Contents

5. Memory Controller

5.1. 實驗目的

了解 memory map 的應用原理，並依此原理熟悉 memory 及 external memory interface 的使用方法，作為日後處理資料的基礎練習。

5.2. 實驗原理

5.2.1. System memory map

The system memory map is shown in Figure 1, which divides memory into many parts.

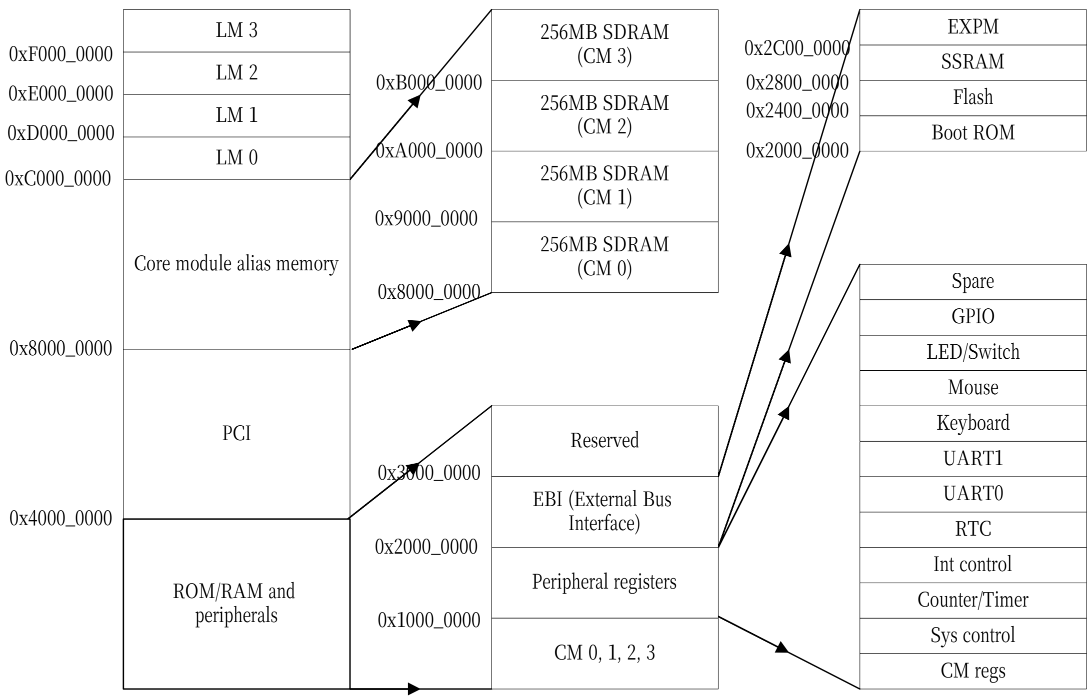

Figure 1 System memory map

(1) The core module has a fixed memory map that maintains compatibility with ARM Integrator motherboards and modules.
(2) All I/Os, bus interface, and memory have their own address.
(3) nMBDET: detect if **motherboard** attach to core module, because core

module can be used alone, just like 8051.

(4) REMAP: ROM is slow & narrow to RAM, so use this register to change memory map after initialization.

5.2.2. Core Module Control Register

Some registers are set value by program to control current memory map attribute. You can see the Table 1.

Little-endian is data stored from MSB to LSB in memory. Big-endian is data stored from LSB to MSB in memory, mostly used in Motorola.

RESET can let core module return initial state.

Bits	Name	Access	Function
31:6	Reserved		
5	BIGEND	R/W	0=little-endian 1=big-endian
4	Reserved		
3	RESET	W	Reset core module
2	REMAP	R/W	0=access Boot ROM 1=access SSRAM
1	nMBDET	R	0=mounted on MB 1=stand alone
0	LED	R/W	0=LED OFF 1=LED ON

Table 1 Control Register in Core Module

5.2.3. Core Module Memory Map

There are four switches on the board, change these switches could control the memory map. These methods are always used when write memory image of program into flash/ROM.

