



嵌入式微處理器

SPCE3200 原理及應用

V1.1 - Aug 29, 2007

繁體版

Important Notice

SUNPLUS TECHNOLOGY CO. reserves the right to change this documentation without prior notice. Information provided by SUNPLUS TECHNOLOGY CO. is believed to be accurate and reliable. However, SUNPLUS TECHNOLOGY CO. makes no warranty for any errors which may appear in this document. Contact SUNPLUS TECHNOLOGY CO. to obtain the latest version of device specifications before placing your order. No responsibility is assumed by SUNPLUS TECHNOLOGY CO. for any infringement of patent or other rights of third parties which may result from its use. In addition, SUNPLUS products are not authorized for use as critical components in life support systems or aviation systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Sunplus.

修訂記錄

版本	日期	作者	修訂內容	對應頁碼
V1.1	2007/08/29		1. 增加序、前言； 2. 內容更正。	
V0.8	2007/04/25		初始版本	

前 言

SPCE3200 是臺灣凌陽科技推出以 S+core7 (凌陽科技自主研發) 為內核的 32 位元嵌入式開發系統，內嵌 12 位 ADC、16 位 DAC；具有 UART、SPI、I2C、SIO、USB 等標準硬體控制器介面；具有 6 個 Timer、即時時鐘和時基；有 Nor 型 Flash、Nand 型 Flash、SD 卡控制器；具有 TFT、STN 型 LCD 控制器及 TV 控制器介面；有 MPEG4 編碼譯碼器、CMOS 介面單元等資源。

SPCE3200 豐富的硬體資源及多媒體特性使其獨具特色，由於大部分的功能由硬體完成，所以實現功能時軟體需要較少的工作；同時凌陽科技還研發了凌陽 32 位集成開發環境 S+core IDE，此工具可在 Window 環境下操作，具有友好的操作介面，支援標準 C 語言、凌陽 32 位元嵌入式開發系統組合語言等，集編輯、編程、仿真等功能於一體，使得 SPCE3200 的開發更加容易方便。

本書主要介紹 SPCE3200 的內核 S+core7 體系結構、指令系統，SPCE3200 嵌入式開發系統插腳及 ADC、UART 等基礎功能模組的工作原理、操作暫存器及基本操作，SPCE3200 開發板系統及其集成開發環境 S+core IDE，最後以一個開發實例介紹了 SPCE3200 的開發方法。全書共分為 7 章：

第 1 章介紹 S+core7 的體系結構，分別從處理器模式、內部暫存器、異常、緩存等方面介紹。

第 2 章介紹 S+core7 的指令系統，介紹了 S+core7 處理器的指令結構和指令編碼，詳細介紹了 S+core7 處理器的兩種指令集：32 位元指令集和 16 位元指令集。

第 3 章介紹 SPCE3200 的使用，介紹了 SPCE3200 的特性、插腳、結構、記憶體對映、中斷控制器、APBDMA、啓動代碼等使用基礎知識。

第 4 章介紹 SPCE3200 的 15 個基礎功能模組，包括 GPIO、TIMER、即時時鐘、時基、看門狗、睡眠喚醒、ADC、UART、SPI、I2C、SIO、Nor 型 Flash、Nand 型 Flash、SD 卡控制器、TFT 型 LCD 控制器的特性、工作原理、操作暫存器，並介紹了基本的操作方法。

第 5 章介紹 SPCE3200 的開發板和集成開發環境 S+core IDE 的使用方法。

第 6 章是一個開發實例，分別從軟硬體介紹 SPCE3200 的開發方法。

第 7 章是附錄，包括常用術語、本書的縮寫和約定解釋、S+core7 內核暫存器速查表、SPCE3200 基礎功能模組硬體暫存器速查表和 S+core7 的虛擬指令速查表。

全書由北陽電子技術有限公司大學計畫處羅亞非處長統一審稿，由大學計畫處技術支持部劉學、封鴻雁、李健編寫。在編寫過程中，臺灣凌陽科技 Jacky Hsiao、Rex Wu 及相關人員給予技術支援和指導，在此致以誠摯的謝意。

由於編者水準有限，書中難免錯誤紕漏，請廣大讀者批評指正。

編者

2007-5-31

目錄

1.1 S+core7 簡介	10
1.2 資料類型	10
1.3 處理器模式	11
1.4 內部暫存器	11
1.4.1 概述	11
1.4.2 通用暫存器 (GPR)	12
1.4.3 Custom Engine 暫存器 (CEH/CEL)	13
1.4.4 特殊功能暫存器 (Sm)	13
1.4.5 控制暫存器 (CR)	13
1.5 異常	21
1.5.1 異常處理流程	21
1.5.2 異常優先順序	22
1.5.3 異常原因	25
1.5.4 異常向量	25
1.6 各種異常描述	26
1.6.1 重設異常	26
1.6.2 NMI 中斷異常	27
1.6.3 位址錯誤異常	27
1.6.4 匯流排錯誤異常	27
1.6.5 陷阱異常	28
1.6.6 系統調用 (SYSCALL) 異常	28
1.6.7 P-EL 異常	28
1.6.8 未定義指令 (RI) 異常	28
1.6.9 控制器或協同處理器不可用 (CCU) 異常	29
1.6.10 Custom Engine 執行 (CeE) 異常 (除 0 引起)	29
1.6.11 協同處理器 z 執行 (CpE) 異常	29
1.6.12 中斷異常	30
1.6.13 Debug 中斷異常	30
1.6.14 單步除錯異常	30
1.6.15 斷點除錯異常	31
1.6.16 資料斷點除錯異常	31
1.6.17 指令斷點除錯異常	31
1.7 緩存簡介	31
1.7.1 指令 Cache	33
1.7.2 資料 Cache	33
1.7.3 存儲一致性	34

1.8 LIM 和 LDM	35
1.8.1 LIM-Local Instruction Memory	35
1.8.2 LDM-Local Data Memory	35
1.9 片上除錯	35
2.1 概述	37
2.2 指令格式與編碼	37
2.3 32 位指令集	41
2.3.1 裝載與存儲指令	41
2.3.2 資料處理指令	49
2.3.3 分支指令	61
2.3.4 特殊指令	63
2.3.5 協同處理器指令	69
2.4 16 位指令集	71
2.4.1 裝載與存儲指令	72
2.4.2 資料處理指令	73
2.4.3 跳越與分支指令	75
2.4.4 特殊指令	76
2.4.5 並行條件執行	76
2.5 合成指令集	76
2.6 S+core7 處理器的 GNU 編譯器	81
2.6.1 S+core7 C 編譯器參數	81
2.6.2 S+core7 C 編譯器的基本資料類型	82
2.6.3 S+core7 C 編譯器的函數調用約定	83
2.7 S+core7 處理器的 GNU 組譯器	85
2.7.1 S+core7 C 組譯器參數	85
2.7.2 組合語言語法	86
2.7.3 組譯器偽指令	86
2.7.4 區段及其重定位	89
2.8 S+core7 處理器的 GNU 鏈接器	90
3.1 SPCE3200 簡介	91
3.1.1 概述	91
3.1.2 特性	91
3.2 SPCE3200 插腳信息	92
3.2.1 插腳分佈	92
3.2.2 插腳描述	94
3.3 結構概述	101
3.4 記憶體對映	102
3.5 鎮相環 PLL 與時鐘發生器 CKG	104
3.5.1 鎮相環 PLL	104

3.5.2 時鐘發生器 CKG.....	106
3.5.3 暫存器描述	106
3.5.4 系統時鐘調整.....	111
3.6 中斷控制器	115
3.6.1 概述.....	115
3.6.2 特性.....	116
3.6.3 中斷源	116
3.6.4 結構框圖.....	118
3.6.5 暫存器描述	119
3.6.6 中斷機制.....	127
3.6.7 應用舉例.....	135
3.7 記憶體介面單元——MIU	136
3.8 APB 匯流排 DMA	141
3.9 啟動代碼	151
3.9.1 檔組成	151
3.9.2 *_Prog.Id	151
3.9.3 *_startup.s.....	152
3.9.4 啟動代碼工作流程.....	154
4.1 通用 I/O 埠——GPIO	155
4.1.1 概述.....	155
4.1.2 插腳描述.....	159
4.1.3 結構.....	159
4.1.4 暫存器描述	160
4.1.5 基本操作.....	165
4.2 計時器——TIMER.....	165
4.2.1 概述.....	165
4.2.2 特性.....	165
4.2.3 插腳描述.....	165
4.2.4 結構.....	166
4.2.5 暫存器描述	166
4.2.6 基本操作.....	174
4.2.7 注意事項.....	184
4.3 即時時鐘——RTC	184
4.3.1 概述.....	184
4.3.2 特性.....	184
4.3.3 暫存器描述	184
4.3.4 基本操作.....	190
4.3.5 應用舉例.....	190
4.4 時基——Time Base	191

4.4.1 概述	191
4.4.2 結構	191
4.4.3 暫存器描述	191
4.4.4 基本操作	196
4.4.5 應用舉例	196
4.5 看門狗——WDOG	197
4.5.1 概述	197
4.5.2 特性	197
4.5.3 結構	198
4.5.4 暫存器描述	198
4.5.5 基本操作	201
4.5.6 注意事項	202
4.6 睡眠與喚醒	202
4.6.1 睡眠	202
4.6.2 睡眠相關暫存器	202
4.6.3 喚醒	204
4.6.4 鍵喚醒相關暫存器	208
4.6.5 應用舉例	209
4.7 ADC	211
4.7.1 概述	211
4.7.2 特性	211
4.7.3 插腳描述	211
4.7.4 結構框圖	212
4.7.5 暫存器描述	213
4.7.6 基本操作	226
4.7.7 注意事項	231
4.8 UART	231
4.8.1 概述	231
4.8.2 特性	231
4.8.3 插腳描述	232
4.8.4 結構框圖	232
4.8.5 暫存器描述	233
4.8.6 基本操作	244
4.8.7 注意事項	250
4.9 SPI	250
4.9.1 概述	250
4.9.2 特性	250
4.9.3 插腳描述	251
4.9.4 結構框圖	251

4.9.5 SPI 描述	252
4.9.6 暫存器描述	255
4.9.7 基本操作	261
4.9.8 注意事項	264
4.10 I2C	264
4.10.1 概述	264
4.10.2 特性	264
4.10.3 插腳描述	265
4.10.4 I2C 描述	266
4.10.5 暫存器描述	272
4.10.6 基本操作	280
4.10.7 注意事項	283
4.11 SIO 控制器	283
4.11.1 概述	283
4.11.2 特性	284
4.11.3 插腳描述	284
4.11.4 結構	285
4.11.5 暫存器描述	285
4.11.6 基本操作	293
4.11.7 注意事項	299
4.12 Nor 型 Flash 控制器	300
4.12.1 概述	300
4.12.2 特性	308
4.12.3 插腳描述	308
4.12.4 暫存器描述	310
4.12.5 基本操作	314
4.13 Nand 型 Flash 控制器	319
4.13.1 概述	319
4.13.2 特性	322
4.13.3 插腳描述	322
4.13.4 結構	323
4.13.5 暫存器描述	323
4.13.6 基本操作	339
4.14 SD 卡控制器	343
4.14.1 概述	343
4.14.2 特性	345
4.14.3 插腳描述	345
4.14.4 結構	346
4.14.5 暫存器描述	346

4.14.6 基本操作.....	355
4.15 TFT LCD 控制器	362
4.15.1 TFT LCD 概述	362
4.15.2 特性.....	364
4.15.3 插腳描述.....	364
4.15.4 暫存器描述	365
4.15.5 基本操作.....	384
5.1.1 SPCE3200 實驗儀.....	386
5.1.2 功能特點.....	386
5.1.3 硬體原理.....	387
5.2 S+core IDE 集成開發環境.....	390
5.2.1 工程的編輯	391
5.2.2 工程的設置	402
5.2.3 工程的除錯	407
5.3 應用舉例	420
6.1 原理概述	428
6.2 應用分析	428
6.3 硬體電路	429
6.4 程式設計	430
6.4.1 主程式	430
6.4.2 軟體 FIFO 管理程式.....	430
6.4.3 UART 收發程式	431
6.4.4 RTC 控制及日期計算程式.....	432
6.4.5 Nor 型 Flash 操作程式.....	434
6.4.6 命令獲取和分配程式.....	435
6.4.7 命令處理程式.....	436
7.1 常用術語、縮寫和約定解釋	440
7.1.1 術語.....	440
7.1.2 縮寫.....	442
7.1.3 約定.....	444
7.2 CPU 內核暫存器速查表	445
7.3 硬體模組暫存器速查表.....	447
7.4 S+core7 指令速查表	457
7.4.1 32 位指令速查表.....	457
7.4.2 16 位指令速查表.....	461
7.5 偽指令速查表	463

1 S+core7 體系結構

1.1 S+core7 簡介

S+core7 微處理器是採用凌陽指令集架構 (Sunplus ISA) 的 32 位的 RISC 處理器，該微處理器架構支援 32 位/16 位混合指令模式以及並行條件執行 (正在申請專利保護)，從而提高了代碼密度、性能，使 S+core7 內核得到了廣泛的應用。在 S+core7 微處理器中採用了 AMBA 匯流排，為 SOC 集成、擴展協同處理器和用戶介面提供了靈活性，S+core7 使用 SJTAG 技術使程式的測試和除錯更加有效。

S+core7 微處理器具有如下特徵：

- 支援 32 位與 16 位混合指令模式
- 支援並行條件執行
- 提供軟體安全設計
- 採用哈佛 (Harvard) 結構，包含 I-Cache (4K) 和 D-Cache (4K)
- 採用 Fixed-MMU (固定對映模式)
- 採用 AMBA 匯流排規格，可以方便的實現 Soc 集成
- 63 個硬體中斷，2 個軟體中斷，中斷採用中斷向量
- 採用 SJTAG 協定

1.2 資料類型

S+core7 處理器支援以下幾種類型的資料：

- 位元組 (Byte) : 8 位
- 半字 (Halfword) : 16 位
- 字 (Word) : 32 位

表 1-1列出了以上各種資料類型對應的資料範圍。

表 1-1 資料類型

資料類型		範圍	備註
位元組	有符號數	$-2^7 \sim +2^7 - 1$	
	無符號數	$0 \sim +2^8 - 1$	
半字	有符號數	$-2^{15} \sim +2^{15} - 1$	必須半字邊界對齊
	無符號數	$0 \sim +2^{16} - 1$	
字	有符號數	$-2^{31} \sim +2^{31} - 1$	必須字邊界對齊

所有的資料處理操作都是以“字”為單位的，例如 ADD 操作。Load/Store 操作可在暫存器和記憶體之間傳送位元組、半字和字的資料，但是當位元組或半字類型的資料被裝載時，會自動進行零擴展或符號擴展。

32 位指令正好是一個字，是字（4 位元組）邊界對齊的，16 位指令正好是一個半字，是半字（2 位元組）邊界對齊的。

1.3 處理器模式

當前，幾乎所有 32 位微處理器在實際應用中都要運行作業系統，為在內核級給作業系統提供支援，微處理器可以在多種模式下運行。

S+core7 微處理器支援三種處理器模式：用戶模式（User Mode）、核心模式（Kernel Mode）、除錯模式（Debug Mode）。

1. 用戶模式

用於執行應用程式或作業系統程式。通常情況下，處理器均處於用戶模式，直到發生異常，處理器切換到核心模式。處理器處於用戶模式時，用戶不能存取被系統保護的資源。

2. 核心模式

該模式是作業系統專用的模式。當處理器透過異常進入核心模式後將一直處於該模式，直到一條從異常中返回的指令（RTE，Return From Exception）被執行。

3. 除錯模式

該模式用於用戶除錯階段。在該模式下，用戶程式可以完全存取用戶模式和核心模式下的暫存器，以及其他一些除錯暫存器。

1.4 內部暫存器

1.4.1 概述

S+core7 處理器包含以下暫存器：

- 32 個通用暫存器（GPR）
- 2 個 Custom Engine 暫存器（CEH、CEL）
- 3 個特殊功能暫存器
 - Sr0：迴圈計數暫存器（CNT）
 - Sr1：裝載合併暫存器（LCR）
 - Sr2：存儲合併暫存器（SCR）
- 19 個系統控制暫存器
- 3 個除錯控制暫存器

處理器運行在用戶模式下，用戶程式可以存取 32 個通用暫存器（GPR）、2 個 Custom Engine 暫存器（CEH、CEL）以及 3 個特殊功能暫存器（CNT、LCR、SCR）。處理器運行在核心模式下，用戶程式除了可存取所有用戶模式下的暫存器外，還可存取 19 個系統控制暫存器。處理器運行在除錯模式下，用戶程式除了可存取所有用戶模式下以及核心模式下的暫存器

外，還可存取 3 個除錯模式下暫存器（DSAVE、DEPC、DREG）。各模式下可存取暫存器參考圖 1-1 所示：

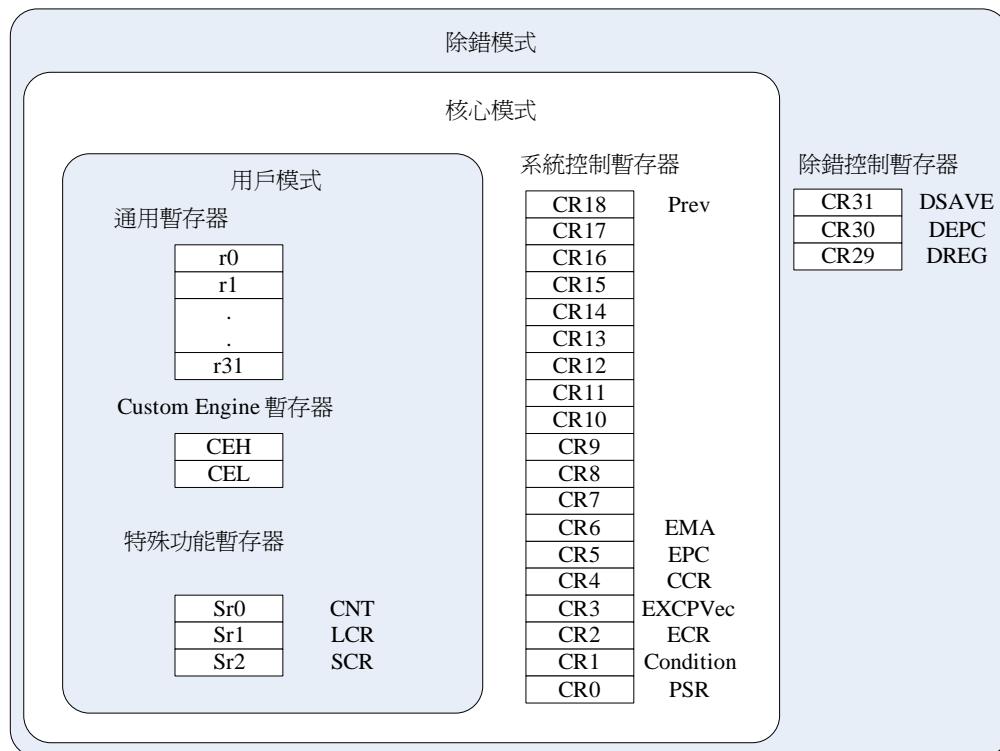


圖 1-1 S+core7 內核暫存器集

1.4.2 通用暫存器（GPR）

S+core7 處理器有 32 個 32 位的通用暫存器（r0~r31）。32 位指令模式下，所有這些通用暫存器均可以被存取。由於指令編碼的限制，16 位指令模式下，只有 r0~r15 可以被存取。在跳越/分支或鏈接指令中，r3 暫存器被用作鏈接暫存器，用於保存下一條指令位址。如圖 1-2 所示：

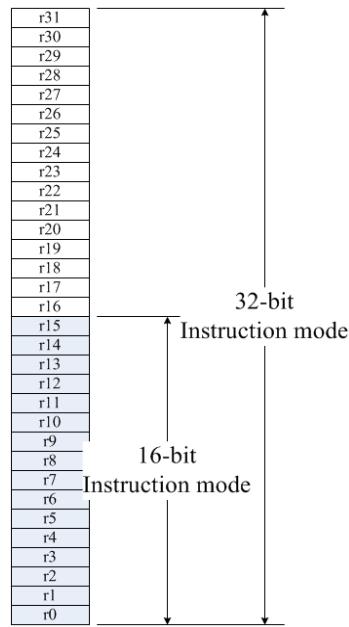


圖 1-2 通用暫存器

1.4.3 Custom Engine 暫存器 (CEH/CEL)

Custom Engine 暫存器包括 CEH 和 CEL 兩個暫存器，用來存儲乘法/除法的運算結果。乘法運算完成後，雙字的運算結果的高字被放到 CEH 暫存器中，低字被放到 CEL 暫存器中。除法運算完成後，餘數放到 CEH 暫存器中，商放到 CEL 暫存器中。

透過 MFCEH、MFCEL、MFCEHL、MTCEH、MTCEL 或 MTCEHL 指令，可實現這兩個暫存器與通用暫存器之間的資料傳送。詳細請參考指令系統一章。

1.4.4 特殊功能暫存器 (Sr_n)

S+core7 處理器有 3 個特殊功能暫存器：CNT (Sr0)、LCR (Sr1)、SCR (Sr2)。CNT 暫存器是一個 32 位暫存器，可用來作迴圈計數。當執行特定的分支指令時，計數減一。例如，執行 bcnz 分支指令時，如果 CNT 暫存器中的值不為零，CNT 暫存器中的值將減 1，程式跳越到目標位址；如果 CNT 暫存器中的值為零，則 bcnz 指令將被視為 nop 指令，CNT 暫存器中的值保持不變。

