

Design Document

Software Development Guide for Modular Inverter

Reference: TU0003

Issue: 003

Status: Issued V1.0

Author: Guoliang Zhang

Updated by: Ali Bazzi / Paul Rancuret

Principal Investigator: J. Kimball

Created: April 2, 2005

Updated: November 23, 2010

Abstract:

This document is intended to archive the modular inverter control software architecture, which is based on Matlab/Simulink Embedded Coder for TI C2000 toolbox. Some customized functions are described in detail.

This document also serves as instruction manual for power lab users that may use the modular inverter to develop digital motor control algorithms or other related applications. A fully functional example is included at the end of this document and links to the related files are provided.

Some useful resources are listed in the appendix to help users locate necessary knowledge base that are required to fulfill advanced functionalities.

Document Revision History

Issue	Date	Comments
001	2/17/2006	First release
002	3/3/2006	Added Flash part, updated TOC
003	11/23/2010	Added Recommendations for Advanced Applications

Contents

1.	Introduction	3
1.1	Overview and Scope	3
1.2	Definitions.....	5
1.3	References	5
2.	User Interface	6
2.1	Matlab/Simulink Environment Setup	6
2.2	Developing Control Model in Simulink	9
2.2.1	Fixed Point Datatype.....	9
2.2.2	IQ Math Library and DMC Library	11
2.2.3	Simulation	12
2.2.4	Integrate I/Os and core functions of the DSP into the control	12
2.3	Generate Code and Run It on DSP	13
2.4	Errors, Faults and Exceptions	13
2.4.1	Board name mismatch	13
2.4.2	Could not find target board	13
2.4.3	Model contains ADC blocks won't work under internal memory mode.....	14
2.4.4	Over-run detected	14
3.	A Step-by-Step Example.....	15
4.	Create Flash Based Project	16

1. Introduction

1.1 Overview and Scope

EZdsp2812 is a board made by Spectrumdigital.com that utilizes a TI TS320F2812 DSP core. Full description on this chip can be found at TI.com. Some of the key features of this DSP controller are:

- **Extended Temperature Performance of -55°C to 125°C**
- **High-Performance Static CMOS Technology**
 - 150 MHz (6.67-ns Cycle Time)
 - Low-Power (1.8-V Core @135 MHz, 1.9-V Core @150 MHz, 3.3-V I/O) Design
- **JTAG Boundary Scan Support**
- **High-Performance 32-Bit CPU (320C28x)**
 - 16 x 16 and 32 x 32 MAC Operations
 - 16 x 16 Dual MAC
 - Harvard Bus Architecture
 - Atomic Operations
 - Fast Interrupt Response and Processing
 - Unified Memory Programming Model
 - 4M Linear Program/Data Address Reach
 - Code-Efficient (in C/C++ and Assembly)
 - 320F24x/LF240x Processor Source Code Compatible
- **On-Chip Memory**
 - Flash Devices: Up to 128K x 16 Flash (Four 8K x 16 and Six 16K x 16 Sectors)
 - ROM Devices: Up to 128K x 16 ROM
 - 1K x 16 OTP ROM
 - L0 and L1: 2 Blocks of 4K x 16 Each Single-Access RAM (SARAM)
 - H0: 1 Block of 8K x 16 SARAM
 - M0 and M1: 2 Blocks of 1K x 16 Each SARAM
- **External Interface (2812)**
 - Up to 1M Total Memory
 - Programmable Wait States
 - Programmable Read/Write Strobe Timing
 - Three Individual Chip Selects
- **Three 32-Bit CPU-Timers**
- **128-Bit Security Key/Lock**
 - Protects Flash/ROM/OTP and L0/L1 SARAM
 - Prevents Firmware Reverse Engineering
- **Motor Control Peripherals**
 - Two Event Managers (EVA, EVB)
 - Compatible to 240xA Devices
- **Serial Port Peripherals**
 - Serial Peripheral Interface (SPI)
 - Two Serial Communications Interfaces (SCIs), Standard UART
 - Enhanced Controller Area Network (eCAN)
 - Multichannel Buffered Serial Port (McBSP)
- **12-Bit ADC, 16 Channels**
 - 2 x 8 Channel Input Multiplexer
 - Two Sample-and-Hold
 - Single/Simultaneous Conversions
 - Fast Conversion Rate: 80 ns/12.5 MSPS
- **Up to 56 General Purpose I/O (GPIO) Pins**
- **Development Tools Include**
 - ANSI C/C++ Compiler/Assembler/Linker
 - Code Composer Studio IDE
 - DSP/BIOS
- **Boot ROM (4K x 16)**
 - With Software Boot Modes
 - Standard Math Tables

Code Composer Studio (CCS) for C2000 is the software development tool for this DSP. Current version is V2.21. It is recommended to have only one CCS installed on one computer to avoid any conflict between applications.

Matlab/Simulink provides the automated code generating tool for this DSP family. Users can communicate with the DSP via Matlab Link for CCS. Under most cases, users should develop their control algorithms in Simulink and use the automatic RTW tool to generate/build/download the code to target DSP and run the executable program in CCS.

To use the digital controller system effectively and efficiently, certain skills/knowledge are required. Based on the complexity that might get involved during the development, two levels of users are defined:

Level 1: Control Algorithm developers

Example: motor control algorithm, digital pwm algorithm, filter designs, etc.

You need to have knowledge on

- basic usage of Matlab/Simulink
- fixed point data type, IQ math(just another form of fixed-point defined by TI)
- discrete time signal processing -specifically, discrete time integration, sample/hold, delay, Z transform.
- PWM parameter settings
- Hardware configuration on the modular inverter and its control board.
- Matlab GUI is a plus
- Real-Time multitask scheduling is a plus.

Level 2: Expanded function developers

Example: DAC, Communication, Stand alone applications, etc. Especially when your needs are beyond the capability provided by our library functions.

Except requirements to level 1 users, you also need knowledge on

- C/C++ and Library functions for C281x provided by TI.
- S-Function in Matlab/Simulink
- RTW(Real-Time Workshop) and TLC(Target Language Compiler)
- Hardware on TI 320F2812 DSP is a plus.

This document focuses on level 1 user. References will be provided to help level 2 users locate the necessary recourses.

A knowledge tree below shows the required skills/knowledge and their relationships for each group of users. Bold text indicates key skill needed for that group of users. Real Time Data Transfer function is only available with the update of CCS 2.20+ and Matlab R14 SP2+.

