode. The Real-Time Workshop is also compatible for the bundled target systems such as DOS and Tornado, as well as generic real-time programs under Windows95, Windows NT, and UNIX. DSP Blocks, Communications Toolbox blocks, all Simulink blocks and Fixed Point Blocks are compatible for RTW. Real-Time Workshop may be used in two stages.

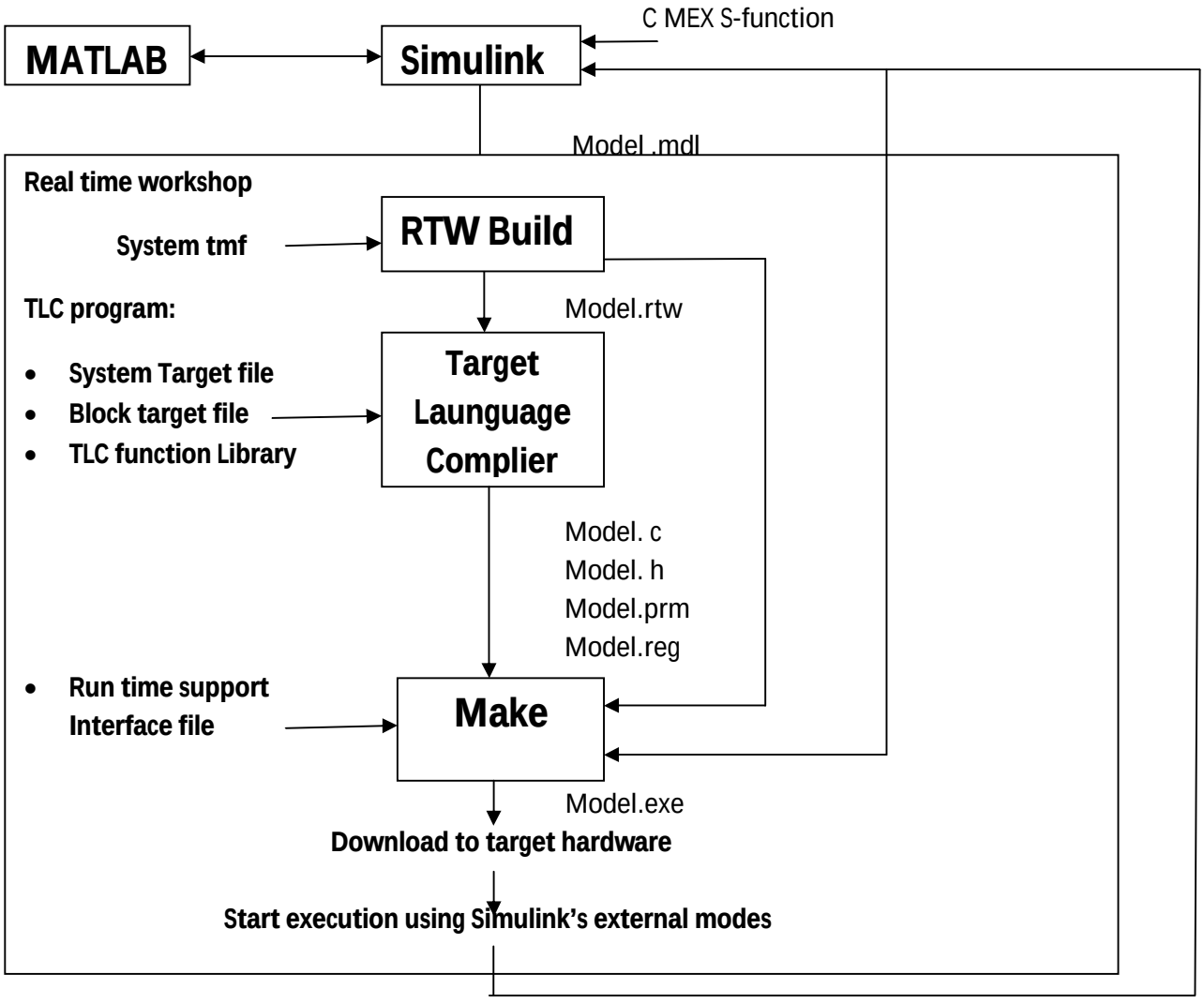

Figure A. 1 Real time workshop architecture

To generate standalone code for the simulation and to validate the generated code and then second stage is to download and run the generated code in real time environment [1-13]. Real time workshop Architecture is as shown in Fig.A.1

Real-Time Workshop generates code from fixed step models, automatic program building and it supports third party hardware tools. It also provides framework for different operating environments such as in DOS, Tornado, and generic real-time environments and different third party hardware modules and tools. In the Build Process it is possible to configure the environment parameters such as dSPACE or Wind River Systems Tornado Development Environment etc. for the code generation. This process generates .rtw file. Real time workshop tools also consist of Target language compiler (TLC) module; Make utility module and S-function Application Program Interface (API). The TLC transforms a description generated by the Real-Time Workshop of Simulink model into target specific code. The TLC program consists of system target file, block target files and a TLC function library. Make Utility is used to compiles and links the generated code into executable code. S-Functions allow adding custom code into generated code and also using of S-function Application Program Interface (API)[1-13]. Files developed by the build process are given as below

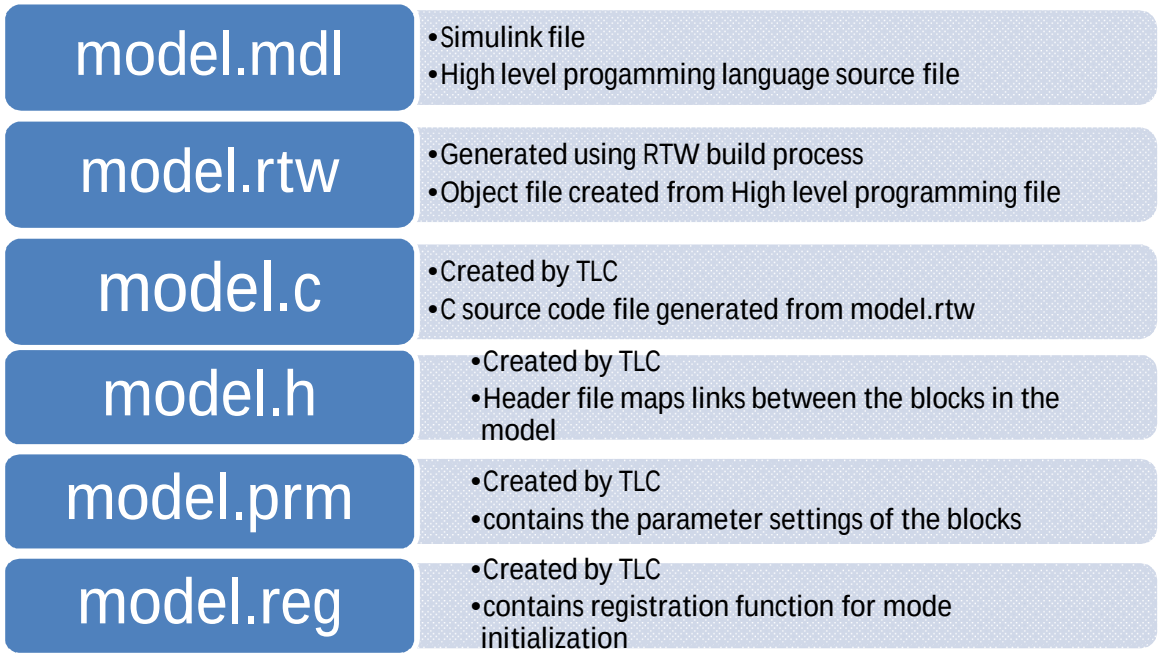

Figure A. 2 Files developed by built Process

Embedded coder

Embedded Coder used to generate codes for embedded processors, microprocessors and third party hardware modules such as TI DSPs. It also supports MATLAB and SIMULIK coder configurations and optimizations. Embedded coder generated codes are compact, optimized for specific target. Types of Target Environments Supported by Embedded Coder is as follows,

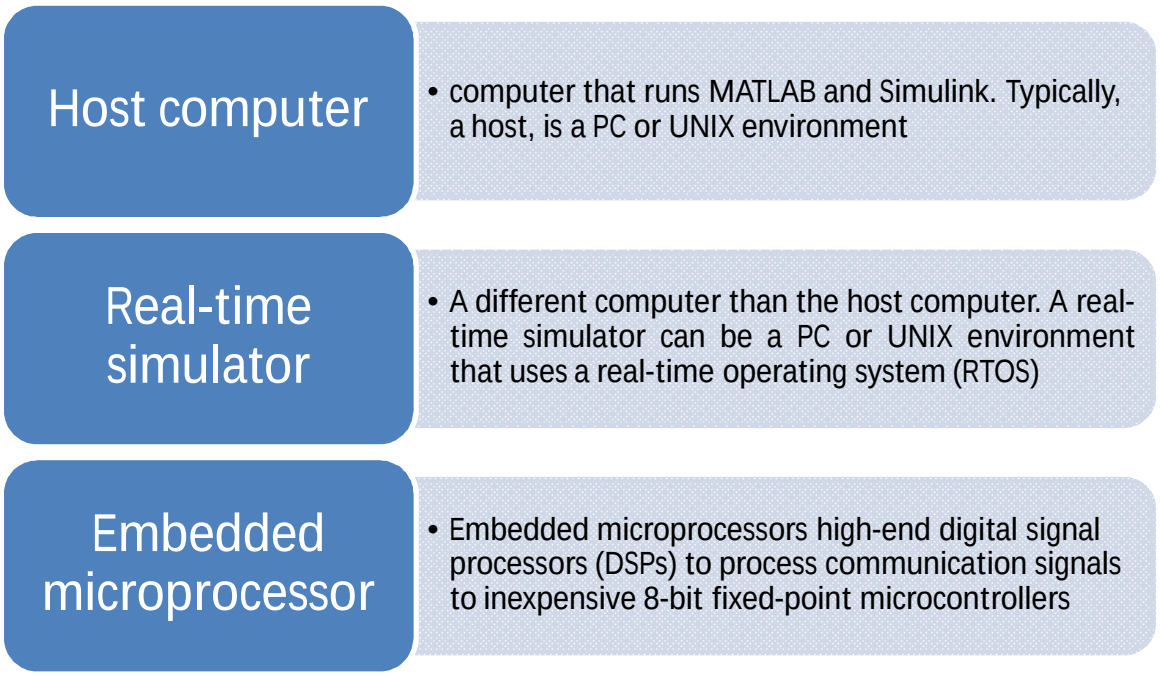

Figure A. 3 Target Environments supported by Embedded Coder

Applications supported by embedded coder for the above Target Environments are as below,

Host Computer

- Accelerated simulation
- Rapid simulation
- System simulation
- Model intellectual property protection

Real-Time Simulator

- Rapid prototyping
- System simulation
- On-target rapid prototyping

Embedded Microprocessor

- Production code generation
- SIL and PIL Simulation
- Hardware-in-the-loop (HIL) testing

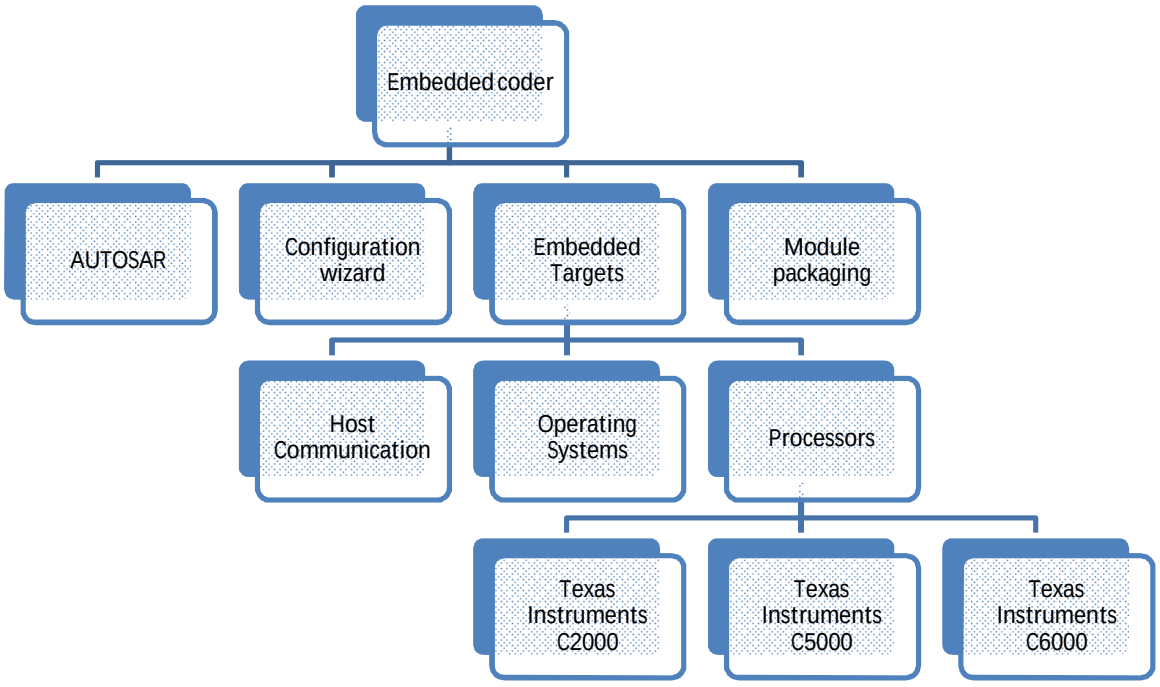

Figure A. 4 Embedded Coder Block sets

MATLAB SIMULINK Library browser of embedded coder consists of block sets as shown in above Fig.A.4. The embedded coder consists of AUTOSAR, Configuration wizard, Embedded Targets and Module packaging block sets. The embedded Targets blocks can be further classified in Host Communication block sets, Operating Systems block sets and Processor block sets. The Processor block sets supports third party Texas instruments of C2000, C5000 and C6000 series of real time hardware modules. Texas Instruments C2000 series supports DSPs like C2802x, C2803x, C2806x, C280x, C281x, C2834x and C28x3x. It also consists of block sets for Memory Operations, optimizations, RTDX Instrumentation, Scheduling, Target Communication and Module packaging[1-13].

