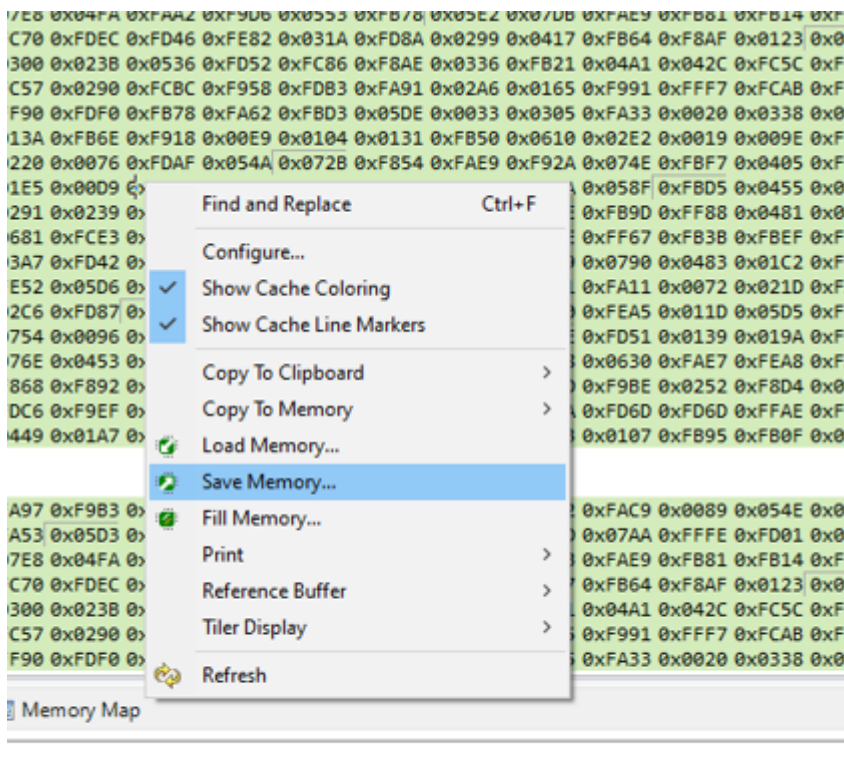
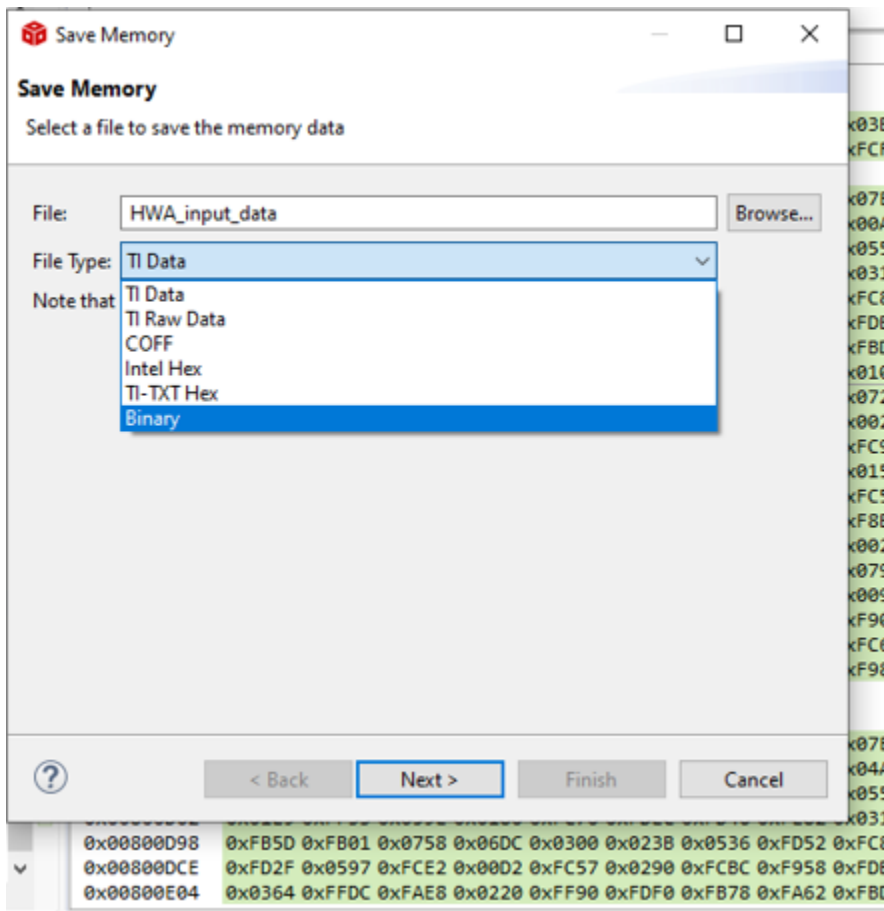


