C File System File Functions

EXPERIMENT 1.2

Propose of the experiment

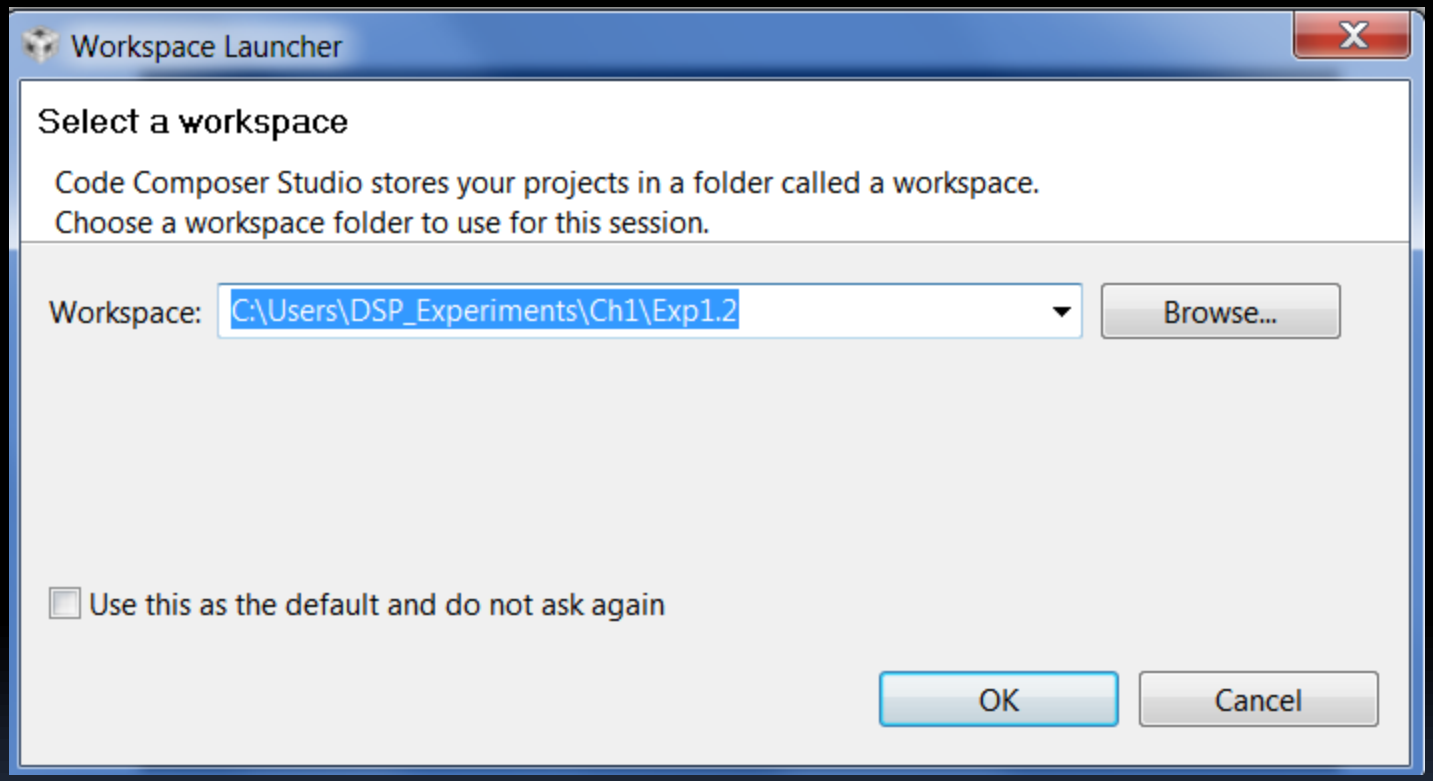
- Continue from previous experiment to be familiar with CCS environment
- Write a C language file input / output (CIO) program to read in a data file and write out a data file
- Understand how binary data will be handled by CCS differently than the ASCII text data
- The CIO feature introduced in this experiment will be used in future experiments

Start CCS

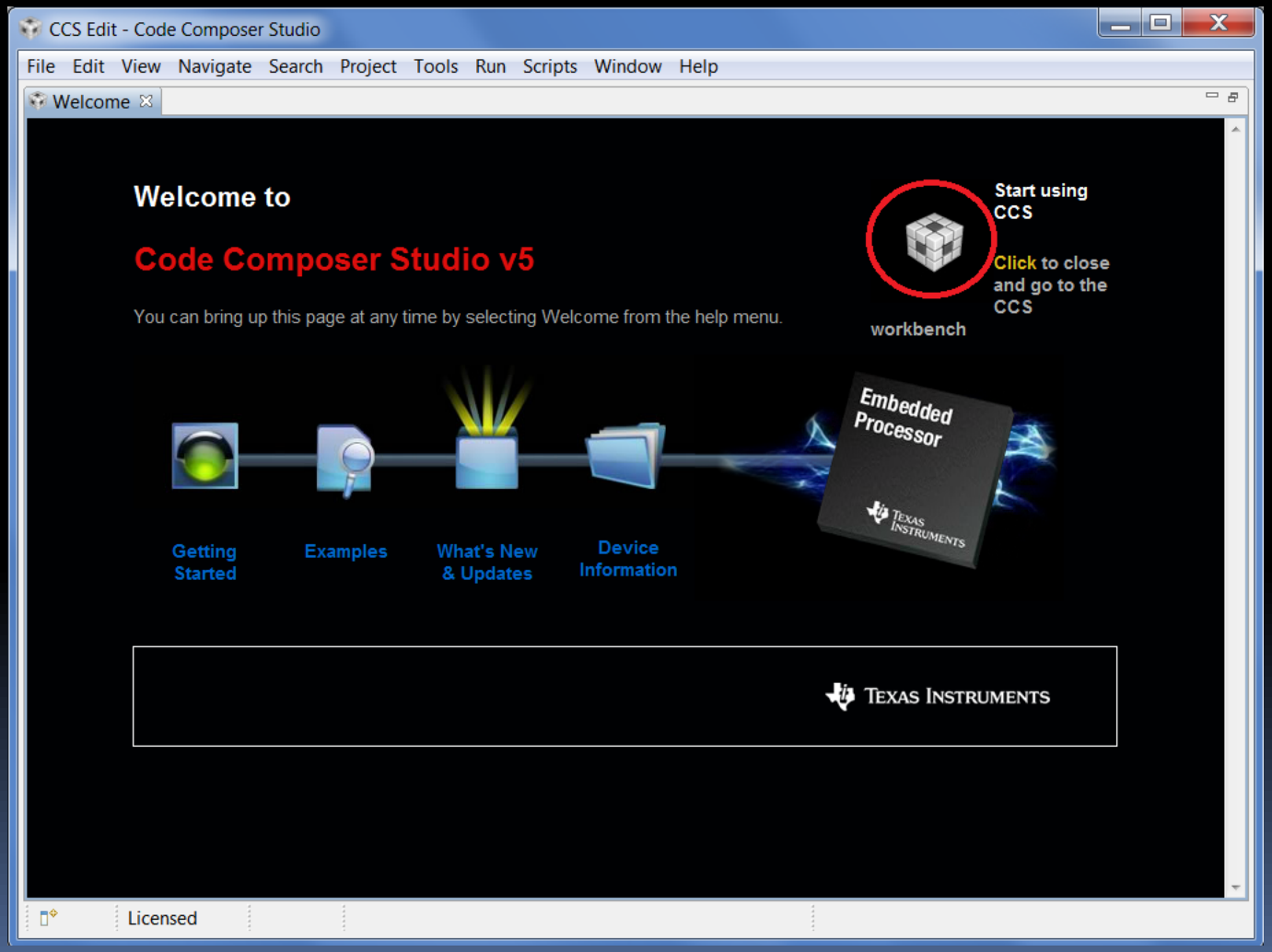
(Example: Code Composer Studio Version 5)

