Design of Secondary Bootloader for Embedded System based on DSP

Jie Yan[1,2], Xiaosu Xu[1,2], Lihui Wang[1,2], Yiting Liu[1,2]

**[1] School of Instrument Science and Engineering, Southeast University
Nanjing, 210096, China**

**[2] Key Laboratory of Micro Inertial Instrument and Advanced Navigation Technology, Ministry of Education
Nanjing, 210096, China**

Abstract

The paper desirables the secondary bootloader based on the TMS320C6713B which connected with external 16 bit width FLASH memory. The secondary bootloader design is one of the difficulties in the project development of the embedded system based on DSP. The paper briefly introduces the EMIF and FLASH chip features. Then the hardware interface design is given. The linker command options are used to create boot copy table and the secondary bootloader for large-scale program code is designed. The last, the paper introduces the process of all the design steps. The result shows that the paper offers the method of the secondary bootloader is convenient and useful. This method not only can be used in the TMS320C6713B but also can be used in the other TMS320C6000s.

Keywords: *Digital Signal Processor(DSP), secondary Bootloader, FLASH memory.*

1. Introduction

With the development of electronic technology, the digital signal processor (DSP) has gained rapid development. TI has created TMS320C6000 series digital signal process with advanced very long instruction word (VLIW) DSP core that makes the chip with superior performance. TMS320C6000s are widely used in radar, audio data processing, images data processing and other demanding real-time data processing field. In the DSP system development, complex signal processing is necessary, so that need to use large amounts of data and algorithm program. However, the ROM spaces in the DSP chip so little that it cannot store large program and data. In this case, program code and data need to be stored in the external ROM. When the DSP reset, program code and data which stored in the external ROM need to be moved from external ROM to the DSP chip internal.

The first task to consider is how to copy the program code and data which are stored in the external ROM to DSP chip internal RAM. The different chips have the different ways to boot. Take TMS320C6000s for example, they offer three types of boot configuration.1) No boot process. 2) ROM boot process. 3) Host boot process. The most commonly selected boot configuration is the ROM boot process.

Take C671x series DSP chip for example, when ROM boot is selected as the boot configuration, 1k byte of code and data will automatically be copied from CE1 to address 0 by the Enhanced Direct Memory Access (EDMA) following the release of RESET while the CPU is stalled. When the program code exceeds 1k byte, a special code which called the secondary bootloader should be developed to copy the additional section of code not copied by the ROM boot.

In practice, the program code often exceeds 1k byte, so we should develop the secondary bootloader code. When reset the system, 1k byte code automatically copied by the EDMA from CE1 space in the address 0x90000000 to DSP internal RAM in the address 0x0000000. When this 1k byte code executed, the CPU will copy the remaining code from the CE1 space in the address 0x90000400 to DSP internal RAM in the address 0x00000400, and then begins to run the main function.

2. Hardware Design

The secondary bootloader for the hardware has contact with two parts of the system. The first part is DSP chip, which used in the system is TMS320C6713B. The second part is FLASH chip. In the system, the FLASH chip model is SST39VF800A produced by SST.

2.1 TMS320C6713B and SST39VF800A

The TMS320C6713B device is a floating-point DSP, which based on the high-performance. Operating at 225MHz, the C6713B delivers up to 1350 million float-point operation per second (MFLOPS), 1800 million instructions per second (MIPS) [1].

The C6713B integrated 32-bit external memory interface (EMIF), and the external space capacity of 64MB. The external space divided into 4 independent spaces such as (CE0~CE3) which can be different control. The data bus width is 32 bit, and also support 8 / 16bit registers are read / write. The EMIF support a glueless interface to a variety of

external devices, including SDRAM, SBSRAM, ROM, FIFOs [2].

The SST39LF800A is produced by SST. The chip writes (Program or Erase) with single 3.3V power supply. When the chip is in the active module, the active current is typical 20mA. When the chip is in the standby mold, the standby current is typical 3μA. The address bus width is 19 bits, and the data bus is 16 bits [3].

