Appendix E

附录E.

Related Tools for DSKs

DESK的相关工具

Introduction

介绍

HIS appendix contains information on several tools that are available for use with the OMAP-L138 and TMS320C6713 DSKs. (Note that to use the tools with the TMS320C6713 DSK, the board must be equipped with the Educational DSP, LLC HPI daughtercard.) The Host Port Interface provides external access to the DSPs memory space, as described in Appendix D. This allows the tools to download and start programs, then read and write DSP memory locations to get data back from the DSK and to control

HIS附录包含有关可用于OMAP-L138和TMS320C6713 DSK的多种工具的信息（注意，要使用带有TMS320C6713 DSK的工具，电路板必须配备Educational DSP，LLC HPI子卡）主机端口接口提供对DSP存储空间的外部访问，如附录D中所述这允许工具下载和启动程序，然后读取和写入DSP存储器位置以从DSK获取数据并进行控制

the program.

该程序

Windows Control Applications

Windows控制应用程序

To control the DSK from a Windows application, programs must be created for both the host computer and the DSK. The sample host computer Windows application is written in Microsoft Visual C++. The interface between the host computer and the DSK can be serial RS-232 or USB. The details of this interface are hidden in a dynamic link library (DLL) file that is included with the host computer program. To transfer data to and from the DSK, the host computer must know the variable addresses where the data is stored on the DSK. To simplify this process, a predefined data structure is used, and the interface software has the ability to determine where the data structure is located in the DSK’s memory space.

要从Windows应用程序控制DISK，必须为主机和DSK创建程序主机和DSK之间的接口可以是串行RS-232或USB此接口的详细信息隐藏在主机程序附带的动态链接库（DLL）文件中要与DESK之间传输数据，主机必须知道数据存储在DESK上的变量地址为了简化此过程，使用预定义的数据结构，接口软件能够确定数据结构在DSK存储空间中的位置

The host computer can perform a few basic operations:

主机可以执行一些基本操作：

Reset the DSP.

重置DSP

Load a program onto the DSP.

将程序加载到DSP上

Start the DSP program.

启动DSP程序

Read and write DSP memory.

读写DSP内存

Control of a DSP program is implemented by writing to variables in the DSP memory space. Program status and output data are obtained by reading from variables in the DSP memory space. However, keeping track of the specific addresses of all the variables in the DSP program is tedious and error-prone, since variable locations can change each time a program is recompiled. To simplify the process of finding variable addresses, the DSP software establishes a special data structure (HostInterfaceData) so that the variables will be in a known location.

通过写入DSP存储空间中的变量来实现DSP程序的控制通过读取DSP存储空间中的变量来获得程序状态和输出数据但是，跟踪DSP程序中所有变量的特定地址是繁琐且容易出错的，因为每次编译程序时变量位置都会改变为了简化查找​​变量地址的过程，DSP软件建立了一个特殊的数据结构（HostInterfaceData），以便变量位于已知位置

397

397

398 APPENDIX E. RELATED TOOLS FOR DSKS

398附录E.相关工具用于桌子

The host software first loads the program onto the DSK. Then, the location of the HostInterfaceData structure is determined by reading the symbol table embedded in the executable file. The host application can determine a variable’s address by adding the desired variable’s offset within the HostInterfaceData structure to the address of the HostInterfaceData symbol. This address is then used in the host read and write functions that access the DSP’s memory.

主机软件首先将程序加载到DESK上然后，通过读取嵌入在可执行文件中的符号表来确定主机接口数据结构的位置主机应用程序可以通过在主机接口数据结构中将所需变量的偏移量添加到HostInterfaceData符号的地址来确定变量的地址然后，该地址用于访问DSP内存的主机读写功能

E.2.1 Sample Windows Control Application

和21 Windows控制应用程序示例

Detailed documentation and the complete source code for the sample Windows Control applications are available in the WIN\_CONTROL\_APPS directory of Appendix E. The basic Windows Control application implements a simple audio talk-through with a gain control. Further enhancements show how to create a simple oscilloscope and spectrum analyzer.

有关示例Windows Control应用程序的详细文档和完整源代码，请参见附录E的WIN\_CONTROL\_APPS目录基本的Windows Control应用程序通过增益控制实现简单的音频通话进一步的增强功能展示了如何创建一个简单的示波器和频谱分析仪

MATLAB Exports

MATLAB出口

Using the MATLAB§R

使用MATLAB§R

program SPTool is a convenient, graphical way to design digital

程序SP工具是一种方便的图形化数字设计方式

filters.1 To be able to use those designs in Code Composer Studio (CCS), we need to export them into a C language format. There are four MATLAB m-files discussed in this appendix and included with the software for the book that can be used to help automate this process (see the Appendix E MatlabExports directory). In all cases, two files are created, a C header file (filename.h) declaring the variables, and a C source file (filename.c) defining them. You can use any file name you wish by specifying it in the argument list for the m-file, as shown in the examples below. These files can then be included in your CCS project. In this appendix it is assumed that the reader is familiar with using MATLAB and SPTool. It is recommended that you add a MATLAB path to the directory where you have installed the m-files.

过滤器1为了能够在Code Composer Studio（CCS）中使用这些设计，我们需要将它们导出为C语言格式本附录中讨论了四个MATLAB m文件，这些文件包含在本书的软件中，可用于帮助自动完成此过程（请参阅附录E Matlab Exports目录）在所有情况下，都会创建两个文件，一个C头文件（文件名h）声明变量和C源文件（文件名c）定义它们您可以通过在m文件的参数列表中指定它来使用您希望的任何文件名，如下面的示例所示然后，这些文件可以包含在CCS项目中在本附录中，假设读者熟悉使用MATLAB和SPTool建议您将MATLAB路径添加到已安装m文件的目录中

Exporting Direct-Form II Implementations

导出直接形式II实现

The filt structure created by MATLAB SPTool contains Direct-Form II numerator co- efficients in filt.tf.num and denominator coefficients in filt.tf.den. If the filter is designed as a finite impulse response (FIR) filter, then only the numerator coefficients need to be exported using fir\_dump2c.m for floating-point coefficients, or fir\_dump2c\_Qxx.m for fixed-point coefficients.

由MATLAB SPTool创建的滤波结构包含滤波器中的直接形式II分子系数TF过滤中的num和分母系数TF该如果滤波器设计为有限脉冲响应（FIR）滤波器，则只需使用fir\_dump2c输出分子系数m表示浮点系数，或fir\_dump2c\_Qxxm表示定点系数

To use fir\_dump2c.m, the following steps should be taken:

使用var\_dump 2cm，应采取以下步骤：

Export the filter design from SPTool to the workspace. Ensure that you have specified an FIR filter design. (The remainder of this procedure assumes that filter design was exported with name filt1.)

将过滤器设计从SPTool导出到工作区确保已指定FIR滤波器设计（此过程的其余部分假定使用名称filt1导出过滤器设计)

Execute a MATLAB cd command to change to the desired destination directory for the exported files.

执行MATLAB cd命令以更改为导出文件的所需目标目录

Run the m-file by typing fir\_dump2c(’coeff’,’B’,filt1.tf.num,length(filt1.tf.num)) at the MATLAB command line.

键入fir\_dump2c（'coeff'，'B'，filt1）运行m文件TFNUM，长度（FILT1TFnum））在MATLAB命令行

1If using another filter design method in MATLAB, such as FDATool, simply adapt the procedure for using the m-files as needed. The m-files will still eliminate the burden of converting from MATLAB variables into the format needed by the C language.

1如果在MATLAB中使用其他滤波器设计方法，例如FDATool，只需根据需要调整使用m文件的过程m文件仍然可以消除从MATLAB变量转换为C语言所需格式的负担

This creates two files, coeff.c and coeff.h, which declare a float array B of length

这会创建两个文件coeffc和coeffh，声明一个长度为float的数组B.

B\_SIZE.

B SIZE

To use fir\_dump2c\_Qxx.m, the following steps should be taken:

要使用fir\_dump2c\_Qxxm，应采取以下步骤：

Export the filter design from SPTool to the workspace. Ensure that you have specified an FIR filter design. (The remainder of this procedure assumes that filter design was exported with name filt1.)

将过滤器设计从SPTool导出到工作区确保已指定FIR滤波器设计（此过程的其余部分假定使用名称filt1导出过滤器设计)

Execute a MATLAB cd command to change to the desired destination directory for the exported files.

执行MATLAB cd命令以更改为导出文件的所需目标目录

Run the m-file by typing fir\_dump2c\_Qxx(’coeff’,’B’,filt1.tf.num,length(filt1.tf.num),15) at the MATLAB command line.

键入fir\_dump2c\_Qxx（'coeff'，'B'，filt1）运行m文件TFNUM，长度（FILT1TFnum），15）在MATLAB命令行

This creates two files, coeff.c and coeff.h, which declare a short array B of length B\_SIZE. The last parameter (Qxx) determines the location of the binary point. Fixed-point number representations are discussed in Appendix C.

这会创建两个文件coeffc和coeffh，声明一个长度为SIZE的短B数组最后一个参数（Qxx）确定二进制点的位置附录C中讨论了定点数表示

If the filter is designed as an infinite-impulse response (IIR) filter, then the numerator and denominator coefficients can either be exported individually using the methods de- scribed for FIR filters above, or both can be exported simultaneously using df2\_dump2c.m.

如果滤波器设计为无限脉冲响应（IIR）滤波器，那么分子和分母系数可以使用上面描述的FIR滤波器的方法单独导出，也可以使用df2\_dump2c同时导出两者米

To use df2\_dump2c.m, the following steps should be taken:

要使用df2\_dump2cm，应采取以下步骤：

Export the filter design from SPTool to the workspace. (The remainder of this proce- dure assumes that filter design was exported with name filt1.)

将过滤器设计从SPTool导出到工作区（此过程的其余部分假定使用名称filt1导出过滤器设计)

Execute a MATLAB cd command to change to the desired destination directory for the exported files.

执行MATLAB cd命令以更改为导出文件的所需目标目录

Run the m-file by typing df2\_dump2c(’HPF\_coeff’,’HPF’,filt1.tf) at the MATLAB command line.