C28x3x block library consists of

- ADC block - perform analog-to-digital conversion of signals
- CAN calibration protocol - implementation of a subset of the CAN Calibration Protocol (CCP)
- Digital Input and Output- This block configures the general-purpose I/O (GPIO) MUX registers that control the operation of GPIO shared pins for digital input and output.
- I2C receive and I2C transmit- Configure the I2C module to receive/transmit data from the two-wire I2C serial bus
- SCI receive and SCI transmit- Receive/transmit data on target via serial communications interface (SCI) from host.
- Software interrupt trigger- Generate software triggered non mask able interrupt.
- Watchdog- Configure counter reset source of DSP Watchdog module
- eCAN transmit and eCAN receive block- generates source code for transmitting/receiving eCAN messages through an eCAN mailbox
- eCAP- Receive and log capture input pin transitions or configure auxiliary pulse width modulator
- eQEP- used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in motion and position-control systems.
- ePWM- Configure Event Manager to generate Enhanced Pulse Width Modulator (ePWM) waveforms.

Enhanced Pulse width Modulator (ePWM)

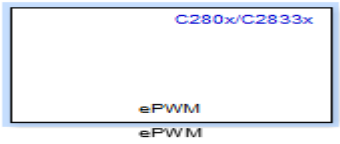

Figure A. 5 ePWM MATLAB SIMULINK Block

It Configures the Event Manager of the C28335 DSP to generate ePWM waveforms. These DSPs contains six ePWM modules and each module has two outputs, ePWMA and ePWMB[1-13]. Following parameters need to set to generate PWM signals.

- **Timer period units**- Specify the units of the Timer period or Timer initial period as Clock cycles
- **Specify timer period via** – specify via dialog or Input port
- **Timer Period-** Set the period of the PWM waveform in clock cycles or in seconds
- **Reload for time base period register (PRDLD)** - The time at which the counter period is reset.

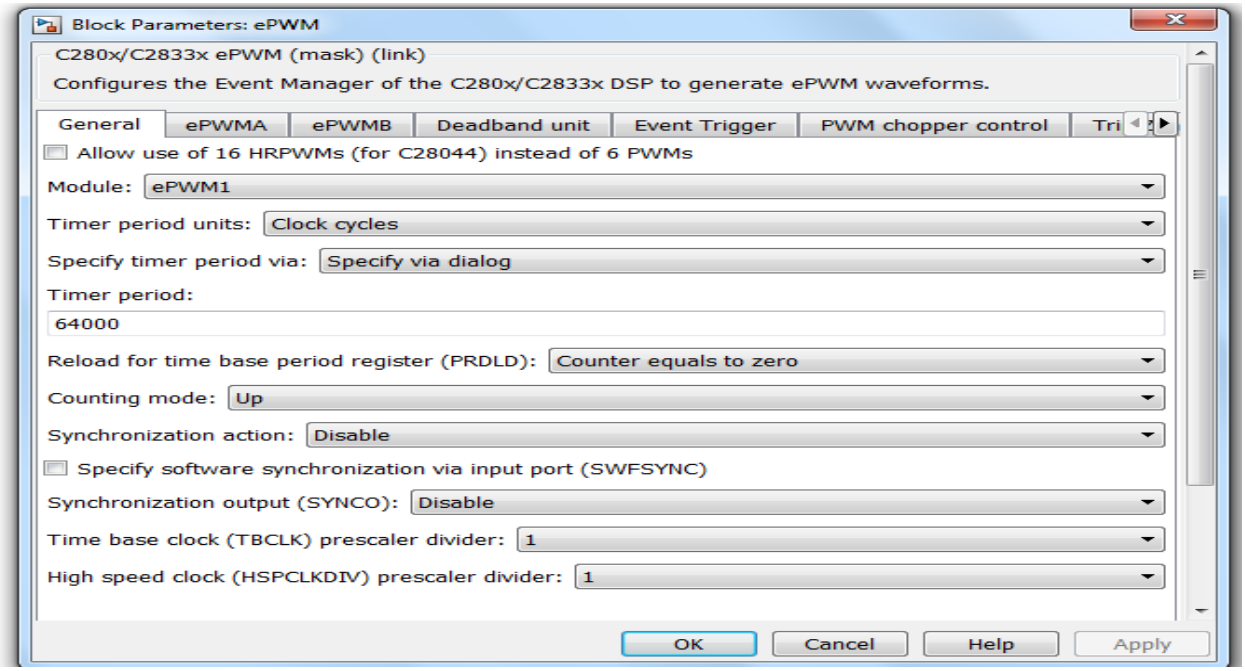

Figure A. 6 General Block Parameters of ePWM

- **Counting mode-** ePWM module can operate in three distinct counting modes: Up, Down and Up-Down.
- **Synchronization action** - Specify the source of a phase offset to apply to the Time-base synchronization input signal from the SYNC input port
- **Synchronization output (SYNCO)** -this parameter to specify the event that generates a Time-base synchronization output signal from the Time-base (TB) sub module.

- **Time base clock (TBCLK) prescaler divider & High speed time base clock (HSPCLKDIV) prescaler divider** - this parameters use to configure the Time-base clock speed (TBCLK)

On ePWMA /ePWMB pane following parameters need to configure for desired PWM signal generation CMPA units, CMPB units-Specify the units used by the compare register

- **Reload for compare A Register (SHDWAMODE), Reload for compare B Register (SHDWBMODE)-** The time at which the counter period is reset is need to configure for following conditions,

Action when counter=ZERO, Action when counter=period (PRD), Action when counter=CMPA on up-count (CAU), Action when counter=CMPA on down-count (CAD),Action when counter=CMPB on up-count (CBU), Action when counter=CMPB on down-count (CBD)

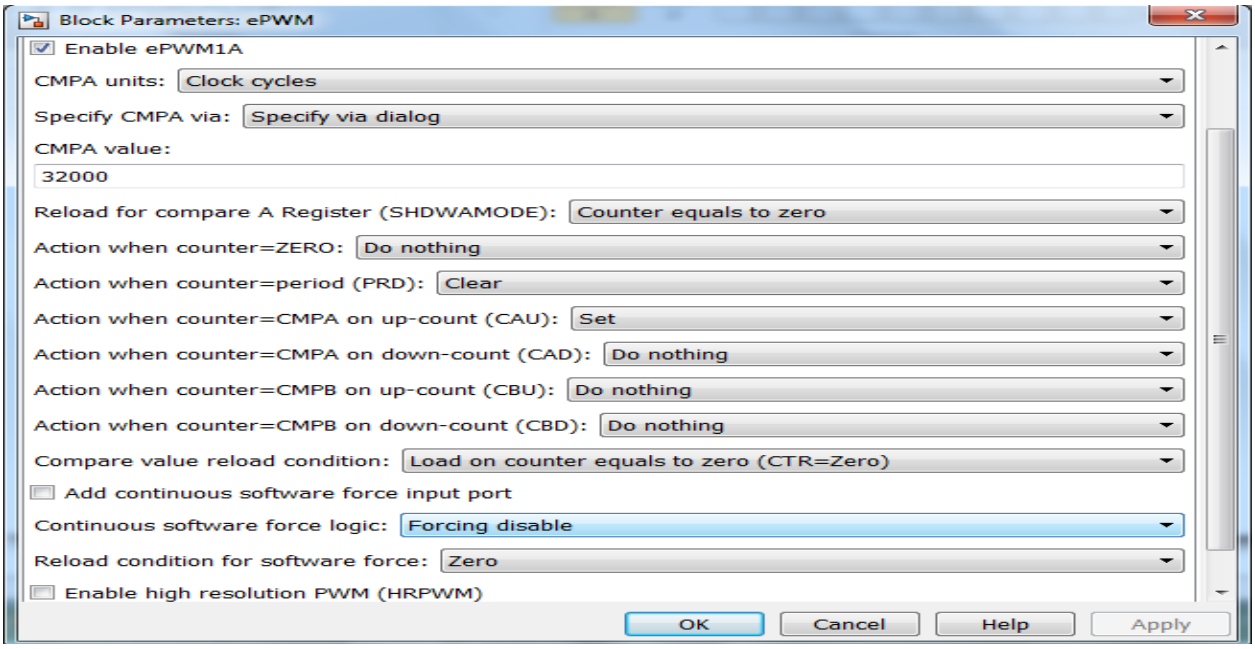

Figure A. 7 Block Parameters of ePWMA

- **Inverted version of ePWMxA-**Invert the ePWMxA signal and output it on the ePWMxB outputs.
- **Enable high resolution PWM (HRPWM)-** enable When the effective resolution for conventionally generated PWM is insufficient.