LCR 暫存器和 SCR 暫存器則是用於存取不對齊的 Load 和 Store 指令操作的。

1.4.5 控制暫存器 (CR)

處理器運行在核心模式下有 19 個系統控制暫存器，在除錯模式下有 3 個除錯控制暫存器。其中有些暫存器沒有被定義，參考表 1-2。

表 1-2 控制暫存器

暫存器名稱	助記符	暫存器編號
程式狀態暫存器	PSR	CR0
條件暫存器	Condition	CR1
異常原因暫存器	ECR	CR2
異常向量暫存器	EXCPVec	CR3
Cache 控制暫存器	CCR	CR4
異常程式計數器	EPC	CR5
異常記憶體位址暫存器	EMA	CR6
-	-	-
LIM 實體框號	LIMPFN	CR15
LDM 實體框號	LDMPFN	CR16
-	-	-
Prev 暫存器	Prev	CR18
Debug 暫存器	DREG	CR29
Debug 異常程式計數暫存器	DEPC	CR30
Debug 異常內容保存暫存器	DSAVE	CR31

■ 程式狀態暫存器 (PSR)

程式狀態暫存器用於指示協同處理器是否可用，中斷遮罩位、大小端及保存處理器的模式等。

表 1-3 程式狀態暫存器 (PSR)

位	b31~b29	b28	b27~b24	b23~b18	b17~b16	b15
讀/寫	R/W	R/W	R/W	R/W	R/W	R
預設值	0	0	0	0	0	1
名稱	CU[2:0]	CRA	-	IM_H[5:0]	IM_S[1:0]	Endian
位	b5	b4	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0

名稱	UMb	IEb	UMs	IEs	UMc	IEc
----	-----	-----	-----	-----	-----	-----

CU[2:0]	b31~b29	CU[n] = 1 : 協同處理器 n+1 是可用的，協同處理器指令可以被執行 CU[n] = 0 : 協同處理器 n+1 是不可用的，協同處理器指令不可以被執行
CRA	b28	0 : 控制暫存器在用戶模式下是不可存取的，控制暫存器指令在用戶模式下不可以被執行 1 : 控制暫存器在用戶模式下是可存取的，控制暫存器指令在用戶模式下可以被執行
-	b27~b24	保留。讀操作時，返回 0；寫操作時，只能寫 0
IM_H[5:0]①	b23~b18	中斷遮罩位元（根據中斷優先順序），對應 63 個硬體中斷
IM_S[1:0]①	b17~b16	中斷遮罩位元（根據中斷優先順序），對應 2 個軟體中斷
Endian	b15	位元組排列方式： 0 : 小端方式 (Little Endian) 1 : 大端方式 (Big Endian)
-	b14~b6	保留。讀操作時，返回 0；寫操作時，只能寫 0。
UMb②	b5	前一次處理器模式的備份： 0 : 核心模式 1 : 用戶模式
IEb②	b4	前一次中斷使能位的備份： 0 : 63 個硬體中斷及 2 個軟體中斷被禁止 1 : 63 個硬體中斷及 2 個軟體中斷被允許
UMs②	b3	前一次處理器模式： 0 : 核心模式 1 : 用戶模式
IEs②	b2	前一次中斷使能位： 0 : 63 個硬體中斷及 2 個軟體中斷被禁止 1 : 63 個硬體中斷及 2 個軟體中斷被允許

UMc②	b1	當前的處理器模式： 0：核心模式 1：用戶模式
IEc②	b0	當前的中斷使能位： 0：63 個硬體中斷及 2 個軟體中斷被禁止 1：63 個硬體中斷及 2 個軟體中斷被允許

注解：

① : IM[7:2]是一個 6 位元的代碼，可編碼成 0, 1, ..., 63 ($2^6 - 1$) 的數值。0 表示所有硬體中斷請求都被允許。n (1~63) 表明優先順序為 n 的以及低於 n 的硬體中斷請求均被遮罩。IM[1:0]是軟體中斷遮罩位元，IM[1]或 IM[0] = 1 表明軟體中斷 1 或 0 的請求被遮罩。

② : UMb、IEb、UMs、IEs、UMc 以及 IEc 六位形成了一個三級的硬體堆疊，用來保存處理器模式 (UM) 和中斷使能 (IE) 的資訊。UMc 和 IEc 保存當前的 UM 和 IE 值；UMs 和 IEs 保存發生異常前的 UMc 和 IEc 值；UMb 和 IEb 是 UMb 和 IEs 的一個備份。例如，異常發生時，硬體先把當前的 UMc 和 IEc 值保存在 UMb 和 IEs 中，然後再重新設置 UMc 和 IEc；執行異常返回指令 RTE 時，UMs 和 IEs 中的值自動拷貝回 UMc 和 IEc。

■ 條件暫存器 (Condition)

條件暫存器指示當前程式運行是否出現溢出、進位 (借位)、零旗標、負旗標及並行條件執行旗標 (T)，分支跳越指令就是根據這些旗標位元判斷程式的流向。

表 1-4 條件暫存器 (Condition)

位	b31~b15	b14	b13	b12	b11	b10	b9	b8
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0	0	0
名稱	-	Tb	Nb	Zb	Cb	Vb	Ts	Ns
位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0	0	0
名稱	Zs	Cs	Vs	Tc	Nc	Zc	Cc	Vc

- b31~b15 保留。讀操作時，返回 0；寫操作時，只能寫 0

Tb b14 前一次的條件並行執行旗標位元 T 的備份

Nb	b13	前一次的負旗標位元 N 的備份
Zb	b12	前一次的零旗標位元 Z 的備份
Cb	b11	前一次的進位元/借位/擴展旗標位元 C 的備份
Vb	b10	前一次的溢出旗標位元 V 的備份
Ts	b9	前一次的條件並行執行旗標位元 T
Ns	b8	前一次的負旗標位元 N
Zs	b7	前一次的零旗標位元 Z
Cs	b6	前一次的進位/借位/擴展旗標位元 C
Vs	b5	前一次的溢出旗標位元 V
Tc	b4	當前的條件並行執行旗標位元 T
Nc	b3	當前的負旗標位元 N
Zc	b2	當前的零旗標位元 Z
Cc	b1	當前的進位元/借位/擴展旗標位元 C
Vc	b0	當前的溢出旗標位元

Tb、Nb、Zb、Cb、Vb、Ts、Ns、Zs、Cs、Vs、Tc、Nc、Zc、Cc 以及 Vc 形成了一個三級的硬體堆疊，用來保存 T、N、Z、C、V 旗標位元資訊。Tc、Nc、Zc、Cc、Vc 保存當前的各旗標位元資訊；Ts、Ns、Zs、Cs、Vs 保存前一次的各旗標位元資訊；Tb、Nb、Zb、Cb、Vb 保存前一次的各旗標位元資訊的備份。

■ 異常原因暫存器 (ECR - Exception Cause Register)

異常原因暫存器指示了引起異常的原因。

表 1-5 異常原因暫存器 (ECR - Exception Cause Register)

位	b31~b24	b23~b18	b17~b16	b15~b8	b7~b6	b5	b4~b0
讀/寫	R/W	R	R/W	R/W	R	R/W	R
預設值	0	0	0	0	0	0	0
名稱	-	IP_H[5:0]	IP_S[1:0]	-	CE[1:0]	-	Exc_code

-	b31~b24	保留。讀操作時，返回 0；寫操作時，只能寫 0
IP_H[5:0] ①	b23~b18	63 個硬體中斷的中斷請求向量
IP_S[1:0]	b17~b16	2 個軟體中斷的中斷請求向量

-	b15~b8	保留。讀操作時，返回 0；寫操作時，只能寫 0
CE[1:0]	b7~b6	當控制器或協同處理器不可用異常發生時，這幾位表明了異常發生的原因： 00：控制暫存器存取異常 01：協同處理器 1 不可用異常 10：協同處理器 2 不可用異常 11：協同處理器 3 不可用異常
-	b5	保留。讀操作時，返回 0；寫操作時，只能寫 0
Exc_code	b4~b0	當中斷發生時，Exc_code 為異常發生原因編碼

注解：

d : IP_H[5:2]是一個 6 位元編碼，指示中斷請求。0 表示沒有中斷請求，n (1~63) 表明優先順序為 n 的中斷請求。IP[1:0]對應軟體中斷，用來設置或取消軟體中斷。

■ 異常向量暫存器 (EXCPVec - Exception Vector Register)

異常向量暫存器保存有所有異常向量位址的基底位址，b0 位指示向量位址的偏移模式選擇位，選擇偏移 0x04 還是 0x10，可以計算出異常向量的位址。

表 1-6 異常向量暫存器(EXCPVec - Exception Vector Register)

位	b31~b16	b15~b1	b0
讀/寫	R/W	R/W	R/W
預設值	0x9F00	0	0
名稱	EXCPVec_Base	-	VO

EXCPVec_Base	b31~b16	所有異常向量位址的基底位址（第 31 位至第 16 位）
-	b15~b1	保留。讀操作時，返回 0；寫操作時，只能寫 0
VO	b0	向量位址的偏移模式選擇位： 0：偏移 0x04 1：偏移 0x10

■ Cache 控制暫存器 (CCR - Cache Control Register)

Cache 控制器控制 Cache 的相關操作。

表 1-7 Cache 控制暫存器 (CCR - Cache Control Register)

位	b16	b9	b8	b7	b6
讀/寫	R/W	R	R	R	R/W
預設值	0	0	0	0	0
名稱	LDM LP	DPFB	IPFB	W-Back	RbBpDis
位	b5	b4	b3	b2	b0
讀/寫	R/W	R/W	R/W	R/W	R
預設值	0	0	0	0	0
名稱	NOP	BTEN	LDM	LIM	WB

-	b31~b17	保留。讀操作時，返回 0；寫操作時，只能寫 0
LDM LP	b16	LDM 低功耗 (LDM Low power) 使能位： 0：禁止（預設） 1：使能
-	b15~b10	保留。讀操作時，返回 0；寫操作時，只能寫 0
DPFB	b9	資料預取暫存器使能位： 0：禁止 1：使能
IPFB	b8	指令預取暫存器使能位： 0：禁止 1：使能
W-Back	b7	資料 Cache 的寫回 (Write-Back) 模式使能位： 0：禁止 1：使能
RbBpDis	b6	寫暫存器的資料讀操作越過寫操作模式的禁止位： 0：禁止 1：使能

NOP	b5	NOP 指令的處理選擇位： 0 : NOP (空操作) 指令正常處理 1 : NOP 指令在管線 (Pipe Line) 中充當 “氣泡”
BTEN	b4	AMBA 汇流排設備支援突發傳輸提前終止 (burst early terminate) 的功能使能位： 0 : 禁止 1 : 允許
LDM	b3	LDM 介面使能位： 0 : 禁止 Local 資料記憶體 (LDM, Local Data Memory) 介面 1 : 允許 Local 資料記憶體 (LDM, Local Data Memory) 介面
LIM	b2	LIM 介面使能位： 0 : 禁止 Local 指令記憶體 (LIM, Local Instruction Memory) 介面 1 : 允許 Local 指令記憶體 (LIM, Local Instruction Memory) 介面
-	b1	保留。讀操作時，返回 0；寫操作時，只能寫 0
WB	b0	寫緩衝使能位： 0 : 禁止 (預設) 1 : 使能

■ 異常程式計數器 (EPC - Exception Program Counter)

異常程式計數器保存程式發生異常時的 PC 程式指標及發生異常時的指令模式。

表 1-8 異常程式計數器 (EPC - Exception Program Counter)

位	b31~b1	b0
讀/寫	R/W	R
預設值	0	0
名稱	EPC	M

EPC	b31~b1	異常發生時，程式計數器 (PC) 的高 31 位有效位元被記錄在此欄位中。執行 RTE 指令後，此欄位中的值將被重載到程式計數器 PC 中
-----	--------	---

M	b0	對於在管線 D 階段 (D Stage) 後產生的異常，該位元表明發生異常的指令模式 (32 位或 16 位)： 0 : 32 位指令模式 1 : PCE 指令或 16 位指令模式
對於在管線 D 階段 (D Stage) 前產生的異常，該位無意義		

引起中斷或異常發生的指令的位址保存在 EPC 暫存器中。其中，Bit0 (M) 表明引起異常發生指令的模式：為 1 表示 PCE 指令或 16 位指令模式；為 0 表示 32 位指令模式。Bit1 (EPC[1]) 表明具體的發生異常的指令位置：為 1 表示低位址的 16 位指令；為 0 表示高位址的 16 位指令或 32 位指令。執行 RTE 指令後，EPC[31:1]的值將被重載到 PC 中，Bit0 忽略。

其他控制暫存器與用戶編寫程式應用關係不大，感興趣的讀者請參考 S+core7 內核資料。

1.5 異常

S+core7 內核最多可以處理 80 個異常，以 S+core7 內核構成的晶片具有強大的中斷處理能力。

1.5.1 異常處理流程

S+core7 微處理器在每執行一條程式前先判斷是否有重設請求，如果有重設請求，會初始化程式狀態暫存器 (PSR)、條件暫存器 (Condition)、異常向量暫存器 (EXCPvec)、CCR 暫存器以及程式計數暫存器 PC；如果沒有重設請求，會執行下一條指令，如果有異常請求，會將相關暫存器進行保存後轉入異常程式處理；如果沒有異常請求，處理器直接判斷是否有重設請求等重複這個過程。

根據異常的優先順序，所有異常均在管線的 M 階段 (Memory Stage) 被識別處理。在執行異常處理後，執行 RTE 指令可以使程式從異常處理中返回。執行 RTE 指令後，PC 指標將指向 EPC 暫存器中保存的位址，且部分 PSR 暫存器或條件暫存器的內容將被右移(彈出堆疊)，CPU 即從 EPC 暫存器保存的位址處執行程式。參考圖 1-3所示：

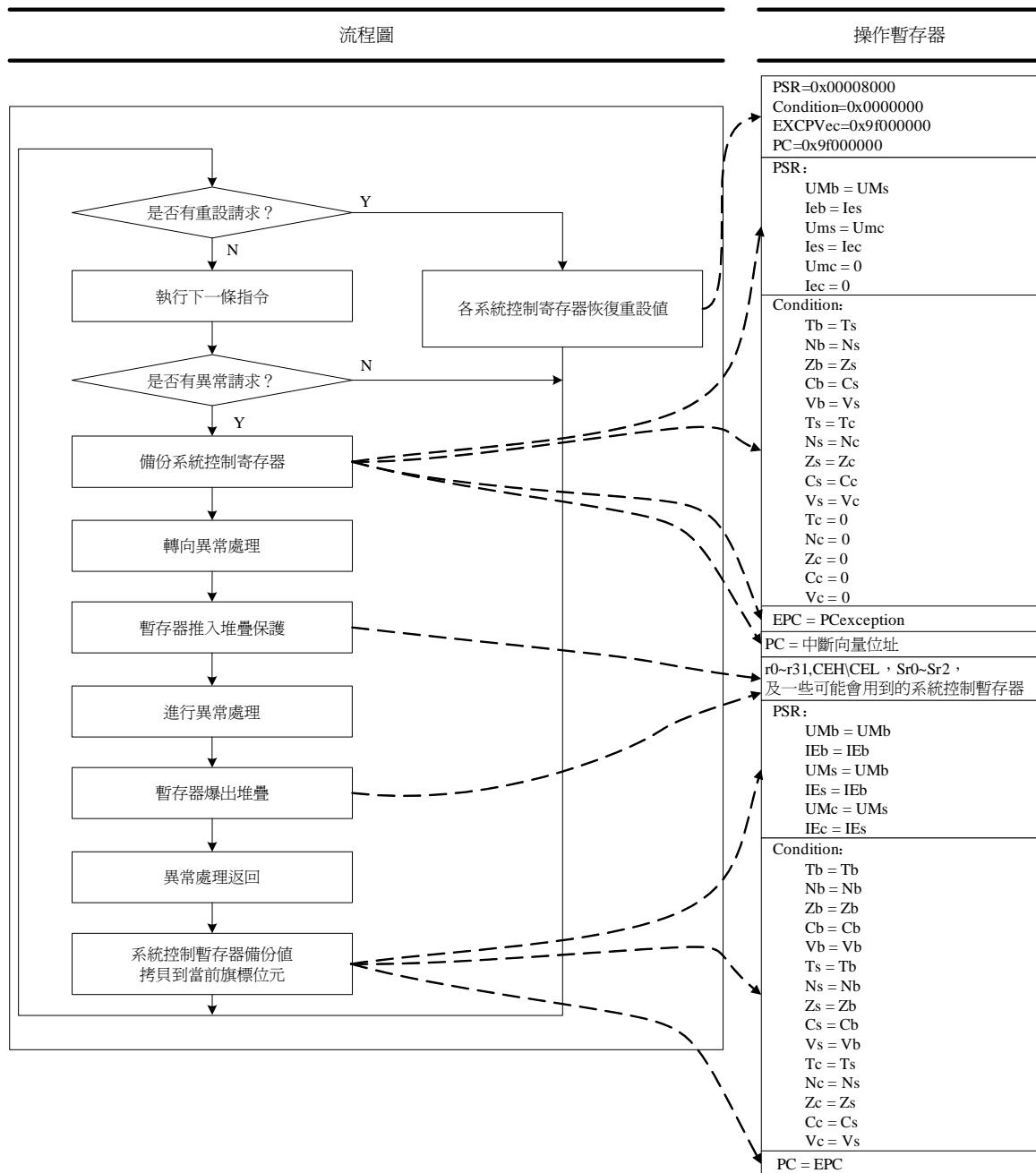


圖 1-3 異常處理流程

1.5.2 異常優先順序

正常模式下，在一條指令執行過程中，可能有多個異常發生，但是根據異常優先順序只能有一個異常向 CPU 發出申請。

表 1-9 異常優先順序

優先順序	異常	描述
1 最高	Reset	重設異常
2	DSS	單步除錯異常 (Debug Single Step)
3	DINT	Debug 汇流排異常或 JTAG 中斷異常
4	DDBLV	執行 Load 操作時發生的資料斷點除錯異常 (位址資料均匹配)
5	NMI	不可遮罩中斷 (Non-maskable Interrupt)
6	Interrupt[63]	外部硬體中斷異常
:	:	:
:	:	:
68	Interrupt[1]	外部硬體中斷異常
69	DIB	指令斷點除錯異常
70	AdEL-instruction	取指位址錯誤異常
71	BusEL-instruction	取指匯流排錯誤異常
72	P-EL	指令中的 P 位檢查錯誤異常
73	DBP	除錯斷點異常 (SDBBP)
73	SYSCALL	系統調用陷阱異常
73	CCU	控制器或協同處理器不可用異常
73	RI	未定義指令異常
73	Trap	條件陷阱異常
74	DDBLA	執行 Load 操作時發生的資料斷點除錯異常
75	DDBS	執行 Store 操作時發生的資料斷點除錯異常
76	AdEL-Data	資料裝載位址錯誤異常
77	AdES	資料存儲位址錯誤異常
78	CeE	Custom Engine 執行異常 (除 0 引起的)
78	CpE	協同處理器執行異常
79	BusEL-data	資料存取匯流排錯誤異常
80 最低	SWI[1]	內部軟體中斷異常 1
80 最低	SWI[2]	內部軟體中斷異常 2

在 Debug 模式下，Debug 異常指令以及與中斷相關的指令將被禁止。表 1-10列出了 Debug 模式下各種異常會引起的處理器行爲。其中有些異常將導致系統再進入 Debug 模式（可以根據 DExcCode 來判斷異常原因）

表 1-10 Debug 模式下異常及其優先順序

優先順序	異常	處理器行爲
1 最高	重設異常	系統重設
2	單步除錯異常 (DSS)	被禁止
3	Debug 中斷異常 (DINT)	被禁止
4	執行 Load 操作時發生的資料斷點調試異常 (DDBLV) 位址資料均匹配	被禁止
5	不可遮罩中斷異常 (NMI)	被禁止
6	外部硬體中斷異常 (Interrupt)	被禁止
7	指令斷點除錯異常 (DIB)	被禁止
8	取指位址錯誤異常	再進入 Debug 模式
9	取指匯流排錯誤異常	再進入 Debug 模式
10	指令中的 P 位檢查錯誤異常	再進入 Debug 模式
11	斷點除錯異常 (SDBBP)	被禁止
12	條件陷阱異常	再進入 Debug 模式
13	系統調用陷阱異常	再進入 Debug 模式
14	控制器或協同處理器不可用異常	再進入 Debug 模式
15	未定義指令異常	再進入 Debug 模式
16	執行 Load 操作時發生的資料斷點除錯異常 (DDBLA)，僅位址匹配	被禁止
17	執行 Store 操作時發生的資料斷點除錯異常 (DDBS)	被禁止
18	資料裝載位址錯誤異常	再進入 Debug 模式
19	資料存儲位址錯誤異常	再進入 Debug 模式
20	Custom Engine 執行異常	再進入 Debug 模式
21	協同處理器執行異常	再進入 Debug 模式

優先順序	異常	處理器行為
22	資料存取匯流排錯誤異常（精確的）	再進入 Debug 模式
23	資料存取匯流排錯誤異常（不精確的）	被禁止
24 最低	內部軟體中斷異常	被禁止