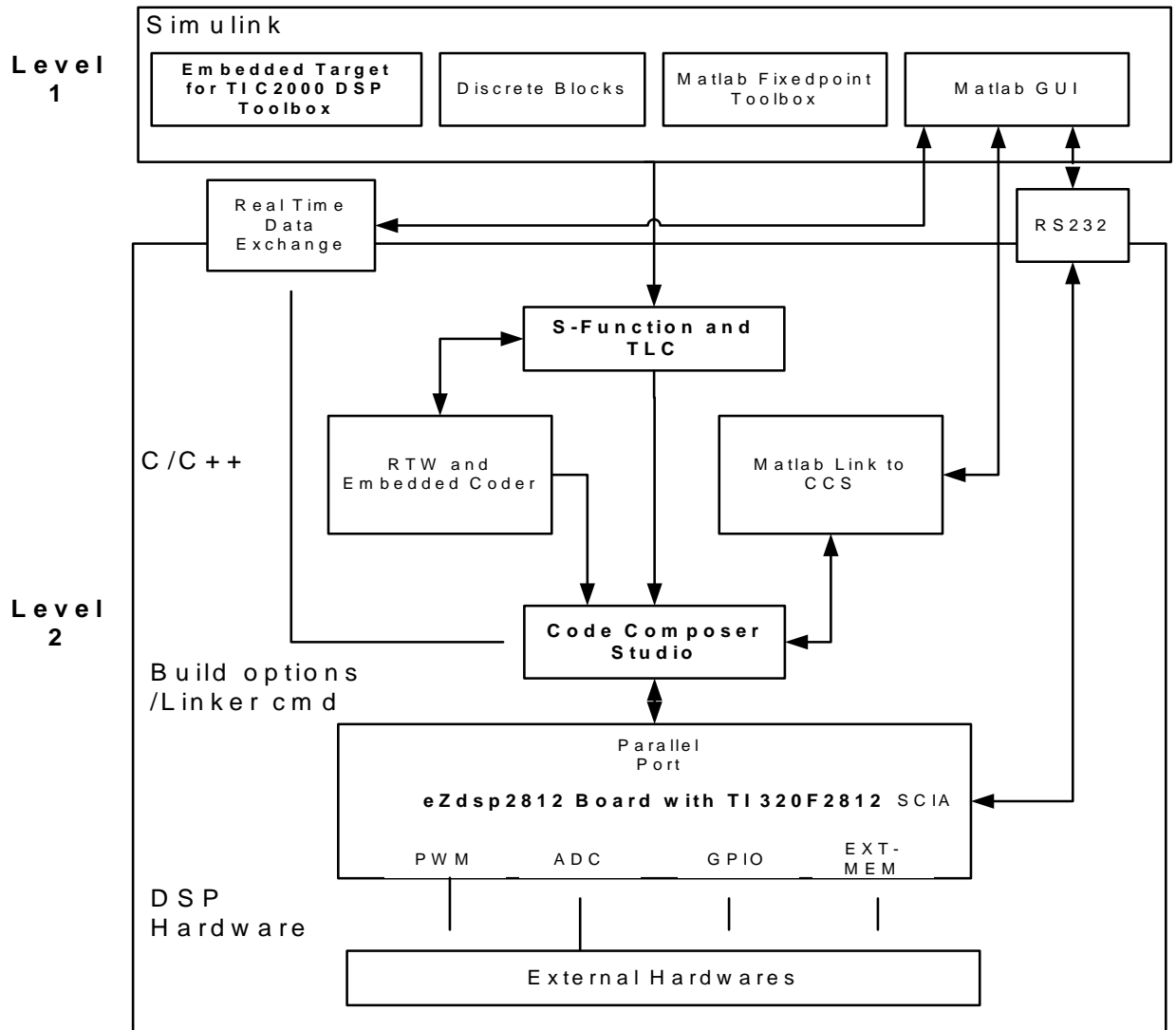


Figure 1 Knowledge Tree

1.2 Definitions

TI	Texas Instrument
DSP	Digital Signal Processing
CCS for C2000	Code Composer Studio, a software development tool for F281x provided by TI
UUT	Unit Under Test

1.3 References

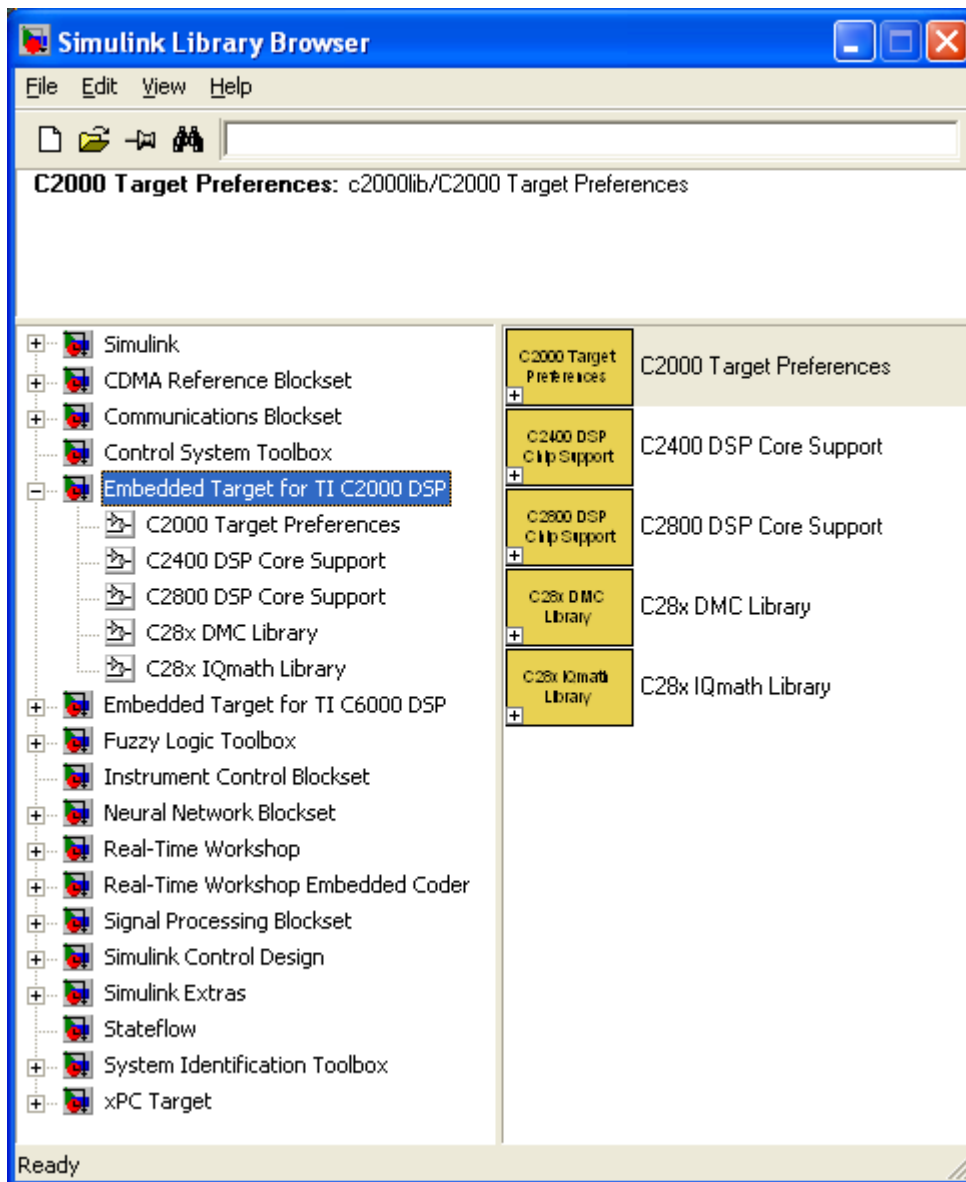
F2812 family Home --- <http://focus.ti.com/docs/prod/folders/print/sm320f2812.html>