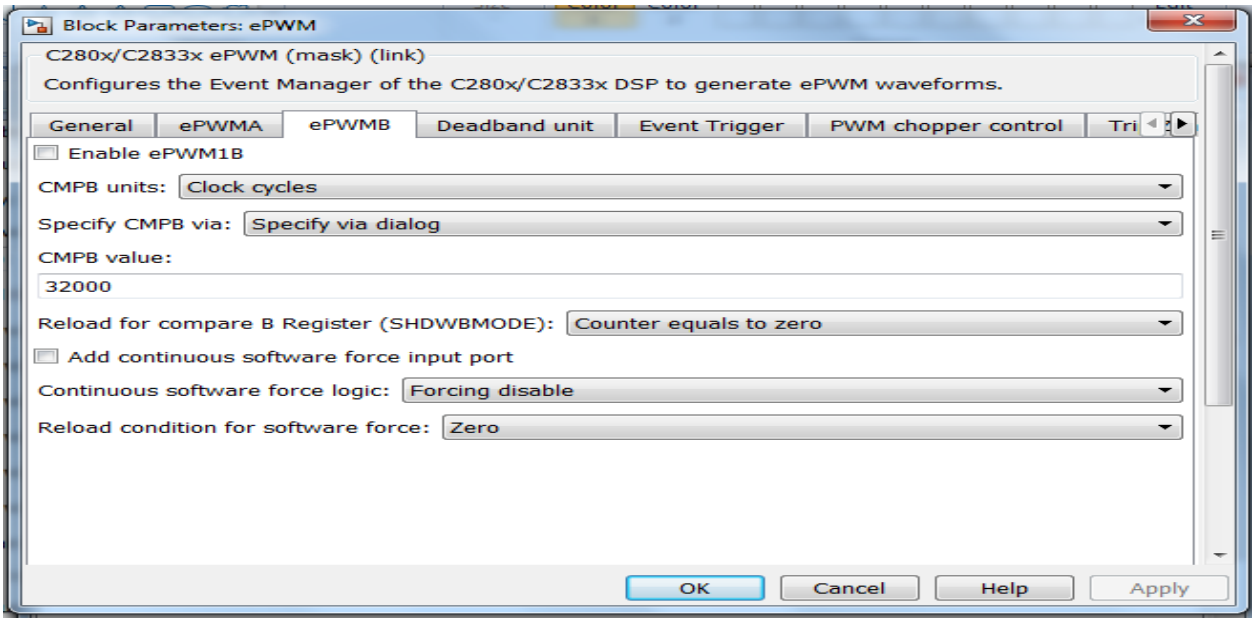

Figure A. 8 Block Parameters of ePWMB

Deadband Unit Pane

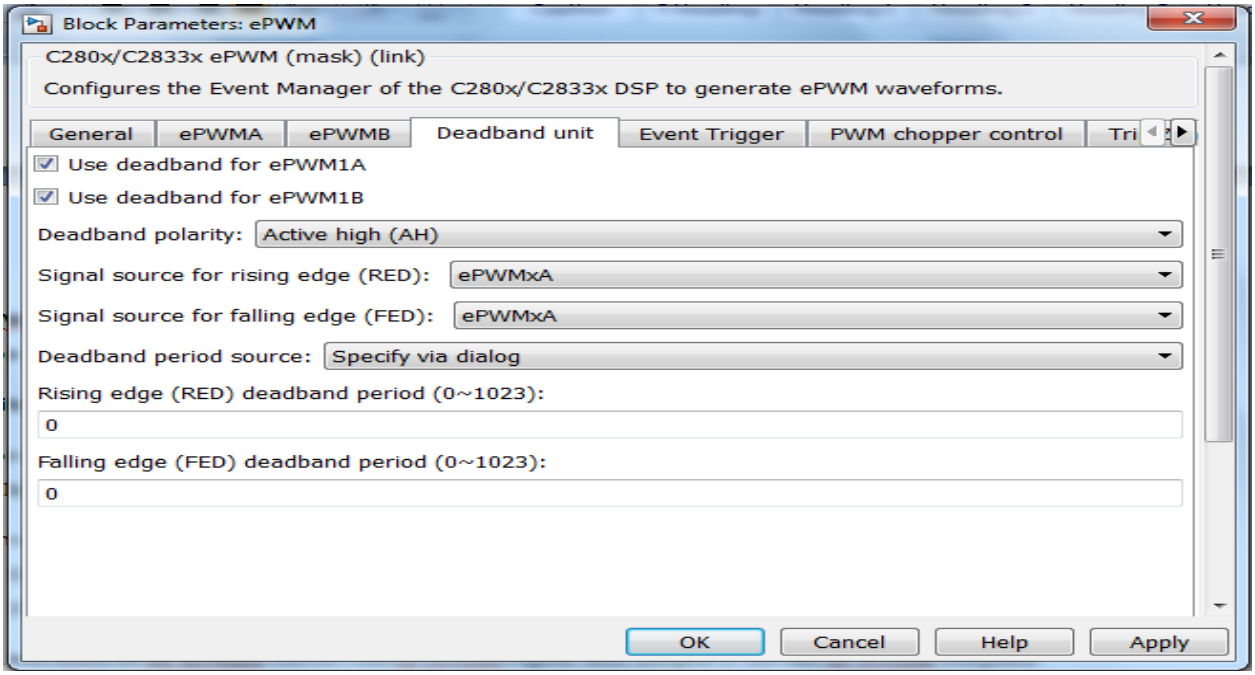

Figure A. 9 Block Parameters of Deadband Unit

- **Use dead band for ePWMxA, Use dead band for ePWMxB-** Enables a dead band area of Rising Edge Delay or Falling Edge Delay cycles without signal .
- **Enable half-cycle clocking-**To double the dead band resolution,
- **Deadband polarity-**Configure the dead band polarity as Active high (AH) , Active low (AL) , Active high complementary (AHC) or Active low complementary (ALC).
- **Signal source for rising edge (RED)-**Use dead band for ePWMxA .
- **Signal source for falling edge (FED)-**Use dead band for ePWMxB
- **Deadband period source-**Specify the source of the control logic.
- **Rising edge (RED) dead band period (0~1023)-** The value for dead band delay for **ePWMxA**
- **Falling edge (FED) dead band period (0~1023)-** The value for dead band delay for **ePWMxB**

ePWM also consists of Event Trigger Pane, PWM Chopper Control Pane ,Trip zone unit and Digital compare unit

- **Event Trigger Unit** - It is use to configure ADC Start of Conversion (SOC) by one or both of the ePWMA and ePWMB outputs.
- **PWM Chopper Control unit**- uses a high-frequency carrier signal to modulate the PWM waveform generated.
- **Trip Zone Unit** - signals can be used to force the ePWM output into a specific state Digital Compare - uses for generating output s with reference to external events or signals.

A.3 Fuzzy Logic

Fuzzy sets and logic theory were introduced by Professor Lotfi Zadeh in 1965. Today Fuzzy logic control system was found in washing machines, Air condition units, speed governor systems, speed control systems of motors and drives. The main advantage of fuzzy logic controller design approach is to use of knowledge gained from the experience in the design. Plant nonlinearity, Plant uncertainty, Multivariables , multi loops and environment constraints, measurements uncertainty, time variant behavior of Plants limits the use of conventional controllers. Fuzzy controllers are more robust than PID controllers can operate with disturbances. Developing a fuzzy controller is cheaper than the conventional controller. Fuzzy controllers are customizable using human operator's strategy in natural linguistic terms. Fuzzy logic controller designed for complex applications such as

Automatic control of dam gates for hydroelectric power plants , control of robots ,control of car engines ,Cruise-control for automobiles ,Optimized planning of bus timetables , Prediction system for early recognition of earthquakes, Automatic motor-control for vacuum cleaners with a recognition of a surface condition, rotor transmission, servo control, missile warning [14,15].

The controller design approach contains the following steps

1. Define the input and control variables

2. Convert the crisp input and control variables using fuzzy set theory into fuzzy inputs.

3. Design the rule base considering control conditions and constrains.

4. Design the algorithms to perform fuzzy computations to obtain fuzzy outputs.

5. Define rules to convert fuzzy output into crisp control actions.

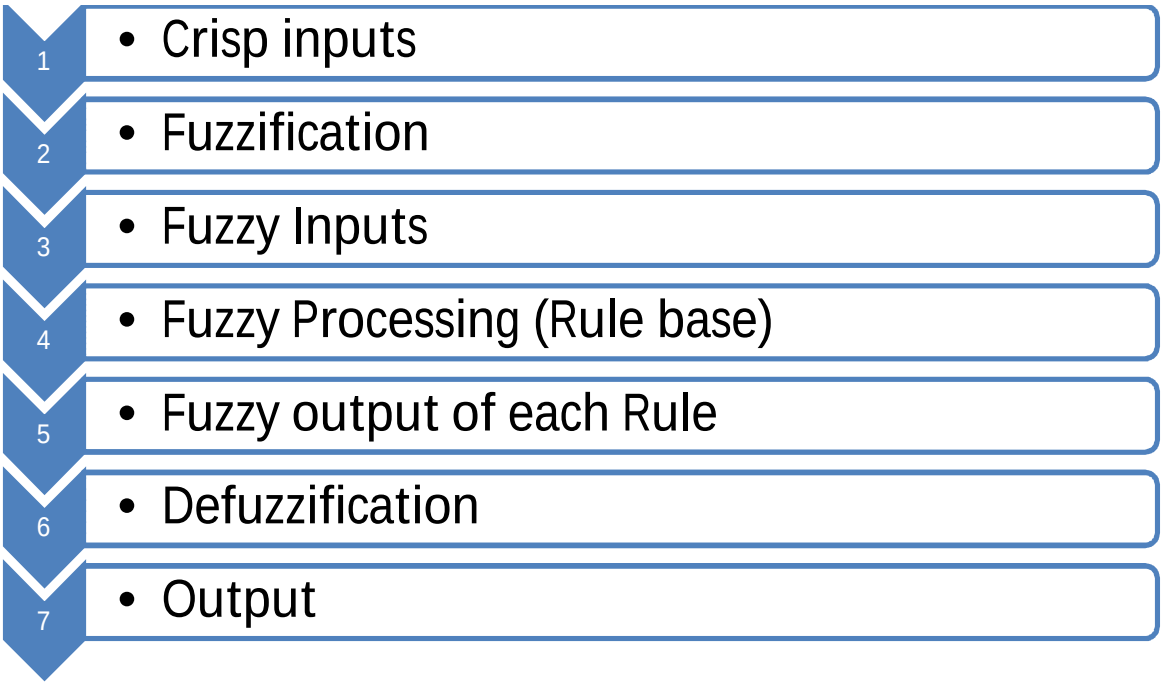

Figure A. 10 Structure of Fuzzy Controller

Linguistic variable is made up of words, sentences or artificial language and used to describe the inputs and outputs of the FLC. Negations, Disjunction, Conjunction and Implication are the fuzzy logic operators. Implication is a most common operator for Fuzzy Logic based control system design because it is based on IF-Then rules. Mamdani's Implication technique is suitable for control

applications and also for hardware implementation of FLC [14-16]. The structure of Fuzzy controller is as shown in Fig.A.10.

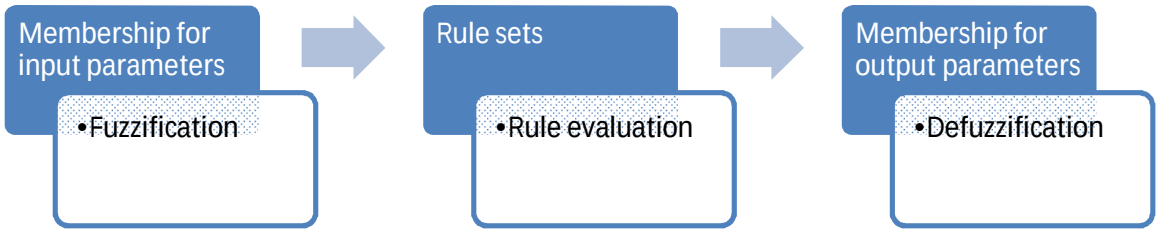

Figure A. 11 Operational block diagram of Fuzzy controller

Fuzzy system development software toolboxes available from the many organizations/developers such as Fuzzy RT/Fuzzy Toolbox for MATRIX by Integrated Systems Inc., Fuzzy Logic Toolbox for MATLAB by The Math Works Inc., FIDE by Aptronix , fuzzy Tech. by Inform, a number of products by Togai Infra Logic Inc., Fuzzy Systems Engineering Inc., Hyper Logic etc.

Fuzzy Logic tool box MATLAB

Fuzzy Logic Toolbox provides MATLAB functions, apps, and a Simulink block for analyzing, designing, and simulating systems based on fuzzy logic[15,16]. Fuzzy Logic tool box provides

• Membership functions

• Mamdani and Sugeno-type FIS

• Use of FLC in a Simulink model

Fuzzy logic is conceptually easy to understand, flexible, can model nonlinear functions and based on the experience of experts. Fuzzy logic can be blended with conventional control techniques and based on natural language. Fuzzy tool box provides support for fuzzification, fuzzy processing based on rulebase , and defuzzification. It is also possible to integrate simulink and other toolboxes like Neural network, Optimization etc.

