

SignalMaster Manual

Version 20190410



Intelligent Hearing Systems Corp. 6860 S.W. 81st Street – Miami, FL 33143 - USA

Introduction:

SignalMaster was developed under a United States Department of Health, National Institutes of Health (NIH), National Institute on Deafness and Other Communication Disorders (NIDCD), Small Business Innovation Research (SBIR) grant to Intelligent Hearing Systems, Corp. The purpose of the grant was to develop an open architecture hardware and software system that will allow researchers to conduct a wide range of studies involving multi-channel acoustic input and output for psychophysical research and hearing aid development. The current system provides users with the ability to develop their own processing algorithms by providing source code examples and the ability to upload programs to the device and communicate with a user developed control program on an external computer. The hardware provides two input microphone and two output speakers, analog channels with up to 32 bit digital conversion resolution and is able to operate independently from battery power. The system provides sufficient processing capabilities to perform real-time signal processing with minimal delay (or latency) between input and output with various point-by-point or batch processing capabilities.

SignalMaster is based on a Texas Instruments (TI) Digital Signal Processor (DSP), TMS320C6746. For full specifications, see the SignalMaster Specifications section below.

Web Site: http://ohsspds.ihsys.info/

Signal processing modules are available online for download by users from the following web address:

This site contains a list of documents that are available. The DSP source code documents (zip files) start with the name "SampleDSPCode_" follow by a descriptive name of the type of processing. These zip files contain all files required by the TI code composer to modify the DSP program. IHS provided executables can be used to upload any DSP code to the hardware.

Getting Started:

Your SignalMaster system doesn't require any special drivers. Simply install the software from your SignalMaster thumb drive. The thumb drive contains example programs and libraries required for developing new applications.

Simply run the Setup.EXE application on the thumb drive:

SignalMa	SignalMaster V 🗗 Search SignalMa				م
^	Name	Date modified	Туре	Size	
	Software	8/10/2017 1:06 PM	File folder		
	System	8/9/2017 10:31 AM	File folder		
	IHSGEN.ICO	1/16/1998 1:37 PM	lcon	1 KB	
	Setup.bmp	5/17/2000 2:23 PM	BMP File	162 KB	_
	setup.exe	9/28/2012 10:14 AM	Application	1,428 KB	
	🛓 SETUP.INI	8/9/2017 10:10 AM	Configuration sett	1 KB	•

After running the Setup program, press the start button to install the software to the C:\SignalMaster directory:

	Si	gnalMaster - 1.00		HEARING
Directory:	C:\SignalMaster			
Work Done:		0%		
This File:		0%		
Files:			0	
Disk Space: Free: 839.65 GB Total: 987 56 GB	Up Date Only		φ.	

It is highly recommended to install the software in the default C:\SignalMaster directory as all the CCS example projects are pre-set to look for libraries in that directory. If you wish to install SignalMaster in a different directory, you will need to modify the default compiler and linking directories in CCS.

The installation utility will default to C:\SignalMaster and contains the following subdirectories:

C:\SignalMaster\...

DSPProjects – containing sample DSP applications

PCProjects – containing sample PC application with their corresponding DSP OUTFiles (For the source code, look under the DSPProjects directory of the corresponding project) **IHSC6748HW** – containing software and hardware specific files All PC programs developed require at a minimum the following files that are provided or can be developed by the user:

- 1) **MyProgram.EXE** Your application program.
- 2) MyDSPProgram.OUT Compiled TI DSP program generated using TI Code Composer.
- 3) **SIGMASDLL.DLL** IHS DLL that provides DSP communication routines and ability to upload programs into the DSP.
- 4) **Out2rprc.EXE** TI provided program (available from your SignalMaster distribution thumbdrive) used by the DLL to convert DSP compiled OUT files to BIN format for upload.
- 5) **IHSWIN.SYS** IHS License file that contain information about your hardware. The DLL uses this file to determine if your hardware is available or not.

Using the provided DLL, you may develop PC based applications in any programming language you like, as long as that language supports DLLs. The DLL will allow you to transfer data to and from the DSP to your application program on your PC. It will also allow you to send user defined command instruction to the DSP.

Before you can continue to develop any programs for SignalMaster, you will need to install TI Code Composer Studio (CCS). The following section will walk you through the installation process.

Installing Code Composer Studio (CCS):

Sections:

- Installing Code Composer Studio
- Importing Projects
- Setting Up The User Interface
- Installing the Compiler
- Updating Project Compiler Settings
- Compiling a Project
- Loading and Running a Project
- Starting a New Project
- Selecting a New Work Space
- Code Composer Studio Compiler Settings (Advanced Options):

Installing Code Composer Studio:

- 1. Download the offline version of the Code Composer Studio (CCS) installer from http://processors.wiki.ti.com/index.php/Download_CCS
- Begin installing the application. If you are greeted with a window like the one below chances are you'll be fine to just click <u>Continue</u>: Note: If you do run into an error, as shown below, then turn off your anti-virus and try again.

😳 Code Composer Studio v8 Setup	-		×
CCS Installation			
The Code Composer Studio installer is checking for any preinstall dependencies. This may take some time.			
Please read the information carefully to determine if you need to take any action prior to continuing.			
Starting dependency checks Operating System Check Windows 10 -> OK Installer Path -> OK Wincide charater Check -> OK			
Anti-Virus Check -> We have detected you are running anti-virus software on this computer. To ensure no problems occur highly recommended that real-time file scanning be turned off before proceeding with the installation. Pending reboot Check -> OK	r during the	install	ation, it is
Checking Windows Updates Done			
Texas Instruments		0	ontinue

3. Now click <u>Accept</u> and <u>Next</u>:

Code Composer Studio v8 Setup			×
icense Agreement			
Please read the following license agreement carefully.			
TECHNOLOGY SOFTWARE PUBLICLY AVAILABLE			^
Copyright (c) 2016 Texas Instruments Incorporated			
All rights reserved not granted herein.			
Limited License Agreement.			
This Limited License Agreement ("Agreement") is a legal agreement betwe Incorporated ("TT). The "Software" consists of the following materials (a) forthware mainlers for the software subject to the terms herein, and any "on any portion thereof (the "Licensed Materials"), and (b) the materials identify forthware mainlers to rule Software, or any portion thereof ("Public Software the licensing terms contained in this Agreement and your use of the Public applicable software manifest and/or identified or included with the materi- under, or grant you rights that superseds, the license terms of any applicable therwise using the Licensed Materials you agree to abide by the terms of terms, do not download or install the Licensed Materials.	the materials identified as TI -line" or electronic documen fied as open source materials re"). For clarification, your uss Software is subject to the seg lals to which they apply. This ole Public Software license ag	proprietary software progra tation associated with these or thrid party proprietary se e of the Licensed Materials is parate licensing terms speci parate licensing terms speci Agreement does not limit y reement. By installing, copy	ams in the e programs, or oftware in the is subject to fied in the your rights ying or
Il hereby grants you a world-wide, royalty-free, non-exclusive license unde have made, use, import, offer to sell and sell ("Utilize") the Licensed Materi granted solely to the extent that any such patent is necessary to Utilize the combinations which include the Licensed Materials, other than combination patent is licensed hereunder.	ials. With respect to the foreg Licensed Materials alone. Th	going patent license, such li e patent license shall not ap	cense is pply to any
Redistributions must preserve existing copyright notices and reproduce thi disclaimer and (if applicable) source code license limitations below) in the			
<			>
I accept the terms of the license agreement.			
○ I do not accept the terms of the license agreement.			Print
exas Instruments			
	< Back	Next > Finish	

4. Click <u>Next</u> again:

😳 Code Composer Studio v8 Setup				×
Choose Installation Location				
Where should Code Composer Studio v8 be installed?				
To change the main installation folder click the Browse button.				
CCS Install Folder				
				Browse
exas Instruments				
	< Back	Next >	Finish	Cancel

5. Select <u>C6000 Power-Optimized</u> DSP when prompted, and click <u>Next</u>:



6. Enable <u>TI XDS Debug Probe Support</u> and <u>Spectrum Digital Debug Probes and Boards</u>, and click <u>Finish</u>: *Note: Allow internet access if prompted.*

Code Composer Studio v8 Setup	3
lect Debug Probes	N.C.
Select the debug probes you want installed and deselect the debug probes you want	to leave out.
TI XDS Debug Probe Support	Description
Itackhawk Debug Probes Spectrum Digital Debug Probes and Boards	Drivers and support files for Spectrum Digital ITAG debug probes and development boards.
Select All	Install Size: 1214.55 MB.

