International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:9, 2008

Practical Guidelines and Examples for the Users of the TMS320C6713 DSK

Abdullah A Wardak

Abstract**—**This paper describes how the correct endian mode of the TMS320C6713 DSK board can be identified. It also explains how the TMS320C6713 DSK board can be used in the little endian and in the big endian modes for assembly language programming in particular and for signal processing in general. Similarly, it discusses how crucially important it is for a user of the TMS320C6713 DSK board to identify the mode of operation and then use it correctly during the development stages of the assembly language programming; otherwise, it will cause unnecessary confusion and erroneous results as far as storing data into the memory and loading data from the memory is concerned. Furthermore, it highlights and strongly recommends to the users of the TMS320C6713 DSK board to be aware of the availability and importance of various display options in the Code Composer Studio (CCS) for correctly interpreting and displaying the desired data in the memory. The information presented in this paper will be of great importance and interest to those practitioners and developers who wants to use the TMS320C6713 DSK board for assembly language programming as well as input-output signal processing manipulations. Finally, examples that clearly illustrate the concept are presented.

Keywords**—**Assembly language programming, big endian mode, little endian mode, signal processing.

I. Introduction

THERE are two different architectures for handling memory storage [1]. They are referred to as little endian and big endian and they are related to the order in which the bytes are stored in the memory [1]. It is worth mentioning that the majority of the single-board computers such as MC600, MC68000, MC68020, and TMS320C30 EVM operate in big endian mode by default. However, the TMS320C6713 DSP Starter Kit (DSK) board can operate in little endian as well as in big endian modes, provided the necessary steps are implemented.

The two terms such as little endian and big endian are taken from "little end in" and "big end in" respectively and they refer to the way in which data is stored in the memory [1]. In little endian mode, the little end of the data is stored first, which means that a number like 0x6B8A is stored in the memory as (0x8A, 0x6B). The little end, or lower end, of the data is stored first (i.e. 0x8A). The same applies to a four-byte number; for example, a number like 0xABCDEF12 would be stored as (0x12, 0xEF, 0xCD, 0xAB). In other words, in little endian mode, the lower memory addresses contain the least significant byte of the data. However, in big endian mode, the big end of the data is stored first, which means that a number like 0xABCD is stored in the memory as (0xAB, 0xCD). The big end, or upper end, of the data is stored first. The same is true for a four-byte number; for instance, a number like 0xABCDEF12 would be stored as (0xAB, 0xCD, 0xEF, 0x12). In other words, in big endian mode, the lower memory addresses contain the most significant byte of the data [1].

It is crucially important for a user of the TMS320C6713 DSK board to check its endianness before embarking on assembly language programming. Clear understanding of the two operating modes of the DSK board can save a tremendous amount of time during the development stage of the application software.

II. The TMS320C6713 DSK Board

Digital signal processors such as the TMS320C6x family of processors are like fast special-purpose microprocessors with a specialized type of architecture and instruction sets suitable for signal processing [2]. The TMS320C6713 DSK board is powerful and relatively cheap, having the necessary supporting tools for real-time signal processing [2]-[11]. It includes the TMS320C6713 floating-point digital signal processor and a 32-bit stereo codec TLV320AIC23 (AIC23) for input and output (see Fig.1). The onboard codec AIC23 uses a sigma–delta technology that provides A/D and D/A [2]. The DSK board operates at 225MHz and it can be easily set to variable sampling rates of 8 to 96 kHz [2]. Four connectors on the DSK board provide input and output. They are: LINE IN for line input, MIC IN for microphone input, LINE OUT for line output, and HEADPHONE for a headphone output. Note that LINE OUT is multiplexed with HEADPHONE. The status of the four user dip switches on the DSK board can be read, which provides the user with a feedback control interface. For more information regarding the TMS320C6713 DSK board, refer to references [12], [13], [18]-[20].

Abdullah Wardak is currently a senior lecturer in Southampton Solent University, School of Computing and Communications, Faculty of Technology, East Park Terrace, Southampton SO14 OYN, UK (phone: 0044(0)2380319213, fax: 0044(0)2380334441, e-mail: Abdullah.wardak@solent.ac.uk).

