

# Contents

5. Memory Controller.....	5-1
5.1. 實驗目的 .....	5-1
5.2. 實驗原理 .....	5-1
5.2.1. System memory map .....	5-1
5.2.2. Core Module Control Register .....	5-2
5.2.3. Core Module Memory Map .....	5-2
5.3. 引導實驗 .....	5-3
5.3.1. 實驗步驟 .....	5-5
5.4. 實驗要求 .....	5-8
5.5. 問題與討論.....	5-9
5.6. 參考文件及網頁.....	5-9

## 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.

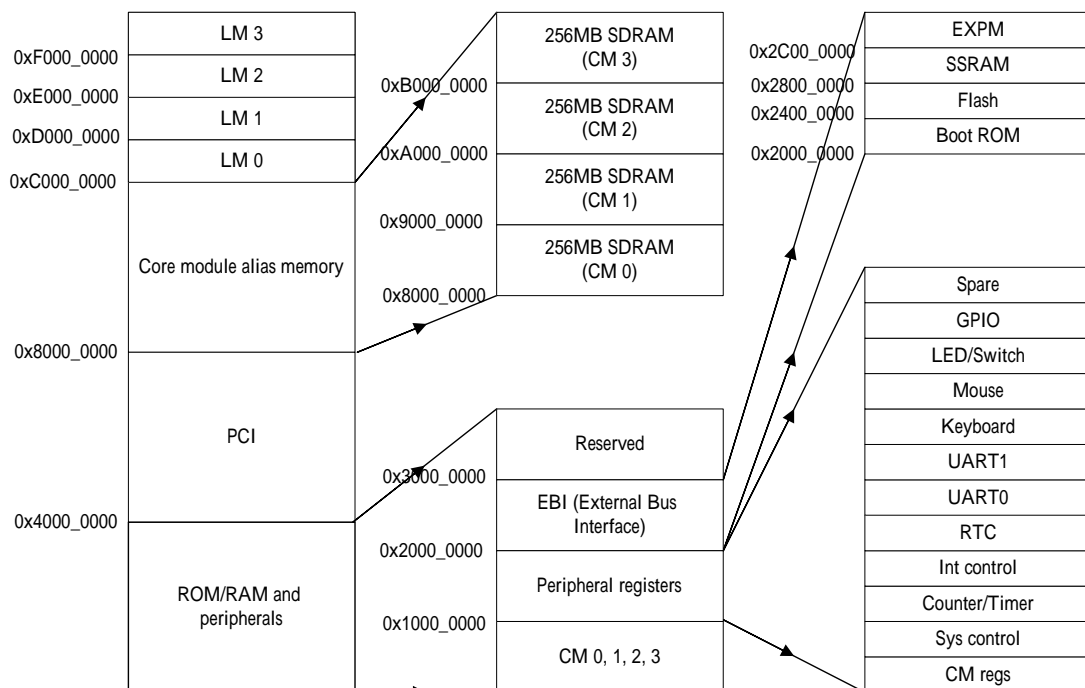


Figure 1 System memory map

- (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

## Memory Controller

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.

