



User Interface for eZdsp



EXPERIMENT 1.3

Propose of the experiment

- Continue from previous experiments to be familiar with CCS environment
- Write a user interface (UI) program in C to interact with the program from CCS Console Window
- The UI functions will be used in future experiments for runtime control of the experimental parameters
- Understand C functions that used for UI interface, condition (*if-else*), and flow control (*switch*) statement
- Familiar with CCS tools (Breakpoint, Memory Watch Window, Variable Watch Window, and Graph plotting)

Experiment preparation

- Start CCS
- Create workspace Exp1.3
- Create a new project and name it as userInterface
- Copy experiment files to workspace folder

Setup build environment

- Right click on project "UI" 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, *userInterface.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 *userInterface.ccxml*
- Select Launch target configuration
- In Debug window, right click on Texas Instruments XDS100v2 USB Emulator_o/C55xx
- Select Connect Target to launch the target
- You shall see target reset and configured automatically



Build and run the program

- Build the project (user Build All)
 - Load the program
 - Run the program
 - Move mouse cursor to CCS Console Window, key in the proper numbers when UI program asks
- 

Verify the program result

- Gain parameter initialization
 - The program accepts an initial gain from -6 to 29
 - Any value outside above range will be replaced with the default 0dB gain
- Sampling Frequency initialization
 - The program requests an input from one of the numbers listed: 8KHz, 12KHz, 16KHz, 24KHz, or 48KHz
 - Any other value given will be invalid if a invalid variable is entered, the Sampling Frequency will be set to default at 48KHz
- Playtime initialization
 - The program will set up the play duration, the range is between 5 seconds and 60 seconds. If the playing time is entered outside above range, the program uses 10 seconds as default
- The Gain, Sampling Frequency and Playtime will be displayed on the CCS Console
- Once the program completed, an array of data will be filled based on the Sampling Frequency

Program Console Interaction

(Example: Gain=3dB, Sampling Freq.=24000Hz, Playtime=20s)

The screenshot displays the Code Composer Studio (CCS) interface during a debug session. The main window is titled "CCS Debug - UI/UI.c - Code Composer Studio". The "Console" window, located at the bottom right, is circled in red and shows the following interaction:

```
UI.c:CIO
Exp. 1.3 --- UI
Enter an integer number for Gain between (-6 and 29)
3
Enter the Sampling Frequency, select one: 8000, 9600
24000
Enter the playing time duration (5 to 60)
20
Gain is set to 3dB
Sampling Frequency is set to 24000Hz
Playing time is set to 20s
```

The "Source" window on the left shows the C code for the program, including headers, a size definition, and a main function that uses a pre-generated sine wave table.

```
8 #include <stdio.h>
9 #include "tistdtypes.h"
10
11 #define SIZE 48
12 Int16 dataTable[SIZE];
13
14 void main()
15 {
16     /* Pre-generated sine wave data,
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30
19         0x6ed8, 0x763f, 0x7ba1, 0x7e
20         0x6ed8, 0x658b, 0x5a81, 0x4d
21         0x0000, 0xef4c, 0xdee0, 0xcf
22         0x9128, 0x89c1, 0x845f, 0x81
23         0x9128, 0x9a76, 0xa57f, 0xb2
24     };
25
26     Int16 a.n.i.i.k.n.m;
```

CCS Breakpoint

(Example: double click the line to set / remove breakpoint)

The screenshot displays the Code Composer Studio (CCS) interface during a debug session. The main window shows the source code for `UI.c` at `/sim/sds12/scratc...`. A breakpoint is set on line 108, which contains the `printf` statement. The Variables window shows the current values of variables `g`, `i`, `j`, `k`, `m`, `n`, `p`, and `sf`. The Console window shows the output of the program, including the prompt `Exp. 1.3 --- UI` and the message `\nExp. completed\n`.

Name	Type	Value
g	short	-32657
i	short	27786
j	short	2888
k	short	-2181
m	short	-15259
n	short	17392
p	short	19708
sf	unsigned long	237815678

```
95     m = 1;
96         break;
97     }
98
99     for (n=k=0, i=0; i<m; i++)    // i
100     {
101         for (j=k; j<SIZE; j+=m)
102         {
103             dataTable[n++] = table[j];
104         }
105         k++;
106     }
107
108     printf("\nExp. completed\n");
109 }
110
```

UI.ccxml:CIO
Exp. 1.3 --- UI
Enter an integer number for Gain between (-6 and 3)
3
Enter the Sampling Frequency, select one: 8000, 9
24000
Enter the playing time duration (5 to 60)
20
Gain is set to 3dB
Sampling Frequency is set to 24000Hz
Playing time is set to 20s