Create workspace

(Example: C:\User\DSP_Experiment\Ch1\Exp1.2)

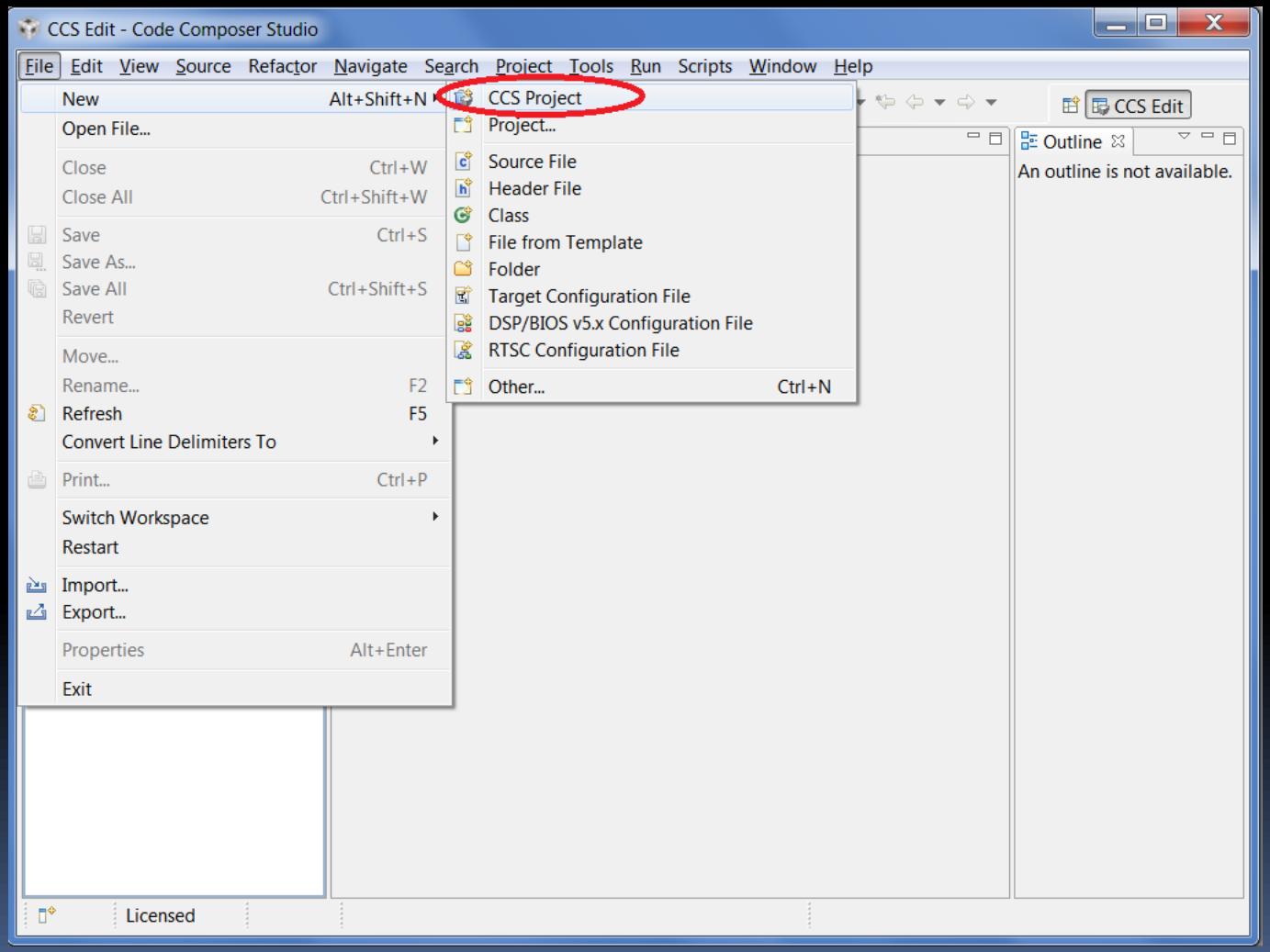


Go to CCS


CCS Edit - Code Composer Studio
File Edit View Navigate Search Project Tools Run Scripts Window Help
Welcome
Welcome to
Code Composer Studio v5
You can bring up this page at any time by selecting Welcome from the help menu.
Start using CCS
Click to close and go to the CCS workbench
Embedded Processor
Texas Instruments
Getting Started
Examples
What's New & Updates
Device Information
Texas Instruments
Licensed

