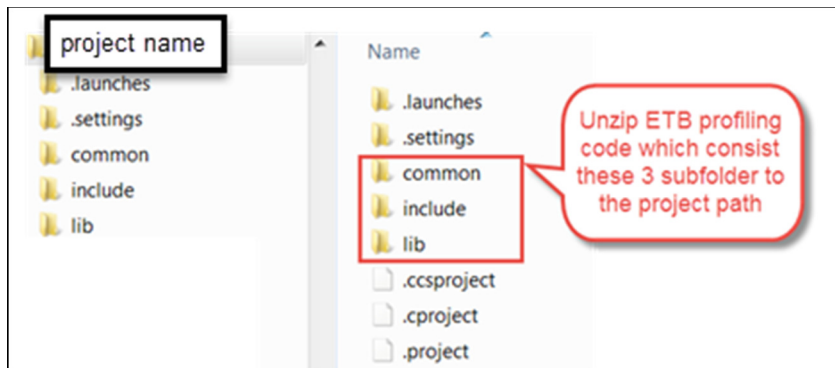


Using ETB Trace Code for Runtime Profiling

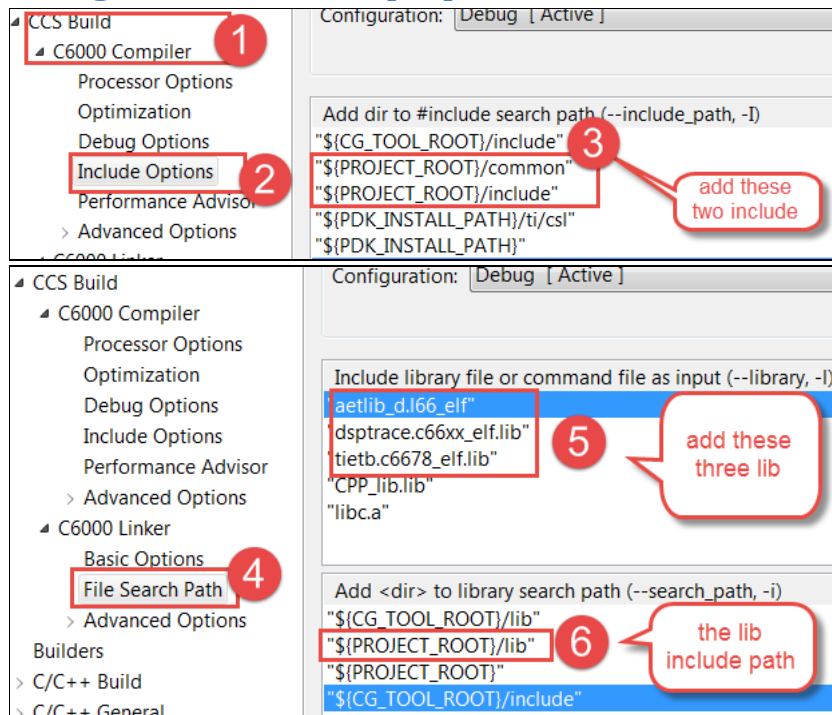
建立工程

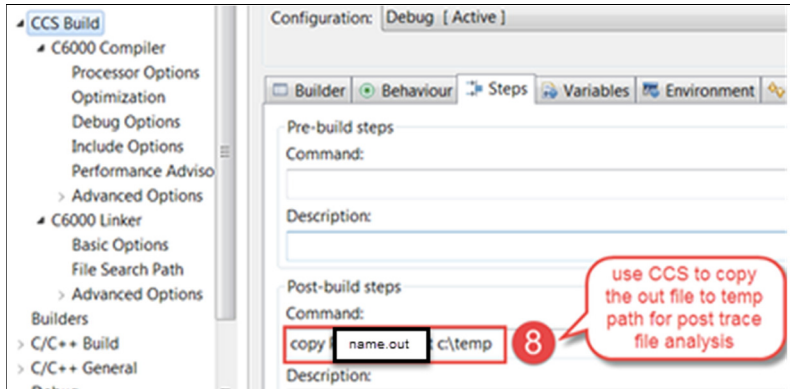
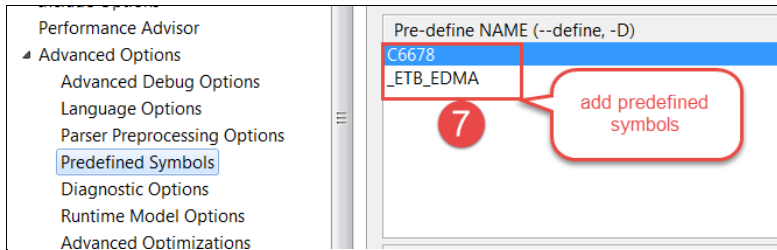
Unzip the ETB code to the project path.

