

**APPLICATION OF  
DIGITAL SIGNAL PROCESSING  
ON  
TMS320C6713 DSK**

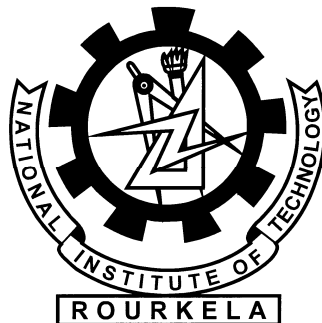
*A PROJECT REPORT*

*Submitted in partial fulfillment of the requirements for the award of the degree  
Of*

**BACHELOR OF TECHNOLOGY  
IN  
ELECTRONICS AND INSTRUMENTATION ENGINEERING**

**by**

**MANAS MURMU (10407030)**



*Department of Electronics and Communication Engineering  
National Institute Of Technology, Rourkela  
Pin-769008, Orissa, INDIA  
2007 – 2008*

**APPLICATION OF  
DIGITAL SIGNAL PROCESSING  
ON  
TMS320C6713 DSK**

*A PROJECT REPORT*

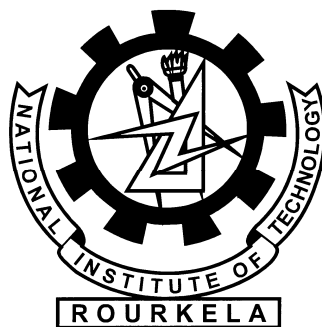
*Submitted in partial fulfillment of the requirements for the award of the degree  
Of*

**BACHELOR OF TECHNOLOGY  
IN  
ELECTRONICS AND INSTRUMENTATION ENGINEERING**

by

MANAS MURMU (10407030)

Under the guidance of  
**Dr.S.K.PATRA**



*Department of Electronics and Communication Engineering  
National Institute Of Technology, Rourkela  
Pin-769008, Orissa, INDIA  
2007 – 2008*



**National Institute of Technology  
Rourkela**

## **CERTIFICATE**

This is to certify that the thesis entitled, “Application of digital signal processing on TMS320C6713 DSK” submitted by Sri Manas Murmu in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics & Instrumentation Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

**Prof. S. K. PATRA**  
Dept. of Electronics & Communication Engg  
National Institute of Technology  
Rourkela - 769008

# ACKNOWLEDGEMENT

*We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide **Prof. S.K.Patra**, Professor, Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, in bringing this report to a successful completion.*

*We are grateful to **Prof. G.Panda**, Head of the Department of Electronics and Communication Engineering, for permitting us to make use of the facilities available in the department to carry out the project successfully. Last but not the least we express our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.*

*Finally we extend our gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.*

Manas Murmu  
(10407030)

# ABSTRACT

Signal processing concepts are often presented in a very mathematical and abstract format. This can discourage students from further exploration because of the apparent irrelevance to real world problems. A common solution is to provide a hands-on laboratory to illustrate applications of abstract concepts. However, hardware-based digital signal processing (DSP) laboratories – which are typically incorporated into senior-level signal processing courses – usually emphasize programming the DSP chip rather than exploring algorithms and applications.

This paper is an report on the familiarization process of the TMS320C6713 and the implementation of digital signal processing projects. The Texas Instruments C6713 DSK platform, which can be programmed using SIMULINK (The Mathworks, Inc.). This gives us added advantage of easily writing codes on MATLAB and implementing it on the DSP processors.

# TABLE OF CONTENTS

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iii</b>
<b>List of Figures and Tables</b>	<b>v</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Introduction	2
1.2 History, Development, and Advantages of TMS320 DSPs	3
1.3 Difference between DSPs and other Microprocessors	3
1.4 Important feature of DSP's	4
1.5 Typical Applications for the TMS320 Family	7
<b>Chapter 2: TMS320C6713 DSK</b>	<b>8</b>
2.1 Introduction	9
2.2 Diagrams of TMS320C6713 DSK	9
2.3 Features	10
2.4 Functional Overview	11
2.5 Architecture	11
2.6 Peripherals of TMS320C6713	15
<b>Chapter 3: Programming the TMS320C6713</b>	<b>21</b>
3.1 Introduction	22
3.2 Power On Self Test (POST)	22
3.3 Code Composer Studio	23
3.4 Steps involved in programming	26
3.5 Matlab Simulink Modelling	30
3.6 Problems faced during the implementation of Chassaing/kehtarnavaz examples	31

<b>Chapter 4: Programming the TMS320C6713</b>	<b>33</b>
4.1.Sinegraph	34
4.2.Create a project in CCS	37
4.3.Building and running of project	37
<b>Chapter 5: Conclusion</b>	<b>40</b>
<b>References</b>	<b>42</b>

# List of Figures and Tables

<i>Figure 1.4</i>	<i>DSP 's architecture</i>
<i>Figure 2.2.1</i>	<i>Block Diagram of TMS320C6713 DSK</i>
<i>Figure 2.2.2</i>	<i>Board Diagram of TMS320C6713 DSK</i>
<i>Figure 2.5.1</i>	<i>Simplified block diagram of TMS320C67xx family</i>
<i>Figure 2.5.1.1</i>	<i>TMS320C67X data path</i>
<i>Table 2.5.3.1:</i>	<i>Functional Units and Descriptions</i>
<i>Figure 3.1.</i>	<i>Simplified Code Composer Studio IDE Development Flow</i>
<i>Figure 3.1.</i>	<i>Diagnostic Utility Check</i>
<i>Figure 3.4.1.</i>	<i>Connecting C6713 DSK</i>
<i>Figure 3.4.2.1</i>	<i>Code composer Studio</i>
<i>Figure 3.4.2.2</i>	<i>After including all necessary files</i>
<i>Figure 3.4.3.1</i>	<i>Compiler Comments</i>
<i>Figure 3.3.4.1</i>	<i>Load Program.</i>
<i>Figure 3.4.4.2</i>	<i>Program running.</i>
<i>Fig 3.5.1</i>	<i>Dissection of model system in matlab coding</i>
<i>Fig 3.5.2</i>	<i>Steps in the modeling of a system using simulink</i>
<i>Figure 4.3.1</i>	<i>Time domain graph property dialog</i>
<i>Figure 4.3.2</i>	<i>Frequency domain graph property dialog</i>



# Chapter 1

INTRODUCTION

## **1.1. Introduction**

Digital signal processing is one of the core technologies, in rapidly growing application areas, such as wireless communications, audio and video processing and industrial control. The number and variety of products that include some form of digital signal processing has grown dramatically over the last few years. DSP has become a key component, in many of the consumer, communications, medical and industrial products which implement the signal processing using microprocessors, Field Programmable Gate Arrays (FPGAs), Custom ICs etc. Due to increasing popularity of the above mentioned applications, the variety of the DSP-capable processors has expanded greatly. DSPs are processors or microcomputers whose hardware, software, and instruction sets are optimized for high-speed numeric processing applications, an essential for processing digital data, representing analog signals in real time. The DSP processors have gained increased popularity because of the various advantages like reprogram ability in the field, cost-effectiveness, speed, energy efficiency etc.

Digital signal processors such as the TMS320C6x (C6x) family of processors are like fast special-purpose microprocessors with a specialized type of architecture and an instruction set appropriate for signal processing. The C6x notation is used to designate a member of Texas Instruments' (TI) TMS320C6000 family of digital signal processors. The architecture of the C6x digital signal processor is very well suited for numerically intensive calculations. Based on a very-long-instruction-word (VLIW) architecture, the C6x is considered to be TI's most powerful processor. Digital signal processors are used for a wide range of applications, from communications and controls to speech and image processing. The general-purpose digital signal processor is dominated by applications in communications (cellular). Applications embedded digital signal processors are dominated by consumer products. They are found in cellular phones, fax/modems, disk drives, radio, printers, hearing aids, MP3 players, high-definition television (HDTV), digital cameras, and so on. These processors have become the products of choice for a number of consumer applications, since they have become very cost-effective. They can handle different tasks, since they can be reprogrammed readily for a different application.

DSP techniques have been very successful because of the development of low-cost software and hardware support. For example, modems and speech recognition can be less expensive using DSP techniques. DSP processors are concerned primarily with real-time signal processing. Realtime processing requires the processing to keep pace with some external event, whereas non-real-time processing has no such timing constraint. The external event to keep pace with is usually the analog input. Whereas analog-based systems with discrete electronic components such as resistors can be more sensitive to temperature changes, DSP-based systems are less affected by environmental conditions. DSP processors enjoy the advantages of microprocessors. They are easy to use, flexible, and economical.

## **1.2. History, Development, and Advantages of TMS320 DSPs**

Advantages of DSP's over Analog Circuits

- Can implement complex linear or nonlinear algorithms.
- Can modify easily by changing software.
- Reduced parts count makes fabrication easier.
- High reliability

## **1.3. Difference between DSPs and Other Microprocessors**

Over the past few years it is seen that general purpose computers are capable of performing two major tasks.

(1) Data Manipulation, and

(2) Mathematical Calculations

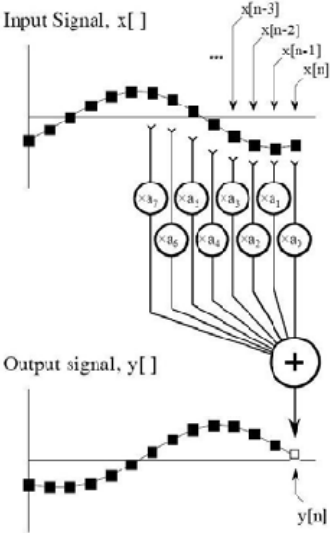
All the microprocessors are capable of doing these tasks but it is difficult to make a device which can perform both the functions optimally, because of the involved technical trade offs like the size of the instruction set, how interrupts are handled etc. As a broad generalization these factors have made traditional microprocessors such as Pentium Series, primarily directed at data manipulation. Similarly DSPs are designed to perform the mathematical calculations needed in Digital Signal Processing,

Data manipulation involves storing and sorting of information. For instance, a word processing program does a basic task of storing, organizing and retrieving of the information. This is achieved by *moving* data from one location to another and testing for inequalities ( $A=B$ ,  $A<B$  etc.). While mathematics is occasionally used in this type of application, it is infrequent and does not significantly affect the overall execution speed. In comparison to this, the execution speed of most of the DSP algorithms is limited almost completely by the number of multiplications and additions required.

In addition to performing mathematical calculations very rapidly, DSPs must also have a *predictable* execution time, [1]. Most DSPs are used in applications where the processing is continuous, not having a defined start or end. The cost, power consumption, design difficulty etc increase along with the execution speed, which makes an accurate knowledge of the execution time, critical for selecting proper device, as well as algorithms that can be applied. DSPs can also perform the tasks in parallel instead of serial in case of traditional microprocessors.

**1.4. Important feature of DSP's**

As the DSP processors are designed and optimized for implementation of various DSP algorithms, most processors share various common features to support the high performance, repetitive, numeric intensive tasks.