键入df2\_dump2c（'HPF\_coeff'，'HPF'，filt1）运行m文件tf）在MATLAB命令行

This creates two files, HPF\_coeff.c and HPF\_coeff.h, which declare the float arrays HPF\_A of length HPF\_A\_SIZE (denominator coefficients), and HPF\_B of length HPF\_B\_SIZE (numer- ator coefficients). The array length is determined by the length of the filt1 numerator and denominator vectors.

这将创建两个文件HPF\_coeffc和HPF\_coeffh，声明浮点数HPF\_A的长度为HPF\_A\_SIZE（分母系数），HPF\_B的长度为HPF\_B\_SIZE（数值系数）数组长度由filt1分子和分母向量的长度确定

Exporting Second-Order Section Implementations

导出二阶段实现

An SPTool filter design can be converted from direct-form II to second-order sections by using the MATLAB function tf2sos (you may want to type help tf2sos in MATLAB for more details). Running tf2sos creates an L 6 matrix, where L is the number of second- order sections needed to implement the filter, with each row of the matrix containing the coefficients (b0, b1, b2, a0, a1, a2) for a single second-order section. These second-order section coefficients can be exported using sos\_dump2c.m.

通过使用MATLAB函数tf2sos，可以将SPTool滤波器设计从直接形式II转换为二阶段（您可以在MATLAB中键入help tf2sos以获取更多详细信息）运行tf2sos会创建一个L 6矩阵，其中L是实现滤波器所需的二阶段数，矩阵的每一行包含系数（b0，b1，b2，a0，a1，a2）一秒钟可以使用sos\_dump2c导出这些二阶截面系数米

To use sos\_dump2c.m, the following steps should be taken:

使用sos\_dump2cm，应采取以下步骤：

Export the filter design from SPTool to the workspace. (The remainder of this proce- dure assumes that filter design was exported with name filt1.)

将过滤器设计从SPTool导出到工作区（此过程的其余部分假定使用名称filt1导出过滤器设计)

Convert the filter design to second-order sections by typing

通过键入将过滤器设计转换为二阶段

filt1.sos=tf2sos(filt1.tf.num,filt1.tf.den)

毯sos = tf2sos（filt1TFnum，filt1TFDEN）

at the MATLAB command line.

在MATLAB命令行

400 APPENDIX E. RELATED TOOLS FOR DSKS

400附录E.相关工具用于桌子

Execute a MATLAB cd command to change to the desired destination directory for the exported files.

执行MATLAB cd命令以更改为导出文件的所需目标目录

Run the m-file by typing sos\_dump2c(’coeff’,’bqd\_coeff’,filt1.sos,size(filt1.sos,1)) at the MATLAB command line.

键入sos\_dump2c（'coeff'，'bad coeff'，filt1）运行m文件酱，你（FILT1sos，1））在MATLAB命令行

This creates two files, coeff.c and coeff.h, which declare the two-dimensional float array bqd\_coeff of size bqd\_coeff\_SIZE-by-5. The a0 coefficient is assumed to be 1 and is not used in the actual filter implementation, so it is ignored in the export.

这会创建两个文件coeffc和coeffh，声明大小为bqd\_coeff\_SIZE-by-5的二维浮点数组bqd\_coeff假设a0系数为1并且未在实际过滤器实现中使用，因此在导出中将忽略它

MATLAB Real-Time Interface

MATLAB实时接口

The MATLAB real-time interface is a software tool that permits MATLAB to interface directly with a DSK. Data can be imported from the DSK inputs into MATLAB variables, and variables can be written to the DSK outputs. The data transfer capabilities are limited by the bandwidth of the host PC to DSK connection, and the speed of the host computer. At lower sample frequencies, real-time behavior can be maintained. At higher sample frequencies, only a portion of the total codec data stream will be able to be transferred.

MATLAB实时接口是一个允许MATLAB直接与DSK接口的软件工具数据可以从DSK输入导入MATLAB变量，变量可以写入DSK输出数据传输能力受到主机PC到DSL连接的带宽以及主机速度的限制在较低的采样频率下，可以保持实时行为在较高的采样频率下，只能传输总编解码器数据流的一部分

An interface that allows the direct importation of real-time DSK data into MATLAB can be used for a number of purposes. The most basic approach is to simply use the DSK as a data acquisition board to obtain live data, and perform all signal processing in MATLAB. This also permits the use of the MATLAB visualization features with real-time DSK data. An interesting example of this approach was the development of real-time, acoustic beam- forming systems using multichannel analog input daughtercards on the DSK. Details of these projects are available in a number of references, including [40, 44, 50].

允许将实时DSK数据直接导入MATLAB的接口可用于多种用途最基本的方法是简单地使用DSK作为数据采集板来获取实时数据，并在MATLAB中执行所有信号处理这也允许使用MATLAB可视化功能和实时DSK数据这种方法的一个有趣的例子是在DSK上使用多通道模拟输入子卡开发实时声波束形成系统这些项目的详细信息可在许多参考文献中找到，包括[40,44,50]

The MATLAB real-time interface driver software and example MATLAB scripts are available in the Appendix E MatlabInterface directory. Detailed descriptions of the inter- face functions are available in the MatlabInterface\Matlab\_API.pdf document that can be found in the Appendix E subdirectory of the docs directory of the book’s software.

MATLAB实时接口驱动程序软件和示例MATLAB脚本可在附录E MatlabInterface目录中找到有关接口函数的详细说明，请参见Matlab接口\ Matlab APIpdf文档，可以在本书软件的docs目录的Appendix E子目录中找到

Appendix F

附录F.

Using the Code Generator with MATLAB

将代码生成器与MATLAB结合使用

Introduction

介绍

HIS book is based on the premise that first perfecting a DSP algorithm in MATLAB and then migrating the resulting m-file to real-time C code is a valuable skill. One of

本书基于这样的前提：首先在MATLAB中完善DSP算法，然后将生成的m文件迁移到实时C代码是一项宝贵的技能之一

the alternatives to developing this skill is to use an automated code-generation tool called

开发此技能的替代方法是使用名为的自动代码生成工具

TM

TM值

the MATLAB Coder to convert a MATLAB function “auto-magically” into C code. Using

MATLAB编码器将MATLAB函数“自动神奇地”转换为C代码运用

a code generator can potentially save time, but the user has much less control over the form of the software created, and will probably have less understanding of the generated software. If errors crop up, this can make debugging quite difficult. Ultimately, the decision of which approach to use is yours. The authors are strong advocates of the first approach, but we didn’t want to completely ignore the MATLAB Coder option in case some of our readers are interested in that approach.

代码生成器可以节省时间，但是用户对创建的软件形式的控制要少得多，并且可能对生成的软件的理解较少如果出现错误，这可能会使调试变得非常困难最终，决定使用哪种方法作者是第一种方法的强烈支持者，但是如果我们的一些读者对这种方法感兴趣，我们不想完全忽略MATLAB Coder选项。

The MATLAB Coder generates standalone C and C++ code from MATLAB code. The generated source code is portable and readable. The MATLAB Coder supports a subset of core MATLAB language features, including program control constructs, functions, and matrix operations. It can also generate MEX functions that let you accelerate computa- tionally intensive portions of MATLAB code and verify the behavior of the generated code. See for more details.

生成的源代码是可移植且可读的MATLAB编码器支持核心MATLAB语言功能的子集，包括程序控制结构，函数和矩阵运算它还可以生成MEX函数，使您可以加速MATLAB代码的计算密集部分并验证生成的代码的行为有关详细信息，请参阅

The MATLAB Coder is a separate MathWorks product and as such, an additional

MATLAB Coder是一个单独的MathWorks产品，因此是另外一个

expense. During its installation, you will be reminded that you must also install one of the supported C/C++ compilers. For MATLAB Release 2016a, the list of supported compilers can be found at

费用对于MATLAB Release 2016a，可以在以下位置找到支持的编译器列表

An FIR Filter Example

FIR滤波器示例

Before Using the MATLAB Coder

在使用MATLAB编码器之前

In Chapter 3, we developed a brute force approach to FIR filtering using MATLAB and the resulting code was shown in Listing 3.2. The code shown in Listing F.1 is a functionalized version of Listing 3.2. The FIR filtering function used in Listing F.1 is shown in Listing F.2. The m-script names include the letters “fun” which is a reference to the fact that these files are function-based.

在第3章中，我们使用MATLAB开发了一种用于FIR滤波的强力方法，结果代码如清单3所示2清单F中显示的代码1是清单3的功能化版本2清单F中使用的FIR过滤函数清单F中显示了12m脚本名称包括字母“fun”，它是对这些文件基于函数的事实的引用

401

401

402 APPENDIX F. USING THE CODE GENERATOR WITH MATLAB

402附录F.使用MATLAB代码生成器

Listing F.1: The funFIR.m code, a functionalized version of Listing 3.2.

清单F.1：funFAIRm代码，清单3的功能化版本2

1 This m-file is used to convolve xLeft[n] and B[n] without using the MATLAB filter command. This is one of the

1此m文件用于在不使用MATLAB过滤器命令的情况下对xLeft [n]和B [n]进行卷积这是其中之一

3 first steps toward being able to implement a real-time

3能够实现实时的第一步

FIR filter in DSP hardware. This m-file uses a function to

DSP硬件中的FIR滤波器这个m文件使用了一个函数

5 calculate the output value, yLeft[0].

5计算输出值yLeft [0]

7 In sample -by-sample filtering , you are only trying to accomplish 2 things ,

7在逐个样本过滤中，您只是尝试完成两件事，

9

9

Calculate the current output value, yLeft[0], based on

根据计算当前输出值yLeft [0]

11 just having received a new input sample , xLeft[0].

11刚收到一个新的输入样本，xLeft [0]

Setup for the arrivial of the next input sample.

设置下一个输入样本的到达

13

13

This is a BRUTE FORCE approach!