Membership functions

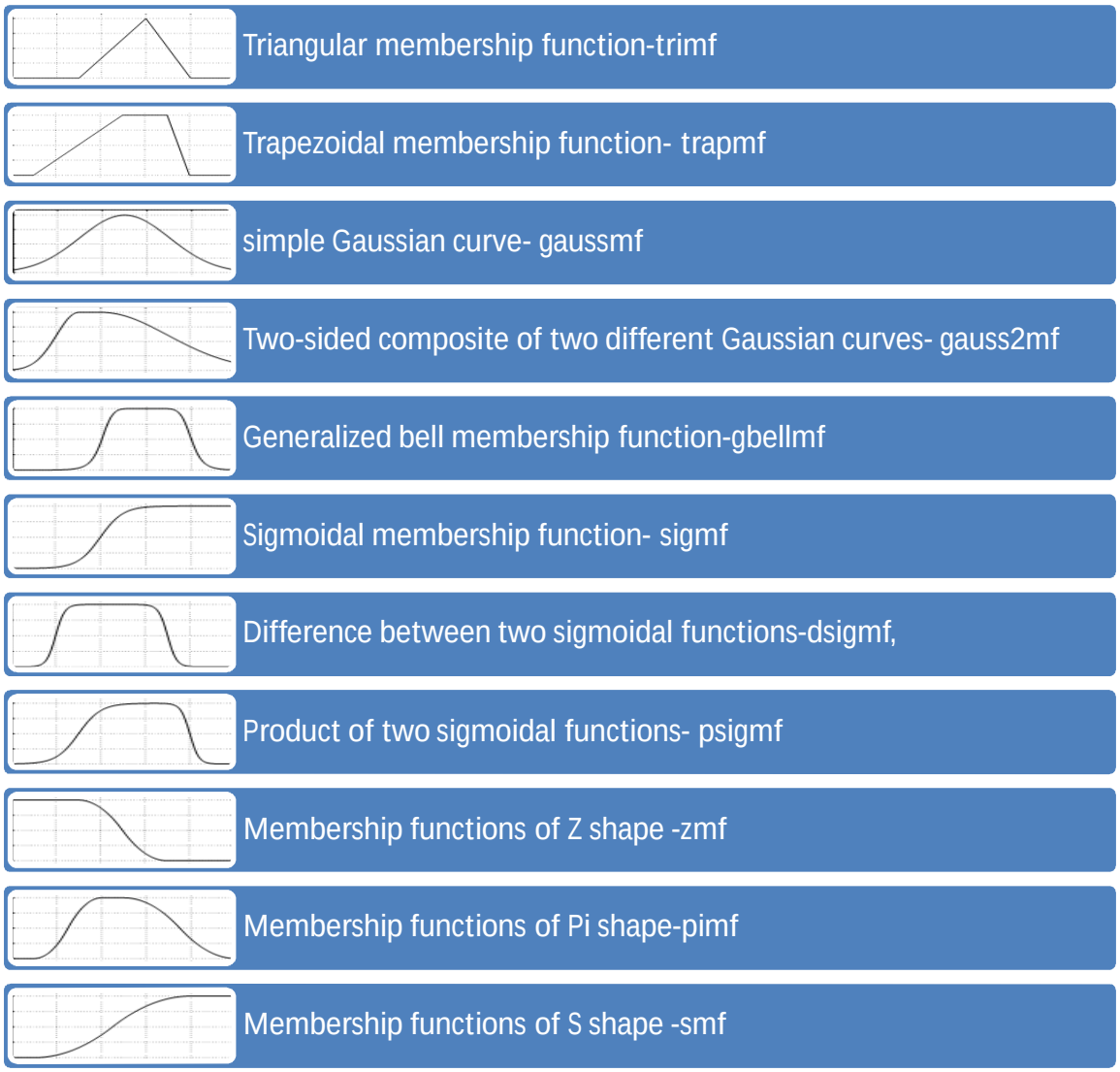

Figure A. 12 Membership functions in Fuzzy Logic tool box-MATLAB

Fuzziness of the input is represented by the Membership functions. The toolbox provides 11 built in membership functions based on

- piece-wise linear functions
- the Gaussian distribution function
- the sigmoid curve
- quadratic and cubic polynomial curves

A membership function for a given fuzzy set maps an input value to its appropriate membership value.

Fuzzy Inference system

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. Fuzzy inference systems is broadly used in Fuzzy logic based controller design, automatic control system design, pattern classification applications. Fuzzy inference based on Boolean logic operations. If-Then rules statements are formed using different control conditions using Fuzzy logic. If-Then rule formation process depends on different forms of statements like Assignment statements, Conditional statements, Unconditional statements. Fuzzy inference systems (FISs) are also known as fuzzy rule-based systems, fuzzy model, based on this the output is obtained. There are two types of fuzzy inference system Mamdani and Sugeno type. These methods are differentiating based on how output is determined. Mamdani's Fuzzy inference method is most common method for fuzzy logic controller design. First fuzzy control system is proposed in 1975 by Ebrahim Mamdani. Mamdani inference system expects Fuzzy sets of output and therefore each output variable needs defuzzificaton. Some control system requires output from single membership function instead of all output variables. Sugeno type fuzzy inference system is preferred for these types of control systems where output membership function is linear in nature. Fuzzy inference process comprises of five parts:

1. Fuzzification
2. Application of the fuzzy operator
    - AND methods- min (minimum) and prod (product)
    - OR methods -max (maximum) and the probabilistic OR- probor
3. Implication –two built in method
    - min (minimum)
    - prod (product)
4. Aggregation -Three built-in methods
    - max (maximum)
    - probor (probabilistic OR)
    - sum (simply the sum of each rule's output set
5. Defuzzification-five built-in methods supported
    - centroid
    - bisector

- middle of maximum (the average of the maximum value of the outputset)
- largest of maximum
- smallest of maximum

Use of FLC in a Simulink model

Fuzzy logic tool box supports use of simulink models. Fuzzy logic controller designed using Fuzzy GUI tool implemented in Simulink model for simulation study. Fuzzy logic block sets are available in simulink library to implement fuzzy logic controllers in the simulink. The Fuzzy Logic Toolbox library contains simulink blocks of the following

- Fuzzy Logic Controller
- Fuzzy Logic Controller with Rule Viewer
- Membership Functions

Fuzzy Logic controller block implements fuzzy inference system and fuzzy Logic controller with rule viewer implements fuzzy inference system with refresh rate and rule viewer and it also shows rule evaluations. These blocks are with the fuzzy wizard and it can be used with built in functions of or Method: max, and Method: min, prod ,imp Method: min, prod and aggMethod: max.fuzzy logic tool box also supports unix, windows environment for fuzzy inference system implementation. It is also possible to integrate other simulink toolboxes like power system tool box, neural network tool box ,control system tool box for the fuzzy logic based controller design and implementation.

A.4 Genetic Algorithm

Simulated annealing, Ant colony optimization, Random cost, Evolution strategy, Genetic algorithms are the non conventional famous search and optimization methods for engineering optimization problems. Simulated annealing and Evolution algorithms are the guided random type search techniques. Genetic algorithms and Evolutionary strategies are developed using the principle of natural selection and natural genetics. Evolutionary computing was introduced by Rechenberg in 1960s.Genetic algorithm (GA) uses the concept of Darwin's theory of evolution, "survival of the fittest." the new generation came into existence through the process of reproduction, crossover, and mutation.GA have been developed by John Holland his colleagues and students at Michigan university during the design of artificial system software in early 1970s.GA is a search procedure based on random choice of parameters. GA formation inspired by the biological concept of organ

formation. Different terms of GA are defined from biological terms such as chromosome, substring, gene, population and generation. GA maps all Optimization problems in terms of strings to find the solution. Performance of GA mainly depends on objective function, generic representation and on genetic operators. GAs is suitable for maximization optimization problems as it is based on the theory of Darwin. Therefore to solve the Minimization problem, some transformation is required. Different generic operators are Inversion, Dominance, Deletion, Intra chromosomal duplication, translocation, segregation, speciation, Migration, sharing, mating[17-19]. Simple GA is composed of three operators

1. Reproduction
2. Crossover
3. Mutation

Reproduction – Individual strings are copied according to their objective function values. Strings are copied according to their objective function values. Strings with higher fitness values have more chance of contributing in next generation. The various methods to implement reproduction operators are roulette wheel, Boltzman selection, Tournament selection, Rank selection and steady state selection. The famous and easiest way to implement reproduction operator in algorithm is by roulette wheel. Reproduction process is done by spinning the weighted roulette wheel. In this way more highly fit strings have more numbers of offspring in the next generations. Once a string has been selected for reproduction, its exact replica also created. This string is entered in a matting pool and used to generate new population. Crossover is the next operation after reproduction.

Crossover includes selection of random pair of two individual strings, random selection of cross site along the string length and the position value swapping. Crossover- crossover may proceed in two steps. In first step members of newly reproduced strings are mated randomly in matting pool. Two strings are picked from the matting pool randomly. Each pair of strings undergoes partial exchanges with single point crossover, two point crossover, multi point crossover, and uniform crossover or with the matrix crossover method.

Mutation- mutation is the random alternation of the value of a string position. Mutation protects loss of useful information during when reproduction and crossover becomes overzealous .Mutation rates are small in natural populations thus it considered as secondary mechanism of genetic algorithm adaptation. This operator is used to maintain diversity of the population. The mutation operator

generates new generic structures in the population randomly. It also helps to come out from the local minima. Mutation causes movement in the search space and it may restore lost information.

GA is more robust and efficient because it reproduces high quality notions based on the performance and crossing these notions with other high performance notions from other strings. Cross over of high performance notions forms new ideas.GA is robust method compare to conventional search methods. GA is different from conventional search methods for following criteria,

- GAs work with set of natural parameters
- Search from populations and via sampling
- Its development based on Objective functions
- It is based on probabilistic transition rules

GA is more preferred for the application domains like control system, design, scheduling, robotics, machine learning and signal processing. Global Optimization Toolbox of MATLAB consist of global search, multi start, pattern search, genetic algorithm, and simulated annealing solvers to solve optimization problems. It is also possible to create a custom GA for multi objective by defining parent selection, crossover, and mutation functions. This tool box also works with the combination of other toolbox and also with simulink [17].

Steps to solve optimization problem using GA in global optimization toolbox of MATLAB

- Define Objective functions and constraints
- Set appropriate parameters in GA options
- Run the solver
- Examine the results in solver outputs & Iterative display pane
- If results are un satisfactory then change your options, start points and run the solver once again

GA is preferred for the problems of non smooth objectives or constraints and for the desired solutions like single local, single global solutions. GA performs well when objective function is discontinuous, non differentiable, stochastic, or highly nonlinear.GA uses selection, crossover and mutation rules to create the next generation from the current population. In MATLAB Optimization

tool box, GA applications design & Implement by calling GA function file from command window or from GA Graphical User Interface (GUI) tool in the application tool bar [17].

GA function syntax is given by,

$$[x\ fval] = ga(@fitnessfun, nvars, options) \qquad (A.1)$$

Where *x*= point at which final value attained

fval= Final value of the fitness function

nvars =No. of Independent variables in the fitness function

Options= Different parameters for population, fitness scaling, selection, reproduction, crossover, mutation, migration, constrains, plot function, output function, Hybrid function, stopping criteria ,display to command window and user function evaluation

GA using GUI tool

Write "optimtool('ga')" in MATLAB command line to start GA in GUI application. Optimization tool in the Problem Setup and Results pane as shown in Fig. ,

- select "ga- Genetic Algorithm " in the solver dropdown menu
- Fitness function- write .m file name of the fitness function in a format @filename.m
- Numbers of variables- write no. of variables for the fitness function.

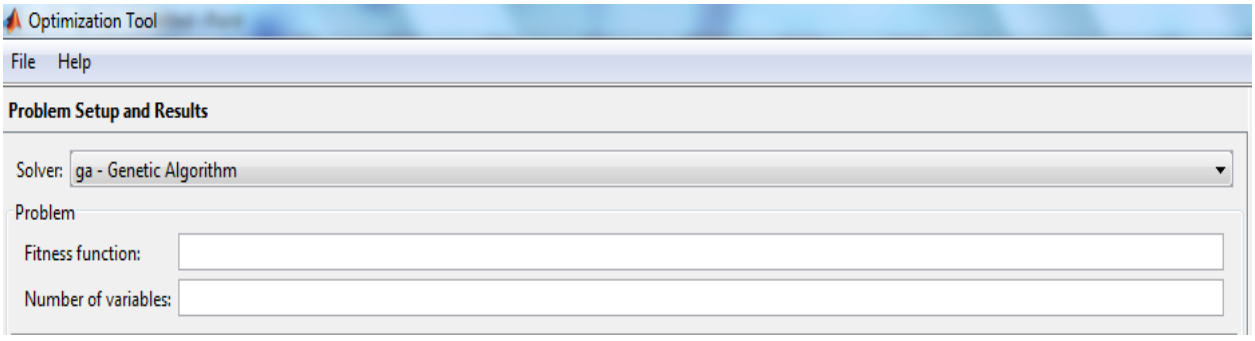

Figure A. 13 Variables of Problem setup and Results

Define constraints for constrained optimization problem solution using GA as shown in Fig.A.14.

- Write the values of constrains like Linear inequalities, Linear equalities, upper and lower bounds, Nonlinear constraint function and Integral variable indices.