## Save Data from Memory to a binary file using CCS Memory Save/Load

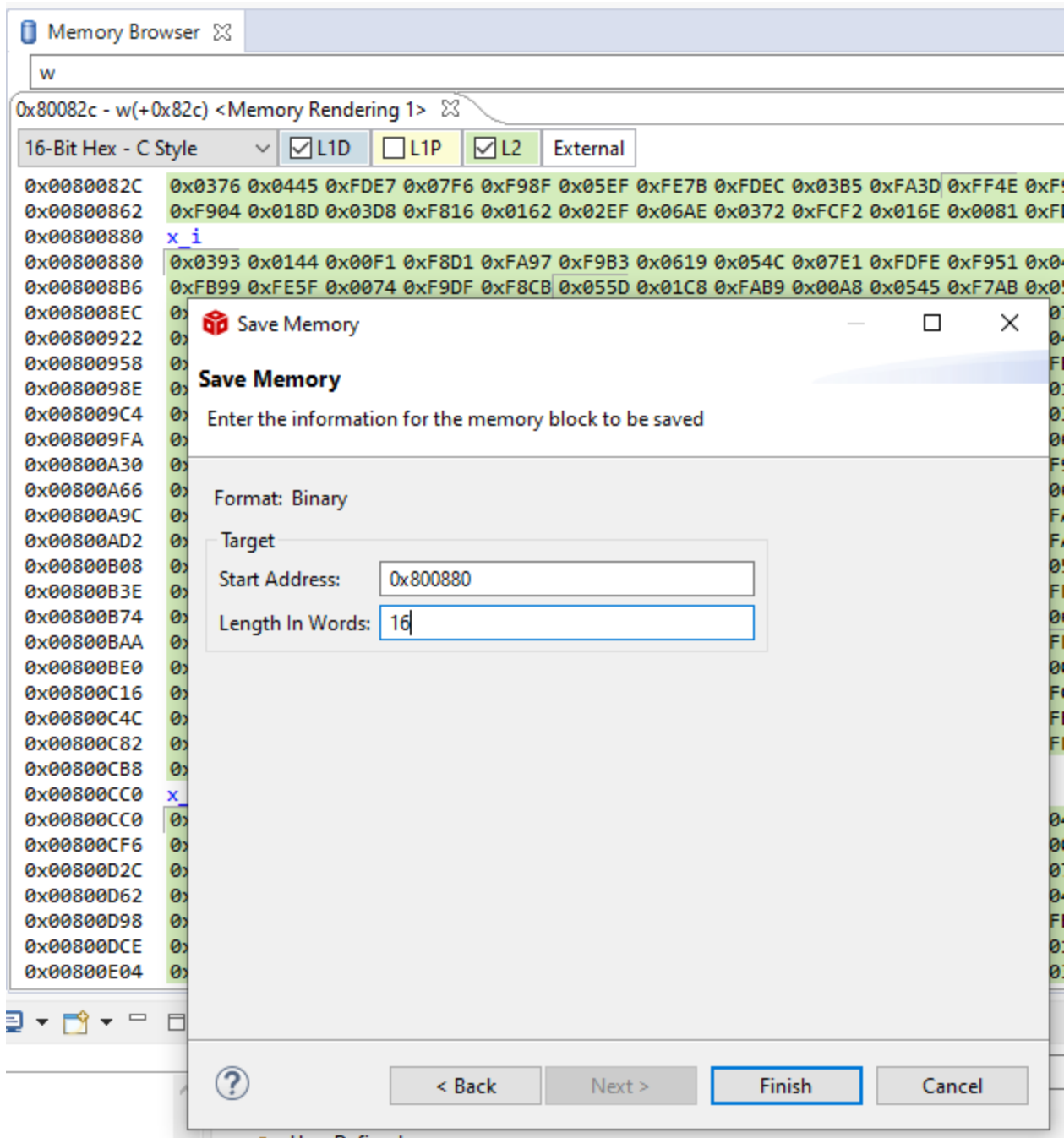
- Connect EVM to CCS
  - Make sure that following file is flashed on the 1843BOOST EVM:  
"C:\ti\mmwave\_sdk\_03\_05\_00\_04\packages\ti\utils\ccsdebug\xwr18xx\_ccsdebug.bin"
- Load the executable using CCS (we load to DSS or MSS core depending what we decided)
- Open Memory
- Do what is needed to make sure that the memory section we want to save to binary file is updated. For example:
  - Set Breakpoint
  - Run Code
- In "CCS Debug" Open "View-> Memory Browser"
  - One can type the address of memory
  - One can also type the name of the array if it is a global variable
- What data format should we use?
  -
- Right Click in the Memory Browser, Select Save Memory.