Set up Variable Watch Window

(Example: View->Variables to open variable viewing window)

The screenshot shows the Code Composer Studio (CCS) interface. The 'View' menu is open, and the 'Variables' option is highlighted. The main window displays a C program with a sine wave table and a main function. The console window shows the program's output, including prompts for gain, sampling frequency, and playing time.

```
8 #include <math.h>
9 #include <stdint.h>
11 #define SIZE 48
12 Int16 dataTable[SIZE];
13
14 void main()
15 {
16     /* Pre-generated sine wave data,
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30
19         0x6ed8, 0x763f, 0x7ba1, 0x7e
20         0x6ed8, 0x658b, 0x5a81, 0x4d
21         0x0000, 0xef4c, 0xdee0, 0xcf
22         0x9128, 0x89c1, 0x845f, 0x81
23         0x9128, 0x9a76, 0xa57f, 0xb2
24     };
25
26     Int16 a.p.i.i.k.n.m;
```

Console Output:

```
ULcxml:CIO
Exp. 1.3 --- UI
Enter an integer number for Gain between (-6 and 29)
3
Enter the Sampling Frequency, select one: 8000, 9600
24000
Enter the playing time duration (5 to 60)
20
Gain is set to 3dB
Sampling Frequency is set to 24000Hz
Playing time is set to 20s
```

CCS Variable Watch Window

(Monitoring variables for debugging)

The screenshot displays the CCS Debug interface. The Variable Watch Window is circled in red and contains the following data:

Name	Type	Value
g	short	3
i	short	2
j	short	49
k	short	2
m	short	2
n	short	48
p	short	20
sf	unsigned long	24000

The main() function code is also visible, with the variable declarations circled in red:

```
13  
14 void main()  
15 {  
16     /* Pre-generated sine wave data,  
17     Int16 table[SIZE] = {  
18         0x0000, 0x10b4, 0x2120, 0x30  
19         0x6ed8, 0x763f, 0x7ba1, 0x7e  
20         0x6ed8, 0x658b, 0x5a81, 0x4d  
21         0x0000, 0xef4c, 0xdee0, 0xcf  
22         0x9128, 0x89c1, 0x845f, 0x81  
23         0x9128, 0x9a76, 0xa57f, 0xb2  
24     };  
25  
26     Int16 g,p,i,j,k,n,m;  
27     Uint32 sf;
```

The Console window shows the following output:

```
ULccxml:CIO  
Exp. 1.3 --- UI  
Enter an integer number for Gain between (-6 and 29)  
3  
Enter the Sampling Frequency, select one: 8000, 9600  
24000  
Enter the playing time duration (5 to 60)  
20  
Gain is set to 3dB  
Sampling Frequency is set to 24000Hz  
Playing time is set to 20s
```

Set up Memory Watch Window

(Example: View->Memory Browser to open memory watch window)

The screenshot shows the Code Composer Studio (CCS) interface. The 'View' menu is open, and 'Memory Browser' is highlighted with a red circle. The 'Variables' window shows a table of variables, and the 'Console' window shows the execution output of a program.

Name	Type	Value
g	short	3
i	short	2
j	short	49
k	short	2
m	short	2
n	short	48
p	short	20
sf	unsigned long	24000

```
13
14 void main()
15 {
16     /* Pre-generated sine wave data,
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30
19         0x6ed8, 0x763f, 0x7ba1, 0x7e
20         0x6ed8, 0x658b, 0x5a81, 0x4d
21         0x0000, 0xef4c, 0xdee0, 0xcf
22         0x9128, 0x89c1, 0x845f, 0x81
23         0x9128, 0x9a76, 0xa57f, 0xb2
24     };
25
26     Int16 g,p,i,j,k,n,m;
27     UInt32 sf;
```

Console Output:

```
UI.ccxm1:CIO
Exp. 1.3 --- UI
Enter an integer number for Gain between (-6 and 29)
3
Enter the Sampling Frequency, select one: 8000, 9600
24000
Enter the playing time duration (5 to 60)
20
Gain is set to 3dB
Sampling Frequency is set to 24000Hz
Playing time is set to 20s
```

CCS Memory Watch Window

(Verifying data memory for debugging)

The screenshot displays the Code Composer Studio interface during a debug session. The main window is titled "CCS Debug - UI/Ui.c - Code Composer Studio". The "Memory Watch" window is open, showing a table of variables and their values:

Name	Type	Value
g	short	3
i	short	2
j	short	49
k	short	2
m	short	2
n	short	48
p	short	20
sf	unsigned long	24000

The source code window shows the following code:

```
13
14 void main()
15 {
16     /* Pre-generated sine wave data, 16
17     int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30fb,
19         0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
20         0x6ed8, 0x658b, 0x5a81, 0x4dea,
21         0x0000, 0xef4c, 0xdee0, 0xcf06,
22         0x9128, 0x89c1, 0x845f, 0x811b,
23         0x9128, 0x9a76, 0xa57f, 0xb216,
24     };
25
26     Int16 g,p,i,j,k,n,m;
27     Uint32 sf;
```

The "Memory Browser" window is also open, showing the memory address 0x01022 and its contents in hex 16-bit TI style:

Hex 16 Bit - TI Style Hex
0x001022 0000 10B4 2120 30FB 3FFF 4DEA 5A81 658B
0x00102A 6ED8 763F 7BA1 7EE5 7FFD 7EE5 7BA1 76EF
0x001032 6ED8 658B 5A81 4DEA 3FFF 30FB 2120 10B4
0x00103A 0000 EF4C DEE0 CF06 C002 B216 A57F 9A75
0x001042 9128 89C1 845F 811B 8002 811B 845F 89C1
0x00104A 9128 9A76 A57F B216 C002 CF06 DEE0 EF4C
0x001052 0003 0014 0002 0031 0002 0030 0002 60C2
0x00105A 0000 5DC0 0010 6149 0008 FF00
0x001060 <u>STACK_END</u> , <u>_sysstack</u>
0x001060 268C 5592 BC59 7582 DCF0 82A7 89F5 7D80
0x001068 8441 550A 1540 1580 0041 DA8F 8DC1 1782
0x001070 0DC1 D3D2 1D41 DC21 9841 E1B4 85C0 5AE4

Set up Graph Tool for Plot

(Example: tools->Graph->Single Time)

The screenshot shows the CCS Debug - UI/UC - Code Composer Studio interface. The 'Tools' menu is open, and the 'Graph' option is selected, which has opened a sub-menu where 'Single Time' is highlighted. Red circles are drawn around the 'Tools' menu, the 'Graph' option, and the 'Single Time' option.

The code editor displays the following C code:

```
10
11 #define SIZE 48
12 Int16 dataTable[SIZE];
13
14 void main()
15 {
16     /* Pre-generated sine wave data, 16-
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30fb,
19         0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
20         0x6ed8, 0x658b, 0x5a81, 0x4dea,
21         0x0000, 0xef4c, 0xdee0, 0xcf06,
22         0x9128, 0x89c1, 0x845f, 0x811b,
23         0x9128, 0x9a76, 0xa57f, 0xb216,
24     };
25
```

The memory map shows the following values:

Value
3
2
49
2
2
48

The hex dump shows the following data:

Hex 16 Bit - TI Style Hex
0x001022 0000 10B4 2120 30FB 3FFF 4DEA 5A81 658B
0x00102A 6ED8 763F 7BA1 7EE5 7FFD 7EE5 7BA1 76EF
0x001032 6ED8 658B 5A81 4DEA 3FFF 30FB 2120 10B4
0x00103A 0000 EF4C DEE0 CF06 C002 B216 A57F 9A75
0x001042 9128 89C1 845F 811B 8002 811B 845F 89C1
0x00104A 9128 9A76 A57F B216 C002 CF06 DEE0 EF4C
0x001052 0003 0014 0002 0031 0002 0030 0002 60CF
0x00105A 0000 5DC0 0010 6149 0008 FF00
0x001060 <u>STACK_END</u> , <u>sysstack</u>
0x001060 D68C 5592 BC59 7582 DCF0 8EA7 89F5 7D80
0x001068 8441 550A 1540 1580 0041 DA8F 8DC1 1782
0x001070 0DC1 D3D2 1D41 DC21 9841 E1B4 85C0 5AE4

CCS Graph Parameter for Plot

(Example: Parameters to plot dataTable[])

The screenshot shows the Code Composer Studio (CCS) interface. The main window displays a C code snippet for plotting data. The code defines a constant `SIZE` as 48 and declares an array `dataTable` of type `Int16`. The `main` function contains a pre-generated sine wave data array.

```
10
11 #define SIZE 48
12 Int16 dataTable[SIZE] = {
13
14 void main()
15 {
16     /* Pre-generated sine wave data, 16-
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30fb,
19         0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
20         0x6ed8, 0x658b, 0x5a81, 0x4dea,
21         0x0000, 0xef4c, 0xdee0, 0xcf06,
22         0x9128, 0x89c1, 0x845f, 0x811b,
23         0x9128, 0x9a76, 0xa57f, 0xb216,
24     };
25
```

The `Graph Properties` dialog box is open, showing the configuration for plotting the `dataTable` array. The `Start Address` is set to `dataTable`. The `Display Data Size` is set to 48. The `Dsp Data Type` is set to 16 bit signed integer. The `OK` button is highlighted.

