Trace Tutorial Overview

The objective of this tutorial is to acquaint you with the basic use of the Trace System software. The Trace System software includes the following:

- The Trace Control – The Trace Control allows you to set trace collection related options, such as synchronizing trace collection with target execution, whether to stall the CPU if the on-chip trace FIFOs become full, and so on.
- The Trace Display - The Trace Display presents the collected trace data in columns and provides capabilities to search, bookmark, save and export the data.
- The CCStudio Breakpoint Manager and Event Sequencer – These tools allow you to create Advanced Event Triggering actions that determine when and how to collect trace data, as well as what data to collect.

The following assumptions are made for this tutorial:

- This tutorial is based on using Code Composer Studio Version 3.3 using a Spectrum Digital 6416 DSK target board, with a TI XDS560T Trace Cable and a Blackhawk USB emulator.

  The Trace System user interface may vary depending upon your version of Code Composer Studio, target board, and emulator. The tutorial attempts to be as agnostic of these issues as possible.
- The tutorial assumes that you have already set up Code Composer Studio, the target board, and the emulator as described in the appropriate documentation.

Note: If you have previously run this tutorial or have used the Trace System software before running this tutorial, the screen shots in the tutorial may be different than what you see on your screen. This can be due to your having previously set options other than what the tutorial covers, and or hiding Trace Display columns, which is a setting preserved across Trace Display invocations.

Initializing the Target System

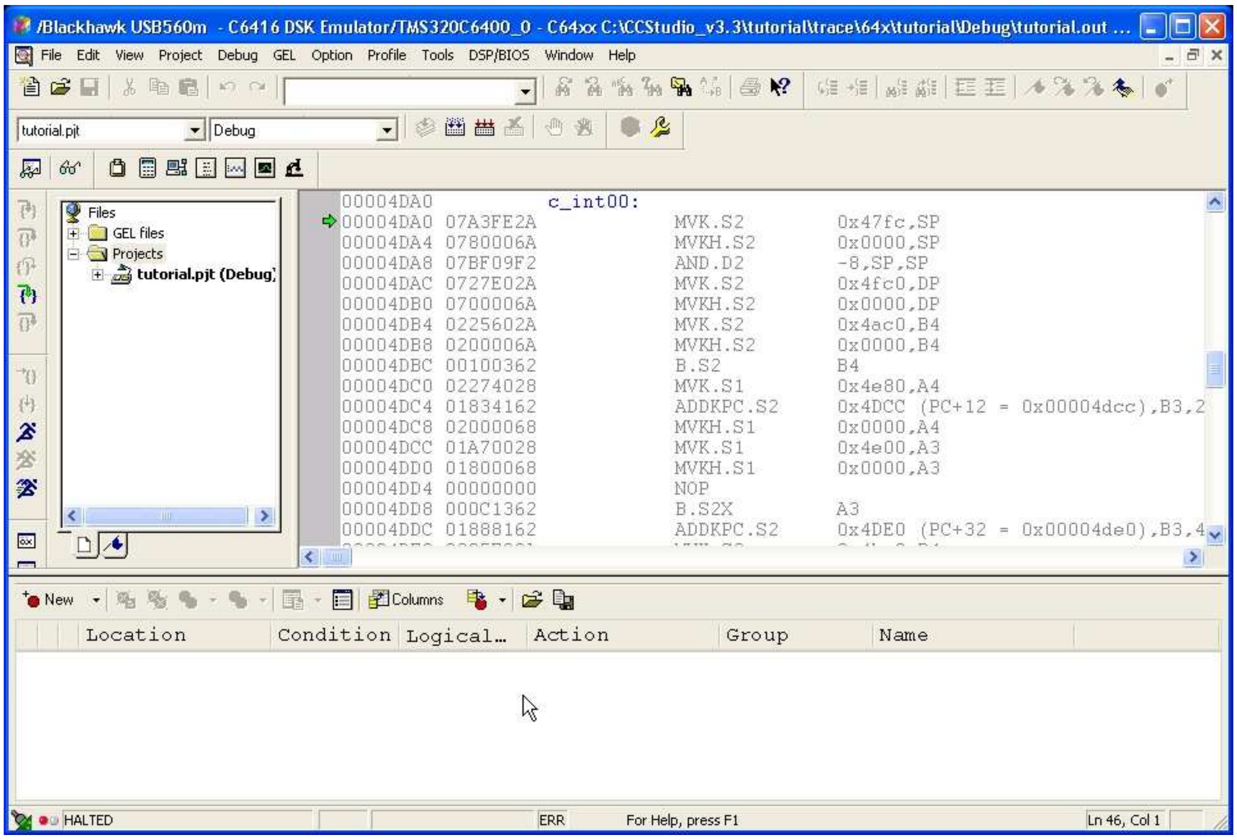
To initialize a target system:

1. Power the target board down and back up again.
2. Start Code Composer Studio.
3. Connect to the target by selecting Debug ® Connect.
4. Remove any existing breakpoints and reset the External Memory Interface by selecting GEL ® Resets ® Reset_BreakPts_and_EMIF. (The exact menu item text may be different for specific devices).
5. Flush the on-chip cache by selecting GEL ® Resets ® Flush Cache. (The exact menu item text may be different for specific devices).
6. Load the tutorial project located at CCStudioInstall \tutorial\trace\Device \tutorial\tutorial.pjt with Project ® Open… (For example: CCStudioInstall\tutorial\trace\64x\tutorial\tutorial.pjt)
7. Rebuild the tutorial project by selecting Project ® Rebuild All.
8. Load the target program by selecting File ® Load Program… then browse to the Debug directory and select tutorial.out.
9. Click Open.

How to Trace Your Target Application

The following steps demonstrate how to use the CCStudio Breakpoint Manager to program the on-chip AET hardware to collect Program Address and cycle accurate Timestamp information from your target application by using the Trace On Action.

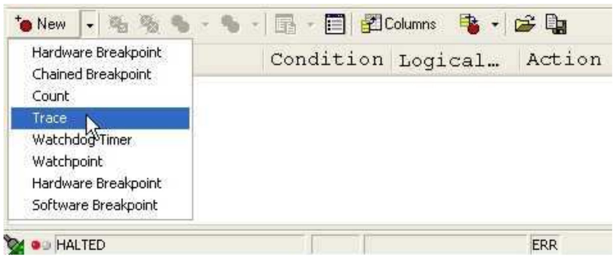
1. When initialization is complete, start the Breakpoint Manager by selecting Tools ® XDS560 Trace ® Setup… or Debug ® Breakpoints. The Breakpoint Manager opens at the bottom of the CCStudio window.


/Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0 - C64xx C:\CCStudio_v3.3\tutorial\trace\64x\tutorial\Debug\tutorial.out ...
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
tutorial.pjt
Debug
Files
GEL files
Projects
tutorial.pjt (Debug)
New
Columns
HALTED
ERR
For Help, press F1
Ln 46, Col 1

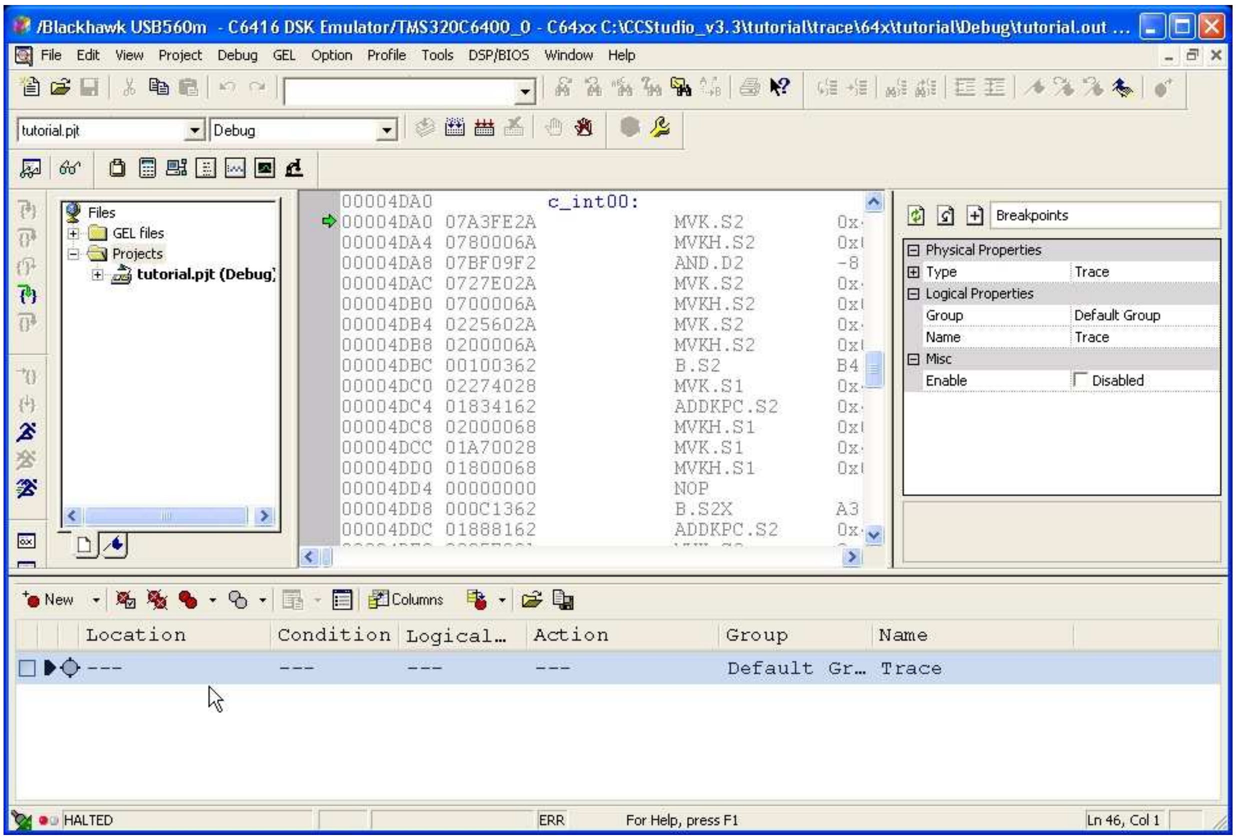
```
00004DA0                c_int00:
00004DA0 07A3FE2A                 MVK.S2        0x47fc,SP
00004DA4 0780006A                 MVKH.S2       0x0000,SP
00004DA8 07BF09F2                 AND.D2        -8,SP,SP
00004DAC 0727E02A                 MVK.S2        0x4fc0,DP
00004DB0 0700006A                 MVKH.S2       0x0000,DP
00004DB4 0225602A                 MVK.S2        0x4ac0,B4
00004DB8 0200006A                 MVKH.S2       0x0000,B4
00004DBC 00100362                 B.S2          B4
00004DC0 02274028                 MVK.S1        0x4e80,A4
00004DC4 01834162                 ADDKPC.S2     0x4DCC (PC+12 = 0x00004dcc),B3,2
00004DC8 02000068                 MVKH.S1       0x0000,A4
00004DCC 01A70028                 MVK.S1        0x4e00,A3
00004DD0 01800068                 MVKH.S1       0x0000,A3
00004DD4 00000000                 NOP
00004DD8 000C1362                 B.S2X         A3
00004DDC 01888162                 ADDKPC.S2     0x4DE0 (PC+32 = 0x00004de0),B3,4
```

	Location	Condition	Logical...	Action	Group	Name

2. Click New item dropdown list (black triangle next to the New icon in the toolbar) in the Breakpoint Manager and select Trace.



3. The Property window for the Trace action should open automatically in the right side of the CCStudio window. If it does not automatically open, click on the Open Property Window icon ( ) in the toolbar of the Breakpoint Manager window.


/Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0 - C64xx C:\CCStudio_v3.3\tutorial\trace\64x\tutorial\Debug\tutorial.out ...

```
00004DA0            c_int00:
00004DA0  07A3FE2A            MVK.S2      0x
00004DA4  0780006A            MVKH.S2     0x
00004DA8  07BF09F2            AND.D2      -8
00004DAC  0727E02A            MVK.S2      0x
00004DB0  0700006A            MVKH.S2     0x
00004DB4  0225602A            MVK.S2      0x
00004DB8  0200006A            MVKH.S2     0x
00004DBC  00100362            B.S2        B4
00004DC0  02274028            MVK.S1      0x
00004DC4  01834162            ADDKPC.S2   0x
00004DC8  02000068            MVKH.S1     0x
00004DCC  01A70028            MVK.S1      0x
00004DD0  01800068            MVKH.S1     0x
00004DD4  00000000            NOP
00004DD8  000C1362            B.S2X       A3
00004DDC  01888162            ADDKPC.S2   0x
```

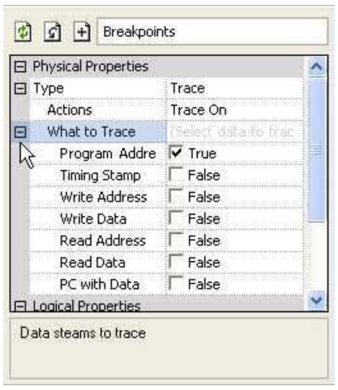
Breakpoints

⊟ Physical Properties	
⊞ Type	Trace
⊟ Logical Properties	
Group	Default Group
Name	Trace
⊟ Misc	
Enable	☐ Disabled

	Location	Condition	Logical...	Action	Group	Name
☐	---	---	---	---	Default Gr...	Trace

HALTED ERR For Help, press F1 Ln 46, Col 1

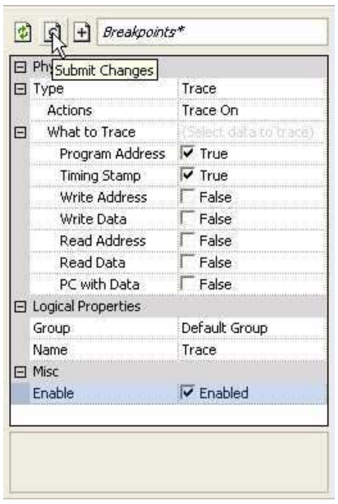
4. In the Property window, expand the Type and What to Trace properties by clicking on the plus sign to the left of those properties. Under Actions, you can see that the default Trace Action is Trace On.