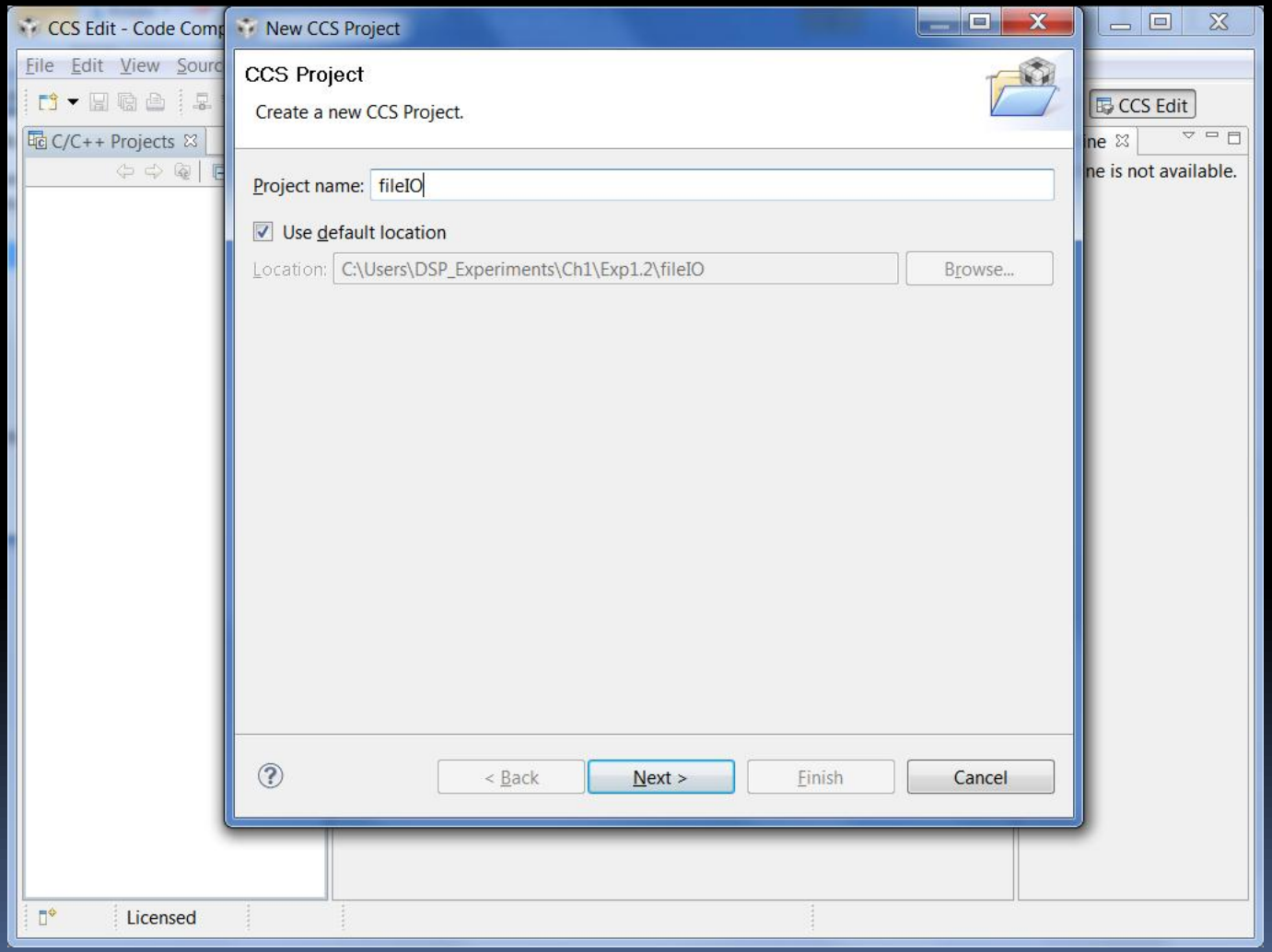
Create a new project

(File -> New -> CCS Project)

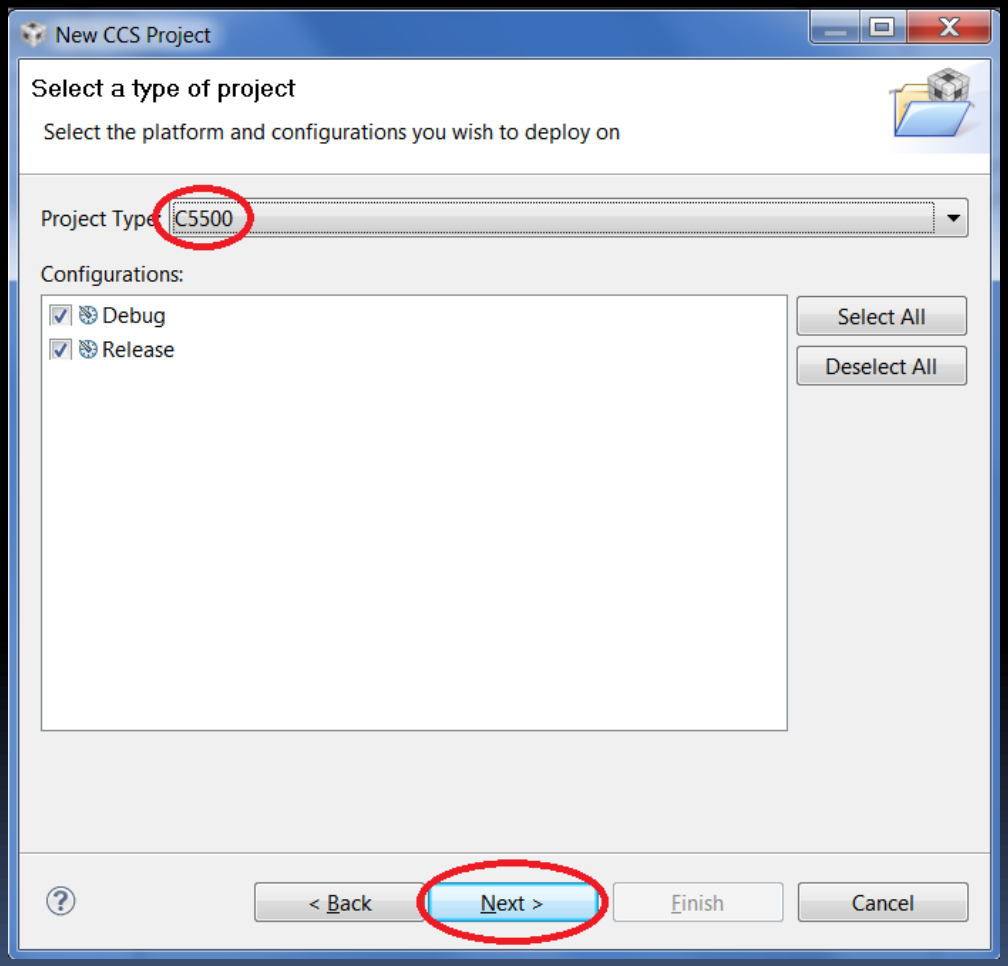


Create a new project name

(Example: fileIO)