1.5.3 異常原因

引起異常可以有各種原因，如硬體中斷、軟體中斷、重設、匯流排異常等。引起異常的原因對應的編碼，參考表 1-11。

表 1-11 異常原因編碼

異常	編碼	異常	編碼
Reset	0	AdEL-Data	11
NMI	1	AdES	12
AdEL-Instruction	2	-	13
-	3	-	14
-	4	-	15
BusEL-Instruction	5	CeE	16
P-EL	6	CpE	17
SYSCALL	7	BusEL-Data	18
CCU	8	SWI	19
RI	9	Interrupt	20
Trap	10		

1.5.4 異常向量

表 1-12 異常向量表

異常類型	向量位址		異常
	VO = 0	VO = 1	
重設異常	0x9F00_0000	0x9F00_0000	重設異常
Debug 異常	Badr + 0x1FC	Badr + 0x1FC	Debug 異常

異常類型	向量位址		異常
	VO = 0	VO = 1	
一般異常	Badr + 0x200	Badr + 0x200	不可遮罩中斷異常 (NMI)
			取指位址錯誤異常 (AdEL-Instruction)
			取指匯流排錯誤異常 (BusEL-Instruction)
			P 位元錯誤異常 (P-EL)
			系統調用異常 (SYSCALL)
			控制器或協同處理器不可用異常 (CCU)
			未定義指令異常 (RI)
			陷阱異常 (Trap)
			資料裝載位址錯誤異常 (AdEL-Data)
			資料存儲位址錯誤異常 (AdES)
			Custom Engine 執行異常 (CeE)
			協同處理器執行異常 (CpE)
			資料存取匯流排錯誤異常 (BusEL-Data)
			內部軟體中斷異常 (SWI1)
			內部軟體中斷異常 (SWI2)
中斷異常	Badr + 0x204	Badr + 0x210	外部硬體中斷[1]
	.	.	.
	.	.	.
	Badr + 0x2FC	Badr + 0x5F0	外部硬體中斷[63]

1.6 各種異常描述

1.6.1 重設異常

起因：一旦重設信號產生，總是會發生重設異常。重設異常是不可遮罩的。

CPU 為重設異常提供一個特殊的重設向量 (0x9F00_0000)，此向量處於非 Cache 區。進入重設異常處理後，CPU 將處於固定對映 (Fixed Mapping) 模式。

重設異常發生後：

- 程式狀態暫存器 (PSR) 中的 IEc 與 UMc 被置 0；

- CCR 暫存器中的 WB 欄位被置 0；
- ECR 暫存器中的 Exc_code 欄位被置 0；
- EXCVEC 暫存器中的 EXCVec_Base 欄位被置為 0x9F00_0000。
其他暫存器的值均未作定義。

1.6.2 NMI 中斷異常

起因：NMI 插腳電平處於下降沿時，會發生 NMI 中斷（Non-Maskable Interrupt）異常，而不管 PSR 暫存器中的 IEc 位是否被置 1。NMI 中斷異常是不可恢復的異常。

NMI 中斷異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值將分別壓入 IEs 與 Ums 位中；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 1；
- EPC 暫存器指向引起發生 NMI 中斷異常的指令。

其他暫存器的值均不會改變。

1.6.3 位址錯誤異常

起因：裝載或存儲記憶體中的一個字時，不是字邊界對齊的；裝載或存儲記憶體中的一個半字時，不是半字邊界對齊的；指令位址不是半字邊界對齊的（指令位址的最低位是+1）；用戶模式下存取了核心模式下的位址；用戶模式或核心模式下存取了 Debug 模式的位址。

位址錯誤異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位被壓入 IEs 與 Ums ；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位為 2 (AdEL-Instruction 異常)、11 (AdEL-Data 異常) 或者 12 (AdES-data 異常)；
- EPC 暫存器指向引起位址錯誤異常的指令；
- 異常存儲位址 (EMA) 暫存器保存沒有正確對齊的或者被保護的位址空間的虛擬位址；
- EMA 暫存器和 PEVN 暫存器保存沒有正確轉換的虛擬位址。

其他暫存器的值均不會改變。

1.6.4 匯流排錯誤異常

起因：當實體電路產生 Event (例如，匯流排存取超時、背板匯流排 (backplane bus) P 位元錯誤、無效的實體存儲位址或者無效的存取類型信號時，會引起匯流排錯誤異常。

匯流排錯誤異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位元為 5 (取指令時發生的異常) 或者 18 (取資料時發生的異常)；

- EPC 暫存器指向引起匯流排錯誤異常的指令。注意：如果處理器中有寫暫存器（Write-Buffer）並已將其使能，那麼，資料匯流排錯誤異常發生時，EPC 暫存器將不能正確地保存指令指標，從而該異常將不能被恢復。

其他暫存器的值均不會改變。

1.6.5 陷阱異常

起因：執行 Trap 指令，會引起陷阱異常。

陷阱異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 10；
- EPC 暫存器指向 Trap 指令。

其他暫存器的值均不會改變。

1.6.6 系統調用（SYSCALL）異常

起因：執行 SYSCALL 指令，會引起系統調用陷阱異常。

SYSCALL 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 7；
- EPC 暫存器指向 SYSCALL 指令。

其他暫存器的值均不會改變。

1.6.7 P-EL 異常

起因：執行一條 P 位元檢查錯誤指令，會引起 P-EL 異常；指令位址（如分支目標位址）指向一個 32 位指令字的低半字，但其是一條 32 位指令，即指令位址字對齊錯誤（Bit1 = 1）。

P-EL 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 6；
- EMA 暫存器指向引起 P-EL 中斷異常的指令；
- EPC 暫存器指向引起 P-EL 中斷異常的指令。

其他暫存器的值均不會改變。

1.6.8 未定義指令（RI）異常

起因：執行一條操作碼（Opcode）未被定義的指令，會引起 RI 異常。

RI 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 9；
- EPC 暫存器指向引起該異常的指令。

其他暫存器的值均不會改變。

1.6.9 控制器或協同處理器不可用 (CCU) 異常

起因：執行如下指令操作中的任一種時，會引起 CCU 異常。

在用戶模式下，執行控制暫存器指令，但控制暫存器沒有旗標為可用狀態 (PSR 暫存器 CRA 位元為 0)；執行協同處理器指令，但該協同處理器沒有標識為可用狀態 (PSR 暫存器 CU 位元為 0)。

CCU 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 8；
- ECR 暫存器中的 CE 欄位表明具體是哪個協同處理器或控制器不可用；
- EPC 暫存器指向引起 CCU 異常的指令。

其他暫存器的值均不會改變。

1.6.10 Custom Engine 執行 (CeE) 異常 (除 0 引起)

起因：Custom Engine 執行一條擴展指令時，執行結果可能會引起 CeE 異常。例如，除 0、乘累加溢出。這裏，定義除 0 引起的異常為預設的 CeE 異常。

CeE 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 16；
- EPC 暫存器指向引起 CeE 異常的指令。

其他暫存器的值均不會改變。

1.6.11 協同處理器 z 執行 (CpE) 異常

起因：協同處理器 z 試圖執行一條協同處理器操作指令時，執行結果可能會引起 CpE 異常。例如，浮點除 0、浮點乘累加溢出。

CpE 異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位被置 17；

- ECR 暫存器中的 CE 欄位表明具體是哪個協同處理器引起的異常；
- EPC 暫存器指向引起 CeE 異常的指令。
其他暫存器的值均不會改變。

1.6.12 中斷異常

起因：當 65 個中斷中的任一個發生時，會引起中斷異常。

中斷異常發生後：

- PSR 暫存器中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- ECR 暫存器中的 Exc_code 欄位為 19 (表明是軟體中斷引起的異常) 或 20 (表明是硬體中斷引起的異常)；
- EPC 暫存器指向引起中斷異常的指令。運行完中斷服務程式後，程式會返回到 EPC 暫存器所指的位址。

其他暫存器的值均不會改變。

注意：

在分配外部中斷 (63 個) 優先順序之前，中斷控制器必須先利用處理器時鐘去擷取外部中斷信號並將其雜訊信號去抖，否則這些雜訊信號可能會引發處理器錯誤地識別中斷。

1.6.13 Debug 中斷異常

起因：非 Debug 模式下，Debug 中斷會引發該異常。

Debug 中斷異常發生後：

- DREG 暫存器 (CR29) 中的 DINT 以及 DM 位被置位；
- DEPC 暫存器 (CR30) 保存異常處理完成後需要恢復的位址。執行 DRTE 指令，可使程式跳越到該暫存器中所指的位址。

其他暫存器的值均不會改變。

1.6.14 單步除錯異常

起因：非 Debug 模式下，當一條指令單步執行完並且 Debug 暫存器中的 SSEn 位被置位後，會發生單步除錯異常。

單步除錯異常發生後：

- DREG 暫存器 (CR29) 中的 DSS 以及 DM 位被置位；
- DEPC 暫存器 (CR30) 保存異常處理完成後需要恢復的位址。執行 DRTE 指令，可使程式跳越到該暫存器中所指的位址。

其他暫存器的值均不會改變。

1.6.15 斷點除錯異常

起因：執行 SDBBP (Software Debug Breakpoint) 指令後，會發生斷點除錯異常。

斷點除錯異常發生後：

- 如果該異常發生在非 Debug 模式下，DREG 暫存器 (CR29) 中的 DBP 位被置位；
- 如果該異常發生在 Debug 模式下，DSS、DBP、DDBL、DDBS、DIB、DDB 和 DINT 這些位被清除，在 CR29 暫存器中 DexcCode 被設置成 SDBBP；
- DREG 暫存器 (CR29) 中的 DM 位被置位；
- DEPC 暫存器 (CR30) 保存異常處理完成後需要恢復的位址。執行 DRTE 指令，可使程式跳越到該暫存器中所指的位址。

其他暫存器的值均不會改變。

1.6.16 資料斷點除錯異常

起因：非 Debug 模式下，執行裝載或存儲指令時，若被存取的資料的位址匹配，則會發生資料斷點除錯異常。

資料斷點除錯異常發生後：

- DREG 暫存器 (CR29) 中的 DDBLA、DDBLV 或 DDBS 位被置位；
- DREG 暫存器 (CR29) 中的 DM 位被置位，其他暫存器的值均不會改變。
- DEPC 暫存器 (CR30) 保存異常處理完成後需要恢復的位址。執行 DRTE 指令，可使程式跳越到該暫存器中所指的位址。

1.6.17 指令斷點除錯異常

起因：非 Debug 模式下，若指令的位址匹配，則會發生指令斷點除錯異常。

指令斷點異常發生後：

- DREG 暫存器 (CR29) 中的 DIB 以及 DM 位被置位；
- DEPC 暫存器 (CR30) 保存異常處理完成後需要恢復的位址。執行 DRTE 指令，可使程式跳越到該暫存器中所指的位址。

其他暫存器的值均不會改變。

1.7 緩存簡介

S+core77 處理器的主頻最高達到 162MHz，而主記憶體操作用動態記憶體 (DRAM)，其存儲週期僅為 100ns~200ns。這樣，如果指令和資料都存儲在主記憶體中，主記憶體的速度將會嚴重制約整個系統的性能。高速緩衝記憶體 (Cache) 和寫緩衝區 (Write buffers) 位於主記憶體和 CPU 之間，可以大大提高存儲系統的性能。Cache 是一塊位址可以改變的高速的記憶體空間，目的在於加快記憶體存取的速度。將 Cache 分成若干塊，Cache-line 是使用 Cache 的最小存儲單元。當 CPU 讀取資料或者指令時，如果當前 cache line 中沒有保存這些資料，cache line 會將這些資料載入。如果在 cache line 載入其他位址的資料之前，CPU 存取相同的資料，那麼 cache 可以提供記憶體存取。

S+core7 處理器支援 2 種獨立的 Cache，指令 Cache (I-Cache) 和資料 Cache (D-Cache)。這種獨立的 Cache 結構可使指令和資料同時得到處理。而 I-Cache 和 D-Cache 均是使用虛擬位址來索引的 (virtual indexed)，用實體位址來標識的 (physically tagged)。

表 1-13 S+core7 處理器 Cache 規格

規格	I-Cache	D-Cache
Cache 大小	4Kbytes	4Kbytes
組相連	2	2
Cache-line 大小	4Words	4Words
寫策略	NA	寫入 (Write Through) 或寫回 (Write Back)
分配策略	讀分配策略 (Read Allocate)	讀分配策略 (Read Allocate)
替換策略	最近最少使用策略 (LRU)	最近最少使用策略 (LRU)
預取一個 Cache-line	虛擬位址模式	虛擬位址模式
預取並鎖定一個 Cache-line	虛擬位址模式	虛擬位址模式
使一個 Cache-line 無效	虛擬位址模式	虛擬位址模式
使整個 Cache 無效	可以	可以
清空寫緩衝區	NA	可以
寫緩衝區	NA	4 words

S+core7 處理器中，Tag 存儲陳列是以 Cache-line 為單位標記資料的；資料是以 Word 為單位存儲的。I-Cache 和 D-Cache 均是 2 路組相連的結構，大小均為 4K Bytes。

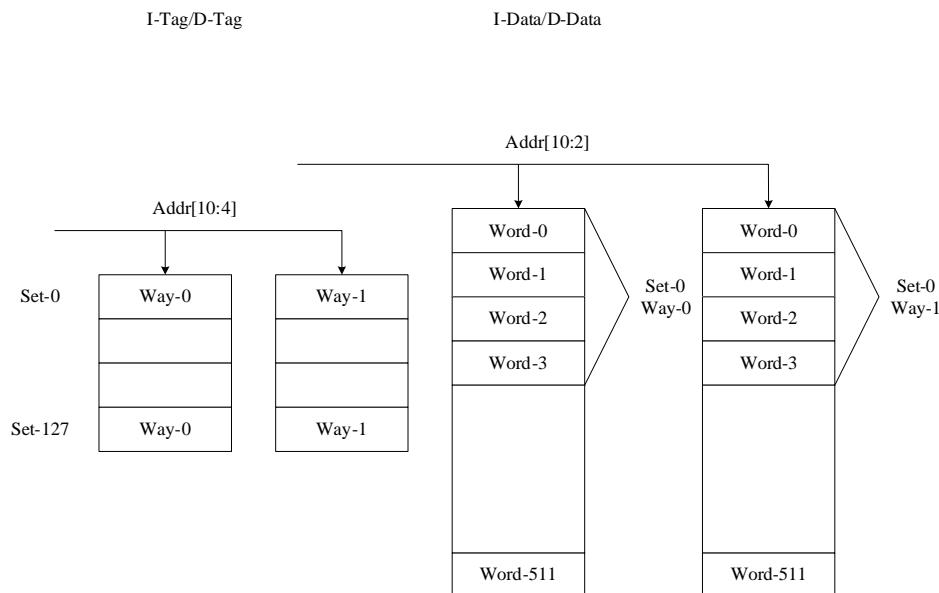


圖 1-4 S+core7 Cache 存儲陣列

1.7.1 指令 Cache

I-Cache 控制器是一個獨立於處理器內核之外的設備。在內核與 I-Cache 控制器之間有兩條請求匯流排：一條是指令匯流排 (I-Bus)，另一條是 I-Cache 指令匯流排。I-Cache 控制器需要處理來自這兩條匯流排的請求。若它不能在下一個 Cycle 處理完請求，I-Cache 控制器將阻塞管線 (pipeline) 去繼續處理請求。

I-Cache 特性：

- 2 路組相聯的 Cache 結構
- Cache 的大小是 4K bytes
- Cache-line 的大小是 4Words
- 採用讀操作分配策略
- 提供使一個 Cache-line 無效的命令（虛擬位址模式下）
- 提供使整個 Cache 內容無效的命令（虛擬位址模式下）
- 提供預取一個 Cache-line 的命令（虛擬位址模式下）
- 提供預取和“Lock”一個 Cache-line 的命令（虛擬位址模式下）
- 在內核和 I-Cache 控制器之間有指令預取請求匯流排和 I-Cache 請求匯流排
- 取指操作時提供精確匯流排錯誤異常（precise bus error exception）
- 處理器內核可以拒絕 I-Cache 設備的當前和前一指令請求

1.7.2 資料 Cache

D-Cache 控制器也是一個獨立於處理器內核之外的設備。在內核與 D-Cache 控制器之間也有兩條請求匯流排 (Request Bus)：一條是用於 Load 或 Store 指令的匯流排 (D-Bus)，另一條是 D-Cache 指令匯流排。D-Cache 控制器需要處理來自這兩條匯流排的請求。若 D-Cache 控制器不能在下一個 Cycle 處理完請求，它將阻塞管線 (pipeline)。

D-Cache 特性：

- 2 路組相聯的 Cache 結構
- 4K bytes 的 Cache 大小
- 4 Words 的 Cache line 大小
- 讀操作分配策略
- 提供使 Cache line 無效的命令
- 提供預取 Cache line 的命令
- 提供預取並鎖定 Cache line 的命令
- Load 操作時提供精確的匯流排異常；若寫緩衝器是禁用的，Store 操作時也提供精確的匯流排異常
- 若寫緩衝器是允許的，Store 操作時則提供不精確的匯流排異常

1.7.3 存儲一致性

系統設計時必須考慮存儲一致性問題。因為 Cache 中保存的是主記憶體資料的一塊拷貝，而當 Cache 中資料正被使用時，其他存儲設備可能去修改主記憶體中的那塊資料，從而使 Cache 中的資料與主記憶體中的不一致。記憶體一致性問題必須透過系統設計或軟體來解決。下面是 4 種典型的存儲一致性問題：

■ 寫指令操作

當處理器內核存儲指令到主記憶體中以待後續執行時，必須保證這些指令被寫到主記憶體中，並且保證 I-Cache 中的相應位置上的內容是無效的。S+core7 處理器中，D-Cache 與 I-Cache 是獨立的。

■ 寫回 Cache 操作

當一個 DMA 設備直接從主記憶體中移出資料時，能獲得正確的資料是至關重要的。如果 D-Cache 使用的是寫回策略，且程式最近曾執行寫資料操作，那麼，要移出的正確資料可能還保存在 D-Cache 中，未被寫入主記憶體。而內核是不清楚這種情況的。因此，必要的話，在 DMA 設備開始從主記憶體中讀資料之前，必須將還保存在 D-Cache 中的正確資料寫入主記憶體中。

■ 從 DMA 設備裝載資料到主記憶體

當從一個 DMA 設備中裝載資料到主記憶體中時，使 Cache 中在位址範圍內的所有 Cache line 無效是至關重要的。否則，當內核從 DMA 設備中讀資料時，會讀到保存在 Cache 中的舊的資料。因此，在內核從 DMA 設備中讀資料之前，必須使相應得 Cache line 無效。

■ 寫緩衝器

當寫緩衝器被允許且其中已有一些資料時，主記憶體中內容或 IO 暫存器值可能是不正確的。為達到存儲資料的一致性，應該透過 Cache 指令將寫緩衝器清空。

1.8 LIM 和 LDM

1.8.1 LIM-Local Instruction Memory

LIM（指令記憶體）系統包括位於 I-Cache 控制器中的 SRAM（Synchronous RAM）。SRAM 提供了存取存儲塊的快速介面。用戶可以透過 CCR 控制暫存器的 Bit2 來使能 LIM。注意，LIM 設備被允許並使用之前，必須正確初始化。S+core7 處理器提供了可以初始化 SRAM 或向 SRAM 填充內容的 Cache 指令。

1.8.2 LDM-Local Data Memory

LDM（資料記憶體）系統也包括 SRAM（Synchronous RAM）。SRAM 提供了存取存儲塊的快速介面。用戶可以透過 CCR 控制暫存器的 Bit3 來使能 LDM。注意，LDM 設備被允許並使用之前，必須正確初始化。S+core7 處理器提供了可以初始化 SRAM 或向 SRAM 填充內容的 Cache 指令。

LDM 的位址範圍可以透過 Cache 指令（填充 LDM）來配置，當執行“填充 LDM”指令時，D-Cache 控制器可以記錄 LDM 設備的實體位址。

1.9 片上除錯

S+core7 處理器的 Debug 機制具備如下特徵：

■ 片外 Debug Memory 存取

SJTAG 允許處理器在 Debug 模式下存取片外的指令或資料。處理器發出的存取申請經 JTAG 協議解析後發送到由 Debug Host PC 控制的 Debug Probe 中。然後，Debug Probe 再將要存取的位址重定位到 Probe Memory 的位址。接下來，對應位址中的資料就會再經過 JTAG 然後被讀到處理器中。

透過這種機制，不需要晶片具備 Debug ROM 就可以進行系統除錯，從而實現了處理器與 Debug Host PC 間的通信。

■ 硬體斷點

提供兩種硬體斷點。用戶可配置這兩種硬體斷點，以便在下列情況時發生 Debug 異常：

- 從特定位址中取指令（Breakpoint）
- 從特定位址中取資料與存取資料（Watchpoint）

■ 軟體斷點指令

提供兩個輔助的指令，這兩個指令可輔助系統除錯。

- 軟體斷點除錯（SDBBP，Software Debug Breakpoint）
- 從 Debug 異常返回（DRET，Debug Exception Return）。

