DSP Applications Using C and the TMS320C6x DSK

Rulph Chassaing
DSP Applications Using C and the TMS320C6x DSK
TOPICS IN DIGITAL SIGNAL PROCESSING

C. S. BURRUS and T. W. PARKS: DFT/FFT AND CONVOLUTION ALGORITHMS: THEORY AND IMPLEMENTATION

JOHN R. TREICHLER, C. RICHARD JOHNSON, JR., and MICHAEL G. LARIMORE: THEORY AND DESIGN OF ADAPTIVE FILTERS

T. W. PARKS and C. S. BURRUS: DIGITAL FILTER DESIGN

RULPH CHASSAING and DARRELL W. HORNING: DIGITAL SIGNAL PROCESSING WITH THE TMS320C25

RULPH CHASSAING: DIGITAL SIGNAL PROCESSING WITH C AND THE TMS320C30

RULPH CHASSAING: DIGITAL SIGNAL PROCESSING LABORATORY EXPERIMENTS USING C AND THE TMS320C31 DSK

RULPH CHASSAING: DSP APPLICATIONS USING C AND THE TMS320C6x DSK
DSP Applications Using C and the TMS320C6x DSK

Ralph Chassaing
Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Copyright © 2002 by John Wiley & Sons, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic or mechanical, including uploading, downloading, printing, decompiling, recording or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ@WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

This title is also available in print as ISBN 0-471-20754-3.

For more information about Wiley products, visit our web site at www.Wiley.com.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xi</td>
</tr>
<tr>
<td>List of Examples</td>
<td>xv</td>
</tr>
<tr>
<td>Programs/Files on Accompanying Disk</td>
<td>xix</td>
</tr>
<tr>
<td>1 DSP Development System</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 DSK Support Tools</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 DSK Board</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 TMS320C6711 Digital Signal Processor</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Code Composer Studio</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1 CCS Installation and Support</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2 Useful Types of Files</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Programming Examples to Test the DSK Tools</td>
<td>7</td>
</tr>
<tr>
<td>1.4.1 Quick Test of DSK</td>
<td>7</td>
</tr>
<tr>
<td>1.4.2 Support Files</td>
<td>8</td>
</tr>
<tr>
<td>1.4.3 Examples</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Support Programs/Files Considerations</td>
<td>24</td>
</tr>
<tr>
<td>1.5.1 Initialization/Communication File</td>
<td>24</td>
</tr>
<tr>
<td>1.5.2 Vector File</td>
<td>26</td>
</tr>
<tr>
<td>1.5.3 Linker File</td>
<td>26</td>
</tr>
<tr>
<td>1.6 Compiler/Assembler/Linker Shell</td>
<td>26</td>
</tr>
<tr>
<td>1.6.1 Compiler</td>
<td>28</td>
</tr>
<tr>
<td>1.6.2 Assembler</td>
<td>29</td>
</tr>
<tr>
<td>1.6.3 Linker</td>
<td>29</td>
</tr>
<tr>
<td>References</td>
<td>30</td>
</tr>
</tbody>
</table>


4 Finite Impulse Response Filters

4.1 Introduction to the \( z \)-Transform 102
  4.1.1 Mapping from \( s \)-Plane to \( z \)-Plane 105
  4.1.2 Difference Equations 106
4.2 Discrete Signals 107
4.3 Finite Impulse Response Filters 108
4.4 FIR Implementation Using Fourier Series 110
4.5 Window Functions 114
  4.5.1 Hamming Window 115
  4.5.2 Hanning Window 115
  4.5.3 Blackman Window 115
  4.5.4 Kaiser Window 116
  4.5.5 Computer-Aided Approximation 116
4.6 Programming Examples Using C and ASM Code 116
References 155

5 Infinite Impulse Response Filters

5.1 Introduction 159
5.2 IIR Filter Structures 160
  5.2.1 Direct Form I Structure 160
  5.2.2 Direct Form II Structure 161
  5.2.3 Direct Form II Transpose 163
  5.2.4 Cascade Structure 164
## 5.2.5 Parallel Form Structure 165