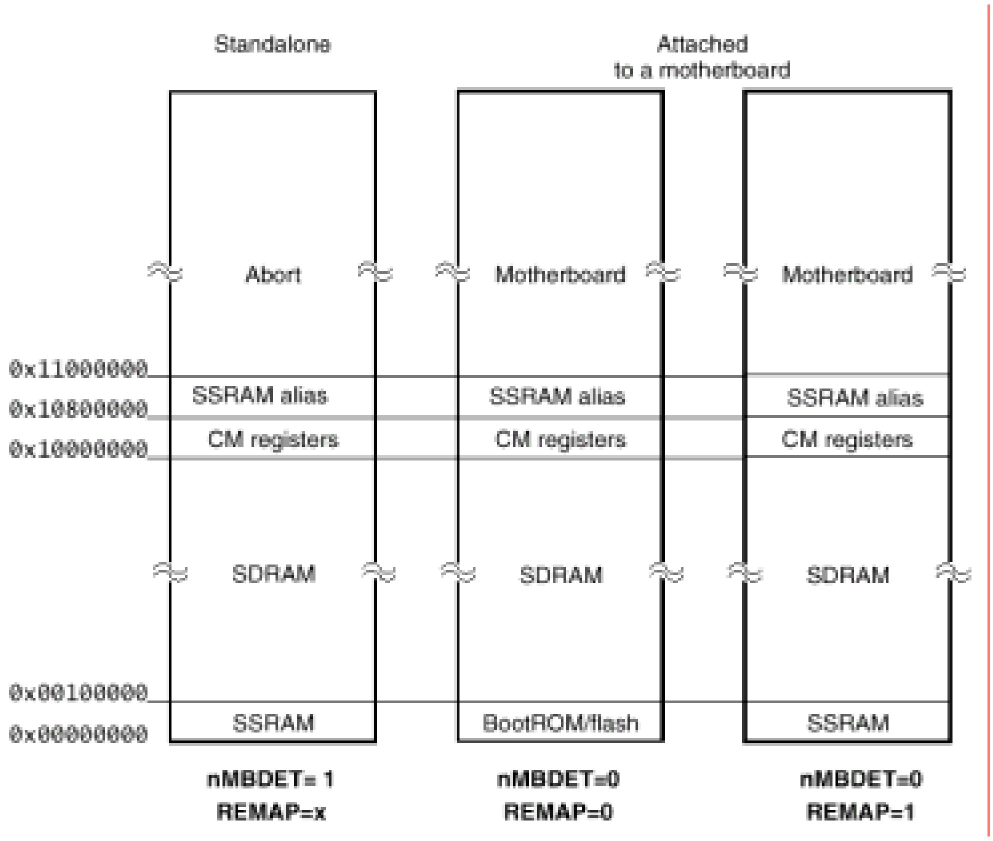

Figure 2 Core Module Memory Map

When nMBDET = 0 & REMAP = 0: Change the switch on the board,
(a) S1[1] = ON: access BootROM;
(b) S1[1] = OFF: access flash.

5.3. 引導實驗

This program does the following tasks:

1. Backup the data in the SSRAM at locations 0x30000 to 0x38000 range to the SDRAM at locations 0x80000000 to 0x80008000.
2. Write values to the SSRAM at locations 0x30000 to 0x38000.
3. Verify the values in the SSRAM at locations 0x30000 to 0x38000.
4. Restore the backup data back to their original locations.

```
#include <stdio.h>

int main(void){
      unsigned int      CM_CTRL_ADDR = 0x1000000C;
      unsigned int      SSRAM_ADDR = 0x00000000;      // CM's SSRAM
ranges 0x0 ~
                                        //0x0003FFFF
      unsigned int      *CM_CTRL_PTR, *SSRAM_PTR, *SDRAM_PTR;
      unsigned int i;
      int SSRAM_test_error = 0;
```

```
	CM_CTRL_PTR = (unsigned int *) CM_CTRL_ADDR;
	SSRAM_PTR = (unsigned int *) SSRAM_ADDR;

	// Memory Remap to SSRAM
	*CM_CTRL_PTR =0x4;

	printf ("SSRAM Write Test\n");
	printf ("Press any key to start SSRAM test!!\n");

	getchar ();
	*CM_CTRL_PTR =0x5;

	printf ("Backup SSRAM data from 0x0000 to 0x8000 to SDRAM at 
0x80000000\n");
	for (i=0;i<0x8000;i+=4)
	{
		SDRAM_PTR = (unsigned int *)(i+0x80000000);
		SSRAM_PTR = (unsigned int *)(i+0x30000);
		*SDRAM_PTR = *SSRAM_PTR;
	}

	printf ("Writing...\n");
	for (i=0;i<0x8000;i+=4)
	{
		SSRAM_PTR = (unsigned int *)i;
		*SSRAM_PTR = i;
	}

	printf ("Verifying...\n");
	for (i=0x0;i<0x8000;i+=4)
	{
		SSRAM_PTR = (unsigned int *)i;
		if (*SSRAM_PTR != i)
		{
			printf ("SSRAM W/R test error!!\n");
			printf ("Error address>> %x\n",i);
			SSRAM_test_error = 1;
			Getchar ();
		}
	}
	if (SSRAM_test_error != 1)
		printf ("SSRAM test passed!!\n");

	*CM_CTRL_PTR =0x4;
	printf ("SSRAM test finished.\n");

	printf ("Restore original SSRAM data from 0x00000000 to 
0x80008000 to SSRAM at 0x8000\n");
	for (i=0;i<0x8000;i+=4)
	{
		SDRAM_PTR = (unsigned int *)(i+0x80000000);
		SSRAM_PTR = (unsigned int *)(i+0x30000);
		*SSRAM_PTR = *SDRAM_PTR;
	}

	return 0;
}
```

