

Signal File Viewer & Analyzer: Debugging signal processing codes for STM32 Series MCU

Application Note

Contents

1. Introduction.....	3
2. STM32 Series Debugging case with SFVA	4
2.1. Check dumped memory utilizing SFVA	4
2.2. Summary.....	4
2.3. View graphs	5
3. How to export memory.....	7
3.1. Example of STM32 Cube IDE.....	7
3.2. Example of IAR Work Bench IDE.....	9
4. Ordering information.....	12
5. GA Technologies, Inc.....	12

1. Introduction

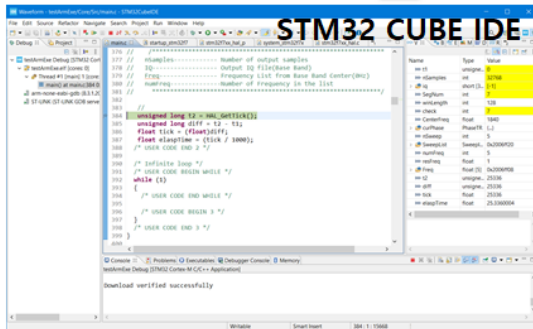
This application note explains example of utilizing Signal File Viewer & Analyzer (SFVA) for debugging signal processing code for MCU board.

After you set Break Point on the part of the code that you want to see if it works normally, you can memory dump the intermediate data in that code. For short-length data arrays, IDE can show those values immediately, but the longer they are, the harder it is to verify that the data generated by running of the code is processed as intended. In this case, you can save the data as a file utilizing memory dump functionality of IDE and visually check the data utilizing SFVA to see exactly whether the code is functioning normally.

SFVA is designed and developed to make it easy to visually view different types of data and even large data files in gigabytes.

Debugging DSP code utilizing Signal File Viewer & Analyzer

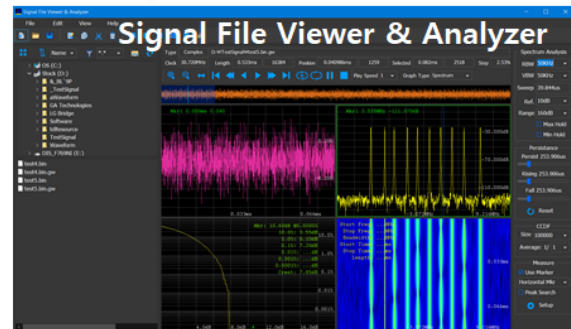
- Set breakpoint
- Run debugging
- Memory dump



- Edit the code



- Load the data file



- Check the signal file visually with graphs



*Memory dump: Functionality to save a data array to a file during debugging code

2. STM32 Series Debugging case with SFVA

2.1. Check dumped memory utilizing SFVA

The Import feature of SFVA allows you to quickly view various graphs of dumped binary file.

During debugging, memory dump data can be visualized with SFVA in different kinds of graphs to determine whether the signal processing code being written behaves as designed or maintains the precision of the processed values.