Breakpoints

⊟ Physical Properties	
⊟ Type	Trace
Actions	Trace On
⊟ What to Trace	(Select data to trac
Program Addre	☑ True
Timing Stamp	☐ False
Write Address	☐ False
Write Data	☐ False
Read Address	☐ False
Read Data	☐ False
PC with Data	☐ False
⊟ Logical Properties	

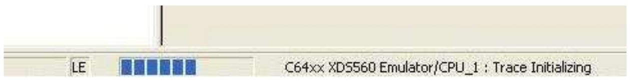
Data steams to trace

5. Make sure that the Program Address and Timing Stamp properties are checked under What to Trace, as well as the Enabled property. Finally, click on the Submit Changes icon in the Properties window toolbar to apply the changes.

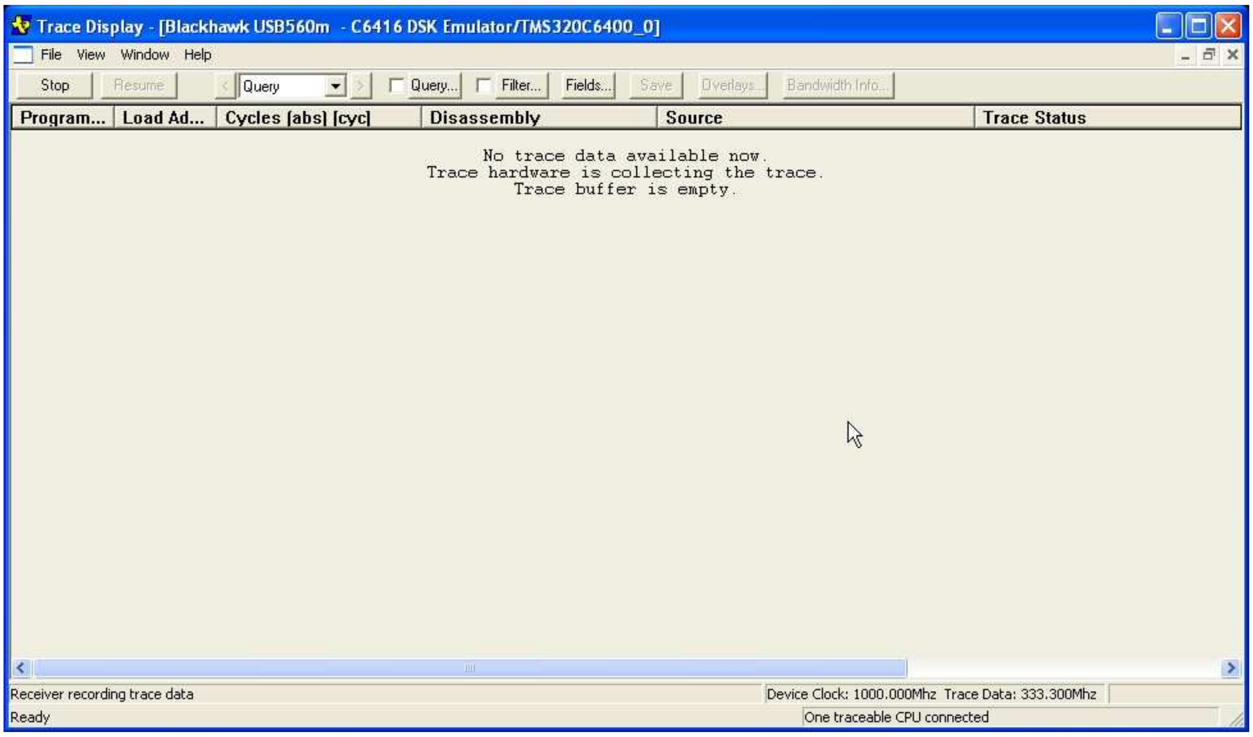

*Breakpoints**

⊟ Phy	
⊟ Type	Trace
Actions	Trace On
⊟ What to Trace	(Select data to trace)
Program Address	☑ True
Timing Stamp	☑ True
Write Address	☐ False
Write Data	☐ False
Read Address	☐ False
Read Data	☐ False
PC with Data	☐ False
⊟ Logical Properties	
Group	Default Group
Name	Trace
⊟ Misc	
Enable	☑ Enabled

Note that the Code Composer Studio Status Bar updates with a Trace Initialization progress bar. All CCS operations are prevented while trace is initializing. When this progress bar disappears, the trace system is finished initializing and you may continue with CCS operations.



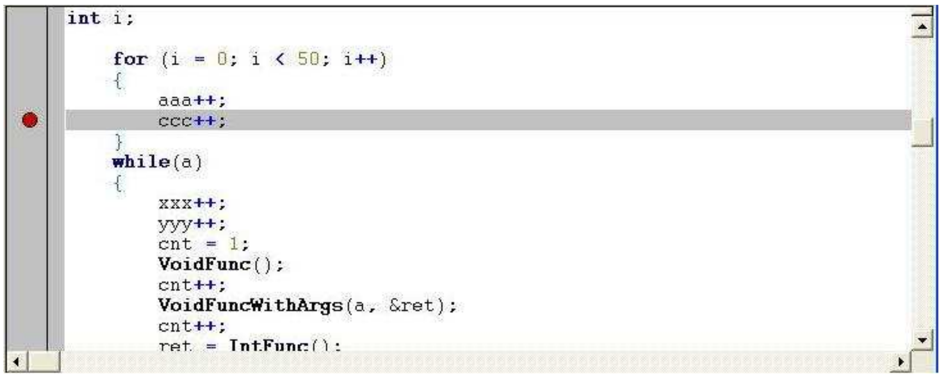
When initialization is complete, the Trace Display window automatically appears in front of the Code Composer Studio window. The Trace Display determines the correct target .out file for the loaded project (required to disassemble the collected trace data), programs the on-chip AET Hardware, initializes and attaches the receiver to the CPU, and creates a trace window for that CPU. The Trace Display's status bar changes to indicate that it is in the recording state Receiver recording trace data because the Trace On action starts recording data as soon as it is applied.



```
No trace data available now.
Trace hardware is collecting the trace.
Trace buffer is empty.
```

Receiver recording trace data
Device Clock: 1000.000Mhz Trace Data: 333.300Mhz
Ready
One traceable CPU connected

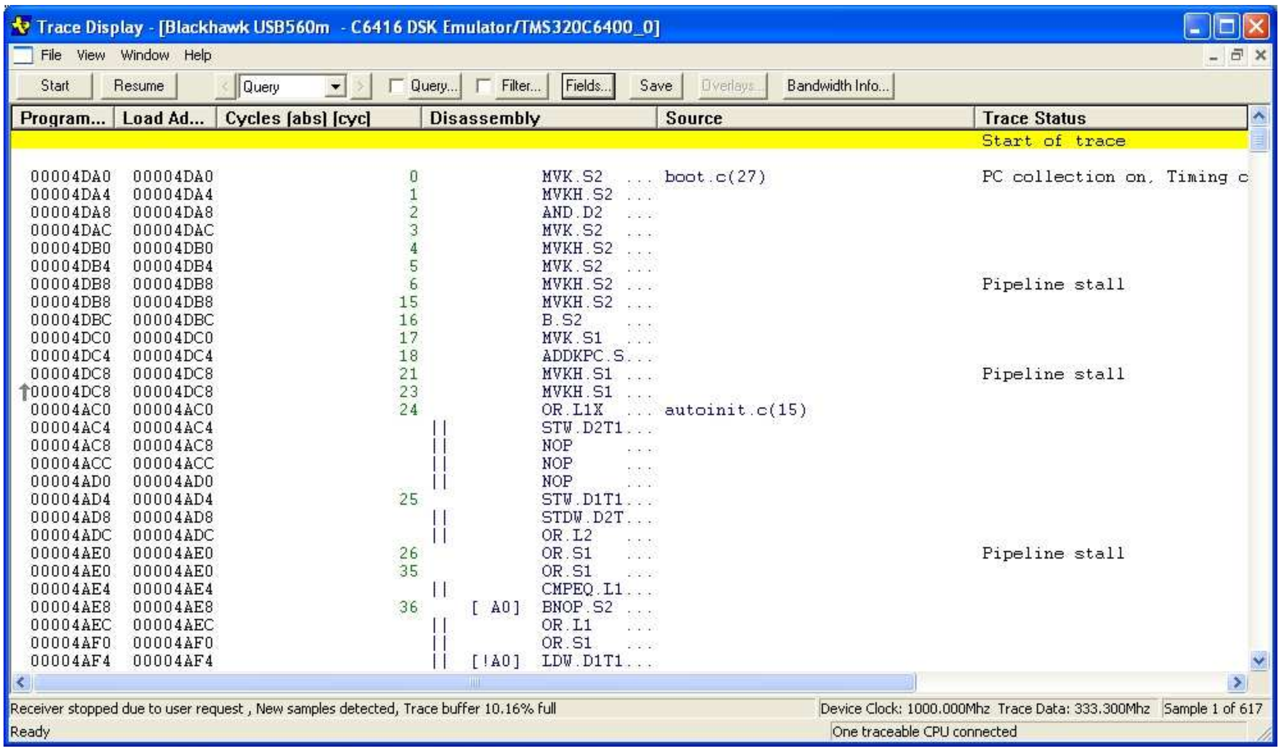
6. In Code Composer Studio, open the source file tutorial.c and set a software breakpoint at line 45, ccc++.



```
int i;

    for (i = 0; i < 50; i++)
    {
        aaa++;
        ccc++;
    }
    while(a)
    {
        xxx++;
        yyy++;
        cnt = 1;
        VoidFunc();
        cnt++;
        VoidFuncWithArgs(a, &ret);
        cnt++;
        ret = IntFunc();
```

7. Run the target by selecting Debug ® Run.

When the target hits the breakpoint, the Trace System stops recording and the Trace Display appears automatically and displays data. This behavior is due to the default Synchronize Trace with Target Execution option. The exact columns that appear in the display depend upon the target device.


Trace Display - [Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0]
File View Window Help
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
Program...  Load Ad...  Cycles [abs] [cyc]  Disassembly         Source           Trace Status
                                                                                 Start of trace