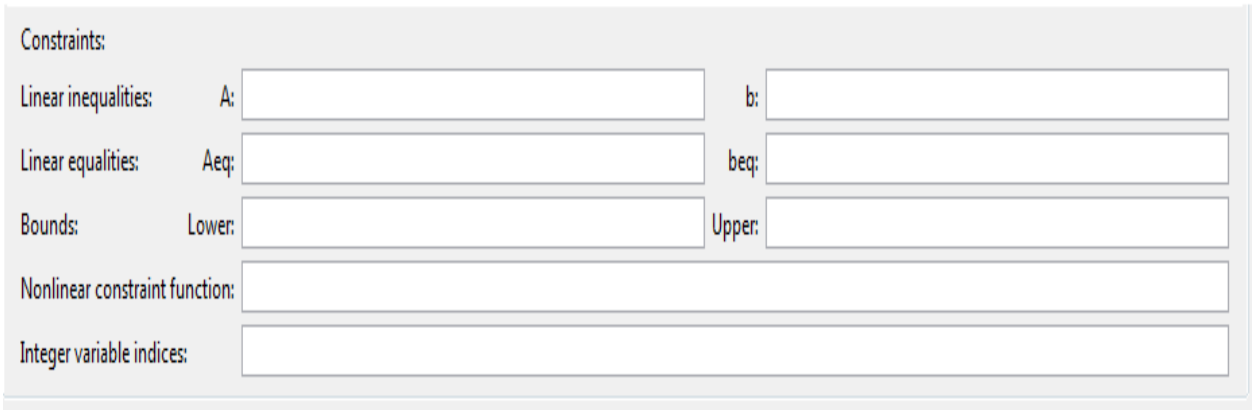

Figure A. 14 Constraints of GA

Figure A. 15 Options of GA tools

The different options available in Optimization tool box are as shown in fig. This menu also includes Plot function for the plotting of the results. Finally to run GA, click on the start button as shown in Fig A.16. Optimization tool displays the results of the optimization in Run solver and view Results pane

Figure A. 16 Run Solver dialog box

HARDWARE BASED SIMULATION

This section presents detail specification & features of Digital Signal Processor (DSP) TMS 320F28335 module & Spectrum Digital C2000 series XDS510LC USB JTAG Emulator. It also presents the methodology of interfacing MATLAB with Code composer studio to implement the algorithms in DSP using JTAG Emulator.

A.5 DSP Module

The C2000 series of microcontroller includes TMS320x281x, TMS320x2833x and TMS320x2823x devices. This all devices are useful for developing real time control applications. C28x Floating point central processing unit is the feature included in the device TMS320x2833x. The2833x/2823x devices includes new control peripherals Enhanced Pulse width modulator module (ePWM), Enhanced capture module (eCAP) and Enhanced Quadrature Encoded Pulse (eQEP). It is very efficient device for developing system control algorithms. Instructions for this device are the extension of the instruction sets of earlier devices. Algorithms developed for the earlier devices are compatible with the device TMS320x2833x.Practice board of TMS 320F28335 is as shown in Fig.A.17

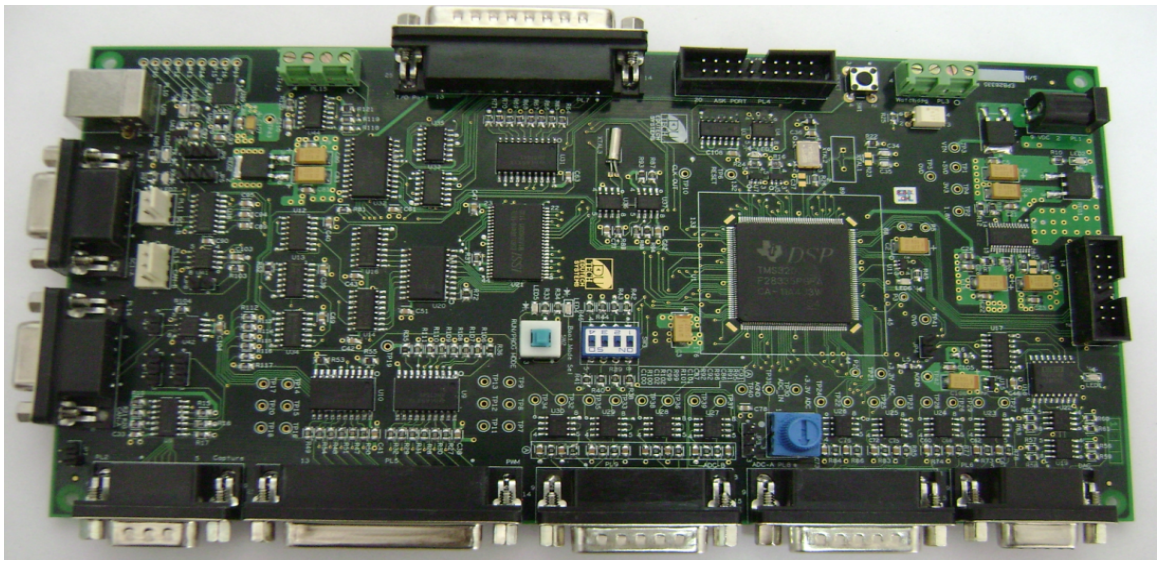

Figure A. 17 Practice board of DSP F28335

EPB28335 Specification are given as follows,

Input Voltage - 9V DC

Main features of DSP TMS320F28335

- C2000 series TMS320F28335 Digital Signal Controller with 150 MHz operating speed
- 68K bytes on-chip RAM
- 512K bytes on-chip Flash memory
- JTAG emulation connector
- 6 channel DMA controller (For ADC, McBSP, ePWM, XINTF, SARAM)

DSP F28335 Module connectors

- Power jack connector
- Capture
- Watchdog
- ASK Port
- PWM
- DAC
- I/O Port
- ADC-A & ADC-B Module
- USB
- SCI-A & SCI-B
- CAN-A & CAN-B
- JTAG

EPB28335 has an RS-232 connector which brings out the PWM signals. PWM output signals available from pin number 1 to pin number 12.The EPB28335 is also supplied with a 14-pin header interface for JTAG emulators to interface to Texas Instruments DSPs.

A.6 JTAG Emulator & PWM Isolator module

The C2000 XDS510LC JTAG Emulator is used with Digital Signal Controllers (DScs) and microprocessors through JTAG interface. Emulator allows the user direct access between the host computer and the TMS320C2000 Platform. Emulator helps to develop applications with Digital Signal Controllers. A JTAG emulation connection is required for debugging software, downloading code, and flash programming.

• It uses with Texas Instrument's Digital Signal Controllers

• Advanced emulation controller provides high performance.

• Supports USB interface with host PC.

• Power provided by host USB port or USB hub

• Compatible with Texas Instruments Code Composer Studio

• Compatible with Spectrum Digital's Flash programming utility

• Compatible with Windows 2000, and Windows XP Operating Systems

PWM Isolator

The PWM Isolator module takes +5V DC PWM waveform as input and gives +15V DC PWM signal output and it also provides isolation to the DSP Module. It is useful for Inverter application. Specification & features are as given below

- Input Power supply range is from +9V DC to +36V DC
- Output +5VD rating is 5V DC and for +15V_SK1 is 15V DC of ASK-29.
- Thermal Trip
- Errors signal capture facility available from inverter
- 12 channel input and output for PWM
- fuse protection at high current and high voltage

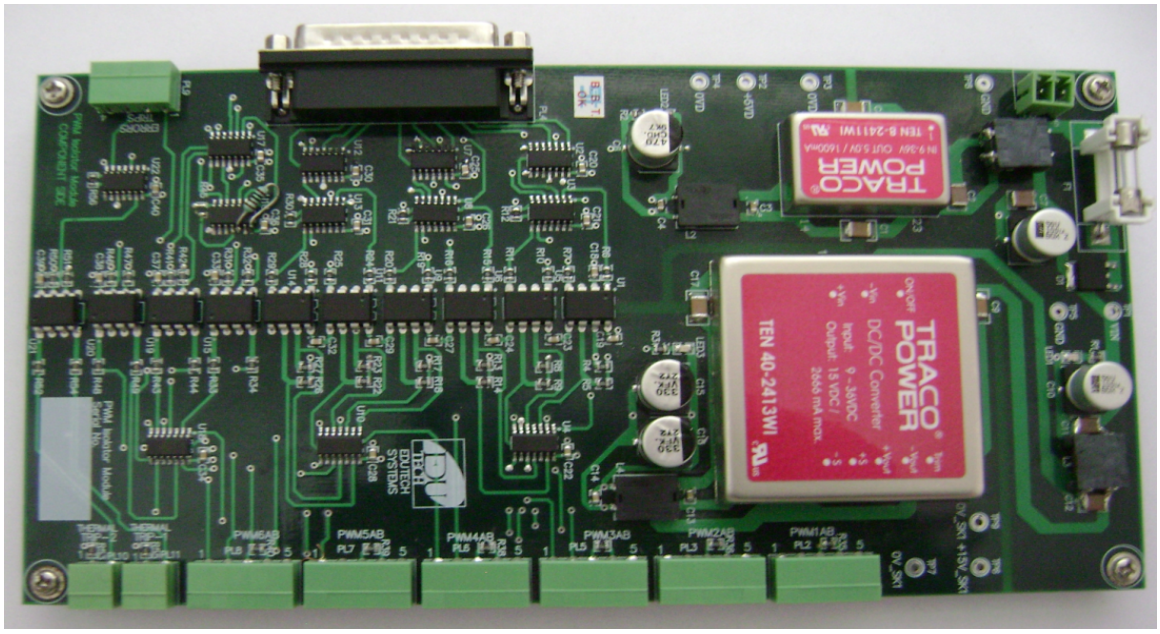

Figure A. 18 PWM Isolator Module

PWM isolation module has 6 PWM units from PWM1 to PWM6.Pin description of PWM1 unit is as follows. Similar pin description is for all other units

Table A. 1 Pin description of PWM

Pin Number	Pin Details
1	PWM1A
2	PWM1B
3	Error
4	15V
5	GND

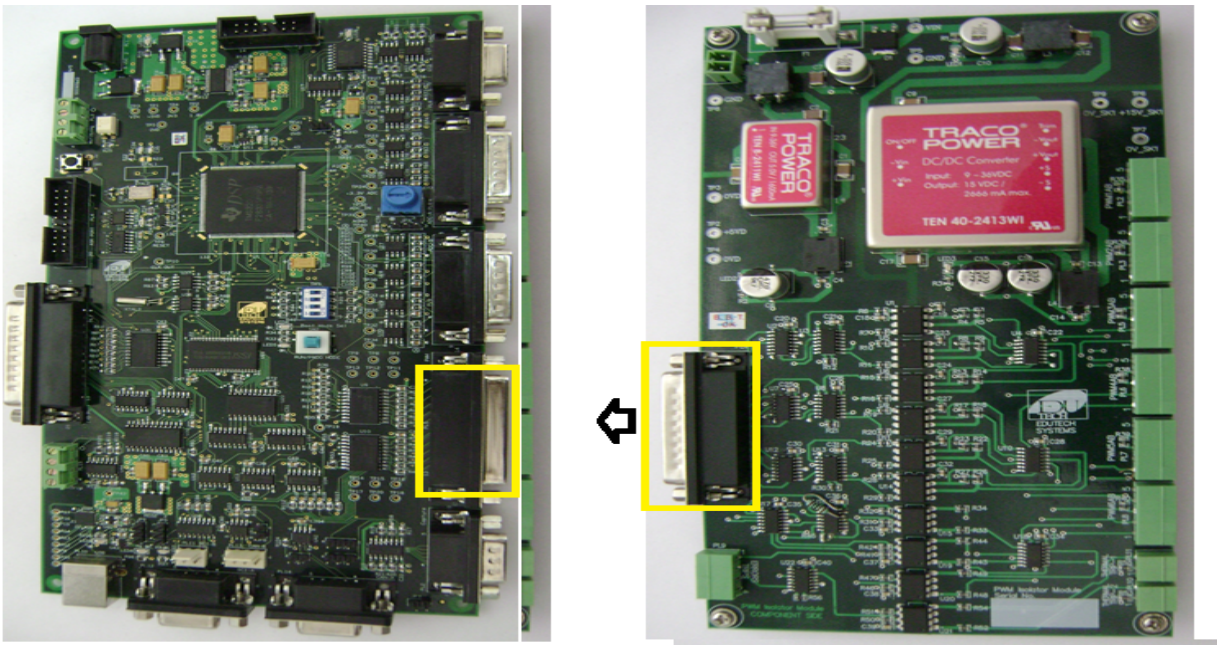

Figure A. 19 Interfacing of DSP module with PWM Isolator module

Voltage source Inverter-Three phase 4 leg IGBT based Inverter specifications are as follows,

Table A. 2 Inverter stack specifications

Stack Type No	MD B6Cl 600/415-10N
Input DC Voltage	600V
Output AC Voltage	415V,Three Phase
Output current	10A (max)
Output Frequency	50Hz
Switching Frequency	10KHz (max)
Cooling Method	Natural cool

It consists of four IGBT module of SKM50GB12V and four GBT drivers SKYPER 32R,one heat sink of MDP3/300mm and two DC link Capacitors of semikron 4700uF with 450V.