■ 單步執行

透過處理器提供的單步異常機制，可實現一條指令一條指令地執行程式，從而進行程式除錯。

■ Debug 中斷

Debug 中斷用於強制處理器在任何時候進入 Debug 模式

■ DMA 存取

由 SJTAG 直接控制 DMA 通道是透過 BIU 來存取系統匯流排的。當用戶需要直接下載代碼到系統記憶體中時，這一特性很有用。

2 S+core7 指令系統

2.1 概述

指令是 CPU 執行某種操作的命令。微處理器（MPU）或微控制器（MCU）所能識別全部指令的集合成為指令系統，是研製廠家在設計 CPU 時所賦予的功能。只有正確書寫和使用指令，才能完成設計任務。因此，學習和掌握指令的功能與應用是程式設計的基礎。本章將詳細介紹 S+core7 的指令系統。

S+core7 指令系統分為 32 位指令集和 16 位指令集。

32 位指令集的效率高、功能強，可以完成所有處理器支援的操作。大多數 32 位指令都具有三個運算元，且支援使用 16 位的立即數。另外，32 位指令可以存取全部 32 個通用暫存器。32 位指令集包含一些系統控制指令以及協同處理器控制指令，這些指令是 16 位指令集不具備的。這些指令可以控制處理器完成一些複雜處理，大大增強系統的功能。

16 位指令集可以看作是 32 位指令集的壓縮形式的子集，它是專為編譯器（compiler）編譯混合模式的指令而設計，以減少代碼容量的，一般不推薦用來進行手動組合語言編碼。由於 16 位指令集的指令寬度的限制，基本上所有的 16 位指令都是兩個運算元的操作，且允許使用小立即數，另外，由於指令長度的限制，指令只能存取 32 個通用暫存器中的前 16 個（r0 ~r15）。當發生 16 位指令不為字對齊的情況，則最好將該 16 位指令改為相應的 32 位指令，而最好不要插入一個 16 位的 nop! 指令來對齊字邊界，因為這樣將增加指令執行個數。

2.2 指令格式與編碼

S+core7 處理器是基於精簡指令集電腦（RISC）原理設計的，指令集和相關譯碼機制較為簡單。S+core7 處理器具有 32 位的指令匯流排，每次將 32 位長度的指令代碼填入譯碼機構。

S+core7 處理器支援 32 位和 16 位兩種指令模式。處理器可以自動識別一個 32 位的編碼是一條 32 位的指令還是兩條 16 位的指令，而不需要任何指令模式的切換。一個 32 位的指令編碼中的 bit31 和 bit15 兩位稱為“P-位”，用來區分指令模式。所以，32 位指令的實際有效寬度為 30 位，而 16 位指令的實際有效寬度為 15 位。如圖 2-1 所示：

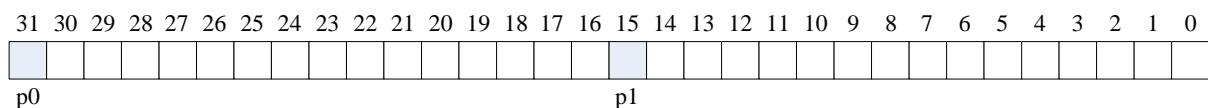


圖 2-1 S+core7 處理器的指令模式

對應於 P-位元的不同編碼，指令格式的意義如表 2-1 所示。

表 2-1 P-位指示的指令格式

P0	P1	意義	格式符號
1	1	32 位指令	32BF
0	0	16 位指令 + 16 位指令	16BF
0	1	並行條件執行 (PCE-Parallel Condition Execute) 指令： • 並行執行條件旗標 T 為 1 時執行高半字的 16 位指令 1 • 並行執行條件旗標 T 為 0 時執行低半字的 16 位指令 2	PCEF
1	0	未定義	UDEF

除去 P-位之後的指令被劃分為若干欄位，每一欄位均代表一定的含義。其中，32 位指令的主要操作碼由第 25~29 位確定，16 位指令的主要操作碼則由第 12~14 位確定。指令中其餘欄位的意義隨指令的不同而有所不同。各類指令包含的欄位如圖 2-2 所示。

30 bit	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
J-form	OP	Disp24(lmm)																										LK														
BC-form	OP	Disp[18:9](lmm)										BC		Disp[8:0](lmm)										LK																		
Special-form	OP	rD	ra		rB		0	0	0	func6										CU																						
I-form	OP	rD(D,S1)	func3	Imm16(S2)																										CU												
RI-form	OP	rD(D)	rA(S1)		Imm14(S2)																									CU												
RIX-form	OP	rD(D)	rA(S1)		Imm12(S2)																									func3												
CENew	OP	USD1	rA(optional)		rB(optional)		USD2		func5																				func5													
CR-form	OP	rD	CR		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MI	func2	II	0	MI										
coprocessor	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
mtc/mfc	OP	rD	CrA		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CP#	Sub-OP																		
ldc/stc	OP	rD	CrA		Imm10																									CP#	Sub-OP											
cop	OP	CrD	CrA		CrB		COP-Code		CP#																					CP#	Sub-OP											
15 bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																											
B<cond>x	OP	EC	Disp8(Imm)																													0	Don't care									
J-form	OP	Disp11(Imm)		LK																																						
R-form	OP	rD _{go}	rA _{go}	func4																																						
I-form-1	OP	rD _{go}	Imm5		func3																																					
I-form-2	OP	rD _{go}	Imm8																																							

圖 2-2 S+core7 處理器指令欄位

S+core7 指令集按照 OP 欄位對指令進行分類，32 位指令集可以分為十一類，分別是：J-form、BC-form、Special-form、I-form、RI-form、RIX-form、CENew、CR-form、mtc/mfc 類、ldc/stc 類以及 cop 類等；16 位指令集可以分為五類，分別是：B<cond>x、J-form、R-form、I-form-1、I-form-2 等。表 2-2 所示的是各分類的具體解釋。如，J-form 表示無條件跳越類的指令。

表 2-2 指令分類對照表

符號	含義
J-form	無條件跳越類指令
BC-form	條件跳越類指令（32 位指令）
Special-form	暫存器做為源運算元的指令（32 位指令）
I-form	立即數做為源運算元的指令
RI-form	暫存器與立即數做為源運算元的指令
RIX-form	變址定址類指令
CENew	用戶擴展類指令
CR-form	特殊功能暫存器操作類指令
B<cond>x	條件跳越類指令（16 位指令）
R-form	暫存器做為源運算元的指令（16 位指令）

為了描述方便，對在指令系統敘述過程中所要用到的符號作統一約定，如表 2-3 所示。

在 S+core7 的指令系統中，根據指令的功能不同，指令具有的語法形式也是不同的。一條典型的 32 位指令語法格式如下所示：

```
<opcode>{.c} {operand1}, {operand2}
```

在上面的格式中，“<>”符號內的項是必須的，“{}”符號內的項是可選的。例如，**<opcode>**是指令助記符，這裏必須書寫；而**{.c}**是可選開關，指該指令執行結束後是否需要對條件旗標位元產生影響（針對某些指令，可以選擇該項）；**operand1**一般是一個通用暫存器，用於保存運算結果；**operand2**可以是一個或者兩個通用暫存器，或者是暫存器與立即數的組合，或者是一個立即數，用於參與運算，或者指定定址位址。

例：

and.c r4, r2, r1

其中，**and** 是必須的，表示該指令完成的操作；**.c** 是可選的，表示該操作將影響條件旗標位元，這裏，**operand1** 是 **r4**，用來保存與操作之後的結果；**operand2** 是 **r2** 和 **r1**，表示 **r2** 與 **r1** 做與操作。

表 2-3 指令欄位含義說明

欄位	含義描述
OP	主操作碼
func	擴展功能碼

欄位	含義描述
CU	條件旗標更新位 0：不更新條件旗標 1：更新條件旗標，以反應操作的結果
LK	鏈接位 0：不更新鏈接暫存器 (r3) 1：更新鏈接暫存器 (r3)。若指令為分支指令，則分支指令的下一條指令的位址將被置於鏈接暫存器 (r3) 內
BN	位操作指令的指定操作位數
BC	分支指令的分支條件 (Branch Condition)
EC	條件執行指令的執行條件碼 (Execute Condition)
TC	T 旗標更新指令的測試條件碼
SA	移位/迴圈指令的移位位數
rD	目標通用暫存器 (GPR)
rA, rB	源通用暫存器 (GPR)
g0	代表 GPR0~15
g1	代表 GPR16~31
Disp	跳越和分支指令的偏移量
Imm	立即數
CR	控制暫存器
Sub-OP	協同處理器指令的擴展子操作碼
USD	擴展定制 (custom) 指令的用戶定義欄位
CrA	源協同處理器暫存器
CrD	目標協同處理器暫存器
CP#	協同處理器編號
COP-code	協同處理器指令的擴展操作碼
Exp	指數
Spar (Imm)	用於陷阱及其系統調用指令傳遞資訊的程式參數
Srn	特殊暫存器編號

2.3 32 位指令集

32 位指令集有五類，包括：裝載與存儲指令、資料處理指令、分支指令、特殊指令和協同處理器指令。其中：

- 裝載與存儲指令可以完成對記憶體的存取操作；
- 資料處理指令完成算術和邏輯運算等操作；
- 分支指令完成程式跳越以及函數調用等操作；
- 特殊指令包含了對 Cache 的操作以及對處理器內核的特殊暫存器的控制等；
- 協同處理器指令可以完成一些影像編碼等特殊功能。

下麵將分別對這五類指令做介紹。

2.3.1 裝載與存儲指令

S+core7 處理器是裝載/存儲體系結構的典型的 RISC 處理器，對記憶體的存取只能透過使用裝載和存儲指令實現。S+core7 的裝載/存儲指令可以實現字、半字、無符號/有符號位元組操作。

裝載指令用於從記憶體中讀取資料放入暫存器中；存儲指令用於將暫存器中的資料保存到記憶體。S+core7 有兩類裝載/存儲指令，一類用於對齊資料位址存儲單元的存取操作，另一類用於非對齊資料位址存儲單元的存取操作。表 2-4 所列為 S+core7 的裝載/存儲指令表。

表 2-4 S+core7 裝載/存儲指令表

助記符	說明	格式
lb	裝載位元組資料	lb rD, [rA, Slmm15]
	裝載位元組資料(後變址定址)	lb rD, [rA]+, Slmm12
	裝載位元組資料(前變址定址)	lb rD, [rA, Slmm12]+
lbu	裝載無符號位元組資料	lbu rD, [rA, Slmm15]
	裝載無符號位元組資料(後變址定址)	lbu rD, [rA]+, Slmm12
	裝載無符號位元組資料(前變址定址)	lbu rD, [rA, Slmm12]+
lh	裝載半字資料	lh rD, [rA, Slmm15]
	裝載半字資料(後變址定址)	lh rD, [rA]+, Slmm12
	裝載半字資料(前變址定址)	lh rD, [rA, Slmm12]+
lhu	裝載無符號半字資料	lhu rD, [rA, Slmm15]
	裝載無符號半字資料(後變址定址)	lhu rD, [rA]+, Slmm12
	裝載無符號半字資料(前變址定址)	lhu rD, [rA, Slmm12]+

助記符	說明	格式
lw	裝載字資料	lw rD, [rA, SImm15]
	裝載字資料(後變址定址)	lw rD, [rA]+, SImm12
	裝載字資料(前變址定址)	lw rD, [rA, SImm12]+
sb	存儲位元組資料	sb rD, [rA, SImm15]
	存儲位元組資料(後變址定址)	sb rD, [rA]+, SImm12
	存儲位元組資料(前變址定址)	sb rD, [rA, SImm12]+
sh	存儲半字資料	sh rD, [rA, SImm15]
	存儲半字資料(後變址定址)	sh rD, [rA]+, SImm12
	存儲半字資料(前變址定址)	sh rD, [rA, SImm12]+
sw	存儲字資料	sw rD, [rA, SImm15]
	存儲字資料(後變址定址)	sw rD, [rA]+, SImm12
	存儲字資料(前變址定址)	sw rD, [rA, SImm12]+
ldi	裝載立即數	ldi rD, SImm16
ldis	裝載移位立即數	ldis rD, SImm16
lcb	裝載合併字資料開始	lcb [rA]+
lcw	裝載合併字資料	lcw rD, [rA]+
lce	裝載合併字資料結束	lce rD, [rA]+
scb	存儲合併字資料開始	scb rD, [rA]+
scw	存儲合併字資料	scw rD, [rA]+
sce	存儲合併字資料結束	sce [rA]+

裝載/存儲指令有以下幾種定址方式：立即數定址、基底位址變址定址、前變址定址和後變址定址。

(1) 立即數定址格式：

<opcode> <rD>, <SImm>

立即數定址是將立即數作為運算元直接參與操作。

(2) 基底位址變址定址格式：

<opcode> <rD>, <[rA, SImm15]>

基底位址變址定址是將源暫存器 rA 的內容與後面給出的立即數偏移量相加而形成運算元的有效位址。

(3) 前變址定址格式：

<opcode> <rD>, <[rA, SIimm12]>

前變址定址是將源暫存器 rA 的內容與後面給出的立即數偏移量相加而形成運算元的有效位址，並在操作完畢後將 rA 的內容更新為本次操作的有效位址。

(4) 後變址定址格式：

<opcode> <rD>, <[rA]+, SIimm12>

後變址定址是指將源暫存器 rA 的內容作為運算元的有效位址，並在操作完畢後，rA 的內容透過與後面給出的立即數相加而更新。

Ib/sb——裝載/存儲位組指令

使用 Ib 指令可以將一個有符號的位元組資料從記憶體裝載到暫存器，sb 可以將一個位元組資料從暫存器保存到記憶體。Ib 和 sb 指令都具有基底位址變址定址、前變址定址和後變址三種定址方式的指令格式。

指令格式：

(1) **Ib rD, [rA, SIimm15]**

基底位址變址定址。從記憶體[rA+SIimm15]位址處讀取位元組資料並做符號擴展，然後保存至 rD 暫存器。例如：Ib r4, [r2, 0x1234]。

(2) **Ib rD, [rA]+, SIimm12**

後變址定址。從記憶體[rA]位址處讀取位元組資料並做符號擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容與 SIimm12 相加並更新。例如：Ib r4, [r2]+, 0x4。

(3) **Ib rD, [rA, SIimm12]+**

前變址定址。從記憶體[rA+SIimm12]位址處讀取位元組資料並做符號擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：Ib r4, [r2, 0x0123]+。

(4) **sb rD, [rA, SIimm15]**

基底位址變址定址。將 rD 暫存器低 8 位保存至記憶體[rA+SIimm15]位址處。例如：sb r4, [r2, 0x123]。

(5) **sb rD, [rA]+, SIimm12**

後變址定址。將 rD 暫存器低 8 位保存至記憶體[rA]位址處。操作完成後 rA 的內容與 SIimm12 相加並更新。例如：sb r4, [r2]+, 0x4。

(6) **sb rD, [rA, SIimm12]+**

前變址定址。將 rD 暫存器低 8 位保存至記憶體[rA+SImm12]位址處。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：sb r4, [r2, 0x0123]+。

Ibu——裝載無符號位組指令

使用 Ibu 指令可以將一個無符號的位元組資料從記憶體裝載到暫存器。

指令格式：

(1) **Ibu rD, [rA, SImm15]**

基底位址變址定址。從記憶體[rA+SImm15]位址處讀取位元組資料並做零擴展，然後保存至 rD 暫存器。例如：Ibu r4, [r2, 0x1234]。

(2) **Ibu rD, [rA]+, SImm12**

後變址定址。從記憶體[rA]位址處讀取位元組資料並做零擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容與 SImm12 相加並更新。例如：Ibu r4, [r2]+, 0x4。

(3) **Ibu rD, [rA, SImm12]+**

前變址定址。從記憶體[rA+SImm12]位址處讀取位元組資料並做零擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：Ibu r4, [r2, 0x0123]+。

Ih/sh——裝載/存儲半字指令

使用 Ih 指令可以將一個有符號的半字資料從記憶體裝載到暫存器，sh 可以將一個半字資料從暫存器保存到記憶體。需要注意的是，記憶體位址必須為半字對齊，否則將發生位址對齊錯誤異常。Ih 和 sh 指令都具有基底位址定址、前變址定址和後變址三種定址方式的指令格式。

指令格式：

(1) **Ih rD, [rA, SImm15]**

基底位址變址定址。從記憶體[rA+SImm15]位址處讀取半字資料並做符號擴展，然後保存至 rD 暫存器。例如：Ih r4, [r2, 0x1234]。

(2) **Ih rD, [rA]+, SImm12**

後變址定址。從記憶體[rA]位址處讀取半字資料並做符號擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容與 SImm12 相加並更新。例如：Ih r4, [r2]+, 0x4。

(3) **Ih rD, [rA, SImm12]+**

前變址定址。從記憶體[rA+SImm12]位址處讀取半字資料並做符號擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：Ih r4, [r2, 0x0124]+。

(4) **sh rD, [rA, SImm15]**

基底位址變址定址。將 rD 暫存器低 16 位保存至記憶體[rA+SImm15]位址處。例如：sh r4, [r2, 0x124]。

(5) sh rD, [rA]+, \$Imm12

後變址定址。將 rD 暫存器低 16 位保存至記憶體[rA]位址處。操作完成後 rA 的內容與 \$Imm12 相加並更新。例如：sh r4, [r2]+, 0x4。

(6) sh rD, [rA, \$Imm12]+

前變址定址。將 rD 暫存器低 16 位保存至記憶體[rA+\$Imm12]位址處。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：sh r4, [r2, 0x0124]+。

Ihu——裝載無符號半字指令

使用 Ihu 指令可以將一個無符號的半字資料從記憶體裝載到暫存器。

指令格式：

(1) Ihu rD, [rA, \$Imm15]

基底位址變址定址。從記憶體[rA+\$Imm15]位址處讀取半字資料並做零擴展，然後保存至 rD 暫存器。例如：Ihu r4, [r2, 0x1234]。

(2) Ihu rD, [rA]+, \$Imm12

後變址定址。從記憶體[rA]位址處讀取半字資料並做零擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容與 \$Imm12 相加並更新。例如：Ihu r4, [r2]+, 0x4。

(3) Ihu rD, [rA, \$Imm12]+

前變址定址。從記憶體[rA+\$Imm12]位址處讀取半字資料並做零擴展，然後保存至 rD 暫存器。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：Ihu r4, [r2, 0x0123]+。

Iw/sw——裝載/存儲字指令

使用 Iw 指令可以將一個字資料從記憶體裝載到暫存器，sw 可以將一個字資料從暫存器保存到記憶體。需要注意的是，記憶體位址必須為字對齊，否則將發生位址對齊錯誤異常。Iw 和 sw 指令都具有基底位址定址、前變址定址和後變址三種定址方式的指令格式。

指令格式：

(1) Iw rD, [rA, \$Imm15]

基底位址變址定址。從記憶體[rA+\$Imm15]位址處讀取字資料，並保存至 rD 暫存器。例如：Iw r4, [r2, 0x1234]。

(2) Iw rD, [rA]+, \$Imm12

後變址定址。從記憶體[rA]位址處讀取字資料，並保存至 rD 暫存器。操作完成後 rA 的內容與 \$Imm12 相加並更新。例如：Iw r4, [r2]+, 0x4。

(3) Iw rD, [rA, \$Imm12]+

前變址定址。從記憶體[rA+SImm12]位址處讀取字資料，並保存至 rD 暫存器。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：lw r4, [r2, 0x0124]+。

(4) **sw rD, [rA, SImm15]**

基底位址定址。將 rD 暫存器內容保存至記憶體[rA+SImm15]位址處。例如：sw r4, [r2, 0x124]。

(5) **sw rD, [rA]+, SImm12**

後變址定址。將 rD 暫存器內容保存至記憶體[rA]位址處。操作完成後 rA 的內容與 SImm12 相加並更新。例如：sw r4, [r2]+, 0x4。

(6) **sw rD, [rA, SImm12]+**

前變址定址。將 rD 暫存器內容保存至記憶體[rA+SImm12]位址處。操作完成後 rA 的內容更新為本次操作的記憶體位址。例如：sw r4, [r2, 0x0124]+。

ldi——裝載 16 位立即數

指令格式：**ldi rD, SImm16**

使用 ldi 指令可以將一個 16 位的立即數進行符號擴展並裝載至目標暫存器 rD。ldi 指令只具有立即數定址一種定址方式。

例如：使用 ldi 指令進行立即數加法運算的程式如程式清單 2.1 所示。

程式清單 2.1：立即數加法

```
.data
sum: .word 0x0          // 定義變數
.text
...
la r4, sum            // 取得變數 sum 的位址
ldi r2, 0x1234         // 立即數裝入 r2
ldi r3, 0x5678         // 立即數裝入 r3
add r1, r2, r3          // 加法
sw r1, [r4, 0]           // 結果保存至 sum 變數
...
```

ldis——裝載移位立即數

指令格式：**ldis rD, Imm16**

使用 ldis 指令可以將一個 16 位的立即數左移 16 位後裝載至目標暫存器 rD。ldis 指令只具有立即數定址一種定址方式。例如：ldi r4, 0xabcd

Icb——裝載合併字資料開始

指令格式：Icb [rA]+

Icb 指令將記憶體中指定位址的資料裝載入特殊暫存器 LCR (Load Combine Register) 中，操作位址由 $rA \& 0xffffffffc$ 確定。完成操作後，暫存器 rA 的內容自動增加 4。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 Icw、Ice 配合可以完成對非對齊位址資料的存取。