7. The application will now take some time to perform the installation... Wait until it finishes...

😳 Code Composer Studio v8 Setup	- 0	×
CCS Installation		
Code Composer Studio is being installed on your computer		
$Installing_com.ti.ccstudio.c6000.feature.group.com.ti.ccstudio.c7000.feature.group.com.ti.xdctools_3_32.update.feature.group$	x,com.ti.dvt2.gra	ph.visu
feature registered		-
Installing com.ti.dsflash.win32 Unzipping c./ti/ccsv8/eclipse/downloads/com.ti.dsflash.win32.8.2.0.1409.zip		
com.ti.dsflash.win32 installed		
Registering com.ti.ccstudio.cloudagent.win32		
feature registered		
Installing com.ti.ccstudio.cloudagent.win32		
Unzipping c/ti/ccsv8/eclipse/downloads/cloudagent_win_0.5.1463.zip		
com.ti.ccstudio.cloudagent.win32 installed		
Registering com.ti.ccstudio.nodeis.win32		
feature registered		
Installing com.ti.ccstudio.nodeis.win32		
Unzipping c/ti/ccsv8/eclipse/downloads/node-8.11.1_win32.zip		
com.ti.ccstudio.nodejs.win32 installed		
Registering com.ti.ccstudio.idemain		- 1
Registering com.ti.ccstudio.workflow		
Registering com.ti.chromium.browser.feature.group		
Registering com.ti.ccstudio.p2tool		
Installing		
com.ti.ccstudio.idemain.feature.group.com.ti.ccstudio.workflow.feature.group.com.ti.chromium.browser.feature.group.com.ti.c ure.group	cstudio.p2tool.f	eat
Registering com.ti.ccstudio.c6000.feature.group.com.ti.ccstudio.c7000.feature.group		
Registering com.ti.xdctools_3_32.update.feature.group		
Registering com.ti.dvt2.graph.visualization.feature.group.com.ti.dvt2.rov.feature.group,com.ti.dvt2.resource.explorer.feature.gr	oup	
Registering com.ti.tirex		
Registering com.ti.dvt2.trace.ccs.feature.group.com.ti.dvt2.trace.control.feature.group.com.ti.dvt2.analysis.suite.feature.group Installing		
com it cestudio.c6000.feature.group.com it iccstudio.c7000.feature.group.com.ti.xdctools_3.22.update.feature.group.com.ti.dvt2. .feature.group.com.ti.dvt2.trov.feature.group.com.ti.dvt2.resource.explorer.feature.group.com.ti.tirex.feature.group.com.ti.dvt2.tr up.com.ti.dvt2.trave.control.feature.group.com.ti.dvt2.analysis.uite.feature.group		
		~
Texas Instruments	Car	ncel

8. Click Finish:



You have now completed the installation process for the interphase portion of CCS. There are still a few more items to take care of, but now you can run CCS. To Run CCS, simple select the CCS icon on your desktop or select from available applications:



When running CCS for the first time, enter <u>C:\SignalMaster\DSPProjects</u> as the default workspace directory.

Eclipse Launcher	×
select a directory as workspace	
Code Composer Studio uses the workspace directory to sto	ore its preferences and development artifacts.
Workspace: C:\SignalMaster\DSPProjects	∽ Browse
Recent Workspaces	
Copy Settings	
0	Launch Cancel

Importing Projects:

From within CCS:

1. Click File > Import...



- 2. Click to <u>C/C++ > CCS Projects</u>
- 3. Click Next



4. For "Select search-Directory:" Browse to <u>C:\SignalMaster\DSPProjects</u>. This will show you all of the projects contained in the <u>DSPProjects</u> folder.

Import CCS Eclipse Projects		Browse For Folder
Select CCS Projects to Import Select a directory to search for existing CCS Eclipse projects.	Select root directory of the projects to import	
Select search-directory: Select archive file: Discovered projects: Automatically import referenced projects found in same search-director Copy projects into workspace Open Resource Explorer to browse a wide selection of example projects	Browse Browse Select All Deselect All Refresh	 Photos Poland Program Files Program Files (x80) Sample001_PointByPointEcho Sample001_PointByPointEchoFilter Sample010_BlockEchoFilter Sample101_BlockEchoFilter Sample101_Shifter Sample20_WaveletCompressor IHSC674sHW
Kext > Finish	Cancel	

5. Click <u>Select All</u>: The dialog box should look similar to how it does in the picture below.

Import CCS Eclipse Proj	ects			
Select CCS Projects to	TR			
Select a directory to search	h for existing (LCS Eclipse projects	K	
Select search-directory:	C:\SignalMa	ster\DSPProjects		Browse
O Select archive file:	Browse			
Discovered projects:				
Sample001_Point				Select All
Sample002_Point		lter		Deselect All
Sample011_Block	kEchoFilter			Refresh
Sample020_Calib				
Sample101_Shift	er			
Sample200_Wave	eletCompresso	pr.		
Automatically import re		ects found in same s	earch-directory	
Copy projects into work	space			
Open Resource Explorer to	browse a wide	e selection of exam	ple projects	
	< Back		-	

(Note: Current 2019 example program names begin with name Samp3xxx)

6. Click <u>Finish</u>. The demonstration projects we supplied should now show in the project explorer window.



(Note: Current 2019 example program names begin with name Samp3xxx)

Setting Up The User Interface:

We find that CCS by default has its windows arranged oddly. We therefore recommend the following rearrangement for better ease of use. Drag and drop the respective tabs by clicking on the drag marker until they are arranged as shown or personally desired.

Drag Tab	Drag Tab			
DSPProjects - SampleQ01_PointByPointEc	cho/main.c - Code Composer Studio	\simeq	σ	×
File Edit View Navigate Project Run		I		
	(4 • 0 = ○ Φ • ○ +	Quick A	Access	8
Project Explorer 11 Sample001_PointByPointEcho [Act Contemporation of the second sec	12//control Array 13//control Array 13extern union infuncData_union infuncDataArray; 14extern union outFuncData_union outFuncDataArray; 15			~
B codecConfig.c B codecilh Grift Functions.c G H45,Utilities.c B main.c	<pre>16//</pre>			
Mainh Mainh Marsh Remosp.c PROTOCOLh KingBufferFunc.c SM_V2.cmd	<pre>22 short STest[AUDIO_SAMPLES_MAX]attribute((section(".ihsExternMem"))); 23 24 25 26 int main(void) 27(</pre>			
Bample002, PointByPointEchoFilter Sample010, BlockEcho Sample011, BlockEchoFilter Sample020, CalibrationModule Sample020, CalibrationModule Sample020, CalibrationModule	<pre>28 // 29 //Setup the DSP 10 //- 11 int sampling_rate; 12 int buffer_length; 13</pre>			
 G Sample101_Shifter G Sample200_WaveletCompressor 	14 INS_initialize05F(); 15 //Setup the audio codec and its interfaces 16 sampling rate=24000: Drag Tab			
Project	109// for(i=0 ; <buffer_length ;="" i++)<br="">110// {</buffer_length>	_	_	~
		E 81	13 13	
Explorer	Yoject 'Sample001_PointByPointEcho': Link successful			
Explore:	DSPL2ROM			1,049k
	DSPL2RAM Other Windows			262k 32,768
	DSPLIPRAM OCTIET WITHOUVUS			32,768
/	SHDSPLZROM			1,049k
/	> SHDSPL2RAM 125k (68%)			182k
/	SHDSPL1PRAM			32,768
1 /	SHDSPL1DRAM			32,768
	EMIFAC53			3,554k
	EMIFACS4			2,097k
	EMIFACS5		6	2,097k
2				