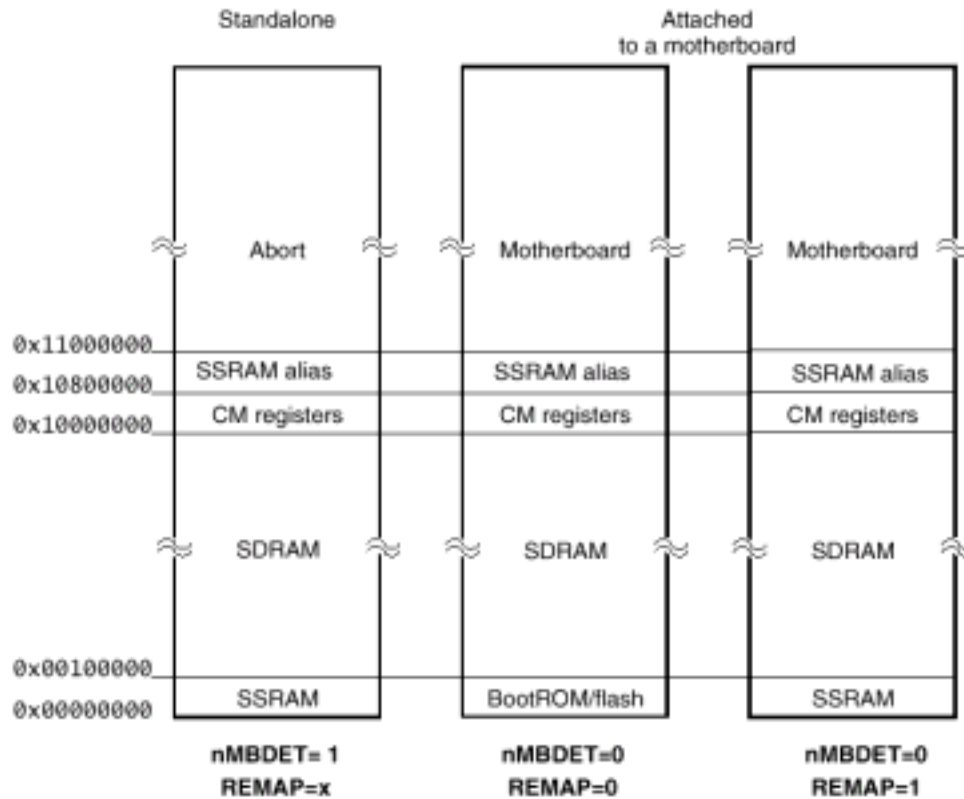


Figure 2 Core Module Memory Map

When nMBDET = 0 & REMAP = 0: Change the switch on the board,

- S1[1] = ON: access BootROM;
- S1[1] = OFF: access flash.

### 5.3. 引導實驗

This program does the following tasks:

- Backup the data in the SSRAM at locations 0x30000 to 0x38000 range to the SDRAM at locations 0x80000000 to 0x80008000.
- Write values to the SSRAM at locations 0x30000 to 0x38000.
- Verify the values in the SSRAM at locations 0x30000 to 0x38000.
- 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;
```

## Memory Controller

```
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.

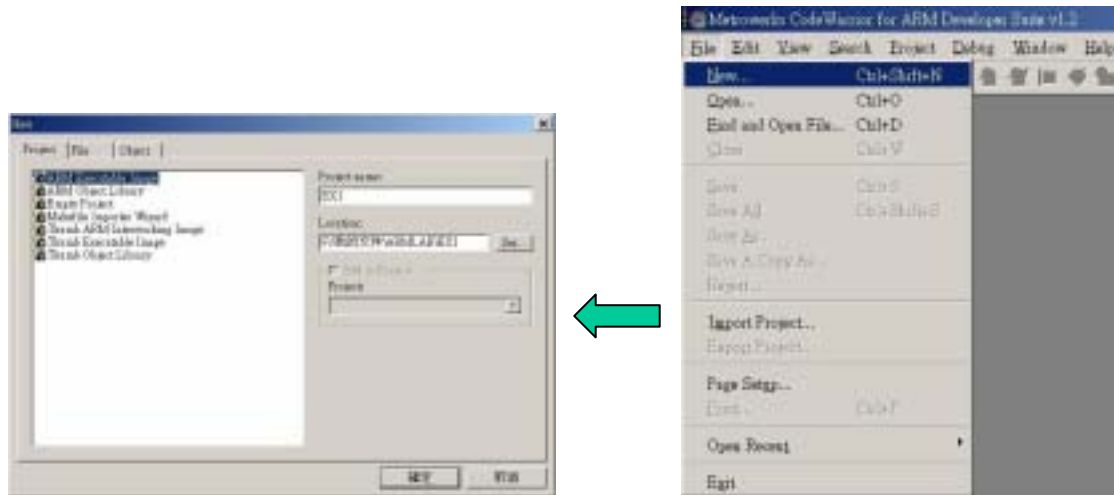


Figure 3 New dialog box

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).

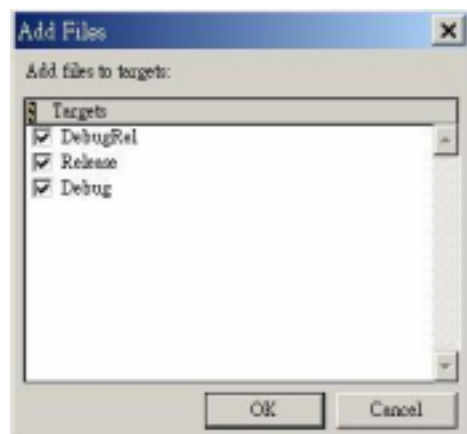


Figure 4 Add files to targets dialog box

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

## Memory Controller

- (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.

