
Digital Signal Processing and Applications with the C6713 and C6416 DSK

Rulph Chassaing

Worcester Polytechnic Institute

 **WILEY-
INTERSCIENCE**

A JOHN WILEY & SONS, INC., PUBLICATION

**Digital Signal Processing
and Applications with the
C6713 and C6416 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*

RULPH CHASSAING: *DIGITAL SIGNAL PROCESSING AND APPLICATIONS WITH THE C6713 AND C6416 DSK*

Digital Signal Processing and Applications with the C6713 and C6416 DSK

Rulph Chassaing

Worcester Polytechnic Institute

 **WILEY-
INTERSCIENCE**

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2005 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400, fax 978-646-8600, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

Library of Congress Cataloging-in-Publication Data:

Chassaing, Rulph.

Digital signal processing and applications with the C6713 and C6416 DSK / by Rulph Chassaing.
p. cm.

Includes bibliographical references and index.

ISBN 0-471-69007-4

1. Signal processing—Digital techniques. 2. Texas Instruments TMS320 series microprocessors.

I. Title.

TK5102.9.C47422 2004
621.382'2—dc22

2004050924

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Contents

Preface	xiii
List of Examples	xvii
Programs/Files on Accompanying CD	xxi
1 DSP Development System	1
1.1 Introduction	1
1.2 DSK Support Tools	2
1.2.1 DSK Board	3
1.2.2 TMS320C6713 Digital Signal Processor	5
1.3 Code Composer Studio	5
1.3.1 CCS Installation and Support	6
1.3.2 Useful Types of Files	7
1.4 Quick Test of DSK	7
1.5 Support Files	8
1.6 Programming Examples to Test the DSK Tools	9
1.7 Support Programs/Files Considerations	27
1.7.1 Initialization/Communication File	27
1.7.2 Vector File	30
1.7.3 Linker Command File	32
1.8 Compiler/Assembler/Linker Shell	33
1.8.1 Compiler	33
1.8.2 Assembler	34
1.8.3 Linker	34
	v

vi Contents

1.9	Assignments	35
	References	36
2	Input and Output with the DSK	39
2.1	Introduction	39
2.2	TLV320AIC23 (AIC23) Onboard Stereo Codec for Input and Output	40
2.3	Programming Examples Using C Code	42
2.4	Assignments	71
	References	72
3	Architecture and Instruction Set of the C6x Processor	73
3.1	Introduction	73
3.2	TMS320C6x Architecture	75
3.3	Functional Units	76
3.4	Fetch and Execute Packets	79
3.5	Pipelining	79
3.6	Registers	81
3.7	Linear and Circular Addressing Modes	82
	3.7.1 Indirect Addressing	82
	3.7.2 Circular Addressing	82
3.8	TMS320C6x Instruction Set	84
	3.8.1 Assembly Code Format	84
	3.8.2 Types of Instructions	85
3.9	Assembler Directives	86
3.10	Linear Assembly	87
3.11	ASM Statement within C	88
3.12	C-Callable Assembly Function	89
3.13	Timers	89
3.14	Interrupts	89
	3.14.1 Interrupt Control Registers	90
	3.14.2 Interrupt Acknowledgment	91
3.15	Multichannel Buffered Serial Ports	92
3.16	Direct Memory Access	92
3.17	Memory Considerations	93
	3.17.1 Data Allocation	93
	3.17.2 Data Alignment	94

3.17.3	Pragma Directives	94
3.17.4	Memory Models	95
3.18	Fixed- and Floating-Point Format	95
3.18.1	Data Types	95
3.18.2	Floating-Point Format	96
3.18.3	Division	97
3.19	Code Improvement	97
3.19.1	Intrinsics	97
3.19.2	Trip Directive for Loop Count	98
3.19.3	Cross-Paths	98
3.19.4	Software Pipelining	98
3.20	Constraints	99
3.20.1	Memory Constraints	99
3.20.2	Cross-Path Constraints	99
3.20.3	Load/Store Constraints	100
3.20.4	Pipelining Effects with More Than One EP within an FP	100
3.21	Programming Examples Using C, Assembly, and Linear Assembly	101
3.22	Assignments	115
	References	117
4	Finite Impulse Response Filters	119
4.1	Introduction to the z -Transform	119
4.1.1	Mapping from s -Plane to z -Plane	122
4.1.2	Difference Equations	123
4.2	Discrete Signals	124
4.3	FIR Filters	125
4.4	FIR Lattice Structure	127
4.5	FIR Implementation Using Fourier Series	131
4.6	Window Functions	135
4.6.1	Hamming Window	136
4.6.2	Hanning Window	136
4.6.3	Blackman Window	136
4.6.4	Kaiser Window	137
4.6.5	Computer-Aided Approximation	137
4.7	Programming Examples Using C and ASM Code	137
4.8	Assignments	173
	References	174