00004DA0    00004DA0     0                    MVK.S2    ... boot.c(27)           PC collection on, Timing c
00004DA4    00004DA4     1                    MVKH.S2   ...
00004DA8    00004DA8     2                    AND.D2    ...
00004DAC    00004DAC     3                    MVK.S2    ...
00004DB0    00004DB0     4                    MVKH.S2   ...
00004DB4    00004DB4     5                    MVK.S2    ...
00004DB8    00004DB8     6                    MVKH.S2   ...                      Pipeline stall
00004DB8    00004DB8    15                    MVKH.S2   ...
00004DBC    00004DBC    16                    B.S2      ...
00004DC0    00004DC0    17                    MVK.S1    ...
00004DC4    00004DC4    18                    ADDKPC.S...
00004DC8    00004DC8    21                    MVKH.S1   ...                      Pipeline stall
00004DC8    00004DC8    23                    MVKH.S1   ...
00004AC0    00004AC0    24                    OR.L1X    ... autoinit.c(15)
00004AC4    00004AC4       ||                 STW.D2T1...
00004AC8    00004AC8       ||                 NOP       ...
00004ACC    00004ACC       ||                 NOP       ...
00004AD0    00004AD0       ||                 NOP       ...
00004AD4    00004AD4    25                    STW.D1T1...
00004AD8    00004AD8       ||                 STDW.D2T...
00004ADC    00004ADC       ||                 OR.L2     ...
00004AE0    00004AE0    26                    OR.S1     ...                      Pipeline stall
00004AE0    00004AE0    35                    OR.S1     ...
00004AE4    00004AE4       ||                 CMPEQ.L1...
00004AE8    00004AE8    36          [ A0]     BNOP.S2   ...
00004AEC    00004AEC       ||                 OR.L1     ...
00004AF0    00004AF0       ||                 OR.S1     ...
00004AF4    00004AF4       ||       [!A0]     LDW.D1T1...
```

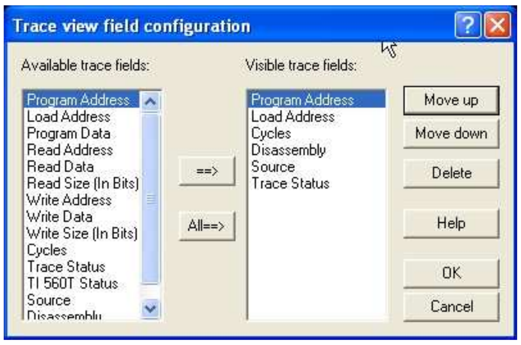
```
Receiver stopped due to user request , New samples detected, Trace buffer 10.16% full    Device Clock: 1000.000Mhz  Trace Data: 333.300Mhz  Sample 1 of 617
Ready                                                                                     One traceable CPU connected
```

The status bar at the bottom of the Trace Display window indicates Receiver stopped due to user request. This is due to the default Synchronize Trace with Target Execution option.

Hiding and Reordering Trace Columns

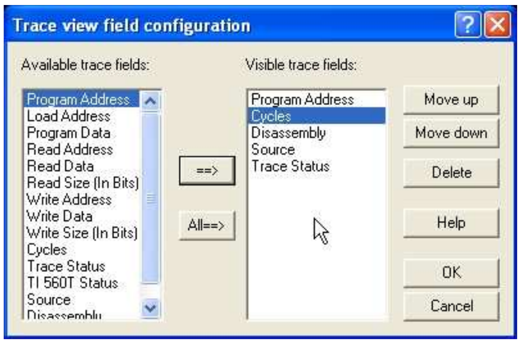
By default, the Trace Display shows the columns of trace data for the most typical debug use cases. It is possible that your debug use case will require collecting and analyzing additional types of trace data. The Trace Display allows you to hide, add, or reorder the columns that are displayed.

1. To hide columns, click on the Fields… button.

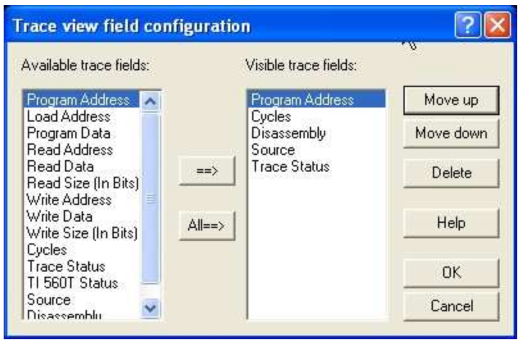


Note: The available trace fields list varies based on the target device.

2. Since the tutorial project does not use software overlays, this field is not necessary for analysis of the trace data. Remove the Load Address field by selecting the Load Address field under the right hand pane of the dialog box (Visible trace fields), and then clicking on the Delete button. The dialog should now resemble the following:



3. You can reorder the fields by selecting the fields in the right hand pane and clicking the Move up or Move down buttons so that the columns appear in the order that makes analysis of the trace data easiest for you. You can add fields by selecting them in the "Available trace fields" list, and then clicking on the => button to move them to the "Visible trace fields" list. For this tutorial, you should not add or reorder the fields being displayed; so be sure that the fields dialog appears as below when you have finished experimenting with adding and reordering fields.



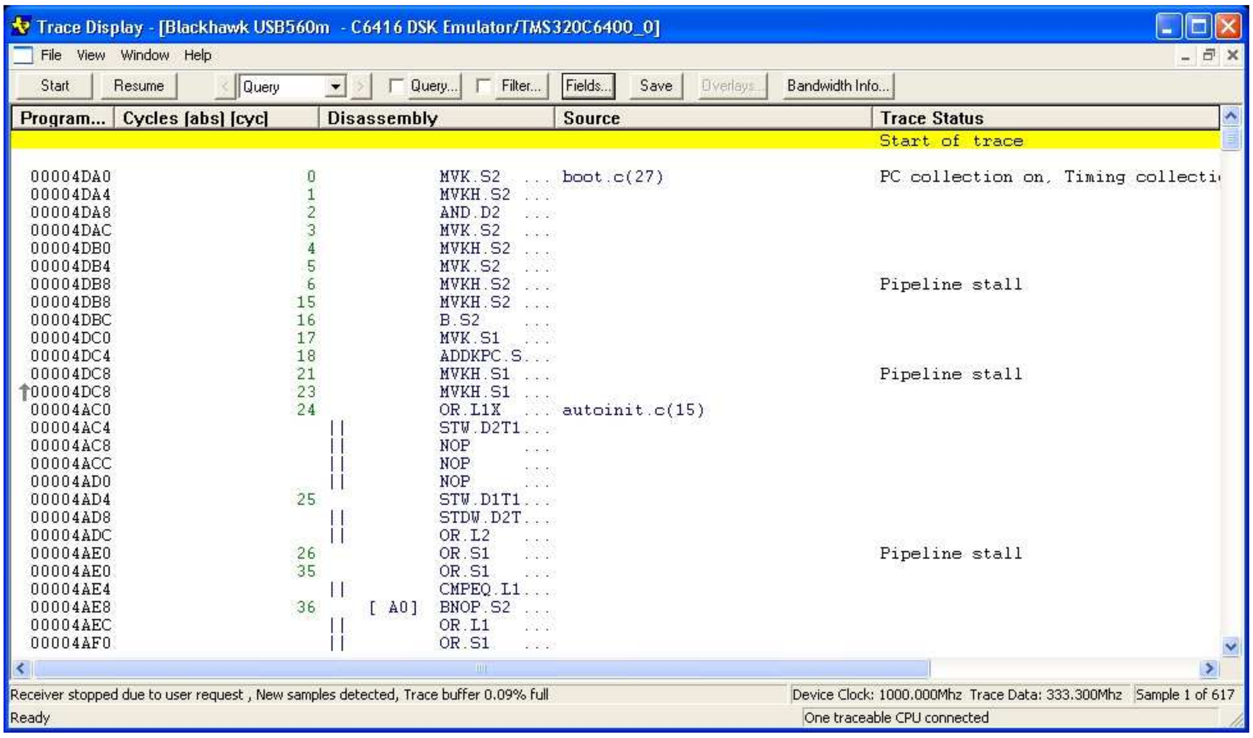
Trace view field configuration

Available trace fields:
- Program Address
- Load Address
- Program Data
- Read Address
- Read Data
- Read Size (In Bits)
- Write Address
- Write Data
- Write Size (In Bits)
- Cycles
- Trace Status
- TI 560T Status
- Source
- Disassembly

Visible trace fields:
- Program Address
- Cycles
- Disassembly
- Source
- Trace Status

4. Click OK to dismiss the Fields dialog.

   The Trace Display columns should now appear as below. (The data may differ.)



Trace Display - [Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0]

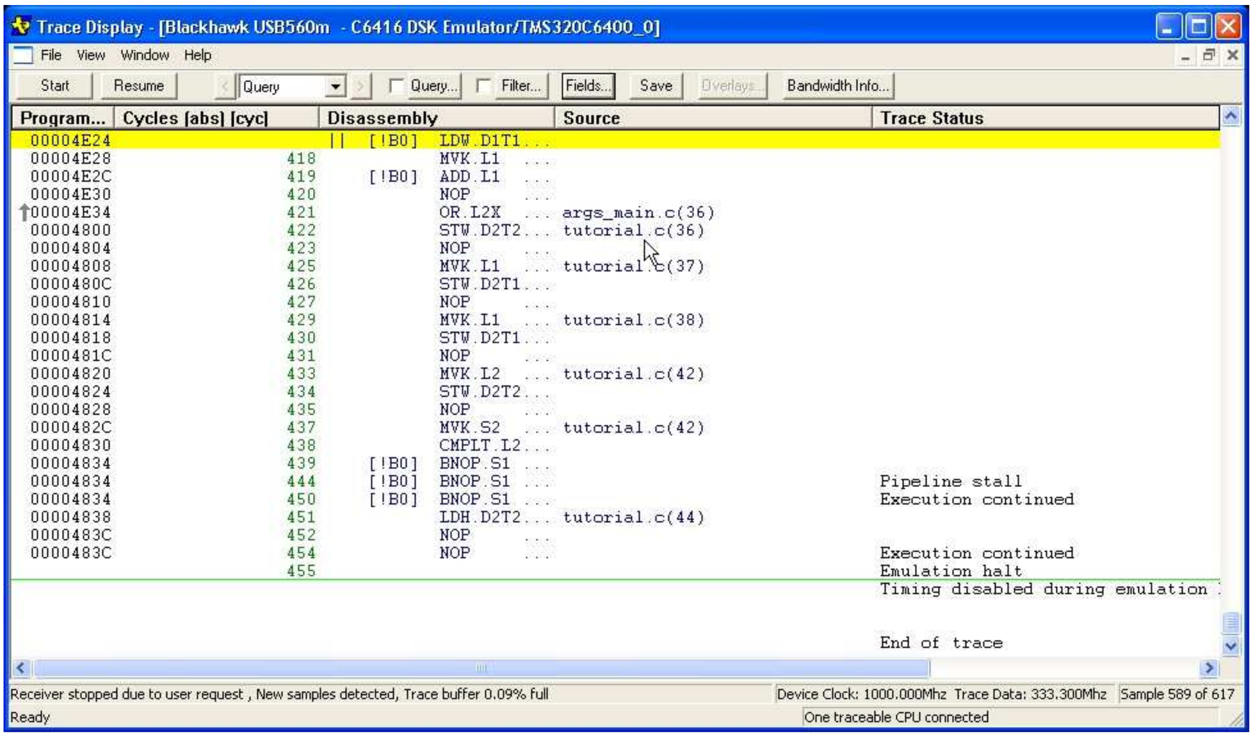
File View Window Help
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
Program...  Cycles (abs) (cyc)   Disassembly          Source         Trace Status
                                                                     Start of trace

00004DA0           0              MVK.S2    ...   boot.c(27)         PC collection on, Timing collecti
00004DA4           1              MVKH.S2   ...
00004DA8           2              AND.D2    ...
00004DAC           3              MVK.S2    ...
00004DB0           4              MVKH.S2   ...
00004DB4           5              MVK.S2    ...
00004DB8           6              MVKH.S2   ...                      Pipeline stall
00004DB8          15              MVKH.S2   ...
00004DBC          16              B.S2      ...
00004DC0          17              MVK.S1    ...
00004DC4          18              ADDKPC.S...
00004DC8          21              MVKH.S1   ...                      Pipeline stall
00004DC8          23              MVKH.S1   ...
00004AC0          24              OR.L1X    ...   autoinit.c(15)
00004AC4             ||           STW.D2T1...
00004AC8             ||           NOP       ...
00004ACC             ||           NOP       ...
00004AD0             ||           NOP       ...
00004AD4          25              STW.D1T1...
00004AD8             ||           STDW.D2T...
00004ADC             ||           OR.L2     ...
00004AE0          26              OR.S1     ...                      Pipeline stall
00004AE0          35              OR.S1     ...
00004AE4             ||           CMPEQ.L1...
00004AE8          36     [ A0]    BNOP.S2   ...
00004AEC             ||           OR.L1     ...
00004AF0             ||           OR.S1     ...
```

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 1 of 617
Ready | One traceable CPU connected

Scrolling and Source Code Display Synchronization

The trace tutorial program contains one C file, tutorial.c, which is not located in the tutorial project directory. In the Trace Display you will see that under the Source column, only the source file name and line number, "tutorial.c(36)", appear for the main() function instructions.

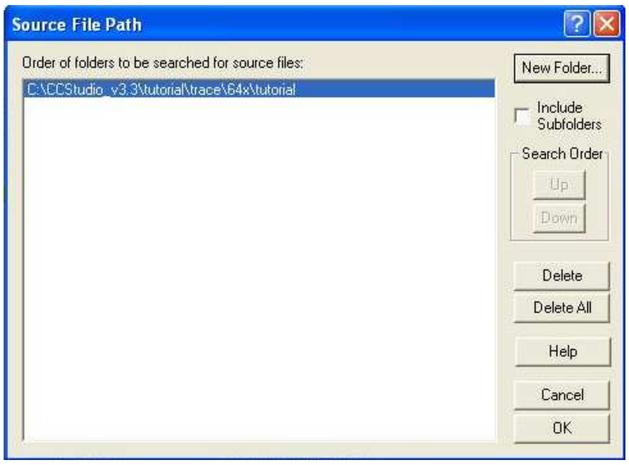

Trace Display - [Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0]
File View Window Help
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
Program...  Cycles [abs] [cyc]  Disassembly              Source              Trace Status
00004E24                        || [!B0]  LDW.D1T1...
00004E28     418                          MVK.L1   ...
00004E2C     419                   [!B0]  ADD.L1   ...
00004E30     420                          NOP      ...
00004E34     421                          OR.L2X   ...  args_main.c(36)
00004800     422                          STW.D2T2...  tutorial.c(36)
00004804     423                          NOP      ...
00004808     425                          MVK.L1   ...  tutorial.c(37)
0000480C     426                          STW.D2T1...
00004810     427                          NOP      ...
00004814     429                          MVK.L1   ...  tutorial.c(38)
00004818     430                          STW.D2T1...
0000481C     431                          NOP      ...
00004820     433                          MVK.L2   ...  tutorial.c(42)
00004824     434                          STW.D2T2...
00004828     435                          NOP      ...
0000482C     437                          MVK.S2   ...  tutorial.c(42)
00004830     438                          CMPLT.L2...
00004834     439                   [!B0]  BNOP.S1  ...
00004834     444                   [!B0]  BNOP.S1  ...                      Pipeline stall
00004834     450                   [!B0]  BNOP.S1  ...                      Execution continued
00004838     451                          LDH.D2T2...  tutorial.c(44)
0000483C     452                          NOP      ...
0000483C     454                          NOP      ...                      Execution continued
             455                                                            Emulation halt
                                                                            Timing disabled during emulation
                                                                            End of trace
```