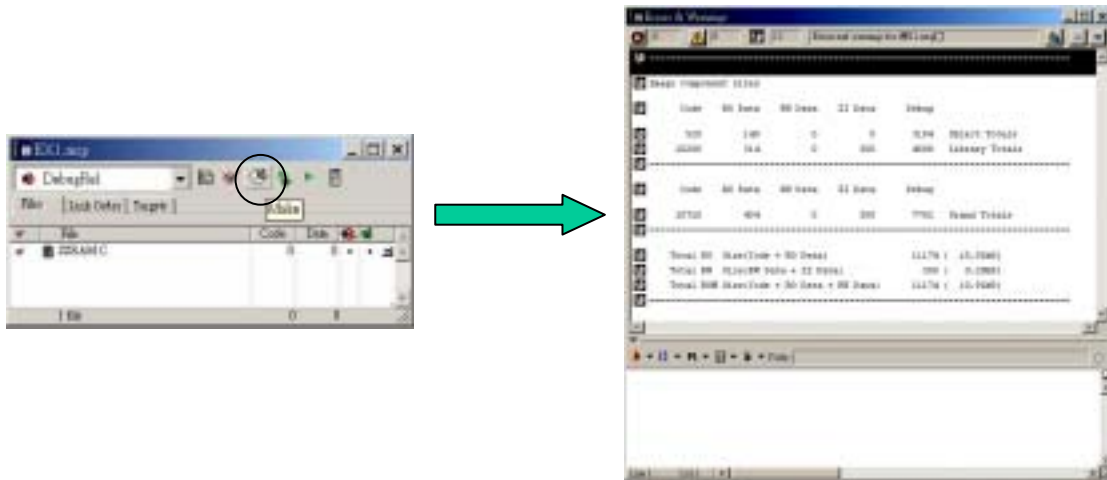


Figure 5 Make the project

### 5. Hit the **Run** button to run the program.

- (1) The CodeWarrior IDE calls AXD debugger to load and execute the image (Figure 6).