If a window is not visible, go to View menu and select any window you may want:

		w Navigate Project Run Scripts Wine Resource Explorer	
		Resource Explorer Resource Explorer Classic	V 🗢 V Quick Access
🎏 Sam	3	Grace Snippets Getting Started CCS App Center	ar ervurnum = cnnum_incennum, rray n infuncData_union infuncDataArray;
i Sam	e	GUI Composer™	> > on outFuncData_union outFuncDataArray;
Sam		Project Explorer Problems Alt+Shift+4 Console Alt+Shift+4 Advice	Henory Reserved for DMA and UART Operations
	0 日本学田 0	Debug Memory Browser Registers Expressions Variables Disassembly Breakpoints Modules	aid) 8
	の事業品	Terminal Scripting Console Target Configurations Outline Alt+Shift+C Stack Usage	<pre>the DSP pling_rate; freq_length; 0 tialize0SP(); the audio codec and its interfaces</pre>
	10	Memory Allocation Optimizer Assistant OtherAlt+Shift+O	g_rate=24000;
		39 bi 40 ci	<pre>decSettings.dmsFlag</pre> WX; //Should be less than or equal to AUDIO_SAMPLES_MAX ffer_length=10; //Should be less than or equal to AUDIO_SAMPLES_MAX decSettings.t codecSetting; decSettings.dmsFlag = CODECONFIG_ND_WA; decSettings.dmsFlag

Installing the Compiler:

After you have installed CCS, you will also need to install the **C6000 Compiler Module**. This module contains the actual c language compiler for the TI family of DSP chips used by SignalMaster. CCS will not be able to compile your code without that module.

1. Click Help > Install New Software:



2. For "Work with", Select Code Generation Tools Updates:

				o x
Available S	oftware			
Select a site	or enter the location of a site.)
Work with:	type or select a site			Add
	type or select a site All Available Sites			reference
type filter te	Code Generation Tools Updates http://software-dl.ti.co/ DVT Updates - http://software-dl.ti.com/ccs/non-esd/dv/	m/dsps/dsps_public_sw/sdo_ccstudio/codegen/Updates/p2	win32/	
Name	http://downloads.ti.com/codegen/eid/mspcc http://software-dii.com/dop/dop.public_av/do_ccsth http://software-dii.com/dop/dop/dop.public_av/do_ccsth http://software-dii.com/dop/dop/dop.public_av/do_ccsth Software-diate-Instantion/dop/dates - http://software-di.ti Code Composer Studio v7 Updates - http://software-di.ti	udio/codegen/Updates/p2mac/ ctrumdigital.com/ccs51/Updates		
Select All	Deselect All			
Select All Details	Deselect All			
	Deselect All			
Details	Deselect All	Hide items that are already installed		
Details		Hide items that are already installed What is <u>already installed</u> ?		
Details	the latest versions of available software			
Details	the latest versions of available software ss by category			

3. Expand <u>TI Compiler Updates</u>, and then click the latest version of <u>C6000 Compiler Tools</u>. The dialog window should look like the picture shown below.

🚱 Install		- 0 X
Available Software		
Check the items that you wish to install.		() _
Work with: Code Generation Tools Updates - http://software-dl.ti.com/dsps	/dsps_public_sw/sdo_ccstudio/codegen/Updates/p2win32/ V Add_	Manage
	http://www.conditionedianeopolicity.premistry -	manage
ype filter text		
Name	Version	· · · · · · · · · · · · · · · · · · ·
ARM Compiler Tools	16.9.9	
ARM Compiler Tools	15.12.7	
ARM Compiler Tools	5.2.9	
ARM Compiler Tools	5.1.14	
ARM Compiler Tools	5.0.11	
ARM Compiler Tools	4.9.9	
ARM Compiler Tools	4.7.1	
ARM Compiler Tools	4.6.6	
C2800 Compiler Tools	18.1.3	
C2800 Compiler Tools	16.9.9	
C2800 Compiler Tools	15.12.7	
C2800 Compiler Tools	6.4.12	
C2800 Compiler Tools	6.2.11	
C2800 Compiler Tools	6.1.10	
C2800 Compiler Tools	6.0.6	
C2800 Compiler Tools	5.2.15	
G2800 Compiler Tools	5.1.6	
George Compiler Tools	8.3.0	
G6000 Compiler Tools	8.1.7	
Select All Deselect All 1 item selected		
Details		
Details		
		í liter a liter
	_	
Show only the latest versions of available software	Hide items that are already installed	
Group items by category	What is already installed?	
Show only software applicable to target environment		
Contact all update sites during install to find required software		
(?)	< Back Next > Finish	Cancel
U	< back INEXT > Finish	Cancel

Note: Demonstration projects may have been compiled and supplied with a previous compiler version. This should not be problem. Continue to select the latest compiler. The next section will cover how to handle compiler discrepancies.

4. Click <u>Next</u>, then <u>next</u> again, then <u>accept</u>, then <u>finish</u>.

Updating Project Compiler Settings:

Chances are the latest compiler version has changed since we supplied the demonstration projects. In this case the projects will by default throw an error "This project was created using a version of compiler that is not currently installed...". If this is the case, then we must update the project's compiler settings.

1. Right click the project in the Project Explorer window (A) and select Properties (B).



 Under <u>General > Project > Tool-Chain > Compiler Version</u> select the compiler version you just installed or wish to use. The dialog option should look similar to the picture below.

	General					$\diamond \bullet \diamond \bullet$
 Resource General Build C6000 Compiler 	Configuration: SM_V2 [/	Active]			~ Manag	e Configurations.
Processor Options Optimization Include Options Performance Advisor	B Project M Products Device Family: C6000					
Predefined Symbols Advanced Options C6000 Linker C6000 Hex Utility [Disable Debug	and the second s	type filter text>	~ Ge	neric C674x D	evice	~
	Connection.	the project's target-configur	ation autom	Verify	(applies to v	whole project)
beau	Tool-chain	former and the second se				
	Compiler version:	TI v8.3.0			~	More
	Output type:	Executable			~	
	Output format:	eabi (ELF)				
	Device endianness:	little			¥	
		SM_V2.cmd			~	Browse
	Linker command file:					

3. Click Apply & Close

Compiling A Project:

This process will generate an .OUT file which will then be used by the PC applications to program the SignalMaster device with the DSP code you wrote. If no errors occurred while compiling, then the .OUT file is automatically stored in the project folder by Code Composer.

Example .OUT file location:

 $C:\SignalMaster\DSPProjects\Sample001_PointByPointEcho\SM_V2\Sample001_PointByPointEcho. out$

1. Click the Build / Compile button.



Loading and Running a Project:

After you have compiled your application, you are ready to load and run the file on the SignalMaster hardware. You can load your programs using the SigMasLoader.EXE program. This program is found in the SignalMaster installation directory.

Make sure that the SignalMaster hardware is connected to one of your computer's USB ports and is turned on. The top left LED on the box will start to blink when it is turned on and ready to accept a program. If you do not see the LED blinking or if you need to reset the hardware at any time, you can toggle the on/off switch until you see the LED blinking.

Use the select button to browse to the location of your compiled .OUT file. Then select the Load to SignalMaster button to upload the program. The program will automatically start running.