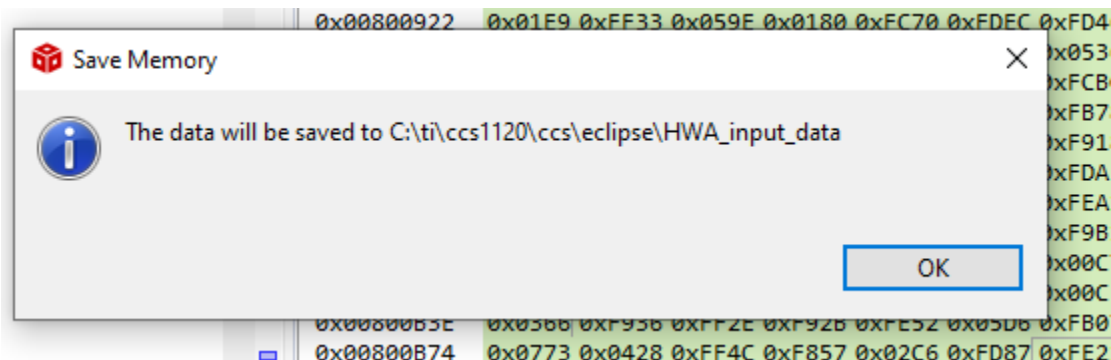
- Provide File Name
- Select File Type: Binary
- Select: Next



- Define
  - Start Address
  - Length in Words: **IMPORTANT – A Word is 4 bytes – 32bit**
  - Select Finish



- The file will be saved to following location



- Use a Hex editor to open the binary file. Here is what the content looks like. Compare to the view in the CCS Memory Browser (for 16-bit and 8-bit)

The image shows a screenshot of the HxD hex editor. The main window displays a binary file named 'HWA\_input\_data' at the path 'C:\ti\ccs1120\ccs\eclipse\HWA\_input\_data'. The editor shows a hex dump with columns for Offset (h), hex values, and Decoded text. The decoded text appears to be a mix of Latin and Cyrillic characters.

Below the main window, there are two smaller windows showing memory rendering. The first window is titled '0x800874 - w(+0x874) <Memory Rendering 1>' and shows a 16-bit hex view. The second window is titled '0x800874 - w(+0x874) <Memory Rendering 1>' and shows an 8-bit hex view. Both windows have checkboxes for L1D, L1P, L2, and External, and a dropdown for '16-Bit Hex - C Style' and '8-Bit Hex - C Style' respectively.

**Hex Editor Content:**

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	B3	03	44	01	F1	00	D1	F8	97	FA	B3	F9	19	06	4C	05	.D.ñ.Ñ-ú'ù..L.
00000010	E1	07	FE	FD	51	F9	92	04	C9	FA	89	00	4E	05	08	00	ä.pýQù'.Éú%.N...
00000020	70	02	A1	03	9F	01	F1	00	2E	FD	ED	00	67	04	8F	FE	p.i.ÿ.ñ..ýi.g..p
00000030	FF	01	08	00	E4	01	99	FB	5F	FE	74	00	DF	F9	CB	F8	ÿ...ä.'ü_pt.äüËø

**16-Bit Hex - C Style Memory Rendering:**

Address	Value
0x00800874	0x016E 0x0081 0xFD21 0x0094 0xF8BD 0xFDF6
0x00800880	x_i
0x00800880	0x0393 0x0144 0x00F1 0xF8D1 0xFA97 0xF9B3 0x0619 0x054C
0x00800890	0x07E1 0xFDFE 0xF951 0x0492 0xFAC9 0x0089 0x054E 0x0008
0x008008A0	0x0270 0x03A1 0x019F 0x00F1 0xFD2E 0x00ED 0x0467 0xFE8F
0x008008B0	0x01FF 0x0008 0x01E4 0xFB99 0xFE5F 0x0074 0xF9DF 0xF8CB

**8-Bit Hex - C Style Memory Rendering:**

Address	Value
0x00800874	0x6E 0x01 0x81 0x00 0x21 0xFD 0x94 0x00 0xBD 0xF8 0xF6 0xFD
0x00800880	x_i
0x00800880	0x93 0x03 0x44 0x01 0xF1 0x00 0xD1 0xF8 0x97 0xFA 0xB3 0xF9 0x19 0x06 0x4C 0x05
0x00800890	0xE1 0x07 0xFE 0xFD 0x51 0xF9 0x92 0x04 0xC9 0xFA 0x89 0x00 0x4E 0x05 0x08 0x00
0x008008A0	0x70 0x02 0xA1 0x03 0x9F 0x01 0xF1 0x00 0x2E 0xFD 0xED 0x00 0x67 0x04 0x8F 0xFE
0x008008B0	0xFF 0x01 0x08 0x00 0xE4 0x01 0x99 0xFB 0x5F 0xFE 0x74 0x00 0xDF 0xF9 0xCB 0xF8

### Step 7: Using Matlab to Check that the HWA functionality is correct

- Here is some sample matlab code to read a bin file

```
dirName = 'C:\Data\Data_from_Dave\';
fileName = 'hwa_ping_output.dat';
filePath = strcat(dirName, fileName);
fid = fopen(filePath, 'rb');
I = fread(fid, 'int16');
sig_cplx = I(1:2:end)+j*I(2:2:end);
```

- Run this code in matlab
- Make sure that the data is similar to the data in CCS Memory Browser