### **1.4.1 MACs and Multiple Execution Units**

The most commonly known and used feature of a DSP processor is the ability to perform one or more multiply-accumulate operation (also called as “MACs”) in a single instruction cycle. The MAC operation is useful in DSP algorithms that involve computing a vector dot product, such as digital filters, correlation, and Fourier transforms. The MAC operation becomes useful as the DSP applications typically have very high computational requirements in comparison to other types of computing tasks, since they often must execute DSP algorithms (such as FIR filtering) in real time on lengthy segments of signals sampled at 10-100 KHz or higher. To facilitate this DSP processors often include several independent execution units that are capable of operating in parallel.

### **1.4.2 Efficient Memory Access**

DSP processors also share a feature of efficient memory access i.e. the ability to complete several accesses to memory in a single instruction cycle. Due to Harvard architecture in DSPs, i.e. physically separate storage and signal pathways for instructions and data, and pipelined structure the processor is able to fetch an instruction while simultaneously fetching operands and/or storing the result of previous instruction to memory. In some recently available DSPs a further optimization is done by including a small bank of RAM near the processor core, often termed as L1 memory, which is used as an instruction cache. When a small group of instructions is executed repeatedly, the cache is loaded with these instructions thus making the bus available for data fetches, instead of instruction fetches.

### **1.4.3 Circular Buffering**

The need of processing the digital signals in real time, where in the output (processed samples) have to be produced at the same time at which the input samples are being acquired, evolves the concept of Circular Buffering. For instance this is needed in telephone communication, hearing aids, radars etc. Circular buffers are used to store the most recent values of a continually updated signal. Circular buffering allows processors to access a block of data sequentially and then automatically wrap around to the beginning address exactly the pattern used to access

coefficients in FIR filter. Circular buffering also very helpful in implementing first-in, first-out buffers, commonly used for I/O and for FIR delay lines

#### **1.4.4 Dedicated Address Generation Unit**

The dedicated address generation units also help speed up the performance of the arithmetic processing on DSP. Once an appropriate addressing registers have been configured, the address generation unit operates in the background. (i.e. without using the main data path of the processor). The address required for operand access is now formed by the address generation unit in parallel with the execution of the arithmetic instruction. DSP processor address generation units typically support a selection of addressing modes tailored to DSP applications. The most common of these is *register-indirect addressing with post-increment*, which is used in situations where a repetitive computation is performed on data stored sequentially in memory. Some processors also support *bit-reversed* addressing, which increases the speed of certain fast Fourier transform (FFT) algorithms.

#### **1.4.5 Specialized Instruction Sets**

The instruction sets of the digital signal processors are designed to make maximum use of the processors' resources and at the same time minimize the memory space required to store the instructions. Maximum utilization of the DSPs' resources ensures the maximum efficiency and minimizing the storage space ensures the cost effectiveness of the overall system.

To ensure the maximum use of the underlying hardware of the DSP, the instructions are designed to perform several parallel operations in a single instruction, typically including fetching of data in parallel with main arithmetic operation. For achieving minimum storage requirements the DSPs' instructions are kept short by restricting which register can be used with which operations and which operations can be combined in an instruction.

Some of the latest processors use VLIW (very long instruction word) architectures, where in multiple instructions are issued and executed per cycle. The instructions in such architectures are short and designed to perform much less work compared to those of conventional DSPs thus requiring less memory and increased speed because of the VLIW architecture.

## 1.5. Typical Applications for the TMS320 Family

- **Telecommunications:** telephone line modems, FAX, cellular telephones, wireless networks, speaker phones, answering machines
- **Voice/Speech:** speech digitization and compression, voice mail, speaker verification, and speech synthesis
- **Automotive:** engine control, antilock brakes, active suspension, airbag control, and system diagnosis
- **Control Systems:** head positioning servo systems in disk drives, laser printer control, robot control, engine and motor control, and numerical control of automatic machine tools
- **Military:** radar and sonar signal processing, navigation systems, missile guidance, HF radio frequency modems, secure spread spectrum radios, and secure voice
- **Medical:** hearing aids, MRI imaging, ultrasound imaging, and patient monitoring
- **Instrumentation:** spectrum analysis, transient analysis, signal generators
- **Image Processing:** HDTV, image enhancement, image compression and transmission, 3-D rotation, and animation

# Chapter 2

TMS320C6713 DSK



## 2.1. Introduction

The TMS320C6x are the first processors to use velociTI architecture, having implemented the VLIW architecture. The TMS320C62x is a 16-bit fixed point processor and the '67x is a floating point processor, with 32-bit integer support. The discussion in this chapter is focused on the TMS320C67x processor. The architecture and peripherals associated with this processor are also discussed.

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.

## 2.2. Diagrams of TMS320C6713 DSK

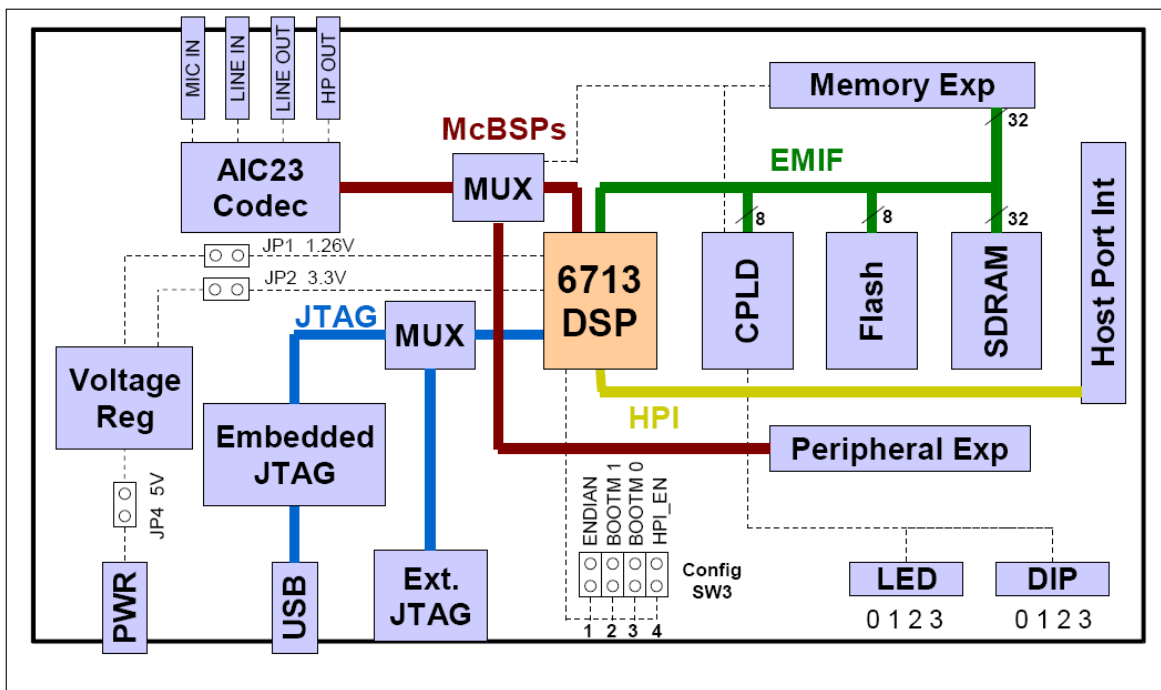
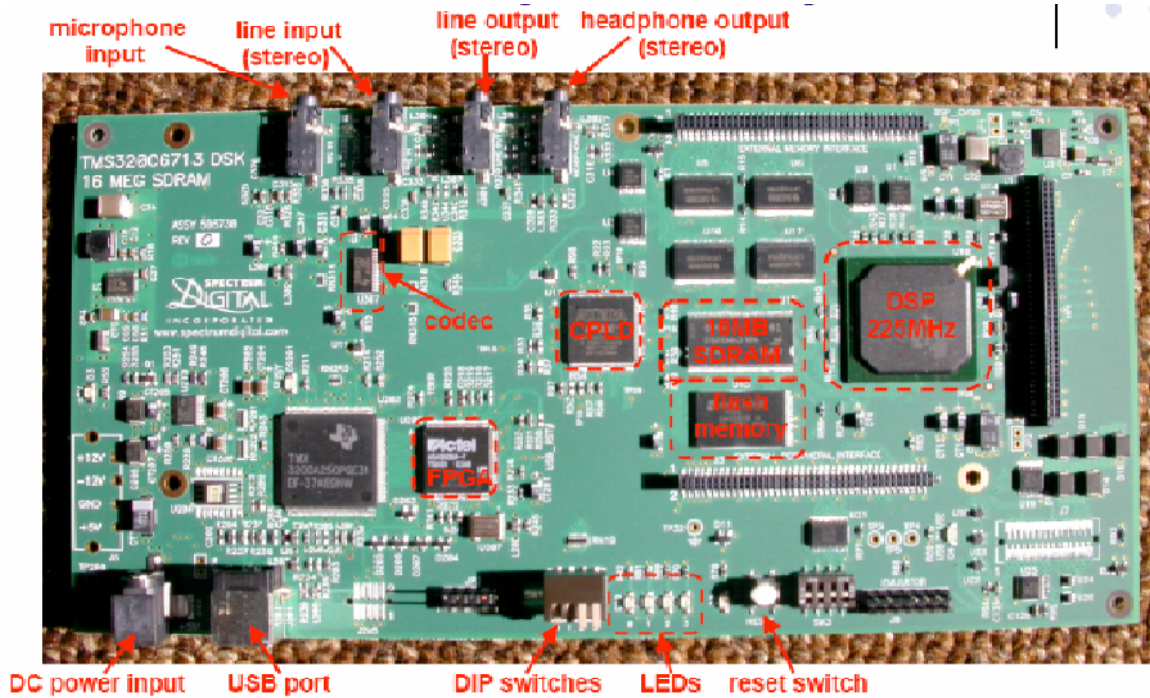


Fig 2.2.1 Block diagram of TMS320C6713 DSK



**Fig 2.2.2 Board diagram of TMS320C6713 DSK**

### 2.3. Features

The DSK comes with a full compliment of on-board devices that suit a wide variety of application environments. Key features include:

- A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec
- 16 Mbytes of synchronous DRAM
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration)
- 4 user accessible LEDs and DIP switches
- Software board configuration through registers implemented in CPLD
- Configurable boot options
- Standard expansion connectors for daughter card use
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
- Single voltage power supply (+5V)

## **2.4. Functional Overview**

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory InterFace). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also connected daughter card expansion connectors which are used for third party add-in boards.

The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP0 is used to send commands to the codec control interface while McBSP1 is used for digital audio data. McBSP0 and McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to its registers.

The DSK includes 4 LEDs and a 4 position DIP switch as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

An included 5V external power supply is used to power the board. On-board switching voltage regulators provide the +1.26V DSP core voltage and +3.3V I/O supplies. The board is held in reset until these supplies are within operating specifications.

Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

## **2.5. Architecture**

The simplified architecture of TMS320C6713 is shown in the Figure 3.1 below. The processor consists of three main parts: CPU, peripherals and memory.