5	Infinite Impulse Response Filters	177
5.1	Introduction	177
5.2	IIR Filter Structures	178
5.2.1	Direct Form I Structure	178
5.2.2	Direct Form II Structure	179
5.2.3	Direct Form II Transpose	181
5.2.4	Cascade Structure	182
5.2.5	Parallel Form Structure	183
5.2.6	Lattice Structure	185
5.3	Bilinear Transformation	190
5.3.1	BLT Design Procedure	191
5.4	Programming Examples Using C and ASM Code	192
5.5	Assignments	205
	References	206
6	Fast Fourier Transform	208
6.1	Introduction	208
6.2	Development of the FFT Algorithm with Radix-2	209
6.3	Decimation-in-Frequency FFT Algorithm with Radix-2	210
6.4	Decimation-in-Time FFT Algorithm with Radix-2	217
6.5	Bit Reversal for Unscrambling	221
6.6	Development of the FFT Algorithm with Radix-4	221
6.7	Inverse Fast Fourier Transform	224
6.8	Programming Examples	225
6.8.1	Fast Convolution	237
6.9	Assignments	245
	References	247
7	Adaptive Filters	249
7.1	Introduction	249
7.2	Adaptive Structures	251
7.3	Adaptive Linear Combiner	254
7.4	Performance Function	257
7.5	Searching for the Minimum	259
7.6	Programming Examples for Noise Cancellation and System Identification	262
	References	282

8	Code Optimization	284
8.1	Introduction	284
8.2	Optimization Steps	285
8.2.1	Compiler Options	285
8.2.2	Intrinsic C Functions	286
8.3	Procedure for Code Optimization	286
8.4	Programming Examples Using Code Optimization Techniques	286
8.5	Software Pipelining for Code Optimization	293
8.5.1	Procedure for Hand-Coded Software Pipelining	293
8.5.2	Dependency Graph	294
8.5.3	Scheduling Table	295
8.6	Execution Cycles for Different Optimization Schemes	302
	References	303
9	DSP/BIOS and RTDX Using MATLAB, Visual C++, Visual Basic, and LabVIEW	304
9.1	Introduction to DSP/BIOS	306
9.2	RTDX Using MATLAB to Provide Interface Between PC and DSK	311
9.3	RTDX Using Visual C++ to Interface with DSK	321
9.4	RTDX Using Visual Basic to Provide Interface Between PC and DSK	332
9.5	RTDX Using LabVIEW to Provide Interface Between PC and DSK	335
	Acknowledgments	342
	References	342
10	DSP Applications and Student Projects	343
10.1	DTMF Detection Using Correlation, FFT, and Goertzel Algorithm	343
10.1.1	Using a Correlation Scheme and Onboard LEDs for Verifying Detection	345
10.1.2	Using RTDX with Visual C++ to Display Detected DTMF Signals on the PC	348
10.1.3	Using FFT and Onboard LEDs for Verifying Detection	350
10.1.4	Using Goertzel Algorithm	350
10.2	Beat Detection Using Onboard LEDs	352