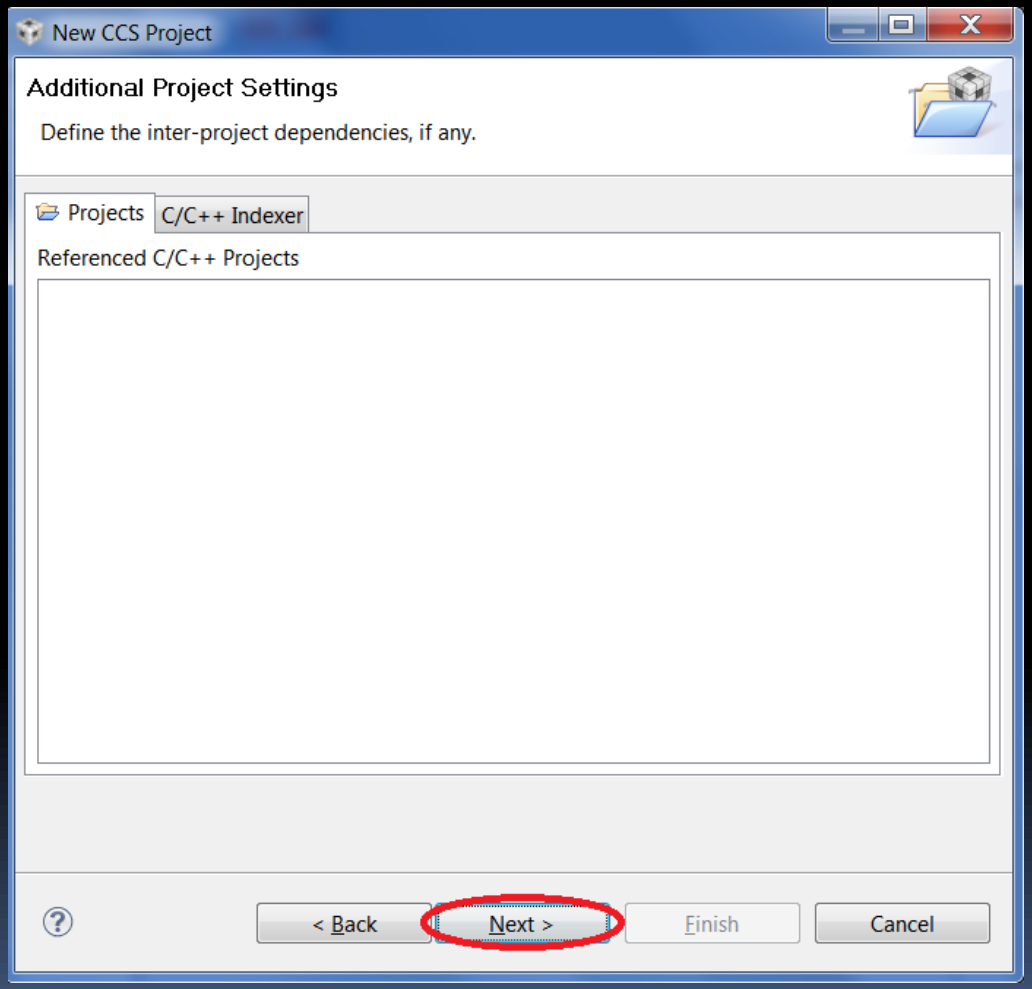


Select C5500 as the new project



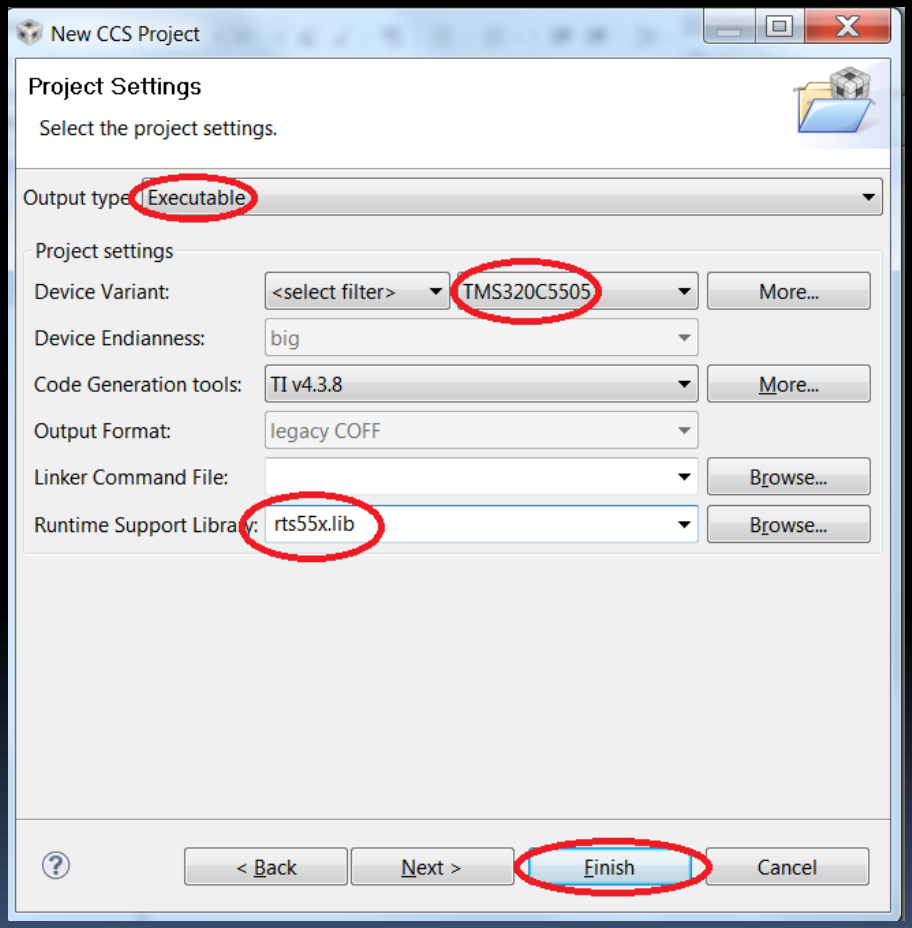
Go to next

(We do not need any additional settings)