## 5.3 Bilinear Transformation 167

5.3.1 Bilinear Transformation Design Procedure 168

## 5.4 Programming Examples Using C Code 169

References 181

## 6 Fast Fourier Transform 182

6.1 Introduction 182

6.2 Development of the FFT Algorithm with Radix-2 183

6.3 Decimation-in-Frequency FFT Algorithm with Radix-2 184

6.4 Decimation-in-Time FFT Algorithm with Radix-2 191

6.5 Bit Reversal for Unscrambling 195

6.6 Development of the FFT Algorithm with Radix-4 195

6.7 Inverse Fast Fourier Transform 198

6.8 Programming Examples 199

6.8.1 Fast Convolution 206

References 214

## 7 Adaptive Filters 216

7.1 Introduction 216

7.2 Adaptive Structures 218

7.3 Programming Examples for Noise Cancellation and System Identification 221

References 237

## 8 Code Optimization 239

8.1 Introduction 239

8.2 Optimization Steps 240

8.2.1 Compiler Options 240

8.2.2 Intrinsic C Functions 241

8.3 Procedure for Code Optimization 241

8.4 Programming Examples Using Code Optimization Techniques 241

8.5 Software Pipelining for Code Optimization 248

8.5.1 Procedure for Hand-Coded Software Pipelining 249

8.5.2 Dependency Graph 249

8.5.3 Scheduling Table 251

8.6 Execution Cycles for Different Optimization Schemes 258

References 259
9 DSP Applications and Student Projects

9.1 Voice Scrambler Using DMA and User Switches 260
9.2 Phase-Locked Loop 261
  9.2.1 RTDX for Real-Time Data Transfer 263
9.3 SB-ADPCM Encoder/Decoder: Implementation of G.722 Audio Coding 263
9.4 Adaptive Temporal Attenuator 264
9.5 Image Processing 265
9.6 Filter Design and Implementation Using a Modified Prony's Method 266
9.7 FSK Modem 266
9.8 μ-Law for Speech Companding 267
9.9 Voice Detection and Reverse Playback 268
9.10 Miscellaneous Projects 268
  9.10.1 Acoustic Direction Tracker 268
  9.10.2 Multirate Filter 269
  9.10.3 Neural Network for Signal Recognition 270
  9.10.4 PID Controller 270
  9.10.5 Four-Channel Multiplexer for Fast Data Acquisition 270
  9.10.6 Video Line Rate Analysis 270

References 272

Appendix A TMS320C6x Instruction Set 276
A.1 Instructions for Fixed- and Floating-Point Operations 276
A.2 Instructions for Floating-Point Operations 276
References 276

Appendix B Registers for Circular Addressing and Interrupts 278
Reference 278

Appendix C Fixed-Point Considerations 281
C.1 Binary and Two’s-Complement Representation 281
C.2 Fractional Fixed-Point Representation 284
C.3 Multiplication 285
Reference 287

Appendix D Matlab Support Tools 288
D.1 MATLAB GUI Filter Designer SPTOOL for FIR Filter Design 288
D.2 MATLAB GUI Filter Designer SPTOOL for IIR Filter Design 290
**Appendix E  Additional Support Tools**

E.1 Goldwave Shareware Utility as Virtual Instrument 303
E.2 Filter Design Using DigiFilter 304
   E.2.1 FIR Filter Design 304
   E.2.2 IIR Filter Design 305
E.3 FIR Filter Design Using Filter Development Package 306
   E.3.1 Kaiser Window 306
   E.3.2 Hamming Window 306
E.4 Visual Application Builder 306
E.5 Miscellaneous Support 308
References 309

**Appendix F  Input and Output with PCM3003 Stereo Codec**

F.1 PCM3003 Audio Daughter Card 310
F.2 Programming Examples Using the PCM3003 Stereo Codec 315
References 324

**Appendix G  DSP/BIOS and RTDX for Real-Time Data Transfer**

References 327

Index 329
Digital signal processors, such as the TMS320 family of processors, are used in a wide range of applications, such as in communications, controls, speech processing, and so on. They are used in fax transmission, modems, cellular phones, and other devices. These devices have also found their way into the university classroom, where they provide an economical way to introduce real-time digital signal processing (DSP) to the student.