X Contents

10.3	FIR with RTDX Using Visual C++ for Transfer of Filter Coefficients	355
10.4	Radix-4 FFT with Frequency Domain Filtering	357
10.5	Radix-4 FFT with RTDX Using Visual C++ and MATLAB for Plotting	357
10.6	Spectrum Display Through EMIF Using a Bank of 32 LEDs	360
10.7	Spectrum Display Through EMIF Using LCDs	364
10.8	Time-Frequency Analysis of Signals with Spectrogram	368
10.8.1	Simulation Using MATLAB	368
10.8.2	Spectrogram with RTDX Using MATLAB	370
10.8.3	Spectrogram with RTDX Using Visual C++	372
10.9	Audio Effects (Echo and Reverb, Harmonics, and Distortion)	373
10.10	Voice Detection and Reverse Playback	375
10.11	Phase Shift Keying—BPSK Encoding and Decoding with PLL	377
10.11.1	BPSK Single-Board Transmitter/Receiver Simulation	377
10.11.2	BPSK Transmitter/Voice Encoder with Real-Time Input	381
10.11.3	Phase-Locked Loop	383
10.11.4	BPSK Transmitter and Receiver with PLL	386
10.12	Binary Phase Shift Keying	390
10.13	Modulation Schemes—PAM and PSK	393
10.13.1	Pulse Amplitude Modulation	393
10.13.2	Phase-Shift Keying	396
10.14	Selectable IIR Filter and Scrambling Scheme Using Onboard Switches	401
10.15	Convolutional Encoding and Viterbi Decoding	404
10.16	Speech Synthesis Using Linear Prediction of Speech Signals	414
10.17	Automatic Speaker Recognition	418
10.18	μ -Law for Speech Companding	422
10.19	Voice Scrambler Using DMA and User Switches	423
10.20	SB-ADPCM Encoder/Decoder: Implementation of G.722 Audio Coding	423
10.21	Encryption Using the Data Encryption Standard Algorithm	425
10.22	Phase-Locked Loop	429
10.23	Miscellaneous Projects	430
10.23.1	Multirate Filter	431
10.23.2	Acoustic Direction Tracker	436
10.23.3	Neural Network for Signal Recognition	437
10.23.4	Adaptive Temporal Attenuator	441

	Contents	xi
10.23.5	FSK Modem	442
10.23.6	Image Processing	443
10.23.7	Filter Design and Implementation Using a Modified Prony's Method	444
10.23.8	PID Controller	444
10.23.9	Four-Channel Multiplexer for Fast Data Acquisition	444
10.23.10	Video Line Rate Analysis	444
	Acknowledgments	444
	References	445
 Appendix A TMS320C6x Instruction Set		 450
A.1	Instructions for Fixed- and Floating-Point Operations	450
A.2	Instructions for Floating-Point Operations	450
	References	450
 Appendix B Registers for Circular Addressing and Interrupts		 452
	Reference	452
 Appendix C Fixed-Point Considerations		 455
C.1	Binary and Two's-Complement Representation	455
C.2	Fractional Fixed-Point Representation	458
C.3	Multiplication	458
	Reference	461
 Appendix D MATLAB Support Tools		 462
D.1	SPTool and FDATool for FIR Filter Design	462
D.2	SPTool and FDATool for IIR Filter Design	465
D.3	MATLAB for FIR Filter Design Using the Student Version	468
D.4	MATLAB for IIR Filter Design Using the Student Version	470
D.5	BLT Using MATLAB and Support Programs on CD	471
D.6	FFT and IFFT	477
	References	478
 Appendix E Additional Support Tools		 479
E.1	Goldwave Shareware Utility as a Virtual Instrument	479
E.2	Filter Design Using DigiFilter	480

xii	Contents	
	E.2.1 FIR Filter Design	480
	E.2.2 IIR Filter Design	481
E.3	FIR Filter Design Using a Filter Development Package	482
	E.3.1 Kaiser Window	482
	E.3.2 Hamming Window	484
E.4	Visual Application Builder and LabVIEW	485
E.5	Alternative Input/Output	485
	References	485
	Appendix F Fast Hartley Transform	486
	References	492
	Appendix G Goertzel Algorithm	493
G.1	Design Considerations	493
	References	496
	Appendix H TMS320C6416 DSK	497
H.1	TMS320C64x Processor	497
H.2	Programming Examples Using the C6416 DSK	498
	References	502
	Appendix I TMS320C6711 DSK	503
	Reference	503
	Index	505

Preface

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 cellular phones, digital cameras, high-definition television (HDTV), radio, fax transmission, modems, 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 introduced the TM320C6x processor, based on the very-long-instruction-word (VLIW) architecture. This new 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 “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 105 programming examples, most using C code, with a few in assembly and linear assembly code. A list of these examples appears on page xvii. A total of 22 students’ projects are also discussed. These projects cover a wide

range of applications in filtering, spectrum analysis, modulation techniques, speech processing, and so on.

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. Many assignments are included at the end of Chapters 1–6.