圖 2-3 說明瞭當特殊暫存器 LCR 和記憶體中各給定一字資料時，裝載合併指令的操作。

注意：

圖 2-3、圖 2-4 僅針對 SPCE3200 如此，以後 S+core7 內核可能有所更改，請讀者注意。

		Original Condition				記憶體中的字資料 (Addr[31:2],00b)		LCR中的字資料																	
						S = Addr[1:0]																			
		Big Endian				Little Endian																			
		<table border="1"> <tr> <td>s=0</td><td>1</td><td>2</td><td>3</td> </tr> <tr> <td> </td><td> </td><td> </td><td> </td> </tr> </table>				s=0	1			2	3					<table border="1"> <tr> <td>s=3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td> </td><td> </td><td> </td><td> </td> </tr> </table>				s=3	2	1	0		
s=0	1	2	3																						
s=3	2	1	0																						
		Mode (Big)	rD	Next LCR	Load Word?	Mode (Little)	rD	Next LCR	Load Word?																
LCB LCB [Addr]		s=0 (Not Set)	ABCD	YES		s=0 (Not Set)	ABCD	YES																	
		s=1 (Not Set)	ABCD	YES		s=1 (Not Set)	ABCD	YES																	
		s=2 (Not Set)	ABCD	YES		s=2 (Not Set)	ABCD	YES																	
		s=3 (Not Set)	ABCD	YES		s=3 (Not Set)	ABCD	YES																	
LCW LCW rD, [Addr]		s=0 <i>abcd</i>	ABCD	YES		s=0 <i>dABC</i>	ABCD	YES																	
		s=1 <i>bcdA</i>	ABCD	YES		s=1 <i>cdAB</i>	ABCD	YES																	
		s=2 <i>cdAB</i>	ABCD	YES		s=2 <i>bcdA</i>	ABCD	YES																	
		s=3 <i>dABC</i>	ABCD	YES		s=3 <i>abcd</i>	ABCD	YES																	
LCE LCE rD, [Addr]		s=0 <i>abcd</i>	(Not Set)	NO		s=0 <i>dABC</i>	ABCD	YES																	
		s=1 <i>bcdA</i>	ABCD	YES		s=1 <i>cdAB</i>	ABCD	YES																	
		s=2 <i>cdAB</i>	ABCD	YES		s=2 <i>bcdA</i>	ABCD	YES																	
		s=3 <i>dABC</i>	ABCD	YES		s=3 <i>abcd</i>	(Not Set)	NO																	

圖 2-3 裝載合併指令操作說明

Icw——裝載合併字資料

指令格式：Icw rD, [rA]+

Icw 指令將記憶體中指定位址的資料裝載入特殊暫存器 LCR (Load Combine Register) 中，操作位址由 $rA \& 0xffffffffc$ 確定。完成操作後，暫存器 rA 的內容自動增加 4。然後，將裝載的資料移位，並根據處理器 **Endian** 模式和 rA 位址的最低有效 2 位將移位結果與 LCR 暫存器中的原始值合併，以實現非對齊的記憶體資料的存取。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 Icb、Ice 配合可以完成對非對齊位址資料的存取。

Ice——裝載合併字資料結束
指令格式：Ice rD, [rA]+

Ice 指令將記憶體中指定位址的資料裝載入特殊暫存器 LCR (Load Combine Register) 中，操作位址由 $rA \& 0xffffffffc$ 確定。完成操作後，暫存器 rA 的內容自動增加 4。然後，將裝載的資料移位，並根據處理器 **Endian** 模式和 rA 位址的最低有效 2 位將移位結果與 LCR 暫存器中的原始值合併，以實現非對齊的記憶體資料的存取。若 rA 的最低有效 2 位為 0，則特殊暫存器 LCR 在 **Ice** 指令執行後不會改變。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 **lcb**、**lcw** 配合可以完成對非對齊位址資料的存取。

scb——存儲合併字資料開始
指令格式：scb rD, [rA]+

scb 指令將記憶體中的字資料與通用暫存器 rD 中的資料合併到特殊暫存器 SCR (Store Combine Register) 中，然後將合併結果存儲到記憶體中，該存儲位址由 $rA \& 0xffffffffc$ 確定。存儲操作完成後，暫存器 rA 內容自動增加 4。存儲資料量和位址取決於處理器 **Endian** 模式和暫存器 rA 中最低有效 2 位。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 **scw**、**sce** 配合可以完成對非對齊位址資料的存取。

圖 2-4 說明利用特殊暫存器 SCR 將記憶體中的字資料與暫存器 rD 的內容合併並存儲的操作。

Original Condition {

<div style="background-color: red; padding: 5px; color: white; font-weight: bold;">Big Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 0 1 2 3</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Big)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">MEM</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Next LCR</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Store Word?</div>	<div style="background-color: green; padding: 5px; color: white; font-weight: bold;">Little Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 3 2 1 0</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Little)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">MEM</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Next LCR</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Store Word?</div>
--	--

SCB SCB rD, [Addr]	<div style="background-color: red; padding: 5px; color: white; font-weight: bold;">Big Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 0 1 2 3</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Big)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">AB C D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 ABCD</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 XABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 XYAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 XYZA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div>	<div style="background-color: green; padding: 5px; color: white; font-weight: bold;">Little Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 3 2 1 0</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Little)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">XYZ A</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">XY A B</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">X Y Z W</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 XYZA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 XYAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 XABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 ABCD</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">CDAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">CDAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div>
SCW SCW rD, [Addr]	<div style="background-color: red; padding: 5px; color: white; font-weight: bold;">Big Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 0 1 2 3</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Big)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">AB C D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 ABCD</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 aABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 abAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 abcA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div>	<div style="background-color: green; padding: 5px; color: white; font-weight: bold;">Little Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 3 2 1 0</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Little)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">abcA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">abAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">aYZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 XYZA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 XYAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 XABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 ABCD</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">CDAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">CDAB</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">DABC</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">BCDA</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">ABC D</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">YES</div>
SCE SCE [Addr]	<div style="background-color: red; padding: 5px; color: white; font-weight: bold;">Big Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 0 1 2 3</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Big)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(not write)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">NO</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 (not write)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 aYZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 abZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 abcW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">NO</div>	<div style="background-color: green; padding: 5px; color: white; font-weight: bold;">Little Endian</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s = 3 2 1 0</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Mode (Little)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">abcW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">abZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">aYZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">(Not Set)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=0 abcW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=1 abZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=2 aYZW</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">s=3 (not write)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">Y(masked)</div> <div style="margin-top: 10px; border: 1px solid black; padding: 2px; display: inline-block;">NO</div>

圖 2-4 存儲合併指令操作說明

scw——裝載合併字資料**指令格式：scw rD, [rA]+**

scw 指令將合併到特殊暫存器 SCR 中的字資料存儲到記憶體中，存儲位址由 $rA \& 0xffffffffc$ 確定。完成操作後，暫存器 rA 的內容自動增加 4。然後，源暫存器 rD 和 SCR 暫存器會根據處理器 **Endian** 模式和暫存器 rA 的最低有效 2 位元進行資料合併，而存儲操作總是以字為單位。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 **scb**、**sce** 配合可以完成對非對齊位址資料的存取。

sce——裝載合併字資料結束**指令格式：sce [rA]+**

sce 指令將特殊暫存器 SCR 中的資料存儲到記憶體中，存儲位址由 $rA \& 0xffffffffc$ 確定。完成操作後，暫存器 rA 的內容自動增加 4。存儲資料量和位址取決於處理器 **Endian** 模式和暫存器 rA 中最低有效 2 位。由於在對記憶體進行存取操作時 rA 的最低兩位被截去，所以不會發生位址對齊錯誤。該指令與 **scb**、**scw** 配合可以完成對非對齊位址資料的存取。

2.3.2 資料處理指令

S+core7 處理器的資料處理指令大致可以分為六類：算術運算指令、邏輯運算指令、位運算指令、資料傳送指令、移位運算指令和擴展操作指令。大多數的資料處理指令都可以選擇.c 尾碼，表示是否需要影響條件旗標位元參與運算。表 2-5 所示為 **S+core7** 的資料處理指令表。

表 2-5 S+core7 資料處理指令表

助記符	說明	格式
add{.c}	加法運算	add{.c} rD, rA, rB
addc{.c}	帶進位加法運算	addc{.c} rD, rA, rB
addi{.c}	立即數加法運算	addi{.c} rD, SIimm16
addis{.c}	移位立即數加法運算	addis{.c} rD, SIimm16
addri{.c}	暫存器立即數加法運算	addri{.c} rD, rA, SIimm14
sub{.c}	減法運算	sub{.c} rD, rA, rB
subc{.c}	帶借位減法運算	subc{.c} rD, rA, rB
subi{.c}	立即數減法運算	subi{.c} rD, SIimm16
subis{.c}	移位立即數減法運算	subis{.c} rD, SIimm16
subri{.c}	暫存器立即數減法運算	subri{.c} rD, rA, SIimm14
neg{.c}	取負	neg{.c} rD, rB
cmp{TCS}.c	比較	cmp{TCS}.c rA, rB

助記符	說明	格式
cmpi.c	立即數比較	cmpi.c rA, SImm16
cmpz{TCS}.c	零比較	cmpz{TCS}.c rA
mul	乘法運算	mul rA, rB
mulu	無符號數乘法運算	mulu rA, rB
div	除法運算	div rA, rB
divu	無符號數除法運算	divu rA, rB
and{.c}	邏輯與	and{.c} rD, rA, rB
andi{.c}	立即數邏輯與	andi{.c} rD, Imm16
andis{.c}	移位立即數邏輯與	andis{.c} rD, Imm16
andri{.c}	暫存器立即數邏輯與	andri{.c} rD, rA, Imm14
or{.c}	邏輯或	or{.c} rD, rA, rB
ori{.c}	立即數邏輯或	ori{.c} rD, Imm16
oris{.c}	移位立即數邏輯或	oris{.c} rD, Imm16
orri{.c}	暫存器立即數邏輯或	orri{.c} rD, rA, Imm14
xor{.c}	邏輯異或	xor{.c} rD, rA, rB
not{.c}	邏輯異非	not{.c} rD, rA
t{cond}	測試並置 T 條件旗標	t{cond}
bittst.c	位測試	bittst.c rD, BN
bitset.c	位置位	bitset.c rD, rA, BN
bitclr.c	位清零	bitclr.c rD, rA, BN
bittgl.c	位翻轉	bittgl.c rD, rA, BN
mv{cond}	資料傳送（有條件）	mv{cond} rD, rA
mtcr	傳送資料到控制暫存器	mtcr rD, Crn
mfcr	從控制暫存器傳送資料	mfcr rD, Crn
rol{.c}	迴圈左移	rol{.c} rD, rA, rB
rolc.c	帶進位迴圈左移	rolc.c rD, rA, rB
roli{.c}	立即數迴圈左移	roli{.c} rD, rA, Imm5
rolic.c	立即數帶進位迴圈左移	rolic.c rD, rA, Imm5

助記符	說明	格式
ror{.c}	迴圈右移	ror{.c} rD, rA, rB
rorc.c	帶進位迴圈右移	rorc.c rD, rA, rB
rori{.c}	立即數迴圈右移	rori{.c} rD, rA, Imm5
roric.c	立即數帶進位迴圈右移	roric.c rD, rA, Imm5
sll{.c}	邏輯左移	sll{.c} rD, rA, rB
slli{.c}	立即數邏輯左移	slli{.c} rD, rA, Imm5
sra{.c}	算術右移	sra{.c} rD, rA, rB
srai{.c}	立即數算術右移	srai{.c} rD, rA, Imm5
srl{.c}	邏輯右移	srl{.c} rD, rA, rB
srl{i}.c	立即數邏輯右移	srl{i}.c rD, rA, Imm5
extsb{.c}	擴展有符號位元組資料	extsb{.c} rD, rA
extzb{.c}	擴展無符號位元組資料	extzb{.c} rD, rA
extsh{.c}	擴展有符號半字資料	extsh{.c} rD, rA
extzh{.c}	擴展無符號半字資料	extzh{.c} rD, rA

1. 算術運算指令

add——加法運算

指令格式：**add{extend}{.c} rD, operand2**

add 指令可以完成暫存器內容或暫存器與立即數之間的加法運算。其中，.c 為可選，帶有.c 的指令在操作完畢後將影響條件旗標位元的狀態，否則將不影響其狀態；第二個運算元 **operand2** 由暫存器或立即數構成；**extend** 為附加的運算說明，可選值有：

- 空：表示不帶進位的加法，**operand2** 由兩個暫存器組成。指令格式為：**add{.c} rD, rA, rB**，完成 $rD=rA+rB$ 的操作；
- “c”：表示帶進位的加法，**operand2** 由兩個暫存器組成。指令格式為：**addc{.c} rD, rA, rB**，完成 $rD=rA+rB+c$ 的操作；
- “i”：表示與立即數進行加法運算，**operand2** 由一個立即數組成，指令格式為：**addi{.c} rD, SIimm16**，完成 $rD=rD+SIimm16$ 的操作；
- “is”：表示與移位立即數進行加法運算，**operand2** 由一個立即數組成，指令格式為：**addis{.c} rD, SIimm16**，完成 $rD=rD+(SIimm16<<16)$ 的操作；
- “ri”：表示暫存器立即數加法運算，**operand2** 由一個暫存器和一個立即數組成，指令格式為：**addri{.c} rD, rA, SIimm14**，完成 $rD=rA+SIimm14$ 的操作。

指令舉例：

```
add r4, r2, r1           // r4=r2+r1, 不影響條件旗標位元  
add.c r4, r2, r1         // r4=r2+r1, 影響條件旗標位元  
addc r4, r2, r1          // r4=r2+r1+c, 不影響條件旗標位元  
addis r4, 0x1234         // r4=r4+0x12340000, 不影響條件旗標位元  
addri r4, r2, 0x123       // r4=r2+0x123, 不影響條件旗標位元
```

sub——加法運算

指令格式：**sub{extend}{.c} rD, operand2**

sub 指令可以完成暫存器內容或暫存器與立即數之間的減法運算。其中，**extend** 為附加的運算說明，可選值有：

- 空：表示不帶借位的減法，**operand2** 由兩個暫存器組成，指令格式為：**sub{.c} rD, rA, rB**，完成 $rD=rA-rB$ 的操作；
- “c”：表示帶借位的加法，**operand2** 由兩個暫存器組成，指令格式為：**subc{.c} rD, rA, rB**，完成 $rD=rA-rB-c$ 的操作；
- “i”：表示與立即數進行減法運算，**operand2** 由一個立即數組成，指令格式為：**subi{.c} rD, SIImm16**，完成 $rD=rD-SIImm16$ 的操作；
- “is”：表示與移位立即數進行減法運算，**operand2** 由一個立即數組成，指令格式為：**subis{.c} rD, SIImm16**，完成 $rD=rD-(SIImm16<<16)$ 的操作；
- “ri”：表示暫存器立即數加法運算，**operand2** 由一個暫存器和一個立即數組成，指令格式為：**subri{.c} rD, rA, SIImm14**，完成 $rD=rA-SIImm14$ 的操作。

指令舉例：

```
sub r4, r2, r1           // r4=r2-r1, 不影響條件旗標位元  
sub.c r4, r2, r1          // r4=r2-r1, 影響條件旗標位元  
subc r4, r2, r1            // r4=r2-r1-c, 不影響條件旗標位元  
subis r4, 0x1234          // r4=r4-0x12340000, 不影響條件旗標位元  
subri r4, r2, 0x123        // r4=r2-0x123, 不影響條件旗標位元
```

neg——取負運算

指令格式：**neg{.c} rD, rB**

neg 指令計算 $0-rB$ 的值，並保存至 **rD** 暫存器內，完成對暫存器內容的求負（取補）運算。

指令舉例：

```
neg r4, r2                // r4=0-r2
```

cmp——比較運算

指令格式：**cmp{TCS}.c rA, rB**

cmp 指令進行 $rA-rB$ 的操作，並更新條件旗標位元，但不會更改任何通用暫存器的內容。

與 **add** 等指令不同的是，**cmp** 指令必須帶有.c 欄位，表示比較完成後更新條件旗標位元；{TCS} 欄位用來指定是否需要更新 T 旗標位元，在實際的指令編碼中，TCS 欄位占 2 位。其編碼意

義如表 2-6所示。

表 2-6 比較指令的 TCS 欄位編碼意義表

	命令	TCS		操作
0	CMPTEQ.c	0	0	比較，若 Z=1 則置位 T，否則清 T
1	CMPTMI.c	0	1	比較，若 N=1 則置位 T，否則清 T
2	-	1	0	保留（類似於 CMP.c）
3	CMP.c	1	1	比較

指令舉例：

```
cmp.c r4, r2 // r4-r2，根據結果更新條件旗標位元
```

mul/mulu——乘法運算

指令格式：**mul rA, rB**

mul 指令對通用暫存器 rA 和 rB 做有符號乘法運算。被乘數為 rA 暫存器內的有符號值，乘數為 rB 暫存器內的有符號值，乘積的低字被保存在 Custom Engine 暫存器 CEL 中，高字保存在 CEH 中。

mulu 指令是對兩個無符號字資料做乘法運算，運算結果同樣保存在 CEL 和 CEH 中。

指令舉例：

```
mul r4, r2  
mulu r4, r2
```

div/divu——除法運算

指令格式：**div rA, rB**

div 指令對通用暫存器 rA 和 rB 做有符號除法運算。被除數為 rA 暫存器內的有符號值，除數為 rB 暫存器內的有符號值，除法運算的商被保存在 Custom Engine 暫存器 CEL 中，餘數保存在 CEH 中。如果除數為 0，則會引起 Custom Engine 執行異常，此時的操作結果是不可預期的。

divu 指令是對兩個無符號字資料做除法運算，運算結果同樣保存在 CEL 和 CEH 中。

指令舉例：

```
div r4, r2  
divu r4, r2
```

2. 邏輯運算指令

and——與運算

指令格式：**and{extend}{.c} rD, operand2**

and 指令可以以位操作的方式完成暫存器內容或暫存器與立即數之間的與運算。其中，**extend** 為附加的運算說明，**extend** 的可選值有：

- 空：表示暫存器之間的與運算，**operand2** 由兩個暫存器組成，指令格式為：**and{.c} rD, rA, rB**，完成 $rD=rA\&rB$ 的操作；
- “i”：表示與立即數進行與運算，**operand2** 由一個立即數組成，指令格式為：**andi{.c} rD, Imm16**，完成 $rD=rD\&Imm16$ 的操作；
- “is”：表示與移位立即數進行與運算，**operand2** 由一個立即數組成，指令格式為：**andis{.c} rD, Imm16**，完成 $rD=rD\&(Imm16<<16)$ 的操作；
- “ri”：表示暫存器與立即數進行與運算，**operand2** 由一個暫存器和一個立即數組成，指令格式為：**andri{.c} rD, rA, Imm14**，完成 $rD=rA\&Imm14$ 的操作。

指令舉例：

```
and r4, r2, r1           // r4=r2&r1, 不影響條件旗標位元  
and.c r4, r2, r1         // r4=r2&r1, 影響條件旗標位元  
andis r4, 0x1234          // r4=r4&0x12340000, 不影響條件旗標位元  
andri r4, r2, 0x123        // r4=r2&0x123, 不影響條件旗標位元
```

or——或運算

指令格式：**or{extend}{.c} rD, operand2**

or 指令可以以位操作的方式完成暫存器內容或暫存器與立即數之間的或運算。其中，**extend** 為附加的運算說明，可選值有：

- 空：表示暫存器之間的或運算，**operand2** 由兩個暫存器組成，指令格式為：**or{.c} rD, rA, rB**，完成 $rD=rA | rB$ 的操作；
- “i”：表示與立即數進行或運算，**operand2** 由一個立即數組成，指令格式為：**ori{.c} rD, Imm16**，完成 $rD=rD|Imm16$ 的操作；
- “is”：表示與移位立即數進行或運算，**operand2** 由一個立即數組成，指令格式為：**oris{.c} rD, Imm16**，完成 $rD=rD|(Imm16<<16)$ 的操作；
- “ri”：表示暫存器與立即數進行或運算，**operand2** 由一個暫存器和一個立即數組成，指令格式為：**orri{.c} rD, rA, Imm14**，完成 $rD=rA|Imm14$ 的操作。

指令舉例：

```
or r4, r2, r1             // r4=r2 | r1, 不影響條件旗標位元  
or.c r4, r2, r1           // r4=r2 | r1, 影響條件旗標位元  
oris r4, 0x1234            // r4=r4 | 0x12340000, 不影響條件旗標位元  
orri r4, r2, 0x123          // r4=r2 | 0x123, 不影響條件旗標位元
```

xor——異或運算

指令格式：**xor{.c} rD, rA, rB**

xor 指令可以以位操作的方式將 rA 和 rB 的內容進行異或運算並將結果保存至 rD 暫存器內。

指令舉例：

```
xor r4, r2, r3          // r4 = r2 xor r3, 不影響條件旗標位元
```

not——取反運算

指令格式：**not{.c} rD, rA**

not 指令可以以位操作的方式將 rA 的內容進行取反並將結果保存至 rD 暫存器內。指令格式如下：

指令舉例：

```
not r4, r2,           // r4 = (r2 按位取反), 不影響條件旗標位元
```

t——測試並置 T 旗標位元

指令格式：**t{cond}**

t 指令用於對 T 旗標進行操作。該 T 旗標用於並行條件執行 (PCE) 指令中。根據 T 的值可以判斷 PCE 指令要執行哪個指令分支。其中，cond 尾碼為 EC (EXECUTE CONDITION) 欄位的條件旗標位元測試條件 (如表 2-7 所示)，如果測試條件為真，則置位 T 旗標，否則，清零 T 旗標。