A.7 Code Generation Method using CCS & MATLAB –SIMULINK Link

Code Composer Studio is the software based environment used to build the project using C2000 Texas Instrument DSPs. It is the host side Integrated Development Environment .It also integrates all the tools for real time project development. It contains the code generation, simulation and emulation technologies to develop, optimize and debug algorithms for the real time systems. It also allows real time analysis. Data visualization is the unique feature of the code composer studio. This environment is flexible for project development using third party tools. The build process includes steps like edit, compile and link. It has code generation tools like C++ compiler, assembly optimizer and linker. It also includes an editor, debugger, project manager, profiler etc. Code composer studio can test the project algorithm through simulation in simulator mode and also with real time hardware like DSPs. This project development environment provides all the necessary tools for DSP software from beginning to end. It also integrates functionality of all tools in single development environment. This environment provides interfacing of host computer and target platforms through RTDX or JTAG Emulator.

- It consists of a library objects called DSP/BIOS-real time operating systems for DSPs
- Real-time communications channel like RTDX
- Simulator mode allows you to work without hardware
- Resource manager to configure resources
- Integrated code generation tools let you build your code
- Debugger
- Real-time Analysis (RTA) tools enabled by DSP/BIOS
- Allows use of plug-ins from the largest network of third parties in DSP

DSP codes can be generated by Embedded Target and Real-Time Workshop toolbox along with CCS. It is mandatory to configure parameters of target hardware and its I/O device drivers in real time workshop model. MATLAB Link for code composer studio is used to generate DSP assembly language code. This code must be compiled and linked using CCS. After the linking, compiled code loaded and executed on a TI DSPs.

Assembler, Compiler, Linker, Code composer studio, TI C2000 miscellaneous utilities, Code Composer Setup Utility and DSP starter kits are the Texas Instrument software and Hardware tools required for DSP based code generation. Assembler Creates object code (.obj) for C2000 boards from assembly code , Compiler Compiles C code from the blocks in Simulink models into object code (.obj). Linker Combines various input files and libraries , Code Composer Studio Texas Instruments integrated development environment (IDE) that provides code debugging and development tools ,TI C2000 miscellaneous utilities used for developing applications for the C2000 digital signal processor. Installation of above tools may be verified by the c2000lib command in MATLAB.CCS installation may be verified by the command ccsboardinfo in MATLAB.

DSP assembly language code can be generated using MATLAB SIMULINK through CCS by converting SIMULINK model into equivalent C code and then load the C code in DSP using CCS or by establishing MATLAB –CCS link and then convert MATLAB SIMULINK model into DSP code. This link connects code composer studio with MATLAB SIMULINK. This link allows user to debug and verify embedded code running on TI DSPs using MATLAB scripts and simulink models. Code Generation Establish CCS –MATLAB SIMULINK link. It is also possible to develop and implement applications through Processor In loop simulations for prototype testing on all TI DSPs supported hardware modules.

- CCS parameter
- TICCS

It is mandatory to develop or convert SIMULINK model file in discrete time domain and configure the MATLAB SIMULINK model Parameter for DSP Code Generation as shown below

- Open simulink model file, From the **Simulation** menu bar select **Model Configuration Parameters.** The Model Configuration parameters dialog box opens.
- Click the **Solver** tab and enter the following parameter values on the Solver pane.

  Simulation time

  Start Time: 0.0; **Stop Time**: 2.0;

  In Solver options:

  Type: Fixed-step. **Solver:** discrete (no continuous states)

  Fundamental step size: auto

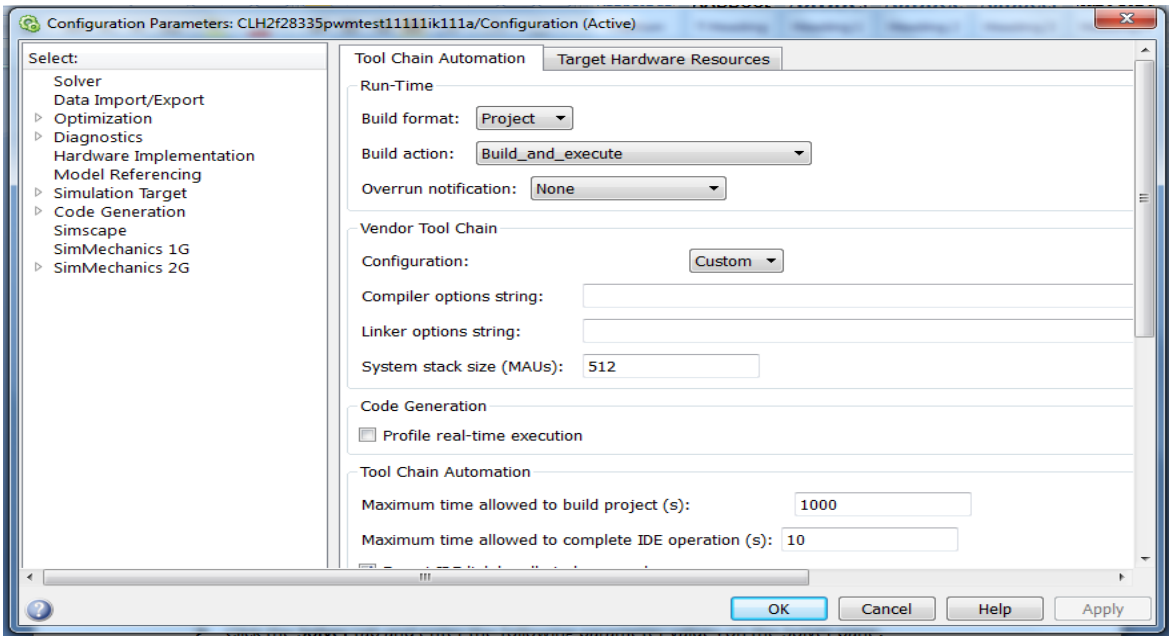

Figure A. 20 Model Configuration parameters

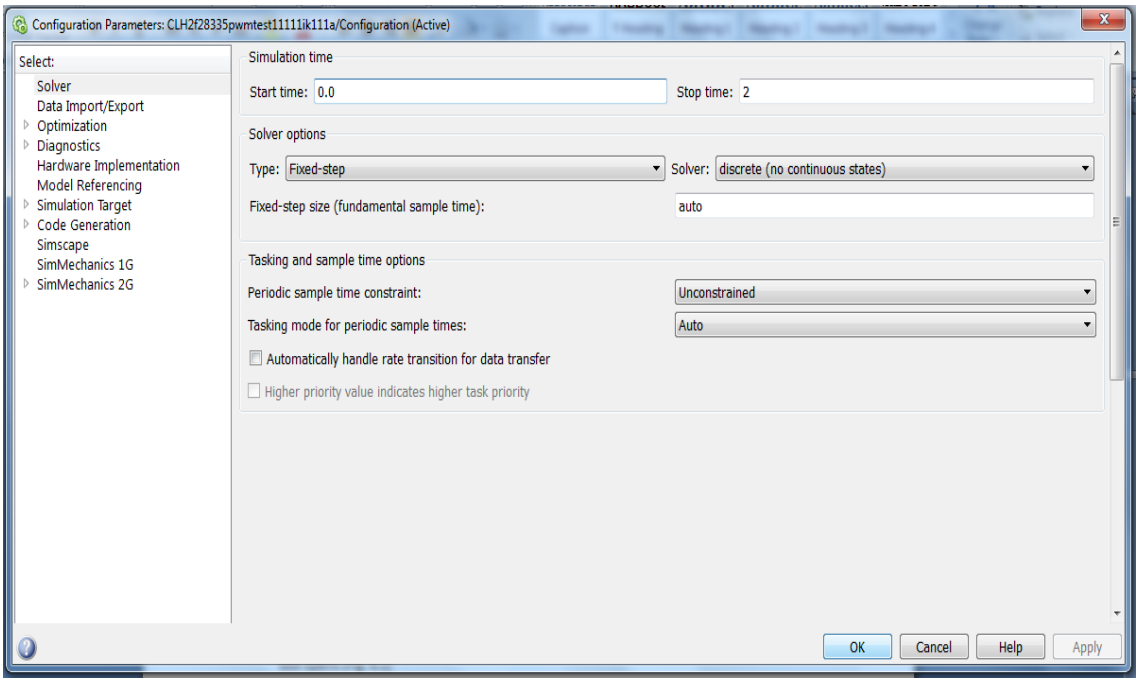

Figure A. 21 Solver parameters

- Click Apply. Then click ok to close the dialog box.

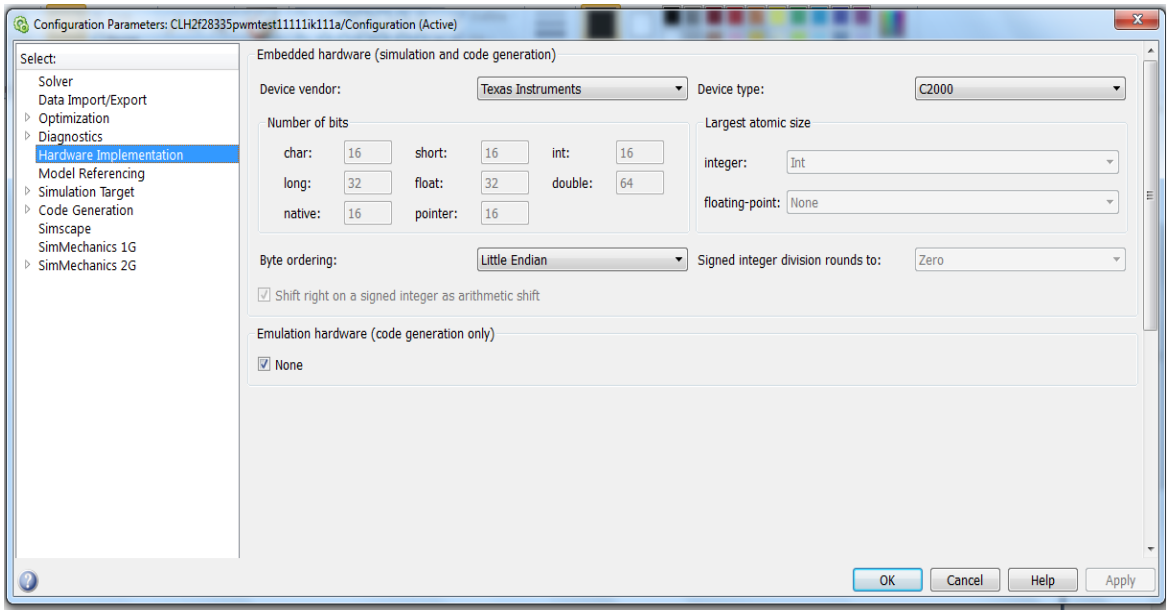

Figure A. 22 Hardware Implementation Tab

- Select Hardware Implementation tab of the Model configuration parameter pane as shown