This book can be used in the following ways:

1. For a DSP course with a laboratory component, using parts of Chapters 1–9. If needed, the book can be supplemented with some additional theoretical materials, since its emphasis is on the practical aspects of DSP. It is possible to cover Chapter 7 on adaptive filtering following Chapter 4 on finite impulse response (FIR) filtering (since there is only one example in Chapter 7 that uses materials from Chapter 5). It is my conviction that adaptive filtering should be incorporated into an undergraduate course in DSP.
2. For a laboratory course using many of the examples and experiments from Chapters 1–7 and Chapter 9. The beginning of the semester can be devoted to short programming examples and experiments and the remainder of the semester for a final project. The wide range of sample projects (for both undergraduate and graduate students) discussed in Chapter 10 can be very valuable.
3. For a senior undergraduate or first-year graduate design project course using selected materials from Chapters 1–10.
4. For the practicing engineer as a tutorial and reference, 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 TMS320C6713 DSP starter kit (DSK). It is essential to perform these 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. Appendix H contains several programming examples using the TMS320C6416 DSK.

In Chapter 2 we illustrate input and output (I/O) with the AIC23 stereo codec on the DSK board through many programming examples. 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 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. Appendix D illustrates MATLAB for the design of FIR and IIR filters.

Chapter 6 covers the development of the fast Fourier transform (FFT). Programming examples on FFT are included using both radix-2 and radix-4 FFT. 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. Students have been very receptive to applications in adaptive filtering. Chapter 8 illustrates techniques for code optimization.

In Chapter 9 we introduce DSP/BIOS and discuss a number of schemes (Visual C++, MATLAB, etc.) for real-time data transfer (RTDX) and communication between the PC and the DSK.

Chapter 10 discusses a total of 22 projects implemented by undergraduate and graduate students. They cover a wide range of DSP applications in filtering, spectrum analysis, modulation schemes, speech processing, and so on.

A CD is included with this book and contains all the programs discussed. See page xxi for a list of the folders that contain the support files for the examples and projects.

Over the last 10 years, faculty members from over 200 institutions have taken my workshops on “DSP and Applications.” Many of these workshops were supported 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 my previous book based on the TMS320C6711 DSK. I appreciate the many suggestions made by Dr. Mounir Boukadoum of the University of Quebec, Dr. Subramaniam Ganesan from Oakland University, and Dr. David Kozel from Purdue University at Calumet. I also thank Dr. Darrell Horning of the University of New Haven, with whom I coauthored my first book, *Digital Signal Processing with the TMS320C25*, for introducing me to “book writing.” I thank all the students at Roger Williams University, the University of Massachusetts at Dartmouth, and Worcester Polytechnic Institute (WPI) who have taken my real-time DSP and senior design project courses, based on the TMS320 processors, over the last 20 years. The contribution of Aghogho Obi, from WPI, is very much appreciated.

The continued support of many people from Texas Instruments is also very much appreciated: Cathy Wicks and Christina Peterson, in particular, have been very supportive of this book.

Special appreciation: The laboratory assistance of Walter J. Gomes III in several workshops and during the development of many examples has been invaluable. His contribution is appreciated.

RULPH CHASSAING
Chassaing@msn.com
Chassaing@ece.wpi.edu

List of Examples

1.1	Sine Generation Using Eight Points with DIP Switch Control	9
1.2	Generation of the Sinusoid and Plotting with CCS	19
1.3	Dot Product of Two Arrays	22
2.1	Loop Program Using Interrupt	43
2.2	Loop Program Using Polling	45
2.3	Stereo Input and Stereo Output	46
2.4	Sine Generation with Two Sliders for Amplitude and Frequency Control	48
2.5	Loop Program with Input Data Stored in Memory	50
2.6	Loop with Data in a Buffer Printed to a File	52
2.7	Square-Wave Generation Using a Lookup Table	53
2.8	Ramp Generation Using a Lookup Table	54
2.9	Ramp Generation without a Lookup Table	55
2.10	Echo	56
2.11	Echo with Control for Different Effects	57
2.12	Sine Generation with Table Values Generated within the Program	59
2.13	Sine Generation with a Table Created by MATLAB	60
2.14	Amplitude Modulation	62
2.15	Sweep Sinusoid Using a Table with 8000 Points	63
2.16	Pseudorandom Noise Sequence Generation	65
2.17	Sine Generation with Dip Switch Control	66
2.18	Use of External Memory to Record Voice	67
2.19	Use of Flash Memory—Programming the Onboard Flash	69
3.1	Efficient Dot Product	102
3.2	Sum of $n + (n - 1) + (n - 2) + \dots + 1$, Using C Calling an Assembly Function	103