🞬 SignalMaster DSP Software Loading Module 1.00 –	-		×
C:\SignalMaster_DelphiProjects\SigMas_Loader\Calib101.out		Sele	ct
Load to SignalMaster			
Call Parameters: OUTFILEPATH&NAME [CLOSE] Program will automatically load file specified in OUTFILEPATH&NAME (No spotional [CLOSE] = Program will close after loading.	paces	in Path a	lowed).

Note: Although CCS provides an option to load and run programs directly from within CCS, you cannot perform this option with the provided USB cable. Direct CCS hardware control and debugging options requires the use of a special Joint Test Action Group (JTAG) cable. We do not recommend using this option as it requires having your SignalMaster system hardware open. If you would like to use a JTAG cable with your SignalMaster hardware, please contact IHS for further information.

Starting a New Project:

When generating a new SignalMaster DSP program, we recommend that you start out from one of the provided examples that is most similar to what you want to accomplish and copy that project.

1. Simply right mouse click over the project you wish to copy and select the copy option on the popup menu:



2. Now right mouse click over an empty region of the window showing the list of projects and select paste from the popup menu:

🍐 Project Explorer 🔀 📃 📄	2 .		@ main.c 23	IHS_Utilities.h
> 202 Calib.202 > 203 Sample001 PointByPoint > 203 Sample002 PointByPoint > 203 Sample001 BiotKdchoi > 203 Sample010 BiotKdchoi > 203 Sample010 RobitStationMy > 203 Sample100 Wavelet > 203 Sample100 Sample100 Shifer > 203 Sample100 Shifer > 203 Sample100 Wavelet TEST > 203 Test.000 Wavelet TEST	ichoFilte er odule		481// 482// 483// 485// 485// 485// 485// 485// 490// 490// 490// 493// 493// 493//	0 8 2 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 0 8 4 9 0 8 4 4 9 0 8 4 4 4 9 0 8 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
> 🍄 Wavelet			496 // 497 498	e Eq2
		New		>
		Show in L	ocal Terminal	
		Add Files		
	100	Copy	75	Ctrl+C
	05	Paste		Ctrl+V
	×	Delete		Delete
		Relactor		>
		Source		2
		Move		
		Rename		F2
		Import		ż
	Aug.	Export		

3. Enter the name of your new project:

😨 Copy Proje	ct —		I X
Project name:	MyProgram		
Use default	location		
Location: C:\S	SignalMaster_DSPProjects\MyProgram		Browse
Cho	ose file system: default 🖂		
?	ОК	Ca	ancel

4. Your new program will now appear in the list of projects:



(Note: Current 2019 example program names begin with name Samp3xxx)

Selecting a New Workspace:

During installation, CCS will ask you to select a Workspace Directory. We recommend that you use the following Workspace Directory as the default:

C:\ SignalMaster\DSPProjects

If for any reasone, you need to have multiple work spaces, you can do this from the File menu, Switch Workspace option.



Code Composer Studio Compiler Settings (Advanced Options):

When you load one of the SignalMaster example programs into CCS, all required settings should automatically be preset in the project files. This section is only provided for reference in case you need to make any changes and so that settings can be verified.

Setting up the complier and linker options:

The following items are under the Project Menu, Properties Item:

	Proj	ect	Run	Scripts	Window	Help	
	1	Ne	w CCS	Project			
	6	Ne	w Ener	gia Sketcl	h		
ł		Еха	mples				
i		Bu	ild Proj	ect			
	010	Bu	ild All				Ctrl+B
1		Bu	ild Con	figuratio	ns		>
		Bu	ild Wo	king Set			>
		Cle	an				
)		Bu	ild Aut	omaticall	y		
		She	ow Bui	ld Setting	s		
1	Ê,	Im	port C	CS Project	is		
,	<u>í</u>	Im	port Le	gacy CCS	w3.3 Projec	ts	
		Ad	d Files.				
,		RT:	SC Too	ls			>
	9	Im	port En	iergia Ske	tch		
1	=	Im	port En	iergia Libi	raries		
1		C/(C++ In	dex			>
1	_	Pro	pertie	;			

After selecting this option, a Properties dialog box will appear showing options for resources and building your applications.

Make sure that the variable **IHSC674xHW_DIR** is declared and pointing to the SignalMaster installation directory: C:\SignalMaster\IHSC674xHW where the hardware support libraries are located. This variable is used in the projects in order to easily point to the directory where SignalMaster has been installed containing all the required files.

Properties for Sample010_BlockEcho				
ype filter text	Linked Resources		$\phi \bullet \phi \bullet$	
Resource Linked Resources Resource Filters General Build 5 C6000 Compiler	Path Variables Linked Resources Path variables specify locations in the file system, including other path variables with the syntax The locations of Inked resources may be specified relative to these path variables. Befined path variables for resource "Sampleti10_BlockEcho":			
 C6000 Linker C6000 Hex Utility [Disabled] Debug 	Name CCS.BASE ROOT	Value C:\bi\ccsv7\ccs base	New	
	CCS_BASE_ROOT	C:\th\ccsv7ccs_base C:\th\ccsv7	Ediţ	
1000	CG_TOOL_ROOT	C:\ti\ccsv7\tools\compiler\c6000_7.4.21	Bemove	
	ECLIPSE_HOME	Ci\th\ccsv7\eclipse\ C\\SignalMaster\IHSC674xHW	-	
	PARSHILOC PROFECTION TUPRODUCTS_DIR TUPRODUCTS_DIR_TIREX WORKSPACE_LOC	C:\SgnulMaster_DSPProjects C:\SgnulMaster_DSPProjects\SampleO10_BlockEcho Club Club C:\SignulMaster_DSPProjects		

Make sure that the rest of the dialog windows are set as shown in the images below:

General				¢•0•
Configuration: SM_V2 [A	ctive]		Manage	e Configuration
Main Main				
200 St. 1				
NOROCO SA CONTRACTOR SA CONTRA	type filter text> ~	Generic C674x Devi	ce	
Connection		Venfy ()	pplier to w	hole project)
Manage	the project's target-configuration a	utomatically		
Advanced settings				
Compiler version: 0	TI v7.4.18 [TI v7.4.21]		*	More
Output type:	Executable		4	
Output format:	eabi (ELF)		~	
Device endianness:	little		~	
Linker command file:	SM_V2.cmd		~	Browse
Runtime support library:	<automatic></automatic>		v	Browse
	Configuration: SM V2 [A Main B) Products Device Eamily: C0000 Yariane: cadeet or Conserblane Advanced settings Compiler version: 0 Outgut type: Outgut type: Outgut type: Device ormati. Device onmain.	Configuration: SMLV2 [Active] Configuration: SMLV2 [Active] Man Tab Products Concerning: C0000 Variate: satisfier texts v Concerning: Complete respect Varget configuration Advanced setting: Compiler variable Output type: Executable Unknet command file Unknet command file	Configuration SM, V2 [Active] Man Sh Products Device Eamily: Consection Verify- Manage the project's target-configuration automatically Advanced setting: Compiler version: • Tiv7A.18 [Tiv7A.21] Output type: Eancable Output type: Eancable Output type: Eancable Output type: Eancable Under eardianness Intel Linker command file SM, V2.Cmd	Configuration: SM V2 [Active] Manage Main: Noducts Second Device: Eamily: C0000 Spinan:

The Linker command file should be **SM_V2.CMD** – this file contains the memory map for the hardware and other important parameters needed to compile your application.