5.3.1. 實驗步驟

1. Start CodeWarrior IDE.
2. Select **File** → **New** to create a new project (Figure 3).
   (1) Select ARM Executable Image under the Project tab.
   (2) Type the project name, EX1 for example. You can see the result in Figure 3.
   (3) Specify the project path.

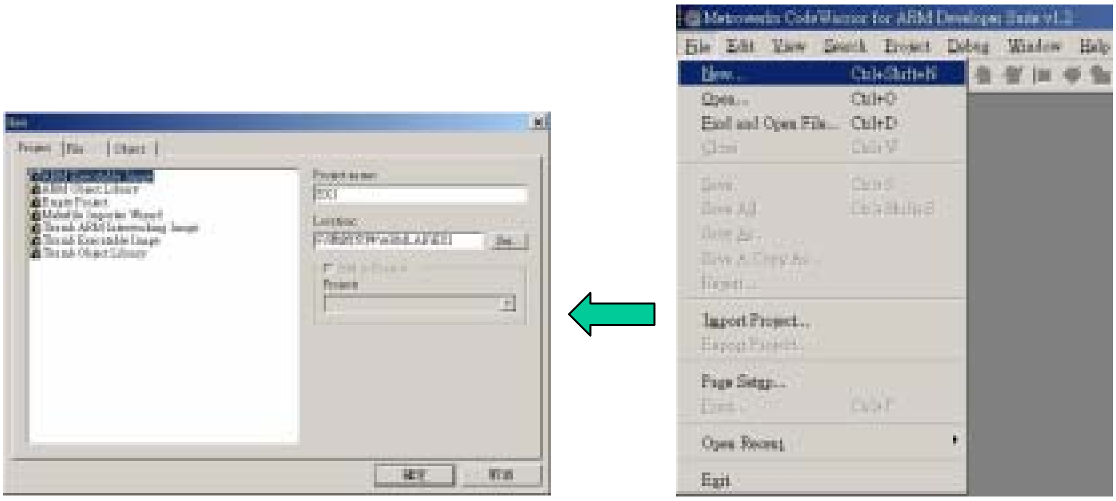

Figure 3 New dialog box

3. Adding source files to the project.
   (1) Copy file SSRAM.C to your EX1 directory.
   (2) Select **Project** → **Add Files.**
   (3) Navigate to the EX1 directory and click on SSRAM.C.
   (4) Click **Open**. Then Add all files to targets (Figure 4).

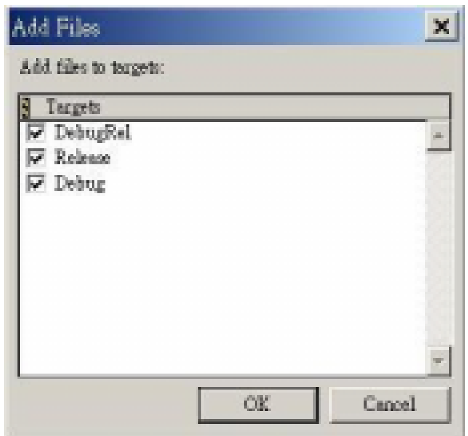

Figure 4 Add files to targets dialog box

4. Hit the **Make** button to compile and link the project.

(1) A compiling and linking status windows would appear to indicate making progress
(2) After finishing compiling and linking, a result message windows would appear (Figure 5). Check for errors and warnings.