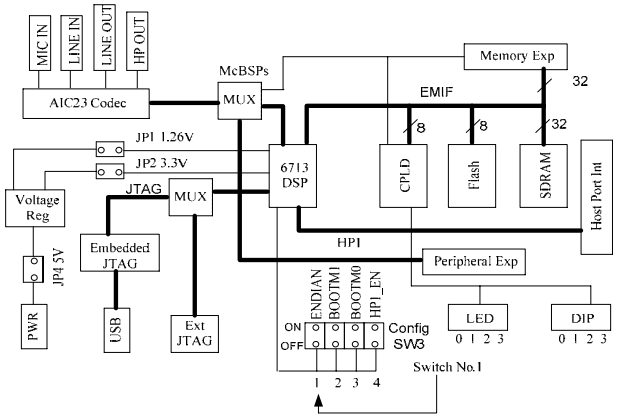


Fig. 1 Block diagram of the TMS320C6713 DSK board [12]

III. Endianness of the TMS320C6713 DSK Board

The TMS320C6713 DSK Board can operate in the little endian as well as in the big endian mode. However, it is set up by default in the little endian mode [12], [13]. A user can change its mode from the little endian to the big endian mode and vice versa by implementing the necessary steps. Anyone who wants to use the TMS320C6713 DSK board for assembly language programming and signal processing needs to identify the endian mode of the board first; otherwise, this will cause unnecessary confusion and erroneous results.

A. To Set the TMS320C6713 DSK Board in the Little Endian Mode

The TMS320C6713 DSK board operates in little endian mode by default [13]. To set the TMS320C6713 DSK board in the little endian mode, all **FOUR** of the following conditions must be satisfied.

1. Switch 1 in the Config SW3 as shown in Fig.1 must be placed in the **OFF** position [12], [14], [15].

2. After setting switch No.1 in the OFF position, **EN** in the Code Composer Studio must indicate 1. This is read only register and indicates the mode of the DSK board. When EN=1, the DSK board operates in the little endian mode and when EN=0, the DSK board operates in the big endian mode. The value of EN can be displayed in the CCS by **Clicking** View → **Clicking** Register → **Clicking** Core Registers as shown in Fig.2.

3. The endianness of the TMS320C6713 DSK board must be set to Little Endian as shown in Fig.2. This can be done by **Clicking** Project → **Click** Build Options → **Highlight** Advanced and make sure the **Endianness** is set to **Little Endian** and press **OK.**

4. Add the library file: **rts6700.lib** for assembly language programming and add the library files: **rts6700.lib, csl6713.lib, dsk6713bsl.lib** for signal processing (i.e. input and output signal manipulation).

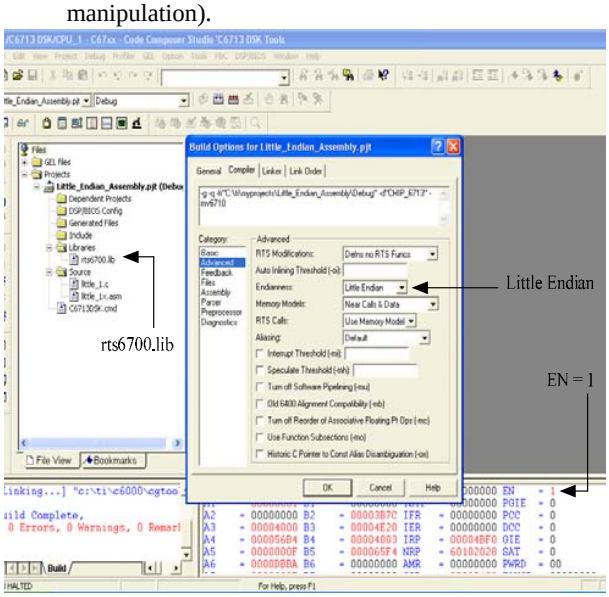


Fig. 2 A screen-shot of the CCS for little endian mode

B. To Set the TMS320C6713 DSK Board in the Big Endian Mode

The TMS320C6713 DSK Board is setup in the little endian mode by default [12], [13]. However, for the TMS320C6713 DSK board to operate in the big endian mode, all **FOUR** of the following conditions must be satisfied.

