# Contents

Memory	v 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
	Memory 5.1. 實驗 5.2. 實驗 5.2.1. 5.2.2. 5.2.3. 5.3. 引導 5.3.1. 5.4. 實驗 5.5. 問題 5.6. 參考	Memory Controller

## 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.

```
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
0x8000000\n");
      for (i=0;i<0x8000;i+=4)</pre>
      {
            SDRAM_PTR = (unsigned int *)(i+0x8000000);
            SSRAM_PTR = (unsigned int *)(i+0x30000);
            *SDRAM_PTR = *SSRAM_PTR;
      }
      printf ("Writing...\n");
      for (i=0;i<0x8000;i+=4)</pre>
      {
            SSRAM_PTR = (unsigned int *)i;
            *SSRAM_PTR = i;
      }
     printf ("Verifying...\n");
      for (i=0x0;i<0x8000;i+=4)</pre>
      {
            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)</pre>
      {
            SDRAM_PTR = (unsigned int *)(i+0x8000000);
            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**  $\rightarrow$  **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.

		Metrowerks Code Wa	mor for ARM I	https://pe	a saak vi.	<u> </u>
Trans The Sheet	×	Eine Edit Vanne Ein Denne Eine and Open Film. Class	erk Ereset Chieffanets Chieff Chieff Chieff	Dubag	Window W (m 4	Help 9 Su
Added a second present de Added Transformer Strauer Present Character Strategy Character	Product scener DX1 Langtonc (7-DB20 Of Walked Add 201 (201) Theorem	Gove Bove Ad. Dove 24 Bove A. Copy So. Frence.	Cund Co's Bulled			
	-	Taport Project Expert Project				
		Page Sitggi	cubr			
1.00		Open Recent		•		
	87 fin	Egit				

Figure 3 New dialog box

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

T	 	_
Release		1
Debug		

#### 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.

		1.00	un få Wene	-					-11 R
		0	1	10	ht le	COLUMN STREET	na Miliona	0	21-1-1
		10 - C							-
		10 ×	ia car	44(1) THE	· · · ·				- 0
		10	. inter	it here	00 Date	11 1444	2 million of		
#EG.acp		50	10	100	1		1.14	MONT TOUS Linney Trans	
Debugfiel     Pier     Debugfiel     Pier     Debugfiel     Pier		-		Al have	-	11 1410	( televal)		
- 11-	Code Tate (2) of 1		3752					Padenti Treasa	
■ TTRAM C	1 (···a)	000	Steal II Steal II Treal II	L Bardin Rayin B Bardin	+ 80 Desa 100 + 22 0 + 30 Jana	n mai Milanai	1115 1115	1 ( 10.000) 1 ( 10.000) 1 ( 10.000)	
158	0 1 2	-1							1
			11 - M -	11+¥+	nej				0
									-
		Alex AL	-	int .					*0

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).

Target   Image +   +	ARMART_0 - Dummably			- ICINI
AF3M520	0000a7c4 [0xe1a0f00e] 0000a7c8 [0x0101010] 	NOT dod HUY Ldr swi Bor dod dod NOT sub cmp addcs add bcs Ldr	pd,r14 Dx010101010 r0,#Dx18 r1,0x0000047+0 ; = #0x000020026 Dx123456 pc,r14 Dx000020026 pd,r14 r13;(r3-r5,r14) r2,r0,#1 r2,%Dx1 r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB r4,pc,#0xed ; #0xa8dB	dler + N
<u>BLANT D</u> Could TERAR teat find Sectors origina	ished. 11 EERAM data from DaDDDDD	000 to De	BD00B000 to SERAM at 0x8000	
bries Output Moaster HDI Log Debag Log   .ng file:				
Demaran Incruised	f normally			

Figure 7 Starts memory test

- 6. From AXD debugger, select **Processor Views**  $\rightarrow$  **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).

Address	0	1	2	1	6	5	4	7	1		- 41	1 b	=	-4		£	A.*
Debooooooo	DD.	0.0	DD.	00	0.6	00	DD	00	00	00	0.0	00	00	00	00	00	
Dx00000010	10	0.0	0.0	0.0	14	0.0	0.0	0.0	19	0.0	0.0	0.0	ic	0.0	0.0	0.0	
0x00000020	20	DD.	D.D.	DD	24	0.0	0.0	0.0	20	0.0	0.0	0.0	SC	00	00	0.0	
DxDDDDDD3D	30	DD.	DD	00	34	00	0.0	00	38	00	00	00	36	00	0.0	00	0
DEDDDDDDAD	10	0.0	DD	00	4.6	00	00	00	40	00	00	00	4C	00	00	00	8D
Debbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb	50	00	DD	00	54	00	DD	00	58	00	00	00	5C	00	00	00	F T
DeDDDDDDDDD	60	DD	DD.	00	64	00	0.0	0.0	68	0.0	0.0	0.0	6C	0.0	0.0	0.0	* eredee
DEDDDDDD7D	70	DD	DD	DD	74	00	0.0	DB	78	88	0.0	00	7C	88	0.0	00	Prester
DEBDDDDDDBD	BD .	DD	DD	BB.	84	00	00	00	88	00	0.0	00	8C	00	00	0.0	
DH00000090	90	00	DD	80	94	00	00	80	98	00	00	00	9C	00	00	00	
DEDDDDDDDAD	AD	00	DD	00	.64	00	00	00	All	00	00	00	AC.	00	00	00	
DEDDDDDDDDDD	20	0.0	00	0.0	54	0.0	00	00	<b>B</b> 8	00	0.0	00	BC.	0.0	0.0	0.0	
DEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD	CB	DD	DD.	00	C4	00	00	00	0.8	00	0.0	00	CC	00	0.0	00	*******
DEDDDDDDDDD	DD	DD	00	00	D4	DD	00	DU.	118	00	00	00	DC	00	0.0	0.0	
DEDDDDDDDDD	ED	0.0	DD	00	E9.	0.0	00	00	88	0.0	0.0	0.0	BC .	0.0	00	0.0	
DEDDDDDDEFD	FD	DD	DD	00	24	00	DD	00	FB	00	00	00	FC	00	00	00	
•1																100	+

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).