这是一种强制性的方法！

15

15

written by Dr. Thad B. Welch , PE {t.b. }

博士写的Thad B.Welch，PE {tb}

17 copyright 2001 , 2015

17版权所有2001,2015

completed on 13 December 2001 revision 1.0

于2001年12月13日完成修订10

19 updated to a function -based script on 21 July 2015 rev 1.1

19更新为2015年7月21日rev 1的基于功能的脚本1

21 Simulation inputs

21模拟输入

xLeft = single ([ 1 2 3 0]) ; input vector xLeft

左=单（[1 2 3 0]）;

23 N = int16 ( 3) ; order of filter= length(B)-1 B = single ( [ 0. 25 0 . 25 0 . 25 0. 25] ) ; FIR filter coefficients B[n]

23 N = int16（3）;25 025 025 025]）;

25 yLeft = single ( 0) ; declare output variable y

25左=单（0）;

27 Calculated terms ( functionalized)

27计算项（功能化）

[ xLeft , yLeft ] = funFilter ( xLeft , B , N ) ;

[xLeft，yLeft] = funFilter（xLeft，B，N）;

29

29

Simulation outputs

模拟输出

31 xLeft notice xLeft(1) = xLeft(2)

31 x左通知左（1）=左（2）

yLeft average of last 4 input values

Lyft平均最后4个输入值

Listing F.2: The funFilter.m code, which defines the function used in Listing F.1.

清单F.2：有趣的过滤器m代码，它定义了清单F中使用的函数1

2

2

4

4

6

6

8

8

10

10

12

12

14

14

To allow for easy comparison between the real-time C code from Listings 3.3 and 3.4, and the C code that we are about to generate, a few code modifications were made. Specifically, the modifications are listed next.

为了便于比较清单3中的实时C代码3和34，以及我们即将生成的C代码，进行了一些代码修改具体而言，接下来列出了修改

The variable x was renamed xLeft, and the variable y was renamed yLeft.

变量x重命名为cLeft，变量y重命名为cLeft

All of the variable types were declared, xLeft as single, N as int16, B as single, and

声明了所有变量类型，xLeft为单，N为int16，B为单，和

yLeft as single.

留下单身

The FIR filter routine was made into a separate MATLAB function and given the name funFilter.m. Remember, there is already a MATLAB function named filter and you must avoid using existing function or variable names to represent more than one thing. If you are not sure if a variable or function name that you are considering using exists, use the MATLAB which command (for example, which filter).

FIR滤波器程序被制作成一个单独的MATLAB函数，并命名为funFilter米请记住，已经有一个名为filter的MATLAB函数，您必须避免使用现有函数或变量名来表示多个函数如果您不确定是否存在正在考虑使用的变量或函数名称，请使用MATLAB which命令（例如，哪个过滤器）

If you run the funFIR.m script, you will get the same results as in Chapter 3, as shown below.

如果你运行funFAIRm脚本，您将得到与第3章相同的结果，如下所示

xLeft =

cLeft =

1 1 2 3

1 1 2 3

yLeft =

yLeft =

1.5000

15000

Using the MATLAB Coder

使用MATLAB编码器

Now the Coder tool can be used. From the MATLAB command line, type coder and then press the “Enter” key. A dialog box similar to the figure below will appear.

现在可以使用编码器工具在MATLAB命令行中，键入coder，然后按“Enter”键将出现类似下图的对话框

Notice the progress indicator near the top of this dialog box. We are currently in the first of six steps which is labeled, Select. Use the browse option (indicated by the ellipsis “. . . ” to the right of the highlighted window) to find and enter funFilter.m which contains our FIR filter function. This will result in a dialog box similar to the next figure shown.

请注意此对话框顶部附近的进度指示器我们目前处于标记为Select的六个步骤中的第一步使用浏览选项（由省略号表示““在突出显示的窗口右侧”查找并输入funFilterm包含我们的FIR滤波器功能这将生成一个类似于下图所示的对话框

Since we have run this process a number of times, a yellow highlighted box has appeared to remind us that this project already exists. We will select the option to Overwrite the project. After clicking on Overwrite, the yellow highlighted information will disappear and we are ready for the next step. This is indicated by the flashing Next in the lower right corner. Clicking on Next will result in a dialog box similar to the next figure shown.

由于我们已多次运行此过程，因此黄色突出显示的框似乎提醒我们该项目已存在我们将选择覆盖项目的选项单击“覆盖”后，黄色突出显示的信息将消失，我们已准备好进行下一步这通过右下角的闪烁Next表示单击Next将生成一个类似于下图所示的对话框

We have chosen to enter the name of the script funFIR.m so that the resulting code will be very similar to what we created in Chapter 3. Use the browse option (indicated by the

我们选择输入脚本funFAIR的名称m以便生成的代码与我们在第3章中创建的代码非常相似使用浏览选项（由表示

ellipsis “. . . ” to the right) to find and enter funFIR.m, which sets up our filtering example and calls our FIR filter function. Click on the Autodefine Input Types button. This will result in a dialog box similar to the next figure shown.

省略号““向右”找到并进入funFAIRm，它设置我们的过滤示例并调用我们的FIR滤波器函数单击“自动定义输入类型”按钮这将生成一个类似于下图所示的对话框

Click on Next. This will result in a pop-up box similar to the next figure shown.

单击“下一步”这将导致弹出框类似于下图所示

Click on Check for Issues. This will result in a dialog box similar to the next figure shown.

单击“检查问题”这将生成一个类似于下图所示的对话框

Click on SETTINGS, then click on Speed (in the left column). This will result in a dialog box similar to the next figure shown.

单击SETTINGS，然后单击Speed（在左栏中）这将生成一个类似于下图所示的对话框

Uncheck the boxes for Saturate on integer overflow and Support non-finite numbers. Click on Close and click on the CHECK FOR ISSUES button. This will result in a pop-up box similar to the next figure shown.

取消选中Saturate on integer overflow和支持非限定数字的方框单击关闭，然后单击CHECK FOR ISSUES按钮这将导致弹出框类似于下图所示

Click on Next in the lower right corner. This will result in a dialog box similar to the next figure shown.

单击右下角的“下一步”这将生成一个类似于下图所示的对话框

Under Production Hardware, pulldown the Device vendor menu and select Texas Instruments. Pulldown the Device type menu and select C6000. Click on the Generate button. This will result in something similar to the next figure shown.

在Production Hardware下，下拉Device vendor菜单并选择Texas Instruments下拉设备类型菜单，然后选择C6000单击Generate按钮这将导致类似于下图所示的内容

Scrolling down in the listing, the important results are shown in the next figure shown.

在列表中向下滚动，重要结果如下图所示

We are now ready to use the pertinent parts of the auto-generated code in a CCS project for real-time execution.

我们现在准备在CCS项目中使用自动生成代码的相关部分进行实时执行

Transferring to a CCS Project

转移到CCS项目

We will now copy and paste the appropriate section of the auto-generated code listing from the previous figure into our real-time project in CCS. Specifically, we will need to use the following parts.

我们现在将上图中自动生成的代码清单的相应部分复制并粘贴到CCS中的实时项目中具体来说，我们需要使用以下部分

Paste lines 25 and 26 from the previous figure into the variable declarations section of the ISR in the CCS project.

将上图中的第25行和第26行粘贴到CCS项目中ISR的变量声明部分

Paste lines 27–37 from the previous figure into the algorithm section of the ISR in the CCS project.

将上图中的第27-37行粘贴到CCS项目中ISR的算法部分

The real-time project is now almost ready to run, using the C code generated by the MATLAB Coder.

使用MATLAB Coder生成的C代码，现在几乎可以运行实时项目了

Observations

意见

After following this procedure, there are a few things to notice.

遵循此程序后，有几点需要注意

The MATLAB Coder created a number of files (11 by our count).

MATLAB Coder创建了许多文件（按我们的计算为11）

We will use none of those 11 files!

我们将不使用这11个文件！

We only need to extract the variable declarations and algorithm parts from the

我们只需从中提取变量声明和算法部分

funFilter.c file.

有趣的筛选c文件

Lines 25 and 26 in the auto-generated code only declared yLeft and i. The variables

自动生成的代码中的第25行和第26行仅声明为Left和i变量

xLeft and B will also need to be declared before the project can be run in CCS.

在项目可以在CCS中运行之前，还需要声明Left和B.

Another minor observation you may notice, when comparing the real-time C code from Chapter 3 with the code generated by the MATLAB Coder, is that MATLAB appears to have trouble accepting the fact that in C code, a memory location with an index of zero is perfectly valid. This probably stems from the original FORTRAN legacy of MATLAB, which is why even today MATLAB only allows array index values to start at 1. When MATLAB was rewritten in C++ many years ago, this behavior was maintained to keep from “breaking” any existing MATLAB code. Indeed, one of the strengths of MATLAB is how well backward compatibility is maintained. The authors have m-files written over 25 years ago that still run without error on the latest version of MATLAB; very few software tools can claim that.

在将第3章的实时C代码与MATLAB编码器生成的代码进行比较时，您可能会注意到的另一个小观察是MATLAB似乎无法接受在C代码中索引为零的内存位置这一事实。这可能源于MATLAB的原始FORTRAN遗留，这就是为什么即使在今天MATLAB只允许数组索引值从1开始实际上，MATLAB的优势之一是如何保持向后兼容性作者拥有超过25年前编写的m文件，这些文件在最新版本的MATLAB上仍可正常运行;

Conclusion

结论

Is the MATLAB Coder for you? If you are willing to develop your MATLAB algorithm as a function, declare your variables in your MATLAB code as if you were using a declarative language (such as C), develop a script file that calls this function, and then reintegrate the resulting C code into your real-time code’s ISR file, then this approach may work for you. As mentioned at the start of this appendix, the authors are strong advocates of first perfecting your DSP algorithm in MATLAB, and then manually migrating the resulting m-file to real-time C code. In our opinion, this results in a better understanding of the

是MATLAB编码器吗？正如本附录开头所述，作者强烈主张首先在MATLAB中完善您的DSP算法，然后手动将生成的m文件迁移到实时C代码我们认为，这可以更好地理解

real-time code.

实时代码

Appendix G

附录G.

Appendix G

附录G.

Battery Power for the DSP Boards

DSP板的电池电量

Introduction

介绍

OME users may desire to operate their DSP board on battery power. Possible reasons for this would certainly include portability, but also electrical isolation from the AC power system. The latter may be desirable, for example, to provide increased patient safety when the DSP board input is connected to a biomedical signal source such as an electrocardiogram (ECG) provided via surface electrodes and appropriate bioinstrumentation buffer circuitry.