表 2-7 EC 欄位的 cond 編碼意義表

序號	cond 尾碼	操作	測試
0	cs	當 C 為 1 時執行操作	C
1	cc	當 C 為 0 時執行操作	$\sim C$
2	gtu	當 C 為 1 且 Z 為 0 時執行操作	$C \& \sim Z$
3	leu	當 C 為 0 或 Z 為 1 時執行操作	$\sim C \mid Z$
4	eq	當 Z 為 1 時執行操作	Z
5	ne	當 Z 為 0 時執行操作	$\sim Z$
6	gt	當 Z 為 0 且 N 為 V 時執行操作	$(Z=0) \& (N=V)$
7	le	當 Z 為 1 且 N 不為 V 時執行操作	$(Z=1) \& (N \neq V)$
8	ge	當 N 為 V 時執行操作	$N=V$
9	lt	當 N 不為 V 時執行操作	$V \neq N$

序號	cond 尾碼	操作	測試
10	mi	當 N 為 1 時執行操作	N
11	pl	當 N 為 0 時執行操作	~N
12	vs	當 V 為 1 時執行操作	V
13	vc	當 V 為 0 時執行操作	~V
14	-	空操作	-
15	al	無條件執行操作	-

指令舉例：

```
tge // 當 N 為 0 時置位元 T 旗標
```

3. 位運算指令

bitst.c——暫存器位測試

指令格式：**bitst.c rA, BN**

bitst 指令測試通用暫存器 **rA** 的第 **BN** 位元，並更新 **Z** 旗標。同時，**N** 旗標也將受到影響。

指令舉例：

```
bitst r4, 10 // 測試 r4 暫存器的 bit10 位
```

bitset.c——暫存器位置 1

指令格式：**bitset.c rD, rA, BN**

bitset 指令可以將通用暫存器 **rA** 的第 **BN** 位置 1，並更新 **Z** 旗標。同時，**N** 旗標也將受到影響。

指令舉例：

```
bitset.c r4, r5, 10 // 將 r5 暫存器的 bit10 位置 1，並將結果保存至 r4
```

bitclr.c——暫存器位置 0

指令格式：**bitclr.c rD, rA, BN**

bitclr 指令可以將通用暫存器 **rA** 的第 **BN** 位置 0，並更新 **Z** 旗標。同時，**N** 旗標也將受到影響。

指令舉例：

```
bitclr.c r4, r5, 10 // 將 r5 暫存器的 bit10 位置 0，並將結果保存至 r4
```

bittgl.c——暫存器位取反**指令格式：bittgl.c rD, rA, BN**

bittgl 指令可以將通用暫存器 rA 的第 BN 位取反，並更新 Z 旗標。同時，N 旗標也將受到影響。

指令舉例：

```
bittgl.c r4, r5, 10          // 將 r5 暫存器的 bit10 位取反，並將結果保存至 r4
```

4. 資料傳送指令**mv{cond}——條件資料傳輸****指令格式：mv{cond} rD, rA**

mv 指令根據對條件旗標位元的測試結果將 rA 暫存器的內容傳送至 rD 暫存器。

其中，cond 尾碼為 EC (EXECUTE CONDITION) 欄位的條件旗標位元測試條件（如表 2-7 所示），如果測試條件為真，則執行資料傳輸操作，否則 rD 暫存器的內容不作改變。

指令舉例：

```
ldi r4, 0x100
cmp r2, r4                  // 比較 r2 與 0x100 的大小
mvge r4, r2                 // 若 r2>=0x100，則將 r2 的內容傳輸到 r4
```

mtcr——傳輸資料到控制暫存器**指令格式：mtcr rD, Crn**

mtcr 指令可以將通用暫存器 rD 的內容傳輸給控制暫存器 Crn，n 表示控制暫存器編號。通常該指令只能用於核心模式，但如果將 PSR 暫存器中的 cra 位置 1，則可以在用戶模式下使用該指令。

指令舉例：

```
mtcr r4, Cr16                // 將 r4 的內容傳輸給 LDM 實體框數暫存器 Cr16
```

mfcr——從控制暫存器傳輸資料**指令格式：mfcr rD, Crn**

mfcr 指令可以將控制暫存器 Crn 的內容傳輸給通用暫存器 rD，n 表示控制暫存器編號。與 mtcr 類似，通常該指令只能用於核心模式，但如果將 PSR 暫存器中的 cra 位置 1，則可以在用戶模式下使用該指令。

指令舉例：

```
mfcr r4, Cr16                // 將 LDM 實體框數暫存器 Cr16 的內容傳輸給通用暫存器 r4
```

5. 移位運算指令

rol——迴圈左移

指令格式：**rol{i}{c}{.c} rD, rA, operand2**

rol 指令將 rA 暫存器的內容向左迴圈移動 operand2 指定的位數，並存放在 rD 暫存器內。其中，帶尾碼 “i” 表示移動的位數 operand2 由一個 5 位的立即數指定，否則由一個通用暫存器的最低 5 位指定；帶尾碼 “c” 表示帶進位的迴圈移位，否則為不帶進位的迴圈移位。

需要注意的是，帶進位元的迴圈移位指令必須帶有.c 尾碼，因為此時 C 旗標位元將受到影響。

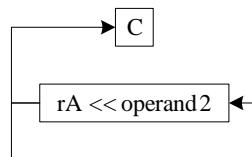


圖 2-5 不帶進位的迴圈左移

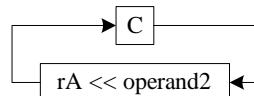


圖 2-6 帶進位的迴圈左移

指令舉例：

```
rol r4, r2, r3          // 不帶進位的迴圈左移，移動位數由暫存器 r3 決定  
rolc.c r4, r2, r3       // 帶進位的迴圈左移，移動位數由暫存器 r3 決定，  
                        // 影響條件旗標位元  
rol i r4, r2, 0x0a       // 不帶進位的迴圈左移，移動位數由立即數 0x0a 決定  
rolic.c r4, r2, 0x03     // 帶進位的迴圈左移，移動位數由立即數 0x03 決定
```

ror——迴圈右移

指令格式：**ror{i}{c}{.c} rD, rA, operand2**

ror 指令將 rA 暫存器的內容向右迴圈移動 operand2 指定的位數，並存放在 rD 暫存器內。其中，帶尾碼 “i” 表示移動的位數 operand2 由一個 5 位的立即數指定，否則由一個通用暫存器的最低 5 位指定；帶尾碼 “c” 表示帶進位的迴圈移位，否則為不帶進位的迴圈移位。

需要注意的是，帶進位元的迴圈移位指令必須帶有.c 尾碼，因為此時 C 旗標位元將受到影響。

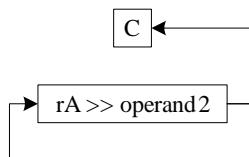


圖 2-7 不帶進位的迴圈右移

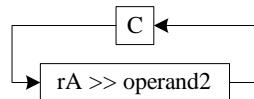


圖 2-8 帶進位的迴圈右移

指令舉例：

ror r4, r2, r3	// 不帶進位的迴圈右移，移動位數由暫存器 r3 決定
rorc.c r4, r2, r3	// 帶進位的迴圈右移，移動位數由暫存器 r3 決定，影響條件旗標位元
rori r4, r2, 0x0a	// 不帶進位的迴圈右移，移動位數由立即數 0x0a 決定
roric.c r4, r2, 0x03	// 帶進位的迴圈右移，移動位數由立即數 0x03 決定

sll——邏輯左移

指令格式：**sll{i}{.c} rD, rA, operand2**

sll 指令將 rA 暫存器的內容向左移動 operand2 指定的位數，並存放在 rD 暫存器內。其中，帶尾碼 “i” 表示移動的位數 operand2 由一個 5 位的立即數指定，否則由一個通用暫存器的最低 5 位指定。

指令舉例：

sll r4, r2, r3	// 邏輯左移，移動位元數由暫存器 r3 決定
slli r4, r2, 0x0a	// 邏輯左移，移動位元數由立即數 0x0a 決定

srl——邏輯右移

指令格式：**srl{i}{.c} rD, rA, operand2**

srl 指令將 rA 暫存器的內容向右移動 operand2 指定的位數，並存放在 rD 暫存器內。其中，帶尾碼 “i” 表示移動的位數 operand2 由一個 5 位的立即數指定，否則由一個通用暫存器的最低 5 位指定。

指令舉例：

srl r4, r2, r3	// 邏輯右移，移動位元數由暫存器 r3 決定
srali r4, r2, 0x0a	// 邏輯右移，移動位元數由立即數 0x0a 決定

sra——算術右移

指令格式：**sra{i}{.c} rD, rA, operand2**

sra 指令將 rA 暫存器的內容向右移動 operand2 指定的位元數，同時用符號位元填充左端的空缺，然後存放在 rD 暫存器內。其中，帶尾碼 “i” 表示移動的位數 operand2 由一個 5 位的立即數指定，否則由一個通用暫存器的最低 5 位指定。

指令舉例：

sra r4, r2, r3	// 算術右移，移動位數由暫存器 r3 決定
srai r4, r2, 0x0a	// 算術右移，移動位數由立即數 0x0a 決定

6. 擴展操作指令

擴展操作指令可以將位元組資料或半字資料的高位元組進行符號填充或零填充而形成字資料。

extsb——擴展有符號位元組資料

指令格式：**extsb{.c} rD, rA**

extsb 指令將 **rA** 暫存器的低位元組進行符號擴展，並將結果保存在 **rD** 暫存器內。執行該操作後，**rD** 暫存器的高三個位元組全部填充為 **rA** 暫存器的最低位元組的符號位元（bit7 位元），如圖 2-9 所示。

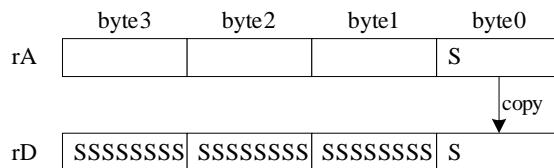


圖 2-9 擴展有符號位元組資料示意圖

指令舉例：

```
ldi r2, 0x80
extsb r4, r2           // 執行該指令後 r4 的內容為 0xffffffff80
ldi r2, 0x70
extsb.c r4, r2         // 執行該指令後 r4 的內容為 0x00000070，條件旗標位元相應改變
```

extzb——擴展無符號位元組資料

指令格式：**extzb{.c} rD, rA**

extzb 指令將 **rA** 暫存器的低位元組進行零擴展，並將結果保存在 **rD** 暫存器內。執行該操作後，**rD** 暫存器的高三個位元組全部被填充為 0，如圖 2-10 所示。

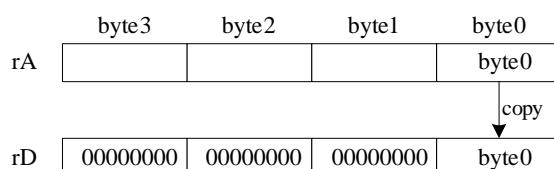


圖 2-10 擴展無符號位元組資料示意圖

指令舉例：

```
ldi r2, 0x80
extzb r4, r2           // 執行該指令後 r4 的內容為 0x00000080
ldi r2, 0x70
extzb.c r4, r2         // 執行該指令後 r4 的內容為 0x00000070，條件旗標位元相應改變
```

extsh——擴展有符號半字資料**指令格式：extsh{.c} rD, rA**

extsh 指令將 rA 暫存器的低半字進行符號擴展，並將結果保存在 rD 暫存器內。執行該操作後，rD 暫存器的高半字全部填充為 rA 暫存器的低半字的符號位元（bit15 位元），如圖 2-11 所示。

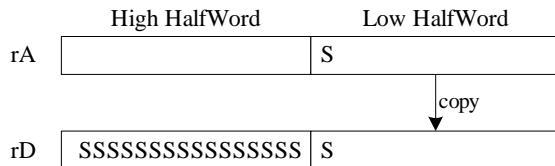


圖 2-11 擴展有符號半字資料示意圖

指令舉例：

```
ldi r2, 0x8000
extsh r4, r2          // 執行該指令後 r4 的內容為 0xffff8000
ldi r2, 0x7000
extsh.c r4, r2        // 執行該指令後 r4 的內容為 0x00007000，條件旗標位元相應改變
```

extzh——擴展無符號半字資料**指令格式：extzh{.c} rD, rA**

extzh 指令將 rA 暫存器的低半字進行零擴展，並將結果保存在 rD 暫存器內。執行該操作後，rD 暫存器的高半字全部被填充為 0，如圖 2-12 所示。

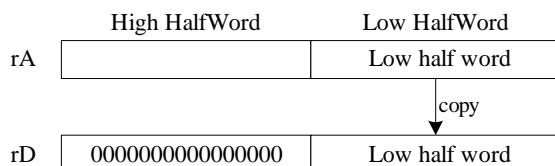


圖 2-12 擴展無符號半字資料示意圖

指令舉例：

```
ldi r2, 0x8000
extzh r4, r2          // 執行該指令後 r4 的內容為 0x00008000
ldi r2, 0x7000
extzh.c r4, r2        // 執行該指令後 r4 的內容為 0x00007000，條件旗標位元相應改變
```

2.3.3 分支指令

分支指令用來實現程式的跳越。表 2-8 所示為 S+core7 的分支指令表。其中，尾碼 “I” 表示帶鏈接的跳越，帶有該尾碼後在跳越前將把本條指令後的下一條指令的位址置入鏈接暫存器 r3 中。

表 2-8 S+core7 分支指令表

助記符	說明	格式
j	無條件跳越	j{l} label
b	條件分支	b{cond}{l} label
br	條件暫存器分支	br{cond}{l} rA

j——無條件跳越

指令格式：j{l} label

j 指令跳越到指定位址處執行程式，並限制跳越範圍為當前指令的±16MB 範圍內。其中 label 是具有 24 位有效位數的偏移位址。目標位址是這樣形成的：先將 24 位偏移位址左移一位，然後在與本條指令的高 7 位結合，形成目標位址。

指令舉例：

```
j label           // 跳越至 label 標號處
jl label          // 跳越至 label 標號處並保存下一條指令位址，可以用於函數調用
```

b——條件分支

指令格式：b{cond}{l} label

b 指令根據 cond 所代表的 BC (Branch Condition) 欄位的條件旗標位元測試結果分支到目標位址。其中 label 是具有 19 位元有效位數的分支偏移量。目標位址為分支偏移量左移一位元並作符號擴展與當前 PC 內容的位址之和。

BC (Branch Condition) 欄位的 cond 編碼如表 2-9 所示。

表 2-9 BC 欄位的 cond 編碼意義表

序號	cond 尾碼	操作	測試
0	cs{l}	當 C 為 1 時執行操作	C
1	cc{l}	當 C 為 0 時執行操作	~C
2	gtu{l}	當 C 為 1 且 Z 為 0 時執行操作	C & ~Z
3	leu{l}	當 C 為 0 或 Z 為 1 時執行操作	~C Z
4	eq{l}	當 Z 為 1 時執行操作	Z
5	ne{l}	當 Z 為 0 時執行操作	~Z
6	gt{l}	當 Z 為 0 且 N 為 V 時執行操作	(Z=0)&(N = V)
7	le{l}	當 Z 為 1 且 N 不為 V 時執行操作	(Z=1)&(N != V)

序號	cond 尾碼	操作	測試
8	ge{I}	當 N 為 V 時執行操作	N=V
9	lt{I}	當 N 不為 V 時執行操作	V!=V
10	mi{I}	當 N 為 1 時執行操作	N
11	pl{I}	當 N 為 0 時執行操作	~N
12	vs{I}	當 V 為 1 時執行操作	V
13	vc{I}	當 V 為 0 時執行操作	~V
14	cnez{I}	CNT>0 時執行操作	CNT>0
15	al{I}	無條件執行操作	-

指令舉例：

```
beq label           // Z=0 時分支至 label 標號處
bgei label          // N 為 V 時分支至 label 標號處並保存下一條指令位址
                    // 可以用於函數調用
```

br——條件暫存器分支

指令格式：**br{cond}{I} rA**

br 指令根據對條件旗標位元的測試結果分支到由 rA 暫存器指定的目標位址。

分支條件 cond 如表 2-9所示。

指令舉例：

```
breq r3             // Z=0 時分支至 r3 指定的位址處
bregel r3            // N 為 V 時分支至 r3 指定的位址處並保存下一條指令位址
                    // 可以用於函數調用
```

2.3.4 特殊指令

S+core7 處理器具有以下幾種特殊指令：Custom Engine 指令、特殊暫存器控制指令、Cache 控制指令、系統控制指令等。表 2-10所示為 S+core7 特殊指令表：

表 2-10 S+core7 特殊指令表

助記符	說明	格式
ceinst	Custom Engine 用戶定義	ceinst func5, rA, rB, USD1, USD2
mtcex	傳送資料到 Custom Engine 暫存器	mtcel rD mtceh rD

助記符	說明	格式
		mtcehl rD, rA
mfcecx	從 Custom Engine 暫存器傳送資料到通用暫存器	mfcel rD mfceh rD mfcehl rD, rA
mtsrx	傳送資料到特殊暫存器	mtsrx rA, Srx
mfsrx	從特殊暫存器傳送資料到通用暫存器	mfsrx rD, Srx
cache	cache 控制指令	cache cache_op, [rA, SImm15]
trap	條件陷阱	trap{cond} Software_Parameter(Imm5)
syscall	系統調用	syscall Software_Parameter(Imm15)
sleep	睡眠	sleep
sdbbp	軟體 Debug 斷點	sdbbp code(Imm5)
pflush	清空管線	pflush
rte	從異常返回	rte
drte	從 debug 異常返回	drte

1. Custom Engine 指令

Custom Engine 指令主要用於 Custom Engine 指令擴展以及對 Custom Engine 暫存器 CEH 和 CEL 的操作。

ceinst——Custom Engine 用戶定義

指令格式：**ceinst func5, rA, rB, USD1, USD2**

ceinst 格式的指令是為了實現 custom engine 指令擴展而定義的。其中的 func5 定義了具體的 Custom Engine 操作，且該格式指令允許至多用兩個通用暫存器作為源運算元。在該指令中定義了 rA 和 rB 欄位，若需要指定一或者兩個源通用暫存器，則相應暫存器索引值就會被置於這兩個欄位內。USD1 和 USD2 為用戶自定欄位，這兩個欄位既可以是用於計算的立即數，亦可以是用於操作控制的參數，或者是目標通用暫存器索引號。若 Custom Engine 不執行這類擴展指令，則會發生未定義指令異常。

指令舉例：

```
ceinst 0x00, r2, r4, 0x01, 0x02
```

mtcex——傳送資料到 Custom Engine 暫存器

指令格式：**mtcel rD**

mtceh rD

mtcehl rD, rA

mtcex 指令可以將通用暫存器的內容寫入 Custom Engine 暫存器 CEL 或 CEH 內。**mtcel** 指令將 rD 暫存器內容寫入 CEL；**mtceh** 指令將 rD 暫存器內容寫入 CEH；**mtcehl** 指令將 rD 暫存器內容寫入 CEH，並將 rA 暫存器內容寫入 CEL。

指令舉例：

```
mtcel r4 // 將 r4 暫存器內容傳遞給 CEL
```

mfcecx——從 Custom Engine 暫存器傳送資料到通用暫存器

指令格式：**mfcel rD**

mfceh rD

mfcehl rD, rA

mfcecx 指令可以將 Custom Engine 暫存器 CEL 或 CEH 的內容傳遞給通用暫存器。**mfcel** 指令將 CEL 內容寫入 rD 暫存器；**mfceh** 指令將 CEH 內容寫入 rD 暫存器；**mfcehl** 指令將 CEH 內容寫入 rD 暫存器，並將 CEL 內容寫入 rA 暫存器。

指令舉例：

```
mfcel r4 // 將 CEL 內容傳遞給 r4 暫存器
```

2. 特殊暫存器控制指令

特殊暫存器控制指令主要完成通用暫存器與特殊暫存器之間的資料傳輸操作。

mtsrx——傳送資料到特殊暫存器

指令格式：**mtsrx rA, Srn**

mtsrx 指令將通用暫存器 rA 的內容傳送給特殊暫存器 Srn (n 為特殊暫存器編號)。

指令舉例：

```
mtsrx r4, Srl
```

mfsrx——從特殊暫存器傳送資料到通用暫存器

指令格式：**mfsrx rD, Srn**

mfsrx 指令將特殊暫存器 Srn (n 為特殊暫存器編號) 的內容傳送給通用暫存器 rD。

指令舉例：

```
mfsrx r4, Srl
```

3. Cache 控制指令

cache——Cache 控制指令

指令格式：cache cache_op, [rA, \$Imm15]

cache 指令根據 cache_op 碼進行 Cache 操作。該操作將基底位址暫存器 rA 的內容與 15 位立即數 \$Imm15 相加形成一個有效的虛擬位址，並透過固定對映模式的 MMU 轉換成一實體位址。cache_op 操作碼對應的操作如表 2-11 所示。