1. Switch No.1 in the Config SW3 as shown in Fig.1, must be in the **ON** position.

2. As a result of step 1, EN in the Code Composer Studio must indicate **0**. This is read only register and indicates the mode of the DSK board. When EN=0, the DSK board operates in the big endian mode. The value of EN can be displayed in the CCS by **Clicking** View → **Clicking** Register → **Clicking** Core Registers as shown in Fig. 3.

3. The endianness of the TMS320C6713 DSK board needs to be set to Big Endian as shown in Fig.3. This can be done by **Clicking** Project → **Click** Build Options → **Highlight** Advanced and make sure the **Endianness** is set to **Big Endian** and press **OK**

4. Add the library file: **rts6701e.lib** for assembly language programming and add the library files: **rts6701e.lib, csl6713e.lib, dsk6713bsle.lib** for signal processing (i.e. input and output manipulation). For more clarity see Figs. 1 and 3.

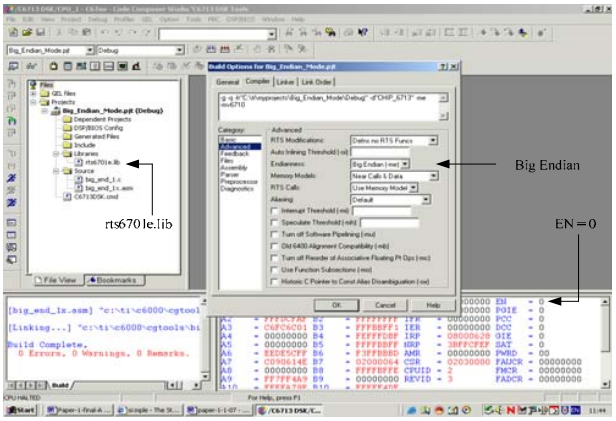


Fig. 3 A screen-shot of the CCS for big endian mode

IV. Various Display Options

The screen-shot of the CCS for displaying the contents of memory locations using various display options when the TMS320C6713 DSK board is operated in the little endian mode, is shown in Fig. 4. These display options include 32-Bit, 16-Bit, and 8-Bit Hex C-Styles and so on. It is very important for a user of the CCS to be aware of these various display options; otherwise, it can lead to a serious confusion regarding the interpretation of the memory contents. For example, in Fig. 4, the contents of the memory location 0x100 can be totally misinterpreted using the 32-Bit display option compared with using the 8-Bit Hex C-Style display options when the TMS320C6713 DSK board is operated in little endian mode.

All users of the CCS without any doubt will interpret the contents of the memory location pointed to by arrow No.1 in Fig. 4, to be stored in the way shown in Fig. 5. However, the correct way in which the data is stored is shown in Fig. 6. The correct result of Fig. 6 is confirmed by the contents of the memory location pointed to by arrow No. 2 in Fig. 4, which is displaying the contents by choosing 8-Bit Hex C Style display option. Therefore, for a user of the TMS320C6713 DSK board, the use of the correct display option plays a crucial rule in the interpretation of the correct result.