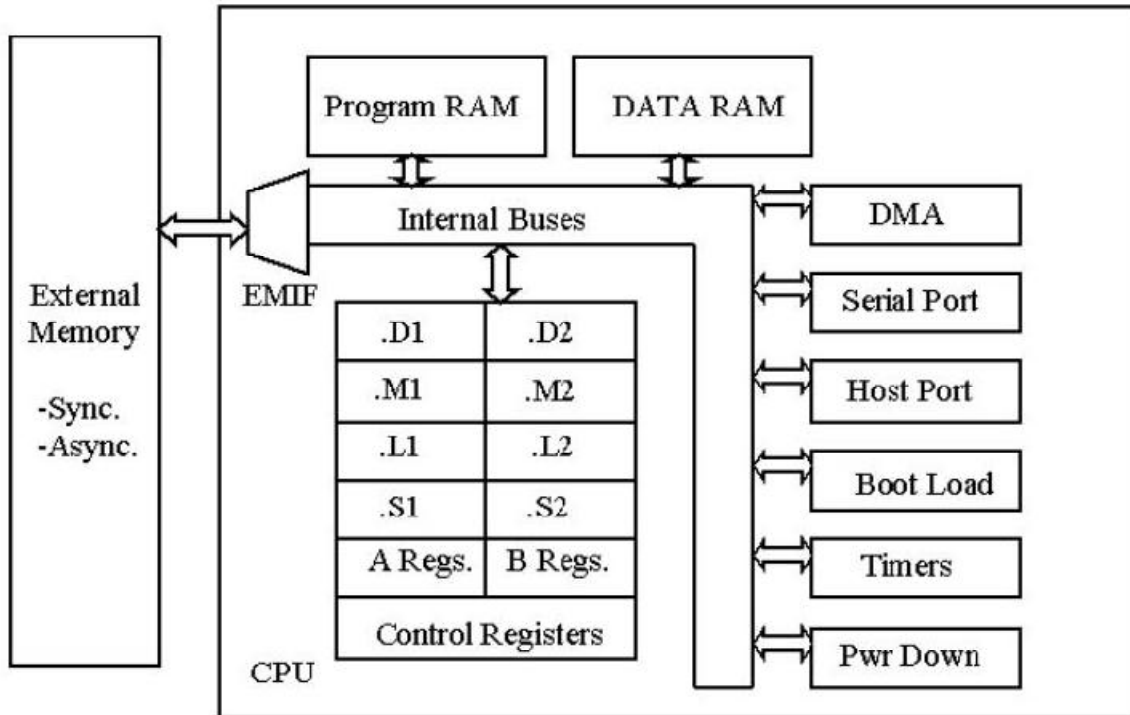
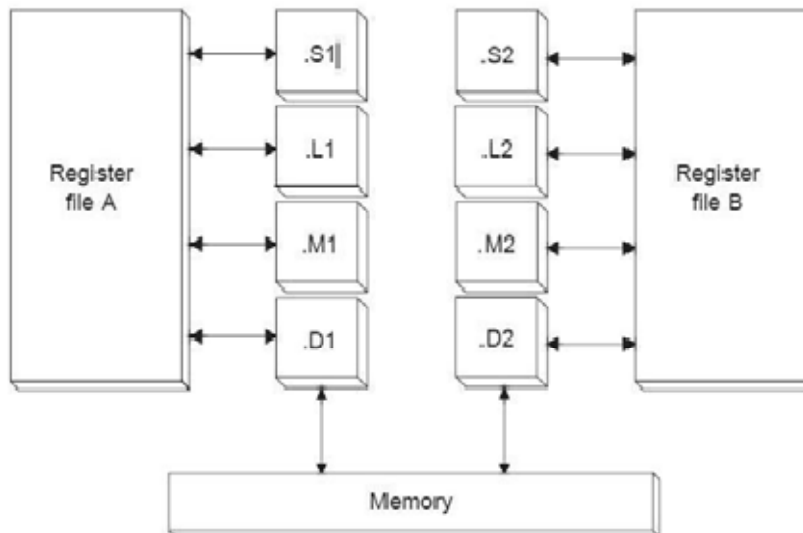


Figure 2.5.1: Simplified block diagram of TMS320C67xx family

### 2.5.1. Central Processing Unit

The CPU contains program fetch unit, Instruction dispatch unit, instruction decode unit. The CPU fetches advanced very-long instruction words (VLIW) (256 bits wide) to supply up to eight 32-bit instructions to the eight functional units during every clock cycle. The VLIW architecture features controls by which all eight units do not have to be supplied with instructions if they are not ready to execute. The first bit of every 32-bit instruction determines if the next instruction belongs to the same execute packet as the previous instruction, or whether it should be executed in the following clock as a part of the next execute packet. Fetch packets are always 256 bits wide; however, the execute packets can vary in size. The variable-length execute packets are a key memory-saving feature, distinguishing the C67x CPU from other VLIW architectures. The CPU also contains two data paths (Containing registers A and B respectively) in which the processing takes place. Each data path has four functional units (.L, .M, .S and .D).

The functional units execute logic, multiply, shifting and data address operation. Figure 2.2 shows the simplified block diagram of the two data paths.



**Figure 2.5.1.1: TMS320C67X data path**

All instructions except loads and stores operate on the register. All data transfers between the register files and memory take place only through two data-addressing units (.D1 and .D2). The CPU also has various control registers, control logic and test, emulation and logic. Access to control registers is provided from data path B.

## **2.5.2. General Purpose Register Files**

The CPU contains two general purpose register files A and B. These can be used for data or as data address pointers. Each file contains sixteen 32-bit registers (A0-A15 for file A and B0-B15 for file B). The registers A1, A2, B0, B1, B2 can also be used as condition registers. The registers A4-A7 and B4-B7 can be used for circular addressing.

These registers provide 32-bit and 40-bit fixed-point data. The 32-bit data can be stored in any register. For 40-bit data, processor stores least significant 32 bits in an even register and remaining 8 bits in upper (odd) register.

### 2.5.3. Functional Units

The CPU features two sets of functional units. Each set contains four units and a register file. One set contains functional units .L1, .S1, .M1, and .D1; the other set contains units .D2, .M2, .S2, and .L2. The two register files each contain sixteen 32-bit registers for a total of 32 general-purpose registers. The two sets of functional units, along with two register files, compose sides A and B of the CPU. Each functional unit has two 32-bit read ports for source operands and one 32-bit write port into a general purpose register file. The functional units .L1, .S1, .M1, and .D1 write to register file A and the functional units .L2, .S2, .M2, and .D2 write to register file B. As each unit has its own 32-bit write port, all eight ports can be used in parallel in every cycle. The .L, .S, and .M functional units are ALUs. They perform 32-bit/40-bit arithmetic and logical operations. .S unit also performs branching operations and .D units perform linear and circular address calculations. Only .S2 unit performs accesses to control register file.

Table 2.1 describes the functional unit along with its description.

Functional Unit	Description
.L unit (.L1, .L2)	32/40-bit arithmetic and compare operations Left most 1, 0, bit counting for 32 bits Normalization count for 32 and 40 bits 32 bit logical operations 32/64-bit IEEE floating-point arithmetic Floating-point/fixed-point conversions
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40 bit shifts and 32-bit bit-field operations 32 bit logical operations Branching Constant generation Register transfers to/from the control register file 32/64-bit IEEE floating-point compare operations 32/64-bit IEEE floating-point reciprocal and square root reciprocal approximation
.M unit (.M1, .M2)	16 x 16 bit multiplies 32 x 32-bit multiplies Single-precision (32-bit) floating-point IEEE multiplies Double-precision (64-bit) floating-point IEEE multiplies
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation

**Table 2.5.3.1: Functional Units and Descriptions**

## **2.5.4. Memory System**

The memory system of the TMS320C671x series processor implements a modified Harvard architecture, providing separate address spaces for instruction and data memory.

The processor uses a two-level cache-based architecture and has a powerful and diverse set of peripherals. The Level 1 program cache (L1P) is a 4K-byte direct-mapped cache and the Level 1 data cache (L1D) is a 4K-byte 2-way set-associative cache. The Level 2 memory/cache (L2) consists of a 256K-byte memory space that is shared between program and data space. 64K bytes of the 256K bytes in L2 memory can be configured as mapped memory, cache, or combinations of the two. The remaining 192K bytes in L2 serve as mapped SRAM.

## **2.6. Peripherals of TMS320C6713**

The TMS320C67x devices contain peripherals for communication with off-chip memory, co-processors, host processors and serial devices. The following subsections discuss the peripherals of 'C6713 processor.

### **2.6.1 Enhanced DMA**

The enhanced direct memory access (EDMA) controller transfers data between regions in the memory map without interference by the CPU. The EDMA provides transfers of data to and from internal memory, internal peripherals, or external devices in the background of CPU operation. The EDMA has sixteen independently programmable channels allowing sixteen different contexts for operation.

The EDMA can read or write data element from source or destination location respectively in memory. EDMA also provides combined transfers of data elements such as frame transfer and block transfer. Each EDMA channel has an independently programmable number of data elements per frame and number of frames per block.

The EDMA has following features:

- Background operation: The DMA operates independently of the CPU.
- High throughput: Elements can be transferred at the CPU clock rate.
- Sixteen channels: The EDMA can keep track of the contexts of sixteen independent transfers.

- Split operation: A single channel may be used simultaneously to perform both receive and transmit element transfers to or from two peripherals and memory.
- Programmable priority: Each channel has independently programmable priorities versus the CPU.
- Each channel's source and destination address registers can have configurable indexes for each read and write transfer. The address may remain constant, increment, decrement, or be adjusted by a programmable value.
- Programmable-width transfers: Each channel can be independently configured to transfer bytes, 16-bit half words, or 32-bit words.
- Authentication: Once a block transfer is complete, an EDMA channel may automatically reinitialize itself for the next block transfer.
- Linking: Each EDMA channel can be linked to a subsequent transfer to perform after completion.
- Event synchronization: Each channel is initiated by a specific event. Transfers may be either synchronized by element or by frame.

### **2.6.2 Host Port Interface**

The Host-Port Interface (HPI) is a 16-bit wide parallel port through which a host processor can directly access the CPU's memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals.

The HPI is connected to the internal memory via a set of registers. Either the host or the CPU may use the HPI Control register (HPIC) to configure the interface. The host can access the host address register (HPIA) and the host data register (HPID) to access the internal memory space of the device. The host accesses these registers using external data and interface control signals. The HPIC is a memory-mapped register, which allows the CPU access.

The data transactions are performed within the EDMA, and are invisible to the user.



### **2.6.3 External Memory Interface**

The external memory interface (EMIF) supports an interface to several external devices, allowing additional data and program memory space beyond that which is included on-chip.

The types of memories supported include:

- Synchronous burst SRAM (SBSRAM)
- Synchronous DRAM (SDRAM)
- Asynchronous devices, including asynchronous SRAM, ROM, and FIFOs. The EMIF provides highly programmable timings to these interfaces.
- External shared-memory devices

There are two data ordering standards in byte-addressable microcontrollers exist:

- Little-endian ordering, in which bytes are ordered from right to left, the most significant byte having the highest address.
- Big-endian ordering, in which bytes are ordered from left to right, the most significant byte having the lowest address.