一些用户可能希望用电池电源操作他们的DSP板可能的原因当然包括便携性，以及交流电源系统的电气隔离后者可能是合乎需要的，例如，当DSP板输入连接到生物医学信号源（例如通过表面电极和适当的生物仪器缓冲电路提供的心电图（ECG））时提供增加的患者安全性。

We have proven that it is quite feasible to use any of the DSP boards supported by this text with battery power.

我们已经证明，使用本文支持的任何DSP板都可以使用电池供电

All of the supported DSP boards are powered by an AC power supply that converts wall outlet power to approximately 5 volts DC. These boards connect to the AC power supply via a DC power plug, which is also called a barrel connector. The outer portion of the barrel is 5.5 mm in diameter (ground) and the inner post is 2.5 mm in diameter (+5 volts DC). The approximate quiescent currents for these boards are shown in Table G.1. When the DSP core is performing low-intensity calculations, these currents will rise slightly (10–20%). Higher-intensity calculations will draw higher amounts of current; this will very much be application dependent, so investigate your particular application.

所有支持的DSP板均由交流电源供电，可将墙上插座电源转换为约5伏直流电这些板通过直流电源插头连接到交流电源，直流电源插头也称为桶形连接器枪管的外部是5直径5毫米（地面），内柱2这些电路板的近似静态电流如表G所示1当DSP内核执行低强度计算时，这些电流将略微上升（10-20％）更高强度的计算会吸收更多的电流;

Method

方法

All of the DSP boards supported by this text operate with a modest amount of current, at +5 volts, as mentioned above. This allows any of the boards to be easily operated independent of AC power. Since the required currents and voltages are within the capability

这使得任何电路板都可以独立于交流电源轻松操作由于所需的电流和电压在能力范围内

Table G.1: Typical quiescent current drawn by DSP boards. The quiescent state is the condition when minimal processing is being performed by the board.

表G.1：DSP板吸收的典型静态电流静止状态是电路板执行最小处理的条件

411

411

412 APPENDIX G. BATTERY POWER FOR THE DSP BOARDS

412附录G.DSP板的电池电量

of most USB-based external batteries, one possible solution is to either buy or assemble a cable that will connect such an external battery to the DSP board. Both options are described here.

对于大多数基于USB的外部电池，一种可能的解决方案是购买或组装将这种外部电池连接到DSP板的电缆这两个选项都在这里描述

Buy a cable: Look for a cable listed as a “USB 2.0 A to 5.5/2.5mm Barrel Connector Jack DC Power Cable” or something similar to that.

购买电缆：寻找列为“USB 2”的电缆0到55/25毫米枪管连接器插孔直流电源线“或类似的东西

Assemble a cable: If you would rather build your own cable, follow these steps. We used two spare cables that we had on hand, and cut off the unneeded portion of the cables. The USB end of the cable to be assembled is shown in Figure G.1(a). The red and black wires correspond to the +5 volt DC and ground connection of the USB specification, respectively. The green and white wires are used for data transfer, and are therefore not used in this application. The DC end of the cable to be assembled is shown in Figure G.1(b). The red wire of the USB end needs to be soldered to whichever wire on the DC end connects to the center part of the DC power plug, and the black wire of the USB end needs to be soldered to the other wire on the DC end. Since color codes or stripes may not be consistent on such DC cables, we used an ohmmeter to verify the wiring of these two connections. After these two connections are soldered together, the exposed conductors need to be properly insulated. We used electrical tape for this purpose; alternatively, heat shrink tubing could have been used.

组装电缆：如果您希望自己制作电缆，请按照以下步骤操作我们使用了手头的两根备用电缆，并切断了不需要的电缆部分要组装的电缆的USB端如图G所示图1（a）绿色和白色线用于数据传输，因此不在本申请中使用要组装的电缆的DC端如图G所示图1（b）USB端的红线需要焊接到直流端的任何一根导线连接到直流电源插头的中心部分，而USB端的黑线需要焊接到直流端的另一根导线上由于此类直流电缆上的颜色代码或条纹可能不一致，我们使用欧姆表来验证这两个连接的接线在将这两个连接焊接在一起之后，暴露的导体需要适当地绝缘我们为此目的使用电工胶带;

The USB end of a cable to be assembled. This end will connect to the external battery.

要组装的电缆的USB端此端将连接到外部电池

The DC end of a cable to be assembled. This end will connect to the DSP board.

待组装电缆的直流端此端将连接到DSP板

Figure G.1: The two ends of a cable for connecting a DSP board to a USB external battery.

图G.1：用于将DSP板连接到USB外部电池的电缆的两端

Figure G.2: An example of a DSP board connected to an external USB battery. The board can operate with no connection to AC power.

图G.2：连接到外部USB电池的DSP板的示例电路板可以在不连接交流电源的情况下运行

Testing

测试

Initial testing

初步测试

After assembly, we connected the cable to our USB battery and observed the output voltage using an oscilloscope (a multimeter could also be used). This step verifies that you have the correct polarity and that your battery is actually providing about +5 volts DC as required. This initial check is highly recommended before connecting the cable to your DSP board.

组装完成后，我们将电缆连接到USB电池，并使用示波器观察输出电压（也可以使用万用表）在将电缆连接到DSP板之前，强烈建议进行初始检查

Final testing

最终测试

Once the cable is verified, connect the cable between the USB battery and the DSP board. If you are using the OMAP-L138 board, you will also need to power up the system by turning on the slide switch. A fully assembled and working system is shown in Figure G.2. The Hyperjuice 100 watt-hour (20 amp-hour) system is shown, which should power the OMAP-L138 board for about three days. We operated the system for more than a day without difficulty!

验证电缆后，连接USB电池和DSP板之间的电缆如果您使用的是OMAP-L138板，则还需要打开滑动开关来打开系统电源完全组装和工作的系统如图G所示2显示Hyperjuice 100瓦时（20安培小时）系统，该系统应为OMAP-L138板供电约三天我们操作系统超过一天没有困难！

Conclusion

结论

Using the method described in this appendix, you can gain increased portability of your DSP board, and also isolate it from the AC power systems. The cable can also power the DSP board from your computer’s USB (type A) port, if desired. This allows you to eliminate one of the power cords needed for your real-time DSP development station.

使用本附录中描述的方法，您可以提高DSP板的可移植性，并将其与交流电源系统隔离如果需要，电缆还可以从计算机的USB（A型）端口为DSP板供电这使您可以消除实时DSP开发站所需的电源线之一

Appendix H

附录H.

Programming Perils and Pitfalls

编程危险和陷阱

ROGRAMMING in a real-time environment can be challenging even for experienced programmers. This appendix is intended to illustrate some of the common problems that are encountered in this environment, and present practical strategies for avoiding them.

即使对于有经验的程序员来说，在实时环境中编程也是一项挑战本附录旨在说明在此环境中遇到的一些常见问题，并提出避免这些问题的实用策略

Debug versus Release Builds

调试与发布版本

When a project is created in Code Composer Studio, there are two build configurations that are established: Debug and Release. The debug configuration will embed debugging information in the object file (information that links assembly instructions to the original source code), and also will not optimize the generated code so that there is a direct corre- spondence between a line of source code and the assembly language that is generated. These permit symbolic debugging, and ensure that the assembly code will execute in the order that the C code was written. The debug configuration is useful when developing software, but the generated code is often significantly slower than the release version. In the release configuration, the compiler attempts to optimize the generated code for best performance, using a number of transformations and algorithms. This means that there may no longer be a 1-to-1 correspondence between the source code and the assembly code; as functionality is moved and reordered, code is reused where possible, and redundancies are eliminated in order to maximize execution speed and/or minimize code size. Debugging the assembly code generated by a release build is a significant challenge for even seasoned programmers. The types and degrees of optimization employed can be controlled on a per-project and per-file basis; further information on this is available in the CCS documentation.

在Code Composer Studio中创建项目时，会建立两种构建配置：调试和发布调试配置将调试信息嵌入到目标文件中（将汇编指令链接到原始源代码的信息），也不会优化生成的代码，以便在一行源代码和汇编之间存在直接的相关性这些允许符号调试，并确保汇编代码将按照编写C代码的顺序执行调试配置在开发软件时很有用，但生成的代码通常比发布版本慢得多在发布配置中，编译器尝试使用大量转换和算法优化生成的代码以获得最佳性能这意味着源代码和汇编代码之间可能不再存在一对一的对应关系;调试由发布版本生成的汇编代码对于经验丰富的程序员来说是一个重大挑战可以基于每个项目和每个文件来控制所采用的优化类型和程度;

The Volatile Keyword

易变的关键字

Two common situations in real-time DSP programming are variables that directly reference hardware, and variables that are used to communicate between interrupt service routines and the main program. Both of these situations require that the volatile keyword be used to control the compiler’s optimization of memory references. In the first case, when a pointer variable is dereferenced to access a hardware register, the compiler’s optimizer will assume that the transfer is being made to a standard read/write memory location. In the second case, the compiler will assume that the memory location will only change when it is written to in the function being compiled, and that no other access will be made concurrent with that function’s execution. For example, suppose there is an integer pointer variable

实时DSP编程中的两种常见情况是直接引用硬件的变量，以及用于中断服务程序和主程序之间通信的变量这两种情况都要求使用volatile关键字来控制编译器对内存引用的优化在第一种情况下，当取消引用指针变量以访问硬件寄存器时，编译器的优化器将假定正在传输到标准读/写存储器位置在第二种情况下，编译器将假定内存位置仅在被编译的函数中写入时才会更改，并且不会与该函数的执行同时进行其他访问例如，假设有一个整数指针变量

415

415

416 APPENDIX H. PROGRAMMING PERILS AND PITFALLS

416附录H.编程危险和坑洼

mcbsp spcr that is used to read the McBSP1 receive register status bit as shown below, in order to wait until data is received by the McBSP.

mcbsp spcr，用于读取McBSP1接收寄存器状态位，如下所示，以便等待直到McBSP接收到数据

2

2

4

4

In the debug build, the code executes as expected because no optimization is done. However, in the release build, the compiler identifies the expression !(\*mcbsp spcr & 0x00020000) to be loop-invariant code, and so pulls it out of the loop and only reads from that location once. While this is generally a very good optimization for true memory locations, in this case we are reading a peripheral register that may change; thus, it is NOT loop-invariant. To force the compiler to actually read the location represented by the variable in each loop iteration, we add the qualifier volatile to the variable declaration as shown below.