2.2. Summary

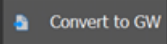
First Run Procedure

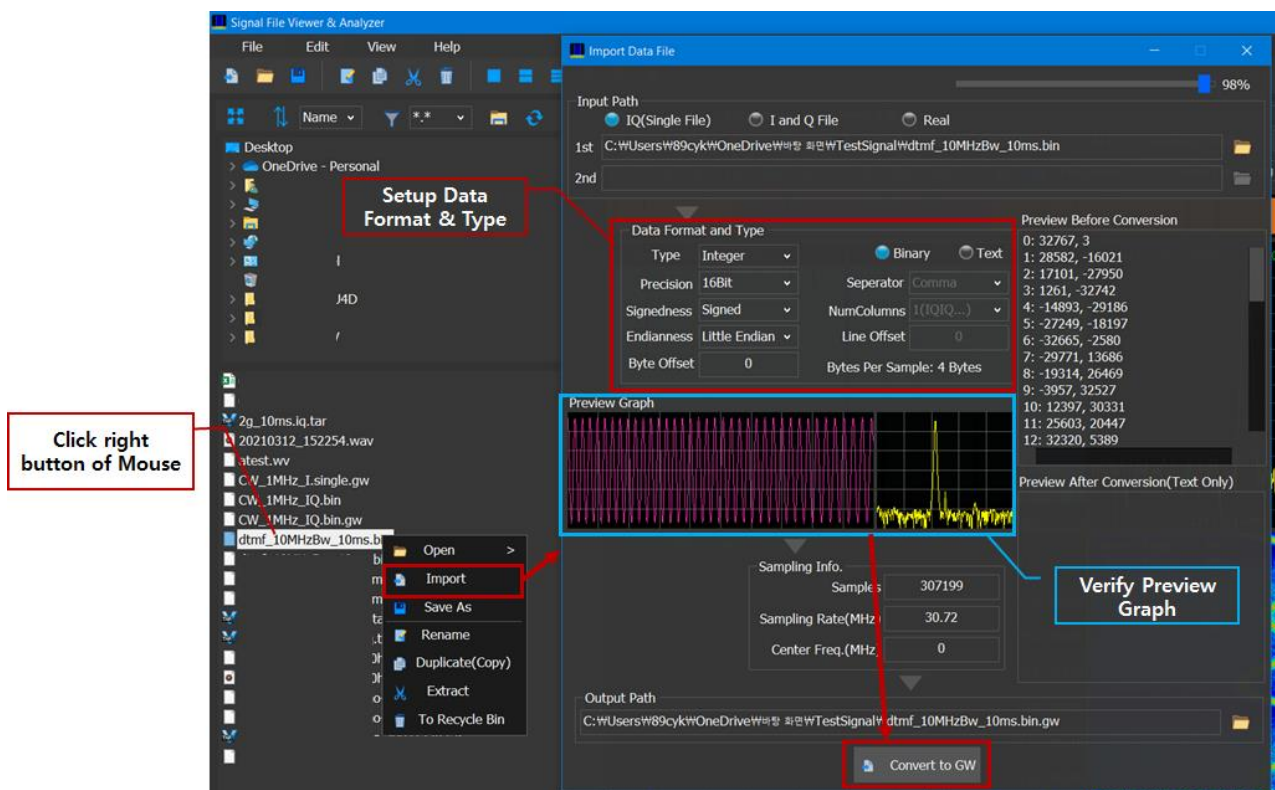
Step 1. Right-click the dumped file in File Manager as shown below

Step 2. Select "Import" when popup menu appears

Step 3. Set "Data Format and Type" when the "Import Data File" dialog opens

Step 4. Check Preview graph to see if the settings are appropriate

Step 5.  Pressing the button automatically converts the file and displays it as a graph

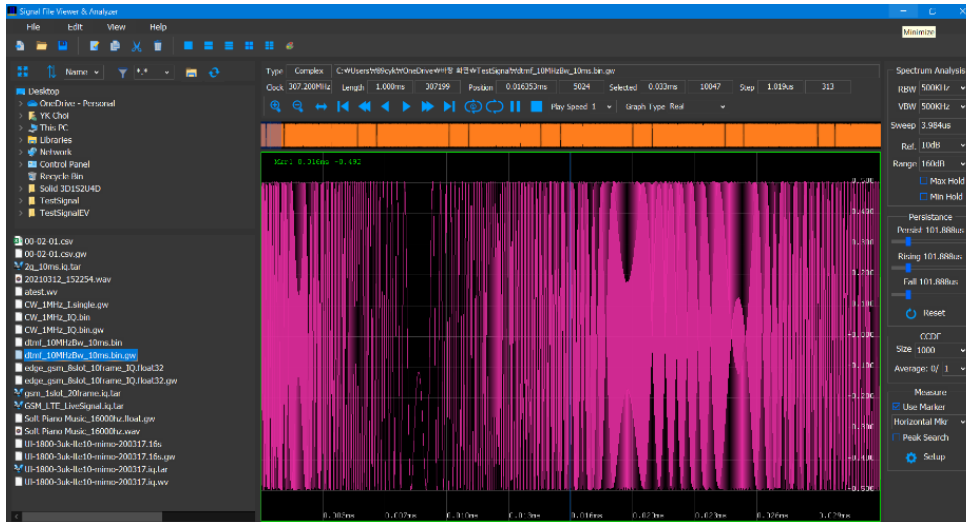


If the data is displayed properly when you first run "Import" as shown above, the settings in the


"Import Data File" dialog will retain the previous settings even if you close this dialog before the SFVA exits, so you can check the graph immediately by pressing the "Convert to GW" button without the need for the "Data Format and Type" setting.