Properties for Test_AudicInOut15am	Perf		- 🗆 ×
type filter test	Basic Options		⇔ • • •
 Resource General Build C6000 Compiler Processor Options 	Configuration: Debug [Active]		 Manage Configurations
Optimization Debug Options	Specify output file name (output, file, -o)	'S(ProjName).out'	
Include Options Performance Advisor	Set C system stack size (stack, size, -stack)	0.600	10. 1
> Advanced Options	Input and output sections listed into <file> (map_file, -m)</file>	"S(ProjName).map"	
 C6000 Linker Basic Options 	Heap size for C/C++ dynamic memory allocation (-heap_size, -heap)	0.600	
CRORD Hex UNITY [Deubled] Debug			
Show advanced settings		[OK Cancel

Compiler Options:

Note that in the compiler and linker options, the path variable is shown in the include search path.

	C6000 Compiler		\$ • \$ • •
Resource General Build C6000 Compiler Processor Options	Configuration: SM_V2	[Active]	Manage Configurations
Optimization Debug Options	Command:	"\$(CG_TOOL_CL)"	
Include Options	Command-line pattern:	S[command] S[flags] S[inputs]	
Performance Advisor > Advanced Options	Summary of flags set:		
C6000 Hex Utility (Disabled) Debug	include_path="C:/Signa include_path="C:/Signa include_path="C:/Signa include_path="C:/Signa include_path="C:/Signa include_path="C:/Signa	Marter, DSPProject, Sampki DD, Blockéchor " Marter, MSC 3444W/mchuder " Marter, MSC 3444W/mchuder Children Marter, MSC 3444W/mchuder Children Marter, MSC 3444W/mchuder Children Marter, MSC 3444W/pdB, Children, J. 4, 0, 0 packages " Marter, MSC 37444W/mSL 35 opcdefine: c5748define diag_vrape.offdiag_varming=225	n:145_5M/2

Make sure that the following are specified under the compiler options:

-mv6746 --abi=eabi -Ooff -g --include_path="C:/ti/ccsv7/tools/compiler/c6000_7.4.21/include" --

include_path="C:/SignalMaster_DSPProjects/Sample010_BlockEcho" --

include_path="C:/SignalMaster/IHSC674xHW/include" --include_path="C:/SignalMaster/IHSC674xHW/include/hw" --

include_path="C:/SignalMaster/IHSC674xHW/include/c674x" --

include_path="C:/SignalMaster/IHSC674xHW/include/c674x/c6748" -

include_path="C:/SignalMaster/IHSC674xHW/grlib" --

include_path="C:/SignalMaster/IHSC674xHW/dsplib_c674x_3_4_0_0/packages" --

include_path="C:/SignalMaster/IHSC674xHW/IHSLib" --gcc --define=c6748 --define=IHS_SMV2 -

display_error_number --diag_wrap=off --diag_warning=225

Properties for Sample010_BlockEcho			– 🗆 ×
type filter text	Include Options		⇔ •⇔••
 Resource General Build C6000 Compiler Processor Options 	Configuration: SM_V2 [Active]	v	Manage Configurations
Optimization Debug Options Include Options Performance Advisor > Advanced Options > C6000 Here Unifity [Disabled] Debug	Add dir to Filectude search path (include_path, -I) FISEORCE (DOI: 10:0011/004807 = FISEORCE () (DOI: 10:0011/004807 = FISEORCE		ର ଲ ି ଲି ରାହି
	Specify a preinclude file (preinclude)		€ £1 £8 €1 £
⑦ Show advanced settings			OK Cancel

Linker Options:

ype filter text	C6000 Linker		⇔ ⇒ + .	
 Resource General Build C6000 Compiler Processor Options 	Configuration: SM_V2	2 [Active]	Manage Configurations.	
Optimization Debug Options	Command:	"S(CG_TOOL_CL)"		
Include Options	Command-line pattern:	S(command) S(flags) S(output_flag) S(output) S(inputs)		
Performance Advisor > Advanced Options	Summary of flags set:			
File Sarah Path > Advanced Options C6000 He. Ubity [Disabled] Debug	i*C:/SignalMaster/HSC i*C:/SignalMaster/HSC i*C:/SignalMaster/HSC i*C:/SignalMaster/HSC i*C:/SignalMaster/HSC i*C:/SignalMaster/HSC	upler(3002,7.42.17.8 ⁻¹ *C-ft/ket/scient/action/compute(3002,7 57.44W)(humary(57.44),cg, ccs/st/ht/behof* 57.44W)(humary(57.44),cg, ccs/st/ht/behof* 57.44W)(humary(57.44),cg, ccs/st/ht/behof* 57.44W)(humary(57.44),cg, ccs/st/ht/behof* 57.44W)(humary(57.44),cg, ccs/st/ht/behof* 57.44W)(humar)(57.45),cg, cs/st/ht/behof* 57.44	ibug" - id_libswarn_sections	

Make sure that the following hardware specific paths are specified:

-i"\${IHSSignalMaster}/IHSC6748HW/binary/c674x/cgt_ccs/utils/Debug"

-i"\${IHSSignalMaster}/IHSC6748HW/binary/c674x/cgt_ccs/c6748/drivers/Debug"

-i"\${IHSSignalMaster}/IHSC6748HW/binary/c674x/cgt_ccs/c6748/system_config/Debug"

-i"\${IHSSignalMaster}/IHSC6748HW/binary/c674x/cgt_ccs/c6748/lcdkC6748/platform/Debug"

-i"\${IHSSignalMaster}/IHSC6748HW/binary/c674x/cgt_ccs/grlib/Debug"

Linker Include Libraries:

type filter text	File Search Path	6+0+
> Resource General > Build > C6000 Compiler	Configuration: [SM,V2 [Active] ~	
Processor Options Optimization Debug Options Include Options Performance Advisor > Advanced Options C6000 Linker Basic Options File Search Path > Advanced Options	Include library file or command file as input (library, -1) untra library untra library drivers.lib platform.3b system.config.lib system.config.lib dipite.lib dipite.lib	ରେ ଲ େ ଲେ ହୋଏ
C6000 Hex Ubility (Doabled) Debug	Add <dirz (iserch.path,)="" 04%="YiG," 100011="" 200,="" 200011="" 70404,="" 70447,="" coll="" d="" doot="" dort="" g.="" ji,="" jinhoude="" jinhoude*="YiJACG" jinhoude<="" likewy="" nobude*="YiJACG" path="" q="" search="" stor="" t="" td="" to="" tool="" yijacg=""><td><u>ରେ ଲ</u>େ ର ୬ ।</td></dirz>	<u>ରେ ଲ</u> େ ର ୬ ।
	Search libraries; resolve backward references (-reread_libr, -x) Classle automatic RTS selection (-reliable_auto_rts)	

Make sure that the following libraries are shown: "libc.a" utils.lib drivers.lib platform.lib system_config.lib grlib.lib

Configuration: Debug [Active]	 Manage Configurations
🗆 Builder 💿 Behaviour 🌲 Steps 📸 Variables 🚾 Environment 🥎 Builder	Link Order 📲 Dependencies
☑ Use default build command	
Build command: \$(CCS_UTILS_DIR)/bin/gmake -k	Variables
Makefile generation	
Generate Makefiles automatically Expand Env. Variable Refs in Mak	efiles
Build location	
Build girectory: \$(workspace_loc:/Test_AudioInOut1SamPerf)/Debug	
Workspace	File system
See <u>General</u> for changing tool versions and device settings	
	Builder Elide default build command Build germand: S(CCS_UTILS_D(R)/bin/gmake -k Makefile generation Signerate Makefiles automatically Espand Env. Variable Refs in Make Build Screet Build girectory: S(workspace_loc:/Test_Aus/oln/Out1SamPerf)/Debug Workspace_