2.2 System hardware interface design

The DSP bootloader mainly associates with DSP EMIF module and FLASH (SST39VF800A). The data bus (D0~D15) in the FLASH correspondingly connect with the data bus (ED0~ED15) in the C6713B. The address bus (A0~A18) in the FLASH correspondingly connect with the address bus (EA2~EA20) in the C6713B. $\overline{CE1}$ pin in the C6713B connects with the corresponding $\overline{CE}$ in the FLASH. $\overline{CE}$ pin is used for device selection. When $\overline{CE}$ is high, the chip is deselected, otherwise the chip is selected. $\overline{AOE}$ pin in the C6713B connects with $\overline{OE}$ pin in the FLASH. $\overline{OE}$ is the output control and is used to gate data from the output pins. The data bus is in high impedance state when either $\overline{CE}$ or $\overline{OE}$ is high. $\overline{AWE}$ pin in the C6713B connects with $\overline{WE}$ pin in the FLASH. $\overline{WE}$ is input control and is used to latch data from the input pins. A command is written by asserting $\overline{WE}$ low while keeping $\overline{CE}$ low. Figure1 shows the EMIF to SST39LF800A interface.

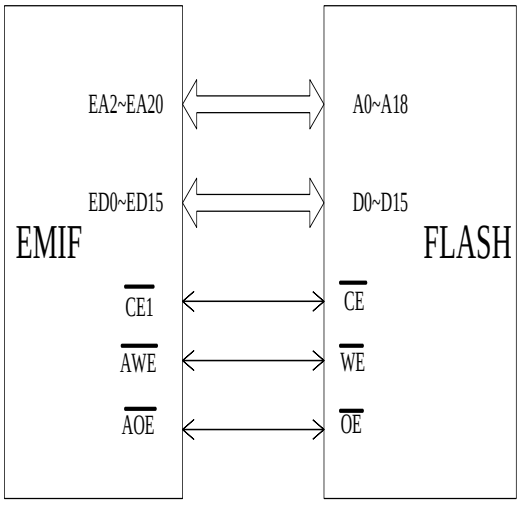

Figure1. EMIF To SST39LF800A Interface

In this system, the FLASH chip data bus width is 16 bit, so we should set the appropriate EMIF registers to 16 bit width. However, the EMIF default bus width is 32 bit. When EMIF connect with 16bit FLASH, EMIF automatically changes the double 16 bit data read from FLASH into a 32 bit data. Each read operation of EMIF is always carries in 32 bit data. The output of the address will automatically shift in order to ensure that the memory read operation in the 16 bit width. Therefore, when EMIF connects with 16 bit width data bus, the EMIF address automatically left one, and the high bit will be automatically removed from discarded [4].

3. Secondary Boot Program Theory

When system resets, the EDMA will automatically copy 1k byte program code from external FLASH. In this moment, the CPU has not yet started work. When EDMA has been copied all 1k byte code from CE1 to address0, the CPU begins to work. The code copied by the EDMA is called secondary bootloader code [5]. Once the project has been built and linked, the secondary bootloader code should be written. The secondary bootloader is typically written in assembly language because the C environment is not initialized at boot-time. The secondary bootloader always achieve following several features:

The first point
Configure the PLL: This step is recommended for C6713B that has software programmable PLL in order to configure the registers in PLL. After configure PLL, the CPU begins to work, and executes the remaining code.

The second point
Configure the EMIF: In order to achieve the glueless connect with external SDRAM, FLASH, the EMIF registers such as EMIF Global Control Register (GBLCTL), EMIF CE Space Control Register (CECTL) and so on.

The third point
Copy the initialized section code from the FLASH: This step achieve the function that copies the code from FLASH which the address space occupy the CE1 in the address 0x90000400 to DSP internal run address 0x00000400.

The fourth point
Call_c_int00: Secondary bootloader copies the complete application to its runtime memory. At completion, the secondary bootloader code branches to _c_int00

The last
Go to Main: After Call_c_int00, the system can support the C environment, the C program can be copied to DSP Main Program and run.

Figure2 shows start-up sequence of application which uses secondary bootloader.

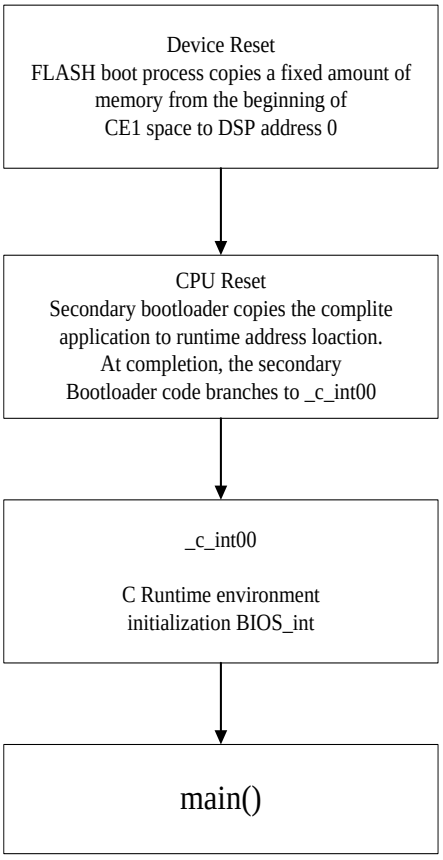

Figure2. Start-up Sequence of Application

4. Secondary Bootloder Copy Table

The secondary bootloader copies the contents of memory sections from its load address to its run address using a section copy table. The copy table contains entries for all the memory sections that need to be copied from their load address to their run address. Each table entry contains information describing the size of the section of memory, the destination address or address from where the section will execute, and the source address or the address where the section was loaded.