表 2-11 Cache OP 欄位編碼表

cache_op [4:0]	I-Cache/ D-Cache	功能	操作位址
0x00	I-Cache	預取一 Cache 行	rA + \$Imm15
0x01	I-Cache	預取和鎖定一 Cache 行	rA + \$Imm15
0x02	I-Cache	無效並解鎖一 Cache 行	rA + \$Imm15
0x03	I-Cache	填充 PFN 和 Size 值到 LIM	PFN 由 rA 指定 Size 由 \$Imm15 指定
0x04	I-Cache	從新填充 PFN 和 Size 值到 LIM	任意
0x08	D-Cache	預取一 Cache 行	rA + \$Imm15
0x09	D-Cache	預取和鎖定一 Cache 行	rA + \$Imm15
0x0A	D-Cache	無效並解鎖一 Cache 行	rA + \$Imm15
0x0B	D-Cache	填充 PFN 和 Size 值到 LDM	PFN 由 rA 指定 Size 由 \$Imm15 指定
0x0C	D-Cache	將 LDM 內的 PFN 和 Size 值寫回主記憶體	PFN 由 rA 指定 Size 由 \$Imm15 指定
0x0D	D-Cache	強制將一個被改寫過的有效 Cache 行中的內容寫回主記憶體並將該行改寫為有效	rA + \$Imm15
0x0E	D-Cache	強制將一個被改寫過的有效 Cache 行中的內容寫回主記憶體並將改行改寫為無效	rA + \$Imm15
0x10	I-Cache	無效全部 Cache	任意
0x11	I-Cache	當指令預取緩存功能使能時，使用該指令使指令預取緩存功能禁止； 當指令預取緩存功能禁止時，使用該指令使指令預取緩存功能使能；	任意

cache_op [4:0]	I-Cache/ D-Cache	功能	操作位址
0x18	D-Cache	無效全部 Cache	任意
0x1A	D-Cache	清空寫暫存器	任意
0x1B	D-Cache	啓動寫緩存功能	任意
0x1D	D-Cache	啓動複寫 D-Cache 功能（使能/禁止）	任意
0x1E	D-Cache	強制將所有被改寫過的有效 D-Cache 行寫回主記憶體，並改為有效	任意
0x1F	D-Cache	強制將所有被改寫過的有效 D-Cache 行寫回主記憶體，並改為無效	任意

指令舉例：

```
cache 0x00, [r2, 0x0123]
```

4. 系統控制指令

trap——條件陷阱

指令格式：**trap{cond} Software_Parameter(Imm5)**

trap 指令是一種控制指令，僅用於核心模式，或在 PSR 暫存器中的 cra 位為 1 時可用於用戶模式。**trap** 指令根據 **cond** 所代表的 EC (EXECUTE CONDITION) 條件旗標位元的測試結果（見表 2-7）觸發陷阱異常。即當測試結果為真時觸發陷阱異常，進入相應的異常處理。陷阱異常發生後，只有以下暫存器的值會改變：

- 處理器狀態暫存器 (PSR) 中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；
- 條件暫存器 (CR1) 的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位
- 異常原因暫存器 (ECR) 中的 Exc_code 欄位碼為 10；
- EPC 暫存器指向 **trap{cond}** 指令。

指令舉例：

```
trapeq 0x008e
```

syscall——系統調用陷阱

指令格式：**syscall Software_Parameter(Imm15)**

syscall 指令可以引發系統調用異常，並進入相應的異常處理程式。該異常發生後，只有以下暫存器的值會改變：

- 處理器狀態暫存器 (PSR) 中的 IEc 與 UMc 位的值被分別壓入 IEs 與 UMs 位；

- 條件暫存器（CR1）的 Tc、Nc、Zc、Cc 及 Vc 位的值分別壓入 Ts、Ns、Zs、Cs 及 Vs 位；
- 異常原因暫存器（ECR）中的 Exc_code 欄位碼為 7；
- EPC 暫存器指向 `syscall` 指令。

指令舉例：

```
syscall 0x008e
```

sleep——睡眠

指令格式：**sleep**

sleep 指令使處理器內核進入省電待機模式，直至收到外部中斷（針對 SPCE3200 晶片 40 個中斷源和鍵喚醒）、非遮罩中斷或 Debug 中斷信號為止。執行睡眠指令之前，用戶需透過 PSR 暫存器中的 IE 欄位和 ECR 暫存器中的 IM 欄位將這些中斷條件使能。本指令只能用於核心模式，或在 PSR 暫存器中的 cra 位為 1 時可用於用戶模式。

sdbbp——軟體 Debug 斷點

指令格式：**sdbbp code(Imm5)**

在非除錯狀態下，**sdbbp** 指令將引發軟體斷點 Debug 異常，並進入異常處理程式。該異常的向量位址取決於線上仿真（ICE）和仿真資訊獲取（Probe）電路。執行本指令後，Debug 暫存器（DREG）中相應的狀態位元會置位元，除錯異常程式計數器（DEPC）將指向 **sdbbp** 指令的位址。其中，code 欄位指定 5 位元軟體參數值。

在除錯狀態下，該指令相當於空操作。

指令舉例：

```
sdbbp 0x0e
```

pflush——清空管線

指令格式：**pflush**

在對於管線型處理器，有些指令序列需要插入軟體 bubble（即 `nop` 指令），以防止管線衝突導致資料錯誤。例如，在莊子合併指令跟隨一目標暫存器為 LCR 的 `mfsr` 指令的情況下，需要 3 個 `nop` 指令，在這種情況下，也可以插入一個 **pflush** 指令，來代替若干個 `nop` 指令，使得軟體編程更方便。

pflush 指令的執行類似於跳越到下一條指令，即該指令首先清空下一條指令的管線狀態，再從 (`pfluash+4`) 位址獲取下一指令繼續執行，這很像非管線處理器的動作。

rte——從異常返回**指令格式：rte**

rte 指令用於從中斷或異常服務程式副程式返回，其操作類似於暫存器跳越（**br**）指令。執行完 **rte** 指令後，程式計數器（PC）的值從 EPC 恢復至中斷之前的位址，同時，處理器狀態暫存器（PSR）中的 UMc 和 IEc 欄位分別恢復自 UMs 和 IEs 欄位；條件暫存器中的 Tc、Nc、Zc、Cc 和 Vc 欄位分別恢復自 Ts、Ns、Zs、Cs 和 Vs 欄位。本指令只能用於核心模式，或在 PSR 暫存器中的 cra 位為 1 時可用於用戶模式。

drte——從 Debug 異常返回**指令格式：rte**

rte 指令用於從 Debug 異常服務程式副程式返回。執行完 **rte** 指令後，程式計數器（PC）的值從 DEPC 恢復至中斷之前的位址。本指令只能用於核心模式，或在 PSR 暫存器中的 cra 位為 1 時可用於用戶模式。

2.3.5 協同處理器指令

按照當前 CPU 核心的設計理念，需要盡可能滿足各種用戶的要求，S+core7 為用戶留出了這些介面，在設計各種晶片的時候可以掛接最多三個協同處理器，輔助完成用戶功能。這樣設計的好處是，特殊的操作可以交由協同處理器完成。協同處理器的控制要透過協同處理器指令實現。表 2-12 所示為 S+core7 的協同處理器指令表。

表 2-12 S+core7 協同處理器指令表

助記符	說明	格式
COPn	協同處理器功能擴展，可用戶定義（n 為協同處理器編號）	COPn cop_code20 或 COPn CrD, CrA, CrB, cop_code5
MTCn	傳送資料到協同處理器資料暫存器	MTCn rD, CrA
MTCCn	傳送資料到協同處理器控制暫存器	MTCCn rD, CrA
MFCn	從協同處理器資料暫存器傳送資料	MFCn rD, CrA
MFCCn	從協同處理器控制暫存器傳送資料	MFCCn rD, CrA
LDCn	裝載資料到協同處理器資料暫存器	LDCn CrA, [rD, SImm12]
STCn	從協同處理器資料暫存器存儲資料	STCn CrA, [rD, SImm12]

COPn——協同處理器功能擴展**指令格式：COPn cop_code20****COPn CrD, CrA, CrB, cop_code5**

其中，n 為 1~3，表示協同處理器編號；cop_code20 指定 20 位協同處理器用戶定義碼；CrD，CrA，CrB 指定用戶定義欄位編號；cop_code5 指定 5 位協同處理器用戶定義碼。

該指令是用戶定義的協同處理器擴展指令，分別有參數為 20 位和參數為 5 位兩種類型的格式。需要注意的是，處理器狀態暫存器（PSR）中的 CU 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。另外，如果沒有協同處理器回應該指令，則會產生未定義指令異常。

MTCn——傳送資料到協同處理器資料暫存器

指令格式： MTCn rD, CrA

其中，n 為 1~3，表示協同處理器編號。該指令將通用暫存器 rD 的內容傳遞給指定協同處理器的資料暫存器 CrA。需要注意的是，處理器狀態暫存器（PSR）中 cu 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
mtc3 r14, cr15
```

MTCCn——傳送資料到協同處理器控制暫存器

指令格式： MTCCn rD, CrA

其中，n 為 1~3，表示協同處理器編號。該指令將通用暫存器 rD 的內容傳遞給指定協同處理器的控制暫存器 CrA。需要注意的是，處理器狀態暫存器（PSR）中 cu 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
mtcc3 r14, cr15
```

MFCn——從協同處理器資料暫存器傳送資料

指令格式： MFCn rD, CrA

其中，n 為 1~3，表示協同處理器編號。該指令將指定協同處理器的資料暫存器 CrA 的內容傳遞給通用暫存器 rD。需要注意的是，處理器狀態暫存器（PSR）中 cu 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
mfc2 r4, cr8
```

MFCCn——從協同處理器控制暫存器傳送資料

指令格式： MFCCn rD, CrA

其中，n 為 1~3，表示協同處理器編號。該指令將指定協同處理器的控制暫存器 CrA 的內容傳遞給通用暫存器 rD。需要注意的是，處理器狀態暫存器（PSR）中 cu 欄位相應位須使能，

否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
mfcc3 r4, cr8
```

LDCn——裝載資料到協同處理器資料暫存器

指令格式： **LDCn CrA, [rD, SImm12]**

其中，n 為 1~3，表示協同處理器編號。該指令從記憶體裝載一字資料到指定協同處理器的資料暫存器 CrA 中，記憶體的操作位址由通用暫存器 rD 的內容與左移 2 位的 10 位有符號數 Imm10 之和指定。注意，記憶體操作位址須為字對齊，否則將發生位址對齊錯誤異常。另外，處理器狀態暫存器 (PSR) 中 cu 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
ldc3 cr8, [r4, 0x0012]
```

STCn——從協同處理器資料暫存器存儲資料

指令格式： **STCn CrA, [rD, SImm12]**

其中，n 為 1~3，表示協同處理器編號。該指令將指定協同處理器的資料暫存器 CrA 的內容存儲到記憶體指定位址。記憶體的操作位址由通用暫存器 rD 的內容與左移 2 位元的 10 位元有符號數 Imm10 之和指定。注意，記憶體操作位址須為字對齊，否則將發生位址對齊錯誤異常。另外，處理器狀態暫存器 (PSR) 中 cu 欄位相應位須使能，否則執行該指令將引起協同處理器使用異常。

指令舉例：

```
stc3 cr8, [r4, 0x0012]
```

2.4 16 位指令集

S+core7 處理器支援完全的 16 位/32 位混合編程，兩種指令集之間不需要進行任何切換。

S+core7 處理器的所有 16 位指令的助記符均帶有 “!”，以和 32 位指令進行區分。大多數 32 位的通用指令加上 “!” 可以轉換成對應的 16 位指令。

16 位指令集包括裝載與存儲指令、資料處理指令、跳越與分支指令和特殊指令四類指令。其中：

- 裝載與存儲指令可以完成對記憶體的存取操作；
- 資料處理指令完成算術和邏輯運算等操作；
- 跳越與分支指令完成程式跳越以及函數調用等操作；
- 特殊指令僅有一條特殊控制指令，用於軟體 Debug 斷點操作。

另外，S+core7 處理器可以將兩條 16 位指令構成特有的並行條件執行 (PCE) 指令。下麵將分別進行介紹。

2.4.1 裝載與存儲指令

16 位裝載/存儲指令僅具有三種基本的定址方式：立即數定址、基底位址定址和基底位址變址定址。另外，還有兩種定址方式：前減量變址定址和後增量變址定址，分別用於 **push!** 和 **pop!** 兩條指令，可以實現對堆疊的操作。

表 2-13 S+core716 位裝載/存儲指令表

助記符	說明	格式
lbu{p}!	裝載無符號位元組資料	lbu! rD _{g0} , [rA _{g0}] 基底位址定址
		lbus! rD _{g0} , Imm5 基底位址變址定址
lh{p}!	裝載半字資料	lh! rD _{g0} , [rA _{g0}] 基底位址定址
		lhp! rD _{g0} , Imm6 基底位址變址定址
lw{p}!	裝載字資料	lw! rD _{g0} , [rA _{g0}] 基底位址定址
		lwp! rD _{g0} , Imm7 基底位址變址定址
sb{p}!	存儲位元組資料	sb! rD _{g0} , [rA _{g0}] 基底位址定址
		sbp! rD _{g0} , Imm5 基底位址變址定址
sh{p}!	存儲半字資料	sh! rD _{g0} , [rA _{g0}] 基底位址定址
		shp! rD _{g0} , Imm6 基底位址變址定址
sw{p}!	存儲字資料	sw! rD _{g0} , [rA _{g0}] 基底位址定址
		swp! rD _{g0} , Imm7 基底位址變址定址
ldiu!	裝載無符號立即數	ldiu! rD _{g0} , Imm8
push!	存儲字資料（前減量變址定址）	push! rD _{g0} , [rA _{g0}]
pop!	裝載字資料（後增量變址定址）	pop! rD _{g0} , [rA _{g0}]

(1) 立即數定址格式：

ldiu! rD_{g0}, Imm8

立即數定址僅用於 **ldiu!** 指令中，將立即數作為運算元直接參與操作。在該指令中，是將八位立即數 **Imm8** 裝載進低組通用暫存器 **rD_{g0}** 內。

指令舉例：

ldiu! r3, 0x80	// 將立即數 0x80 裝載進 r3 暫存器
----------------	-------------------------

(2) 基底位址定址格式：

<opcode>! rD_{g0}, [rA_{g0}]

基底位址定址是以源暫存器內的內容作為操作位址對記憶體進行裝載/存儲操作。

指令舉例：

```
lh! r3, [r2]           // 將 r2 的內容為位址的存儲單元的半字資料裝載進 r3 暫存器  
sw! r3, [r2]           // 將 r3 暫存器的字資料存儲到 r2 的內容為位址的存儲單元內
```

(3) 基底位址變址定址格式：

```
<opcode>p! rDg0, Imm
```

基底位址變址定址是以基底位址指標暫存器 r2 的內容加上立即數偏移量做為操作位址對記憶體進行裝載/存儲操作。在指令後加上 p 尾碼則表示使用基底位址變址定址。需要注意的是，在執行半字資料或字資料的裝載/存儲操作時，偏移量需保證位址對齊。

指令舉例：

```
lhp! r3, 0x04          // 將 r2 的內容加上 0x04 做為位址指向的存儲單元的半字資料  
                      // 裝載進 r3 暫存器  
swp! r3, 0x10          // 將 r2 的內容加上 0x10 做為位址指向的存儲單元的字資料  
                      // 裝載進 r3 暫存器
```

(4) 前減量變址定址格式：

```
push! rDg0, [rAg0]
```

前減量變址定址僅用於 push! 指令中，以基底位址暫存器 rA_{g0} 的內容減去 4 做為操作位址對記憶體進行操作，並且，當操作完成後，基底位址暫存器 rA_{g0} 的內容更新為該操作位址。push! 指令完成將 rD_{g0} 暫存器的內容存儲至該操作位址指向的存儲單元內。

需要注意的是，記憶體操作位址需為字邊界對齊，否則將發生位址對齊錯誤異常。

指令舉例：

```
push! r15, [r2]         // r2 內容減 4，然後將暫存器 r15 的內容壓入 r2 指向的存儲單元
```

(5) 後增量變址定址格式：

```
pop! rDg0, [rAg0]
```

後增量變址定址僅用於 pop! 指令中，以基底位址暫存器 rA_{g0} 的內容做為操作位址對記憶體進行操作，並且，當操作完成後，基底位址暫存器 rA_{g0} 的內容更新為該操作位址加 4。pop! 指令完成將操作位址指向的存儲單元內的內容裝載至 rD_{g0} 暫存器。

需要注意的是，記憶體操作位址需為字邊界對齊，否則將發生位址對齊錯誤異常。

指令舉例：

```
pop! r15, [r2]           // 將 r2 指向的存儲單元的資料裝載至 r15  
                      // 然後將 r2 的內容加 4 並更新
```

2.4.2 資料處理指令

S+core7 處理器的 16 位資料處理指令可以分為五類：算術運算指令、邏輯運算指令、位運算指令、資料傳送指令和移位運算指令。與 32 位指令不同的是，16 位資料處理指令均可以影響條件旗標位而不需要帶有.c 尾碼。表 2-14 所示為 S+core7 的 16 位資料處理指令表。

表 2-14 S+core7 16 位資料處理指令表

助記符	說明	格式	完成操作
add!	16 位元加法運算	add! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} + rA _{g0}
addc!	16 位元帶進位加法運算	addc! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} + rA _{g0} + C
addei!	16 位元立即數加法運算	addei! rD _{g0} , Imm4	rD _{g0} = rD _{g0} + 2 ^{Imm4}
sub!	16 位元減法運算	sub! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} - rA _{g0}
subei!	16 位元立即數減法運算	subei! rD _{g0} , Imm4	rD _{g0} = rD _{g0} - 2 ^{Imm4}
neg!	16 位元取負	neg! rD _{g0} , rA _{g0}	rD _{g0} = -rA _{g0}
cmp!	16 位元比較	cmp! rD _{g0} , rA _{g0}	做 rD _{g0} - rA _{g0} 運算並根據結果對條件旗標位元置位
and!	16 位元邏輯與	and! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} 按位與 rA _{g0}
or!	16 位元邏輯或	or! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} 按位或 rA _{g0}
xor!	16 位元邏輯異或	xor! rD _{g0} , rA _{g0}	rD _{g0} = rD _{g0} 按位異或 rA _{g0}
not!	16 位元邏輯異非	not! rD _{g0} , rA _{g0}	rD _{g0} = rA _{g0} 取反
t{cond}!	16 位元測試並置 T 條件旗標	t{cond}!	同 32 位 t 指令
bittst!	16 位元位測試	bittst! rD _{g0} , BN(Imm5)	測試 rD _{g0} 的第 BN 位並修改 N 和 Z 旗標
bitset!	位元置位	bitset! rD _{g0} , BN	將 rD _{g0} 第 BN 位置 1
bitclr.c	位元清零	bitclr! rD _{g0} , BN	將 rD _{g0} 第 BN 位置 0
bittgl.c	位翻轉	bittgl! rD _{g0} , BN	將 rD _{g0} 第 BN 位取反
mv!	16 位元暫存器資料傳送 (低組->低組)	mv! rD _{g0} , rA _{g0}	rD _{g0} = rA _{g0}
mlfh!	16 位元暫存器資料傳送 (高組->低組)	mlfh! rD _{g0} , rA _{g1}	rD _{g0} = rA _{g1}
mhfl!	16 位元暫存器資料傳送 (低組->高組)	mhfl! rD _{g1} , rA _{g0}	rD _{g1} = rA _{g0}
sll!	16 位元邏輯左移	sll! rD _{g0} , rA _{g0}	將 rD _{g0} 內容邏輯左移 rA _{g0} 位並保存至 rD _{g0}
slli!	16 位元立即數邏輯左移	slli! rD _{g0} , Imm5	將 rD _{g0} 內容邏輯左移 Imm5 位並保存至 rD _{g0}
sra!	16 位元算術右移	sra! rD _{g0} , rA _{g0}	將 rD _{g0} 內容算術右移 rA _{g0} 位並保存至 rD _{g0}

助記符	說明	格式	完成操作
srl!	16 位元邏輯右移	srl! rD _{g0} , rA _{g0}	將 rD _{g0} 內容邏輯右移 rA _{g0} 位並保存至 rD _{g0}
srl!l	16 位元立即數邏輯右移	srl!l rD _{g0} , Imm5	將 rD _{g0} 內容邏輯右移 Imm5 位並保存至 rD _{g0}

2.4.3 跳越與分支指令

S+core7 處理器具有 j{l}!、b{cond}! 和 br{cond}! 三條跳越與分支指令。指令格式和功能解釋如下：

j{l}!——無條件跳越

指令格式：j{l}! label

該指令首先將 **label** 代表的 11 位偏移位址左移 1 位，並與當前指令的高 20 位結合形成目標位址，進行分支操作。同時，若指令帶有 l 尾碼，則使用下一條指令的位址更新連接暫存器 GPR_{r3}。

指令舉例：

j! label	// 跳越至 label 標號處，跳越偏移限制在±2K 範圍內
j!l label	// 跳越至 label 標號處並保存下一條指令位址，可以用於函數調用

b{cond}!——條件分支

指令格式：b{cond}! label

該指令的分支條件 cond 見表 2-7。label 的有效位數為 8 位（7 位元有效偏移經符號擴展並左移 1 位）。注意，該指令不具備鏈接功能。

指令舉例：

beq! label	// z=0 時分支至 label 標號處
------------	-----------------------

br{cond}!——條件暫存器分支

指令格式：br{cond}! rAg0

br 指令根據對條件旗標位元的測試結果分支到由 rA 暫存器指定的目標位址。分支條件 cond 如表 2-9 所示。

指令舉例：

breq! r3	// z=0 時分支至 r3 指定的位址處
----------	-----------------------

2.4.4 特殊指令

S+core7 處理器的 16 位指令集中僅有一條特殊控制指令：**sdbbp!**，用於軟體 Debug 斷點操作。

指令格式：sdbbp! code(Imm5)

該指令的功能與 32 位指令集中的 **sdbbp** 指令相同。

指令舉例：