Texas Instruments recently introduced the TM320C6x processor, based on the very-long-instruction-word (VLIW) architecture. This newer architecture supports features that facilitate the development of efficient high-level language compilers. Throughout the book we refer to the C/C++ language simply as C. Although TMS320C6x/assembly language can produce fast code, problems with documentation and maintenance may exist. With the available C compiler, the programmer must consider to “let the tools do the work.” After that, if the programmer is not satisfied, Chapters 3 and 8 and the last few examples in Chapter 4 can be very useful.

This book is intended primarily for senior undergraduate and first-year graduate students in electrical and computer engineering and as a tutorial for the practicing engineer. It is written with the conviction that the principles of DSP can best be learned through interaction in a laboratory setting, where students can appreciate the concepts of DSP through real-time implementation of experiments and projects. The background assumed is a course in linear systems and some knowledge of C.

Most chapters begin with a theoretical discussion, followed by representative examples that provide the necessary background to perform the concluding experiments. There are a total of 76 solved programming examples, most using C code, with a few in assembly and linear assembly code. A list of these examples appears on page xv. Several sample projects are also discussed.
Programming examples are included throughout the text. This can be useful to the reader who is familiar with both DSP and C programming but who is not necessarily an expert in both.

This book can be used in the following ways:

1. For a DSP course with a laboratory component, using Chapters 1 to 7 and Appendices D to F. If needed, the book can be supplemented with some additional theoretical materials, since the book’s emphasis is on the practical aspects of DSP. It is possible to cover Chapter 7 on adaptive filtering, following Chapter 4 on FIR filtering (since there is only one example in Chapter 7 that uses material from Chapter 5). It is my conviction that adaptive filtering (Chapter 7) should be incorporated into an undergraduate course in DSP.

2. For a laboratory course using many of the examples and experiments from Chapters 1 to 7. The beginning of the semester can be devoted to short programming examples and experiments and the remainder of the semester used for a final project.

3. For a senior undergraduate or first-year graduate design project course, using Chapters 1 to 5, selected materials from Chapters 6 to 9, and Appendices D to F.

4. For the practicing engineer as a tutorial, and for workshops and seminars, using selected materials throughout the book.

In Chapter 1 we introduce the tools through three programming examples. These tools include the powerful Code Composer Studio (CCS) provided with the TMS320C6711 DSP starter kit (DSK). It is essential to perform these three examples before proceeding to subsequent chapters. They illustrate the capabilities of CCS for debugging, plotting in both the time and frequency domains, and other matters.

In Chapter 2 we illustrate input and output (I/O) with the codec on the DSK board through many programming examples. Alternative I/O with a stereo audio codec that interfaces with the DSK is described. Chapter 3 covers the architecture and the instructions available for the TMS320C6x processor. Special instructions and assembler directives that are useful in DSP are discussed. Programming examples using both assembly and linear assembly are included in this chapter.

In Chapter 4 we introduce the z-transform and discuss finite impulse response (FIR) filters and the effect of window functions on these filters. Chapter 5 covers infinite impulse response (IIR) filters. Programming examples to implement real-time FIR and IIR filters are included.

Chapter 6 covers the development of the fast Fourier transform (FFT). Programming examples on FFT are included. In Chapter 7 we demonstrate the usefulness of the adaptive filter for a number of applications with least mean squares (LMS). Programming examples are included to illustrate the gradual cancellation of noise or system identification. Chapter 8 illustrates techniques for code opti-
mization. In Chapter 9 we discuss a number of DSP applications and student projects.

A disk included with this book contains all the programs discussed. See page xix for a list of the folders that contain the support files for all the examples.