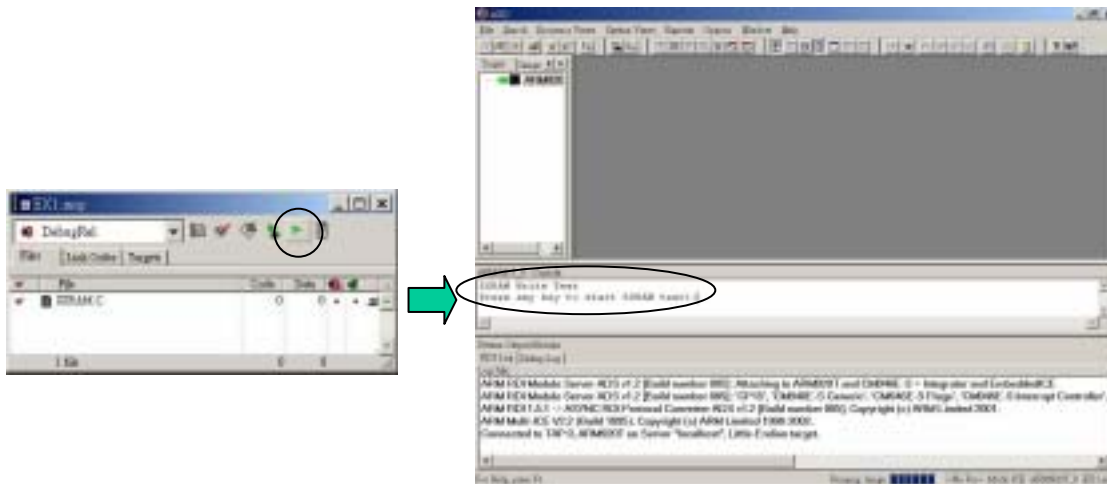


Figure 6 Run the project

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

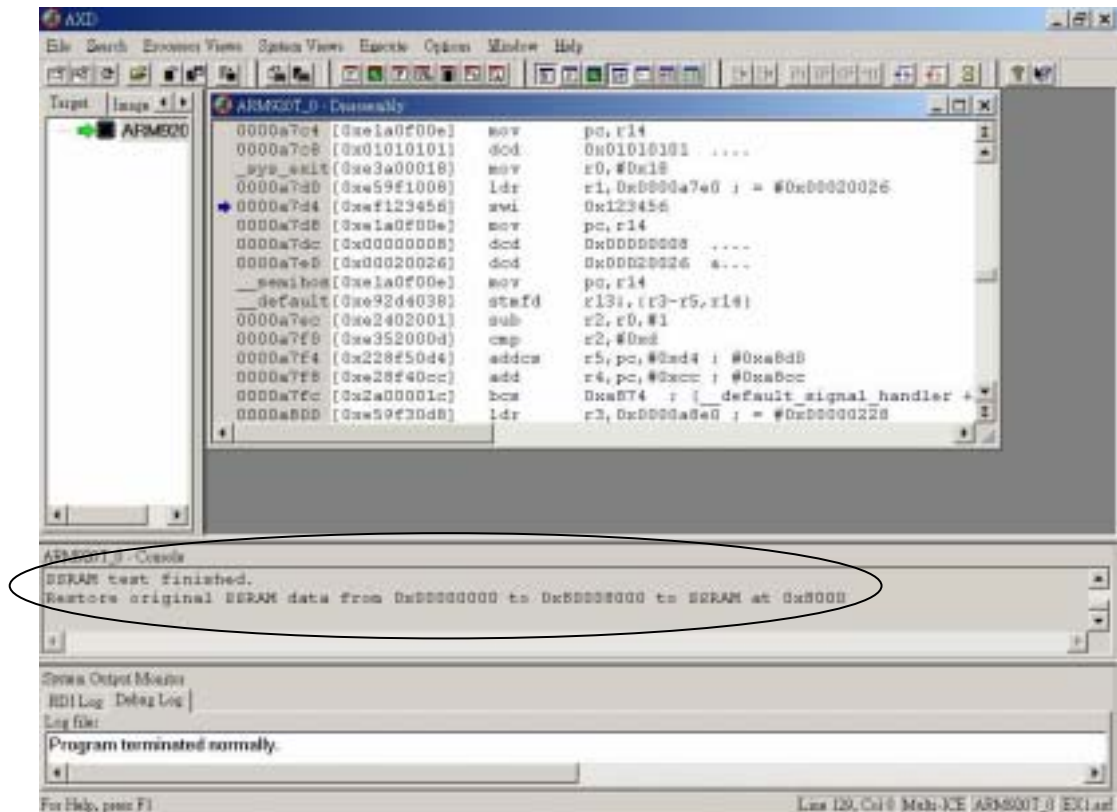


Figure 7 Starts memory test

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).