2. User Interface

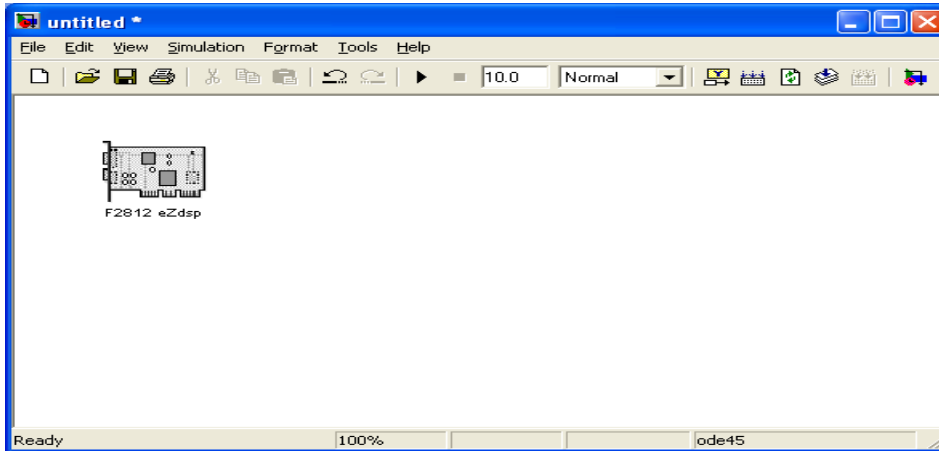
2.1 Matlab/Simulink Environment Setup

User will develop control algorithm under Simulink environment. To successfully run the process, several toolboxes are required. They are: Fixed Point Toolbox, RTW toolbox, Embedded Coder toolbox and Embedded Coder for TI C2000 Toolbox. Type “ver” at Matlab command window to see if all of those toolboxes are installed on the computer you plan to work on. We have one license for each toolbox for the whole group. Also, a Microsoft VC6.0 should be installed on the computer.

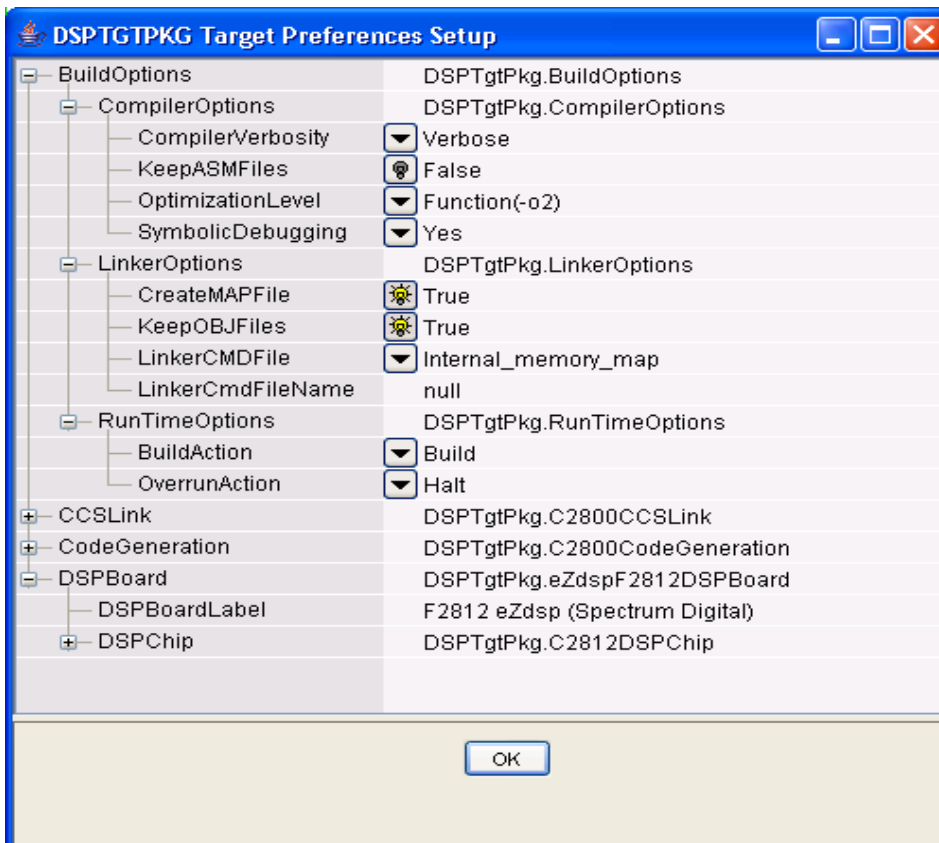
Upon opening the Simulink Library browser, an “Embedded Target for TI C2000 DSP” item should show up on the left, all the build-in functions can be found under this group of library modules.