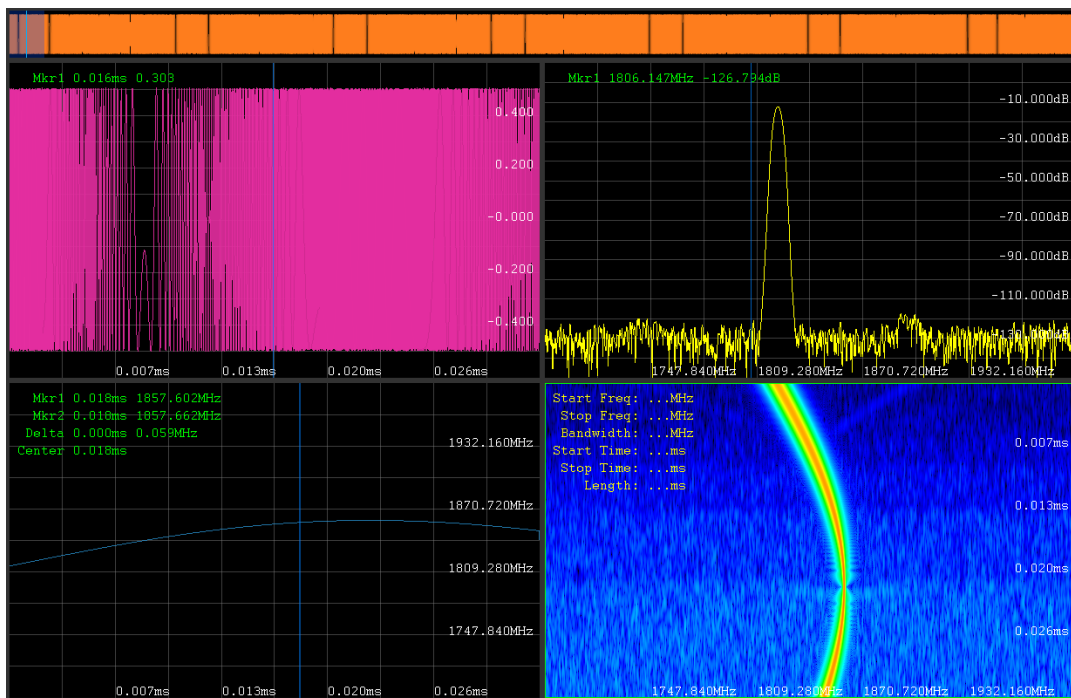
2.3.View graphs

Step 1. When you click "Convert to GW" button, the data is converted to *.gw, the default file type for SFVA, and the graph displays as shown below. The figure below shows View Mode in Graph1x1.