Over the last six years, faculty members from over 150 institutions have taken my “DSP and Applications” workshops. These workshops were supported for three years by grants from the National Science Foundation (NSF) and subsequently, by Texas Instruments. I am thankful to NSF, Texas Instruments, and the participating faculty members for their encouragement and feedback. I am grateful to Dr. Donald Reay of Heriot-Watt University, who contributed several examples during his review of the book. I appreciate the many suggestions made by Dr. Robert Kubichek of the University of Wyoming during his review of the book. I also thank Dr. Darrell Horning of the University of New Haven, with whom I coauthored the text Digital Signal Processing with the TMS320C25, for introducing me to “book writing.” I thank all the students (at Roger Williams University, University of Massachusetts, Dartmouth, and Worcester Polytechnic Institute) who have taken my real-time DSP and senior design project courses, based on the TMS320 processors, over the last 16 years. I am particularly indebted to two former students, Bill Bitler and Peter Martin, who have worked with me over the years. The laboratory assistance of Walter J. Gomes III in several workshops and during the development of many examples has been invaluable. The continued support of many people from Texas Instruments is also very much appreciated: Maria Ho and Christina Peterson, in particular, have been very supportive of this book. I would be remiss if I did not mention the librarians in Herkimer, New York (where I was stranded for two weeks) for the use of their facility to write Chapter 8.

RULPH CHASSAING

Chassaing@msn.com
List of Examples

1.1 Sine Generation with Eight Points 8
1.2 Generation of Sinusoid and Plotting with CCS 17
1.3 Dot Product of Two Arrays 19
2.1 Loop Program Using Interrupt 38
2.2 Loop Program Using Polling 39
2.3 Sine Generation Using Polling 40
2.4 Sine Generation With Two Sliders for Amplitude and Frequency Control 42
2.5 Loop Program with Input Data Stored in Memory Buffer 43
2.6 Loop with Data in Buffer Printed to File 44
2.7 Square-Wave Generation Using Lookup Table 47
2.8 Ramp Generation Using Lookup Table 48
2.9 Ramp Generation without a Lookup Table 48
2.10 Echo 49
2.11 Echo Using Two Interrupts with Control for Different Effects 51
2.12 Sine Generation with Table Values Generated within Program 53
2.13 Sine Generation with Table Created by MATLAB 53
2.14 Amplitude Modulation 56
2.15 Sweep Sinusoid Using Table with 8000 Points 57
2.16 Pseudorandom Noise Sequence Generation 59
3.1 Efficient Dot Product 91
3.2 Sum of $n + (n - 1) + (n - 2) + \ldots + 1$ Using C Calling Assembly Function 92
List of Examples