xviii List of Examples

3.3	Factorial of a Number Using C Calling an Assembly Function	104
3.4	32-bit Pseudorandom Noise Generation Using C Calling an Assembly Function	105
3.5	Code Detection Using C Calling an ASM Function	107
3.6	Dot Product Using Assembly Program Calling an Assembly Function	109
3.7	Dot Product Using C Function Calling a Linear Assembly Function	112
3.8	Factorial Using C Calling a Linear Assembly Function	114
4.1	FIR Filter Implementation: Bandstop and Bandpass	139
4.2	Effects on Voice Using Three FIR Lowpass Filters	144
4.3	Implementation of Four Different Filters: Lowpass, Highpass, Bandpass, and Bandstop	147
4.4	FIR Implementation with a Pseudorandom Noise Sequence as Input to a Filter	148
4.5	FIR Filter with Internally Generated Pseudorandom Noise as Input to a Filter and Output Stored in Memory	151
4.6	Two Notch Filters to Recover Corrupted Input Voice	154
4.7	FIR Implementation Using Four Different Methods	156
4.8	Voice Scrambling Using Filtering and Modulation	158
4.9	Illustration of Aliasing Effects with Down-Sampling	161
4.10	Implementation of an Inverse FIR Filter	163
4.11	FIR Implementation Using C Calling an ASM Function	164
4.12	FIR Implementation Using C Calling a Faster ASM Function	167
4.13	FIR Implementation Using C Calling an ASM Function with a Circular Buffer	168
4.14	FIR Implementation Using C Calling an ASM Function with a Circular Buffer in External Memory	172
5.1	IIR Filter Implementation Using Second-Order Stages in Cascade	192
5.2	Generation of Two Tones Using Two Second-Order Difference Equations	196
5.3	Sine Generation Using a Difference Equation	199
5.4	Generation of a Swept Sinusoid Using a Difference Equation	200
5.5	IIR Inverse Filter	202
5.6	Sine Generation Using a Difference Equation with C Calling an ASM Function	205
6.1	DFT of a Sequence of Real Numbers with Output from the CCS Window	225
6.2	FFT of a Real-Time Input Signal Using an FFT Function in C	227
6.3	FFT of a Sinusoidal Signal from a Table Using TI's C-Callable Optimized FFT Function	229