The first thing to do after creating a blank model under Simulink is to choose the F2812 eZdsp target from “C2000 Target Preference” library. Now there should be a blank Simulink model window with target preference block, it should look like this:

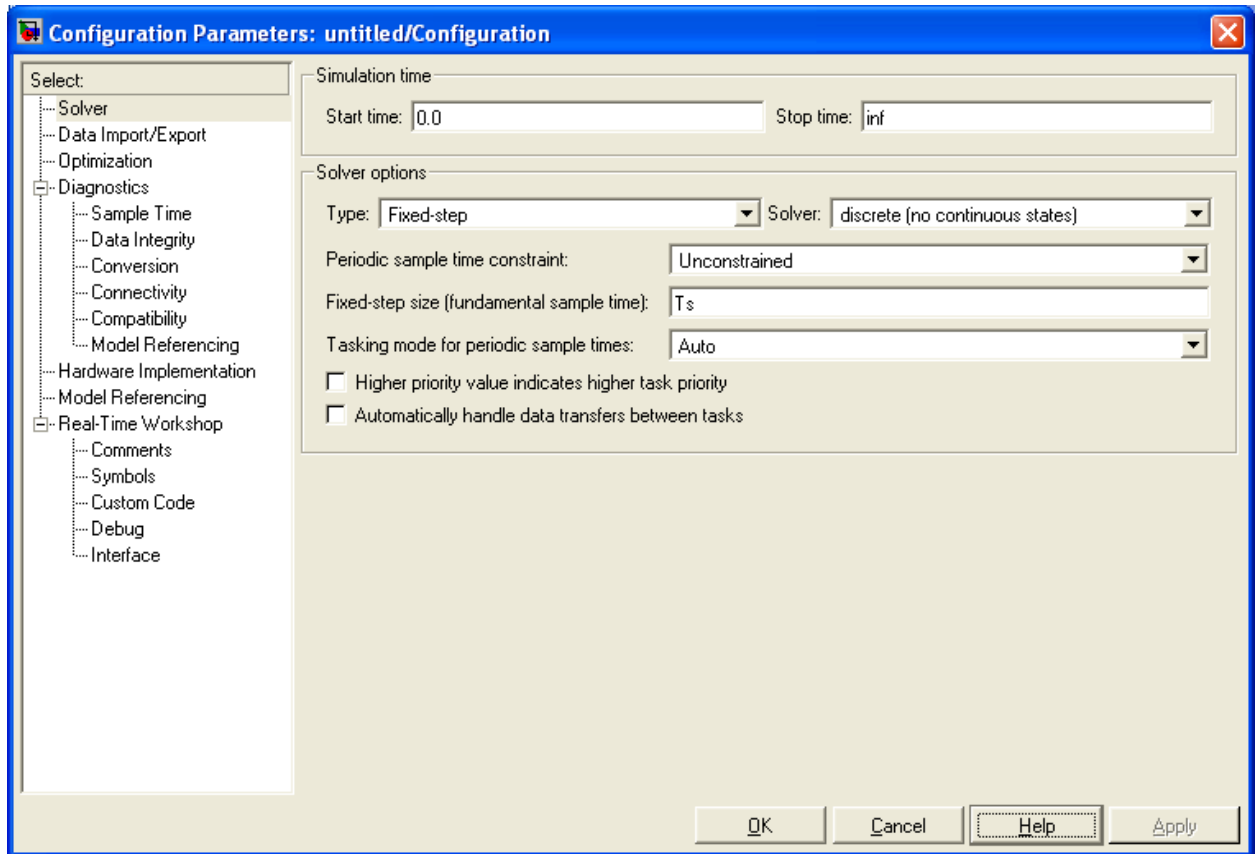


This block contains options for target board hardware configuration and compiler/linker preferences. You need to change some of the options to make it like this:



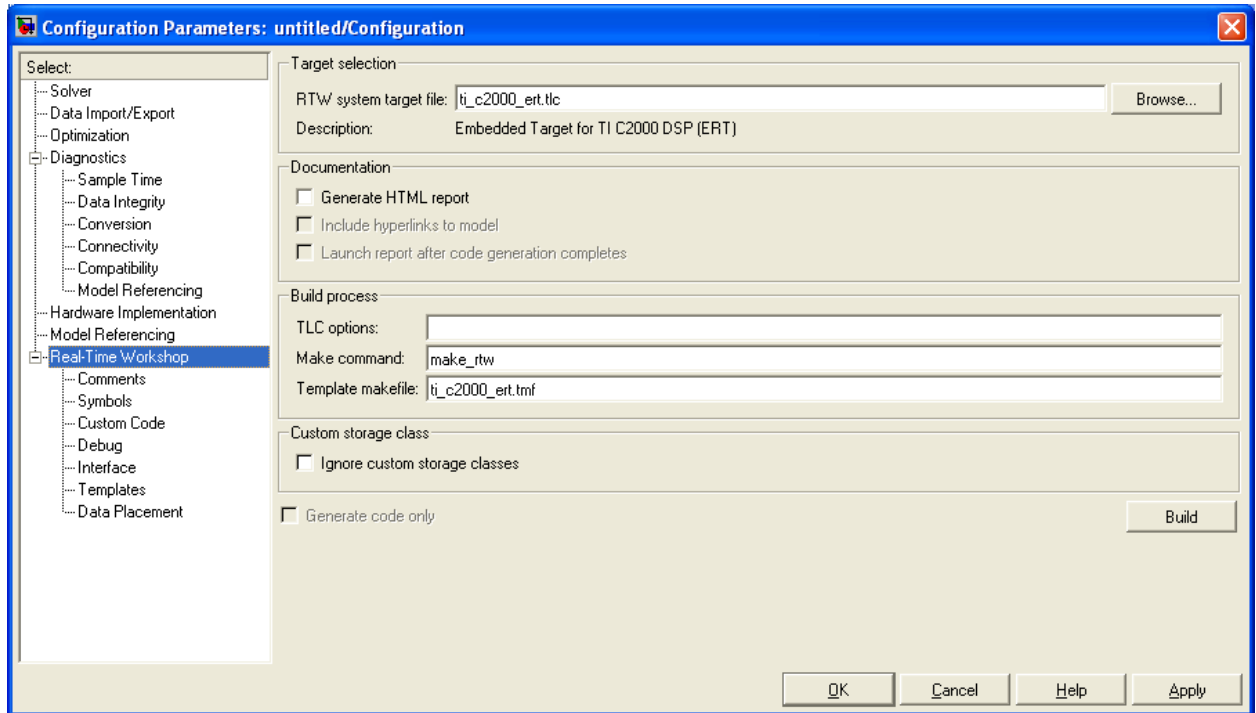
You might need to change the DSPBoardLabel option according to CC_SETUP definition. See 2.4.1 for more information. Other un-shown items should be kept as default.