3.3 Factorial of a Number Using C Program Calling Assembly Function
3.4 Dot Product Using Assembly Program Calling Assembly Function
3.5 Dot Product Using C Function Calling Linear Assembly Function
3.6 Factorial Using C Calling a Linear Assembly Function
4.1 FIR Filter Implementation: Bandstop and Bandpass
4.2 Effects on Voice Using Three FIR Lowpass Filters
4.3 Implementation of Four Different Filters: Lowpass, Highpass, Bandpass, and Bandstop
4.4 FIR Implementation with Pseudorandom Noise Sequence as Input to Filter
4.5 FIR Filter with Frequency Response Plot Using CCS
4.6 FIR Filter with Internally Generated Pseudorandom Noise as Input to Filter and Output Stored in Memory
4.7 Two Notch Filters to Recover Corrupted Input Voice
4.8 FIR Implementation Using Four Different Methods
4.9 Voice Scrambler Using Filtering and Modulation
4.10 Illustration of Aliasing Effects with Down-Sampling
4.11 Implementation of an Inverse FIR Filter
4.12 FIR Implementation Using C Calling ASM Function
4.13 FIR Implementation Using C Calling Faster ASM Function
4.14 FIR Implementation with C Program Calling ASM Function Using Circular Buffer
4.15 FIR Implementation with C Program Calling ASM Function Using Circular Buffer in External Memory
5.1 IIR Filter Implementation Using Second-Order Stages in Cascade
5.2 Generation of Two Tones Using Two Second-Order Difference Equations
5.3 Sine Generation Using a Difference Equation
5.4 Generation of a Swept Sinusoid Using a Difference Equation
5.5 IIR Inverse Filter
6.1 DFT of a Sequence of Real Numbers with Output from CCS Window
6.2 FFT of a Real-Time Input Signal Using an FFT Function in C
6.3 FFT of a Sinusoidal Signal from a Table Using TI's C Callable FFT Function
6.4 Fast Convolution With Overlap-Add for FIR Implementation Using TI's Floating-Point FFT Functions
6.5 Graphic Equalizer
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Adaptive Filter Using C Code Compiled with Borland C/C++</td>
<td>221</td>
</tr>
<tr>
<td>7.2</td>
<td>Adaptive Filter for Noise Cancellation</td>
<td>224</td>
</tr>
<tr>
<td>7.3</td>
<td>Adaptive FIR Filter for System ID of Fixed FIR</td>
<td>227</td>
</tr>
<tr>
<td>7.4</td>
<td>Adaptive FIR for System ID of Fixed FIR with Weights of Adaptive Filter Initialized as FIR Bandpass</td>
<td>227</td>
</tr>
<tr>
<td>7.5</td>
<td>Adaptive FIR for System ID of Fixed IIR</td>
<td>232</td>
</tr>
<tr>
<td>7.6</td>
<td>Adaptive Predictor for Cancellation of Narrowband Interference Added to Desired Wideband Signal</td>
<td>232</td>
</tr>
<tr>
<td>8.1</td>
<td>Sum of Products With Word-Wide Data Access for Fixed-Point Implementation Using C Code</td>
<td>242</td>
</tr>
<tr>
<td>8.2</td>
<td>Separate Sum of Products With C Intrinsic Functions Using C Code</td>
<td>243</td>
</tr>
<tr>
<td>8.3</td>
<td>Sum of Products With Word-Wide Access for Fixed-Point Implementation Using Linear ASM Code</td>
<td>243</td>
</tr>
<tr>
<td>8.4</td>
<td>Sum of Products with Double-Word Load for Floating-Point Implementation Using Linear ASM Code</td>
<td>244</td>
</tr>
<tr>
<td>8.5</td>
<td>Dot Product with No Parallel Instructions for Fixed-Point Implementation Using ASM Code</td>
<td>244</td>
</tr>
<tr>
<td>8.6</td>
<td>Dot Product with Parallel Instructions for Fixed-Point Implementation Using ASM Code</td>
<td>245</td>
</tr>
<tr>
<td>8.7</td>
<td>Two Sums of Products with Word-Wide (32-bit) Data for Fixed-Point Implementation Using ASM Code</td>
<td>245</td>
</tr>
<tr>
<td>8.8</td>
<td>Dot Product with No Parallel Instructions for Floating-Point Implementation Using ASM Code</td>
<td>246</td>
</tr>
<tr>
<td>8.9</td>
<td>Dot Product with Parallel Instructions for Floating-Point Implementation Using ASM Code</td>
<td>246</td>
</tr>
<tr>
<td>8.10</td>
<td>Two Sums of Products With Double-Word-Wide (64-bit) Data for Floating-Point Implementation Using ASM Code</td>
<td>247</td>
</tr>
<tr>
<td>8.11</td>
<td>Dot Product Using Software Pipelining for a Fixed-Point Implementation</td>
<td>252</td>
</tr>
<tr>
<td>8.12</td>
<td>Dot Product Using Software Pipelining for a Floating-Point Implementation</td>
<td>253</td>
</tr>
<tr>
<td>D.1</td>
<td>MATLAB GUI Filter Designer SPTOOL for FIR Filter Design</td>
<td>288</td>
</tr>
<tr>
<td>D.2</td>
<td>MATLAB GUI Filter Designer SPTOOL for IIR Filter Design</td>
<td>290</td>
</tr>
<tr>
<td>D.3</td>
<td>FIR Filter Design Using MATLAB’s Student Version</td>
<td>292</td>
</tr>
<tr>
<td>D.4</td>
<td>Multiband FIR Filter Design Using MATLAB</td>
<td>293</td>
</tr>
<tr>
<td>D.5</td>
<td>IIR Filter Design Using MATLAB’s Student Version</td>
<td>294</td>
</tr>
<tr>
<td>F.1</td>
<td>Loop Program Using Polling with the PCM3003 Stereo Codec</td>
<td>315</td>
</tr>
<tr>
<td>F.2</td>
<td>Loop Program Using Interrupt with the PCM3003 Codec</td>
<td>317</td>
</tr>
<tr>
<td>F.3</td>
<td>FIR Filter Implementation Using the PCM3003 Codec</td>
<td>319</td>
</tr>
</tbody>
</table>
List of Examples