There are a number ways to create the section copy table, 1) Inspecting the map file. If this method is chosen to create the copy table, then the appropriate value of the size of the section, the destination address of the section, and the source address of the section must be manually filled in for that entry. 2) Using the –boot option in the hex conversion utility. The hex conversion utility provided with Code Composer Studio provides a more convenient method for creating the section copy table by automatically building the copy table when the appropriate options are specified in the hex conversion utility command file. 3) Using linker options.

This embedded system based C6000 DSP uses the linker options to create the section copy table. This method is simple to use, so selects this way to create copy table. The linker options include three parts such as LOAD_START, RUN_START, and SIZE. The user gives the definition of all parts. In the following code is the definition in the link2.cmd file [6].

```
ROMS
{
   FLASH:org= 0x9000000 ,
   Len = 0x40000,
   Rowwidth = 8,
}

.SECTIONS
{
  .boot_load : load = FLASH_BOOT,
               run =BOOTRAM

  .text : load = FLASH_REST,
          run = IRAM

   LOAD_START(_text_ld_start),

   RUN_START(_text_rn_start),

   SIZE(_text_size)

  .cinit      >       FLASH_REST
  .const      >       IRAM
  .stack      >       IRAM
  .bss        >       IRAM
  .data       >       IRAM
  .far        >       IRAM
  .switch     >       IRAM
  .sysmem     >       IRAM
  .cio        >       IRAM

  }
```

In the following code is the definition about the copy table in the boot_c671x.s62 file.

```
copyTable:

   ; count
   ; flash start (load) address
   ; ram start (run) address

      ;; .text
         .word _text_size
         .word _text_ld_start
         .word _text_rn_start

   ;; end of table
         .word 0
```

```
.word 0
.word 0
```

When function project has built successfully in the CCS, boot_c671x.s62 file and link2.cmd file will be load to the project in order to generate COFF format file.

5. Programming Flash

Code Composer Studio (CCS) comes with several utilities that help with flashing applications into ROM. For C6000s a Flash programmer tool is offered in the CCS. A hex conversion utility is also provided. Because the Flash programmer works only with the hex format, the COFF format file obtained from CCS must be converted to .hex file through the hex conversion utility [7] [8].

The following procedure describes how to create an application to program into flash. First of all, build the target project in order to generate the .out file. Secondly, use the hex conversion utility to create a .hex file from the .out file [9]. The last, DSP program the flash with the .hex file. Figure 3 shows the flash programming sequence.

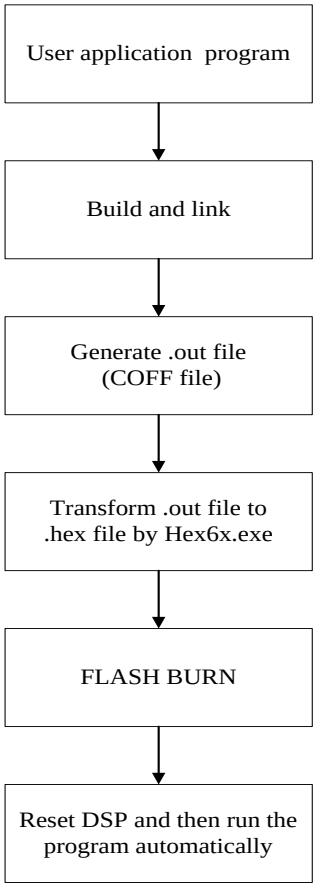

Figure 3. Flash Programming Sequence

The boot.cmd code lists in the following which can convert the .out file to .hex file.

```
BOOT.out
-map BOOT.map
-a
-image
-zero
-memwidth 8

ROMS
{
FLASH: org = 0x90000000,
len = 0x40000,
romwidth = 8,
files = {BOOT.hex}
}

SECTIONS
{

.boot_load
.text
.cinit
.pinit
.const
.switch

}
```

The SST39LF800A FLASH supports sector and block erase operation [10]. The sector size is 4k byte, and the block size is 64k byte. Flash erase operation is on the basis of sector and block. Once FLASH programming, user should erase the chip, and then program the code into the Flash. The Sector Erase operation is initiated by executing a six-byte command sequence with Sector-Erase command (30H) and sector address (SA) in the last cycle. The Block-Erase operation is initiated by executing a six-byte command sequence with Block-Erase command (50H) and block address (BA) in the last bus cycle. Any commands issued during the Sector or Block-Erase operation are ignored.