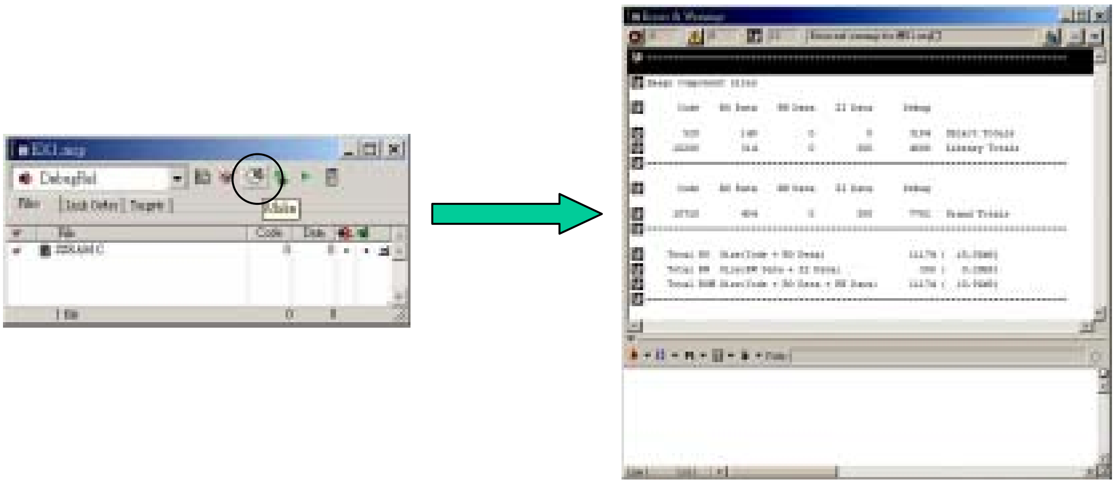

Figure 5 Make the project

5. Hit the **Run** button to run the program.
   (1) The CodeWarrior IDE calls AXD debugger to load and execute the image (Figure 6).

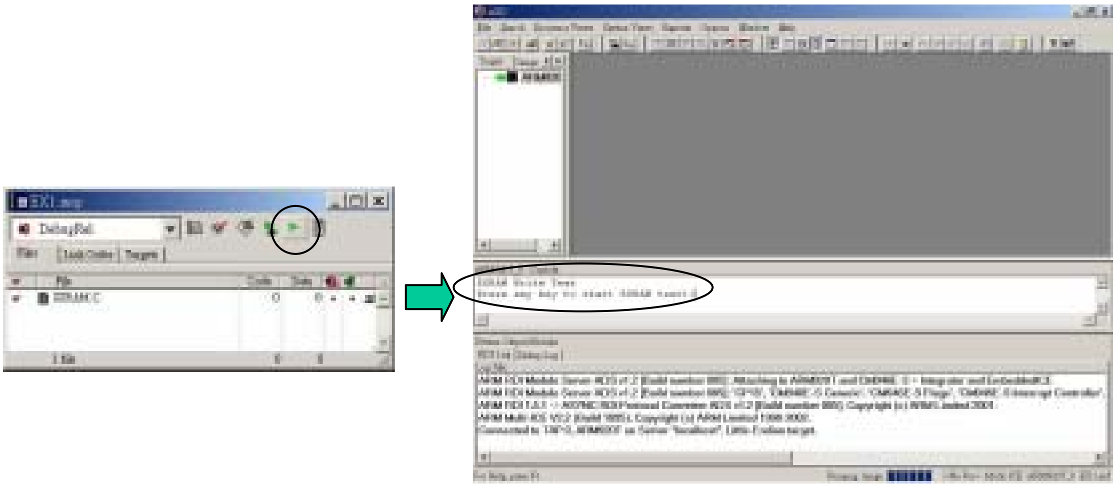

Figure 6 Run the project

   (2) Press any key and AXD debugger starts memory test. It shows SSRAM finished when AXD is done (Figure 7).