		Q • Q •
Configuration: Debug [Active]	~ Ma	nage Configurations.
🖾 Builder 🛞 Behaviour 📫 Steps	💫 Variables 🛛 🧮 Environment 😽 Link Order	• Dependencies
8	Use optimal jobs (8) Use parallel jobs: 8 +	
Workbench Build Behavior Workbench build type:	Make build target:	
Build on resource save (Auto build)	all	Variables
Note: See Workbench automatic build p	preference	
Build (Incremental build)	all	Variables
🖸 Clean	clean	Variables
	Build settings	□ Stop on first build error □ Enable parallel build ● Use optimal jobs (8) ● Use optimal jobs (8) ● Use optimal jobs (8) ● Use optimal jobs (8) ● Workbench Build Behavior ● Use optimal jobs (8) ● Workbench Build Behavior ● Make build target: ● Build on resource save (Auto build) all ■ Build (incremental build) all

ype filter text	Build				⇔ • ⇔ •
 Resource General Build C6000 Compiler Processor Options 	Configuration: Debug [Act	tive]		v [Manage Configurations.
Optimization	🔲 Builder 🛞 Behaviour	📬 Steps 🙀 Variables 🎴	Environment	Ny Link Or	der 📲 Dependencies
Debug Options Include Options	Variable	Value	Origin		Add
Performance Advisor 3 Advanced Options	CCS_JAVA_HOME CCS_UTILS_DIR	C:\tr\ccsv7\eclipse\jre C:\tr\ccsv7\utils	BUILD SYSTEM		Select
> C6000 Linker C6000 Hex Utility [Disabled]	CWD	C:\SignalMaster_Project	USER: CONFIG		Edit
Debug	PWD	C:\SignalMaster_Project	USER: CONFIG		Delete
					Undefine
	Append variables to nativ Replace native environm				
	See 'General' for changing tool	versions and device settings			

Note that in the above image, CCS_UTILS_DIR shows, C:\ti\ccsv7\utils – the ccsv7 is a reference to the version (version 7) of CCS that was installed at the time the image was generated. If you have installed a later or earlier version of CCS, that string may read ccsv8 (for version 8). This will not affect the compilation of your program.

Other tabs: Steps, Variables. Link Order and Dependencies should be blank.

Project Menu:

Properties...

Include Options – make sure that the correct directory is shown in the path

Developing PC Applications:

PC applications can be developed in any programming language that allows the use of a DLL. These include all high level languages such as C++, Pascal and Visual Basic. Other programming environments such as Matlab and LabView also allow the use of DLLs. The DLL provides routines to load programs to the DSP and to communicate with the DSP during execution. For a full list of DLL commands and data types, see the **IHS SignalMaster DLL Commands** section below.

Every PC application will need to use DSP program. The DSP program receives commands from the PC and either uploads data from the PC or downloads data to the PC at the PC's request. Applications can be written to present sounds at different frequencies and intensities and record from the DSP microphones. The DSP can also work independently and automatically process the microphone signals and output them back through the earphones. The PC can also simply provide processing parameters to adjust band pass filters to change the processing and characteristics of signals being presented to a subject.

After setting up the DLL declarations in your program as shown in the **IHS SignalMaster DLL Commands** section below, you will be able to load DSP programs and start sending commands and data to the DSP.

First load the DSP code by calling the SM_Initialize routine specifying the location of the .OUT file that is generated by the TI Code Composer Suite (CCS) (See **TI Code Composer Studio Setup** section below). Make sure that both the **our2sprc.exe** program and **SIGMAS.DLL** are in the same directory as your application.

SM_Initialize('C:\MyPath\MyDSPProgram.OUT');

You may also program a DSP application with all processing parameters coded within the DSP code and not use a PC program. In this case, you can simply load a DSP application by running the **SigMasLoader.EXE** program (See the **Loading Programs to the DSP** section below for additional information).

Sending Function Calls and Data to the DSP:

Data may be sent to DSP using one of several arrays types defined in the SignalMaster DLL. To transfer bytes, simply use a byte array (TxArray in this example). You can load the values into the array:

TxArray[1]:= value1; TxArray[2]:= value2; : TxArray[n]:= valuen;

Then call the SM_PutByteArray command specifying the array and number of bytes.

SM_PutByteArray(TxArray,n);

You may also use a Short Integer (16 bit) and Integer (32 bit) type arrays with their corresponding calls.

After you perform the transfer call, the data is sent to the DSP, but you need to let the DSP know that it has data waiting to be process, so you will need to implement a function call:

SM_DSPFuncTx(MyFunctionNumber, n);

Where MyFunctionNumber is a function that you have defined in your DSP code and n is the number of bytes that the function will read. Make sure that the number of bytes (n) matches the actual number sent.

Requesting Data from the DSP:

To request data from the DSP, you will need to call another function you have defined for that purpose:

SM_DSPFuncTx(MyFunctionNumber, n);

In this case, n can be equal to 0 if you are not sending any parameters to that function. In some cases, you might want to read data from one microphone or another, you may specify the microphone number by first call filling in the corresponding data array and then transferring the parameters using the **SM_PutByteArray(TxArray,n)** function before calling the DSP function as done in the previous example.

After the function call, read the data by first calling:

n:=SM_DSPFuncRx(MyFunctionNumber);

The function will return the number of bytes (n) transferred by the DSP. Note that if you call multiple data request function calls without calling the SM_DSPFuncRx receive function, the previous data requests will be discarded.

After the SM_DSPFuncRx, you will need to transfer the actual data from the DSP to the PC application. Simply call the corresponding SM_Get function corresponding to the data type you wish to read:

SM_GetIntArray(RxArrayInt,(n/4));

In the above example, the Integer array RxArrayInt will be filled with n integers of data. Important note: The SM_DSPFuncRx function returns the number of bytes transferred, however, the SM_Get functions will read the number of data points for the corresponding data type. In this example, the data type is integer (32 bits = 4 bytes), therefore n is divided by 4 when calling the SM_GetIntArray function.

With these basic command, you will be able to develop complex applications that call on the SignalMaster DSP to perform any number of functions you define and transfer data to and from the DSP. Several PC application source code examples are provided in the PCProjects subdirectory under the various programming languages. Although currently all the examples are provided in Delphi Pascal, additional examples in other languages will also be provided soon.

IHS SignalMaster DLL Commands:

Type Declarations:

ArrayType	= Array[140000] of Byte;
ArraySIntType	= Array[120000] of SmallInt;
ArrayIntType	= Array[110000] of Integer;

//Byte Array
//Small Integer Array (16 Bits)
//Integer Array (32 Bits)

Functions:	
------------	--

Name:	Returns:	Description:
SM_DLLVer	PANSIChar	Returns a pointer to an array of characters
		with the version of the dll in use.
SM_Initialize(DSPProgram:ShortString)	Integer	Initializes the SignalMaster system and
···· <u>·</u> ·······························		loads the DSPProgram OUT file.
SM_Initialize(n:Integer)	Integer	Opens communication using COM port
_ ()	U	specified by n without loading a DSP
		program. This is used when working with
		a JTAG debugging connection and the
		program has already been uploaded
		directly from the complier and you simply
		wish to establish communication with your
	-	PC application.
SM_Close	Integer	Closes SignalMaster system and resets
		Bootloader to wait for next program
		upload. If you terminate an application
		without calling this function, you will need to turn your SignalMaster system off and
		on to reset the bootloader. Otherwise, you
		will not be able to load another application.
SM_DSPFuncTx(FuncNum,Count:Integer)	Integer	Calls a user defined or IHS predefined
	Integer	DSP function number FuncNum to send
		data to the DSP. Counter specifies the
		number of bytes passed as parameters for
		use by the function.
SM_DSPFuncRx(FuncNum:Integer)	Integer	Calls a user defined or IHS predefined
		DSP function number FuncNum to recieve
		data to the DSP. The returned integer
		specifies the number of bytes passed to
		the PC from the DSP.
SM_GetByteArray(var A:ArrayType;	Integer	Download Count bytes from byte array.
Count:Integer)		
SM_GetSIntArray(var A:ArraySIntType;	Integer	Download Count bytes from Small Integer
Count:Integer)	late est :	(16 bit) array.
SM_GetIntArray(var A:ArrayIntType;	Integer	Download Count bytes from Integer (32
Count:Integer)	Intogor	bit) array.
SM_PutByteArray(var A:ArrayType;	Integer	Upload Count bytes from byte array.
Count:Integer) SM_PutSIntArray(var A:ArraySIntType;	Integer	Upload Count bytes from Small Integer (16
Count:Integer)	nitegel	bit) array.
SM_PutIntArray(var A:ArrayIntType;	Integer	Upload Count bytes from Integer (32 bit)
Count:Integer)	integer	array.
oounininogor/		