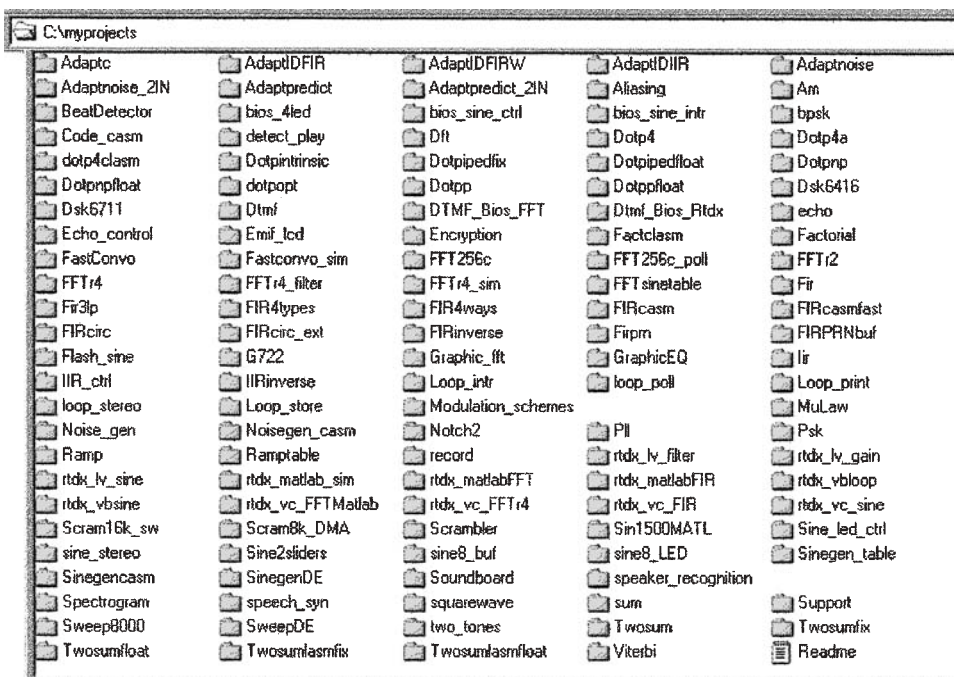
6.4	FFT of Real-Time Input Using TI's C-Callable Optimized Radix-2 FFT Function	232
6.5	Radix-4 FFT of Input from a Lookup Table Using TI's C-Callable Optimized FFT Function	234
6.6	Radix-4 FFT of Real-Time Input Using TI's C-Callable Optimized FFT Function	236
6.7	Fast Convolution With Overlap-Add for FIR Implementation Using TI's Floating-Point FFT Functions	237
6.8	Fast Convolution with Overlap-Add Simulation for FIR Implementation Using a C-Coded FFT Function	241
6.9	Graphic Equalizer	242
7.1	Adaptive Filter Using C Code Compiled with Borland C/C++	262
7.2	Adaptive Filter for Sinusoidal Noise Cancellation	265
7.3	Adaptive FIR Filter for Noise Cancellation Using External Inputs	267
7.4	Adaptive FIR Filter for System ID of a Fixed FIR as an Unknown System	270
7.5	Adaptive FIR for System ID of a Fixed FIR as an Unknown System with Weights of an Adaptive Filter Initialized as an FIR Bandpass	272
7.6	Adaptive FIR for System ID of Fixed IIR as an Unknown System	275
7.7	Adaptive Predictor for Cancellation of Narrowband Interference Added to a Desired Wideband Signal	275
7.8	Adaptive Predictor for Cancellation of Narrowband Interference Added to a Desired Wideband Signal Using External Inputs	280
8.1	Sum of Products with Word-Wide Data Access for Fixed-Point Implementation Using C Code	287
8.2	Separate Sum of Products with C Intrinsic Functions Using C Code	288
8.3	Sum of Products with Word-Wide Access for Fixed-Point Implementation Using Linear ASM Code	288
8.4	Sum of Products with Double-Word Load for Floating-Point Implementation Using Linear ASM Code	289
8.5	Dot Product with No Parallel Instructions for Fixed-Point Implementation Using ASM Code	289
8.6	Dot Product with Parallel Instructions for Fixed-Point Implementation Using ASM Code	290
8.7	Two Sums of Products with Word-Wide (32-Bit) Data for Fixed-Point Implementation Using ASM Code	290
8.8	Dot Product with No Parallel Instructions for Floating-Point Implementation Using ASM Code	291
8.9	Dot Product with Parallel Instructions for Floating-Point Implementation Using ASM Code	292

XX List of Examples

8.10	Two Sums of Products with Double-Word-Wide (64-Bit) Data for Floating-Point Implementation Using ASM Code	292
8.11	Dot Product Using Software Pipelining for a Fixed-Point Implementation	297
8.12	Dot Product Using Software Pipelining for a Floating-Point Implementation	299
9.1	Sine Generation with DIP Switch Control through DSP/BIOS	306
9.2	Blinking of LEDs at Different Rates Using DSP/BIOS	309
9.3	Sine Generation Using BIOS to Set Up Interrupt INT11	310
9.4	MATLAB–DSK Interface Using RTDX	311
9.5	MATLAB–DSK Interface Using RTDX, with MATLAB For FFT and Plotting	314
9.6	MATLAB–DSK Interface Using RTDX For FIR Filter Implementation	317
9.7	Visual C++–DSK Interface Using RTDX for Amplitude Control of the Sine Wave	321
9.8	Visual C++–DSK Interface Using RTDX, with MATLAB Functions for FFT and Plotting	327
9.9	Visual Basic–DSK Interface Using RTDX for Amplitude Control of a Sine Wave	332
9.10	Visual Basic–DSK Interface Using RTDX for Amplitude Control of Output in a Loop Program	334
9.11	LabVIEW–DSK Interface Using RTDX for FIR Filtering	336
9.12	LabVIEW–DSK Interface Using RTDX for Controlling the Gain of a Generated Sinusoid	339
9.13	LabVIEW–DSK Interface Using RTDX for Controlling the Amplitude of a Generated Sinusoid with Real-Time Output from the DSK	341
D.1	SPTool and FDATool for FIR Filter Design	462
D.2	SPTool and FDATool for IIR Filter Design	465
D.3	FIR Filter Design Using MATLAB’s Student Version	468
D.4	Multiband FIR Filter Design Using MATLAB	469
D.5	IIR Filter Design Using MATLAB’s Student Version	470
H.1	Sine Generation with DIP Switch Control Using the C6416 DSK	498
H.2	Loop Program Using the C6416 DSK	499
H.3	FIR/IIR Implementation Using the C6416 DSK	499
H.4	FFT with C-Coded FFT Function Using the C6416 DSK	500
H.5	Adaptive FIR Filter Implementation Using the C6416 DSK	501
H.6	DTMF Implementation on the C6416 DSK Using the Goertzel Algorithm and the FFT, With RTDX Using Visual C++	501
I.1	Loop Program Using the C6711 DSK	503