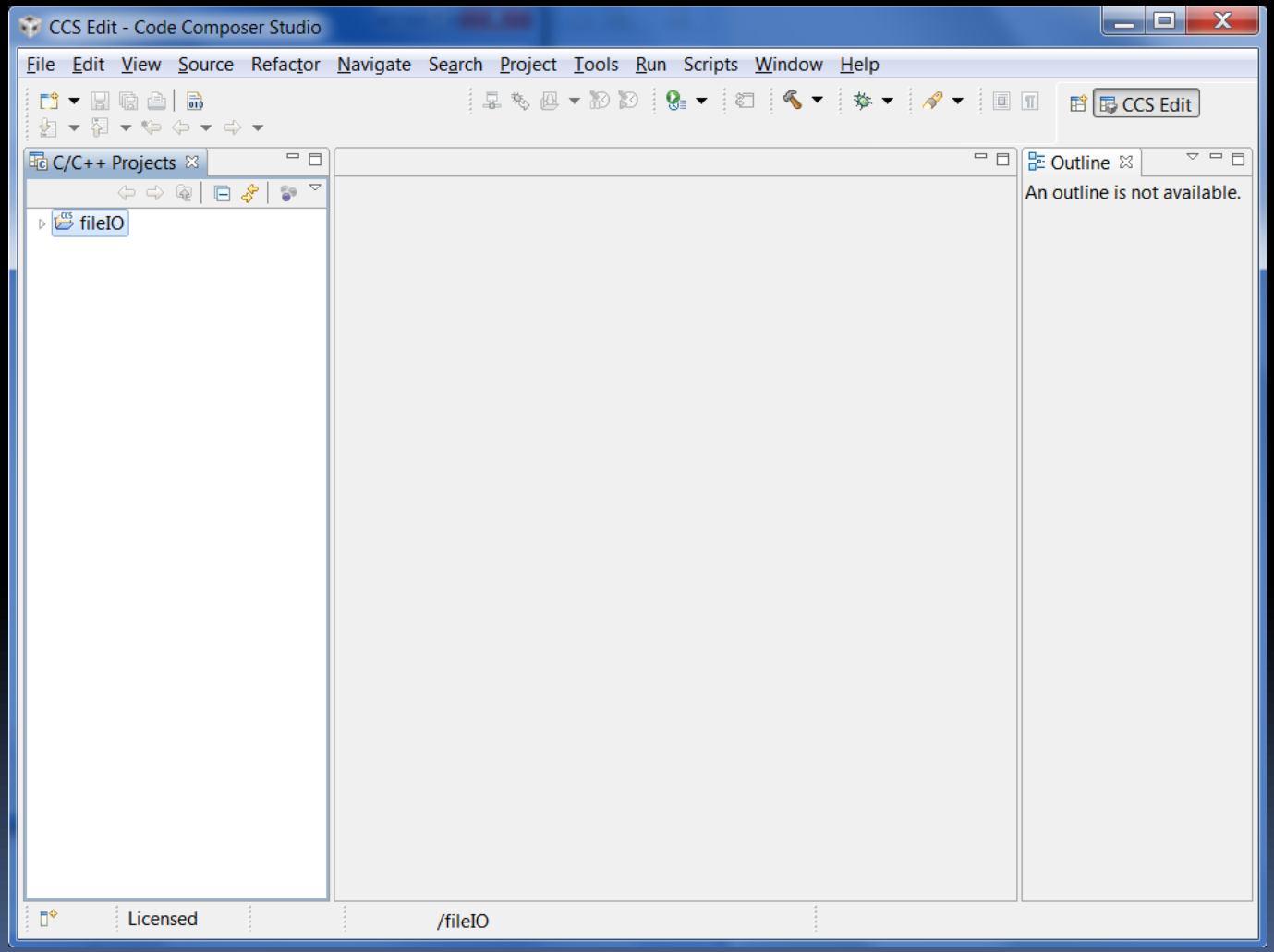


Set up the project

(Executable output, Device C5505, Library rts55x.lib)