在调试版本中，代码按预期执行，因为没有进行优化虽然这通常是对真实存储器位置的非常好的优化，但在这种情况下，我们正在读取可能改变的外设寄存器;为了强制编译器在每次循环迭代中实际读取变量表示的位置，我们将限定符volatile添加到变量声明中，如下所示

The volatile keywords informs the compiler that it may not optimize out any accesses to this

volatile关键字通知编译器它可能不会优化对此的任何访问

variable, so it generates code that actually reads the McBSP register in each iteration. A similar situation occurs when a variable is used to write repeatedly to a hardware register, in this case McBSP1’s transmit data register, as illustrated below.

变量，因此它生成的代码实际上在每次迭代中读取McBSP寄存器当变量用于重复写入硬件寄存器（在本例中为McBSP1的发送数据寄存器）时会出现类似的情况，如下图所示

1

1

3

3

5

5

If this code is compiled under a release build, only a single write of the value 3 will occur. Declaring mcbsp dxr to be volatile will force the compiler to perform the three separate write operations.

如果此代码是在发布版本下编译的，则只会发生一次值3的写入将mcbsp dxr声明为volatile将强制编译器执行三个单独的写操作

When using global variables to communicate between the main program and interrupt service routines (or between interrupt service routines), you should normally declare the global variables to be volatile as well, particularly if they are used in loops. Otherwise, the compiler may optimize out the variable references and you will miss any changes that occur to the variable during an interrupt.

当使用全局变量在主程序和中断服务程序之间（或中断服务程序之间）进行通信时，通常应该声明全局变量也是易变的，特别是如果它们在循环中使用否则，编译器可能会优化变量引用，并且您将错过在中断期间对变量发生的任何更改

Function Prototypes and Return Types

函数原型和返回类型

If a function is not declared before it is used, the C language requires that the compiler assume that function return type is int. This seemingly benign behavior has been observed as the cause of many programs that fail to work properly, because failing to declare a function in C is not an error and so it is not flagged as such. Consider the code below, noting that the sinf function was never declared.

如果函数在使用之前未声明，则C语言要求编译器假定函数返回类型为int这种看似良性的行为已被观察为许多程序无法正常工作的原因，因为未能在C中声明函数不是错误，因此它不会被标记为考虑下面的代码，注意到sinf函数从未声明过

1

1

3

3

In the C6000 architecture, register A4 is used for the return of 32-bit or smaller values. The sinf function actually returns a single-precision floating-point number in A4. However, since the function was never declared, the compiler assumes that the return type is int. So, the compiler assumes A4 contains an integer, and adds code (specifically the INTSP

在C6000架构中，寄存器A4用于返回32位或更小的值sinf函数实际上返回A4中的单精度浮点数但是，由于函数从未声明，因此编译器假定返回类型为int因此，编译器假定A4包含一个整数，并添加代码（特别是INTSP

instruction) to convert the value in it to a float before storing it in x. As can be imagined, taking a floating-point bit pattern and performing an integer to floating-point conversion on it produces meaningless values. To prevent this situation, it is important (and good programming practice) to declare all functions before use. In the code below, the math.h header has been included to ensure the proper declaration of the sinf function.

指令）在将其存储在x中之前将其中的值转换为浮点数可以想象，采用浮点位模式并对其执行整数到浮点转换会产生无意义的值为了防止这种情况，在使用之前声明所有函数是很重要的（并且良好的编程习惯）在下面的代码中，数学已包含h标头以确保正确声明sinf函数

Now knowing that the sinf function returns a float, the compiler will take the return value in register A4 and transfer it into the variable x directly, giving the correct result. Always ensuring that all functions are declared before use will avoid these situations. This can be quite difficult to debug because the code is otherwise correct.

现在知道sinf函数返回一个浮点数，编译器将获取寄存器A4中的返回值并将其直接传递给变量x，得到正确的结果始终确保在使用前声明所有功能将避免这些情况这可能很难调试，因为代码是正确的

Arithmetic Issues

算术问题

A high-level language compiler typically supports a number of arithmetic operations. The compiler will guarantee correct results; however, in real-time software we are also concerned with how long it will take to do the calculation. For an operation that is supported in the processor (i.e., add), the compiler will generate code to use the hardware to perform the calculation. For operations that are not supported in the processor hardware, the compiler will generate software to accomplish the calculation. In general, the software calculations will be much slower than those performed in hardware.

高级语言编译器通常支持许多算术运算编译器将保证正确的结果;对于处理器支持的操作（i和，add），编译器将生成代码以使用硬件来执行计算对于处理器硬件不支持的操作，编译器将生成软件以完成计算通常，软件计算将比硬件中的计算慢得多

The TMS320C6x DSPs do not have divider hardware, so division should be avoided whenever possible. In the code below, the calculations are numerically equivalent.

TMS320C6x DSP没有分频器硬件，因此应尽可能避免分频在下面的代码中，计算在数值上是等价的

2

2

4

4

Calculation A specifies a division, so the compiler will insert a call to a subroutine to perform the calculation in software. Since the processor has a hardware multiplier, the calculation B can be accomplished much faster. Note that the “F” suffix on the numbers indicates that they are constants of type float — otherwise they would be interpreted as type double, requiring the promotion of x to type double before performing the calculation as a double-precision operation.

计算A指定一个除法，因此编译器将插入对子程序的调用以在软件中执行计算由于处理器具有硬件乘法器，因此可以更快地完成计算B.请注意，数字上的“F”后缀表示它们是float类型的常量 - 否则它们将被解释为double类型，在将计算作为双精度运算执行之前，需要将x提升为double类型

When using an array variable to implement circular buffering, the index needs to be “wrapped around” when the end of the buffer is reached. In the code sample below, a buffer and an index variable are allocated.

当使用数组变量实现循环缓冲时，索引需要在到达缓冲区末尾时“缠绕”在下面的代码示例中，分配了缓冲区和索引变量

2

2

In the following examples, we will assume the index value is being incremented. A decrementing index would be handled in a similar fashion. Perhaps the most immediately intuitive way to accomplish index wraparound is to simply check the index value and set it back to 0 when it reaches the end of the buffer.

在以下示例中，我们假设索引值正在递增递减索引将以类似的方式处理也许最简单直观的方法来完成索引回绕就是简单地检查索引值并在它到达缓冲区末尾时将其设置回0

1

1

3

3

Note that this requires a comparison, and is limited to index increments of 1. If arbitrary increments are needed, we need another approach. The modulus operation (%) provides a seemingly simple fix.

请注意，这需要进行比较，并且仅限于索引增量1如果需要任意增量，我们需要另一种方法模数运算（％）提供了看似简单的修正

1

1

However, the modulus operator computes the remainder of the index value divided by BUFFER SIZE, so we are implicitly invoking a division operation. As an alternative, note that if the increment is less than BUFFER SIZE, we can obtain the remainder by subtracting BUFFER SIZE whenever the index is greater than or equal to BUFFER SIZE.

但是，模数运算符计算索引值的剩余部分除以BUFFER SIZE，因此我们隐式调用除法运算作为替代，请注意，如果增量小于BUFFER SIZE，我们可以通过在索引大于或等于BUFFER SIZE时减去BUFFER SIZE来获得余数。

2

2

This reduces the modulus calculation to a simple subtraction operation, which is sup- ported in hardware. However, it still requires a comparison to see if the index needs to be wrapped around. In real-time code, we may find even this operation to be prohibitively expensive. To eliminate the comparison completely, the buffer size is set to 2n. Then, the wraparound is accomplished with only a logical AND operation of the index with 2n 1. If the index is less than BUFFER SIZE, the AND operation will leave it unchanged. If the index is greater than or equal to BUFFER SIZE, the AND operation will result in the same result as the modulus operation.

这将模数计算减少为简单的减法运算，这在硬件中得到支持但是，它仍然需要进行比较，以查看是否需要包含索引在实时代码中，我们甚至可能发现这种操作非常昂贵为了完全消除比较，缓冲区大小设置为2n然后，仅使用2n 1的索引的逻辑AND运算来完成环绕如果索引小于BUFFER SIZE，则AND操作将保持不变如果索引大于或等于BUFFER SIZE，则AND操作将导致与模数运算相同的结果

1 #define BUFFER SIZE 512 // must be a power of 2

float x [ BUFFER\_SIZE ] = 0. 0 F ;

float x [BUFFER\_SIZE] = 00楼;

3 int index = 0;

3 int index = 0;

5 index++;

index = index & ( BUFFER\_SIZE − 1) ;

Note that in this implementation the buffer size must be a power of 2. This is a classic software trade-off between size and speed, and is often seen in production code.

请注意，在此实现中，缓冲区大小必须是2的幂这是尺寸和速度之间的经典软件权衡，常见于生产代码中

Controlling the Location of Variables in Memory

控制内存中变量的位置

When declaring a variable in software, we normally do not concern ourselves with the actual variable location in memory. Rather, we simply refer to the variable by name. The compiler and linker are responsible for making sure that the correct memory location(s) are accessed. However, there are times when we will want to control where variables are placed in memory. To do that, we need to do two things:

在软件中声明变量时，我们通常不关心内存中的实际变量位置相反，我们只是按名称引用变量编译器和链接器负责确保访问正确的内存位置但是，有时我们会想要控制变量放在内存中的位置要做到这一点，我们需要做两件事：

instruct the linker where the physical memory is in our system, and

指示链接器物理内存在我们系统中的位置，以及

tell the compiler which variables we want placed in locations other than the default locations.

告诉编译器我们想要将哪些变量放在默认位置以外的位置

When our code is compiled, the compiler places the output into a number of predefined sections. Global variables are typically placed in the .data or .bss sections. The linker command file (i.e., lnk6748.cmd) lists the physical memory available in the system, and indicates which sections are placed into which memory areas. In the linker command files used throughout the text, all compiler output is placed into the DSP’s on-chip memory (the IRAM area). This is a relatively small memory area, so if we want to have large data buffers we need to place them in the much larger off-chip memory. In the linker command

编译代码时，编译器会将输出放在许多预定义的部分中全局变量通常放在数据或bss部分链接器命令文件（i和, لنكطح٤٨cmd）列出了系统中可用的物理内存，并指出哪些部分放在哪些内存区域中在整个文本中使用的链接器命令文件中，所有编译器输出都被放入DSP的片上存储器（IRAM区域）这是一个相对较小的内存区域，所以如果我们想拥有大数据缓冲区，我们需要将它们放在更大的片外内存中在链接器命令中

file, this area is designated SDRAM ; all compiler output in section “CE0” will be placed there.

