DSP Implementation using the TMS320C62x Processors

(Code Composer)

By Dr. Naim Dahnoun

Email: Naim_Dahnoun@hotmail.com

HOME

This is a list of DSP codes for Code Composer accompanying the Book.

Chapter 1 Introduction:

This introductory chapter provides the reader with general knowledge on general-purpose DSP processors and also provides an up-to-date TMS320 roadmap showing the evolution of Texas Instruments' DSP chips in terms of processing power.

Return to Index

Chapter 2 The TMS320C62x/C67x Architecture:

The objective of this chapter is to provide a comprehensive description of the 'C6x architecture. This includes a detailed description of the Central Processing Unit (CPU) and program control along with an overview of the memory organisation, serial ports, boot function and internal timer.

Return to Index

Chapter 3 Software development tools and TMS320C6201 EVM overview:

This chapter is divided into three main parts. The first part describes the software development tools, the second part describes the Evaluation Module (EVM) and finally the third part describes the codec, and use of interrupts along with some useful programs for testing the TMS320C6201 EVM.

Source Code Directory: Inout (Chap3)

Source Files:

- Inout (Chap3)\INOUT.C
- Inout (Chap3)\INICODEC.C

Return to Index

Chapter 4 Software optimisation:

To introduce the need for code optimisation, this chapter starts by developing the concept of pipelining. Since the TMS320C62xx and the TMS320C67xx each have eight units, which are dedicated to different operations, and since different instructions can have different latencies, the programmer or the tools are left with the burden of scheduling the code. Backed by examples, this chapter explains the different techniques used to optimise DSP code on these processors.

Return to Index

Chapter 5 Finite Impulse Response (FIR) filter implementation:

The purpose of this chapter is twofold. Primarily, it shows how to design an FIR filter and implement it on the TMS320C62xx processor, and secondly, it shows how to optimise the code as discussed in Chapter 4. This chapter discusses the interface between C and assembly, how to use intrinsics, and how to put into practice material that has been covered in the previous chapters.

Codes Directory 1: Fir (Chap5)\C\

Source Files:

- Fir (Chap5)\C\Fir.c

Codes Directory 2: Fir (Chap5)\ASM\

Source Files:

- Fir (Chap5)\ASM\Fir asm.c
- Fir (Chap5)\ASM\Firasm.asm

Return to Index

Chapter 6 Infinite Impulse Response (IIR) filter implementation

This chapter introduces the IIR filters and describes two popular design methods, that is the bilinear and the impulse invariant methods. Step by step, this chapter shows the procedures necessary to implement typical IIR filters specified by their transfer functions. Finally, this chapter provides complete implementation of an IIR filter in C language, assembly and linear assembly, and shows how to interface C with linear assembly. implementation of an IIR filter in C language, assembly and linear assembly, and shows how to interface C with linear assembly.

Codes Directory 1: Iir (Chap6)\C\

Source Files:

- Iir (Chap6)\C\Iir.c

Codes Directory 2:

Source Files:

- Iir (Chap6)\SA\Iir sa.c
- Iir (Chap6)\SA\Iirsa.sa

Return to Index

Chapter 7 Adaptive filter implementation:

This chapter starts by introducing the need for an adaptive filter in com„munications. It then shows how to calculate the filter coefficients using the Mean Square Error (MSE) criterion, exposes the Least Mean Square (LMS) algorithm and, finally, shows how the LMS algorithm is implemented in both C and assembly.

Codes Directory 1: Adaptive (Chap7)\LMS_C\

Source Files:

- Adaptive (Chap7)\LMS_C\Lms_c.c

Codes Directory 2: Adaptive (Chap7)\LMS ASM\

Source Files:

- Adaptive (Chap7)\LMS ASM\Lmsasm.asm

Return to Index

Chapter 8 Goertzel algorithm implementation:

This chapter deals with Dual Tone Multi-Frequency (DTMF) detection and provides a practical example of the Goertzel algorithm. This chapter also shows how to produce optimised code by the pen and paper method, describes linear assembly and demonstrates how to program the Direct Memory Access (DMA).

Codes Directory 1: Goertzel (Chap8)\BASIC\

Source Files:

- Goertzel (Chap8)\BASIC\Goertzel.c

Codes Directory 2: Goertzel (Chap8)\GTZSA\

Source Files:

- Goertzel (Chap8)\GTZSA\Gtzsa.c
- Goertzel (Chap8)\GTZSA\Gtz_sa.sa

Codes Directory 3: Goertzel (Chap8)\GTZDMA

Source Files:

- Goertzel (Chap8)\GTZDMA\Gtzdma.c

Return to Index

Chapter 9 Implementation of the Discrete Cosine Transform

This chapter starts by introducing the need for video compression to reduce the channel bandwidth requirement, then explains the Joint Photographic Experts Group (JPEG) image codec. This includes a detailed discussion and the implementation of the Discrete Cosine Transform (DCT) and Inverse Discrete Cosine Transform (IDCT) and concentrates on their optimisation. An explanation of the PC-DSP communication via the PCI bus is also provided.

Codes Directory 1: Dct (Chap9)\SLOWDCT\

Source Files:

- Dct(Chap9)\SLOWDCT\Dct_Main.c
- Dct(Chap9)\SLOWDCT\Dct.c
- Dct(Chap9)\SLOWDCT\Dct_Main.h

Codes Directory 2: Dct(Chap9)\FASTDCT\

Source Files:

- Dct (Chap9)\FASTDCT\Dct Main.c
- Dct (Chap9)\SLOWDCT\Dct.c
- Dct (Chap9)\FASTDCT\Dct Main.h

Codes Directory 3: Dct (Chap9)\DctDma\

Source Files:

- Dct (Chap9)\DCTDMA\DctDma.c
- Dct (Chap9)\DctDma\DCTHOST.C

Return to Index

Author's Contact:

Bristol University

Queen's building

University walk

Bristol BS8 1TR, UK.

Return to Index

_