Next Step is to configure the model parameters. Since the embedded software is running periodically forever, we need the model to be implemented in discrete time with no continuous state. Stop time should be inf under most cases. The solver setup page should be like:



Notice that the fundamental sample time must be fastest sample time in the model and all other blocks can only be run at integer times this rate. Faster tasks (blocks) have higher priority.

Another page need to be configured is the Real-Time Workshop preference. It will look like this after you change the RTW system target file name to "ti_c2000_ert.tlc" from the Browse...window.



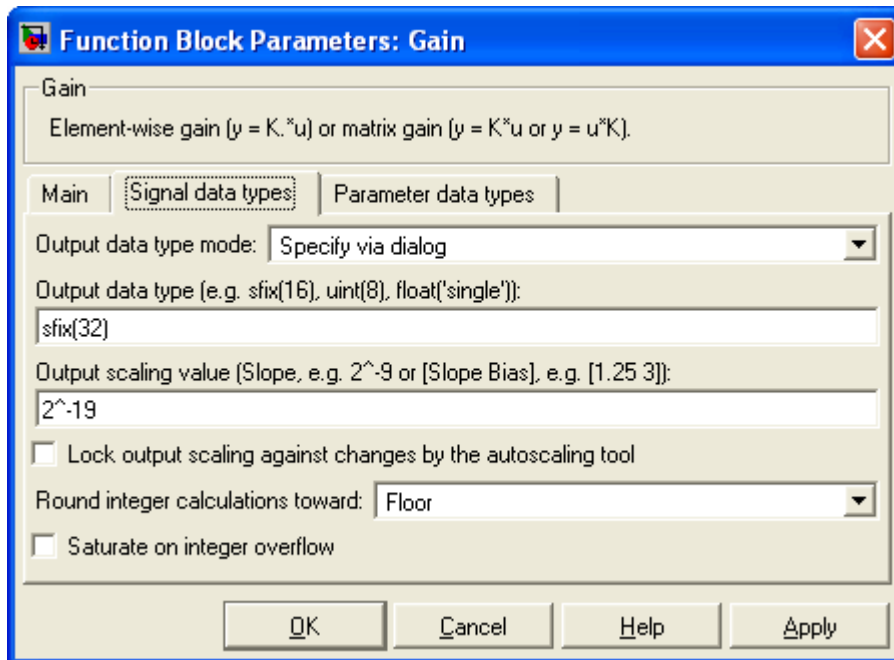
This finishes the configuration of the model. It is recommended to set the sampling color and data type of each port display options on from the simulink model window toolbar \Format\Port and Signal Displays option.

2.2 Developing Control Model in Simulink

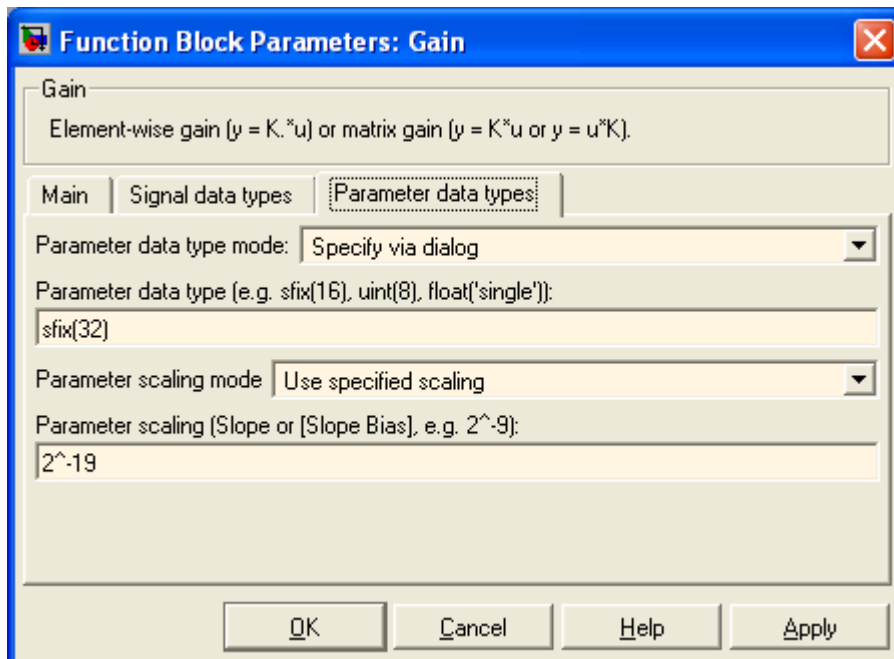
In this part, we describe some key concept to build a control model.

2.2.1 Fixed Point Datatype

Assume you are familiar with the fixed point data concept. In Simulink, to assign certain datatype to a specific block is done by changing signal/parameter settings in the block property tab. For example, we want to use a 32bit fixed point gain of 3.14159, and want 1/100000 resolution on the output within +/-2000. Then we need to setup the signal data type and parameter data type as:



And



32^{-19} gives a resolution about $2e-6$ for the fractional part of a 32bit number. And integer part gives us the range of 0 to +/- 4096, which would satisfy our requirements.