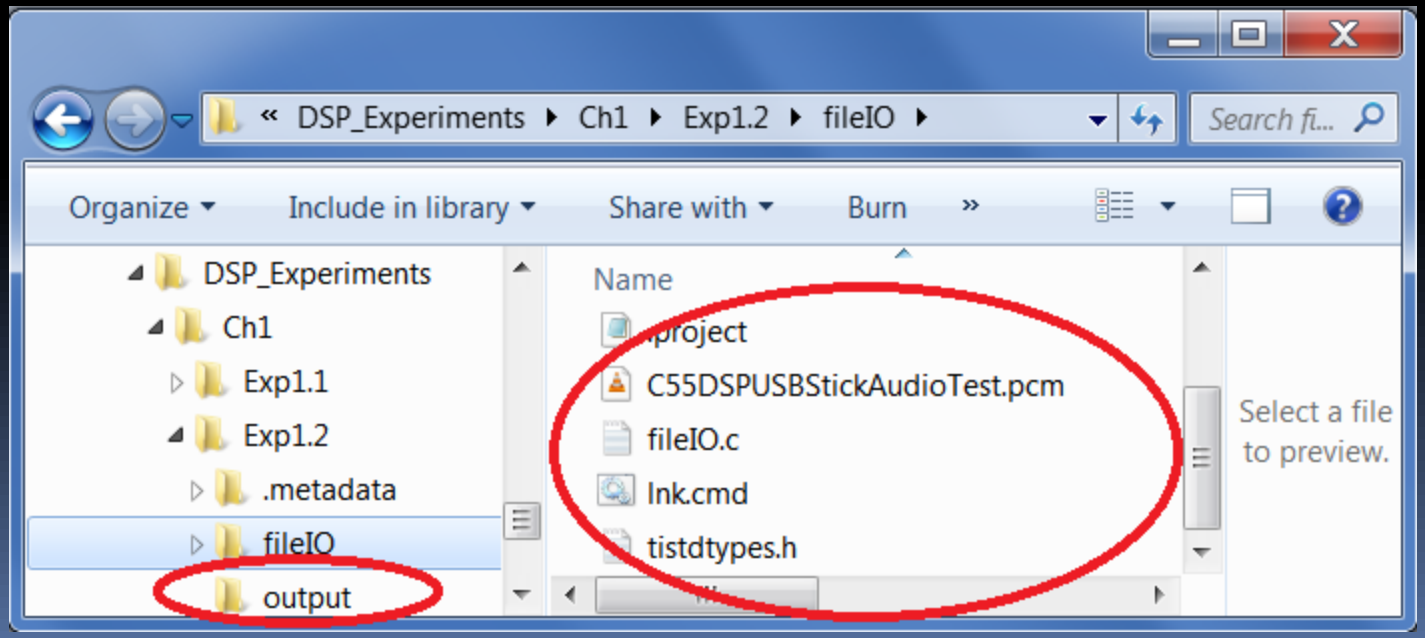


Project: fileIO



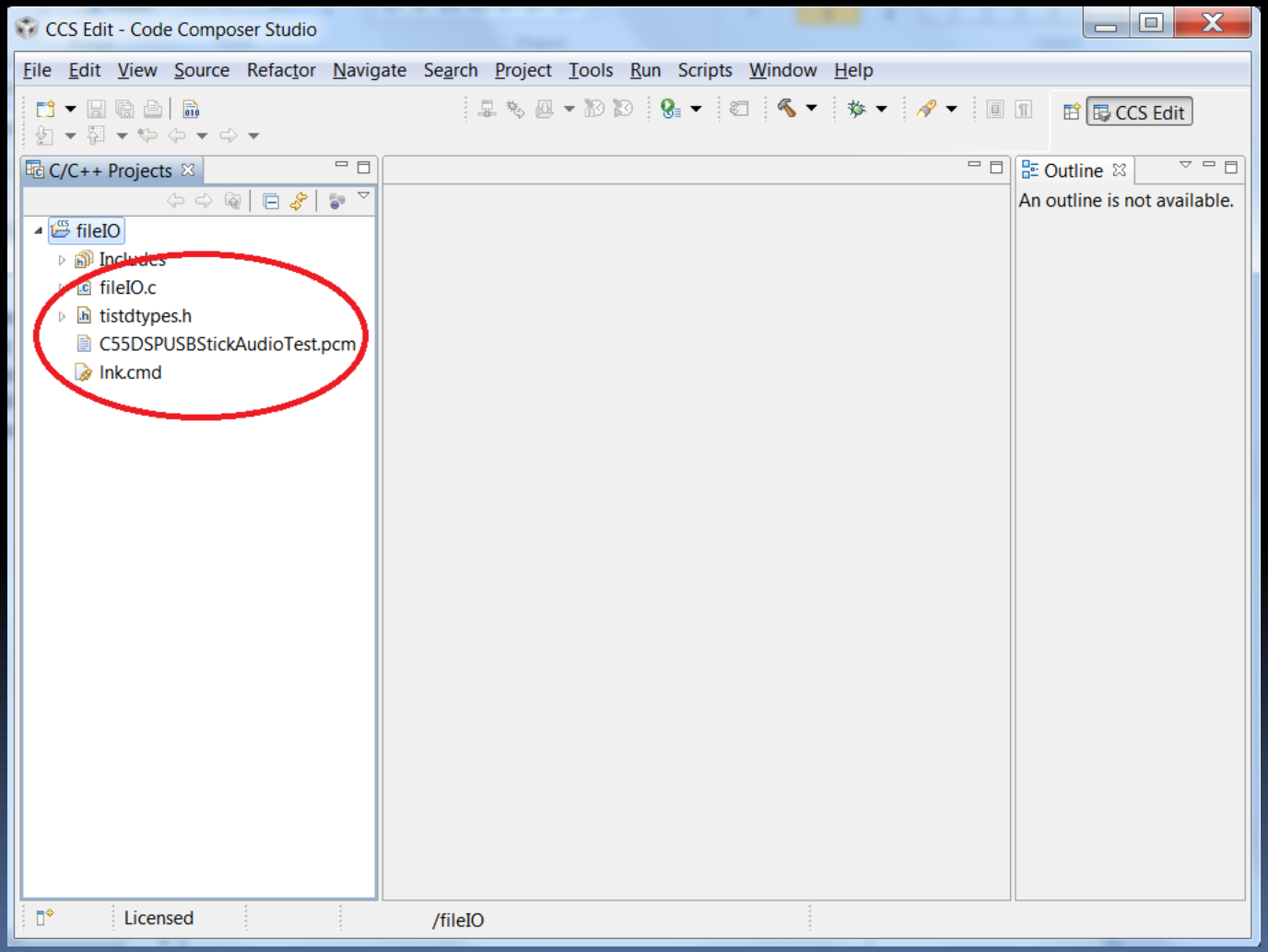
Copy files to workspace

- *Copy the following files cam with the book to workspace (folder fileIO)*
  - *lnk.cmd*
  - *fileIO.c*
  - *tistdtypes.h*
  - *C55DSPUSBStickAudioTest.pcm*
- Create a folder under Exp1.2 and name the folder "output"



CCS project

(The fileIO.c and lnk.cmd are included in the project)



Setup build environment

- *Right click on fileIO then select Property*
- *Select and expand C/C++ Build option*
- *Select Settings, then Runtime Options*
- *Set type size to 16 and memory model to large*

Set target configuration

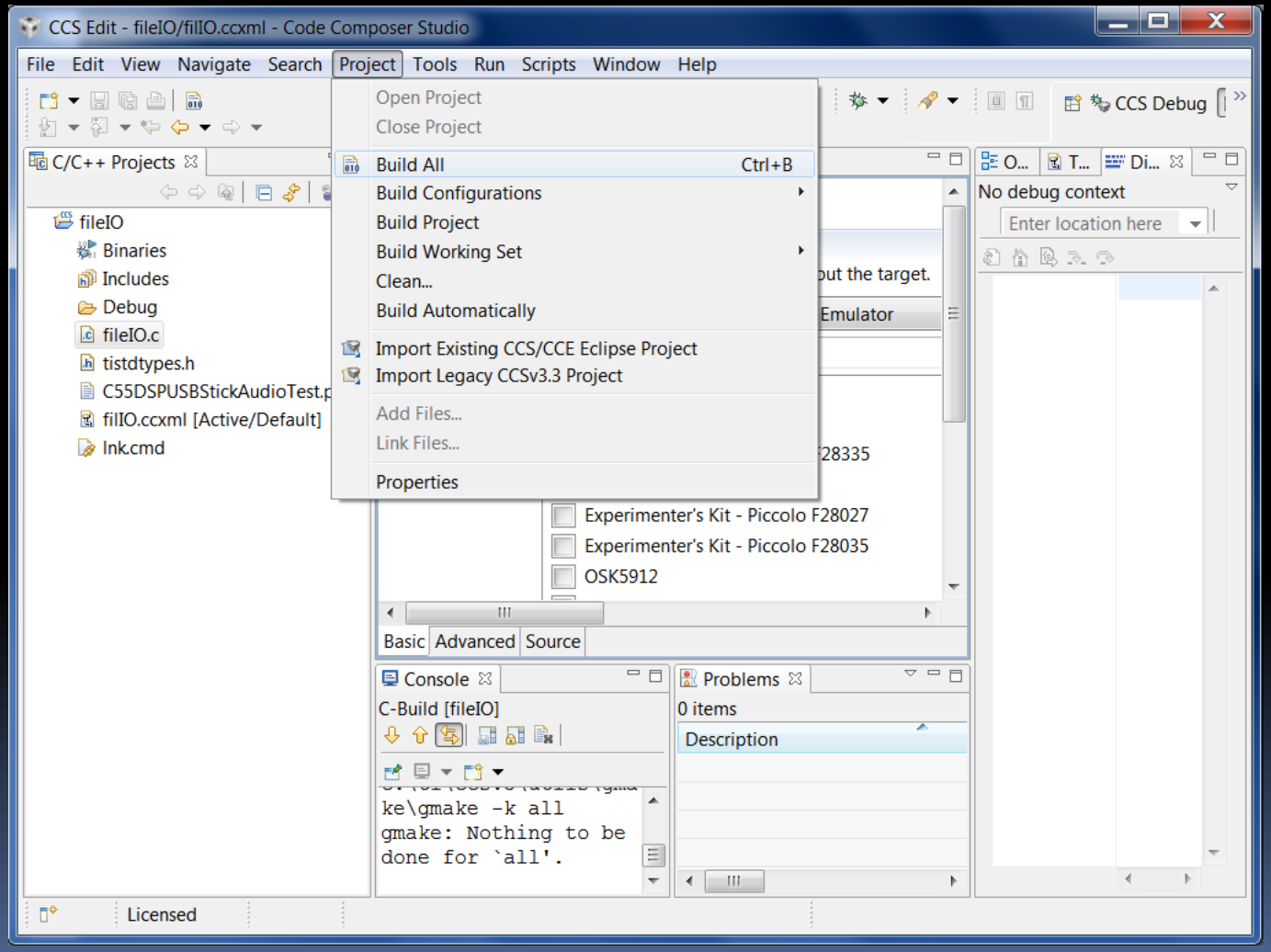
- *Right click on Project->New->Target Configuration File*
- *Create a target configuration file name, fileIO.ccxml*
- *Select Texas Instruments XDS100v2 USB Emulator*
- *Check the box for USBSTK5505*
- *Save the configuration*