Implementation:

var hdll:THandle; Const Туре

ArrayType= Array[1..40000] of byte;ArraySIntType= Array[1..20000] of SmallInt;ArrayIntType= Array[1..10000] of Integer;HeaderType= Array[1..8] of byte;

Function SM_DLLVer:PANSIChar; cdecl; EXTERNAL SIGMASDLL; Function SM_Initialize(DSPProgram:ShortString):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_DSPFuncTx(FuncNum,Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_GetByteArray(var A:ArrayType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_GetSIntArray(var A:ArraySIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_GetIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_GetIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutByteArray(var A:ArraySIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutByteArray(var A:ArraySIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutSIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutSIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutSIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL; Function SM_PutIntArray(var A:ArrayIntType; Count:Integer):Integer; cdecl; EXTERNAL SIGMASDLL;

var

TxArray: ArrayType;TxArraySInt: ArraySIntType;TxArrayInt: ArrayIntType;TxArrayInt: ArrayIntType;RxHeader: HeaderType;RxArray: ArraySIntType;RxArraySInt: ArraySIntType;RxArrayInt: ArrayIntType;

//You must first initialize the DLL on your application... procedure TAudioANC.FormCreate(Sender: TObject);

begin

Hdll:=0; //Initialize...

Hdll:=LoadLibrary(PChar('SIGMASDLL.dll'#0));

If Hdll=0 then ShowMessage('Error Loading IHSInterface DLL Library');

end;

Developing DSP Applications:

To modify the DSP code provided or develop your own applications, you will need to use the TI Code Composer Studio (CCS). Instructions are provided in the TI Code Composer Studio (CCS) Setup section on how to set up CCS on your computer. Please make sure you follow the CCS setup instructions very carefully. Additional information is also provided from TI documentation. IHS code examples are provided in the distribution thumb drive, in the DSPProjects subdirectory and from the SignalMaster webpage.

At this point you should have already installed CCS and selected a Work Space directory. We recommend using C:\ SignalMasterProjects. You should have also Imported the samples provided by IHS into your work space directory.

The easiest way to start a new DSP application is to look for one of the examples provided by IHS that is most similar to the application you want to develop. In CCS, simply right mouse click on the application you want to copy, and press copy from the drop down menu:



Starting a new DSP project: 1) Right mouse click on desired project and select copy, 2) Right mouse click on open area of Project Explorer (Left Panel) and select Paste, 3) Enter name of new project.

You code should be primarily limited to the main.c file. Other files and libraries contain important files that if modified could affect the function of the DSP and other project using the same file.

In the Main.C file, the DSP application has two main areas of code:

- 1) Initialization
- 2) While Loop

1) The initialization code is used to set up all the important parameters, registers and variables needed for the DSP to operate. Most of the important hardware components are initialized in the procedure: IHS_initializeDSP(); , but others require your attention as shown in the code example below (Sample3010_BlockEcho). 2) The main application While loop is where all the work is done while your application is running. The main application While is subdivided in to 2a) a region used to monitor commands from an external program and 2b) a region used for data acquisition, output and signal processing.

Therefore, your DSP application source code should focus primarily in the main While loop in the Main.C file. This loop deals with receiving function calls from the PC program and sending and receiving data based on your define function calls. After the DSP code is loaded into the system, the DSP waits in an infinite loop (see the sample source code below) for instructions from the host PC and for hardware interrupt notifications that data is ready to be processed. All DSP programs should contain code as shown in the example below:

Sample3010_BlockEcho:

```
#include "main.h"
PRIVATE VARIABLE DEFINITIONS
/* Application-----*/
                                        [NUM I2S CHANNELS][AUDIO SAMPLES MAX];
int stim
int audio_rx [NUM_I2S_CHANNELS][AUDIO_SAMPLES_MAX];
struct FilterStorage_t filter_handle [FILTER_NUM_MAX];
int main(void)
{
     /* Hardware Initialization-----*/
     /* Setup the DSP */
     struct CommandHandle t *command handle p = IHS InitializeDSP();
     int32_t *data_array_int_ptr = (int32_t *)&(command_handle_p->data_array[0]);
     /* Setup the audio codec and its interfaces */
     double sampling_rate = 24000;
     codecSettings_t codecSettings;
                                                CODECONFIG_USE_DMA;
     codecSettings.dmaFlag =
                                                sampling rate; /* 24kHz, 44.1, 48, 96, 192 */
     codecSettings.samplingRate =
                                          WORD_SIZE;
     codecSettings.wordSize =
                                              1; /* Enable headphone output for left ch */
    codecSettings.enableHPL = 1; /* Enable headphone output for left <u>ch</u> */
codecSettings.enableHPR = 1; /* Enable headphone output for right <u>ch</u> */
codecSettings.enableIN1L = 1; /* Enable <u>mic</u> input for left <u>ch</u> */
codecSettings.enableIN1R = 1; /* Enable <u>mic</u> input for right <u>ch</u> */
codecSettings.enableLOR = 1; /* Enable <u>aux</u> line output for left <u>ch</u> */
codecSettings.enableIN2R = 0; /* Enable <u>aux</u> line input for right <u>ch</u> */
codecSettings.gainMicLeft = 0; /* 0 to 47.5 dB */
codecSettings.gainHeadphoneLeft = 0; /* -6 to 29 dB */
     codecSettings.enableHPL =
```

```
codecSettings.gainLineoutRight = 0; /* -6 to 29 dB */
codecSettings.loopback_codec = 0; /* The codec will route ADC data to its DAC */
/* Application Initialization-----*/
uint32 t i, ii;
uint32_t sample_size = 600; //Should be less than or equal to AUDIO SAMPLES MAX
if(sample size > AUDIO SAMPLES MAX)
    IHS_ErrorSend(COMMAND_ID_ERROR_APP_SAMPLESIZE, __LINE__, __FILE__);
/* Update the transmit arrays */
IHS_Audio_WriteTxBuffers(&stim[0][0], &stim[1][0], sample_size);
                                                                    ----*/
/* Begin the audio loop-----
int32 t result = IHS InitializeCodec(&codecSettings, sample size);
I2SDataTxRxActivate(&codecSettings);
while(1)
{
    /* Check if the PC has sent data via Bluetooth or USB-UART */
    if(IHS_CommandRxCheck() == COMMAND_RXCHECK_AVAILABLE)
    {
        switch(command handle p->id)
        {
            /* Send left mic to PC */
            case 201:
            i = 0;
            for(ii=0 ; ii<sample_size ; ii++)</pre>
            {
                data array int ptr[ii] = (int)audio rx[i][ii];
            }
            IHS CommandSend(201, sample size*sizeof(int32 t));
            break;
            /* Send right mic to PC */
            case 202:
            i = 1;
            for(ii=0 ; ii<sample_size ; ii++)</pre>
            {
                data_array_int_ptr[ii] = (int)audio_rx[i][ii];
            }
            IHS CommandSend(202, sample size*sizeof(int32 t));
            break;
            /* Undefined function call from PC */
            default:
            IHS_ErrorSend(COMMAND_ID_ERROR_PROCO_UNDEFINED_FUNCTION_RX, __LINE_, __FILE__);
            break;
        }
    }
    /* Check if we finished receiving a new array */
    if(IHS Audio RxCheck())
    {
        IHS LED Set(1, 1);
        IHS_Audio_UpdateBuffIndex();
        /* Read Last Buffer */
        IHS Audio Receive(&audio rx[0][0], &audio rx[1][0], sample size);
```

```
/* Send Next Audio Buffers */
IHS_Audio_Send(&audio_rx[0][0], &audio_rx[1][0], sample_size);
IHS_LED_Set(1, 0);
}
```