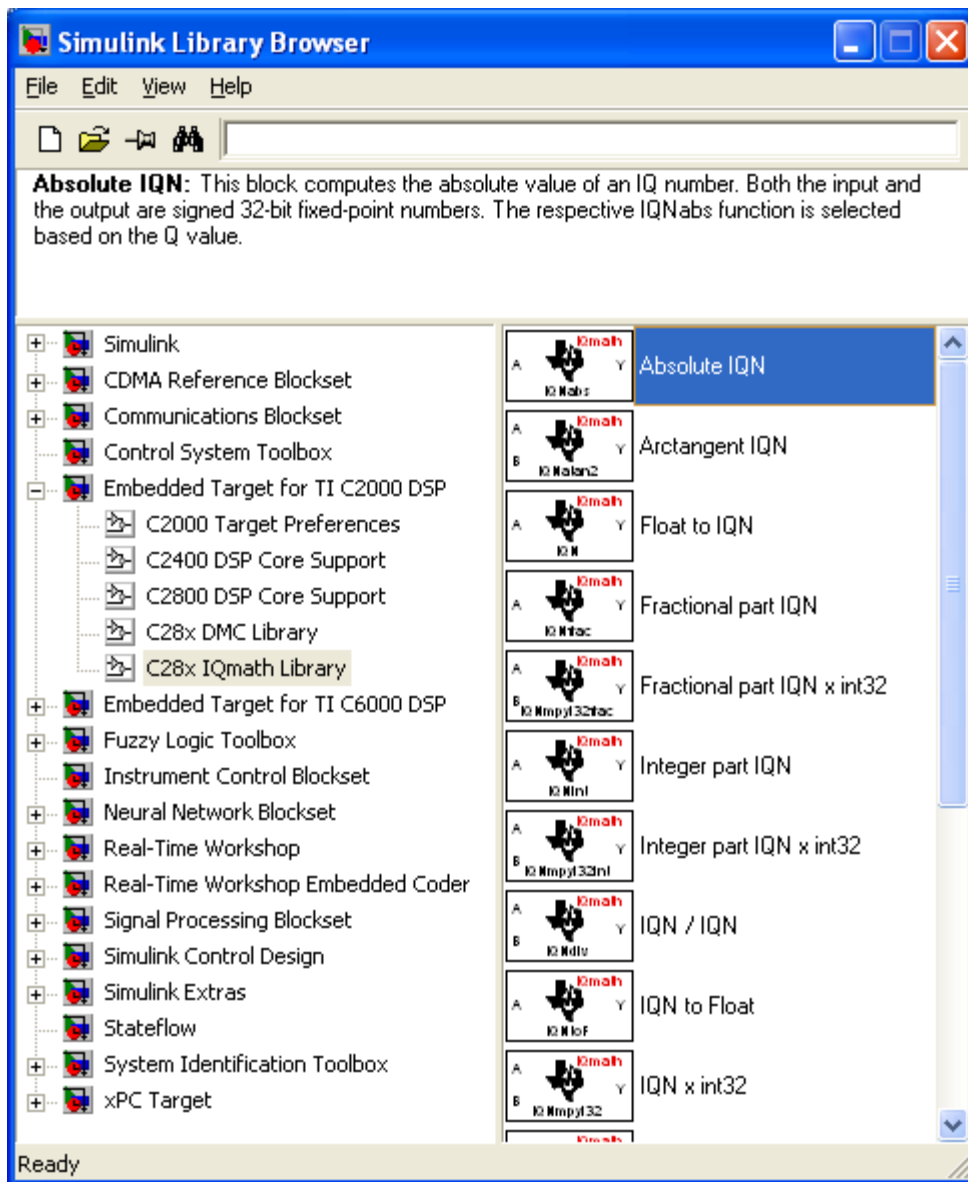
Important: it is crucial to have all signals that may flow in the model/control be falling into the data range you have defined. It is especially true when dealing with **discrete integrators** because the stepsize in our model are usually very small. Rule of thumb: use 32^{-16} for all arithmetic operations, use 32^{-26} for critical integrator internal signals. Notice that 32^{-26} have the integer part range from -32 to 32 only, it's important not to have it overflow.

2.2.2 IQ Math Library and DMC Library

Under Simulink Library Browser, you can find C28x IQmath Library and DMC Library under Embedded Target for TI C2000 DSP toolbox. These are functions that would generate much more efficient code than using common Matlab blocks.

When choosing arithmetic operation blocks, you should always use the one from IQmath library instead the ones from Simulink Block sets. For example, to implement a gain, instead of using the gain from Simulink\Math Operations\Gain, you should use a constant with proper data type and an IQN x IQN or other multiplication functions under IQmath Library. In the DMC Library, there are transforms that have already been implemented in an integrated block format.

Some other functions like filters, discrete integrators should be implemented by blocks from Simulink\Discrete library.



2.2.3 Simulation

After a model is built, the next step is to simulate the system to see if the control is functional as expected and we might need to make necessary changes to the parameters or control structure.

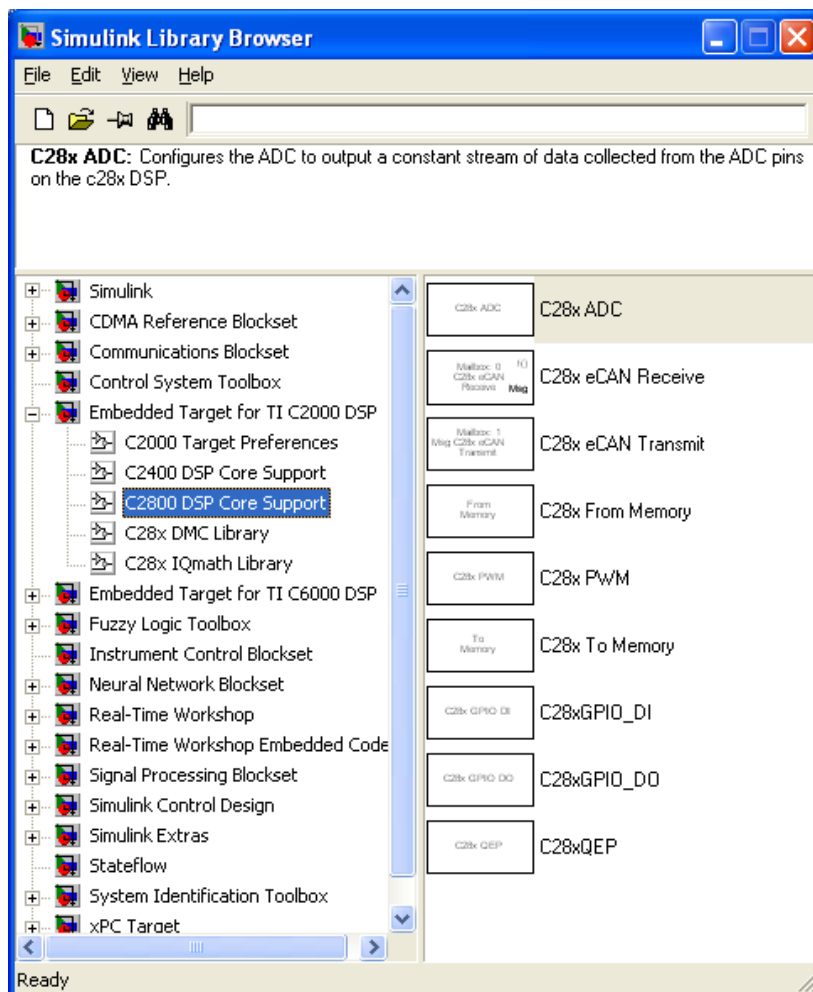
The simulation is no different than an ordinary Simulink model simulation. You can use the scopes and display blocks. You should be able to use any analysis tool provided in Simulink environment. When simulating controller with continuous plant model, choose proper solver in the simulation configuration page.