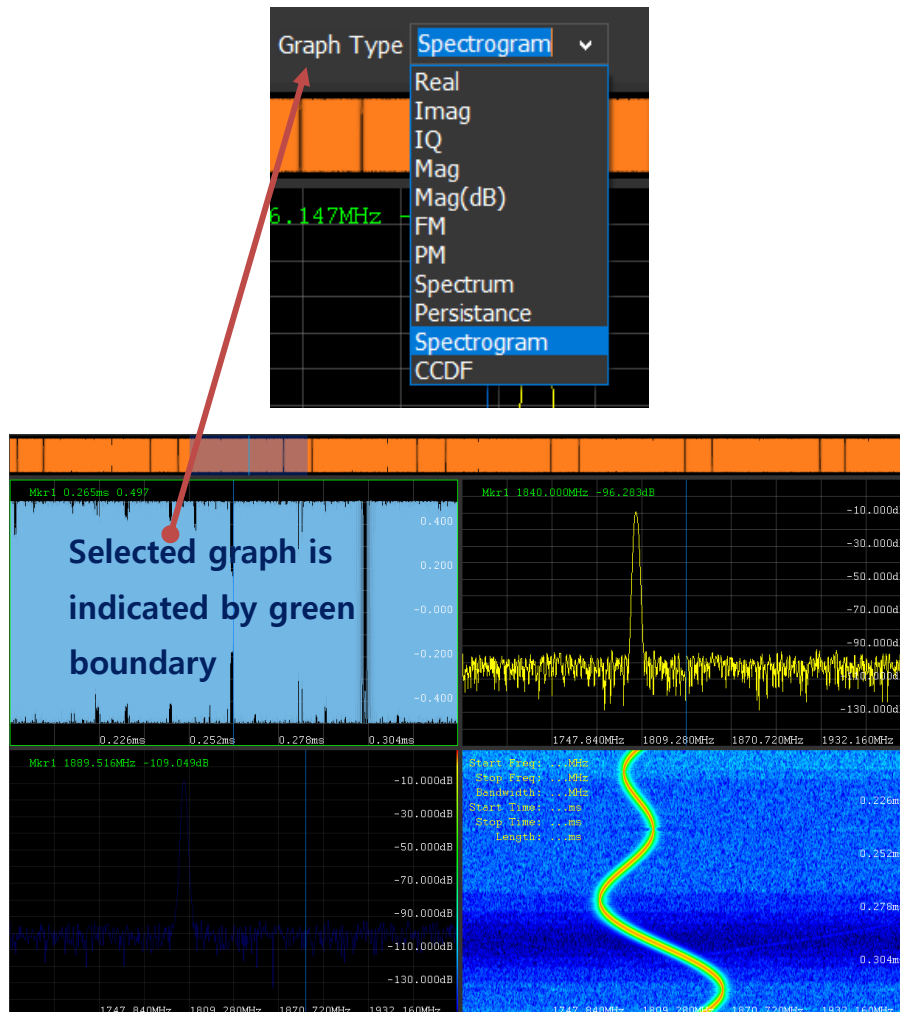
Step 2. View Mode selection displays multiple graphs on one screen.

Selecting View Mode as Graph 2x2 (using View/Graph 2x2 or shortcut icon ) changes the graph display as shown below. This allows you to see 4 different kinds of graphs at once.