The EMIF reads and writes both big- and little-endian devices. There is no distinction between ROM and asynchronous interface. For all memory types, the address is internally shifted to compensate for memory widths of less than 32 bits.

### **2.6.4 Multichannel Buffered Serial Port**

The C62x/C67x multichannel buffered serial port (McBSP) is based on the standard serial port interface found on the TMS320C2000 and C5000 platforms. The standard serial port interface provides:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for reception and transmission
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- External shift clock generation or an internal programmable frequency shift clock

- Multichannel transmission and reception of up to 128 channels.
- An element sizes of 8-, 12-, 16-, 20-, 24-, or 32-bit.
- $\mu$ -Law and A-Law companding.
- 8-bit data transfers with LSB or MSB first.
- Programmable polarity for both frame synchronization and data clocks.
- Highly programmable internal clock and frame generation.

Data communication between McBSP and the devices interfaced takes place via two different pins for transmission and reception – data transmit (DX) and data receive (RX) respectively. Control information in the form of clocking and frame synchronization is communicated via CLKX, CLKR, FSX, and FSR. 32-bit wide control registers are used to communicate McBSP with peripheral devices through internal peripheral bus. CPU or DMA write the DATA to be transmitted to the Data transmit register (DXR) which is shifted out to DX via the transmit shift register (XSR). Similarly, receive data on the DR pin is shifted into the receive shift register (RSR) and copied into the receive buffer register (RBR). RBR is then copied to DRR, which can be read by the CPU or the DMA controller. This allows internal data movement and external data communications simultaneously.

### **2.6.5 Timers**

The 'C62x/C67x has two 32-bit general-purpose timers that can be used to:

- Time events
- Count events
- Generate pulses
- Interrupt the CPU
- Send synchronization events to the DMA controller

The timer works in one of the two signaling modes depending on whether clocked by an internal or an external source. The timer has an input pin (TINP) and an output pin (TOUT). The TINP pin can be used as a general purpose input, and the TOUT pin can be used as a general-purpose output.

When an internal clock is provided, the timer generates timing sequences to trigger peripheral or external devices such as DMA controller or A/D converter respectively. When an external clock is provided, the timer can count external events and interrupt the CPU after a specified number of events.

### **2.6.6 Multichannel Audio Serial Port**

The 'C6713 processor includes two Multichannel Audio Serial Ports (McASP). The McASP interface modules each support one transmit and one receive clock zone. Each of the McASP has eight serial data pins which can be individually allocated to any of the two zones. The serial port supports time-division multiplexing on each pin from 2 to 32 time slots. The C6713B has sufficient bandwidth to support all 16 serial data pins transmitting a 192 kHz stereo signal. Serial data in each zone may be transmitted and received on multiple serial data pins simultaneously and formatted in a multitude of variations on the Philips Inter-IC Sound (I2S) format,

In addition, the McASP transmitter may be programmed to output multiple S/PDIF IEC60958, AES-3, CP-430 encoded data channels simultaneously, with a single RAM containing the full implementation of user data and channel status fields.

The McASP also provides extensive error-checking and recovery features, such as the bad clock detection circuit for each high-frequency master clock which verifies that the master clock is within a programmed frequency range.

### **2.6.7 Power Down Logic**

Most of the operating power of CMOS logic is dissipated during circuit switching, from one logic state to another. By preventing some or all of the chip's logic from switching, significant power savings can be realized without losing any data or operational context. Power-down mode PD1 blocks the internal clock inputs at the boundary of the CPU, preventing most of its logic from switching, effectively shutting down the CPU. Additional power savings are accomplished in power-down mode PD2, in which the entire on chip clock structure (including multiple buffers) is halted at the output of the PLL. Power-down mode PD3 shuts down the entire internal clock tree (like PD2) and also disconnects the external clock source (CLKIN) from reaching the

PLL. Wake-up from PD3 takes longer than wake-up from PD2 because the PLL needs to be relocked, just as it does following power up.

# Chapter 3

Programming the TMS320C6713

### 3.1. Introduction

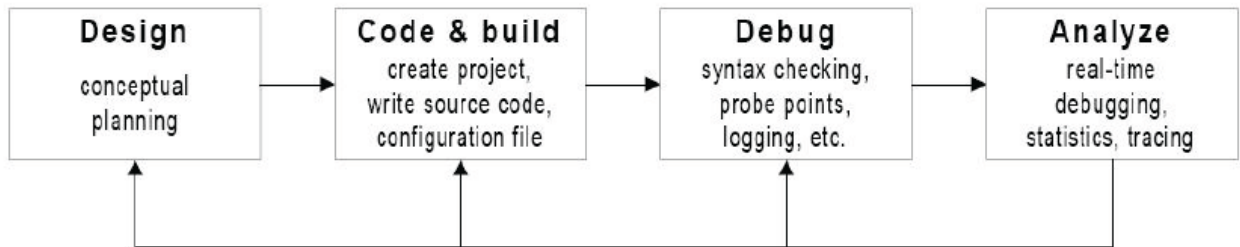
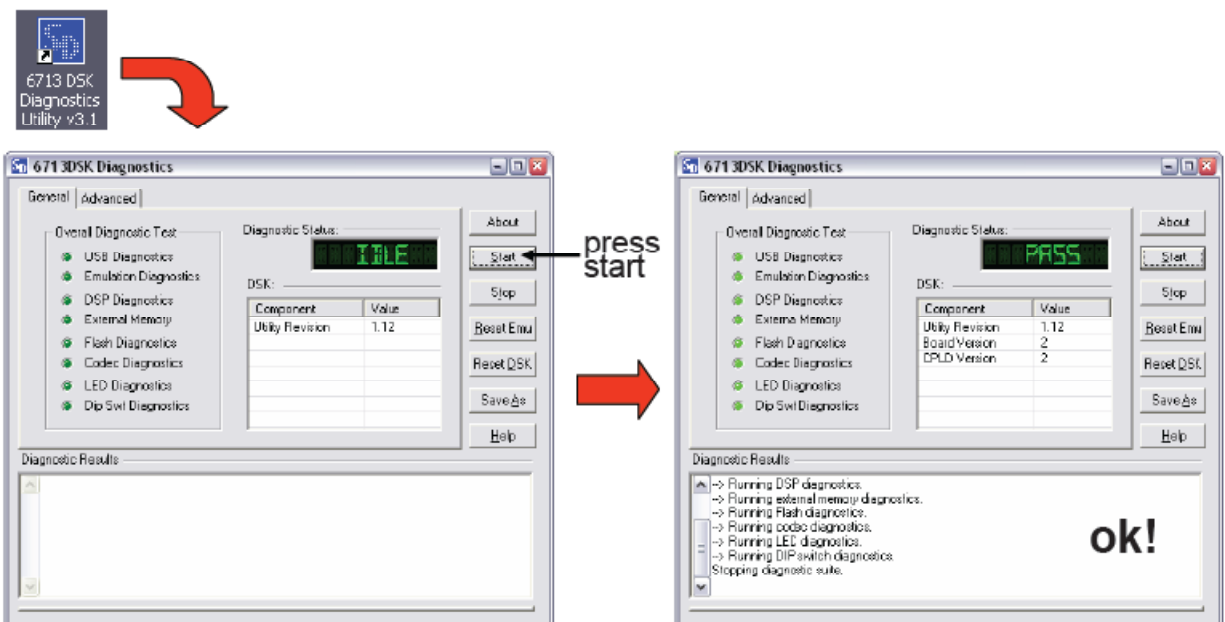


Figure 3.1. Simplified Code Composer Studio IDE Development Flow

### 3.2. Power On Self Test (POST)

- Power up DSK and watch LEDs
- Power On Self Test (POST) program stored in FLASH memory automatically executes
- POST takes 10-15 seconds to complete
- All DSK subsystems are automatically tested
- During POST, a 1kHz sinusoid is output from the AIC23 codec for 1 second
- Listen with headphones or watch on oscilloscope
- If POST is successful, all four LEDs blink 3 times and then remain on
- Use DSK diagnostic utility to test DSK functionality



### **3.3. Code Composer Studio (CCS)**

CCS provides an IDE to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs.

The C compiler compiles a C source program with extension *.c* to produce an assembly source file with extension *.asm*. The assembler assembles an *.asm* source file to produce a machine language object file with extension *.obj*. The linker combines object files and object libraries as input to produce an executable file with extension *.out*. This executable file represents a linked common object file format (COFF), popular in Unix-based systems and adopted by several makers of digital signal processors [25]. This executable file can be loaded and run directly on the C6713 processor. A linear optimizer optimizes this source file to create an assembly file with extension *.asm* (similar to the task of the C compiler).

To create an application project, one can “add” the appropriate files to the project. Compiler/linker options can readily be specified. A number of debugging features are available, including setting breakpoints and watching variables; viewing memory, registers, and mixed C and assembly code; graphing results; and monitoring execution time. One can step through a program in different ways (step into, over, or out).

Real-time analysis can be performed using real-time data exchange (RTDX). RTDX allows for data exchange between the host PC and the target DSK, as well as analysis in real time without stopping the target. Key statistics and performance can be monitored in real time. Through the joint team action group (JTAG), communication with on-chip emulation support occurs to control and monitor program execution. The C6713 DSK board includes a JTAG interface through the USB port.

#### **3.3.1. CCS installation and Support**

Use the USB cable to connect the DSK board to the USB port on the PC. Use the 5-V power supply included with the DSK package to connect to the +5-V power connector on the DSK to turn it on. Install CCS with the CD-ROM included with the DSK, preferably using the *c:\C6713* structure (in lieu of *c:\ti* as the default).

The CCS icon should be on the desktop as “C6713DSK CCS” and is used to launch CCS. The code generation tools (C compiler, assembler, linker) are used with CCS version 2.x.

CCS provides useful documentations included with the DSK package on the following (see the Help icon):

1. Code generation tools (compiler, assembler, linker, etc.)
2. Tutorials on CCS, compiler, RTDX
3. DSP instructions and registers
4. Tools on RTDX, DSP/basic input/output system (DSP/BIOS), and so on. An extensive amount of support material (*pdf* files) is included with CCS. There are also examples included with CCS within the folder *c:\C6713\examples*. They illustrate the board and chip support library files, DSP/BIOS, and so on. CCS Version 2.x was used to build and test the examples included in this book. A number of files included in the following subfolders/directories within *c:\C6713* (suggested structure during CCS installation) can be very useful:

1. *myprojects*: a folder supplied only for your projects. All the folders in the accompanying book CD should be placed within this subdirectory.
2. *bin*: contains many utilities.
3. *docs*: contains documentation and manuals.
4. *c6000\cgtools*: contains code generation tools.
5. *c6000\RTDX*: contains support files for real-time data transfer.
6. *c6000\bios*: contains support files for DSP/BIOS.
7. *examples*: contains examples included with CCS.
8. *tutorial*: contains additional examples supplied with CCS.