Property	Value
Data Properties	
Acquisition Buffer Size	548
Dsp Data Type	16 bit signed integer
Index Increment	1
Q_Value	0
Sampling Rate HZ	1
Start Address	dataTable
Display Properties	
Axis Display	<input checked="" type="checkbox"/> true
Data Plot Style	Line
Display Data Size	48
Grid Style	Major Grid
Magnitude Display	Linear
Time Display Unit	sample
Misc	
Use Dc Value For Gain	<input type="checkbox"/> false

CCS Graph Plot

(Example: `dataTable[]` has been plotted after execution)

The screenshot displays the Code Composer Studio (CCS) interface during a debug session. The main window shows the source code for `UI.c` at address `0xff6adc`. The code defines a constant `SIZE` as 40 and declares an array `dataTable` of type `Int16`. The `main` function contains a comment indicating that pre-generated sine wave data is stored in the `table` array.

```
10
11 #define SIZE 40
12 Int16 dataTable[SIZE];
13
14 void main()
15 {
16     /* Pre-generated sine wave data, 16-
17     Int16 table[SIZE] = {
18         0x0000, 0x10b4, 0x2120, 0x30fb,
19         0x6ed8, 0x763f, 0x7ba1, 0x7ee5,
20         0x6ed8, 0x658b, 0x5a81, 0x4dea,
21         0x0000, 0xef4c, 0xdee0, 0xcf06,
22         0x9128, 0x89c1, 0x845f, 0x811b,
23         0x9128, 0x9a76, 0xa57f, 0xb216,
24     };
25
```

The `Variables` window shows the current state of the program's variables:

Name	Type	Value
g	short	3
i	short	2
j	short	49
k	short	2
m	short	2
n	short	48
p	short	20
sf	unsigned long	24000

The `Single Time` window displays a graph of the sine wave data. The x-axis is labeled `sample` and ranges from 0 to 40. The y-axis ranges from -36000 to 34000. The graph shows a periodic sine wave with two full cycles visible.

New experiment assignments

- **CCS Breakpoint**
 - Build and load Exp1.3 program, set up a breakpoint on the last line of the program, and run the program (F8), what happens?
 - Reload program, set a new breakpoint in the middle of the program, run the program again. What happens? Resume the program (F8), what happens?
- **CCS Data and Variable Watch Windows**
 - Set up variable watch window, as Exp1.3. What other data types the watch window can supports? Change the data type and observe the watch window display.
 - How can you find out what memory address to use for setting up the watch windows?
 - Setup both data and variable watch windows, single step through the program and watch how the data and variables being updated.
 - Do you know you can modify data memory via CCS by directly editing the memory location?
- **CCS Graph Tool**
 - Restart the Exp1.3 program and check what graphs CCS can plot. How do you select the Graph parameters, why? Plot the data[] in the same window.
 - Add x-y axis labels to your plot. Add a grid to the graph plot.
 - Change the display from line to large square.
- **CCS Programming**
 - Replace the *switch ()* statement in Exp1.3 with *if-else* statement and re-run the experiment. That is, the program uses only the *if-else* statement. If you see incorrect results, using breakpoint and watch window tools to debug your program and fix the error.
 - Replace the *if-else* statement in Exp1.3 with *switch ()* statement and re-run the experiment. That is, the program uses only the *switch ()* statement.
- **CCS User Specified File Name**
 - Write a program that will read character string from CCS console.
 - Modify Exp 1.2 such that you can tell the program the file name of the output WAV file through the CCS console.

Programming quick review

- The *scanf()* function is used to accept user keyboard (standard input device) inputs. The basic data type the *scanf()* can accept includes "%c", "%s", "%i", "%d", "%x", "%u", "%f", for character, stream, integer, decimal integer, hexadecimal, unsigned decimal integer, and floating-point data.
- This experiment used C conditional statement *if-else* to check the parameters entered by user. Different actions will be taken depend upon the condition is true or false.
- To control the program flow, the *switch()* statement is used.



References

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