文件，此区域指定为SDRAM;

To instruct the compiler to place a given variable into the “CE0” section, we use a compiler pragma. In general, pragmas are compiler-specific directives that permit detailed control over various aspects of the compiler’s operation. To control the section into which a variable is placed, we can use the DATA\_SECTION pragma. This instructs the compiler to place the variable named as the first parameter into the section named as the second parameter.

为了指示编译器将给定变量放入“CE0”部分，我们使用编译器编译指示通常，编译指示是特定于编译器的指令，允许对编译器操作的各个方面进行详细控制要控制放置变量的部分，我们可以使用DATA\_SECTION编译指示这指示编译器将名为第一个参数的变量放入名为第二个参数的节中

2

2

Further information on the various pragmas available in Code Composer Studio can be found in the online help and the C compiler user’s manual.

有关Code Composer Studio中可用的各种编译指示的更多信息，请参见联机帮助和C编译器用户手册

Real-Time Schedule Failures

实时计划失败

One of the most difficult challenges in writing real-time software is determining if the soft- ware will in fact be able to meet the real-time schedule. In particular, for an interrupt driven system, each interrupt service routine (ISR) must complete its processing before the next interrupt occurs, and the programmer must allow sufficient “slack time” to account for interrupt service overhead. One simple and effective way to measure the time that an ISR takes is to change the state of a logic signal on entering and leaving the ISR, and then monitor that signal with an oscilloscope. The WriteDigitalOutputs() function allows this to be done easily by setting the state of four digital signals on the DSKs, as shown below.

编写实时软件最困难的挑战之一是确定软件是否实际上能够满足实时计划特别是，对于中断驱动系统，每个中断服务程序（ISR）必须在下一个中断发生之前完成其处理，并且程序员必须允许足够的“松弛时间”来解决中断服务开销测量ISR所需时间的一种简单有效的方法是在进入和离开ISR时改变逻辑信号的状态，然后用示波器监视该信号WriteDigitalOutputs（）函数允许通过设置DISK上的四个数字信号的状态轻松完成，如下所示

Example code is shown below.

示例代码如下所示

Listing H.1: Checking for real-time schedule failure using the WriteDigitalOutputs()

清单H.1：使用写入数字输出检查实时计划故障（）

function.

功能

2

2

4

4

6

6

The approximate percentage of CPU time spent in the ISR is then approximately the duty cycle of the digital signal. It is approximate because the time required to recognize the interrupt and start executing the ISR, and the time required to resume normal execution after the ISR, are not measurable this way.

然后，在ISR中花费的CPU时间的近似百分比大约是数字信号的占空比这是近似的，因为识别中断和开始执行ISR所需的时间，以及在ISR之后恢复正常执行所需的时间，这种方式无法测量

As an alternative, the state of the interrupt flags register (IFR) can be examined at the end of the ISR. For most of the code in this book, hardware interrupt INT12 is used. If the 12th-bit of the IFR is a 1 at the end of the ISR, that means that another interrupt is pending before you finished servicing the current one, so the real-time schedule has not been met. Example code to implement this method is shown next.

作为替代方案，可以在ISR结束时检查中断标志寄存器（IFR）的状态对于本书中的大多数代码，使用硬件中断INT12如果IFR的第12位在ISR结束时为1，则意味着在完成当前服务之前，另一个中断处于待处理状态，因此实时计划未得到满足接下来显示实现此方法的示例代码

Listing H.2: Checking for real-time schedule failure using the interrupt flags register.

清单H.2：使用中断标志寄存器检查实时调度失败

2

2

4

4

6

6

8

8

Variable Initialization

变量初始化

In the C programming language, declaring a variable does not automatically cause that variable to be initialized to a known value. In general, variables must always be set to a value before they are evaluated in a C program. This does not mean that they need to be initialized in the declaration, as long as they are written to in an assignment statement before they are evaluated. The code below implements a simple IIR filter. In this example, the variables x and y are intentionally (and incorrectly) left uninitialized.

在C编程语言中，声明变量不会自动将该变量初始化为已知值通常，在C程序中计算变量之前，必须始终将变量设置为一个值这并不意味着它们需要在声明中初始化，只要它们在被评估之前写入赋值语句中即可。下面的代码实现了一个简单的IIR过滤器在此示例中，变量x和y有意（并且错误地）保持未初始化

Listing H.3: Example IIR filter code with incorrect variable initialization.

清单H.3：具有不正确的变量初始化的示例IIR过滤器代码

1

1

3

3

5

5

7

7

9

9

11

11

For x[0] and y[0], this oversight will not cause a problem. The variable element x[0] is assigned a value on line 8 before it is evaluated on lines 9 and 10. Similarly, y[0] is assigned a value on line 9 before it is evaluated on lines 11 and 12. However, it is not acceptable for x[1] and y[1], since they are both evaluated on line 9 before either has been assigned a value. Although this may seem minor, it is in fact a major problem. If either variable randomly has a large numeric value, that is equivalent to a large transient that may take a long time to decay. The worst case situation is if either x[1] or y[1] has a value of NaN (not a number). In that case, the result of the line 9 calculation with a NaN results in a value of NaN being assigned to y[0], which is then assigned to y[1] on line 11. This means that the line 9 assignment to y[0] thereafter will always be a NaN, so the filter will never function. To prevent this, the x and y variables should be initialized as shown below.

对于x [0]和y [0]，这种疏忽不会导致问题在第9行和第10行评估变量元素x [0]之前，它在第8行被赋值类似地，在第11行和第12行评估之前，在第9行为y [0]分配一个值但是，x [1]和y [1]是不可接受的，因为它们都在第9行被评估之前被分配了一个值虽然这看起来很小，但事实上这是一个主要问题如果任一变量随机地具有较大的数值，则相当于可能需要很长时间才能衰减的大瞬态最糟糕的情况是，如果x [1]或y [1]的值为NaN（不是数字）在这种情况下，使用NaN计算第9行的结果导致将NaN的值分配给y [0]，然后将其分配给第11行的y [1]这意味着之后对y [0]的第9行赋值将始终为NaN，因此过滤器将永远不起作用为防止这种情况，应初始化x和y变量，如下所示

Listing H.4: Correct variable initialization.

清单H.4：更正变量初始化

float x [ 2] = { 0. 0 , 0 . 0 } ; // input

float x [2] = {00 , 00};

2 float y [ 2] = { 0. 0 , 0 . 0 } ; // output

2 float和[2] = {00 , 00};

Integer Data Sizes

整数数据大小

The C programming language does not specify a fixed size for integer data types such as int, short, long, etc. Rather, the data type int is set to be the machine word size for a specific compiler target. In the C6000 DSPs, the registers are 32-bits, so the size of the int data type is 32-bits. Since we only are creating code for the C6000 family, this does not present a problem once you learn the sizes of the different data types. However, suppose you then wanted to reuse your code on a different architecture. The size of the integer data types might be different; if so, you would have to go through your code and change all of your variable declarations to be the correct size.

C编程语言没有为整数数据类型指定固定大小，例如int，short，long等而是将数据类型int设置为特定编译器目标的机器字大小在C6000 DSP中，寄存器是32位，因此int数据类型的大小是32位由于我们只为C6000系列创建代码，因此一旦您了解了不同数据类型的大小，就不会出现问题但是，假设您希望在不同的体系结构上重用代码整数数据类型的大小可能不同;

To make your code more portable across different architectures, a common technique is to define a set of data types that explicitly indicate the size of the variable. The C

为了使代码在不同体系结构中更具可移植性，常见的技术是定义一组明确指示变量大小的数据类型C

programming language supports defining new data types using the typedef compiler direc- tive. An example is shown in Listing H.5 below. The typedef directive on line 1 tells the compiler that Uint32 is a new name for the data type unsigned int. By coding with a set of explicitly sized types such as Uint32 and Int16, it is easy for the programmer to select the required variable size.

编程语言支持使用typedef编译器指令定义新的数据类型清单H中显示了一个示例5以下第1行的typedef指令告诉编译器Uint32是unsigned int数据类型的新名称通过使用一组显式大小的类型（如Uint32和Int16）进行编码，程序员可以轻松选择所需的变量大小

Listing H.5: C6000 typedef directives.

清单H.5：C6000 typedef指令

2

2

We use names for our typedef directives that we hope are unambiguous to the reader. For example, Uint32 is a 32-bit unsigned integer, Int16 is a 16-bit signed integer, Uint8 is an 8-bit unsigned integer, and so on.

我们使用我们的typedef指令的名称，我们希望这些指令对读者来说是明确的例如，Uint32是32位无符号整数，Int16是16位有符号整数，Uint8是8位无符号整数，依此类推

When porting your code to a different architecture, the only change required is to include an appropriate set of typedef directives. For example, if you were porting code to a C5000 DSP architecture, the size of the unsigned int is 16-bits but the size of the unsigned long is 32-bits. Therefore, you would use the typedef directives shown in Listing H.6 below.

将代码移植到不同的体系结构时，唯一需要的更改是包含一组适当的typedef指令例如，如果您将代码移植到C5000 DSP架构，则无符号整数的大小为16位，但无符号长整数的大小为32位因此，您将使用清单H中所示的typedef指令6下面

Listing H.6: C5000 typedef directives.

清单H.6：C5000 typedef指令

2

2

All the necessary typedef directives needed for projects in this book are contained in the tistdtypes.h file that you include in each project. Using conditional compilation, the correct set of typedef directives is automatically selected when you compile your code.

