Digital Filter Designer's Handbook

Featuring C Routines

C. Britton Rorabaugh

TAB Books
Division of McGraw-Hill, Inc.
Blue Ridge Summit, PA 17294-0850

Contents

List of Programs

Preface

If you're going to own only one book on digital filters, this is the one to have. If you already own several, you need this book anyway—it contains quite a lot of useful information not available in any other book. I wrote this book for individuals faced with the need to design working digital filters—it is not intended as an academic text. All the necessary theoretical background is provided in the early chapters, and practical digital filter design techniques are provided in the later chapters. These design techniques are supported by numerous computer routines written in the C programming language. The techniques and programs presented in this book will prove to be very useful to engineers, students, and hobbyists engaged in the design of digital filters.

All of the programs in this book were written and tested using Think C for the Apple Macintosh computer. I made a conscientious effort to limit the programs to the ANSI standard subset of Think C and to avoid any machine dependencies. Potential efficiencies were sacrificed for the sake of portability and tutorial clarity. However, a few specific items need to be pointed out:

1. Constants used by several different functions are collected into a single "include" file called **globDefs.h** (a listing of this file is provided in App. A). The "new" style of ANSI prototyping was used throughout all of the software generated for this book. All the pertinent prototypes are collected in a file called **protos.h**, which is provided in App. B.

2. Nice long file names such as **computeRemezAmplitude.c** are allowed on the Macintosh, but on MS-DOS machines file names are limited to eight characters plus a three-character extension. Except for the two header files mentioned above, all the files on the accompanying disk have names that are keyed to the chapter number in which the listing appears.

3. I found it convenient to define a new type **real** that is the same as **double.** For use on machines with limited memory, **real** could be redefined as **float** to save memory, but accuracy could suffer. Being a long-time Fortran user, I also found it convenient to create a **logical** type. The lack of intrinsic complex types in C was overcome via a **complex** structure definition, and a set of complex arithmetic functions is detailed in App. C.

Britt Rorabaugh