```
Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full    Device Clock: 1000.000Mhz  Trace Data: 333.300Mhz   Sample 589 of 617
Ready                                                                                   One traceable CPU connected
```

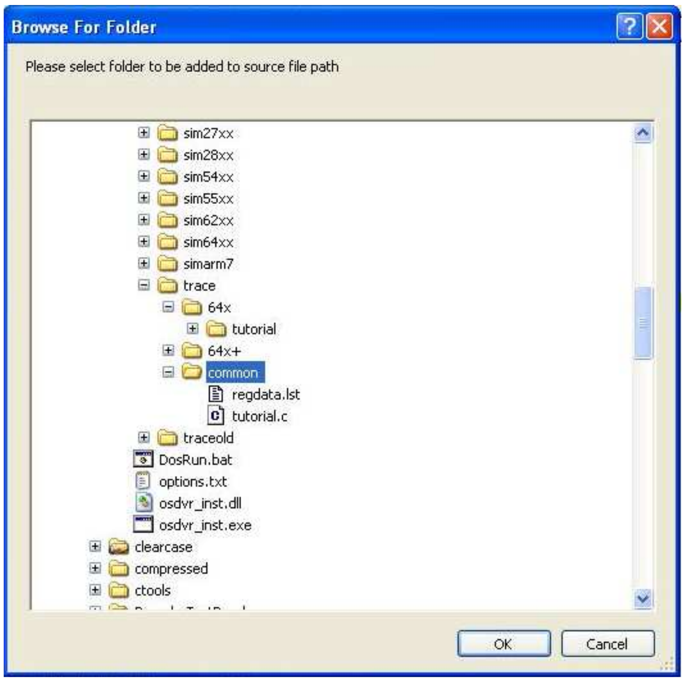
To inform the Trace Display where the tutorial.c file is located, you must add the directory to the Trace Display's Source Directory List.

1. Right click in the Trace Window and select Source ® Set Source Directory… to bring up the Source File Path Dialog.

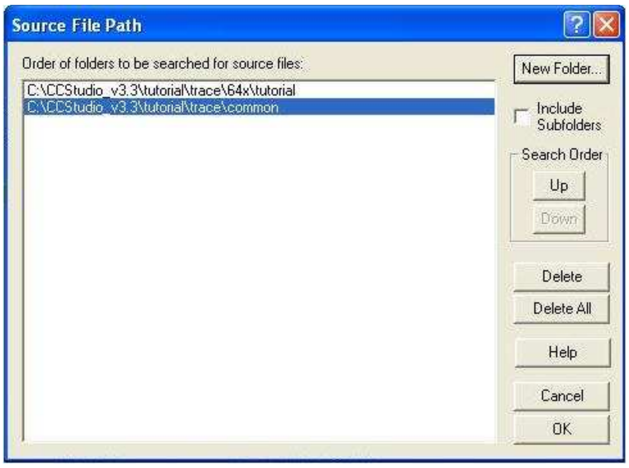


C:\CCStudio_v3.3\tutorial\trace\64x\tutorial

2. Click on the New Folder… button and navigate to and select the common directory one level up from the tutorial project directory.



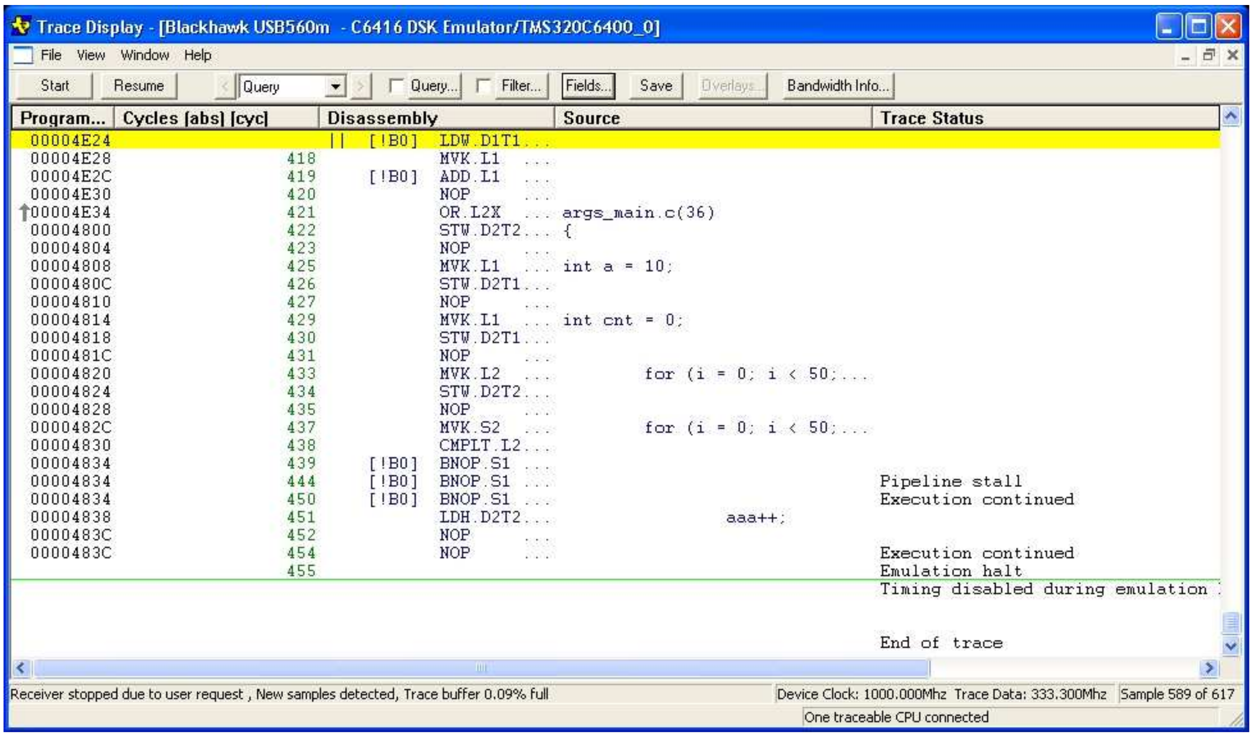
3. Click OK to dismiss the dialog. The Source File Path dialog should now contain both the tutorial project directory and the common directory as shown below.



C:\CCStudio_v3.3\tutorial\trace\64x\tutorial
C:\CCStudio_v3.3\tutorial\trace\common

4. Click OK to dismiss the Source File Path dialog.

   The Trace window should now show the source code lines for the main() function in tutorial.c as seen below, rather than just the file name and line numbers.


Trace Display - [Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0]
File View Window Help
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
Program...  Cycles (abs) (cyc)  Disassembly                 Source                      Trace Status
00004E24                        ||  [!B0]  LDW.D1T1...
00004E28                   418             MVK.L1    ...
00004E2C                   419      [!B0]  ADD.L1    ...
00004E30                   420             NOP       ...
00004E34                   421             OR.L2X    ...  args_main.c(36)
00004800                   422             STW.D2T2...    {
00004804                   423             NOP       ...
00004808                   425             MVK.L1    ...  int a = 10;
0000480C                   426             STW.D2T1...
00004810                   427             NOP       ...
00004814                   429             MVK.L1    ...  int cnt = 0;
00004818                   430             STW.D2T1...
0000481C                   431             NOP       ...
00004820                   433             MVK.L2    ...          for (i = 0; i < 50;...
00004824                   434             STW.D2T2...
00004828                   435             NOP       ...
0000482C                   437             MVK.S2    ...          for (i = 0; i < 50;...
00004830                   438             CMPLT.L2...
00004834                   439      [!B0]  BNOP.S1   ...
00004834                   444      [!B0]  BNOP.S1   ...                                  Pipeline stall
00004834                   450      [!B0]  BNOP.S1   ...                                  Execution continued
00004838                   451             LDH.D2T2...                  aaa++;
0000483C                   452             NOP       ...
0000483C                   454             NOP       ...                                  Execution continued
                           455                                                            Emulation halt
                                                                                          Timing disabled during emulation
                                                                                          End of trace
```

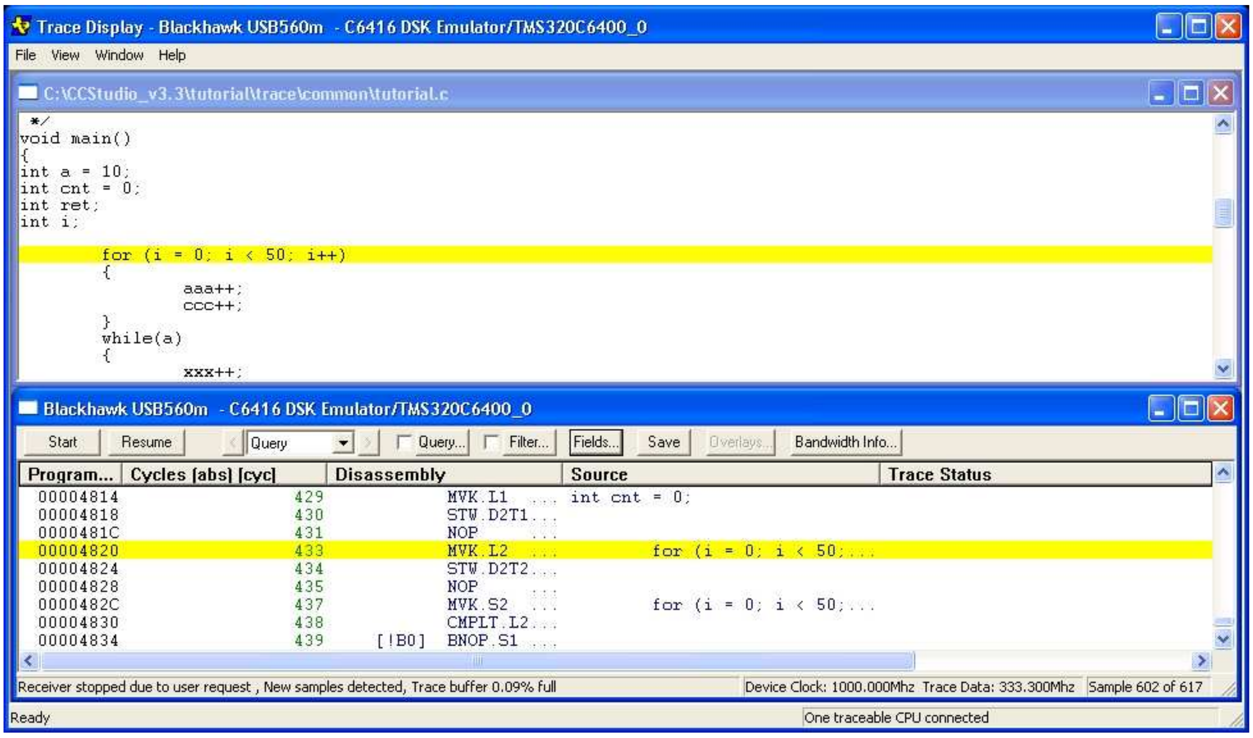
Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 589 of 617

One traceable CPU connected

As you begin to scroll through the trace window, the Trace Display opens a separate window for the tutorial.c source file. You may need to select Window ® Tile to see both the trace window and the source window in the Trace Display.

The highlighted line in the source code window will change to match the highlighted source line in the trace window as you continue to scroll in either direction. For example:

Initial scrolling results in:


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c

```
 */
void main()
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
```

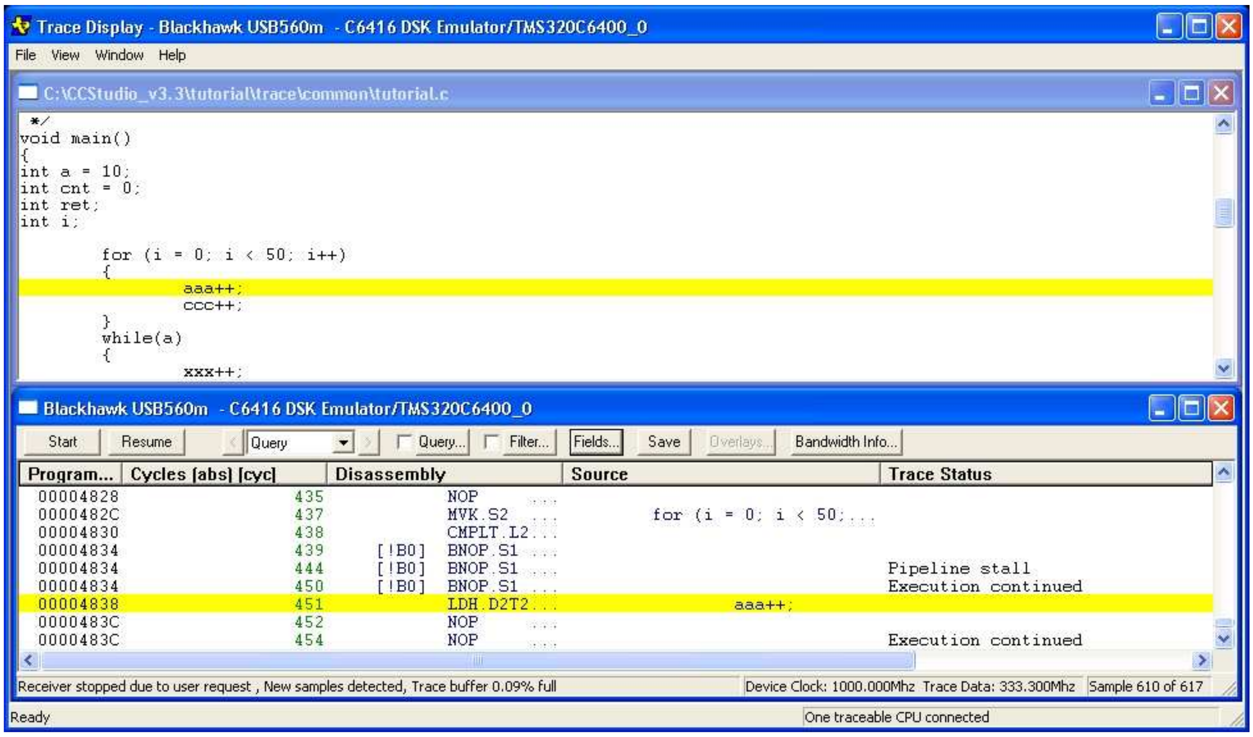
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
Program...  Cycles (abs) (cyc)  Disassembly                 Source                      Trace Status
00004814                   429             MVK.L1    ...  int cnt = 0;
00004818                   430             STW.D2T1...
0000481C                   431             NOP       ...
00004820                   433             MVK.L2    ...          for (i = 0; i < 50;...
00004824                   434             STW.D2T2...
00004828                   435             NOP       ...
0000482C                   437             MVK.S2    ...          for (i = 0; i < 50;...
00004830                   438             CMPLT.L2...
00004834                   439      [!B0]  BNOP.S1   ...
```

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 602 of 617

Ready | One traceable CPU connected

Second scrolling results in:


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c

```
 */
void main()
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
```

Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start
Resume
Query
Query...
Filter...
Fields...
Save
Overlays...
Bandwidth Info...