2.2.4 Integrate I/Os and core functions of the DSP into the control

It is recommended that the user to keep two copies of the model: One for simulation purpose and one for actual code generation purpose. The unit under test (UNT) should be kept identical in the two models.

In the code generation model, we replace the dummy/test inputs from simulation model with interface blocks that are supported by this DSP. We also remove the non-codable blocks which includes all continuous blocks and display blocks for simulation purpose only.

The I/O blocks are under C2800 DSP core support tab in the library browser.



Complete descriptions on these blocks can be found in the help files.

2.3 Generate Code and Run It on DSP

If all the software/toolboxes are installed properly and board configuration is correct, by clicking on the “build” button on top of the Simulink window, an automated process would be invoked and the code should have been built in CCS. Then an active CCS project should appear on the left side project manager window. In theory, the processor needs to be reset before loading the code into the target DSP. In practice, it might not be necessary. To load and run the program, select File/Load Program... from CCS main window, the *modelname.out* executable file usually appears by default. Otherwise, you should be able to locate it manually under *modelname_c2000_rtw/* directory. After the code is loaded in the DSP successfully, press F5 or choose “run” button to run the program on the target. If the target stops immediately after a run command, you might need to do File/Reload Program... and run it again. The code generation may take around 1-2 minutes depending on the complexity of the model and the computer you are using.

Explanations on some terms in this stage:

Generate Code: Matlab Target Language Compiler (TLC) to compile the Simulink model into C code and other resources needed by later process.

Create CCS Project: Matlab TLC creates a project file and include all necessary resources into the *modelname_c2000_rtw* directory.

Build: CCS C compiler is invoked to compile the project source files into .obj files.

Link: CCS Linker links all .obj files, library files from TI library into one executable *modelname.out* file.

Load: CCS loader load the *modelname.out* file to the RAM on DSP according to the linker command file *modelname.cmd*, which was generated by Matlab during “Generate Code” stage.

2.4 Errors, Faults and Exceptions

Some common errors, faults, exceptions and their possible causes are presented in this section.

2.4.1 Board name mismatch

Symptom: After clicking the “Build” button, error message saying board name not matching the label defined in Target Preference Block.

Cause: Board name defined in target preference block is different than that from CC_setup.

Solution: Change the target preference board information to whatever indicated by error message.

2.4.2 Could not find target board

Symptom: After clicking the “Build” button, error message saying cannot find board.

Cause: May caused by power loss on the target when Matlab is still running.

Solution: Make sure the power is connected to the board then reset the board from CCS Debug/Reset Emulator. Then go to Matlab command window and type >> clear CCS_Obj.

2.4.3 Model contains ADC blocks won't work under internal memory mode

Symptom: Loading error in CCS. Or the program is not running as expected if ADC is used and internal memory mode is selected in target preference block.

Cause: Bug in Matlab when generating the *model.cmd* file. A DSECT is declared to a section that would be used in ADC initialization. A DSECT-ed code section will not be loaded to the DSP RAM by the loader.

Solution: 2 ways, either one will work:

1, go to `C:\MATLABroot\toolbox\rtw\targets\tic2000\tic2000\src` and open `DSP281x_usDelay.asm` file with any text editor, comment out the line of `.sect "ramfuncs" with ";"`.

2: go to `C:\MATLABroot\toolbox\rtw\targets\tic2000\tic2000` and open `ti_c2800_cmd.tlc` with any text editor, find the line of:
`ramfuncs : > PRAMH0, PAGE = 0, TYPE = DSECT`

Cross out the `“, TYPE = DSECT”` part of this line. So it looks like:
`ramfuncs : > PRAMH0, PAGE = 0`

2.4.4 Over-run detected

Symptom: After several tries of clicking the “run” button, the program goes to “Halt” position repetitively.

Cause: Over-run is detected, CPU overloaded.

Solution: Increase the stepsize in the model and recompile. Make sure that there are no unnecessary double precision or non-lookup-table type trigonometric functions in the model.

3. Recommendations for Advanced Applications

3.1 ADC and RTDX Issues

The number of ADC and RTDX channels in a Simulink block diagram affects that maximum allowable sampling rate (T_s) that can be used on the DSP. The ADC has two main ports, A and B, each having eight channels. While the maximum allowable rates are given in the eZdspF2812 datasheet, these do not reflect actual implementations in advanced applications that require significant processing. It is advised that the minimum number of ADC channel is used to achieve higher sampling rates. The reason this is true is that every ADC port, i.e., A or B, has a multiplexed structure. This means that the multiplexer needs enough time to fetch ADC data for the DSP to process it appropriately.

A similar problem exists with RTDX channels, and sampling rates might be even slower in this case. RTDX channels communicate in real time between the DSP registers and MATLAB, thus significant lag exists between on these hardware-software interface channels. Higher sampling rates can be achieved with less RTDX channels.

When the sampling rate is high compared to what the DSP can handle for a given number of ADC and RTDX channels, a block diagram might still compile without errors or warnings. But, when it is run using Code Composer Studio or a MATLAB GUI, the DSP will crash. This can be noticed with the “Enable” LED flashes (this LED is on the power stage front panel). A recommended practice is to start with a relatively low sampling rate, if possible for a given application, and then increasing the rate until reaching the sampling limit.

Note that an essential step in building a Simulink block diagram using the real-time workshop is that sampling rates should be multiples of each other. Down sampling and over sampling a signal is always possible with the “rate transition” Simulink blocks if this step is obeyed.