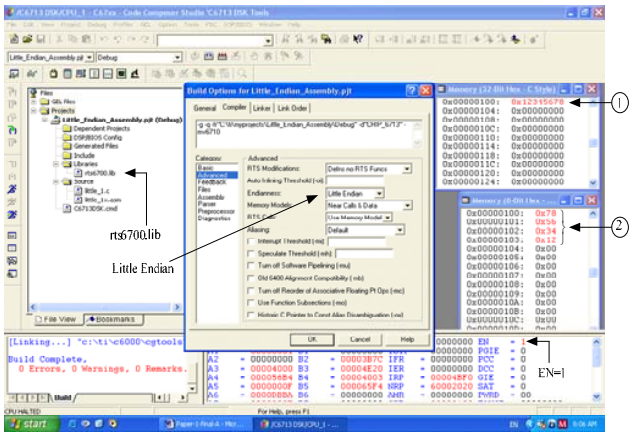


Fig. 4 A screen-shot of the CCS for little endian mode

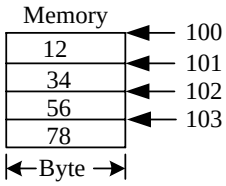


Fig. 5 Wrong interpretation

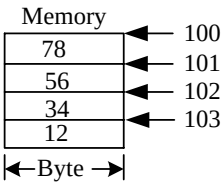


Fig. 6 Correct interpretation

V. Storing/Loading Data Into/From Memory in Big/Little Endian Mode

A user, who has used a particular single-board computer for his/her application and then decides to use another one, needs to know how data is stored in the memory of the new single-board computer [17]. This can cause a major confusion if it is not understood properly. Figs. (7a-12a) illustrate how the registers values are stored into the memory when the TMS320C6713 DSK board is operated in the big endian mode. Similarly, Figs. (7b-12b) explain how the registers values are stored in the memory when the TMS320C6713 DSK board is operated in the little endian mode. As Fig. 7a illustrates, in the big endian mode, the big end (i.e. the most significant byte) of the data is stored first. This means that a hex number like 0xAA23B4F3 is stored in the memory as shown in Fig. 10a. The big end, or upper end, of the data (i.e. 0xAA) is stored first (see Fig. 10a). The same is true for a two-byte value; for example, a hex number like 0x6DC9 would be stored as shown in Fig. 11a. In other words, in the big endian mode, the lower memory addresses contain the most significant byte of the data [1].

However, in the little endian mode, the little end (i.e. the least significant byte) of the data is stored first (see Fig. 7b). This means that a hex number like 0xAA23B4F3 is stored in the memory as shown in Fig. 4b. The little end, or lower end, of the data (i.e. 0xF3) is stored first. The same is true for a two-byte value; for example, a hex number like 0x6DC9 would be stored as shown in Fig. 11b. In other words, in the little endian mode, the lower memory addresses contain the least significant byte of the data [1].

Loading data from the memory also varies from one system to another. This can also become very confusing for a user of a system if not understood properly. Figs. (13a-15a) explain how registers are loaded from the memory when the TMS320C6713 DSK board is operated in the big endian mode. Figs. (13b-15b) describe how registers are loaded from the memory when the TMS320C6713 DSK board is operated in the little endian mode.

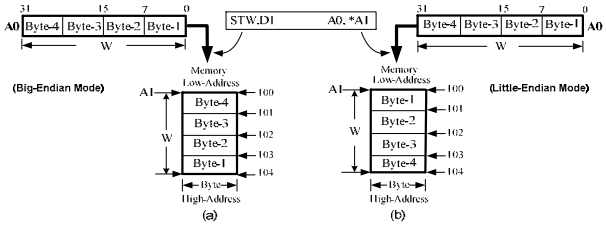


Fig. 7 Storing a word (W) of data in the memory in big and little endian mode

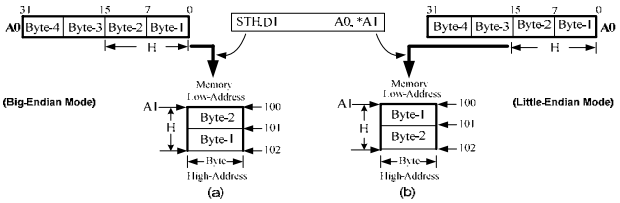


Fig. 8 Storing a half-word (H) of data in the memory in big and little endian mode

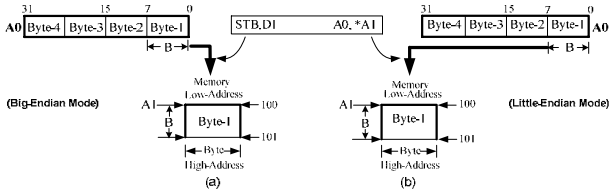


Fig. 9 Storing a byte (B) of data in the memory in big and little endian mode

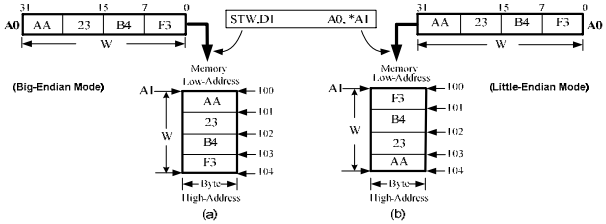


Fig. 10 Example of storing a word (W) in the memory in big and little endian mode