```
Program...  Cycles (abs) (cyc)  Disassembly            Source                  Trace Status
00004828                 435          NOP      ...
0000482C                 437          MVK.S2   ...     for (i = 0; i < 50;...
00004830                 438          CMPLT.L2...
00004834                 439    [!B0] BNOP.S1  ...
00004834                 444    [!B0] BNOP.S1  ...                             Pipeline stall
00004834                 450    [!B0] BNOP.S1  ...                             Execution continued
00004838                 451          LDH.D2T2...            aaa++;
0000483C                 452          NOP      ...
0000483C                 454          NOP      ...                             Execution continued
```

```
Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full   Device Clock: 1000.000Mhz  Trace Data: 333.300Mhz   Sample 610 of 617
Ready                                                                                    One traceable CPU connected
```

Note: As you scroll through the Trace data, the corresponding source file opens in a separate window in the Trace Display. The highlighted line in the source code window changes to match the highlighted source line in the Trace window as you continue to scroll in either direction.
For source synchronization to work, the Trace Display must know the location of the source file. Source files located in the Code Composer Studio project directory are found automatically by the Trace Display. If the Trace Display does not know the location of the source file, the Trace Display will not open a source window, and instead will show only the source file name and line number in the Source column.
As an example, this behavior is not true for the code at the beginning of the Trace Tutorial program, which is the C Run-Time Library. The source files for the C Run-Time Library are distributed as an archive file in the CCStudioInstall/Device/cgtools/lib/ directory titled rts.src. You can use the Code Generation archiver tool to extract the source files for the C Run-Time Library.

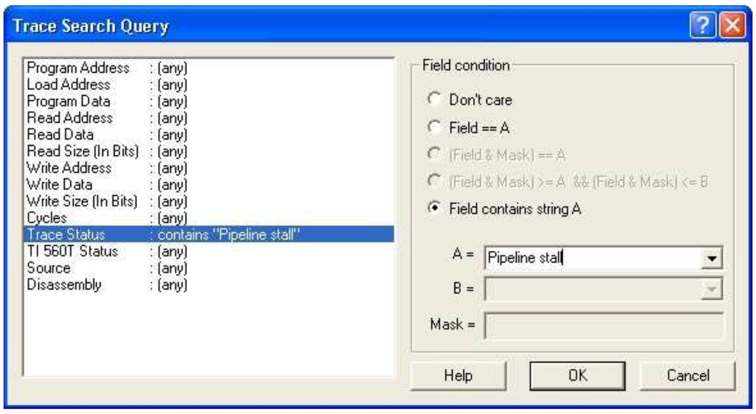
Searching Trace Data

You can also search through the data in the Trace Display by clicking on the Query… button to bring up the Trace Search Query dialog.

1. Maximize the Trace Window by clicking on the Maximize icon in the upper right corner. Then click on the Home button on your keyboard to scroll to the beginning of the trace data.

2. Select the Trace Status column in the left hand pane and click the Field contains string A radio button.

3. Enter the text Pipeline stall into the A = text edit box, then click OK.

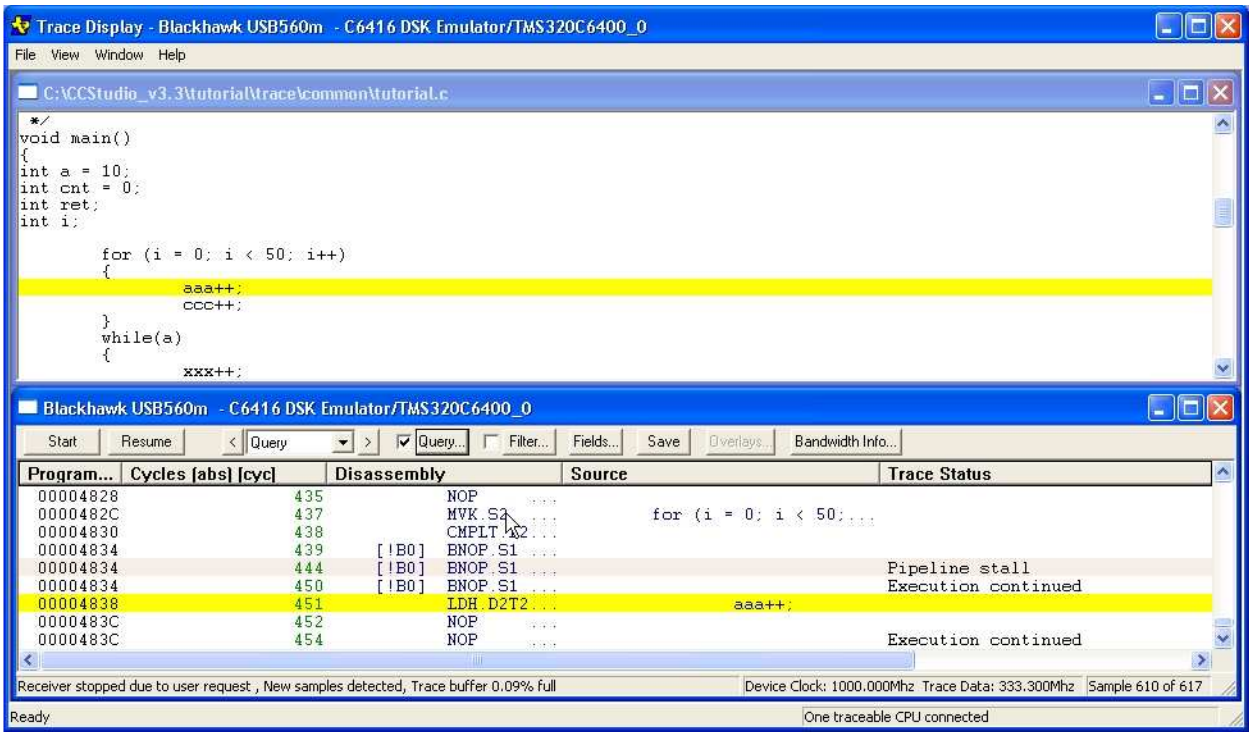


Program Address	: (any)
Load Address	: (any)
Program Data	: (any)
Read Address	: (any)
Read Data	: (any)
Read Size (In Bits)	: (any)
Write Address	: (any)
Write Data	: (any)
Write Size (In Bits)	: (any)
Cycles	: (any)
Trace Status	: contains "Pipeline stall"
TI 560T Status	: (any)
Source	: (any)
Disassembly	: (any)

Don't care
Field == A
(Field & Mask) == A
(Field & Mask) >= A && (Field & Mask) <= B
[✓ Field contains string A]

A = Pipeline stall
B =
Mask =

The Trace Display now highlights in light gray the rows that match the specified Query text.


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start
Resume
Query
Query...
Filter...
Fields...
Save
Overlays...
Bandwidth Info...

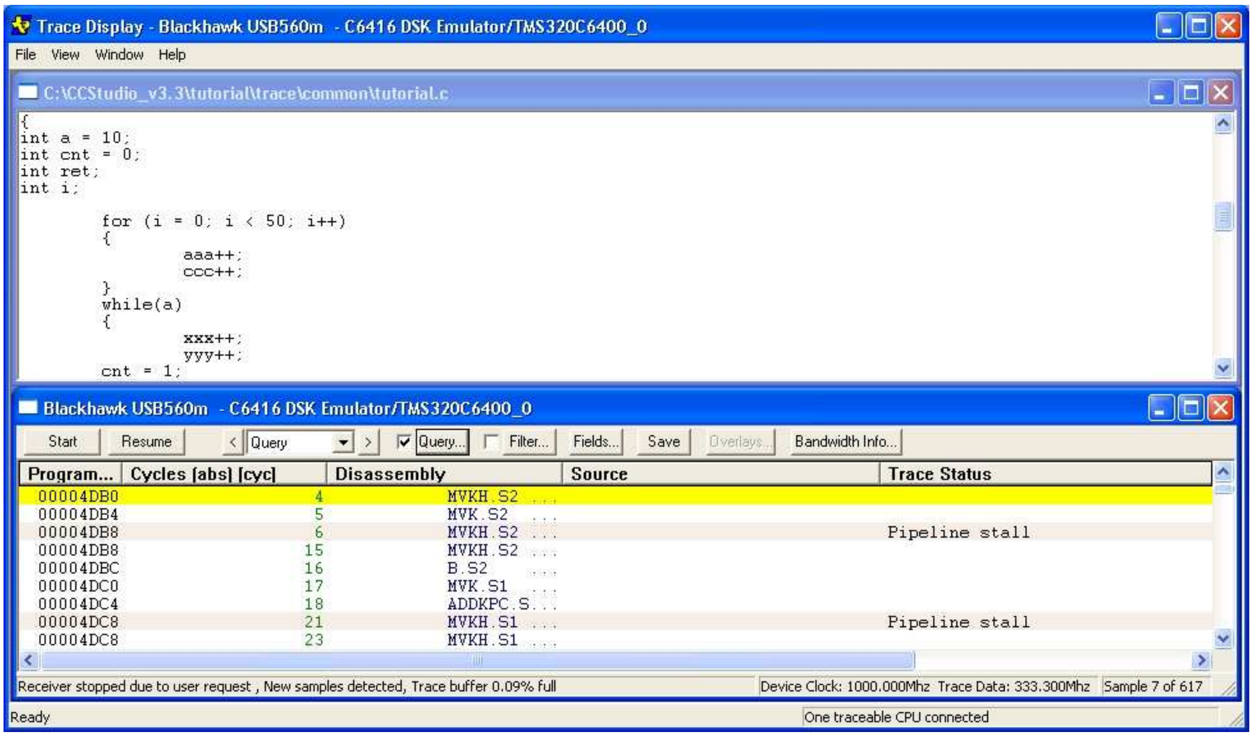
```
 */
void main()
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
```

Program...	Cycles (abs) (cyc)	Disassembly	Source	Trace Status
00004828	435	NOP ...		
0000482C	437	MVK.S2 ...	for (i = 0; i < 50;...	
00004830	438	CMPLT.S2...		
00004834	439	[!B0] BNOP.S1 ...		
00004834	444	[!B0] BNOP.S1 ...		Pipeline stall
00004834	450	[!B0] BNOP.S1 ...		Execution continued
00004838	451	LDH.D2T2...	aaa++;	
0000483C	452	NOP ...		
0000483C	454	NOP ...		Execution continued

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 610 of 617

Ready | One traceable CPU connected

4. You may need to click on the Home key and scroll down to see more than one Pipeline stall in the Trace Display window.


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start
Resume
Query
Query...
Filter...
Fields...
Save
Overlays...
Bandwidth Info...

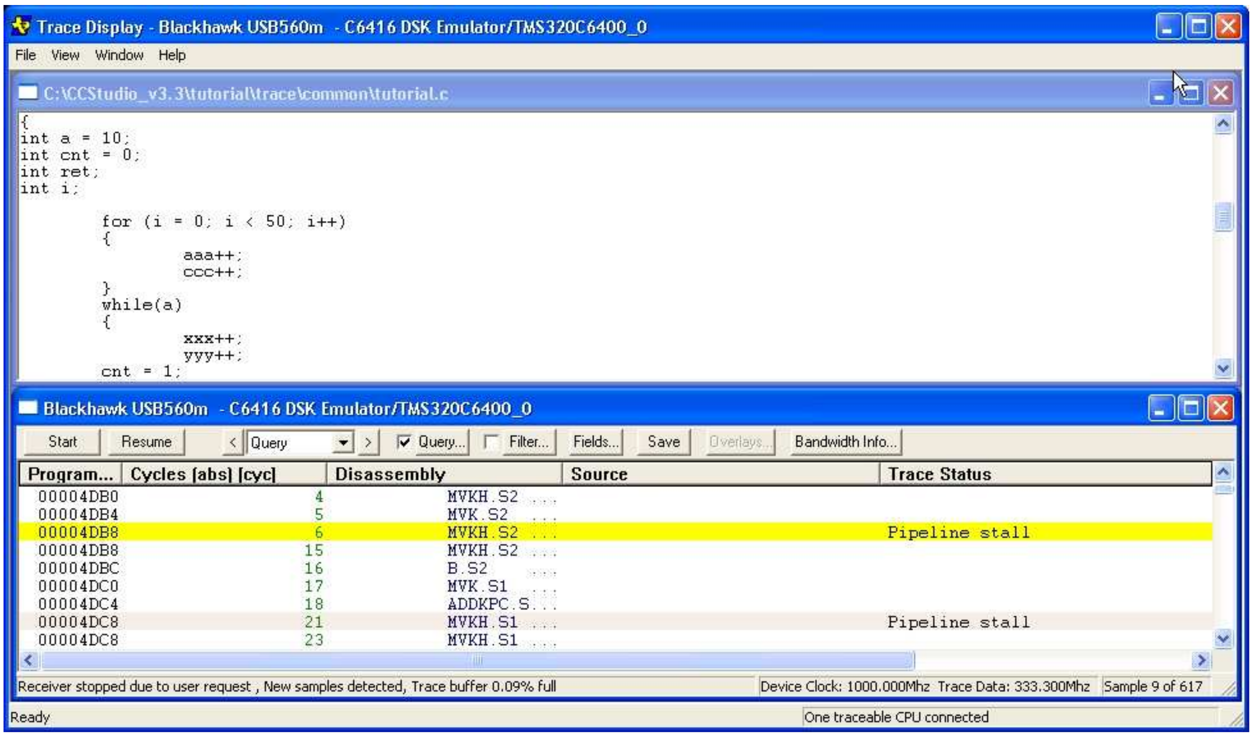
```
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
                yyy++;
        cnt = 1;
```

Program...	Cycles (abs) (cyc)	Disassembly	Source	Trace Status
00004DB0	4	MVKH.S2 ...		
00004DB4	5	MVK.S2 ...		
00004DB8	6	MVKH.S2 ...		Pipeline stall
00004DB8	15	MVKH.S2 ...		
00004DBC	16	B.S2 ...		
00004DC0	17	MVK.S1 ...		
00004DC4	18	ADDKPC.S...		
00004DC8	21	MVKH.S1 ...		Pipeline stall
00004DC8	23	MVKH.S1 ...		

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 7 of 617

Ready | One traceable CPU connected

5. In the Trace Display window, click on the > button next to the Query drop down box. The first matching line is highlighted in yellow. Clicking the > button highlights the next match, and clicking the < button highlights the previous match.



Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0

C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c