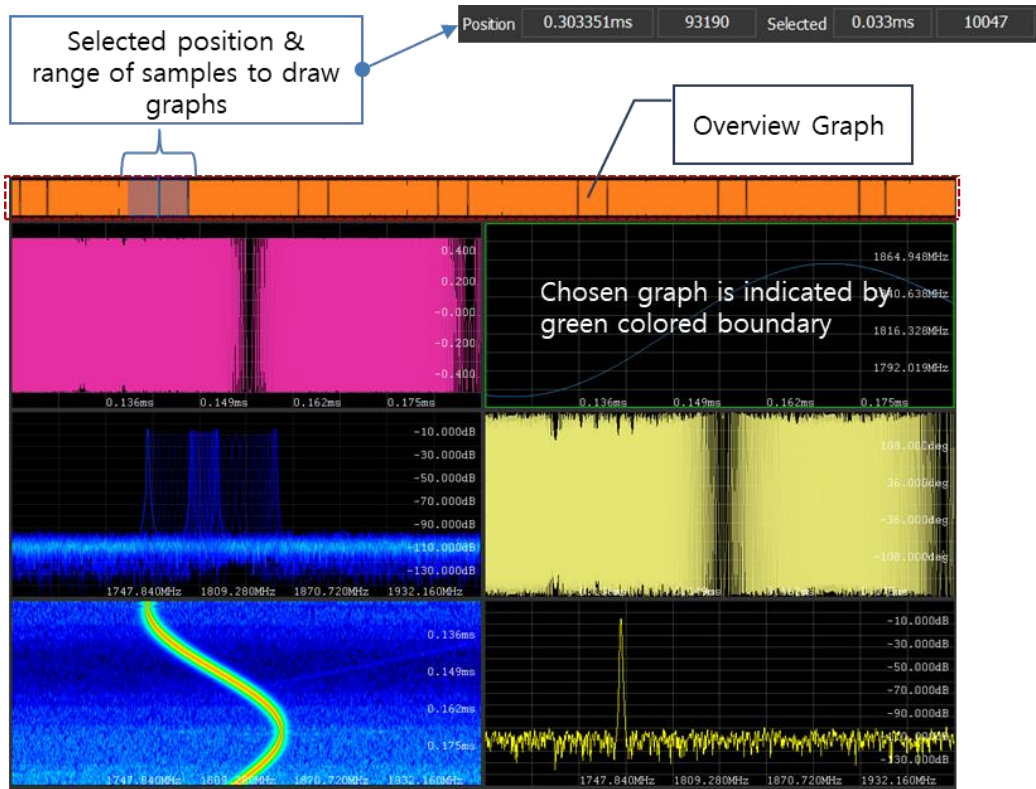


Step 3. Select graph type

If you click the graph you want to change in the above four graphs, and then select the graph type in Graph Type combo box at the top, the type of graph selected changes. This allows you to select all kind of graph is available for the selected graph, and memory dump data can be quickly viewed through various time domain and frequency domain graphs.

**Step 4.** Select a specific location on a data file, zoom in, or play to view the data

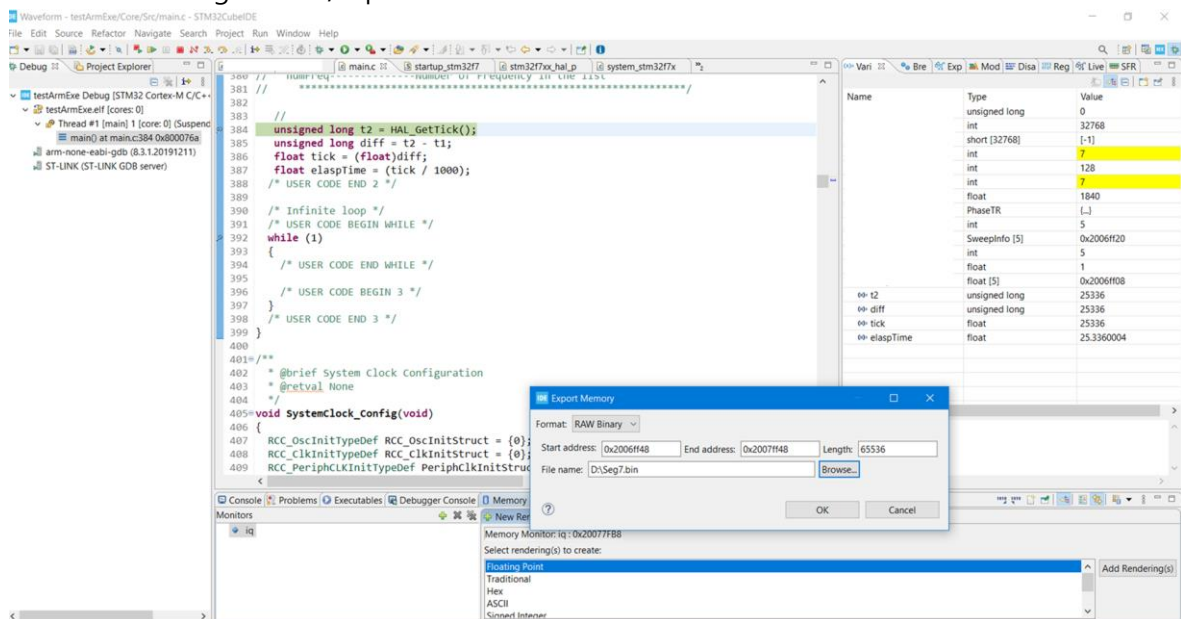
Overview graph in the figure below shows the full outline of the input dump data file and allows you to select a specific location in the currently open file and adjust the range you draw or zoom in using Mouse. Also, for time domain graphs, you can click the graph and use mouse wheel to zoom in around the clicked portion.



3. How to export memory

3.1. Example of STM32 Cube IDE

During debugging, as shown in the figure below, you can see variables in the code, and you can stop running of the code with break point set up at a line of the code, and then save the intermediate result data as a file using the File/Export feature.