### **3.3.2. Useful Types of Files**

You will be working with a number of files with different extensions. They include:

1. *file.pjt*: to create and build a project named file
2. *file.c*: C source program
3. *file.asm*: assembly source program created by the user, by the C compiler, or by the linear optimizer
4. *file.sa*: linear assembly source program. The linear optimizer uses *file.sa* as input to produce an assembly program *file.asm*
5. *file.h*: header support file



6. file.lib: library file, such as the run-time support library file rts6700.lib
7. file.cmd: linker command file that maps sections to memory
8. file.obj: object file created by the assembler
9. file.out: executable file created by the linker to be loaded and run on the C6713 processor
10. file.cdb: configuration file when using DSP/BIOS

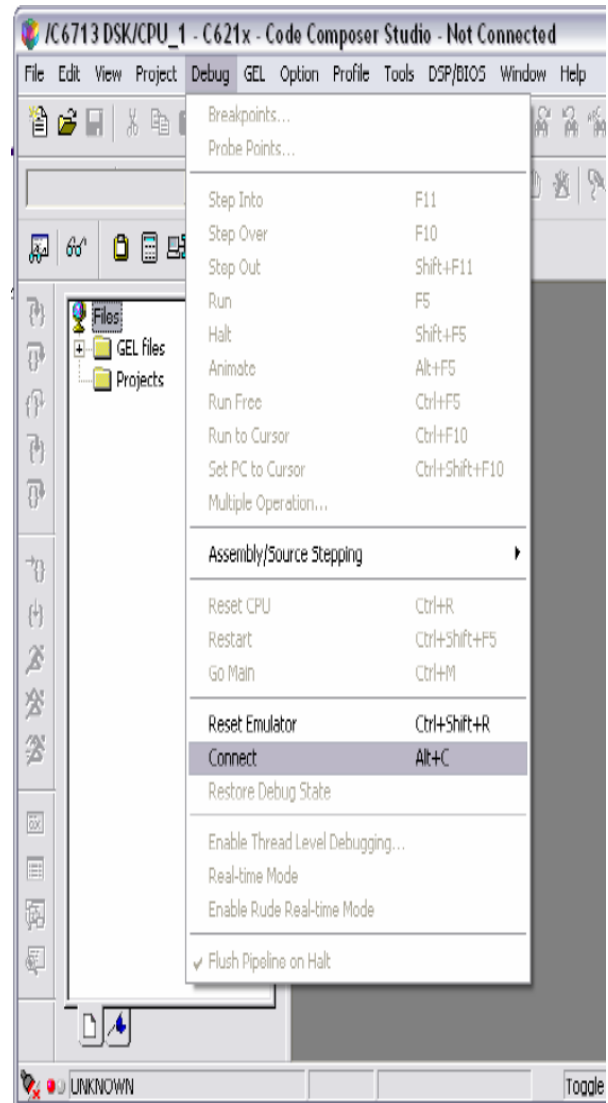
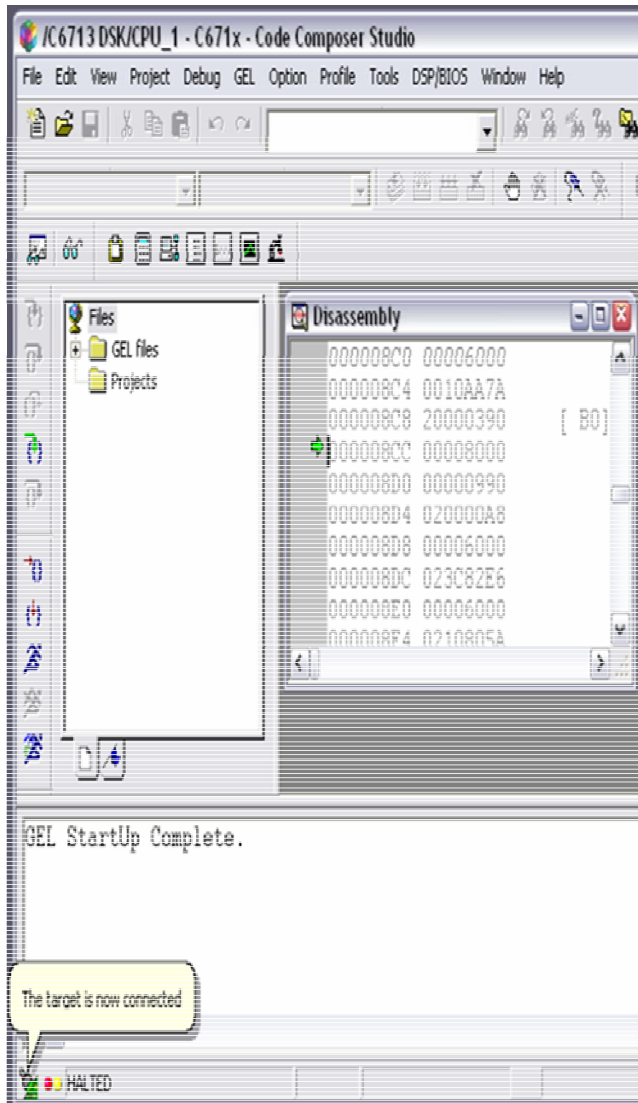
### 3.3.3. Support Files

The following support files located in the folder support (except the library files) are used for most of the examples and projects discussed in this book:

1. C6713dskinit.c: contains functions to initialize the DSK, the codec, the serial ports, and for I/O. It is not included with CCS.
2. C6713dskinit.h: header file with function prototypes. Features such as those used to select the mic input in lieu of line input (by default), input gain, and so on are obtained from this header file (modified from a similar file included with CCS).
3. C6713dsk.cmd: sample linker command file. This generic file can be changed when using external memory in lieu of internal memory.
4. Vectors\_intr.asm: a modified version of a vector file included with CCS to handle interrupts. Twelve interrupts, INT4 through INT15, are available, and INT11 is selected within this vector file. They are used for interrupt-driven programs.
5. Vectors\_poll.asm: vector file for programs using polling.
6. *rts6700.lib*, *dsk6713bsl.lib*, *csl6713.lib*: run-time, board, and chip support library files, respectively. These files are included with CCS and are located in *C6000\cgtools\lib*, *C6000\dsk6713\lib*, and *c6000\bios\lib*, respectively.

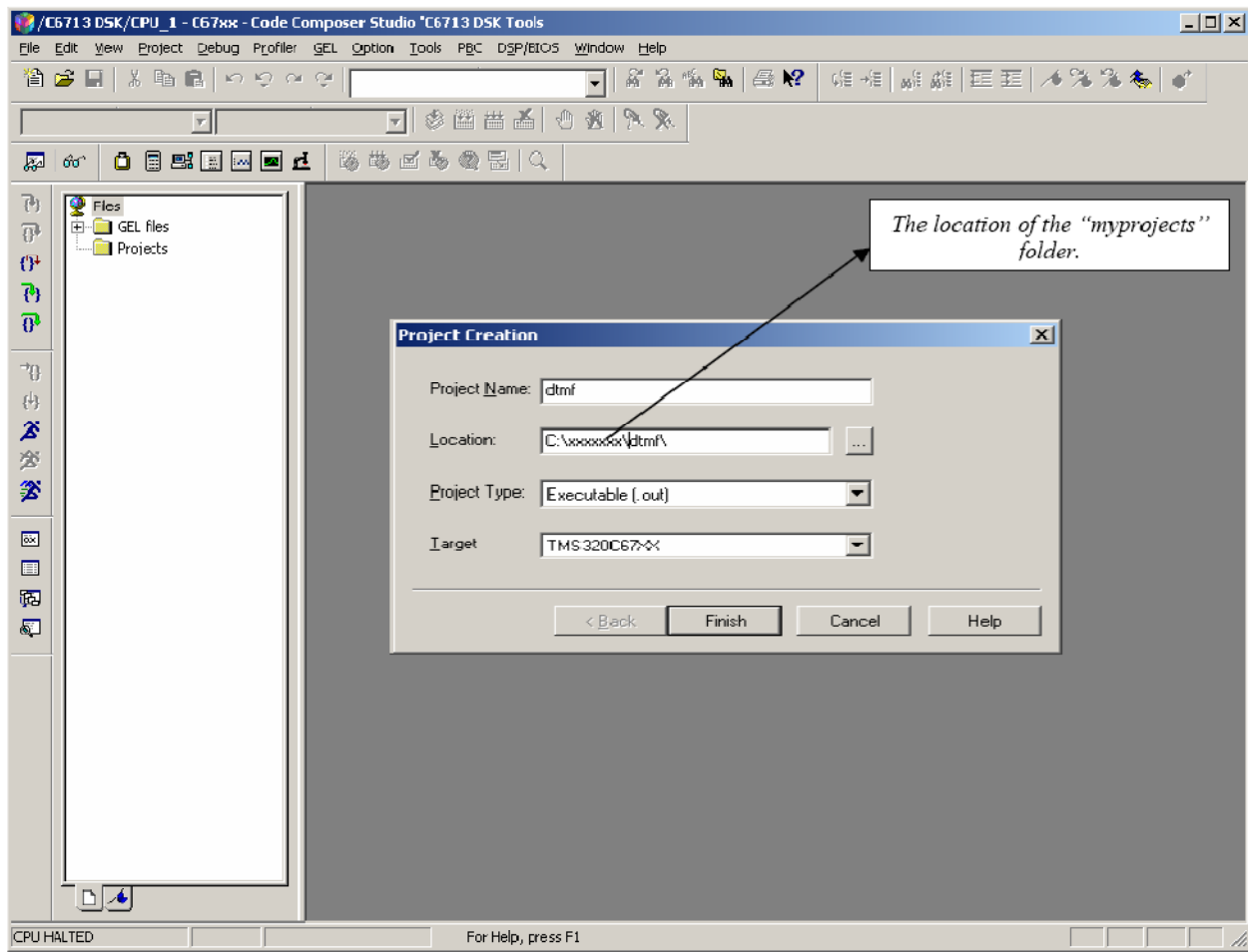
### 3.4. Steps involved in programming

#### 3.4.1 Connecting the C6713 DSK



## 3.4.2 Programming

- 1 Create a folder called “myprojects” on the desktop.



**Figure 3.4.2.1: Code composer Studio.**

- 2 Run the C6713 DSK Code Composer Studio (C6713 DSK CCS).
- 3 Go to Project ☺ New, create a project called dtmf.
- 4 Make sure the Project Type is Executable (.out) and Target is TMS320C67XX.
- 5 Download the **dtmf.cdb** file from <http://www.ece.mtu.edu/labs/EElabs/EE3306/resources>.
- 6 Once the project is created, go to Project ☺ Add files to project, add the DSK6713 Board support Library (BSL)file. This library will simplify the communication with the board using C language. The file is located at C:\ti\c6000\dsk6713\lib\dsk6713bsl.lib. More information about this library can be found in Help☺ TMS320C6713 DSK ☺ Software ☺ Board Support Library.