The ETB profiling code consists of 3 sub folder: Common/include/lib



Configure related build properties.





集成代码

Declare the header file and global variable

- 1 Include "ETB_Profile.h"
- 2 The pDmaMemory point to the trace data buffer
- 3 The pETBHandle and pDSPHandle used for trace job setup and release

```
#include "ETB_Profile.h"

uint32_t *pDmaMemory = (uint32_t *)EDMA_DEST_ADDRESS;
ETBHandle* pETBHandle;
DSPTraceHandle* pDSPHandle;
```

Call the API to set up the trace job

```
TraceJOBConfig tProfileConfig;

Init_Profile(pDmaMemory, &tProfileConfig);
tProfileConfig.tracetype = AET_TRACE_TIMING | AET_TRACE_PA;
tProfileConfig.startProfileAddr = (uint32_t) &runFunction;
#if !USE_TEND
tProfileConfig.endProfileAddr = (uint32_t) &doneFunction;
#endif

Setup_ETB_Profile(&tProfileConfig, &pETBHandle, &pDSPHandle);
```

change the
address
based on
your
application

Release trace job once it is done

```
Stop_ETB_Profile(&tProfileConfig,&pETBHandle,&pDSPHandle);
```

Change below trace buffer destination and length if needed.

These macro are defined on “ETB_Profile.h”

```
#define EDMA_DEST_ADDRESS 0x0c000000
#define EDMA_BFR_WORDS    0x1000 /* 0x4000 (16384) bytes */

#define TEMP_BFR_READ_SIZE 0x400
```

Change the trace file path if needed.

The related codes are on file “ETB_Profile_Func.c”

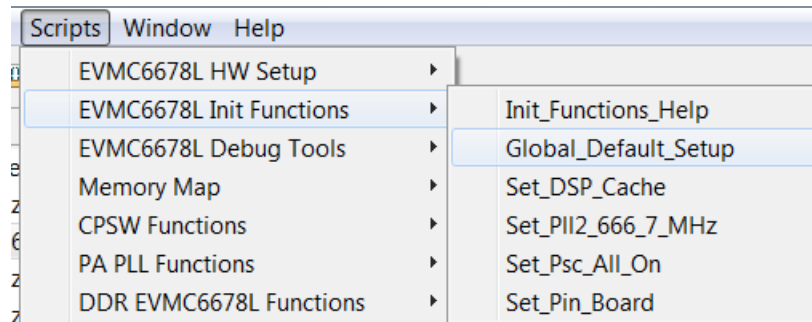
```
char * pFileName = "C:\\temp\\DSP_etbdata.bin";
```

运行程序

DSP 初始化

“evmc6678l.gel” is needed to initiate PLL/DDR/PSC if the application code does not cover it.

- 1 Use Menu “tools->gel files -> load GEL” to load the evmc6678l.gel
- 2 Initiate PLL and DDR by Menu “Scripts->EVMC6678L Init Functions -> Global_Default_Setup”



- 3 Execute Scripts Menu “Set_Psc_All_On” which will enable trace PSC.
- 4 Load the *.out file and get the trace file data.