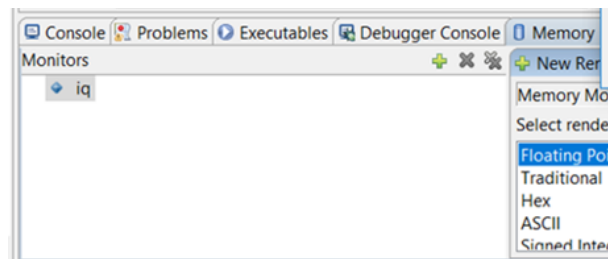
Step 1. After setting break point at a line of the code, run debugging(F11)

```

382
383 //
384 unsigned long t2 = HAL_GetTick();
385 unsigned long diff = t2 - t1;
386 float tick = (float)diff;
387 float elaspTime = (tick / 1000);
388 /* USER CODE END 2 */

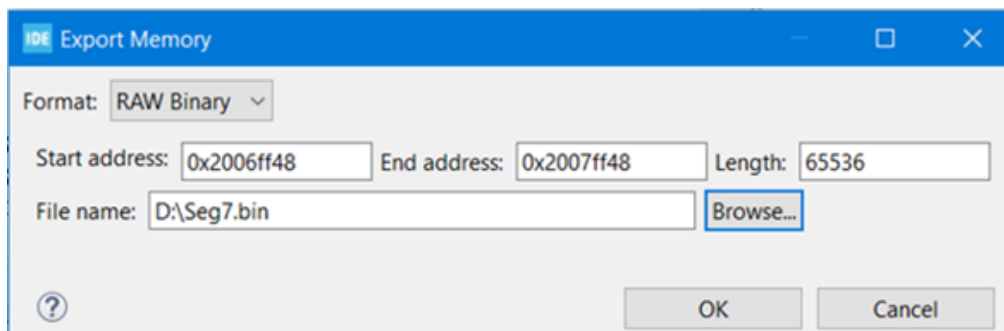
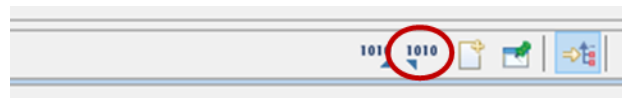
```

Step 2. After executing the code, running stops at break point.



In the Memory tab at the bottom, enter name of the data that you want to save using the "+" button in the Monitors entry.

Step 3. Select the File/Export menu or click the located button on the Memory tab to open the Export dialog, as shown below.



Step 4. Select Format as "RAW Binary", and then set address, length, and browse the path where the file is saved. Set the address referring to "Variable Monitor" Table on the right. The address should be accurate to prevent data loss.

Step 5. After checking out the addresses and length, click OK button to start saving data to file in the memory.

3.2.Example of IAR Work Bench IDE

Step 1. Create a *.mac file with the following:

Copy and paste the following on the Note Pad and save it as "DumpMem.mac".

/Macro Code****

// Example of command to enter in QuickWatch command line:

// DumpMem("dumpmem.bin", &trace_buffer, sizeof(trace_buffer))

DumpMem(aFileName, aAddress, aSize)

{

 __var fileHandle;

 __var memoryByte;

 __message "--- macro DumpMem(file: ", aFileName, ", address: 0x", aAddress:%X, ", size: ", aSize:%d, ");

 fileHandle = __openFile(aFileName, "wb");

 if (fileHandle)

 {

 while ((int)aSize > 0)

 {

 memoryByte = __readMemory8((unsigned int)aAddress, "Memory");

 __writeFileByte(fileHandle, memoryByte);

 aAddress = (int)aAddress + 1;

 aSize = (int)aSize - 1;