  Device vendor: Texas Instruments **Device type:** C2000

  Byte ordering: Little Endian

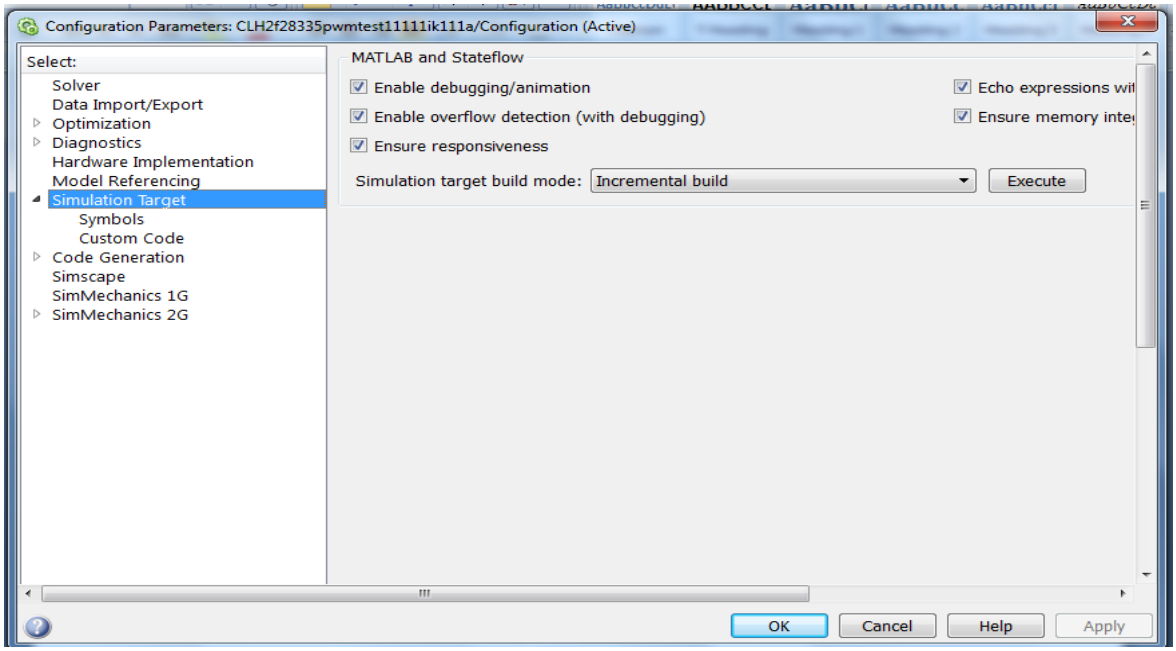

Figure A. 23 Simulation Target

- Select Simulation target tab of the Model configuration parameter pane as shown

  Simulation target build mode: Incremental build

- Select Code Generation tab of the Model configuration parameter pane as shown

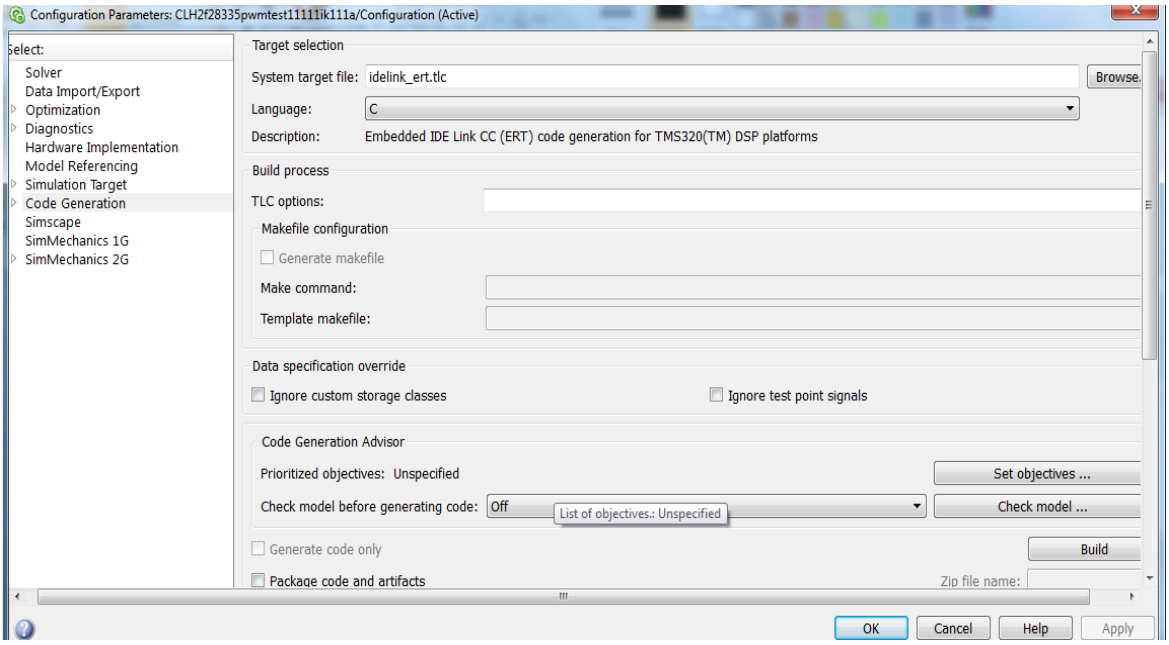

Figure A. 24 Code Generation

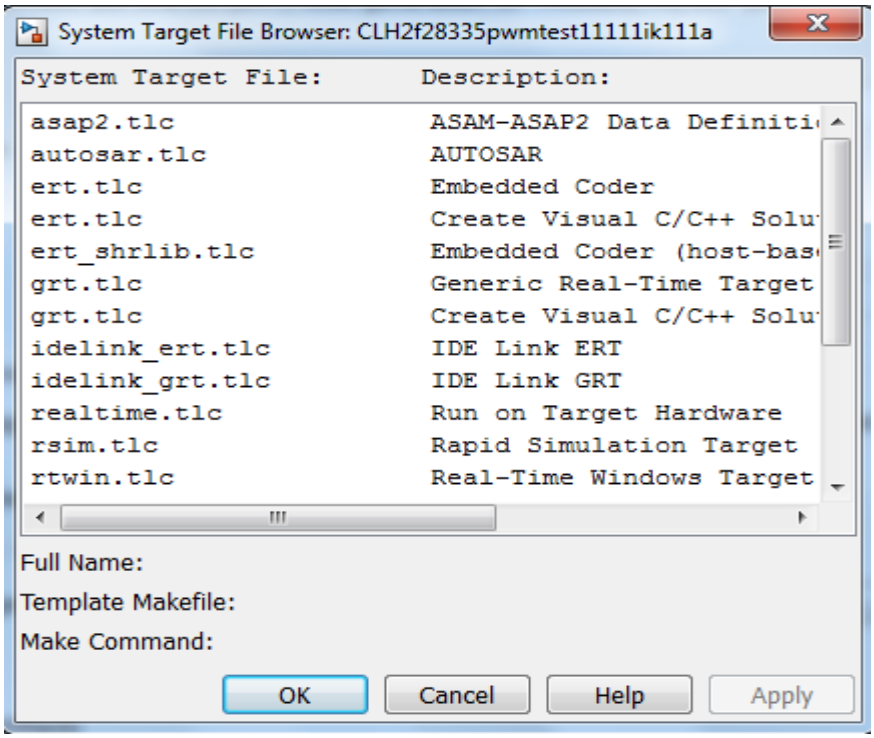

Figure A. 25 System Target File Browser

- Click the **Browse** button next to the **System target file** field. This opens the System Target File Browser, illustrated below. The browser displays a list of all currently available target configurations. When you select a target configuration, chooses the appropriate system target file as idelink_ert.tlc.
- Select Coder target of Code Generation tab and configure the parameter as shown

  In Tool chain Automation tab

  Build Format: Project

  Build Action: Build_and_execute

  In Target Hardware Resources

  IDE / Tool chain: Texas Instruments Code Composer Studio

  Board Properties

  Board: C2000 custom

  Processor: F28335

  CPU Clock: 150MHz

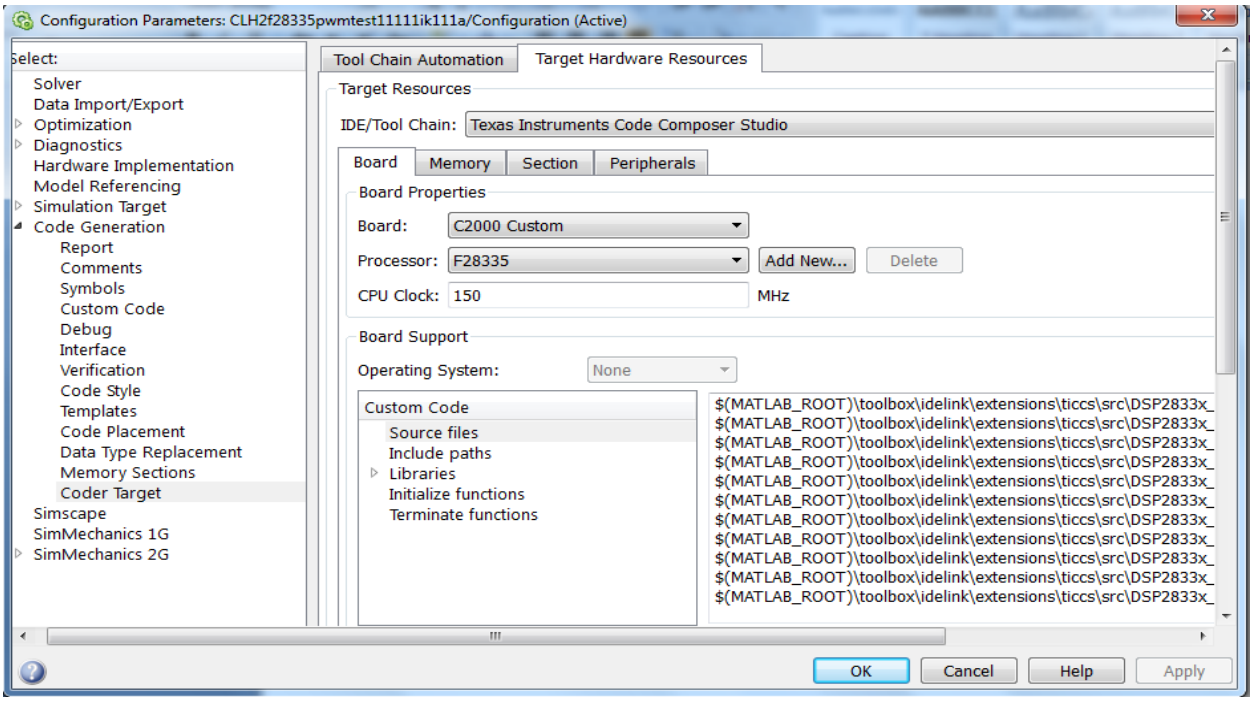

Figure A. 26 Configuration parameters for Target Hardware

- Click Apply. Then click ok to close the dialog box.
- Click the **Build & Run** button to generate code.
- It directly opens the code composer studio.
- It generates the .out file, .asm files.
- In code composer studio Go to File – Load program, select .out file.
- Click Run button in CCS.
- It generates .pjt file for DSP.

A.7 Signal Generation using MATLAB-SIMULINK & EPWM

This example shows the PWM generation using ePWM block of DSP 28335 blocks in Simulink library. Here PWM generated by comparing sawtooth wave form with the constant signal. In ePWM counter mode is selected as "Up" counting mode.

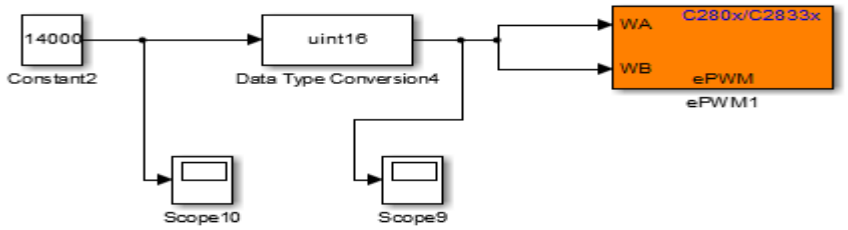

Figure A. 27 MATLAB –Simulink based PWM Generation

Timer period value for ePWM block obtained from following equation

$$TxPR = \frac{CPU\ clock\ Frequency}{desired\ frequency} \qquad (A.2)$$

10 KHz switching frequency and DSP 320F28335 with clock frequency of 150MHz,

$$TxPR = \frac{150MHz}{10Khz} = 15000 \qquad (A.3)$$

Case-1 PWM generation for 6.9 % duty cycle

If we take compare value is equal to 14000 for $TxPR = 15000$ then duty cycle obtained as 6.9%, which is clearly visible in Fig.-A.28

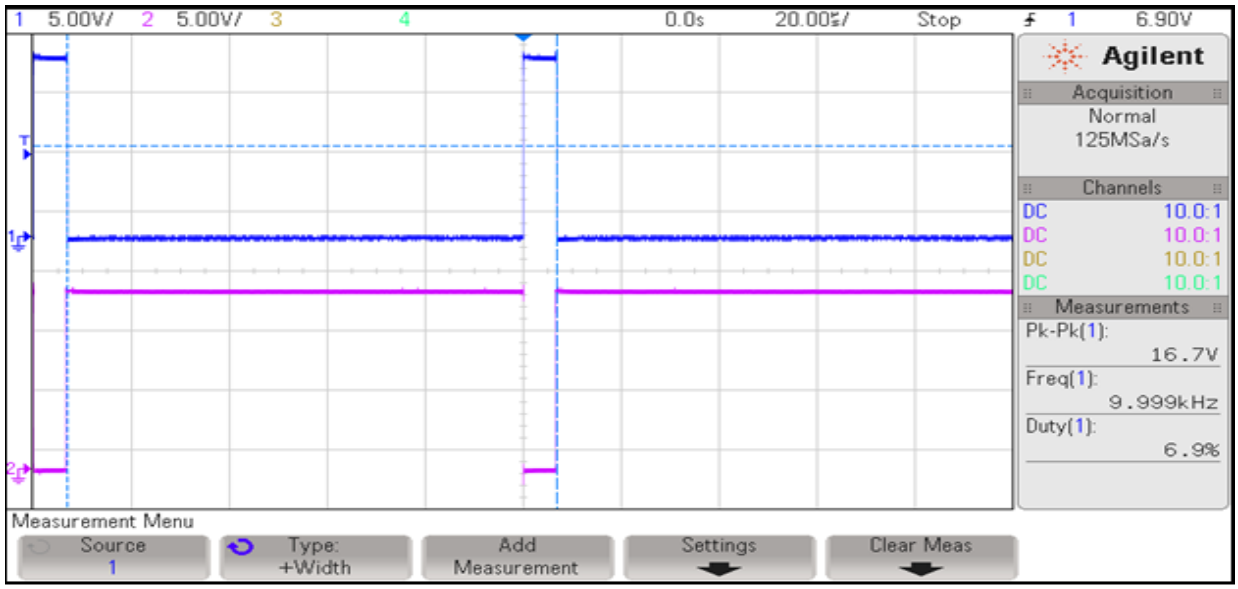

Figure A. 28 PWM for 6.9% duty cycle

Case-2 PWM generation for 25.2 % duty cycle

If we take compare value is equal to 11250 for $TxPR = 15000$ then duty cycle obtained as 25.2 %, which is clearly visible in Fig.-A.29.