F.4 Adaptive FIR Filter for Noise Cancellation Using the PCM3003 Codec 319
F.5 Adaptive Predictor for Cancellation of Narrowband Interference Added to Desired Wideband Signal, Using the PCM3003 Codec 324
A list of the folders included on the accompanying disk is shown below. The folders contain the programs/files for all the examples/projects covered in the book.
DSP Applications Using C and the TMS320C6x DSK
Chapter 1 introduces several tools available for digital signal processing (DSP). These tools include the popular Code Composer Studio (CCS), which provides an integrated development environment (IDE); the DSP starter kit (DSK) with the TMS320C6711 floating-point processor onboard and complete support for input and output. Three examples are included to test both the software and hardware tools included with the DSK.

1.1 INTRODUCTION

Digital signal processors such as the TMS320C6x (C6x) family of processors are like fast special-purpose microprocessors with a specialized type of architecture and 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. They are found in cellular phones, fax/modems, disk drives, radio, and so on. These processors have become the product 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. Real-time processing means that the processing must 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. While 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 such as temperature. DSP processors enjoy the advantages of microprocessors. They are easy to use, flexible, and economical.

A number of books and articles have been published that address the importance of digital signal processors for a number of applications [1–20]. Various technologies have been used for real-time processing, from fiber optics for very high frequency to DSP processors very suitable for the audio-frequency range. Common applications using these processors have been for frequencies from 0 to 20kHz. Speech can be sampled at 8kHz (how quickly samples are acquired), which implies that each value sampled is acquired at a rate of 1/(8kHz) or 0.125ms. A commonly used sample rate of a compact disk is 44.1kHz. A/D-based boards in the megahertz sampling rate range are currently available.

The basic system consists of an analog-to-digital converter (ADC) to capture an input signal. The resulting digital representation of the captured signal is then processed by a digital signal processor such as the C6x and then output through a digital-to-analog converter (DAC). Also included within the basic system is a special input filter for antialiasing to eliminate erroneous signals, and an output filter to smooth or reconstruct the processed output signal.

1.2 DSK SUPPORT TOOLS

Most of the work presented in this book involves the design of a program to implement a DSP application. To perform the experiments, the following tools are used:

1. **TI’s DSP starter kit (DSK).** The DSK package includes:
   
   (a) Code Composer Studio (CCS), which provides the necessary software support tools. CCS provides an integrated development environment (IDE), bringing together the C compiler, assembler, linker, debugger, and so on.
   
   (b) A board, shown in Figure 1.1a, that contains the TMS320C6711 (C6711) floating-point digital signal processor as well as a 16-bit codec for input and output (I/O) support.
   
   (c) A parallel cable (DB25) that connects the DSK board to a PC.
   
   (d) A power supply for the DSK board.
2. An IBM-compatible PC. The DSK board connects to the parallel port of the PC through the DB25 cable included with the DSK package.

3. An oscilloscope, signal generator, and speakers. A signal/spectrum analyzer is optional. Shareware utilities are available that utilize the PC and a sound card to create a virtual instrument such as an oscilloscope, a function generator, or a spectrum analyzer.

FIGURE 1.1. TMS320C6711-based DSK board: (a) board; (b) diagram (Courtesy of Texas Instruments).
All the files/programs listed and discussed in this book (except the student project files in Chapter 9) are included on the accompanying disk. Most of the examples can also run on the fixed-point C6211-based DSK (which has been discontinued). A list of all the examples is given on pages xv–xviii.