Launch target

- *From Target Configuration window*
- *Open Project and right click on fileIO.ccxml*
- *Select Launch target configuration*
- *In Debug window, right click on Texas Instruments XDS100v2 USB Emulator_0/C55xx*
- *Select Connect Target to launch the target*
- *You shall see target reset and configured automatically*

Build the project

(Project->Build All)



Load program fileIO.out

- *From CCS Run->Load->Load Program*
- *Blows and select the executable file fileiO.out from C:\Users\DSP_Experiments\Ch1\Exp1.2\fileIO\fileIO.out*
- *Click Open and then OK to load the program*

Once the program is loaded

(Program counter is at entry point of function main())

CCS Debug - fileIO/fileIO.c - Code Composer Studio

File Edit View Source Refactor Navigate Search Project Tools Run Scripts Window Help

Debug

filIO.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C55xx (Susp
main() at fileIO.c:24 0x000234B1
_args_main() at args_main.c:25 0x000243F0 (_args_mai

Variables | Expressions

Name	Type	Value	Location
fp1	struct un...	0x5192B...	0x00001...
fp2	struct un...	0xF18A...	0x00001...
i	unsigne...	2393348...	0x00001...

Breakpoints

Ide...	Na...	Con...	Cou...	Acti...
	Brea...		0 (0)	Rem...

filIO.ccxml | fileIO.c | tistdtypes.h | 0xff6adc

```
23 void main()
24 {
25     FILE *fp1,*fp2;            // file pointers
26     Uint32 i;                  // unsigned long integer used as a co
27
28     printf("Exp. 1.2 --- file IO\n");
29
30     fp1 = fopen("..\\C55DSPUSBStickAudioTest.pcm", "rb");
31     fp2 = fopen(".\\..\\..\\output\\C55DSPUSBStickAudioTest.wav", "
32
33     if (fp1 == NULL)           // check if the input file exists
34     {
35         printf("Failed to open input file 'C55DSPUSBStickAudioTest.
36         exit(0);
37     }
38
39     fseek(fp2, 44, 0);         // advance output file point 44 bytes
40     i=0;
41     while (fread(ch, sizeof(Uint8), SIZE, fp1) == SIZE) // read SIZ
```

Target Co... | Console

C-Build [fileIO]

```
**** Build of
configuration Debug
for project fileIO
****

C:\ti\ccsv5\utils\gmak
e\gmake -k all
gmake: Nothing to be
done for `all'.
```

Licensed | Writable | Sma...ert

Use Resume Run the program

(CCS console displays experiment messages)

CCS Debug - fileIO/fileIO.c - Code Composer Studio

File Edit View Source Refactor Navigate Search Project Tools Run Scripts Window Help

Debug

filIO.ccxml [Code Composer Studio - Device Debugging]

Texas Instruments XDS100v2 USB Emulator_0/C55xx (Suspended)

abort() at exit.c:60 0x000245A2 (abort has only skeletal debug

filIO.ccxml | tistdtypes.h | fileIO.c | abort() at /sim/sds12/scratch/a0322878_rt...

```
23 void main()
24 {
25     FILE *fp1,*fp2;                          // file pointers
26     Uint32 i;                                // unsigned long
27
28     printf("Exp. 1.2 --- file IO\n");
29
30     fp1 = fopen("..\\C55DSPUSBStickAudioTest.pcm", "rb");
31     fp2 = fopen(".\\..\\..\\output\\C55DSPUSBStickAudioTest.wav",
32
33     if (fp1 == NULL)                         // check if the
34     {
35         printf("Failed to open input file 'C55DSPUSBStickAudioTe
36         exit(0);
37     }
38
39     fseek(fp2, 44, 0);                       // advance outpu
40     i=1;
```

Help | Console

filIO.ccxml:CIO

```
Exp. completed
Exp. 1.2 --- file IO
1024
2048
3072
4096
5120
6144
7168
8192
9216
10240
11264
12288
13312
14336
15360
16384
17408
18432
19456
20480
21504
22528
23552
24576
```

Licensed | Writable | Sma...ert

Verify the program result

- After running the experiment, your program shall create a WAV file named as "C55DSPUSBStickAudioTest.wav" in the output folder that you have created
- Play this WAV file using a player such as Windows Media Player. You shall hear the male voice saying: "C55 DSP USB Stick Audio Test".