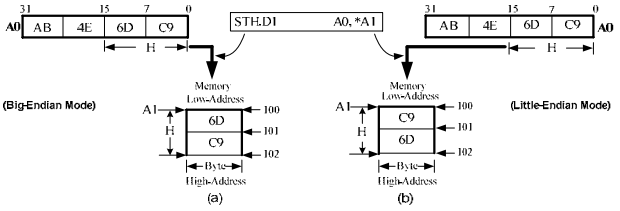


Fig. 11 Example of storing a half-word(H) in the memory in big and little endian mode

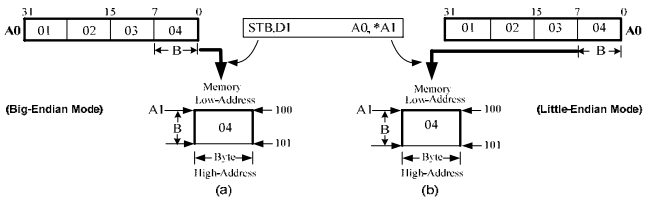


Fig. 12 Storing a byte (B) in memory in big/little endian mode

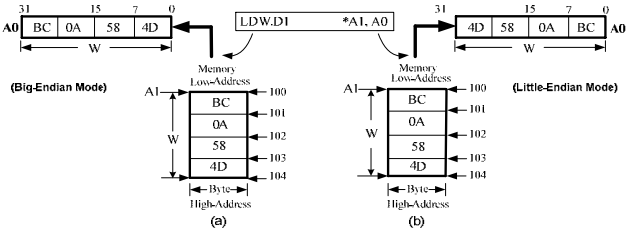


Fig. 13 Example of reading a word (W) from memory in big and little endian mode

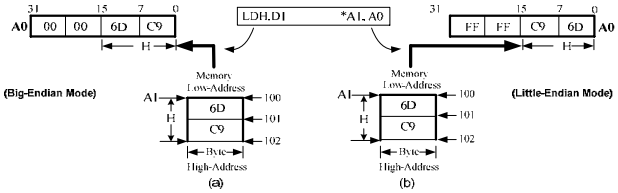


Fig. 14 Example of reading half-word (W) from memory in big and little endian mode

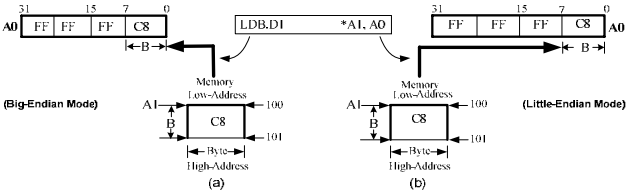
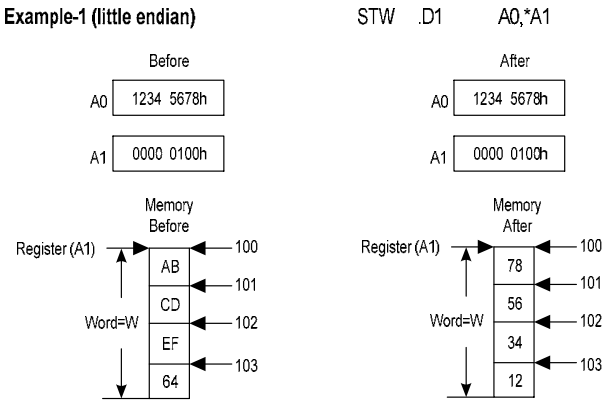