本书中项目所需的所有必要的typedef指令都包含在tistdtypes中您在每个项目中包含的文件使用条件编译时，编译代码时会自动选择正确的typedef指令集

# Appendix I

附录一

Comparison of DSP Boards

DSP板的比较

Introduction

介绍

E are firm believers in the need for a working knowledge of real-time DSP to be part of a complete EE/ECE curriculum. Such a working knowledge cannot come only through books, lectures, or MATLAB demos; students need to use actual DSP hardware and get real-time applications to run successfully before they can acquire a practical working knowledge of real-time DSP. Throughout the years, we’ve used a number of boards in our labs for this purpose using both fixed- and floating-point Texas Instruments (TI) processors, such as the C50, C31, C6201, C6211, C6711, C6713, and most recently the multi-core OMAP-L138 (which includes both a C6748 core and an ARM926 core). Of these boards, several are now only of historical interest, while the boards based on the C6713 and the

E坚信需要将实时DSP的工作知识作为完整的EE / ECE课程的一部分这样的工作知识不能只通过书籍，讲座或MATLAB演示来实现;多年来，我们使用固定和浮点德州仪器（TI）处理器（例如C50，C31，C6201，C6211，C6711，C6713和大多数）在我们的实验室中使用了许多电路板。在这些董事会中，有几个现在只有历史意义，而董事会则基于C6713和

OMAP-L138 remain our primary targets of interest.

OMAP-L138仍然是我们感兴趣的主要目标

Three Boards

三个板子

The Spectrum Digital C6713 DSK, the Logic PD OMAP-L138 Zoom Experimenters Kit (ZEK), and the newer Texas Instruments OMAP-L138 Low Cost Development Kit (LCDK) can all be used effectively with this book. How do these three boards compare? Table I.1 provides the most salient comparative details.

Spectrum Digital C6713 DSK，Logic PD OMAP-L138变焦实验套件（ZEK）和较新的德州仪器OMAP-L138低成本开发套件（LCDK）都可以有效地用于本书这三块板如何比较？1提供了最显着的比较细节

We provide support for the C6713 DSK in the latest edition of our book mainly for legacy purposes, as many universities have labs populated with these boards. For those just getting started, populating new labs, or for those wanting to upgrade existing labs, one of the two OMAP-L138 boards seems to make more sense. Using basic criteria such as price and available I/O, the new LCDK appears to be the better choice.1 The only slight disadvantage to the LCDK is the need for an external XDS100 emulator, but this is a minor issue. After using both boards, we now prefer the LCDK. Beyond price and I/O, the comparison of OMAP boards gets a bit more interesting, if you start to consider more subtle issues.

我们在本书的最新版本中为C6713 DSK提供支持，主要用于传统目的，因为许多大学都有实验室填充这些板对于那些刚入门，填充新实验室或想要升级现有实验室的人来说，两个OMAP-L138板之一似乎更有意义使用价格和可用I / O等基本标准，新的LCDK似乎是更好的选择1 LCDK唯一的缺点是需要外部XDS100仿真器，但这是一个小问题使用两块板后，我们现在更喜欢LCDK除了价格和I / O之外，如果您开始考虑更微妙的问题，OMAP板的比较会更有趣

For example, while both OMAP-L138 boards use the identical audio codec chip, the two manufacturers chose to integrate that codec into the overall board design in different ways. The ZEK only makes available line in and line out, whereas the LCDK also makes available an amplified microphone input (handy for commonly available non-powered microphones).

例如，虽然两个OMAP-L138板都使用相同的音频编解码器芯片，但两家制造商选择以不同方式将该编解码器集成到整个电路板设计中ZEK仅提供线路输入和线路输出，而LCDK还提供放大的麦克风输入（适用于常用的无源麦克风）

1The OMAP-L138 Experimenter Kit is no longer being manufactured, but many universities still use

1 OMAP-L138实验套件已不再生产，但许多大学仍在使用

this board.

这个董事会

423

423

424 APPENDIX I. COMPARISON OF DSP BOARDS

424附录一DSP板的比较

Table I.1: A comparison of the three primary DSP boards that are supported by this book.

表一1：本书支持的三个主要DSP板的比较

aOlder boards shipped with 64 MB of mDDR SDRAM.

桤木板附带64MB DDR SDRAM

bEarly versions of the LCDK included an Authentec fingerprint swipe sensor, but since Authentec sub- sequently discontinued production of this sensor, newer LCDKs do not include a fingerprint reader.

b早期版本的LCDK包括一个Authentec指纹刷卡传感器，但由于Authentec随后不再生产这种传感器，因此较新的LCDK不包括指纹识别器

cNo longer available.

不再可用

dNote: to program the LCDK in C (using Code Composer Studio from TI), you will also need an inexpensive XDS100 emulator, since it isn’t part of the main board. These are available at a suggested retail price of $79 from TI’s e-store or from a variety of third-party vendors.

d注意：要用C编程LCD（使用TI的Code Composer Studio），您还需要一个便宜的XDS100仿真器，因为它不是主板的一部分TI的电子商店或各种第三方供应商的建议零售价为79美元

Figure I.1: Spectrum analyzer screen shot showing codec noise associated with the Zoom Experimemters Kit.

图一1：频谱分析仪屏幕截图显示与Zoom Experimenters Kit相关的编解码器噪声

Furthermore, the power supply decoupling of the codec chip is very different between the two boards, and this is important to some applications.

此外，编解码器芯片的电源去耦在两个板之间是非常不同的，这对于一些应用来说是重要的

It can be seen from the board schematics that the codec on the ZEK connects directly to the “noisy” switching DC power supply, whereas the LCDK uses LC filtering on the power connection. From a design standpoint, the better board is the LCDK, since power supply noise is not coupled into the codec as it is in the ZEK. One could argue that such power supply noise is well above the audio range intended for the audio codec, so why “waste” money on filter components? But such high-frequency noise can still be problematic in a laboratory setting where processed signals are routinely analyzed using traditional test and measurement equipment. This unnecessary high-frequency noise can be clearly seen in the screen capture of Figure I.1, which shows the average spectrum of the ZEK’s audio codec output.

从电路板原理图可以看出，ZEK上的编解码器直接连接到“噪声”开关直流电源，而LCDK在电源连接上使用LC滤波从设计的角度来看，更好的电路板是LCDK，因为电源噪声没有像ZEK那样耦合到编解码器中有人可能会说这种电源噪声远远高于音频编解码器的音频范围，那么为什么要“浪费”过滤器组件呢？在图1的屏幕截图中可以清楚地看到这种不必要的高频噪声图1显示了ZEK音频编解码器输出的平均频谱

In this figure, the Bx marker (seen at the far left of the display) is placed at the first spectral null near 45 kHz, so the energy shown in Figure I.1 is all well above the audio frequencies. But it can still cause a problem. Often, the system’s output is analyzed by test and measurement equipment (e.g., an oscilloscope, spectrum analyzer, or a vector signal analyzer) that is usually of the high-speed sampled type (e.g., a digital sampling oscilloscope or DSO). Such test equipment typically does not incorporate antialiasing filters in the front end, meaning that aliasing of this “out of the audio band energy” can end up in the audio range and become a huge problem for certain DSP applications. This problem could easily have been avoided by a more prudent circuit design. The design used in the LCDK clearly demonstrates this, as evidenced by its quieter code output.

在此图中，Bx标记（在显示屏的最左侧看到）放置在45 kHz附近的第一个光谱零位，因此图1中显示的能量1都远高于音频但它仍然可能导致问题通常，系统的输出由测试和测量设备分析（例如，G，示波器，频谱分析仪或矢量信号分析仪），通常是高速采样类型（例如，G，数字采样示波器或DSO）这种测试设备通常不在前端包含抗混叠滤波器，这意味着这种“超出音频带能量”的混叠可能会在音频范围内结束，并成为某些DSP应用的巨大问题通过更谨慎的电路设计可以很容易地避免这个问题LCDK中使用的设计清楚地证明了这一点，其更安静的代码输出就是证明

426 APPENDIX I. COMPARISON OF DSP BOARDS

426附录一DSP板的比较

Figure I.2: Spectrum analyzer screen shot showing much lower levels of codec noise associ- ated with the LCDK.

图一2：频谱分析仪屏幕截图显示与LCDK相关的编解码器噪声水平低得多

Figure I.2 shows the average spectrum of the LCDK’s audio codec output. Compare this to the previous figure showing the ZEK codec output. The LCDK uses exactly the same AIC3106 codec chip that is used in the ZEK, and the sample frequency, scales, and markers for the two figures are as close to identical as we could get them. The figure of the LCDK’s output clearly shows a substantial reduction (approximately 30 dB) in the “out of audio band noise” compared to the ZEK.

图一图2显示了LCDK音频编解码器输出的平均频谱将此与上图显示ZEK编解码器输出进行比较LCDK使用与ZEK中使用的完全相同的AIC3106编解码器芯片，两个数字的采样频率，比例和标记尽可能接近相同。与ZEK相比，LCDK输出的数字清楚地表明“音频带外噪声”显着减少（约30 dB）

Conclusion

结论

Based on price, available I/O, and the noise characteristics of the codec output, the better of the two OMAP-L138 boards seems to be the LCDK. That being said, this book fully sup- ports all three boards: the Spectrum Digital C6713 DSK, the Logic PD Zoom OMAP-L138 Experimenters Kit (ZEK), and the Texas Instruments OMAP-L138 Low Cost Development Kit (LCDK).

基于价格，可用I / O和编解码器输出的噪声特性，两个OMAP-L138板中的两个似乎更好的是LCDK话虽如此，本书完全支持所有三个主板：Spectrum Digital C6713 DSK，Logic PD Zoom OMAP-L138实验套件（ZEK）和Texas Instruments OMAP-L138低成本开发套件（LCDK）

Appendix J

附录J.

Abbreviations, Acronyms, and Symbols

缩写，缩略语和符号

HIS is a partial list of abbreviations, acronyms, and symbols used in the text, provided in the hope that it will be helpful to some readers.

HIS是文本中使用的缩写词，首字母缩略词和符号的部分列表，希望它对某些读者有所帮助

( ) used for a continuous function.

（）用于连续函数

[ ] used for a discrete function.

[]用于离散函数

α feedback coefficient for simple IIR filters, such as those used for a type of echo generation for guitar special effects.

简单IIR滤波器的α反馈系数，例如用于吉他特效的一种回声生成

λ wavelength.

波长λ

π ratio of a circle circumference to diameter, 3.1415926535897932. . .