- 7 Next, include the “dtmf.cdb” file from the webpage. This configuration file will setup all the DSP/BIOS correctly for the use of this lab.
- 8 Finally, you can create a new file and begin the laboratory assignment. Remember to save the file as “dtmf.c” and include it to the project in order for it to run.

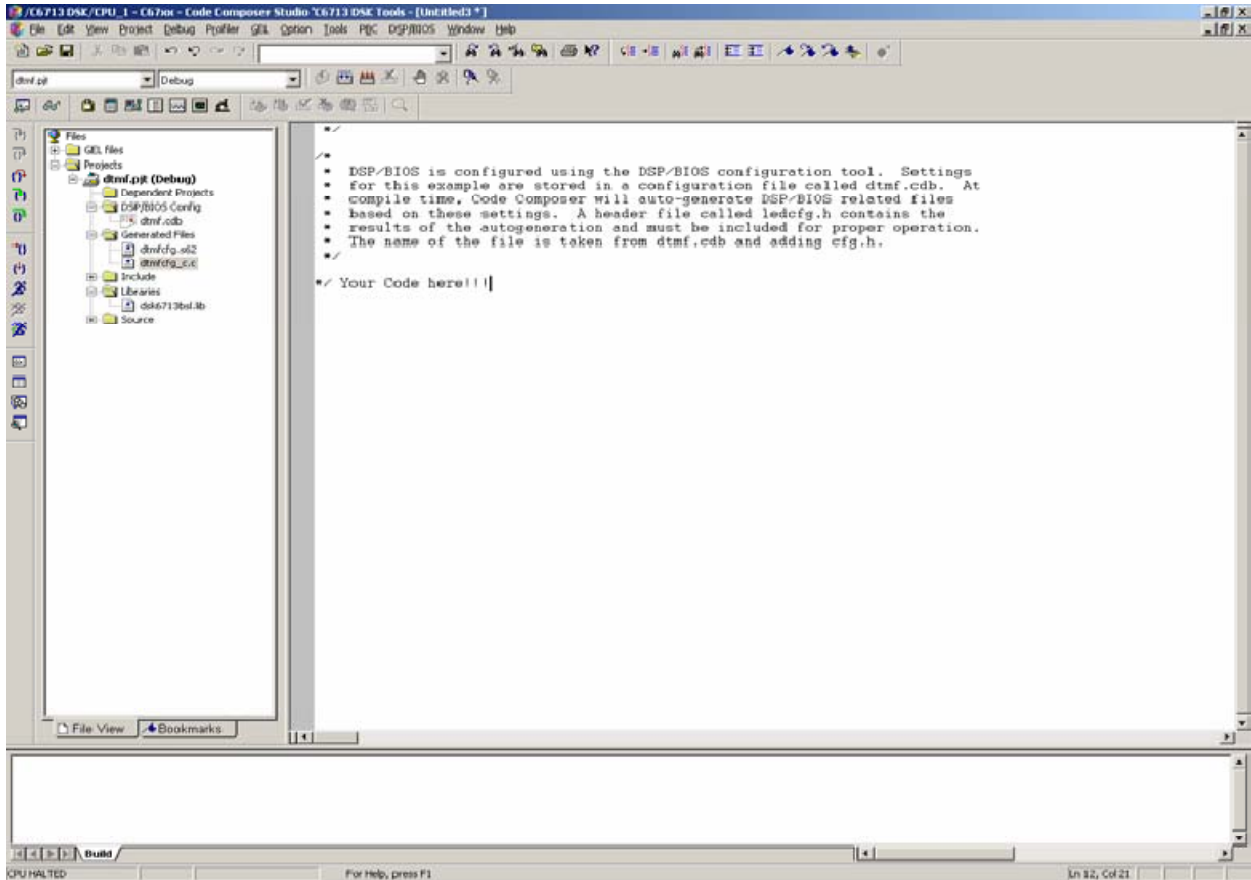



Figure 3.4.2.2: After including all necessary files.

### 3.4.3 Compiling into Machine Language

- 1 After writing the code, the next step is to compile the code to machine language. Go to Project  Build.
- 2 The Build command will compile all the files that are include in this project and make an executable file for the DSP.
- 3 Compiler results are shown at the bottom of the window.

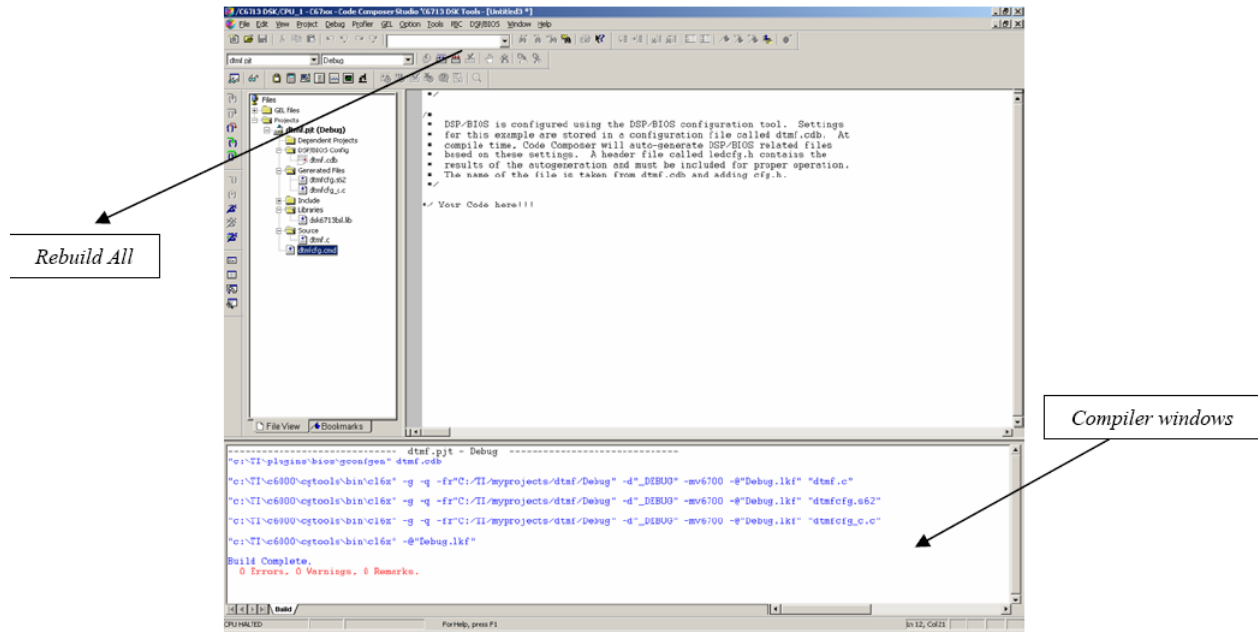


Figure 3.4.3.1: Compiler Comments.

### 3.4.4 Loading program into DSP Processor

- 1 Finally, to run the program, load the program into the DSP. Go to File □ Load Program. Load the executable file (.out) that the compiler generated (generally in the Debug directory of the project).

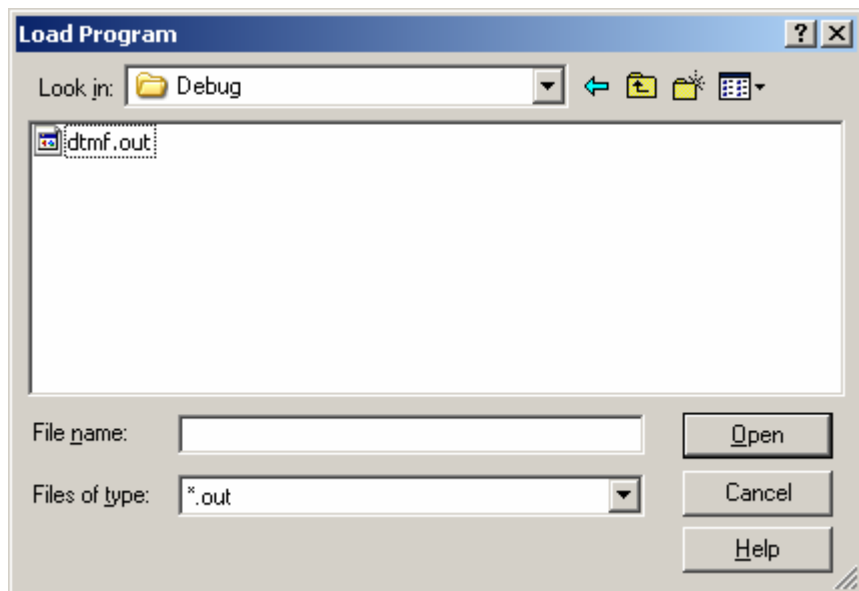


Figure 3.3.4.1: Load Program.

2 The run the file loaded into the DSP. Go to Debug  Run.

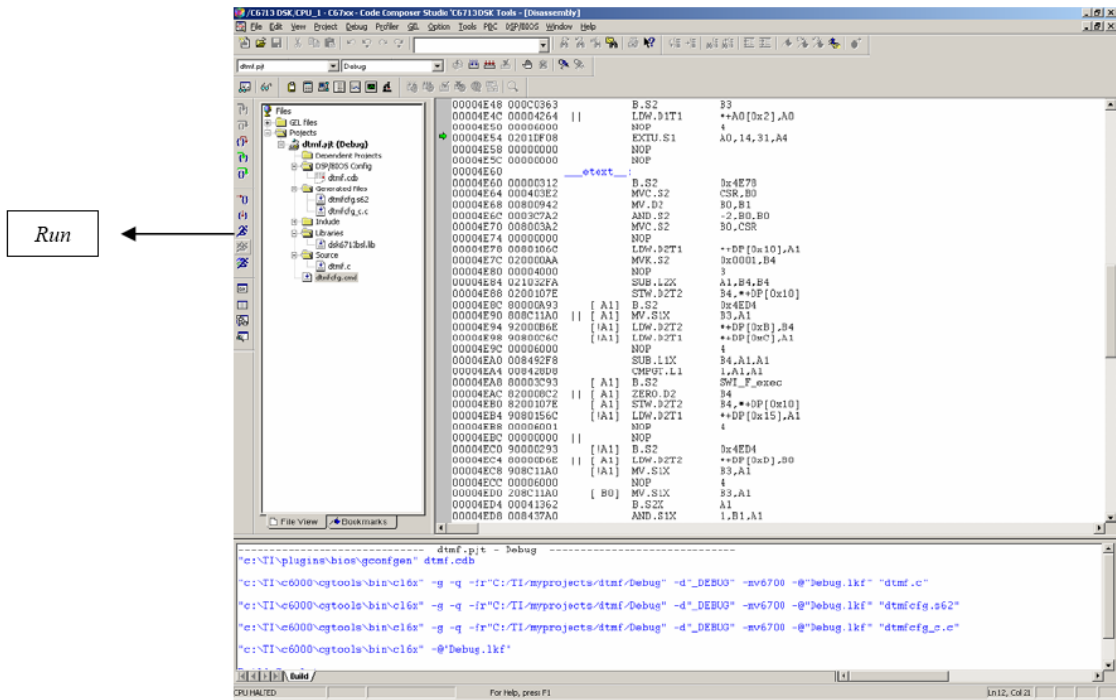


Figure 3.4.4.2: Program running.