Fig. 15 Example of reading a byte (B) from memory in big and little endian mode

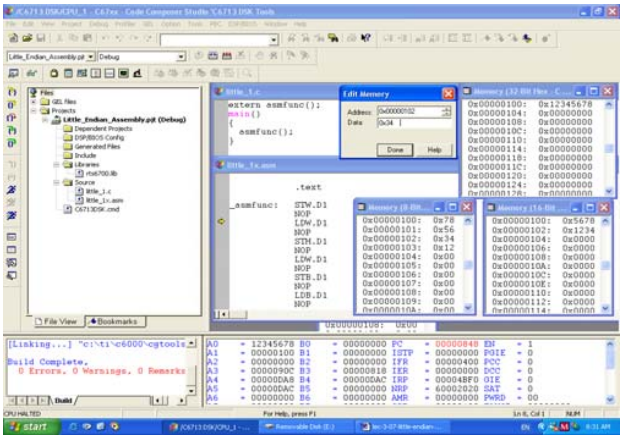
VI. Presented Examples

The examples implemented in this section, highlight the comparisons and contrasts of the two endian modes in great detail. For better understanding, the screen-shots of the Code Composer Studio for these examples are also included.

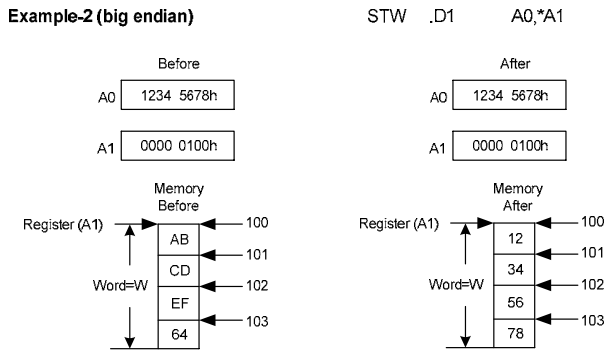
Example-1: In this example, all 32-bits of A0 are stored at the memory location, whose address is in A1. A word (W) in the TMS320C6713 DSP environment means 32-bits (4 bytes). Note that in this example, the DSK board is operated in the little-endian mode.



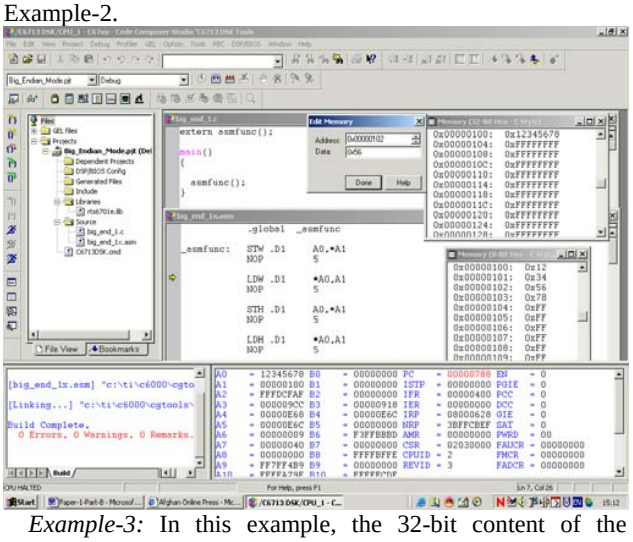
Following is the screen-shot of the CCS that is taken **after** the execution of the instruction presented in Example-1.



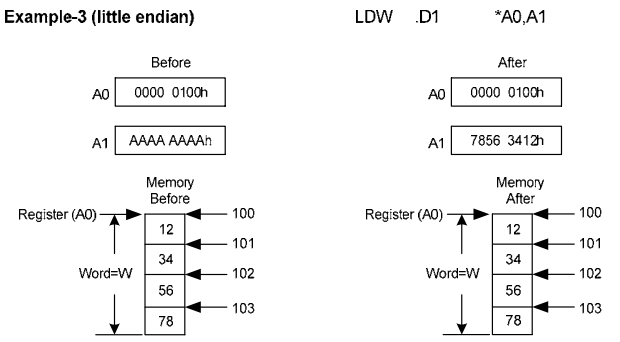
Example-2: In this example, all 32-bits of A0 are stored at the memory location, whose address is in A1. A word (W) in the TMS320C6713 DSP environment means 32-bits (4 bytes). Note that in this example, the DSK board is operated in the big-endian mode.