New experiment assignments

- Write a C program that will
  - Read the input data file, C55DSPUSBStickAudioTest.pcm, as Exp1.2
  - Open a file to write the data in ASCII integer, the name of the output file can be "C55DSPUSBStickAudioTest.xls"

    hint: use *fprintf( )* function to replace *fwrite( )*
  - Convert every two -byte input data values into an integer (16-bit) number as:

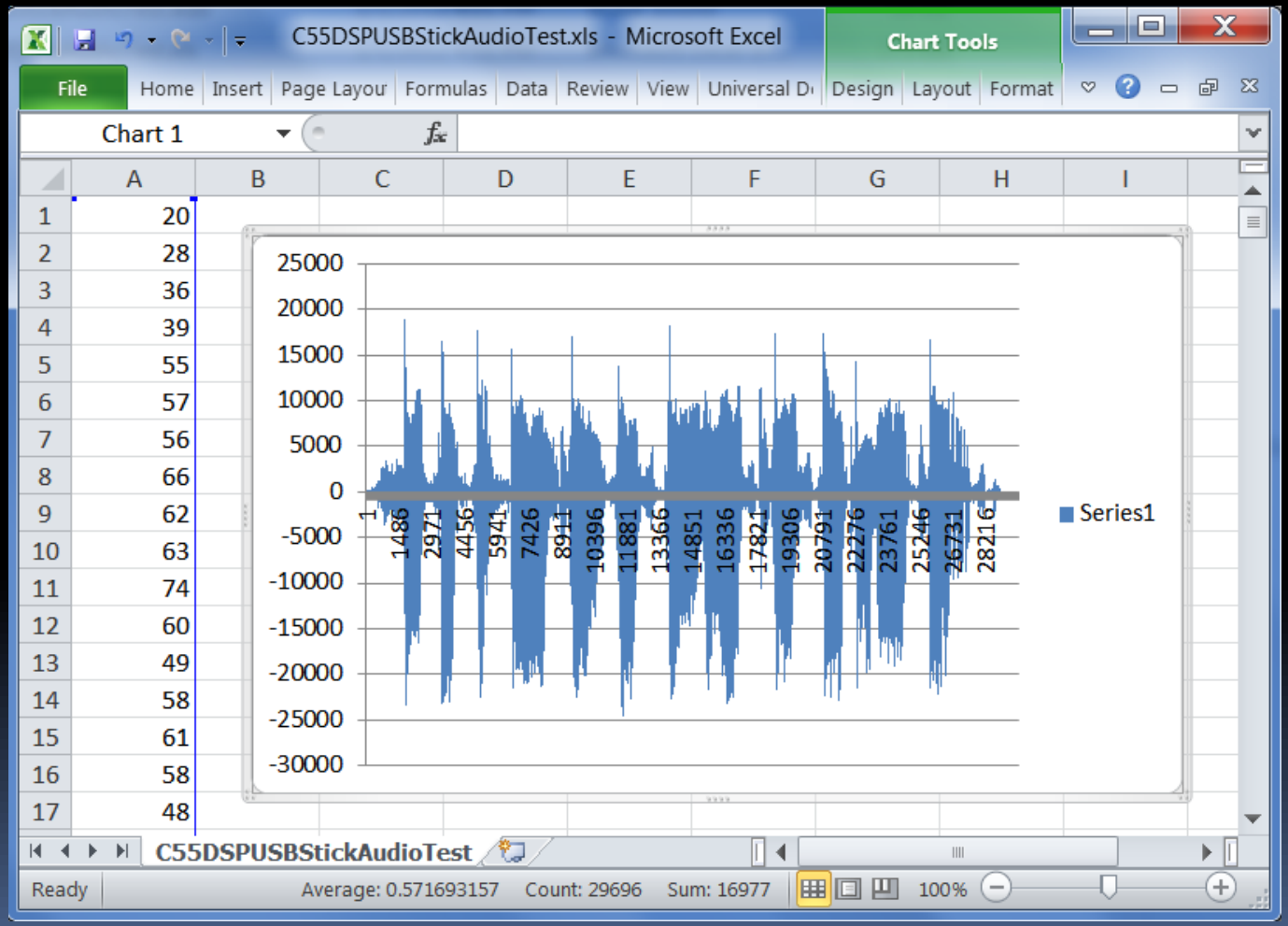
    o = ch[j] | (ch[j+1]<<8); // where j = 0, 2, ...

    and write input data in ASCII integer to output file
  - Build and run the program to generate your output file
  - using Microsoft Excel to open the output file, select the data column to plot the data to view the waveform of "C55DSPUSBStickAudioTest", see next page
- Write a C program that will
  - Read the input file "C55DSPUSBStickAudioTest.xls" created in above experiment and generated a WAV file output
  - Play the WAV file on a computer and listen to the audio.

    Q1: Does it sound the same as the WAV file output obtained from the experiment given in section 1.5.2 (Exp1.2)?

The new experiment result

(The waveform of the audio C55 DSP USB Stick Audio Test)



Programming quick review 1

- This experiment used a few more C file IO functions, such as *fopen( )*, *fread( )*, *fwrite ( )*, and *fclose( )*. These functions are defined in the header file stdio.h.
- The *fopen( )* and *fclose( )* are used to open and close the files used in the experiment. The *fread( )* and *fwrite( )* are used to read data and write data from and to the input and output devices.
- The *fopen( )* function has two arguments, the first argument is a character string for the name of the file and the second argument is also a character string telling the function how to use the file opened. The second argument "rb" and "wb" used in the fileIO experiment are for reading binary data in and writing binary data out.
- For any file opened, it must be properly closed after the operation. This can be done by the *fclose( )* function.
- The *fread( )* and *fwrite( )* functions have 4 arguments. The *fread* reads from stream (4th argument) into the array (1st argument), the number of data (3rd argument) with the size of (2nd argument). The *fwrite* writes data in the array (1st argument) to the stream (4th argument), the number of data (3rd argument) with the size of (2nd argument).
- The *fseek( )* and *rewind( )* functions are used to set the starting point of the files for writing the output data. These functions help manage the file access.

Programming quick review 2

- C language has many different data types.
- To improve portability of the programs written for one type of computer devices to another, we use a unique type define header file, tistdtypes.h, to specify the data type to avoid any ambiguity.
- The following data types are used in this experiment
  - Uint8 (unsigned 8-bit integer data, unsigned char)
  - Uint32 (unsigned 32-bit integer data, unsigned long)
  - FILE * (file pointer)
- Special notice: *fread( )* and *fwrite( )* functions in CCS these functions will only read or write byte size (8-bit) binary data during memory access. When the *fread( ) function* reads a 16-bit data into a data memory location, for example 0x1234, it reads 0x34 first then 0x12, and place them into memory ch[0]=0x34 and ch[1]=0x12. When *fwrite( ) function* writes data 0x1234. It only writes 0x34 but not write 0x12. So it is important to convert each 16-bit data into two 8-bit data and read or write two bytes for experiment.

References

- C Programming Language (2nd Edition), by Brian Kernighan and Dennis Ritchie, Prentice Hall