Programs/Files on Accompanying CD

A list of the folders included on the accompanying CD is shown below. The folders contain the programs/files for the examples/projects covered in the book.



1

DSP Development System

- Testing the software and hardware tools with Code Composer Studio
- Use of the TMS320C6713 DSK
- Programming examples to test the tools

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), and the DSP starter kit (DSK) with the TMS320C6713 floating-point processor onboard and complete support for input and output. Three examples illustrate both the software and hardware tools included with the DSK. It is strongly suggested that you review these three examples before proceeding to subsequent chapters.

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 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. Real-time 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.

A number of books and articles address the importance of digital signal processors for a number of applications [1–22]. Various technologies have been used for real-time processing, from fiber optics for very high frequency to DSPs very suitable for the audio-frequency range. Common applications using these processors have been for frequencies from 0 to 96 kHz. Speech can be sampled at 8 kHz (the rate at which samples are acquired), which implies that each value sampled is acquired at a rate of $1/(8\text{ kHz})$ or 0.125 ms. A commonly used sample rate of a compact disk is 44.1 kHz. Analog/digital (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 are a special input filter for anti-aliasing 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 1.1, that contains the TMS320C6713 (C6713) floating-point digital signal processor as well as a 32-bit stereo codec for input and output (I/O) support.
 - (c) A universal synchronous bus (USB) cable that connects the DSK board to a PC.
 - (d) A 5V power supply for the DSK board.
2. *An IBM-compatible PC.* The DSK board connects to the USB port of the PC through the USB 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.

All the files/programs listed and discussed in this book (except some student project files in Chapter 10) are included on the accompanying CD. Most of the examples (with some minor modifications) can also run on the fixed-point C6416-based DSK. See Appendix H for the appropriate support files along with five illustrative examples. Reference 1 contains examples implemented on the C6711-based DSK (which has been discontinued). A list of all the examples is given on pages xv–xviii.

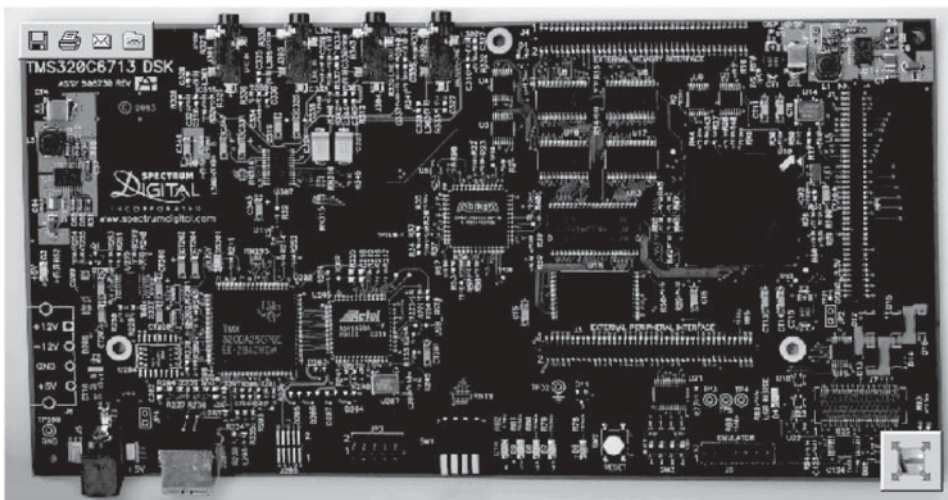
1.2.1 DSK Board

The DSK package is powerful, yet relatively inexpensive (\$395), with the necessary hardware and software support tools for real-time signal processing [23–43]. It is a complete DSP system. The DSK board, with an approximate size of 5×8 in., includes the C6713 floating-point digital signal processor and a 32-bit stereo codec TLV320AIC23 (AIC23) for input and output.