Sampling might pose to be a problem when high sampling rates are required, e.g., having an estimation loop that is significantly faster than the controller.

For ADCs, signal-to-noise-ratio (SNR) is a very sensitive factor that needs to be considered especially in advanced applications such as ripple-based control. SNR can be enhanced with appropriate communication cable wiring and filtering — analog filtering on the control board and digital filtering on the DSP.

3.2 Memory

The flash memory on the eZDSP TMS320F2812 DSP is limited to 128K words. In most applications, this is plenty of program memory. However, with some advanced applications, it may be very important to conserve memory as this limit is approached.

With field-oriented control of an induction motor, for example, it may be necessary to have many embedded control laws to perform separate functions simultaneously using similar information. Steps should be taken to ensure that the program does not have redundant calculations. For instance, if two sections of your program need to know the currents in the d-q frame, have one outside block of your program calculate these currents in the d-q frame, and use these values in both sections. Do not have each section compute the d-q currents separately.

It is also very easy to use up memory words when filtering vector signals. It is recommended to use the lowest order filter to achieve the goal, especially with vector signals. A second order filter for a three-element vector uses much more memory than a first order filter.

Memory may also be saved by ensuring that constants are never stored more than once in memory. For instance, if several pieces of your program require multiplication by 180, only store the number 180 once and refer to it by both blocks in the program. In Simulink, these means using only one ‘Constant’ block for the number 180. You can then connect this single block to both subsystems which need to use the value 180.

4. Examples

4.1 V/Hz Example

SW00025 folder contains documents (including this one) that are dedicated to the modular inverter control software architecture, which is based on Matlab/Simulink Embedded Coder for TI C2000

toolbox. The eZdspLib.zip file contains the matlab code and model needed for Modular Inverter dsp software generation and execution.

Extract the file to a folder that you are working under, and make sure this folder is included in the matlab path setting. Keep the HyperTerminal file some where convenient for easy access. Navigate in the matlab path selection tab and make the current work directory is your matlab path/eZdspLib/VperHzTest. Two models are provided. One is using Serial communication the other is using RTDX communication. ezdsp2812_vphz32 is to be used with the HyperTerminal file you extracted earlier. ezdsp2812_vphz32_wRTDX is used with .m file under the same directory.

4.2 Flux Estimation Example

Refer to the following paper for a summary of software and hardware requirements in a flux estimation example using the modular inverter:

A.M. Bazzi and P.T. Krein, "Comprehensive Flux Estimator Implementation Procedures For Advanced Control Of Inverter-Fed Induction Machines," *How2Power Today Online Magazine/Portal*, June 2010.

5. Create Flash Based Project

SW00025 also contains a Flash_Based.zip file. It is the project that derived from V/Hz project in the example but suitable to run as stand alone program.

To make a project generated from Matlab flashable. Follow the steps listed below:

- i. Replace the .cmd file generated from Matlab/Simulink with the one in Fash_Based.zip .cmd file. Change the file name to your project name.
- ii. Add two files to the project by right click on *project_name* and choose *Add files...* in CCS: coeosl.c and DSP281x_CodeStartBranch.asm
- iii. Add one line in YourProjectName_main.c in main() so that it looks like:

```
void main(void)
{
    init_COECSL(); // memcopy from Flash to RAM

    init_board();
    ...
}
```

- iv. Right click on project name, choose *Build Option...* click on *Linker* tab; change one of the linker options from *-cr* to *-c*.
- v. Change the jumper 7 position from 2-3 to 1-2 on the DSP board.
- vi. Re-compile and build the project under CCS. **DO NOT RECOMPILE UNDER SIMULINK, OTHERWISE YOU LOSE ALL YOUR WORK.**

- vii. Instead of “loading” your program to the RAM, now you need to use flash programmer under CCS *tools* to burn the flash.
- viii. After the process is done, reset the DSP and it should go.

NOTES:

1. All work in this chapter is not related to Matlab/Simulink any more. Work solely under CCS.
2. CCS software package that comes with the boards doesn't have a functional flash programmer. Need to update the flash API patch, which can be obtained from c2000flashprogs_w_v112.zip under SW00025 directory. Or please check the latest version of CCS.
3. To save your work, change the directory name so that Matlab/Simulink won't erase it next time you recompile your model. Move it to a safer place.
4. Don't forget to change the jumper back to its original position when done.

!!! important information for upgrading the system !!!

as of 24th of May 2006, two ezdsp systems has been upgrade the Matlab to R2006a.

several known problems and solutions:

1. CCS 3.1 is required. we have one on exchange drive.
2. some blocks not backward compatible. if there is an error (example the pwm blocks) or warning, read carefully and replace the old block with new one. save your work before you do so since once saved in later version, cannot go back easily.
3. internal mem map loading issue. if experience difficulty when loading the file in CCS, try following:

open installationDir\toolbox\rtw\targets\tic2000\tic2000\ti_c281x_cmd.tlc

find the following text

...

```
%% Function: constructSectionAllocationx281xDSP =====
```

```
%%
```

```
%function constructSectionAllocationx281xDSP() void
```

```
%openfile sectionAllocation
```

```
SECTIONS
```

```
{
```

```
    codestart      : > PRAMH0,    PAGE = 0
```

```

.text      : > PRAMH0,    PAGE = 0
.cinit     : > PRAMH0,    PAGE = 0
ramfuncs   : > PRAMH0,    PAGE = 0
.reset     : > RESET,    PAGE = 0, TYPE = DSECT
%if (c2000ModelIRInfo.numRTDXs != 0)
.rtdx_text : > PRAMH0,    PAGE = 0
%endif
...

```

add a line assigning memory for .pinit after .cinit assignment so it looks like

```

%% Function: constructSectionAllocationx281xDSP
=====
%%
%function constructSectionAllocationx281xDSP() void

%openfile sectionAllocation

SECTIONS
{
    codestart      : > PRAMH0,    PAGE = 0
    .text          : > PRAMH0,    PAGE = 0
    .cinit         : > PRAMH0,    PAGE = 0
    .pinit         : > PRAMH0,    PAGE = 0
    ramfuncs       : > PRAMH0,    PAGE = 0
    .reset         : > RESET,    PAGE = 0, TYPE = DSECT
%if (c2000ModelIRInfo.numRTDXs != 0)
    .rtdx_text     : > PRAMH0,    PAGE = 0
%endif
%endif

```