 }

 __closeFile(fileHandle);

 }

 else

 {

 __message "*** Could not open file ", aFileName, "\n";

 }

 __message "--- macro Done";

}

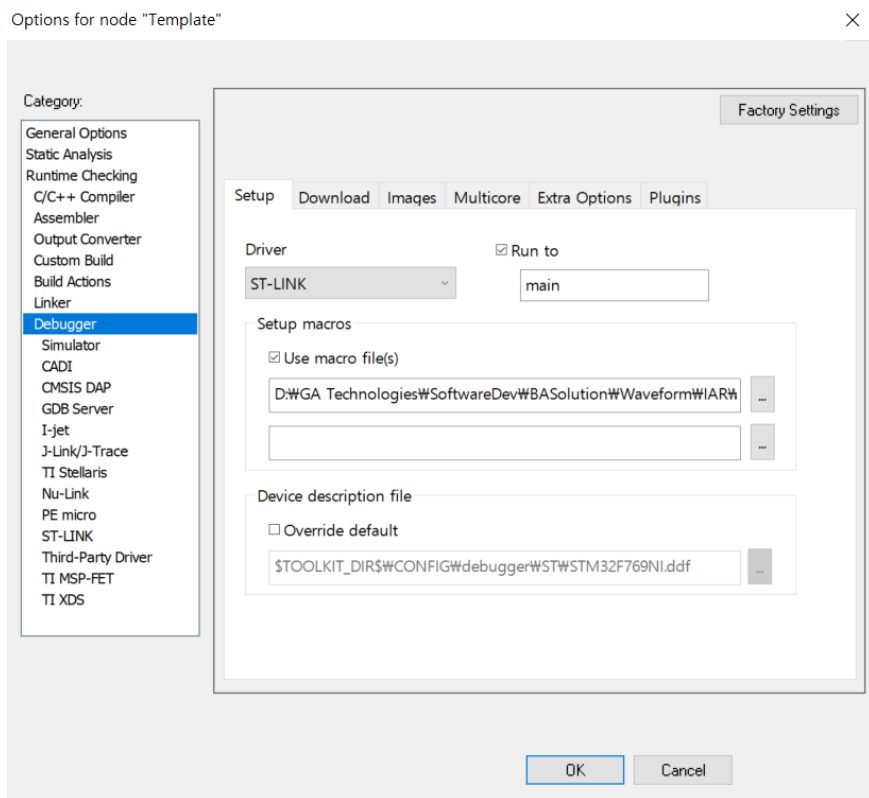
/End of Macro**

Step 2. Set the following so that you can use macro above.

Select Project/Options/Debugger

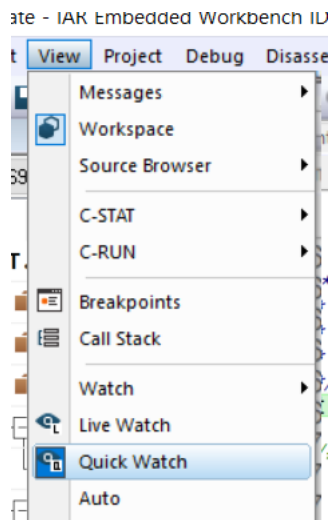
"Setup macros": check "Use macro file(s)" box

Browse and add file path of "DumpMem.mac" file as below figure



Step 3. During debugging, break point is used to stop execution at a specific code location and then store the intermediate result as shown below.

Step 4. Select View/Quick Watch



Enter macro commands at the top of Quick Watch

Usage: DumpMem("Path and file name", variable name, size of the array);

Following is an example command.

```
DumpMem("D:\\wofdm2_100MHz_CheckFlt.bin", iq, sizeof(short)*16384)
```



4. Ordering information

For more information on how to use the SFVA, see Online Help in the link below.

<https://www.ga-technologies.com/onlinhelp>

You can learn more about SFVA through the link below.

<https://www.ga-technologies.com/signal-file-viewer-analyzer>

If necessary, you can download it via the link below and install, and evaluate the software for 30days with trial license before purchase.

<https://www.ga-technologies.com/download>

SFVA can be purchased online from our website.

<https://www.ga-technologies.com/purchase>

5. GA Technologies, Inc.

We supply software and hardware for testing and measuring mobile communications and wireless communication equipment. We have long experience and know-how in the application of RF Vector Signal Generator, the necessary waveform generation and signal analysis. Please feel free to contact us as we can assist you in the above-mentioned test and measurement areas.

Tel: +82-31-825-3866

Email: sales@ga-technologies.com

Home page: www.ga-technologies.com