}

Sample DSP Programs Provided:

The following example DSP programs are provided:

Sample Name:	Description:
Sample3001_PointByPointEcho	Acquires data on a point by point basis based on the selected
	clock rate and outputs same point to speakers.
Sample3002_PointByPointEchoFilter	Acquires data on a point by point basis based on the selected
	clock rate, applies filters and outputs same point to speakers.
Sample3010_BlockEcho	Reads a block of data from the microphones and outputs the
	blocks to the speakers.
Sample3011_BlockEchoFilter	Reads a block of data from the microphones and outputs the
	blocks to the speakers after applying IIR filters. Weight
	constants can be added to the filter banks to provide different
	gain factors for each bank. This would be similar to a linear
	analog hearing aid.
Sample3020_CalibrationModule	Generates calibration tone and outputs through speakers.
	Also reads data through microphone for analysis. This
Comple2400 Weyelet	application can be used with
Sample3100_Wavelet	Reads a block of data and performs an FFT. The FFT can be
	weighted to adjust the outputs of each bin. An iFFT is preformed and the result outputted. An overlapping window
	scheme is used compensate for window-to-window amplitude
	difference and edge effects. This program can be used with
	PC application – SigMasEqual.EXE in folder
	\PCProjects\Delphi_Pascal\SigMas_Equalizer
Sample3101_Shifter	Acquires a block of data, performs FFT and does a frequency
· –	shift of components, then performs iFFT to generate time
	domain output. Resulting sound output has a remapping of
	frequency components. This program can be used with PC
	application – SigMasEqual.EXE in folder
	\PCProjects\Delphi_Pascal\SigMas_Equalizer. You will need
	to modify the source code to load this OUT file instead of the
	default Sample100_Wavelet.out
Sample3200_WaveletCompressor	Performs compressor algorithm for HA applications with two
	transfer functions. Performs a 128 point FFT, selects the
	appropriate transfer function based on the sound level, attack
	and release timing of each frequency band, and perform an
	iFFT in less than 5.33 msec for both ears. This program can
	be used with PC application – SigMasComp.EXE in folder \PCProjects\Delphi_Pascal \SigMas_Compressor
	<u> </u>

Debugging DSP Programs:

The easiest method to debug your DSP applications is to develop your function calls one at a time and make sure they return specific values that you can check to make sure they are functioning as expected. You can also turn the LEDs on your device on and off in order to monitor the progress of your application as the execution move from one routine to another.

To turn ON an LED call:

IHS_LED_Set(n,1); // Where n=LED number (1-4)

To turn ON ALL LEDs call:

IHS_LED_Set (0,1); // Turns on ALL LEDs (1-4)

To turn OFF an LED call:

IHS_LED_Set (n,0); // Where n=LED number (1-4)

To turn OFF ALL LEDs call:

IHS_LED_Set (0,0); // Turns OFF ALL LEDs (1-4)

Note: Although CCS provides the ability to do live debugging of code during the development process from the compiler, this feature is not directly available in the enclosed SignalMaster product ver 2.00.

Loading Programs to the DSP:

After you have developed your own DSP applications, you can load your programs on to the DSP using the SigMasLoader.EXE program. The source code for this program is provided in the PCProjects\Delphi_Pascal\SigMas_Loader subdirectory. The program uses the DLL to call the SM_Initialize routine to load the corresponding DSP compiled program (OUT File). When you run this program, it will ask you to select the DSP program you want to run.

🞬 SignalMaster DSP Software Loading Module 1.00	_		×
C:\SignalMaster_DelphiProjects\SigMas_Loader\Calib101.out		Sele	ct
Load to SignalMaster			
Call Parameters: OUTFILEPATH&NAME [CLOSE] Program will automatically load file specified in OUTFILEPATH&NAME Optional [CLOSE] = Program will close after loading.	(No space	s in Path a	allowed).

You can also call this program from another application using call parameters. The first parameter should be the path and name of the DSP compiled program you wish to load. For example, if you wish to run the program, MyDSPProgram.OUT, you can call: "SigMasLoader.EXE C:\MyDirectory\MyDSPProgram.OUT". This will automatically load that program. If you also want to close SigMasLoader after loading the program, simply add [CLOSE] as a second parameter: "SigMasLoader.EXE C:\MyDirectory\MyDSPProgram.OUT [CLOSE]". When passing parameters to another application, remember not to use spaces in the file path or file name. Spaces indicate a new parameter and will result in the loading program not being able to find you DSP program.

SignalMaster Ver 2.00 Specifications:

Processing Family:	Texas Instrument TMS320C6746 32/64 bit floating point processor.
	For complete datasheet, visit:
	http://www.ihsys.com/ohsspds/signalmaster/tms320c6746.pdf
Processing Speeds:	The TMS320C6746 is capable of performing 2100 million floating point operations per second (MFLOPS) with a 2.8 ns cycle time
Codec:	TLV320AIC3254 Ultra Low Power Stereo Audio Codec with imbedded miniDSP. For additional information, visit: http://www.ihsys.com/ohsspds/Documents/slaa408a.pdf
Memory:	256KB (Kilo Bytes) of internal memory and addressing lines providing access to 4MB (Mega Bytes) of external asynchronous memory.
Data Sampling Rate:	Programmable with rates from 8 to 96kHz.
PC Communication:	USB & Ethernet
Analog Input:	Microphone inputs with stereo drivers & A/Ds (programmable 16/20/24/32 bits)
Analog Output:	Stereo headphones outputs with drivers & D/As (Programmable 16/20/24/32 24 bits)
Universal Asynchro	nous Receiver/Transmitter (UART) communication modules: The system provides a USB/Serial communication and EtherNet UARTs for data exchange with a PC for programming, parameter selection and data exchange.
LEDs:	Battery power indicators and programmable logic indicators.
Power Supply:	Options to run off battery or A/C power adaptor.
Battery:	ICR18650 Li-Ion 2 X 3.7 V 2600 mAh (10 hours) (1.67 oz each)
Size:	190 mm X 90 mm X 30 mm
Weight:	<8 oz



Photo of SignalMaster device showing dual microphones and stereo headphones connected to the system. The system can use any standard 8-12 ohm headphones.

SignalMaster[™] Board



Photo of inside of SignalMaster enclosure showing DSP circuit board, battery charging circuit board, power switch, and power, microphone, headphone connectors.

USE EXTREME CAUTION WHEN OPENING THE SignalMaster ENCLOSURE Sensitive electronic components can be damaged by electrostatic discharge. Do not mishandle or damage Lithium-Ion batteries. Do not ship system if batteries are damaged. Follow all Lithium-Ion battery shipment regulations.

Fire Hazard if batteries shorted, mishandled or damaged!

ACAUTION



LITHIUM ION or LITHIUM POLYMER RECHARGEABLE BATTERIES INSIDE

Do not damage or mishandle this package. If package is damaged, batteries must be quarantined, inspected and repacked. For additional information, call:

> Intelligent Hearing Systems Corp. 1-800-447-9783

Technical Support:

For any questions or technical support with your SignalMaster system, email: support@ihsys.com Please make sure to enter "SignalMaster Support" in the subject line and the serial number of your system.