π圆周长与直径的比率，31415926535897932

τ time constant.

τ时间常数

ω radian frequency.

ω弧度频率

filter coefficient associated with an output term, y. When used in a transfer function, the a coefficients are associated with the denominator of the transfer function.

与输出项y相关的滤波器系数当在传递函数中使用时，a系数与传递函数的分母相关联

vector or array containing all of the a terms.

包含所有术语的向量或数组

ADC analog-to-digital converter.

ADC模数转换器

AIC analog interface circuit (see codec).

AIC模拟接口电路（见编解码器）

AGC automatic gain control.

AGC自动增益控制

AM amplitude modulation.

AM幅度调制

ANC adaptive noise cancellation.

ANC自适应噪声消除

ARM Advanced RISC Machine, a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA) developed by ARM Holdings.

ARM高级RISC机器，ARM Holdings开发的32位精简指令集计算机（RISC）指令集架构（ISA）

AWGN additive white Gaussian noise.

AWGN加性高斯白噪声

427

427

428 APPENDIX J. ABBREVIATIONS, ACRONYMS, AND SYMBOLS

428附录J.缩写，缩略语和符号

filter coefficient associated with an input term, x. When used in a transfer function, the b coefficients are associated with the numerator of the transfer function.

与输入项x相关联的滤波器系数当在传递函数中使用时，b系数与传递函数的分子相关联

vector or array containing all of the b terms.

包含所有b项的向量或数组

BW bandwidth of a bandpass signal.

带通信号的带宽

BP bandpass.

BP带通

BPF bandpass filter.

BPF带通滤波器

BPSK binary phase shift keying.

BPSK二进制相移键控

value of capacitance.

电容值

CCS Texas Instruments’ Code Composer StudioTM.

CCS德州仪器的Code Composer Studio

CD-ROM compact disk read-only memory.

CD-ROM光盘只读存储器

CISC complex instruction set computer.

CISC复杂指令集计算机

codec coder-decoder. An integrated circuit that contains both an ADC and a DAC.

编解码器编码器 - 解码器集成电路，包含ADC和DAC

CPU central processing unit.

CPU中央处理单元

DAC digital-to-analog converter.

DAC数模转换器

DC direct current (0 Hz).

直流直流电（0 Hz）

DDS direct digital synthesizer or direct digital synthesis.

DDS直接数字合成器或直接数字合成

DF-I direct form I.

DF-I直接表格I

DF-II direct form II.

DF-II直接形式II

DFT discrete Fourier transform.

DFT离散傅里叶变换

DMA direct memory access.

DMA直接内存访问

DSK DSP starter kit.

DSK DSP入门套件

DSP digital signal processing or digital signal processor.

DSP数字信号处理或数字信号处理器

DTFT discrete-time Fourier transform.

DTFT离散时间傅立叶变换

DTMF dual-tone, multiple-frequency signals as defined by telephone companies.

电话公司定义的DTMF双音多频信号

ECG electrocardiogram.

心电图（ECG）

EDMA enhanced direct memory access.

EDMA增强了直接内存访问

FCC Federal Communications Commission.

FCC联邦通信委员会

FIR finite impulse response.

FIR有限脉冲响应

FFT fast Fourier transform.

FFT快速傅里叶变换

FT Fourier transform.

FT傅里叶变换

F Fourier transform.

F傅立叶变换

F−1 inverse Fourier transform.

F-1逆傅里叶变换

fh highest or maximum frequency that is present in a signal.

fh信号中存在的最高或最高频率

Fs sample frequency (samples/second) = 1/Ts.

Fs采样频率（采样/秒）= 1 / Ts

GPP general purpose processor.

GPP通用处理器

GPU graphics processing unit.

GPU图形处理单元

H(ejω) discrete-time frequency response.

H（ejω）离散时间频率响应

H(jω) continuous-time frequency response.

H（jω）连续时间频率响应

h[n] discrete-time impulse response or unit sample response.

h [n]单位样本响应的离散时间脉冲响应

h[t] continuous-time impulse response.

h [t]连续时间脉冲响应

H(s) continuous-time transfer or system function.

H（s）连续时间传递或系统功能

H(z) discrete-time transfer or system function.

H（z）离散时间传递或系统功能

HDTV high-definition television.

高清电视高清电视

HP highpass.

惠普高通

HPF highpass filter.

HPF高通滤波器

HPI host port interface.

HPI主机端口接口

Hz hertz (cycles per second).

Hz赫兹（每秒周期数）

IEEE 754 floating point number format.

IEEE 754浮点数格式

IF intermediate frequency.

IF中频

IFFT inverse fast Fourier transform.

IFFT逆快速傅里叶变换

IIR infinite impulse response.

IIR无限脉冲响应

ISA instruction set architecture.

ISA指令集架构

ISR interrupt service routine.

ISR中断服务程序

j √ 1; identifies the imaginary part of a complex number. Some authors use the

j√1;有些作者使用了

letter i instead of the letter j.

字母i而不是字母j

JTAG Joint Test Action Group, commonly used as the name of a debugging interface for printed circuit boards and IC chips. Formalized as IEEE Std 1149.1 in 1990.

JTAG联合测试行动小组，通常用作印刷电路板和IC芯片调试接口的名称形式化为IEEE Std 11491990年1

L Laplace transform.

L拉普拉斯变换

L−1 inverse Laplace transform.

L-1逆拉普拉斯变换

value of inductance.

电感值

LCDK Low Cost Development Kit.

LCDK低成本开发套件

LFSR linear feedback shift register.

LFSR线性反馈移位寄存器

LP lowpass.

LP低通

LPCM linear pulse code modulation.

LPCM线性脉冲编码调制

LPF lowpass filter.

LPF低通滤波器

LSB lower sideband, also used for least significant bit.

LSB下边带，也用于最低有效位

the number of bands in a graphic equalizer.

图形均衡器中的波段数

MA moving average.

MA移动平均线

McASP multi-channel audio serial port. McBSP multi-channel buffer serial port. ML maximum likelihood.

McASP多声道音频串口McBSP多通道缓冲串口ML最大可能性

n index or sample number.

n索引或样本号

often used as filter order; in other contexts, it is used for the length of a sequence, or for the length of an FFT.

经常用作过滤顺序;

NCO numerically controlled oscillator.

NCO数控振荡器

OMAP Open Multimedia Application Platform, a family of proprietary multi-core sys- tem on chips (SoCs) by Texas Instruments.

OMAP开放式多媒体应用平台，德州仪器（TI）的一系列专有芯片系统（SoC）

PC personal computer.

PC个人电脑

PCM pulse code modulation.

PCM脉冲编码调制

PLL phase-locked loop.

PLL锁相环

PN pseudonoise.

PN伪噪声

PSK phase shift keying.

PSK相移键控

quality factor. Q = bandwidth of a BP filter divided by its center frequency.

品质因素Q = BP滤波器的带宽除以其中心频率

The higher the value of Q, the more selective the BP filter is.

Q值越高，BP滤波器的选择性越高

QAM quadrature amplitude modulation.

QAM正交幅度调制

QPSK quadrature phase shift keying.

QPSK正交相移键控

magnitude of a pole. This is a measure of how far the pole is from the origin.

杆的大小这是衡量杆距离原点的距离

value of resistance.

抵抗价值

RC resistor-capacitor.

RC阻容

RISC reduced instruction set computer.

RISC减少了指令集计算机

RF radio frequency.

射频射频

APPENDIX J. ABBREVIATIONS, ACRONYMS, AND SYMBOLS 431

附录J.缩写，缩略语和符号431

the Laplace transform independent variable, s = σ + jω.

SoC system on chip.

片上SoC系统

SOS second-order section.

SOS二阶部分

τ a dummy variable often used in convolution.

τ常用于卷积的虚拟变量

time.

时间

T period of a signal or function.

T信号或功能的周期

TED timing error detector.

TED定时误差检测器

Ts sample period = 1/Fs.

Ts采样周期= 1 / Fs

TI Texas Instruments.

TI德州仪器

u[n] discrete-time unit step function.

u [n]离散时间单位阶跃函数

u(t) unit step function.

u（t）单位阶跃函数

U.S. United States (of America).

该小号美国）

USB upper sideband; also used for Universal Serial Bus.

USB上边带;

V voltage in volts.

V电压，单位为伏特

Vin input voltage.

Vin输入电压

Vout output voltage.

Vout输出电压

VLIW very long instruction word; this is a type of architecture for DSPs.

VLIW指令字很长;

winDSK original Windows-based program for the C31 DSK, created by Mike Morrow.

winDSK原创基于Windows的C31 DSK程序，由Mike Morrow创建

winDSK6 Windows-based program, the follow-on to winDSK, for the C6x DSK series. It was created by Mike Morrow.

基于Windows的winDSK6程序，即winDSK的后续程序，适用于C6x DSK系列它是由Mike Morrow创建的

winDSK8 Windows-based program, the follow-on to winDSK6, for both OMAP-L138 multi-core boards and the C6713 DSK. It was created by Mike Morrow.

适用于OMAP-L138多核板和C6713 DSK的winDSK8基于Windows的程序，winDSK6的后续程序它是由Mike Morrow创建的

X(jω) result of the Fourier transform F{x(t)}; it shows the frequency content of x(t). x[n] a discrete-time input signal.

傅立叶变换的X（jω）结果F {x（t）};x [n]是离散时间输入信号

x(t) a continuous-time input signal.

x（t）连续时间输入信号

Y (jω) result of the Fourier transform F{y(t)}; it shows the frequency content of y(t). y[n] a discrete-time output signal.

傅立叶变换的Y（jω）结果F {y（t）};y [n]是离散时间输出信号

432 APPENDIX J. ABBREVIATIONS, ACRONYMS, AND SYMBOLS

432附录J.缩写，缩略语和符号

y(t) a continuous-time output signal.

y（t）连续时间输出信号

z the independent transform variable for discrete-time signals and systems.

z离散时间信号和系统的独立变换变量

z−1 a delay of 1 sample time.

z-1延迟1个采样时间

Zc impedance of a capacitor.

电容器的Zc阻抗

Z z-transform.

Z变换

Z−1 inverse z-transform.

Z-1逆z变换

ZEK Zoom Experimenter Kit

ZEK Zoom实验套件