Saving data from the Keil µVision5 debugger to a file

Example: I wish to save the contents of the following array of 16 8-bit data values.

```
char keys[] = { 1,  4,  7, 14,      //Column 1/PB4, Rows 0-3
                2,  5,  8,  0,      //Column 2/PB5, Rows 0-3
                3,  6,  9, 15,      //Column 3/PB6, Rows 0-3
                10, 11, 12, 13    //Column 4/PB7, Rows 0-3
               };
```

Method 1:

Display the values in the command window while logging them to a file. In the command window, create a log file and "dump" the contents of the array or memory block:

```
>log > keys.log            -- create "log file"  keys.log
>d &keys[0], &keys[15]     --"dump" data between the two specified addresses
>0x20000008  01 04 07 0E 02 05 08 00 - 03 06 09 0F 0A 0B 0C 0D ................
>log off                   -- stop logging data
```

Contents of saved file *keys.log*:

```
0x20000008      01 04 07 0E 02 05 08 00 - 03 06 09 0F 0A 0B 0C 0D ................
Start Address  Data                                                equivalent "characters"
```

Method 2:

Write the values in any desired format to a file via a user-defined function in the debugger. The function can be invoked from the µVision command line or from a button in the toolbox. Define a function in the µVision Function Editor (see Figure 1), which is opened from the Debug menu (***Debug > Function Editor).*** Enter the function, compile it, and save it in an "initialization file" (eg. *displayvalues.ini*). The function in Figure 1 will create a log file, print the contents of array *keys[]* to the file, using the format in the *printf* statement, and then close the log file.

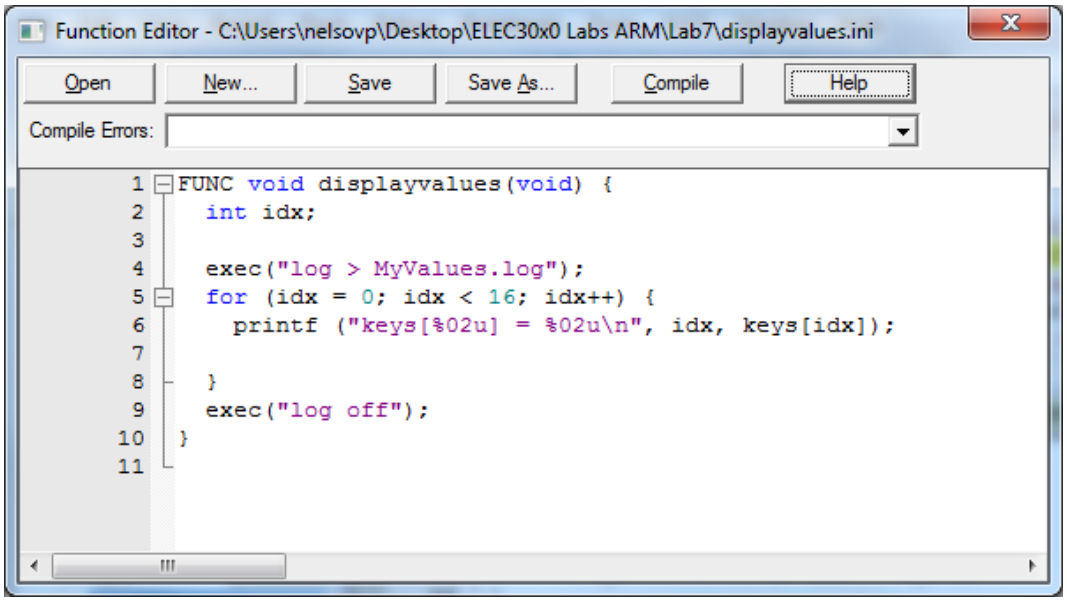

Figure 1. User-defined function to print the contents of array *keys[]* to log file *MyValues.log*.

The function is loaded into the debugger in one of two ways.

1. Specify the file in the field **Options for Target — Debug — Initialization File**. The file is loaded and processed each time a debugging session is initiated.
2. Use the INCLUDE command during a debugging session to read and process the file.

```
>INCLUDE displayvalues.ini
```

Debug functions defined in this file are made available for use during the debugging process.

While the simulator is stopped, the function can be invoked in the µVision command line:

```
displayvalues()   // function invocation
```

or you can define a button in the toolbox to start the function:

```
define button "Log Array", "displayvalues()"
```

Contents of the saved Log File "MyValues.log":

```
keys[00] = 01
keys[01] = 04
keys[02] = 07
keys[03] = 14
keys[04] = 02
keys[05] = 05
keys[06] = 08
keys[07] = 00
keys[08] = 03
keys[09] = 06
keys[10] = 09
keys[11] = 15
keys[12] = 10
keys[13] = 11
keys[14] = 12
keys[15] = 13
```

For more efficient post-processing of the data in the saved file, the formatting in the printf statement can be changed, along with the file extension, to essentially make what you export a **.csv** (comma-separated values) file, which can be directly imported into Excel or MATLAB

Method 3:

Enter the "save" command in the command window to save addresses and data to an "Intel hex" file, from which the data can be extracted.

```
Save keys.hex &keys[0], &keys[15]   -- write data between the two addresses
```

Saved file "keys.hex":

```
:02000004 2000DA                  --Extended address of start of data area (0x20000000)
:08000800 0104070E02050800C7      - First 8 bytes, beginning at address 0008 of data area
:08001000 0306090F0A0B0C0D99      - Next 8 bytes, beginning at address 0010 of data area
:00000001FF                       --Conclusion of data
```

Intel Hex Format:

2-digit byte count | 4-digit address | 2-digit record type (00=data) | data bytes | 2-digit checksum