数据分析

- 1 Execute the “convert_trace_file.bat” file which is packed with the demo code.
The successful execution output is as below:

```
c:\temp>convert_trace_file.bat

c:\temp>C:\ti\ccsv5\ccs_base\emulation\analysis\bin\bin2tdf -bin C:/temp/DSP_etb
data.bin -app C:/temp/profile_with_ETB.out -procid 66x -sirev 1 -rcvr ETB -output
t C:/temp/myTrace.tdf -dcmfile c:/temp/dcm.txt

bin to TDF conversion successful - no errors.
```

- 2 Use the CCS tools to analysis the trace data.
 - a. Open *.tcf and *.out file and get trace data view.

The screenshot shows the CCS Tools menu with the following items: Memory Map, GEL Files, Debugger Options, Pin Connect, Port Connect, Save Memory, Load Memory, Fill Memory, RTOS Object View (ROV), RTOS Analyzer, System Analyzer, Hardware Trace Analyzer, Graph, Image Analyzer, Profile, XDAIS Tools, and RTSC Tools. The Hardware Trace Analyzer sub-menu is open, showing options like Code Coverage, Function Profiling, Statistical Function Profiling, Stall Profiling, Cache Analysis, Memory Throughput and Access Analysis, PC Trace, Custom Core Trace, Custom System Trace, Open File, Analysis Dashboard, and Import Configuration... The Open File option is highlighted with a red circle and the number 3.

The Trace Viewer window shows a table of trace data with columns: Program Address, Code, Delta Cycles, Cycle, Trace S..., Function, Line Number, Source, and Disassembly. The data is organized into rows, with some rows grouped together. The table shows various instructions and their corresponding cycle counts and addresses.

Program Address	Code	Delta Cycles	Cycle	Trace S...	Function	Line Number	Source	Disassembly
0x80DA8A	0x2640	1	1581	startFunction	110			ADD.L1 A4.1,A4
0x80DA8C	0x20D0274	1	1582	startFunction	110			STW.D1T1 A4,*A3(8)
0x80DA90	0x7BD005A	1	1583	startFunction	113			ADD.L2 8.B15,B15
0x80DA94	0x8CA362	6	1584	startFunction	113			BNOP.S2 B3.5
0x80DADA	0x102ADA	5	1590	appFunction	122		while(dummy2 > 1)	LDW.D2T2 *B15[2],B4
0x80DAE0	0x102ADA	1	1595	appFunction	122			CMPLT.L2 1.B4,B0
0x80DAE4	0x2FECA120	6	1596	appFunction	122			[B0] BNOP.S1 0x80dab8.5
0x80DAB8	0xDC4D	5	1602	appFunction	124		dummy2--;	LDW.D2T2 *B15[2],B4
0x80DABA	0xEE41	1	1607	appFunction	124			ADD.L2 B4-1,B4
0x80DAC0	0xDC45	1	1608	appFunction	124			STW.D2T2 B4,*B15[2]
0x80DAC2	0xDC4D	5	1609	appFunction	125		if(dummy2 & 1)	LDW.D2T2 *B15[2],B4
0x80DAC4	0x102F5A	1	1614	appFunction	125			AND.L2 1.B4,B0
0x80DAC8	0x3007A120	6	1615	appFunction	125			[B0] BNOP.S1 0x80dace.5
0x80DACE	0xDC4D	5	1621	appFunction	129		if(dummy2 & 0x4)	LDW.D2T2 *B15[2],B4
0x80DAD0	0x108F5A	1	1626	appFunction	129			AND.L2 4.B4,B0
0x80DAD4	0x300DA120	6	1627	appFunction	129			[B0] BNOP.S1 0x80dada.5
0x80DAD8	0xF95B	6	1633	appFunction	131		startFunction();	CALLP.S2 0x80da54.B3
0x80DA54	0x7BF005A	1	1639	startFunction	104			SUB.L2 B15.0x8.B15
0x80DA58	0x627	1	1640	startFunction	105		volatile int dummy1 ...	MVK.L2 0.B4
0x80DA5A	0x8C45	1	1641	startFunction	105			STW.D2T2 B4,*B15[1]
0x80DA60	0xDC45	1	1642	startFunction	105			STW.D2T2 B4,*B15[2]
0x80DA62	0x8C4D	1	1643	startFunction	106		dummy1++;	LDW.D2T2 *B15[1],B4
0x80DA64	0x6C6E	4	1644	startFunction	106			NOP 4

- b. Use right-click menu to analyze the data, for example get function profile.