1.2.1 DSK Board

The DSK package is powerful, yet relatively inexpensive ($295), with the necessary hardware and software support tools for real-time signal processing [21–33]. It is a complete DSP system. The DSK board, with an approximate dimension of $5 \times 8$ inches, includes the C6711 floating-point digital signal processor [22] and a 16-bit codec AD535 for input and output.

The onboard codec AD535 [34] uses a sigma–delta technology that provides analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC). A 4-MHz clock onboard the DSK connects to this codec to provide a fixed sampling rate of 8 kHz.

A daughter card expansion is also provided on the DSK board. We will illustrate input and output by plugging an audio daughter card based on the PCM3003 stereo codec (not included with the DSK package) into an 80-pin connector on the DSK board. The audio daughter card is available from Texas Instruments and is described in Appendix F. The PCM3003 codec has variable sample rates up to 72 kHz and can be useful for applications requiring higher sampling rates and two accessible input and output channels.

The DSK board includes 16 MB (megabytes) of synchronous dynamic RAM (SDRAM) and 128 kB (kilobytes) of flash ROM. Two connectors on the board provide input and output and are labeled IN (J7) and OUT (J6), respectively. Three of the four user dip switches on the DSK board can be read from a program (a project example on voice scrambling makes use of these switches). The onboard clock is 150 MHz. Also onboard the DSK are voltage regulators that provide 1.8 V for the C6711 core and 3.3 V for its memory and peripherals.

1.2.2 TMS320C6711 Digital Signal Processor

The TMS320C6711 (C6711) is based on the very-long-instruction-word (VLIW) architecture, which is very well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. For example, with a clock rate of 150 MHz, the C6711 is capable of fetching eight 32-bit instructions every 1/(150 MHz) or 6.66 ns.

Features of the C6711 include 72 kB of internal memory, eight functional or execution units composed of six ALUs and two multiplier units, a 32-bit address bus to address 4 GB (gigabytes), and two sets of 32-bit general-purpose registers.

The C67xx (such as the C6701 and C6711) belong to the family of the C6x floating-point processors; whereas the C62xx and C64xx belong to the family of the C6x fixed-point processors. The C6711 is capable of both fixed- and floating-
point processing. The architecture and instruction set of the C6711 are discussed in Chapter 3.

1.3 CODE COMPOSER STUDIO

The Code Composer Studio (CCS) provides an integrated development environment (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 [21]. This executable file can be loaded and run directly on the C6711 processor.

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, or over, or out).

Real-time analysis can be performed using real-time data exchange (RTDX) associated with DSP/BIOS (Appendix G). RTDX allows for data exchange between the host and the target and 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 C6711 DSK board includes a JTAG emulator interface.

1.3.1 CCS Installation and Support

Use the parallel (printer) cable DB25 to connect the DSK board (J2) to the parallel port on the PC, such as LPT1 or LPT2. Use the 5-V adapter included with the DSK package to connect to the power connector J4, to turn on the DSK. Install CCS with the CD-ROM included with the DSK, preferably using the \ti structure (as default).

The CCS icon should be on the desktop as “CCS 2 ['C 6000]” and is used to launch CCS. The code generation tools (C compiler, assembler, linker) Version 4.1 are used.

On power, the three LEDs located near the four user dip switches should count from 1 to 7 (binary).
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, advanced DSP/BIOS
3. DSP instructions and registers
4. Tools on RTDX, DSP/BIOS, and so on.

An extensive amount of support material (pdf files) is included with CCS (see Refs. 22 to 34). There are also a few examples included with CCS, such as a confidence test example for the DSK, an audio example, and an example associated with the onboard flash.

CCS Version 2 was used to build and test the examples included in this book. A number of files included in the following subfolders/directories within c:\ti can be very useful:

1. docs: contains documentation and manuals.
2. myprojects: supplied for your projects. All the programs and projects discussed in this book can be placed within this subdirectory.
3. c6000\cgtools: contains code generation tools.
4. bin: contains many utilities.
5. c6000\examples: contains examples included with CCS.
6. c6000\RTDX: contains support files for real-time data transfer.
7. c6000\bios: contains support files for DSP/BIOS.