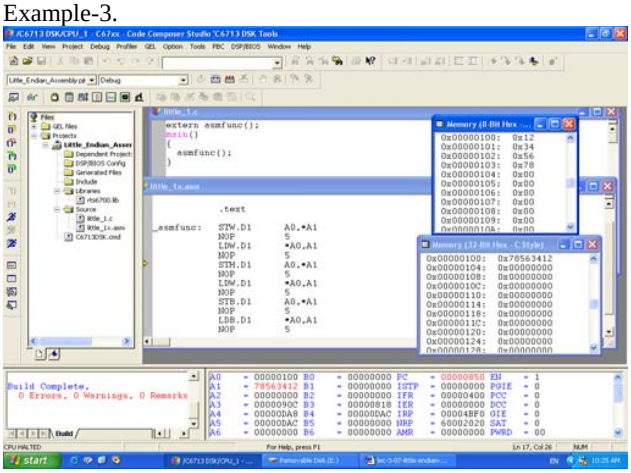
Following is the screen-shot of the Code Composer Studio that is taken **after** the execution of the instruction presented in Example-2.



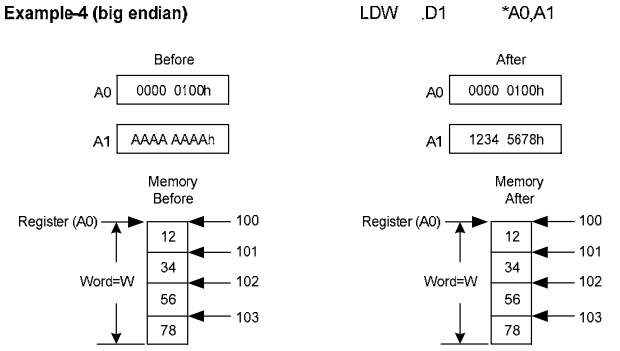
Example-3: In this example, the 32-bit content of the memory location, whose address is in A0, is loaded into A1. Note that in this example, the DSK board is operated in the little-endian mode.



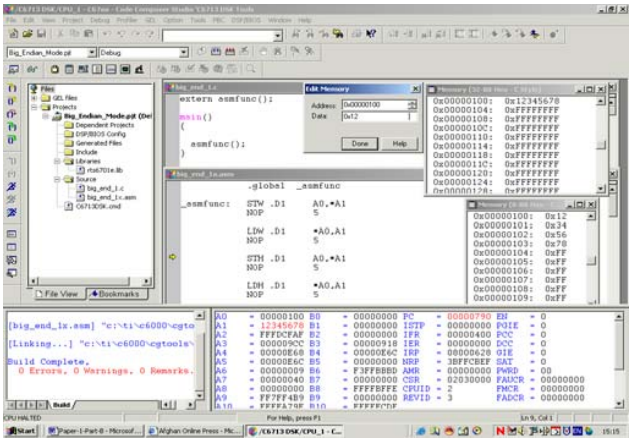
Following is the screen-shot of the Code Composer Studio that is taken **after** the execution of the instruction presented in Example-3.



Example-4: In this example, the 32-bit content of the memory location, whose address is in A0, is loaded into A1. Note that in this example, the DSK board is operated in the big-endian mode.



Following is the screen-shot of the Code Composer Studio that is taken **after** the execution of the instruction presented in Example-4.



VII. Important Remarks

Most of the TMS320C6713 instructions require one CPU clock cycle for their execution [6]. However, it is very important to note that the instructions such as LDW need to access slow external memory and the results of the load instructions are not available immediately at the end of the execution. This execution-delay result is referred to as **a delay slot**. For instance, consider loading up the content of memory at address pointed by A0 to A1 and then moving A1 to A2 [19].

```
LDW .D1      *A0, A1
MV.D1        A1, A2
```

Note that the result of the LDW is not immediately available after LDW instruction is executed. Hence, the MV instruction does not copy the desired value of A1 to A2. To stop this, the CPU needs to wait until the result of the LDW instruction is correctly loaded into A1 before executing the MV instruction. For load instructions, 4 extra clock cycles are needed until the load results are valid. Each NOP instruction makes the CPU idle for one clock cycle and hence the resulting code will be like this [19]:

```
LDW .D1   *A0, A1
NOP        4
MV .D1     A1, A2
```

Then a question might be asked as to why the designer of the CPU did not allocate 5 clock cycles in the first place to the LDW instruction than let the programmer insert 4 NOPs? The answer is that a user can insert other instructions than NOPs as long as those instructions do not use the result of LDW instruction above. By doing this, the CPU can execute additional instructions while waiting for the result of the LDW instruction above [19].