6. Conclusions

As is known to all, in the development of the embedded system which is based on DSP, the internal ROM is so little to storage too much user code. Hence, a second-level bootloader should be created by user, which is a difficult task in the DSP development. The paper which is based on TMS320C6713B describes an easy and convenient way to create a secondary bootloader. This method uses ROM boot process and uses linker options to create the section copy table. The experiment results show that the method offered by this paper is effective and convenient. The

method is not only could be used in C6713B device also can be used in other C6000 devices.

7. Acknowledgment

The project was supported by following funds: National Natural Science Foundation of China (Nos. 51175082, 61203192, 61273056), Natural Science Foundation of Jiangsu Province (BK2012326), Open Research Fund of State Key Laboratory of Transient Optics and Photonics, Chinese Academy of Sciences (SKLST201108).

References

[1] Texas Instruments Incorporated, "TMS320CB713B FLOATING-POINT DIGITAL SIGNAL PROECSSOR", June, 2006.

[2] Texas Instruments Incorporated, "TMS320C6000 DSP External Memory Interface (EMIF) Reference Guide", April, 2008.

[3] Silicon Storgae Technology, Inc, "2 Mbit/ 4Mbit / 8Mbit (x16) Multi-Purpose Flash", 2002.

[4] SanHengXing Technology, TMS320C6713 DSP Theory and Application, BeiJing Electronic Industry Press，2009.

[5] Texas Instruments Incorporated, "Creating a Second-Level Bootloader for FLASH Bootloading on TMS320C6000 Platform with Code Composer Studio", May, 2006.

[6] Zhang guolong, Xu xiaosu, "Implemention of Secondary Bootloader for Large-scale Embedded System", Computer Engineering, vol. 36, No.13, July, 2010, pp 219-221.

[7] Vijayarajeswaran, R., "DSP Implementation of a Power Factor Correction Strategy for BLDC Motor Drive" , International Journal of Computer Science Issues, v 9, n 1 1-3, January 2012, pp 215-220.

[8] Radhakrishnan N.and Ramaswamy M. "DSP implementation of Fuzzy based Power Quality enhancement strategy for IPOS converter fed drive", International Journal of Computer Science Issues, v 9, n 2 2-2, 2012, pp 301-305.

[9] Suthar, A.C., Vayada, Mohammed and Patel, C.B. et al, "Hardware Software co-simulation for Image Processing Applications", International Journal of Computer Science Issues, v 9, n 22-2, January 2012, pp 560-562

[10] Chen daiyuan, "External FLASH Memory's Bootloader System for C6000," Telecommunication Engineering, vol. 49, No5, May, 2009, pp. 86-88.

Jie Yan, male, is now a Ph.D. graduate in the School of Instrument Science and Engineering, Southeast University, Nanjing, China. His main research interest is Embedded Systems, SINS/GNSS integrated navigation.

Xiaosu Xu, male, received his Ph.D. degree in the School of Automation, Southeast University, Nanjing, China, in 1991. He is a professor in the School of Instrument Science and Engineering at Southeast University, also is the director of Key Laboratory of Micro Inertial Instrument and Advanced Navigation Technology, Ministry of Education, China. His current research interests include inertial navigation, integrated navigation.

Lihui Wang, male, received BS, MS, and Ph.D. degrees, all from Harbin engineering university, China, in 2002, 2005, and 2009, respectively. From 2009, he joined post-doctoral research center in Southeast University, China, and now he is an associate professor in southeast university. He has been engaged in developing fiber optic gyroscopes (FOG) and fiber optic current sensor (FOCS). His major research interests are error analysis of fiber optic sensors, application of FOG in strapdown inertial navigation system and FOCS in electric power system.

Yiting Liu, male, received his master degree in detection technology and automatic equipment form Wuhan instititude of technology, Hubei,China. And now, he is a Ph.D. graduate in the School of Instrument Science and Engineering, Southeast University, Nanjing, China. His main research field is embedded system and inertial navigation.