```
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
                yyy++;
        cnt = 1;
```

Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0

```
Program...  Cycles (abs) (cyc)  Disassembly     Source    Trace Status
00004DB0                     4  MVKH.S2 ...
00004DB4                     5  MVK.S2  ...
00004DB8                     6  MVKH.S2 ...              Pipeline stall
00004DB8                    15  MVKH.S2 ...
00004DBC                    16  B.S2    ...
00004DC0                    17  MVK.S1  ...
00004DC4                    18  ADDKPC.S...
00004DC8                    21  MVKH.S1 ...              Pipeline stall
00004DC8                    23  MVKH.S1 ...
```

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full | Device Clock: 1000.000Mhz Trace Data: 333.300Mhz | Sample 9 of 617

Ready | One traceable CPU connected

◀ ▶

Filtering Trace Data

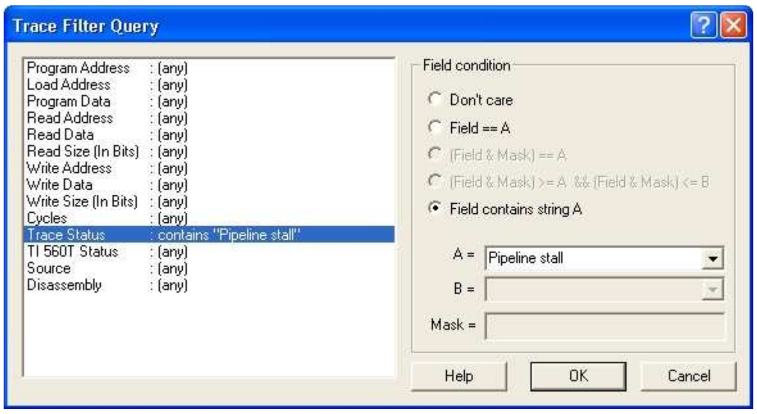
While it is often useful to be able to search for data in this find-next/previous manner, it is also useful to find all the places that match a particular set of search criteria and display only those lines. This batch searching capability is called Filtering.

1. Click on the checkbox next to the Query… to turn off the Query highlighting.

2. Next, click on the Filter… button and set up the same search criteria that we did in the Trace Search Query dialog in the Trace Filter Query dialog:

   a. Select the Trace Status column in the left hand pane and click the Field contains string A radio button.

   b. Enter the text Pipeline stall into the A = text edit box.



Trace Filter Query

Program Address	: (any)
Load Address	: (any)
Program Data	: (any)
Read Address	: (any)
Read Data	: (any)
Read Size (In Bits)	: (any)
Write Address	: (any)
Write Data	: (any)
Write Size (In Bits)	: (any)
Cycles	: (any)
Trace Status	: contains "Pipeline stall"
TI 560T Status	: (any)
Source	: (any)
Disassembly	: (any)

Field condition

- Don't care
- Field == A
- (Field & Mask) == A
- (Field & Mask) >= A && (Field & Mask) <= B
- [✓ Field contains string A]

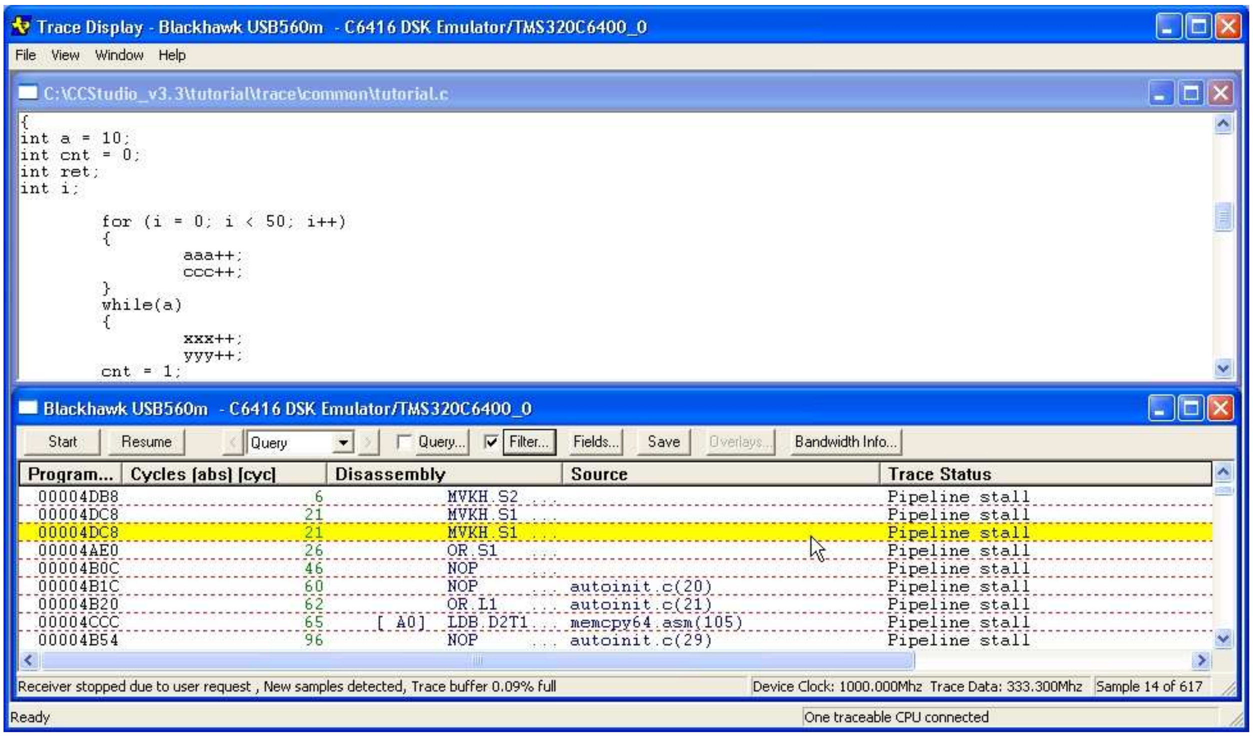
A = Pipeline stall

B =

Mask =

3. Click OK.

   The Trace Display now shows only those lines that match the Filter search criteria. The currently selected row is still highlighted.


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

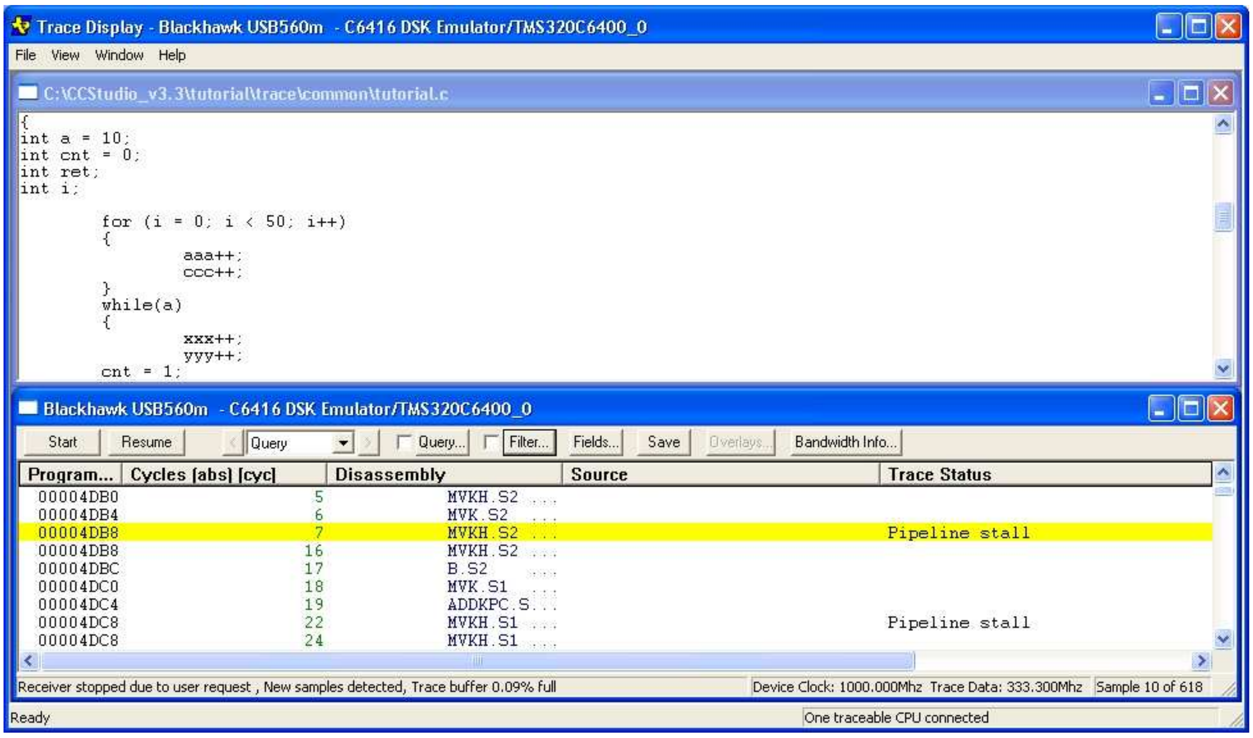
```
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
                yyy++;
        cnt = 1;
```

Program...	Cycles [abs] [cyc]	Disassembly	Source	Trace Status
00004DB8	6	MVKH.S2 ...		Pipeline stall
00004DC8	21	MVKH.S1 ...		Pipeline stall
00004DC8	21	MVKH.S1 ...		Pipeline stall
00004AE0	26	OR.S1 ...		Pipeline stall
00004B0C	46	NOP ...		Pipeline stall
00004B1C	60	NOP ...	autoinit.c(20)	Pipeline stall
00004B20	62	OR.L1 ...	autoinit.c(21)	Pipeline stall
00004CCC	65	[A0] LDB.D2T1...	memcpy64.asm(105)	Pipeline stall
00004B54	96	NOP ...	autoinit.c(29)	Pipeline stall

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full Device Clock: 1000.000Mhz Trace Data: 333.300Mhz Sample 14 of 617

Ready One traceable CPU connected

4. You can turn off filtering by clicking the checkbox to the left of the Filter… icon. Note that the currently selected sample in the filtered view remains selected in the unfiltered view. This is a useful way to find a particular occurrence of a filtered sample (such as the third Pipeline stall), and then see the surrounding code once filtering is turned off.


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start Resume Query Query... Filter... Fields... Save Overlays... Bandwidth Info...

```
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
                yyy++;
        cnt = 1;
```

Program...	Cycles [abs] [cyc]	Disassembly	Source	Trace Status
00004DB0	5	MVKH.S2 ...		
00004DB4	6	MVK.S2 ...		
00004DB8	7	MVKH.S2 ...		Pipeline stall
00004DB8	16	MVKH.S2 ...		
00004DBC	17	B.S2 ...		
00004DC0	18	MVK.S1 ...		
00004DC4	19	ADDKPC.S...		
00004DC8	22	MVKH.S1 ...		Pipeline stall
00004DC8	24	MVKH.S1 ...		

Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full Device Clock: 1000.000Mhz Trace Data: 333.300Mhz Sample 10 of 618

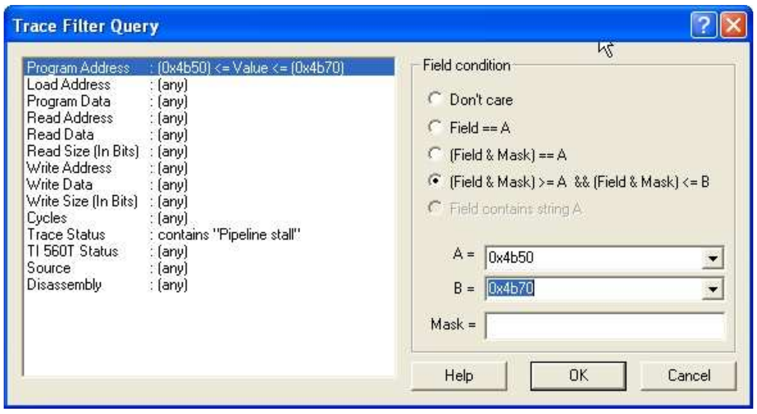
Ready One traceable CPU connected

More Complex Filtering

Filtering is useful to determine if a particular Trace Status value, say Pipeline stalls (see Filtering Trace Data), has occurred at all in the collected trace data. By setting more than one field's search criteria, you can narrow down your search. For example, you could search for all Pipelines stalls while the Program Address value is in the range of the start and end addresses of a particular function.

To demonstrate more complex Filtering criteria, let us assume we are only interested in Pipeline stalls that occur in the Program Address range 0x4B50 – 0x4B70.

1. Click on the Filter button to bring up the Trace Filter Query dialog again.
2. Select the Program Address column in the left hand pane.
3. Click on (Field & Mask) >= A && (Field & Mask) <= B, and enter the value 0x4B50 for A and 0x4B70 for B, then click OK.

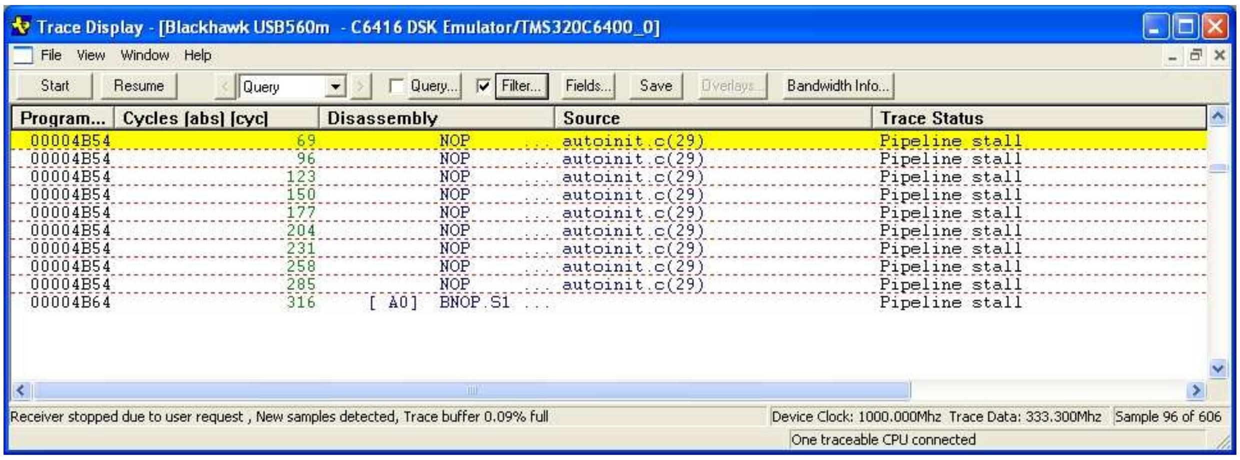