1.3.2 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 rts6701.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 processor.

1.4 PROGRAMMING EXAMPLES TO TEST THE DSK TOOLS

Three programming examples are introduced to illustrate some of the features of CCS and the DSK board. The primary focus is to become familiar with both the software and hardware tools. It is strongly suggested that you complete these three examples before proceeding to subsequent chapters.

1.4.1 Quick Test of DSK

Launch CCS from the icon on the desktop. Press GEL → Check DSK → Quick Test. The Quick Test can be used for confirmation of correct operation and installation. The following message is then displayed:

```
Switches: 7
Revision: 2
Target is OK
```

This assumes that the first three switches, USER_SW1, USER_SW2, and USER_SW3, are all in the up (ON) position. Change the switches to (1 1 0 x)₂ so that the first two switches are up (press the third switch down). The fourth switch is not used.

Repeat the procedure to select GEL → Check DSK → Quick Test and verify that the value of the switches is now 3 (with the display “Switches: 3”). You can set the value of the first three user switches from 0 to 7. Within your program you can then direct the execution of your code based on these eight values. Note that the Quick Test cycles the LEDs three times.

A confidence test program example is included with the DSK to test and verify proper operation of the major components of the DSK, such as interrupts, LEDs, SDRAM, DMA, serial ports, and timers.

**Alternative Quick Test of DSK**

1. Open/launch CCS from the icon on the desktop. Select File → Load Program. Access the accompanying disk. Click on the folder sine8_intr to Open (load) the file sine8_intr.out. This loads the executable file sine8_intr.out into the C6711 processor.

2. Select Debug → Run. Connect the OUT (connector J6) on the DSK board to a speaker or to an oscilloscope and verify the generation of a 1-kHz tone. The IN/OUT connectors (J7/J6) on the DSK board use a 3.5-mm jack audio cable.
The folder sine8_intr contains the necessary files to implement Example 1.1, which introduces some features of the tools.

### 1.4.2 Support Files

Create a new folder within your PC hard drive and name it *sine8_intr*. It is recommended that you place this folder in `c:\ti\myprojects` (it is assumed that you have installed CCS in `c:\ti`). Some of the same support files that are used in many examples in this book are included on the accompanying disk in the folder *Support*. For now, don’t worry too much about the content or functions of these files. Additional support files are included in the CCS CD with the DSK package. Copy the following support files from the folder *Support* (on the accompanying disk) into the folder *sine8_intr* that you created in your hard drive:

2. `C6xdsk.h`: header file that defines addresses of external memory interface, the serial ports, etc. (TI support file included with CCS).
3. `C6xinterrupts.h`: contains init functions for interrupt (TI support file included with the DSK).
4. `C6xdskinit.h`: header file with the function prototypes.
5. `C6xdskinit.c`: contains several functions used for the example `codec_poll` included with CCS. It includes functions to initialize the DSK, the codec, the serial ports, and for input/output.
6. `Vectors_11.asm`: version of `vectors.asm` included with CCS, but modified to handle interrupts. Twelve interrupts, INT4 through INT15, are available, and INT11 is selected within this vector file.

Also copy the C source file `sine8_intr.c` and the GEL file `amplitude.gel` from the disk (*sine8_intr* folder) into the folder *sine8_intr* on your hard drive.

*Note*: If you are using a C6211 DSK (which has been discontinued), change `XINT0` to `XINT1` within the function `comm_intr` in the file `C6xdskinit.c`. This is due to a silicon bug associated with the C6211.

### 1.4.3 Examples

#### Example 1.1: Sine Generation with Eight Points (*sine8_intr*)

This example generates a sinusoid using a table-lookup method. More important, it illustrates some features of CCS for editing, building a project, accessing the code generation tools, and running a program on the C6711 processor. The C source program `sine8_intr.c` shown in Figure 1.2 implements the sine generation.