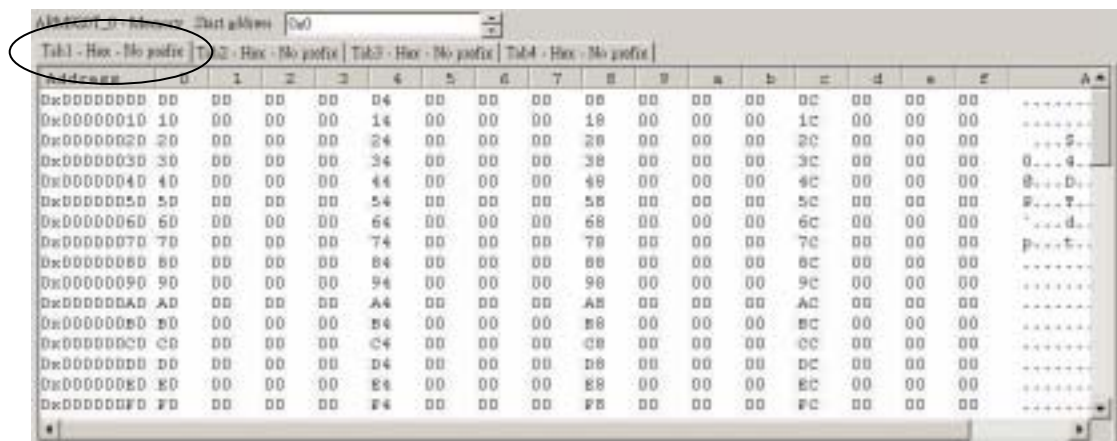


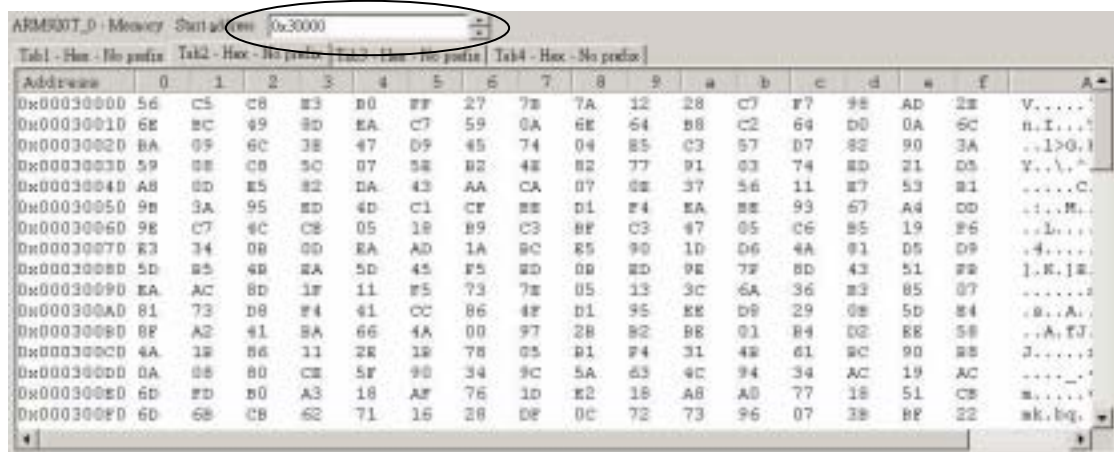
Figure 8 Write to SSRAM values

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



## Memory Controller

(Figure 9).

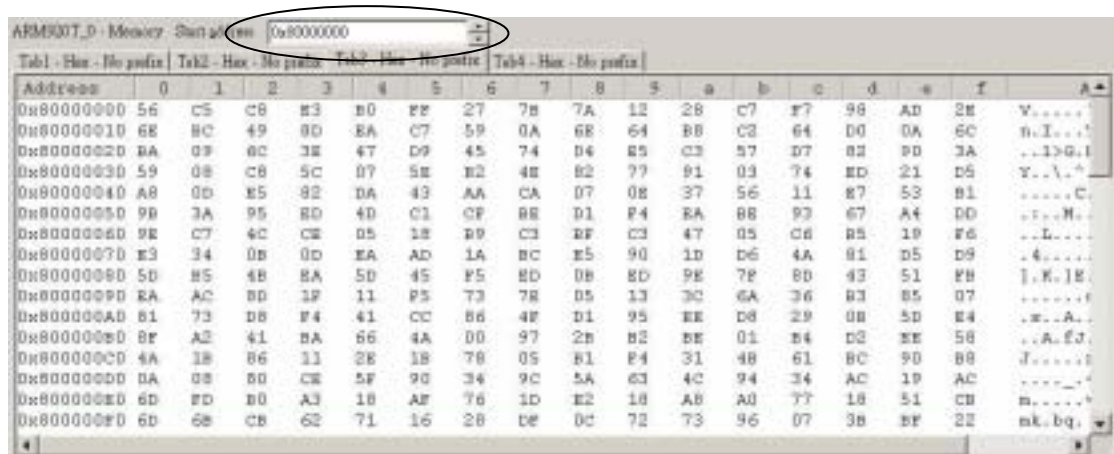


The screenshot shows the ARM930T\_D Memory Controller interface. The 'Start Address' field is highlighted with a red circle and contains the value '0x30000'. Below the input field, there are four tabs: 'Tab1 - Hex - No prefix', 'Tab2 - Hex - No prefix', 'Tab3 - Hex - No prefix', and 'Tab4 - Hex - No prefix'. The main display area shows a memory dump starting at address 0x00030000. The data is organized into columns labeled 0 through f, and the last column is labeled 'A'. The first few rows of data are:

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	A
0x00030000	56	c5	c8	e3	80	ff	27	7b	7a	12	28	c7	f7	98	ad	2e	V.....
0x00030010	6e	8c	49	8d	8a	c7	59	0a	6e	64	88	c2	69	d0	0a	6c	n.I....
0x00030020	8a	09	6c	3e	47	d9	45	74	04	85	c3	57	d7	82	90	3a	...1>G.l
0x00030030	59	08	c8	5c	07	5e	b2	4e	82	77	91	03	74	ed	21	d5	Y...^+
0x00030040	a8	0d	e5	82	da	43	aa	ca	07	08	37	56	11	e7	53	81	.....C.
0x00030050	9b	3a	95	8d	4d	c1	cf	88	d1	f4	8a	88	93	67	a4	dd	...M...
0x00030060	9e	c7	4c	ce	05	18	b9	c3	8f	c3	47	05	c6	85	19	f6	...L....
0x00030070	e3	34	0b	0d	8a	ad	1a	8c	e5	90	1d	d6	4a	81	d5	d9	.4.....
0x00030080	5d	85	48	8a	5d	45	f5	8d	08	8d	9e	7e	8d	43	51	fb	]..K.IE.
0x00030090	8a	ac	8d	1f	11	f5	73	7e	05	13	3c	6a	36	b3	85	07	.....
0x000300a0	81	73	d8	f4	41	cc	86	4f	d1	95	ee	d8	29	08	5d	e4	...A...
0x000300b0	8f	a2	41	8a	66	4a	00	97	28	82	8e	01	84	d2	8e	58	...A.fj.
0x000300c0	4a	18	86	11	28	18	78	05	d1	f4	31	48	61	8c	90	88	J.....
0x000300d0	0a	08	80	ce	5f	90	34	9c	5a	63	4c	94	34	ac	19	ac	.....
0x000300e0	6d	fd	80	a3	18	af	76	1d	e2	18	a8	ad	77	18	51	cb	n.....
0x000300f0	6d	68	cb	62	71	16	28	de	0c	72	73	96	07	38	8f	22	mk.bq.

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).



The screenshot shows the ARM930T\_D Memory Controller interface. The 'Start Address' field is highlighted with a red circle and contains the value '0x80000000'. Below the input field, there are four tabs: 'Tab1 - Hex - No prefix', 'Tab2 - Hex - No prefix', 'Tab3 - Hex - No prefix', and 'Tab4 - Hex - No prefix'. The main display area shows a memory dump starting at address 0x80000000. The data is organized into columns labeled 0 through f, and the last column is labeled 'A'. The first few rows of data are:

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	A
0x80000000	56	c5	c8	e3	80	ff	27	7b	7a	12	28	c7	f7	98	ad	2e	V.....
0x80000010	6e	8c	49	8d	8a	c7	59	0a	6e	64	88	c2	69	d0	0a	6c	n.I....
0x80000020	8a	09	6c	3e	47	d9	45	74	04	85	c3	57	d7	82	90	3a	...1>G.l
0x80000030	59	08	c8	5c	07	5e	b2	4e	82	77	91	03	74	ed	21	d5	Y...^+
0x80000040	a8	0d	e5	82	da	43	aa	ca	07	08	37	56	11	e7	53	81	.....C.
0x80000050	9b	3a	95	8d	4d	c1	cf	88	d1	f4	8a	88	93	67	a4	dd	...M...
0x80000060	9e	c7	4c	ce	05	18	b9	c3	8f	c3	47	05	c6	85	19	f6	...L....
0x80000070	e3	34	0b	0d	8a	ad	1a	8c	e5	90	1d	d6	4a	81	d5	d9	.4.....
0x80000080	5d	85	48	8a	5d	45	f5	8d	08	8d	9e	7e	8d	43	51	fb	]..K.IE.
0x80000090	8a	ac	8d	1f	11	f5	73	7e	05	13	3c	6a	36	b3	85	07	.....
0x800000a0	81	73	d8	f4	41	cc	86	4f	d1	95	ee	d8	29	08	5d	e4	...A...
0x800000b0	8f	a2	41	8a	66	4a	00	97	28	82	8e	01	84	d2	8e	58	...A.fj.
0x800000c0	4a	18	86	11	28	18	78	05	d1	f4	31	48	61	8c	90	88	J.....
0x800000d0	0a	08	80	ce	5f	90	34	9c	5a	63	4c	94	34	ac	19	ac	.....
0x800000e0	6d	fd	80	a3	18	af	76	1d	e2	18	a8	ad	77	18	51	cb	n.....
0x800000f0	6d	68	cb	62	71	16	28	de	0c	72	73	96	07	38	8f	22	mk.bq.

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]