```
Program Address    : (0x4b50) <= Value <= (0x4b70)
Load Address       : (any)
Program Data       : (any)
Read Address       : (any)
Read Data          : (any)
Read Size (In Bits): (any)
Write Address      : (any)
Write Data         : (any)
Write Size (In Bits): (any)
Cycles             : (any)
Trace Status       : contains "Pipeline stall"
TI 560T Status     : (any)
Source             : (any)
Disassembly        : (any)
```

Field condition:
- ( ) Don't care
- ( ) Field == A
- ( ) (Field & Mask) == A
- (●) (Field & Mask) >= A && (Field & Mask) <= B
- ( ) Field contains string A

A = 0x4b50
B = 0x4b70
Mask =

The Trace Display should now appear as below.



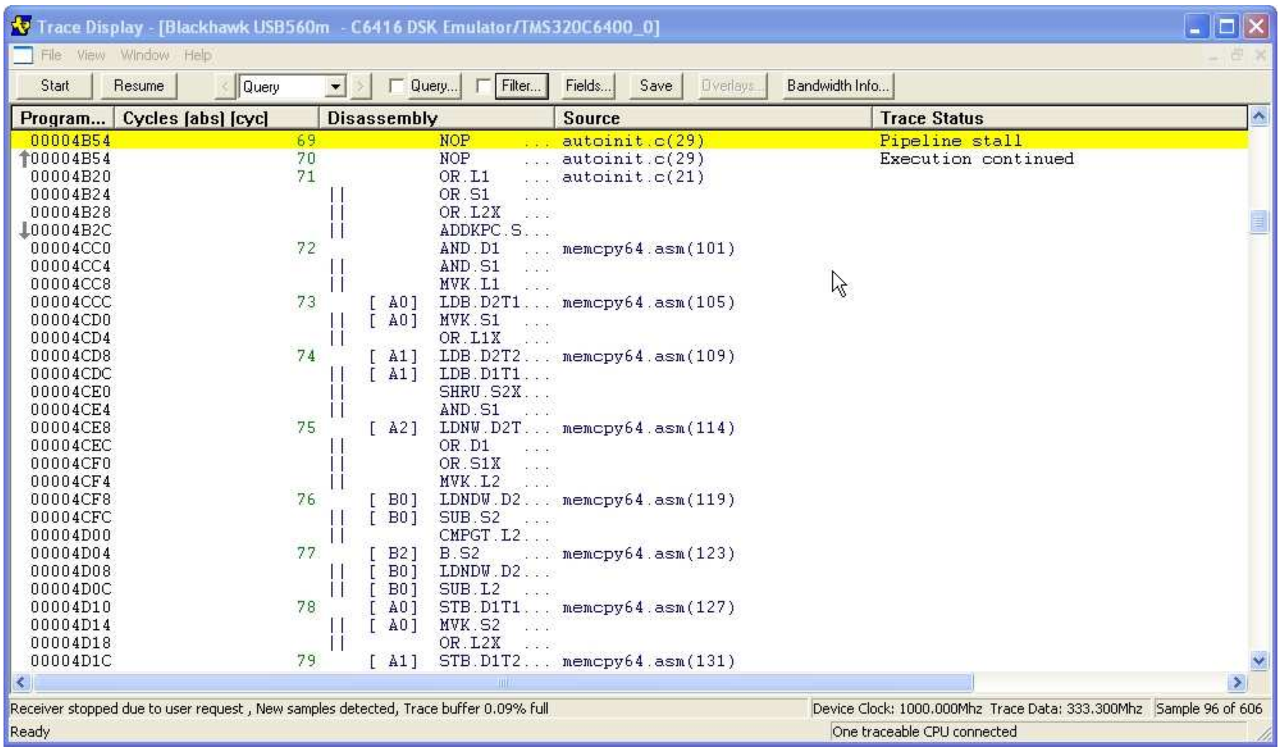
```
Program...  Cycles (abs) (cyc)  Disassembly          Source              Trace Status
00004B54     69                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54     96                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    123                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    150                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    177                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    204                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    231                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    258                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54    285                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B64    316          [ A0]  BNOP.S1    ...                     Pipeline stall
```

```
Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full   Device Clock: 1000.000Mhz  Trace Data: 333.300Mhz  Sample 96 of 606
                                                                                        One traceable CPU connected
```

Tip: Another useful search strategy is to filter for the Trace Status rows that contain “Interrupt”, then turn off filtering by un-checking the Filter checkbox to see where each interrupt occurred. This is helpful to verify that interrupts are occurring at the correct priority level. This is also a good way to detect if your Interrupt Service Table Pointer Register contains the correct address by noting where the interrupt branches to when it does occur.

4. Un-check the checkbox next to the Filter… button to turn off all filtering now.

   The Trace Display format should now appear as below. (Your data may differ.)



```
Program...  Cycles (abs) (cyc)  Disassembly          Source              Trace Status
00004B54     69                 NOP        ...  autoinit.c(29)     Pipeline stall
00004B54     70                 NOP        ...  autoinit.c(29)     Execution continued
00004B20     71                 OR.L1      ...  autoinit.c(21)
00004B24             ||         OR.S1      ...
00004B28             ||         OR.L2X     ...
00004B2C             ||         ADDKPC.S...
00004CC0     72                 AND.D1     ...  memcpy64.asm(101)
00004CC4             ||         AND.S1     ...
00004CC8             ||         MVK.L1     ...
00004CCC     73          [ A0]  LDB.D2T1...     memcpy64.asm(105)
00004CD0             ||  [ A0]  MVK.S1     ...
00004CD4             ||         OR.L1X     ...
00004CD8     74          [ A1]  LDB.D2T2...     memcpy64.asm(109)
00004CDC             ||  [ A1]  LDB.D1T1...
00004CE0             ||         SHRU.S2X...
00004CE4             ||         AND.S1     ...
00004CE8     75          [ A2]  LDNW.D2T...     memcpy64.asm(114)
00004CEC             ||         OR.D1      ...
00004CF0             ||         OR.S1X     ...
00004CF4             ||         MVK.L2     ...
00004CF8     76          [ B0]  LDNDW.D2...     memcpy64.asm(119)
00004CFC             ||  [ B0]  SUB.S2     ...
00004D00             ||         CMPGT.L2...
00004D04     77          [ B2]  B.S2       ...  memcpy64.asm(123)
00004D08             ||  [ B0]  LDNDW.D2...
00004D0C             ||  [ B0]  SUB.L2     ...
00004D10     78          [ A0]  STB.D1T1...     memcpy64.asm(127)
00004D14             ||  [ A0]  MVK.S2     ...
00004D18             ||         OR.L2X     ...
00004D1C     79          [ A1]  STB.D1T2...     memcpy64.asm(131)
```

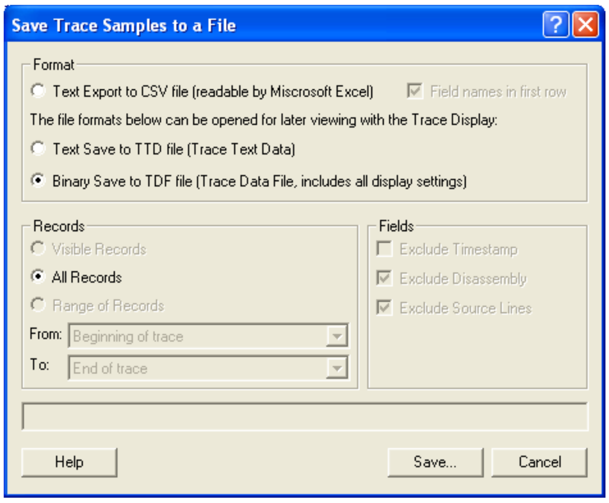
```
Receiver stopped due to user request , New samples detected, Trace buffer 0.09% full   Device Clock: 1000.000Mhz  Trace Data: 333.300Mhz  Sample 96 of 606
Ready                                                                                   One traceable CPU connected
```

◀ ▶

Saving and Exporting Trace Data

You can save the collected trace data in different formats for subsequent offline analysis or re-opening in the Trace Display.

1. Click on the Save button in the Trace Display.



Save Trace Samples to a File

Format
- ( ) Text Export to CSV file (readable by Miscrosoft Excel) — ✓ Field names in first row
- The file formats below can be opened for later viewing with the Trace Display:
- ( ) Text Save to TTD file (Trace Text Data)
- (●) Binary Save to TDF file (Trace Data File, includes all display settings)

Records
- ( ) Visible Records
- (●) All Records
- ( ) Range of Records
- From: Beginning of trace
- To: End of trace

Fields
- ☐ Exclude Timestamp
- ✓ Exclude Disassembly
- ✓ Exclude Source Lines

The trace data can be saved in three different file formats:

- Comma Separated Value (.csv) – This format is used for importing into spreadsheets and with scripting languages such as Perl or Python. This format cannot be reopened by the Trace Display.
- Text (.ttd) – This format is useful for parsing with scripting languages such as Perl or Python. This format can be reopened by the Trace Display.
- Binary (.tdf) – This format contains the encoded trace data as it comes out of the receiver and all of the information necessary to reopen the file with the Trace Display.

There are many options in the Save File dialog that allow you to save only the trace data that is important to you rather than saving all of the trace data, which can become quite sizeable.

When saving in binary format, any Trace Display options you have set at the time you save your file are preserved in the file and restored when you re-open the file. For instance, if you have your data filtered, the filter search criteria are restored when you re-open the file.

2. Save the file in binary format as tutorial.tdf.

   You can re-open this file using the Trace Display after finishing the Trace Tutorial to explore more of the Trace Display’s capabilities.

Using Trace System Options

Now that we have seen some of the features of the Trace Display, let us examine a few of the more important Trace System options.

- Using the Synchronize Trace With Target Execution Option
- Restarting and Resuming Trace Collection
- Choosing the Trace Buffer Type
- Changing the Trace System Receiver

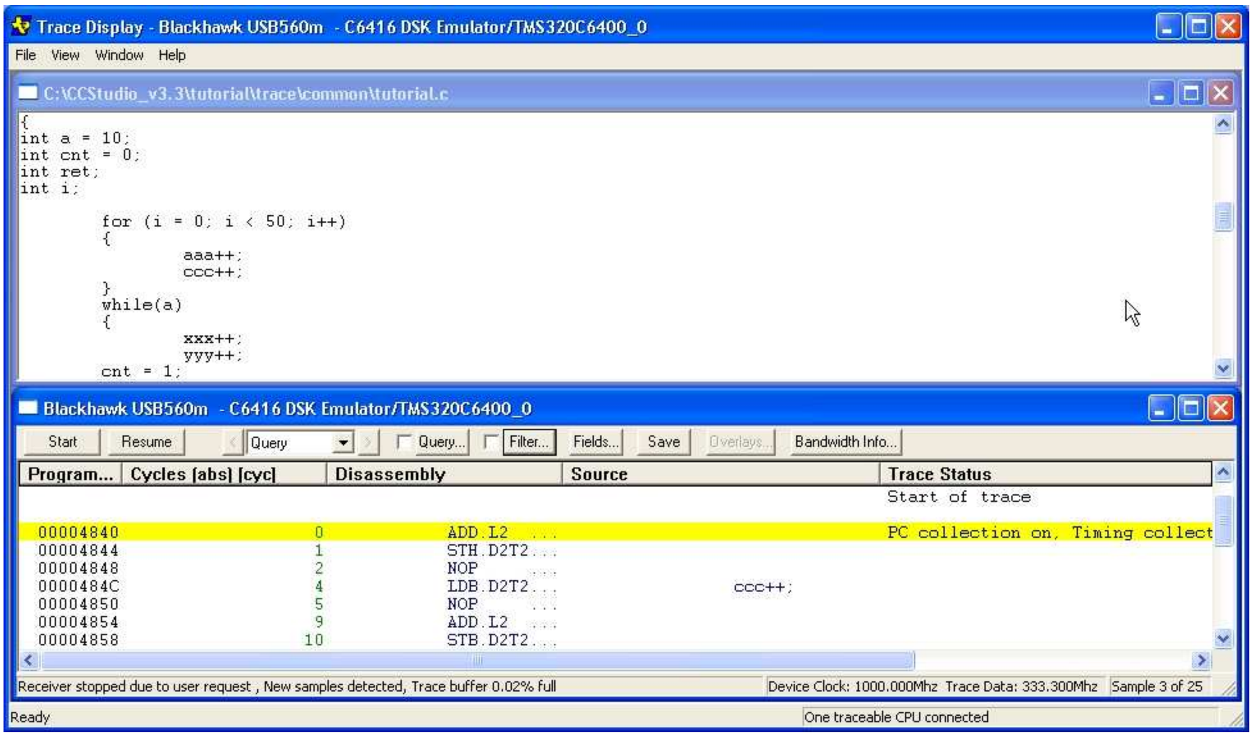
Using the Synchronize Trace With Target Execution Option

By default, when the target hits a breakpoint, the Trace System stops recording and the Trace Display appears automatically and displays data. This behavior is due to the default Synchronize Trace with Target Execution option. The exact columns that appear in the display depend upon the target device.

To see this option in action, select Debug → Run to run to the breakpoint again.

When the breakpoint is hit again, the Trace Display updates automatically, and contains all and only the code executed in the second iteration of the loop between “ccc++;” statements. This is because we have asked the Receiver to synchronize with target execution, and this option forces the Trace Receiver to automatically clear its buffer and restart collection when the target is run. As long as you keep hitting the Run button in Code Composer Studio, the Trace System will clear the Receiver’s buffer and record only what is between the loop iterations.

For example:


Trace Display - Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
File View Window Help
C:\CCStudio_v3.3\tutorial\trace\common\tutorial.c
Blackhawk USB560m - C6416 DSK Emulator/TMS320C6400_0
Start
Resume
Query
Query...
Filter...
Fields...
Save
Overlays...
Bandwidth Info...