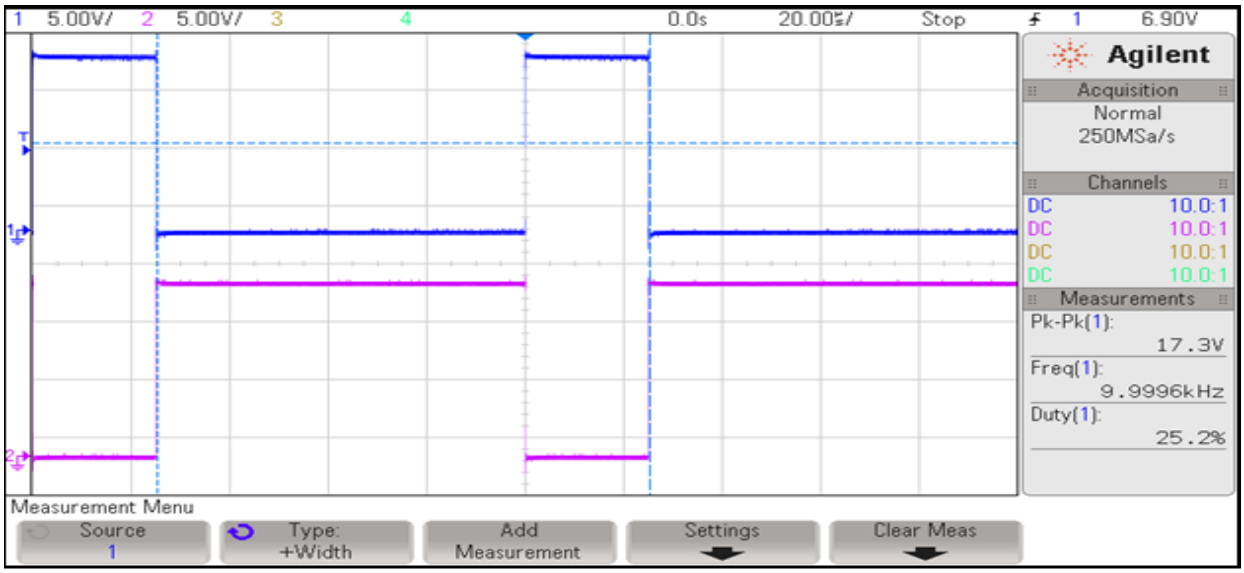

Figure A. 29 PWM for 25.2% duty cycle

Case-3 PWM generation for 50% duty cycle

If we take compare value is equal to 7500 for $TxPR = 15000$ then duty cycle obtained as 50.2 %, which is clearly visible in Fig.-A.30.

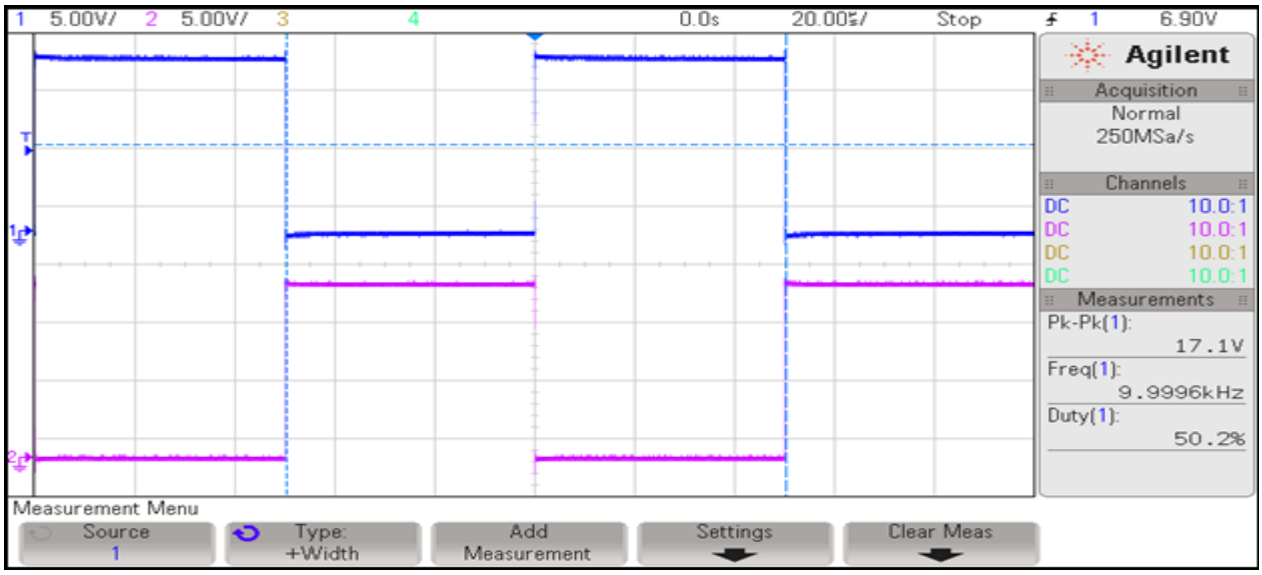

Figure A. 30 PWM for 50.2% duty cycle

Case-4 PWM generation for 75.2% duty cycle

If we take compare value is equal to 3750 for $TxPR = 15000$ then duty cycle obtained as 75.2 %, which is clearly visible in Fig.-A.31.

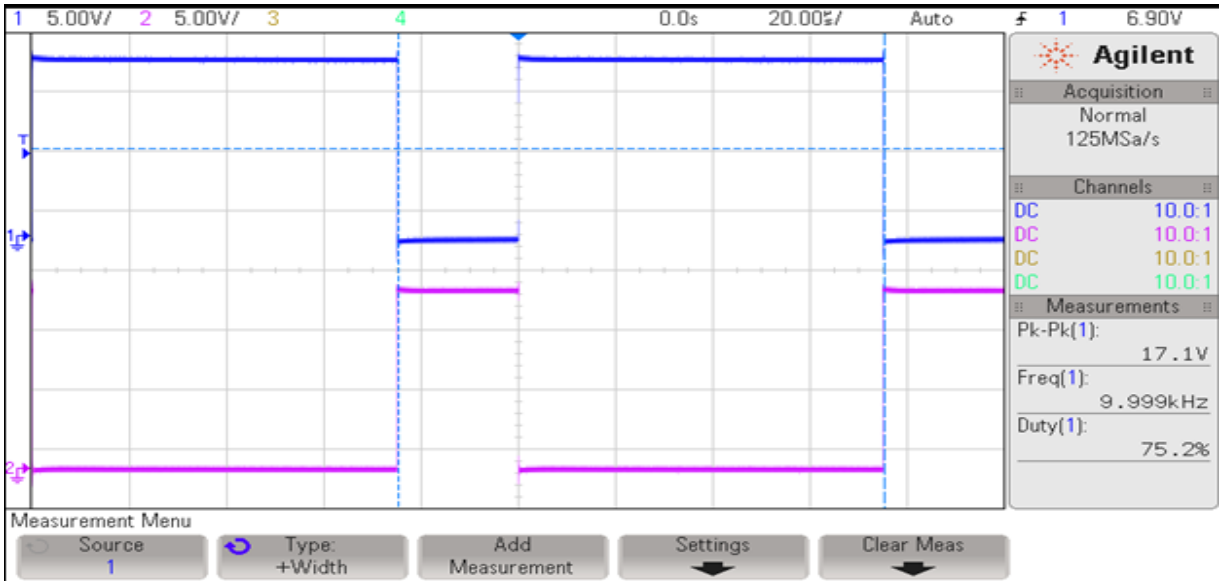

Figure A. 31 PWM for 75.2% duty cycle

Case-5 PWM generation for 93.6% duty cycle.

If we take compare value is equal to 1000 for $TxPR = 15000$ then duty cycle obtained as 93.6 %, which is clearly visible in Fig.-A.32.

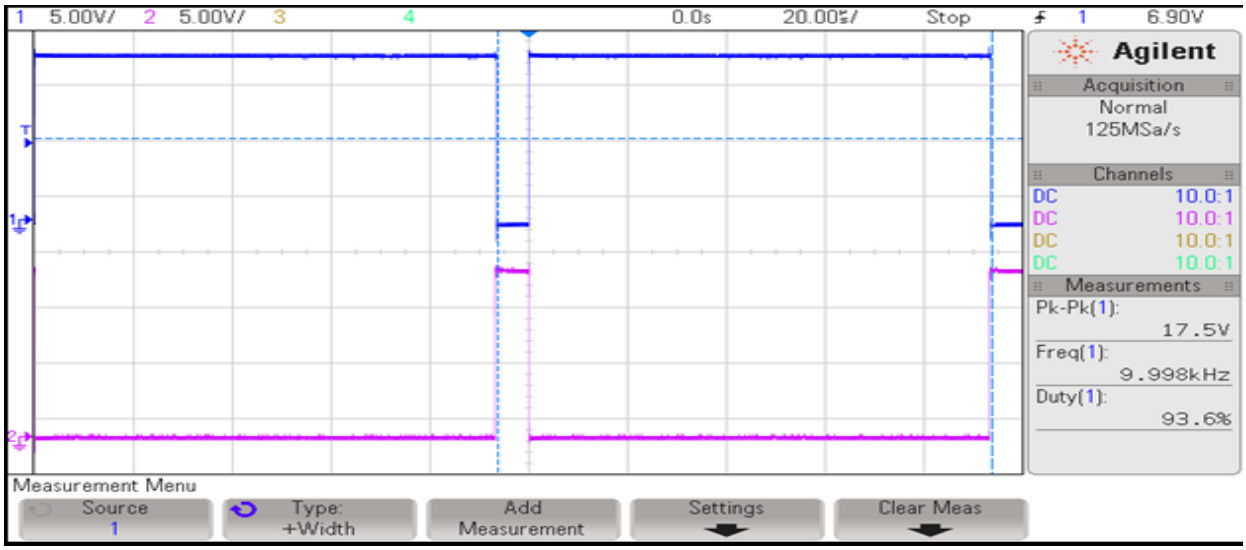

Figure A. 32 PWM for 93.6% duty cycle

A.8 Summary

This chapter presents different software tools used for controller design of motors and drives. It also describes features of real time workshop tool, Fuzzy logic tool box and GA tool box of MATLAB for the design and implementation of different applications. This chapter also presents in detail ePWM block of embedded coder and its parameter configuration for real time code generation. This chapter also gives details specifications of all the hardware components utilize for the experimental work. It also shows the detail method for real time code generation using MATLAB-Simulink real time workshop tools. It also discusses the parameter configuration for real time code generation from the simulink model. Pulse width Modulated signal generation of different duty cycle is also presented with real time implementation results captured using high resolution DSO.