VIII. Conclusion

The concept of the little endian and the big endian mode of the TMS320C6713 DSK board have been comprehensively described. The concept presented in this paper will be of great value and interest to many users who are employing a micro-based system for their applications; and especially for those users who want to use the TMS320C6713 DSK board for assembly language programming and signal processing. It is strongly recommended to the users of the TMS320C6713 DSK board to identify the endian mode of the board first and then employ the board for signal processing purpose; especially in assembly language program. Otherwise, it would cause lots of confusion and erroneous results as far as storing/loading the data into/from the memory are concerned. Furthermore, the users of the TMS320C6713 DSK board are advised to use the board in the big endian mode rather than in the little endian mode, as this will not cause unnecessary confusions. Examples (1-4) present the comparisons and contrasts of the two endian modes in great detail. For better understanding, the screen-shots of the Code Composer Studio for some of the examples are also presented.

Acknowledgment

I would like to thank Southampton Solent University, Faculty of Technology, for giving me the opportunity and help to carry out this work. I would also like to thank Kevin Walsh and Jomo Batola for their support and encouragement.

References

[1] http://support.microsoft.com/kb/q102025/
[2] R. Chassaing, Digital Signal Processing and Applications with the 6713 and C6416 DSK. New York: Wiley, 2005, Ch. 1.
[3] TMS320C6000 Programmer's Guide, SPRU198G, Texas Instruments, Dallas, TX, 2002.
[4] TMS320C6211 Fixed-Point Digital Signal Processor–TMS320C6711 Floating-Point Digital Signal Processor, SPRS073C, Texas Instruments, Dallas, TX, 2000.
[5] TMS320C6000 Peripherals Reference Guide, SPRU190D,Texas Instruments, Dallas, TX, 2001.
[6] TMS320C6000 Optimizing C Compiler User's Guide, SPRU187K, Texas Instruments, Dallas, TX, 2001.
[7] TMS320C6000 Technical Brief, SPRU197D, Texas Instruments, Dallas, TX, 1999.
[8] TMS320C64x Technical Overview, SPRU395, Texas Instruments, Dallas, TX, 2000.
[9] TMS320C6x Peripheral Support Library Programmer's Reference, SPRU273B, Texas Instruments, Dallas, TX, 1998.
[10] Code Composer Studio User's Guide, SPRU328B, Texas Instruments, Dallas, TX, 2000.
[11] TMS320C6000 Code Composer Studio Tutorial, SPRU301C, Texas Instruments, Dallas, TX, 2000.
[12] TMS320C6713 DSK Technical Reference, 506735-0001 Rev.A, May, 2003.
[13] TMS320C6713 Floating Point Digital Signal Processor, Literature Number: SPRS186L, December 2001 - Revised November 2005, P.69.
[14] http://www4.ncsu.edu/~cayunker/mae586/MAE586-tech-manual.pdf
[15] Embedded Target for the TI TMS320C6000™ DSP Platform For Use with Simulink® User's Guide *Version 2* p.76 little endian
[16] http://www.mathworks.com/access/helpdesk_r13/help/pdf_doc/tic6000/tic6000.pdf
[17] A A Wardak, "Real-Time 3-D Image Generation with TMS320C30 EVM", Journal of Microcomputer Applications, Vol. 18, pp 355-373, 1995, Academic Press Limited.
[18] TMS320C6000 CPU and Instruction Set Reference Guide, Literature Number: SPRU189F, Section 1.5, P. 1-6, October 2000.
[19] Rice University, ELEC434, C62x Assembly Premier II, Lab3. Fall 2004.
[20] Code Composer Studio, Getting Started Guide, Literature Number: SPRU509C, November 2001.
[21] TMS320C6000 Assembly Language Tools, User's Guide, Literature Number: SPRU186K, October 2002.