```
{
int a = 10;
int cnt = 0;
int ret;
int i;

        for (i = 0; i < 50; i++)
        {
                aaa++;
                ccc++;
        }
        while(a)
        {
                xxx++;
                yyy++;
        cnt = 1;
```

Program...	Cycles [abs] [cyc]	Disassembly	Source	Trace Status
				Start of trace
00004840	0	ADD.L2 ...		PC collection on, Timing collect
00004844	1	STH.D2T2...		
00004848	2	NOP ...		
0000484C	4	LDB.D2T2...	ccc++;	
00004850	5	NOP ...		
00004854	9	ADD.L2 ...		
00004858	10	STB.D2T2...		

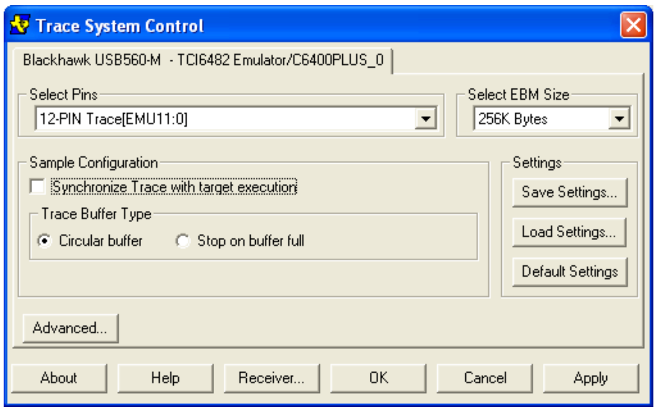
Receiver stopped due to user request , New samples detected, Trace buffer 0.02% full Device Clock: 1000.000Mhz Trace Data: 333.300Mhz Sample 3 of 25

Ready One traceable CPU connected

Restarting and Resuming Trace Collection

We can modify the default Trace behavior so that the Trace system appends data to the existing Receiver buffer rather than clearing it.

1. In Code Composer Studio, select Tools ® XDS560 Trace ® Control… to bring up the Trace Control window once again.
2. Turn off Synchronize Trace with Target Execution and click Apply.



Select Pins: 12-PIN Trace[EMU11:0]
Select EBM Size: 256K Bytes
Sample Configuration
☐ Synchronize Trace with target execution
Trace Buffer Type: [✓ Circular buffer] Stop on buffer full
Settings

3. When Trace is done reinitializing, click OK to dismiss the Trace Collection dialog.
4. Select Debug ® Run.
5. When the breakpoint is hit, go to the Trace Display and click on the Stop button.
6. The Trace Display contains just the one iteration of the loop.
7. Click the Resume button rather than the Start button.
8. Repeat the Run/Stop/Resume steps two or three more times.

   Each time the display contains more data, having appended each iteration of the loop because we have been using the Resume button rather than the Start button. Use Start when you want the Receiver to clear its buffer before you run the target again. Use the Resume button when you want to append to the existing Receiver buffer.

Choosing the Trace Buffer Type

There are two choices for the Trace Buffer Type in the Trace Control: Circular Buffer and Stop on Buffer Full. The default is Circular Buffer.

In Stop on Buffer Full, recording stops automatically when enough trace data is collected to fill the trace buffer. Recording can stop before the trace buffer is full if the target halts, an End Trace Action trigger occurs, or you hit the Stop button in the Trace Display.

In Circular Buffering, the buffer wraps and new trace data is stored starting at the beginning of the trace buffer when enough trace data is collected to fill the buffer, so that older trace data is continually overwritten. Recording does not stop until the target stops executing, an End Trace Action trigger occurs, or you hit the Stop button in the Trace Display.

In Circular Buffering, if the buffer has not wrapped and recording stops, the data in the buffer contains everything from the time recording started until recording stopped. This is equivalent to Stop on Buffer Full. However, in Circular Buffering, if the buffer has wrapped and recording stops, the data in the buffer no longer contains information from when recording started, but only data backwards in time from when recording stopped.

Thus, in Circular Buffering the buffer usually wraps, which allows you to look backward from the point you stopped recording, while Stop on Buffer Full allows you to look forward in time from the point you started recording.

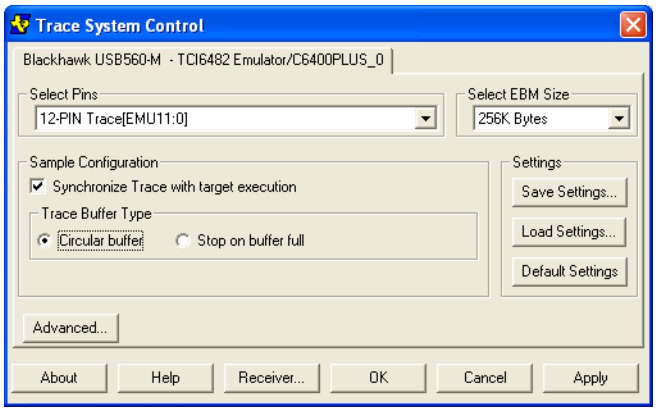
Circular Buffering would be good to use to determine the program flow that led to a problem at a known program location where the problem occurred. To perform this function, you would first program a Trace On action, set a breakpoint in your target application at the problem location, and then run your target application. When execution halts on the breakpoint, trace recording will be stopped. Trace Display will then show your target application actions from the time you hit the Stop button backwards in time, and allowing you to observe the problem path and discover the root cause of the problem.

Stop on Buffer Full would be good to use when your target application runs outside of the program image space defined in your .out/COFF file, soon after a certain function is called. In this case, you would set a breakpoint at that function, run your target application until it hits the breakpoint, then program a Trace On action and continue executing your target application from that breakpoint. When the trace buffer is full, recording will stop automatically, and Trace Display will show as much information as allowed by the trace data buffer capacity, starting from the breakpoint on. You can see exactly where your program flow started to run outside of the program image space because the Trace Status column will show the message *Bad PC*. If the trace data buffer does not have the capacity to capture the problem, you may use it to find a point further in the target application program flow to set another breakpoint so that you can capture the problem.

Changing the Trace System Receiver

Trace assumes you are using a TI XDS560T receiver. You can change the receiver by following these steps:

1. Bring up the Trace Control window by selecting the Tools ® XDS560 Trace ® Control… menu item.


Trace System Control

Blackhawk USB560-M - TCI6482 Emulator/C6400PLUS_0

Select Pins: 12-PIN Trace[EMU11:0]

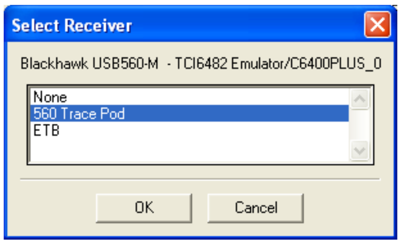
Select EBM Size: 256K Bytes

Sample Configuration

[✓] Synchronize Trace with target execution

Trace Buffer Type: [● Circular buffer] ○ Stop on buffer full

2. Click on the Receiver button. Highlight the desired receiver name and click OK.


Select Receiver

Blackhawk USB560-M - TCI6482 Emulator/C6400PLUS_0

None
[✓ 560 Trace Pod]
ETB

3. Click Apply in the Trace Control window. The Trace System presents a dialog and a progress indicator in the Code Composer Studio status bar while it is initializing the Trace System.

Summary of Trace Jobs

This tutorial used only one of the Trace actions, Trace On, and one type of trace, called Standard Trace, to demonstrate basic Trace System capability.

The other type of trace is called Event Trace, it allows you collect one of four categories of events:

- Memory event tracing records cache events, such as L1P Read Miss, L1D Write, and so on.
- Stall event tracing records the reason for the stall.
- Predication event tracing records whether an instruction executed (predicated true or false).
- External event tracing records events you program your target to generate. See the Interrupt Controller Spec for your target to learn how to program External Events. For example, for a C64x+ target, see the TMS320C64x+ DSP Megamodule Reference Guide (SPRU871E).

Note that the Trace On action starts recording trace immediately. All of the other Trace actions start recording when a trigger condition is true. The most frequently used trigger event is when the PC reaches a specified location or is within a specified range. Other triggers are possible, such as data memory accesses and events.

For more information about Event Trace and trace triggers, see the Trace System online help.

The following are other Trace actions with a brief description of what they do and some example use cases.

- Trace On – The Trace On action turns trace on immediately. This action is useful for when you first start debug and do not know where in your program flow a problem may have occurred. See Choosing the Trace Buffer Type for examples.

- Start Trace and End Trace – The Start and End Trace actions allow you to start and end trace at any program address location or within any program address range. The Start and End Trace actions also allow data memory triggers or event triggers. Thus, you can start or end collecting trace data whenever a write occurs to a data address range. Or you can start or end collecting trace data whenever a particular event occurs, for example a cache hit or miss, or an interrupt acknowledge. Start/End Trace actions follow the program beyond function call boundaries. You can have more than one Start/End trace action. As with Trace On actions, knowing approximately where the problem occurs helps assure that you can capture the problem within the trace data buffer you collect. Start/End trace is useful for debugging stack corruption problems since the captured program flow will show you where function returns do not return to the correct calling function.

- Trace In Range – The Trace In Range action allows you to specify a particular program address range to trace, for example, a particular function or range of functions contiguous in memory (such as ISRs). The Trace In Range action does not trace program flow outside of the program range start/end addresses. The Trace in Range action is useful when you can limit the problem to a specific program address range. This is particularly useful in debugging Interrupt Service Routines, which otherwise can not be debugged by setting breakpoints. Collecting Time Stamp information for ISRs is useful for determining if you are exceeding the time in which the interrupt must be serviced.

- Don't Trace In Range – The Don't Trace In Range action allows you to specify a particular program address range not to trace, such as a particular function or range of functions contiguous in memory (such as ISRs). The Don't Trace In Range action helps keep Interrupt Service Routine code from using space in the trace Receiver's buffer when the problem is known not to be in the ISR.

- End All Trace – The End All Trace action turns off all tracing. That is, the trigger for a Start Trace action will not cause Trace to start recording again after an End All Trace trigger has occurred. The End All Trace action is useful if you have more than one Start Trace action and wish to turn off all tracing at a particular program address location or range in your program flow. If you explicitly hit the Start button in the Trace Display or have the Synchronize Trace with Target Execution Trace Control option on, then recording will start again when the trigger for a Start Trace action occurs.

- Trace Variable – The Trace Variable action allows you to specify a variable name or absolute data address for which trace collection should occur. The Trace Variable action is useful when you know that a specific variable is being corrupted. Turning on the Collect PC w/Data option shows you the code that corrupted the data. It is important to note that you should not specify stack variables since their address is program-flow dependent. Another good example use of Trace Variable is to set a breakpoint beyond the initialization of a DSP data table and then add a Trace Variable action with the range of the data table to see how its values are changing. You can then export the data to a spreadsheet for graphing.

- Store Sample – The Store Sample action records trace data for the current cycle whenever the trigger for the action is true. The Store Sample action is useful since it allows data memory, or event triggers, which some other Trace action do not. Thus, you can collect trace data whenever a write occurs to a data address range, not just one address as in the Trace Variable action. Or you can collect trace data whenever a particular event occurs, for example a cache hit or miss, or an interrupt acknowledge.

- Don't Store Sample - The Don't Store Sample action prevents recording trace data for the current cycle whenever the trigger for the action is true. The Don't Store Sample action takes precedence over the Store Sample action. The general use of the Don't Store Sample action would be to prevent Store Sample actions from taking place under certain trigger conditions, such as eliminating a sub-range of a data address range (e.g., do not record data for accesses to array elements 6-10).

- Insert Trace Marker – The Insert Trace Marker action records a special marker within the trace data whenever the trigger for the action is true. This marker is identified in the Trace Status column of the Trace Display with the message Trace Marker. Like the Store Sample action, the Insert Trace Marker action allows data memory, or event triggers. Thus, the Insert Trace Marker action can be used as an indication that you have reached a particular location in your program flow, that a write access to a particular data memory range has occurred, or a specific event has occurred, such as a cache hit or miss, a CPU stall, and so on.

All Trace actions allow you to specify what trace information to collect: Program Address, Write Address, Read Address, and so on. The more types of data you ask to collect, the more likely you will exceed the ability of the Trace Hardware to collect and transmit all of the data off the chip. You can prevent this by choosing the option in the Trace Control Advanced Settings… dialog to stall the CPU to allow the Trace Hardware time to export all data. Be aware that stalling the CPU has an impact on the real time aspect of your program.

There are many other Trace System options in both the Trace Control and the Trace Display which are meant to help you debug and optimize your target code. Please explore these options and use the online help, which explains the options and their potential uses. For example, the Event Sequencer includes trace actions that, combined with the AET state machine HW capabilities, provide you with more precise control over trace collection in complex program flow situations.