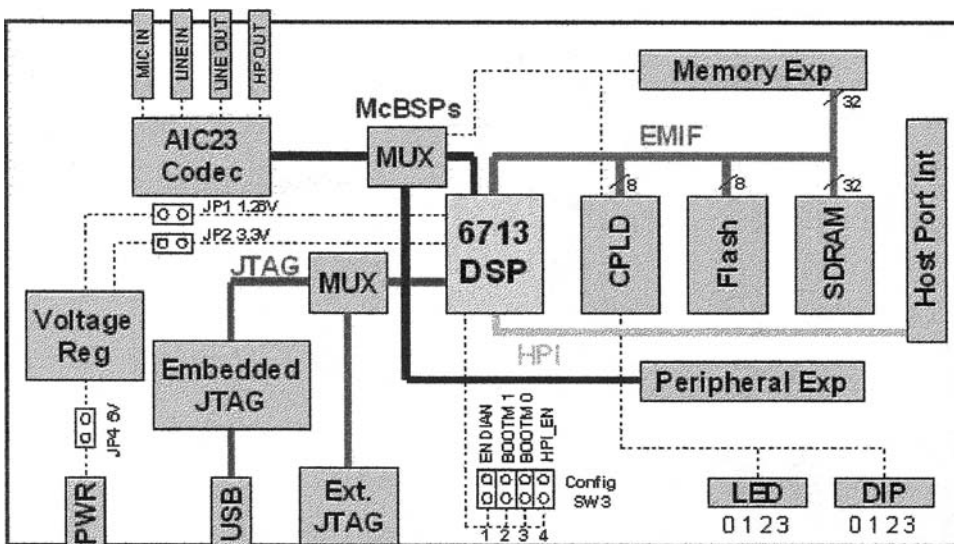
The onboard codec AIC23 [37] uses a sigma–delta technology that provides ADC and DAC. It connects to a 12-MHz system clock. Variable sampling rates from 8 to 96 kHz can be set readily.

A daughter card expansion is also provided on the DSK board. Two 80-pin connectors provide for external peripheral and external memory interfaces. Two project examples in Chapter 10 illustrate the use of the external memory interface (EMIF) with light-emitting diodes (LEDs) and liquid-crystal displays (LCDs) for spectrum display.

The DSK board includes 16MB (megabytes) of synchronous dynamic random access memory (SDRAM) and 256kB (kilobytes) of flash memory. Four connectors on the board provide input and output: MIC IN for microphone input, LINE IN for line input, LINE OUT for line output, and HEADPHONE for a headphone output (multiplexed with line output). The status of the four user dip switches on the DSK board can be read from a program and provides the user with a feedback control interface. The DSK operates at 225 MHz. Also onboard the DSK are voltage



(a)



(b)

FIGURE 1.1. TMS320C6713-based DSK board: (a) board; (b) diagram. (Courtesy of Texas Instruments)

regulators that provide 1.26V for the C6713 core and 3.3V for its memory and peripherals.

Appendix H illustrates a DSK based on the fixed-point processor C6416.

1.2.2 TMS320C6713 Digital Signal Processor

The TMS320C6713 (C6713) is based on the 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 225MHz, the C6713 is capable of fetching eight 32-bit instructions every $1/(225\text{MHz})$ or 4.44ns.

Features of the C6713 include 264kB of internal memory (8kB as L1P and L1D Cache and 256kB as L2 memory shared between program and data space), eight functional or execution units composed of six arithmetic-logic units (ALUs) and two multiplier units, a 32-bit address bus to address 4GB (gigabytes), and two sets of 32-bit general-purpose registers.

The C67xx (such as the C6701, C6711, and C6713) 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 C6713 is capable of both fixed- and floating-point processing. The architecture and instruction set of the C6713 are discussed in Chapter 3.

1.3 CODE COMPOSER STUDIO

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. Chapter 3 introduces the linear assembly source file with extension `.sa`, which is a cross between C and assembly code. 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 monitor-

ing 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) (Chapter 9). 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.

1.3.1 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.