The screenshot shows the Trace Viewer window with a right-click menu open. The menu options include: Column Settings..., Copy (Ctrl+C), Freeze Update (Shift+F5), Data, Enable Grouping (Shift+G), Groups, Insert a Bookmark, Trace Viewer, and Analyze. The Analyze option is highlighted.

The table in the background shows the same trace data as the previous screenshot, with columns: Program Address, Code, Disassembly, Delta Cycles, Cycle, Source, and Function Start.

Program Address	Code	Disassembly	Delta Cycles	Cycle	Source	Function Start
0x8067F4	0x300DA1...	[B0] BNOP.S1 0x8067fa.5	6	370950		0
0x8067FA	0xDC4D	LDW.D2T2 *B15[2],B4	5	370956	<Sou...	0
0x806800	0x102ADA	CMPLT.L2 1.B4,B0	1	370961		
0x806804	0x2FECA1...	[B0] BNOP.S1 0x8067d8.5	6	370962		
0x806808	0x1BC92E6	LDW.D2T2 *++B15[4]...	5	370968	<Sou...	
0x80680C	0x8CA362	BNOP.S2 B3.5	5	370973		
0x80681A	0x8CCD	LDW.D2T2 *B15[5],B4	5	370979	<Sou...	
0x806820	0x2641	ADD.L2 B4-1,B4	1	370984		
0x806822	0x8CC5	STW.D2T2 B4,*B15[5]	1	370985		
0x806824	0xED3	MVK.S2 200,B5	1	370986		
0x806826	0x8E89	CMPLT.L2 B4,B5,B0	1	370987		
0x806828	0x2FFCA1...	[B0] BNOP.S1 0x806b18.5	6	370988		
0x80682C	0xD19B	CALLP.S2 0x806838.B3	6	370994	<Sou...	
0x806838	0x8CA362	BNOP.S2 B3.5	6	371000	<Sou...	1
0x80682E	0x627	MVK.L2 0.B4	1	371006	<Sou...	0
0x806830	0xFEC5	STW.D2T2 B4,*B15[23]	1	371007		0
0x806832	0x627	MVK.L2 0.B4	1	371008	<Sou...	0
0x806834	0xD223	SET.S2 B4,22,22,B4	1	371009		0
0x806836	0x9F45	STW.D2T2 B4,*B15[24]	1	371010		0

c. The example profile result:

Function Profiler views												
	Function	Calls	Excl Min	Excl Max	Excl Average	Excl Total	Incl Min	Incl Max	Incl Average	Incl Total	Excl Percent	Incl Percent
1	appFunction	22	4831	4831	4746.64	104426	17143	17143	16842.64	370538	28.14	36.77
2	doneFunction	1	6	6	6.00	6	6	6	6.00	6	0.00	0.00
3	endFunction	10...	216	216	216.00	228744	216	216	216.00	228744	61.64	22.70
4	main	1	-	-	503.00	503	-	-	371104.00	371104	0.14	36.82
5	startFunction	10...	36	36	36.00	37368	36	36	36.00	37368	10.07	3.71
6	unknown_0x808fc0_0x...	1	-	-	11.00	11	-	-	11.00	11	0.00	0.00
7	unknown_0x809200_0...	1	35	35	35.00	35	46	46	46.00	46	0.01	0.00
8	unknown_0x809258_0...	1	2	2	2.00	2	11	11	11.00	11	0.00	0.00
9	unknown_0x809260_0...	1	9	9	9.00	9	9	9	9.00	9	0.00	0.00