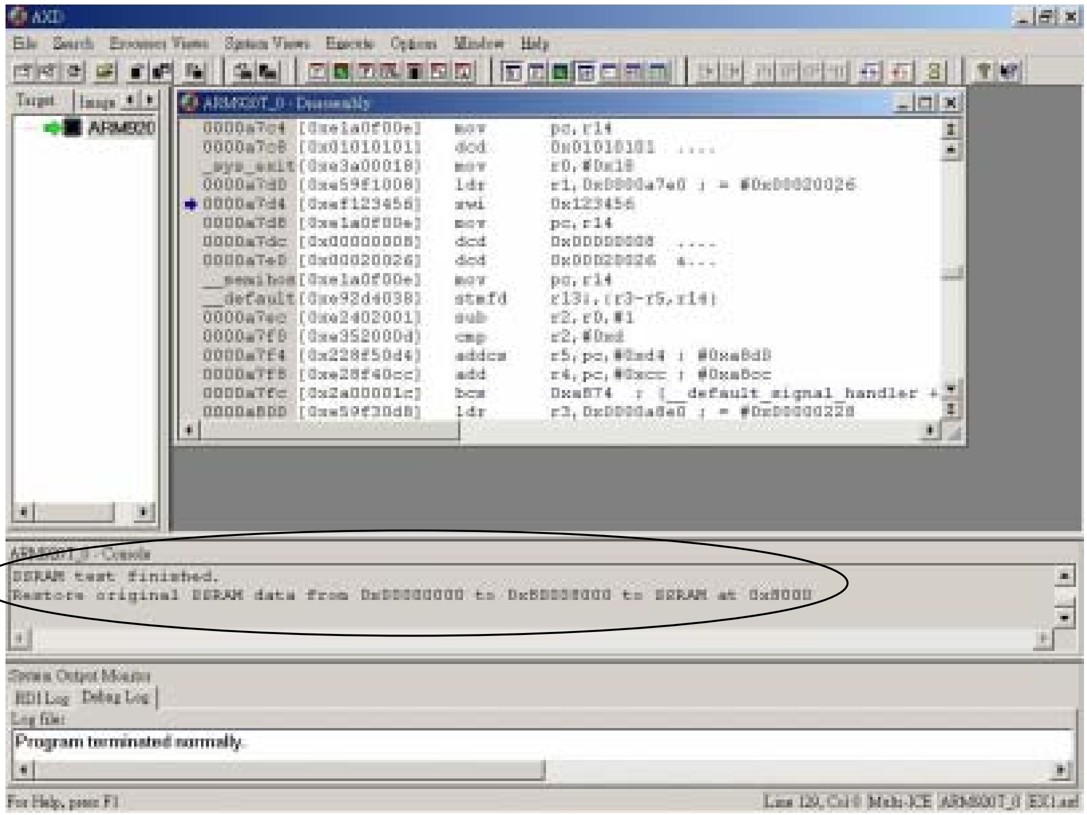

Figure 7 Starts memory test

6. From AXD debugger, select **Processor Views → Memory**.
7. Check write to SSRAM values
   (1) Click Tab 1 – Hex - No Prefix. You can see the Address Values are increased by 0x4 (Figure 8).

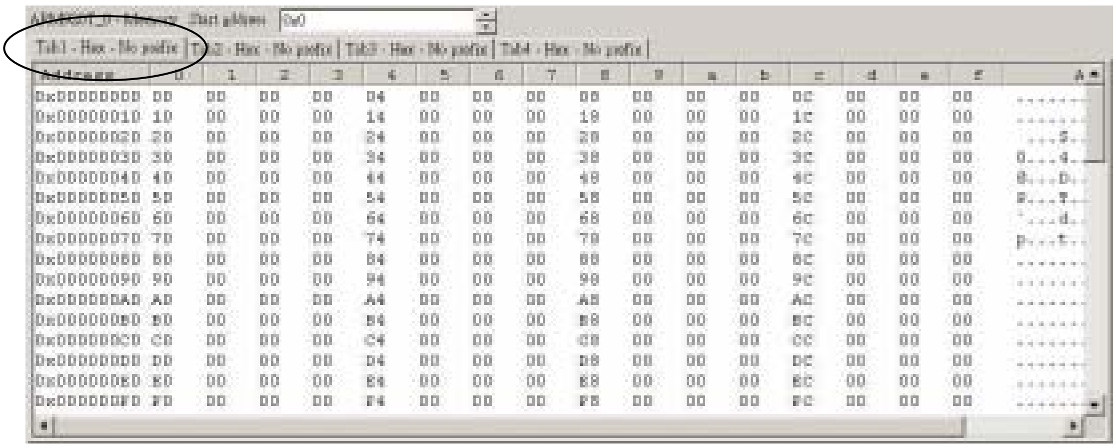

Figure 8 Write to SSRAM values

8. Check Address value
   (1) In the Tab 2 – Hex - No Prefix, type 0x30000 into the Memory Start Address and press Enter. You can see the Address values in the table

(Figure 9).