---------	-----

Address	0	1	2	3	4	5	6	7	8	- <u>P</u>	( a)	ъ	c	:d	1 - A1	f	:A.*
0x0002000x0	56	C5	C8	#3	n0	27	27	710	7A.	12	28	C7	17	98	.AD	21	Vereite
0x00030010	68	BC	49	00	EA.	C7	59	0A	6E	64	88	C2	64	b0	ŪA.	60	n. I
0800030050	BA	09	6C	38	47	09	45	74	04	85	C3	67	D7	82	9.0	3A	1>0.1
0x00030030	59	08	CB	50	07	58	82	48	82	77	91	03	74	ED.	21	D3	Y \*-
Dx0003004D	AB	00	85	82	IA.	4.3	AA.	CA	07	08	37	56	11	17	53	81	C.
0x80030050	98	BA.	95	#D	40	Cl	CF	日本	D1	24	EA.	15	93	67	.A.4	DD	- 1 M
0x00030060	9.8	07	4C	CB	05	18	89	C3	BF	C3	-17	05	06	85	19	26	In
0x00030070	83	34	08	00	EA	AD.	1A	BC.	85	90	1D	D6	4A	01	D5	D9	Acres
0200200080	50	85	48	第名	50	45	85	80	0.0	2D	PE	77	dB.	43	51	22	1.8.18.
0400500080	EA.	AC.	BD	15	11	#5	73	7=	05	13	30	6A	36	83	8.5	07	
DA00E000HD	81	73	DB	24	41	CC.	86	42	01	95	88	09	29	08	50	214	- 8A.
0x00030080	85	A2	+1	BA.	66	48	0.0	97	28	82	BE	01	职业	02	BB	58	A. TJ.
D×000300CD	4A.	18	86	11	28	19	78	05	81	24	31	48	61	#C	90	28	J
0x000200000	DA.	- 10.6	80	CE	51	90	34	9C	5A.	63	90	94	34	AC.	1.9	AC	
0x00030080	60	2D	50	A3	18	AF	76	1.0	<b>E</b> 2	18	AB	A0	77	18	51	CB	Manage V.
DH0003008D	60	68	CB	62	71	1.6	28	DF	0c	72	73	96	07	38	BF	22	ak.bq
a la constante de la constante																1	

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).

Address	0	1	1 2	3		6	6	7	.8	5		- b	0	d		f	
01000000880	56	C5	¢θ	#3	20	55	27	78	7A.	12	28	c7	\$7	98	AD	2#	Y
01000000810	68	HC.	49	0.0	EA	C7	59	0A	6E	64	BB	C2	64	DO	DA.	60	n. I
0x80000020	BA.	0.9	00	32	47	D9	45	7.4	0.6	85	03	57	D7	82	PD.	3A	1>0.1
0x80000030	59	08	CB	SC	07	510	82	40	82	22	91	0.9	.74	ED	21	DS .	Y \
0100000810	AB	00	85	82	DA	4.9	AA.	CA	07	08	37	56	11	87	53	81	C.
0x80000050	9.0	38	95	RD	40	C1.	CP.	BE.	D1	P.4	RA.	88	92	67	.84	DD	+ 7 × + H++
0200000880	92	C7	40	CE	05	18	20	CB	RF.	623	47	05	Cf	85	19	86	+ - It
0x80000030	<b>E</b> 3	34	0.8	0p	EA	AD.	1.4	BC	#5	90	10	b6	4.A.	81	p5	D9	
0800000810	50	#5	48	BA	50	45	85	80	DB.	8D	98	78	BD	43	51	E.H.	1.8.18
04000000880	RA.	AC.	BD	18	11	85	73	78	0.5	13	30	GA .	36	83	85	07	++++++
0x800000AD	81	73	DB	17.4	.41	CC	56	410	D1	95	R.E.	D8	2.9	08	50	24	. #1. A
0x800000080	Br	AE:	41	BA	66	44	DD	97	28	82	DE	01	24	50	3 E	58	A.fJ.
ривааааасср	4A	18	86	11	38	18	78	0.5	81	E-4	31	48	61	BC .	90	88	Januar
0×60000000	DA.	08	80	CR	58	30	34	9C	5A	63	4C	94	34	AC	19	AC	
0x8000008x0	6D	TD.	00	A3	18	AIT	76	10	#2	18	AB	AB	77	18	51	CB	B
DRSCOODOSD	6D	68	CB	62	71	16	28	De	0C	72	73	96	07	38	318	22	nk.bq.
4																100	The second

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]