### 3.5 Matlab Simulink Modelling

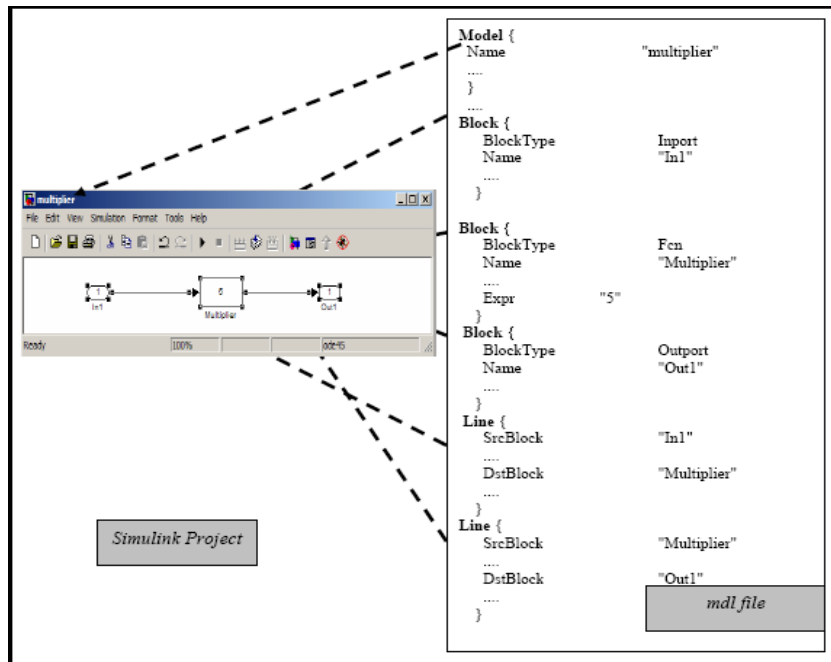


Fig 3.5.1 Dissection of model system in matlab coding

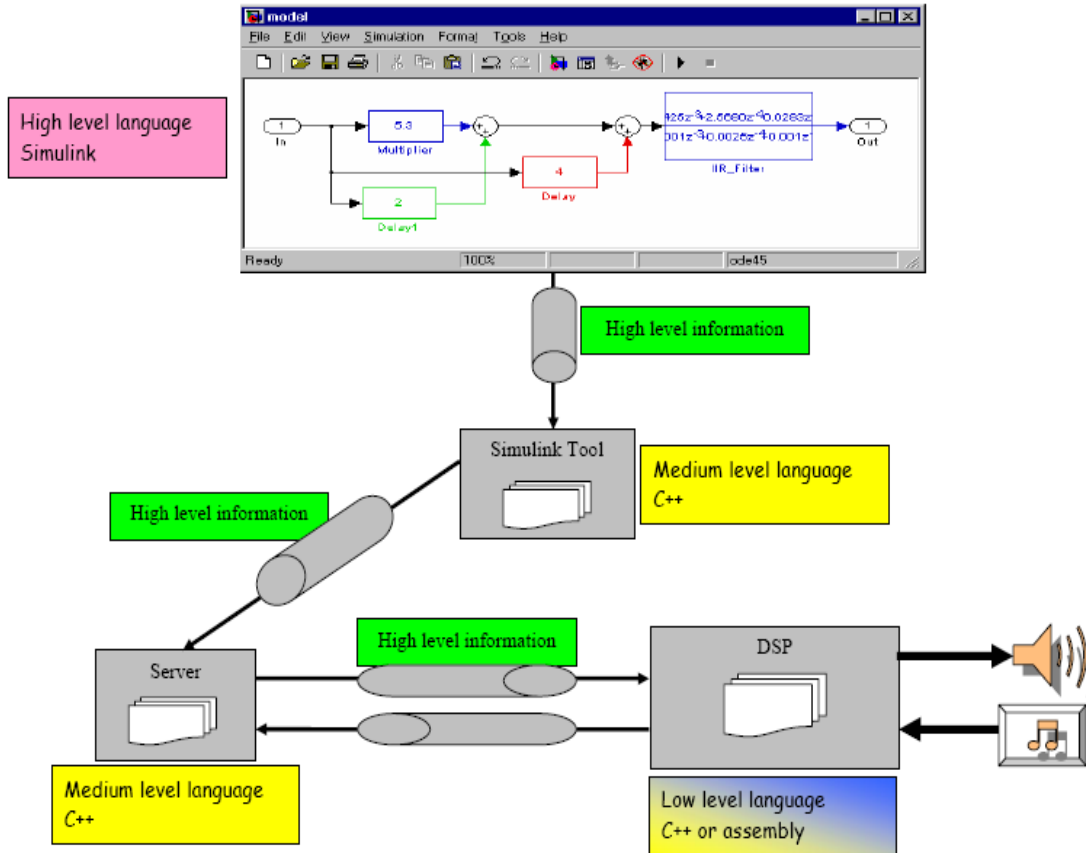


Fig 3.5.2 Steps in the modeling of a system using simulink

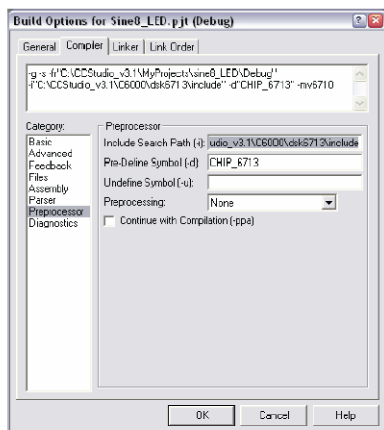
### 3.6 Problems faced during the implementation of Chassaing/kehtarnavaz examples

#### 3.6.1. During the compilation, the compiler can't find some header (.h) files

**Solution:-**

Add C:\CCStudio\_v3.1\CC6000\dsk6713\include to the search path

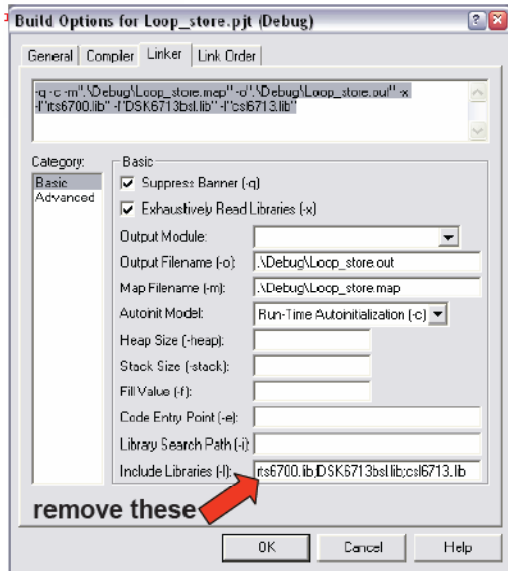
Project ->  
Build Options



### 3.6.2. During Compilation, the linker can't find some libraries ?

#### Solution :-

Remove hard links to libraries and add libraries and add libraries and manually to the project.



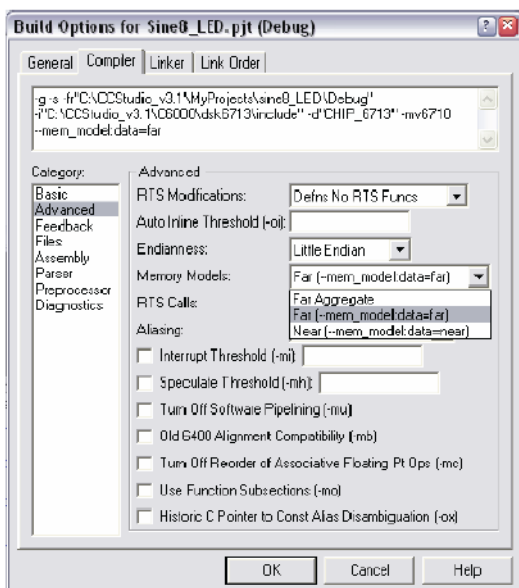
Problem is caused by a bad path for the include libraries in the linker options (Project -> Build Options -> Linker tab)

A fix for this is do remove rts6700.lib, DSK6713bsl.lib, and csl6713.lib from the linker options and add these files manually (Project -> Add files to Project...)

### 3.6.3. During Compilation, you get warnings about “far calls” to data ?

#### Solution :-

Set the memory model to “data = far”





# Chapter 4

## IMPLEMENTATION

## 4.1. Sinegraph

The main loop of the code writes each data point in the sine wave table out to the codec using the AIC23 codec package of the BSL. Each write function sends a single 16 bit sample to the codec. In this case the same data is sent out twice, once to the left channel and once to the right channel. The codec is configured to accept data at a rate of 48,000 stereo samples per second. Since the sine table is 48 entries long, the resulting output wave will be a 1KHz sine wave with the same output on both the left and right channels.

The serial port is used to transmit data to the codec at a much slower rate than the DSP can process data. It accepts data 16 bits at a time and shifts them out slowly one at a time. The write function returns a 1 if the write is completed successfully or a 0 if the serial channel is busy. The while() loop around the writes waits while the serial port is busy so program can be synchronized to the data rate of the codec.

Program :--

```
// sine graph . c
// The C6713 Board Support Library (BSL) has several
// modules , each of which has it ' s own include file .
// The file dsk6713 . h must be used in every program
// that uses the BSL. This example also includes
// dsk6713 led . h and dsk6713 dip . h because it uses
// the LED and DIP control on the board
#include " dsk6713 . h"
#include " dsk6713 aic23 . h"
#include " dsk6713 led . h"
#include " dsk6713 dip . h"
// table index
short loop = 0 ;
// gain factor
short gain = 10 ;
// output buffer
Int16 outbuffer [ 256 ]
// size of buffer
const short BUFFERLENGTH = 256;
// counter for buffer
int i = 0 ;
// Codec configuration
DSK6713 AIC23 Config config = { \ 0x0017 , /_ 0 DSK6713 AIC23 LEFTINVOL\ 0x0017 ,
/_ 1 DSK6713 AIC23 RIGHTINVOL\ 0x00d8 , /_ 2 DSK6713 AIC23 LEFTHPVOL\ 0x00d8 ,
```

```

/_ 3 DSK6713 AIC23 RIGHTHPVOL\ 0x0011 ,/_ 4 DSK6713 AIC23 ANAPATH\ 0x0000 ,/_
5 DSK6713 AIC23 DIGPATH\ 0x0000 ,/_ 6 DSK6713 AIC23 POWERDOWN\ 0x0043 ,/_ 7
DSK6713 AIC23 DIGIF\ 0x0081 ,/_ 8 DSK6713 AIC23 SAMPLERATE\ 0x0001 /_ 9
DSK6713 AIC23 DIGACT\
};
// Lookup table
Int16 sineTable[48] = { 0x0000, 0x10b4, 0x2120, 0x30fb, 0x3fff, 0x4dea,
0x5a81, 0x658b, 0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
0x7ffd, 0x7ee5, 0x7ba1, 0x76ef, 0x6ed8, 0x658b,
0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4,
0x0000, 0xef4c, 0xdee0, 0xcf06, 0xc002, 0xb216,
0xa57f, 0x9a75, 0x9128, 0x89c1, 0x845f, 0x811b,
0x8002, 0x811b, 0x845f, 0x89c1, 0x9128, 0x9a76,
0xa57f, 0xb216, 0xc002, 0xcf06, 0xdee0, 0xef4c
};
Uint32 fs = DSK6713 AIC23 FREQ 48KHZ;
// main() - Main code routine, initializes BSL and
// runs LED application
void main()
{ DSK6713 AIC23 CodecHandle hCodec;
// Initialize the board support library, must be first
// BSL call
DSK6713 init();
// Initialize the LED and DIP switch modules of the BSL
DSK6713 LED init();
DSK6713 DIP init();
// Start the codec
hCodec = DSK6713 AIC23 openCodec(0, &config);
// DIP Switch API
// DSK6713 DIP get ( Uint32 dipNum)
// Return value 0 Specified switch is off
// Return value 1 Specified switch is on
// infinite loop
while(1)
{ if(DSK6713 DIP get(0) == 0)
{// turn LED#0 on
DSK6713 LED on(0);
outputBuffer[i] = sineTable[loop];
// while(return value is not zero)
// see DSK6713 AIC23 write(...)
// send data to left channel
// output every Ts SW0

0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4,
0x0000, 0xef4c, 0xdee0, 0xcf06, 0xc002, 0xb216,
0xa57f, 0x9a75, 0x9128, 0x89c1, 0x845f, 0x811b,