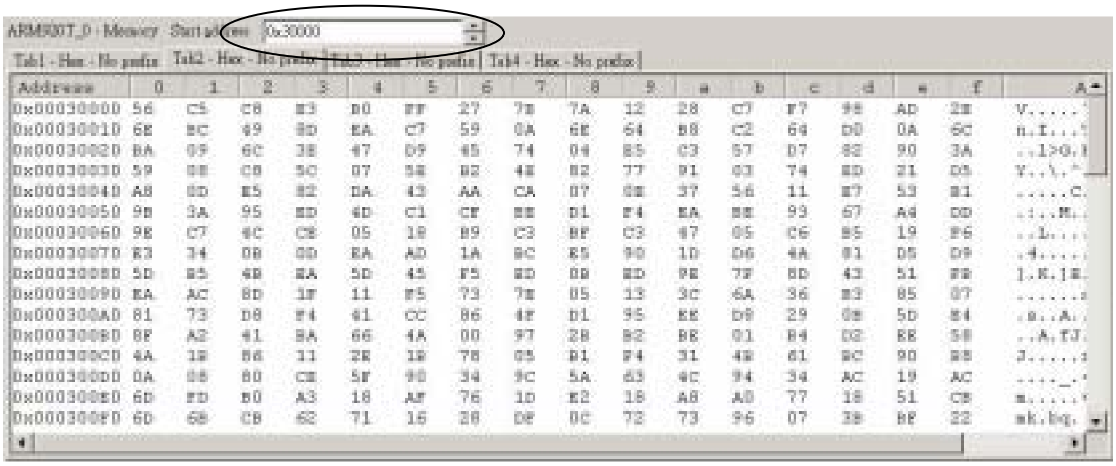

Figure 9 Check Address value at 0x30000

(2) Click Tab 3 – Hex - No Prefix and type 0x80000000 into the Memory Start Address and press Enter. You can see the Address values are the same as those in 0x30000 (Figure 10).

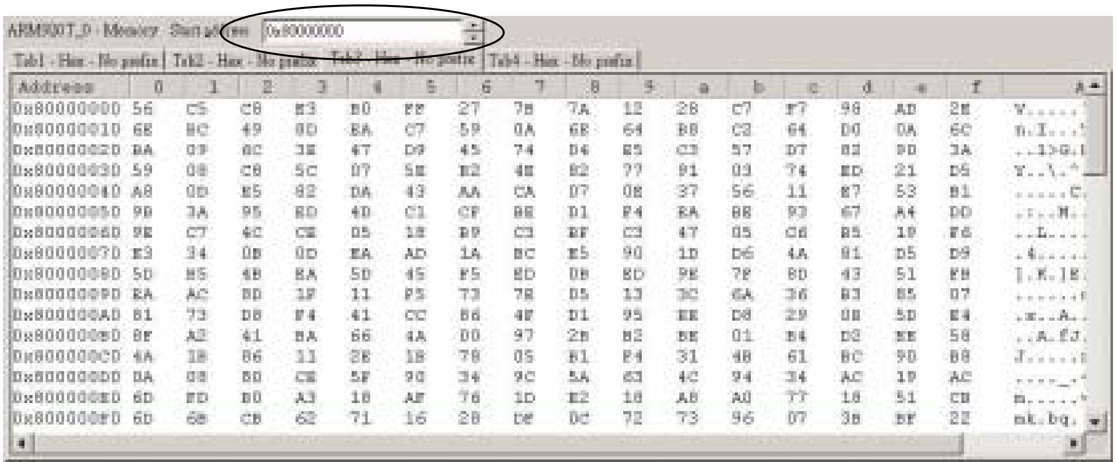

Figure 10 Check Address value at 0x80000000

5.4. 實驗要求

1. Try to compile this program by using *Thumb* code to get the same result. Modify the memory usage example if needed. Show the statistics about ARM codes and Thumb codes and compare their differences.
2. Compare the performance between using SSRAM and SDRAM.

5.5. 問題與討論

Discuss the following items about Flash, RAM, and ROM.
(1) Speed
(2) Capacity
(3) Internal/External

5.6. 參考文件及網頁

- Integrator ASIC platform [DUI_0098B_AP_UG]
- System Memory Map [DUI_0098B_AP_UG 4.1]
- Core Module [DUI_0126B_CM9TDMI]
- Core Module Registers [DUI_0126B_CM9TDMI 4.2]
- Core Module Memory Organization [DUI_0126B_CM9TDMI 4.1]
- SSRAM [DUI_0126B_CM9TDMI 3.2]
- SDRAM [DUI_0126B_CM9TDMI 3.4]