```

```

0x8002 , 0x811b , 0 x845f , 0x89c1 , 0x9128 , 0x9a76 ,
0 xa57f , 0xb216 , 0xc002 , 0 xcf06 , 0xdee0 , 0 xef4c
};
Uint32 fs = DSK6713 AIC23 FREQ 48KHZ ;
// main () – Main code routine , initialize s BSL and
// runs LED application
void main ()
{ DSK6713 AIC23 CodecHandle hCodec ;
// Initialize the board support library , must be first
// BSL call
DSK6713 init () ;
// Initialize the LED and DIP switch modules of the BSL
DSK6713 LED init () ;
DSK6713 DIP init () ;
// Start the codec
hCodec = DSK6713 AIC23 openCodec ( 0 , &c on f i g ) ;
// DIP Switch API
// DSK6713 DIP get ( Uint32 dipNum)
// Return value 0 Specified switch is off
// Return value 1 Specified switch is on
// infinite loop
while ( 1 )
{ if ( DSK6713 DIP get ( 0 ) == 0 )
{ // turn LED#0 on
DSK6713 LED on ( 0 ) ;
outbuffer [ i ] = sine table [ loop ] ;
// while ( return value is not zero )
// see DSK6713 AIC23 write ( . . . )
// send data to left channel
// output every Ts SW0

```

The array `out_buffer[]` stores the sine data for plotting within CCS. The statement `while(1)` within the function `main` creates an infinite loop. When dip switch #0 is pressed, LED#0 turns on and the sinusoid is generated. The loop index is incremented until the end of the table is reached, after which it is re-initialized to zero. The following two commands are used to initialize and shut down the audio codec and are found at the beginning and end of all programs that use the BSL codec module.

```
DSK6713 openCodec ( )
```

returns a handle that is passed to each of the other codec functions.

```
hCodec = DSK6713 AIC23 openCodec ( 0 , &c o n f i g ) ;
```

opens the codec and

```
hCodec = DSK6713 AIC23 closeCodec ( 0 , &c o n f i g ) ;
```

closes the codec.

## 4.2. Create a Project in CCS

- 1 Type the code and save as sinegraph.c
- 2 Create a project in CCS (Project!New). Save the project as sinegraph.pjt
- 3 Add sinegraph.c to the project.
- 4 Add the required library files to project (ti/c6000/dsk6713/lib/).
- 5 Scan file dependencies (Project ! Scan file dependencies).
- 6 Set the appropriate compiler options (Project ! Build options). The following compiler options are suggested.
  - For the 'Basic' category:  
target version : C670x  
gen. debug info : full  
opt speed vs size: speed most critical  
prog level opt :none
  - For 'Feedback' category:  
interlisting : opt/c and ASM(-s)
  - For 'Preprocessor' category :  
Define symbols:CHIP\_6713

## 4.3. Building and running of the project

The project can now be built and run.

- 1 Select Project ! Rebuild All or press the toolbar with the three down arrows.This compiles and assembles the source file(s). The resulting object files are then linked with the library files. This creates an executable file sinegraph.out. that can be loaded into the C6713 processor and run. The building process causes all the dependent files to be included (in case one forgets to scan for all the file dependencies).

- 2 Select File ! Load Program in order to load sinegraph.outto the DSK. It should be in the folder sinegraph\Debug.Select Debug ! Run or use the toolbar with the running man. Connect a speaker to the LINE OUT connector on the DSK. Press the dip switch #0.

Plotting with CCS The output buffer is updated continuously every 256 points .CCS can be used to plot the current output data stored in the buffer out\_buffer.

- 1 Select View!Graph!Time/Frequency. Change the Graph Property Dialog so that the options are as indicated in figure 3. The starting address of the output buffer is out\_buffer.The other options can be left as default.
- 2 Choose a fast Fourier transform (FFT) order so that the frame size is 2 order. Press OK and verify that the FFT magnitude plot is as shown (figure 4) ).The spike at 1000 Hz represents the frequency of the sinusoid generated.

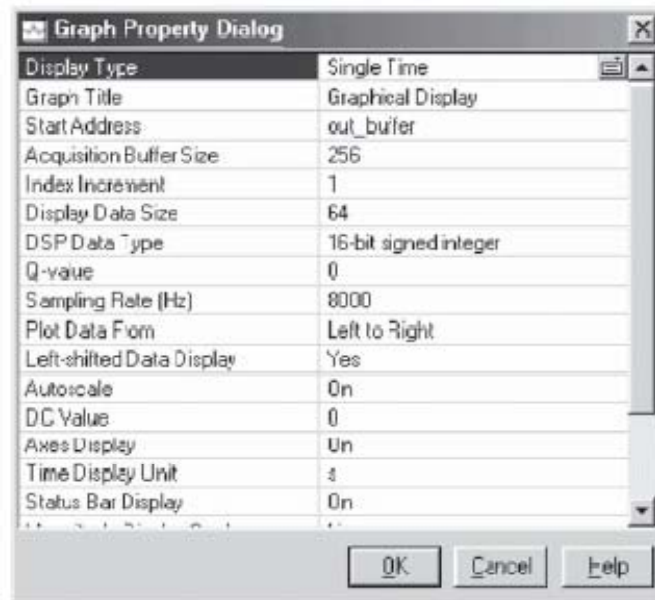


Figure 4.3.1: Time domain graph property dialog

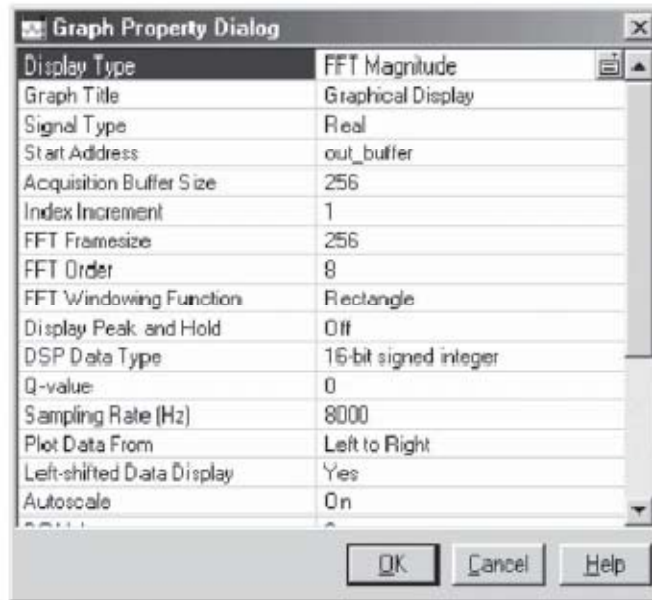


Figure 4.3.2: Frequency domain graph property dialog

# Chapter 5

CONCLUSION



There are many applications for which the Digital Signal Processor becomes an ideal choice as they provide the best possible combination of performance, power and cost. Most of the DSP applications can be simplified into multiplications and additions, so the MAC formed a main functional unit in early DSP processors. The designers later incorporated more features, like pipelining, SIMD, VLIW etc, in the processors to deliver improved performance.

There has been a drive to develop new benchmarking schemes as the improvement in the processor architecture made the earlier benchmarking schemes, obsolete and less reliable.

Power issues are gaining importance as DSP processors are incorporated in to handheld, mobile and portable devices. This leads to development of an important class of DSP processors namely fixed-point processors.

Based on the current trends seen in the DSP processor development we may predict that the manufacturers will follow the path of general purpose processors. With new IC manufacturing technologies available we may expect to see more on-chip peripherals and memory; and in fact the system on chip may not be too far away.

## References :-

- [1] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing, Second Edition*, California Technical Publishing, 1999.
- [2] Berkeley Design Technology, Inc., "The Evolution of DSP Processors", World Wide Web, <http://www.bdti.com/articles/evolution.pdf>, Nov. 2006.
- [3] Berkeley Design Technology, Inc., "Choosing a Processor: Benchmark and Beyond", World Wide Web, [http://www.bdti.com/articles/20060301\\_TIDC\\_Choosing.pdf](http://www.bdti.com/articles/20060301_TIDC_Choosing.pdf), Nov. 2006.
- [4] University of Rochester, "DSP Architectures: Past, Present and Future", World Wide Web, [http://www.ece.rochester.edu/research/wcng/papers/CAN\\_r1.pdf](http://www.ece.rochester.edu/research/wcng/papers/CAN_r1.pdf), Nov. 2006.
- [5] Gene Frantz, "Digital Signal Processor Trends", *Proceedings of the IEEE Micro*, Vol. 20, No. 6, 2000, pp. 52-59.
- [6] Texas Instruments, *TMS320VC5510/5510A, Fixed-Point Digital Signal Processors, Data Manual*, Dallas, TX, July 2006.
- [7] Texas Instruments, *TMS320C62X/C67X, Programmers' Guide*, Dallas, TX, May 1999.
- [8] Texas Instruments, *TMS320C6000, Peripherals, Reference Guide*, Dallas, TX, March 2001.
- [9] Texas Instruments, Inc *TMS320C55x, Technical Overview*, Dallas, TX, Feb. 2000.
- [10] Texas Instruments, *TMS320C6713B, Floating-Point Digital Signal Processors, Data Sheet*, Dallas, TX, June 2006.
- [11] Texas Instruments, *TMS320C55x DSP Peripherals Overview Reference Guide*, Dallas, TX, April 2006.