```
sdbbp! 0x00
```

2.4.5 並行條件執行

S+core7 處理器特有並行條件執行（PCE）功能。將兩條 16 位指令使用 “||” 連接起來，便構成了並行條件執行結構。如下所示：

```
ADD! r2, r7 || SUB! r2, r7
```

處理器根據並行執行條件旗標 T 來決定執行兩條指令中的哪一條。當 T=true 時，處理器執行前者，即 ADD! r2, r7；當 T=false 時，處理器執行後者，即 SUB! r2, r7。

一般並行條件執行結構的指令需要配合 T 指令共同工作。首先使用 T 指令修改 T 旗標，接著使用並行條件執行結構讓處理器判斷應該執行哪一條指令。

並行條件執行結構有效解決了程式跳越問題，使管線結構的處理器能更大效能的發揮其處理性能。

2.5 合成指令集

在 S+core7 處理器的 32 位指令集中，部分指令可以等效為兩條或兩條以上的指令操作，從而方便用戶的編程。

li——裝載立即數

指令格式：li rD, Imm32

該指令可以將一個 32 位的立即數裝載至通用暫存器 rD 內。

- 當 Imm32>-32768 且 Imm32<32767 時，該指令等效於 ldi rD, SImm16，如 li r3, 0x1234；
 - 當 Imm32 的低半字為零時，該指令等效於 ldis rD, SImm16，如 li r3, 0x12340000；
 - 其他情況下，該指令等效於下麵兩步操作：
 - ldis rD, Hi16(Imm32)
 - ori rD, Lo16(Imm32)
- 如：li r3, 0x12345678

Ia——裝載立即數**指令格式：** **Ia rD, label****Ia rD, value**

該指令可以將一個符號的位址裝或一個 32 位的立即數裝載至通用暫存器 rD 內。其中，label 為符號名，value 為 32 位立即數。

■ Ia rD, label 等效於下麵兩步操作：

- ldis rD, Hi16(label)
- ori rD, Lo16(label)

如：Ia r3, label

■ Ia rD, value 等效於下麵兩步操作：

- ldis rD, Hi16(value)
- ori rD, Lo16(value)

如：Ia r3, 0x1234

Ib——裝載有符號位元組資料**指令格式：** **Ib rD, [rA]****Ib rD, label****Ib rD, value**

該指令將一個有符號位元組資料經符號擴展裝載進通用暫存器 rD 內。其中，label 為符號名，value 為 32 位立即數，表示記憶體的操作位址（絕對定位）。

■ Ib rD, [rA] 等效於 Ib rD, [rA, 0x0000]。

如：Ib r2, [r3]

■ Ib rD, label 等效於下麵兩步操作：

- Ia r1, label
- Ib rD, [r1]

如：Ib r6, label

當 label 處於 sbss 或 sdata 中，則 Ib rD, label 等效於 Ib rD, [gp, offset]。其中，offset 為 label 和 r28 之間的偏移量。

■ Ib rD, value 等效於下麵兩步操作：

- Ia r1, value
- Ib rD, [r1]

該指令使用 value 所表示的值做為位址進行定址操作。如：Ib r2, 0

Ibu——裝載無符號位元組資料**指令格式：** Ibu rD, [rA]**Ibu rD, label****Ibu rD, value**

該指令將一個無符號位元組資料經零擴展裝載進通用暫存器 rD 內，它的使用方式和等效方式與前面的 Ib 類似。

指令舉例：

```
Ibu r2, [r3]           // 等效於 Ibu r2, [r3, 0]
Ibu r6, label          // 等效於 la r1, label 和 Ibu r6, [r1]兩條指令
Ibu r2, 0               // 絕對定位，等效於 la r1, 0 和 Ibu r2, [r1]兩條指令
```

Ih——裝載有符號半字資料**指令格式：** Ih rD, [rA]**Ih rD, label****Ih rD, value**

該指令將一個有符號半字資料經符號擴展裝載進通用暫存器 rD 內，它的使用方式和等效方式與前面的 Ib 類似。

指令舉例：

```
Ih r2, [r3]           // 等效於 Ih r2, [r3, 0]
Ih r6, label          // 等效於 la r1, label 和 Ih r6, [r1]兩條指令
Ih r2, 0               // 絕對定位，等效於 la r1, 0 和 Ih r2, [r1]兩條指令
```

Ihu——裝載無符號半字資料**指令格式：** Ih rD, [rA]**Ih rD, label****Ih rD, value**

該指令將一個無符號半字資料經零擴展裝載進通用暫存器 rD 內，它的使用方式和等效方式與前面的 Ib 類似。

指令舉例：

```
Ihu r2, [r3]           // 等效於 Ihu r2, [r3, 0]
Ihu r6, label          // 等效於 la r1, label 和 Ihu r6, [r1]兩條指令
Ihu r2, 0               // 絕對定位，等效於 la r1, 0 和 Ihu r2, [r1]兩條指令
```

lw——裝載字資料**指令格式： lw rD, [rA]****lw rD, label****lw rD, value**

該指令將一個字資料裝載進通用暫存器 rD 內，它的使用方式和等效方式與前面的 lb 類似。

指令舉例：

```
lw r2, [r3]           // 等效於 lw r2, [r3, 0]
lw r6, label          // 等效於 la r1, label 和 lw r6, [r1]兩條指令
lw r2, 0              // 絕對定位，等效於 la r1, 0 和 lw r2, [r1]兩條指令
```

sb——存儲位元組資料**指令格式： sb rD, [rA]****sb rD, label****sb rD, value**

該指令將通用暫存器 rD 低 8 位內容寫入到記憶體，它的使用方式和等效方式與前面的 lb 類似。

指令舉例：

```
sb r2, [r3]           // 等效於 sb r2, [r3, 0]
sb r6, label          // 等效於 la r1, label 和 sb r6, [r1]兩條指令
sb r2, 0              // 絕對定位，等效於 la r1, 0 和 sb r2, [r1]兩條指令
```

sh——存儲半字資料**指令格式： sh rD, [rA]****sh rD, label****sh rD, value**

該指令將通用暫存器 rD 低 16 位內容寫入到記憶體，它的使用方式和等效方式與前面的 lb 類似。

指令舉例：

```
sh r2, [r3]           // 等效於 sh r2, [r3, 0]
sh r6, label          // 等效於 la r1, label 和 sh r6, [r1]兩條指令
sh r2, 0              // 絕對定位，等效於 la r1, 0 和 sh r2, [r1]兩條指令
```

sw——存儲字資料**指令格式： sw rD, [rA]****sw rD, label****sw rD, value**

該指令將通用暫存器 rD 的內容寫入到記憶體，它的使用方式和等效方式與前面的 lb 類似。

指令舉例：

```
sw r2, [r3]           // 等效於 sw r2, [r3, 0]
sw r6, label          // 等效於 la r1, label 和 sw r6, [r1] 兩條指令
sw r2, 0              // 絕對定位，等效於 la r1, 0 和 sw r2, [r1] 兩條指令
```

mul/mulu——乘法運算**指令格式： mul rD, rA, rB****mulu rD, rA, rB**

完成 rAxrB 的操作，並將結果的低字由 CEL 傳送至 rD 暫存器。操作等效於：

- mul rA, rB (mulu rA, rB)
- mfcel rD

指令舉例：

```
mul r4, r6, r7
mulu r4, r6, r7
```

div/divu——除法運算**指令格式： div rA, rB****divu rA, rB**

完成 rA÷rB 的操作，並將商由 CEL 傳送至 rD 暫存器。操作等效於：

- div rA, rB (divu rA, rB)
- mfceh rD

指令舉例：

```
div r4, r6
divu r4, r6
```

rem/remu——除法運算**指令格式： rem rD, rA, rB****remu rD, rA, rB**

這兩條指令完全等效於 div rD, rA, rB 和 divu rD, rA, rB 。

2.6 S+core7 處理器的 GNU 編譯器

2.6.1 S+core7 C 編譯器參數

在命令行下鍵入 Score-linux-elf-gcc 即可運行 S+core7 處理器的 C 編譯器，並可設置一些編譯參數。

語法：**Score-linux-elf-gcc [option|file]**

各參數內容含義如下：

-mSCORE5U	選擇 S+core5U 的編譯器
-mSCORE5	選擇 S+core5 的編譯器
-mSCORE7	選擇 S+core7 的編譯器
-meb/-mel	選擇大端或小端模式
-S	編譯成組合語言
-E	對指定的 C 程式僅用預處理器進行預處理
-o file	將編譯結果輸出到檔 file 中
-help	在螢幕上列出 GCC 定義的命令行的功能描述
-ansi	支援所有 ANSI 標準 C 語言程式，並禁止那些與標準 C 語言不相容的 GCC 程式編譯特性。注意，該參數並不拒絕非標準 C 語言程式。若要 GCC 拒絕非標準 C 語言程式並產生警告資訊，需將 -pedantic 參數放到-ansi 後面。
-pedantic	對標準 C (ANSI C 和 ISO C++ 語言程式產生警告)
-w	禁止所有警告資訊(warning)的輸出
-Wall	允許所有警告資訊(warning)的輸出。這些警告是針對用戶感覺可疑的問題，且這些問題很容易避免或修正，甚至與巨集相關的問題亦是如此
-Werror	將所有警告性質的問題設置成錯誤(error)性質
-Q	編譯時顯示被編譯的函數名稱列表，並同時顯示編譯過程中各階段的耗時資訊
-Dmacro	定義名稱為 macro 的巨集
-Dmacro=defn	定義名稱為 macro 的巨集的值為 defn。命令行中的所有 “-D” 參數都在處理 “-U” 參數前被處理
-Umacro	取消定義名稱為 macro 的巨集
-gstab+	為除錯器(debuggers)產生除錯(debug)資訊
-ldir	在標頭檔案裏的搜索目錄列表中增添名為 “dir” 的目錄

-O0	不進行最佳化處理
-O1	編譯器進行關於減少代碼容量及執行時間的最佳化處理
-O2	比 O1 更進一步最佳化，執行除了容量-速度折中最佳化之外的所有最佳化措施
-O3	比 O2 更進一步最佳化，除了 O2 所有最佳化措施外，還增加內聯 (in-lining) 功能
-Os	最佳化容量，即允許所有 O2 最佳化，又不明顯增加代碼容量，且做更進一步減少代碼容量的最佳化
-nostartfiles	不鏈接 crt*.o
-nostdlib	不鏈接標準庫(libc.a, libm.a……)

例 1：

在命令行視窗中鍵入：gcc -S test.c -o test.s

其中，test.c 是一個 C 語言代碼檔的名稱；該命令會產生一名為 test.s 的組合語言代碼檔。

例 2：

在命令行視窗中鍵入：gcc -S -gstabs+ test.c -o test.s

其中，test.c 是一個 C 語言代碼檔的名稱；該命令會產生一名為 test.s 的組合語言代碼檔，並且包含有 Debug 資訊。

例 3：

在命令行視窗中鍵入：gcc -S -O2 -gstabs+ test.c -o test.s

其中，test.c 是一個 C 語言代碼檔的名稱；該命令會產生一名為 test.s 的經過 O2 最佳化的組合語言代碼檔，並且包含有 Debug 資訊。

2.6.2 S+core7 C 編譯器的基本資料類型

S+core7 處理器的編譯器支援表 2-15所示的資料類型。

表 2-15 S+core7 C 編譯器的基本資料類型

資料類型	位數
char/unsigned char	8
short/unsigned short	16
int/unsigned int	32
long/unsigned long	32
long long/unsigned long long	64

資料類型	位數
float	32 位 IEEE 浮點格式
double	64 位 IEEE 浮點格式

2.6.3 S+core7 C 編譯器的函數調用約定

用戶可以在 C 語言中直接使用組合語言中的標號代表函數名來調用組合函數，而無需任何其他轉換。同樣，用戶可以在組合語言中直接使用形如 “bl <C 函數名>” 這樣的指令直接調用使用 C 語言編寫的函數。

S+core7 處理器有 32 個通用暫存器。這裏，將分類說明這些暫存器的用法，具體見表 2-16。

表 2-16 S+core7 通用暫存器在編譯器中的使用分配

暫存器	功能描述	函數調用時是否被保存
r0	堆疊指標	是
r1	暫存器，通常用於組譯器	否
r2	框（頁）指標	是
r3	鏈接暫存器。函數調用時，此暫存器保存返回位址	是
r4~r7	用於傳遞參數及返回值	否
r8~r11	調用者保存的暫存器	否
r12~r21	被調用者保存的暫存器，或選擇用於保存框指標	是
r22~r27	調用者保存的暫存器	否
r28	全局指標	是
r29	編譯器保留	是
r30~r31	僅為作業系統使用	是
hi	乘/除法運算特殊暫存器，保存乘法運算結果的高 32 位或除法結果的餘數	否
lo	乘/除法運算特殊暫存器，保存乘法運算結果的低 32 位或除法結果的商	否
sr0	特殊暫存器用於計數器操作	否
sr1	特殊暫存器用於裝載合併指令操作	否
sr2	特殊暫存器用於存儲合併指令操作	否

C 編譯器用暫存器 r4~r7 傳遞第 1 至第 4 個參數，且將返回值置於暫存器 r4 中。如果有多於 4 個參數的函數，則第 5 個參數會被置於記憶體[sp+16]中，而第 6 個參數會被置於記憶體 [sp+20]中，以此類推。用戶在處理組合語言與 C 相互調用期間的參出及返回值傳遞時需要注意。

用法舉例：

這裏列舉一個參數傳遞的例子，上面的程式為 C 語言根源程式，下面的程式為編譯器編譯出來的組合語言程式，由此可以看到，第 5 參數被置於[sp+16]記憶體裏，而第 6 參數被置於 [sp+20]記憶體裏。

```
void arg6(int a, int b, int c, int d, int e, int f);
int main(void)
{
    arg6(1,2,3,4,5,6);
}

void arg6(a, b, c, d, e, f)
    int a, b, c, d, e, f
{}
```

對應的組合代碼如下：

```
.text
    .align 2
    .globl main
main:
    sw r2, [r0, -4]+
    sw r3, [r0, -4]+
    sw r0, [r0, -4]+
    subi r0, 24
    mv r2, r0
    la r8, __main
    brl r8
    li r8, 5
    sw r8, [r0, 16]
    li r8, 6
    sw r8, [r0, 20]
    li r4, 1
    li r5, 2
    li r6, 3
    li r7, 4
    la r8, arg6
    brl r8
    addi r2, 24
    lw r0, [r2]+, 4
    lw r3, [r0]+, 4
    lw r2, [r0]+, 4
    br r3

    .align 2
    .globl arg6
arg6:
```

```
sw r2, [r0, -4]+  
sw r0, [r0, -4]+  
mv r2, r0  
lw r0, [r2]+, 4  
lw r2, [r0]+, 4  
br r3
```

2.7 S+core7 處理器的 GNU 組譯器

2.7.1 S+core7 C 組譯器參數

在命令行下鍵入 Score-linux-elf-as 即可運行 S+core7 處理器的 C 組譯器，並可設置一些組合參數。

語法：Score-linux-elf- as [option|file]

各參數內容含義如下：

-SCORE5U	指示組譯器按照 SCORE5U 產生組合代碼
-SCORE5	指示組譯器按照 SCORE5 產生組合代碼
-SCORE7	指示組譯器按照 SCORE7 產生組合代碼
-EB	選擇大端模式（預設參數）
-EL	選擇小端模式
-FIXDD	指示組譯器解決資料相依問題（當前不支援）
-NWARN	組譯器不產生關於資料相依問題的警告資訊
-USE_R1	組譯器不產生關於 R1 暫存器（組譯器用暫存器）的警告資訊
-G0Gp	開關控制值為 0，即不使用 gp（全局指標）定址方式
-gstabs	對每一組合指令行都產生 stabs 格式的除錯資訊
-I dir	在標頭檔案的搜索目錄列表中增添名為 dir 的目錄，該目錄中的檔案用偽指令 include 包含進原始檔案中
-o objfile	指定組譯器的輸出檔案名為 “objfile”
-O0	組譯器不對組合代碼進行最佳化

舉例：

在命令行視窗中鍵入：Score-linux-elf-as -SCORE7 -EL -USE_R1 -gstabs -I . -o test.o test.s

其中，test.s 檔是一個組合語言程式檔，該命令會產生一名為 test.o 的針對 SCORE7 的小端格式存儲的不包含關於 R1 暫存器的目標檔。

2.7.2 組合語言語法

組合語言根源程式包含一系列的組合語句，一行一句。每條語句都有如下的格式，且每一部分都可根據情況選擇是要還是不要。

Label: instruction #comment

其中，Label（標號）是允許編程者以標號的形式指定程式中某一位址，即 PC 的位置。一個標號可以用任意一有效符號後面加上 “:” 來表示。所謂有效符號是指由字母字元 A~Z、a~z、數位字元 0~9 以及字元 “_” 、 “-” 和 “\$” 組成的字串，並且字串不能以數位開頭。

comment(注釋)必須跟在符號 “#” 的後面，亦即任何跟在符號 “#” 後面的字元(除#include 和#define 外)都會被組譯器忽略掉。在此，也可以用 C 語言程式的注釋符 /*和*/或//來引用注釋句。

instruction (指令) 是用戶程式的實質內容，用戶既可以使用 S+core7 處理器的任何組合語言指令（用於組合成 S+core7 處理器要執行的操作碼），也可以使用偽指令，用於組譯器要執行的偽指令操作。偽指令的作用及其用法將在後面討論。

2.7.3 組譯器偽指令

所有的偽指令名前都必須用符號 “.” 開頭。這裏，以字母順序列出用戶組合語言程式中可能會經常用到的偽指令。

.align

語法：

```
.align alignment[, [fill][, max]]
```

用法：

```
.align 1           // 將位置計數器 PC 對齊到值為 2 的 1 次幕的位址 (半字對齊)
```

```
// 若 PC 的值已經是 2 的 1 次幕整數倍，則不需要任何改變
```

```
.align 2           // 將位置計數器 PC 對齊到值為 2 的 2 次幕的位址 (字對齊)
```

```
// 若 PC 的值已經是 2 的 2 次幕整數倍，則不需要任何改變
```

.ascii

語法：

```
.ascii strings
```

用法：

```
.ascii "JNZ"      // 插入字串 "JNZ"，即插入位元組資料 0x4a 0x4e 0x5a
```

.asciz

語法：

```
.asciz strings
```

用法：

```
.asciz "JNZ"      // 插入包含有字串結束旗標的字串 "JNZ"，
```

```
// 即插入位元組資料 0x4a 0x4e 0x5a 0x00
```

.byte

語法：

```
.byte expressions
```

用法：

```
.byte 64, 'A'           // 插入位元組資料 0x40 0x41
.byte 0x42               // 插入位元組資料 0x42 (0x 或 0X 均為 16 進制數的首碼)
.byte 0b1000011, 0104   // 插入位元組資料 0x43, 0x44
                           // 0b 或 0B 均為 2 進制數的首碼，0 為 8 進制數的首碼
```

.data

語法：

```
.data [subsection]
```

用法：

```
.data                  // 切換到.data 區段
```

.equ

語法：

```
.equ symbol, expression
```

用法：

```
.equ zzz, (5*8)+2      // 全局符號 zzz 的位址賦值為 42
                           // 本偽指令等效於操作：zzz = 42
```

.extern

語法：

```
.extern symbol
```

用法：

```
.extern label_zzz       // 指明符號 label_zzz 定義在其他原始檔案裏
```

.global

語法：

```
.global symbol
```

用法：

```
.global _start          // 指明_start 為所有模組使用的全局符號，包括鏈接器
```

.hword

語法：

```
.hword expression
```

用法：

```
.hword 0xaa55, 12345    // 插入半字資料，存儲順序為 0x55, 0xaa, 0x39, 0x30
```

.include

語法：

```
.include filename
```

用法：

```
.include "aaa"          // 引入其他檔案到當前檔案內
```

.int

語法：

```
.int expressions
```

用法：

```
.int aaa // 在未初始化的區段裏插入符號 aaa 的 4 個位元組資料
```

.org

語法：

```
.org new-lc [, fill]
```

用法：

```
.org 0x1234 // 將當前區段的位置計數器 PC 後移到位址 0x1234
```

.section

語法：

```
.section NAME [, "FLAGS" [, @TYPE[, @ENTSIZE]]]
```

FLAGS 常用的有以下幾種：

`a' 可分配區段

`w' 可寫區段

`x' 可執行區段

data section 具有的 FLAGS 通常是'wa'；

text section 具有的 FLAGS 通常是'ax'。

用法：

```
.section .text1, "wa" // 定義一個.text1 區段，且該區段為可寫的以及可分配的
```

.set

語法：

```
.set symbol, expression
```

用法：

```
.set nor1 // 從當前位置開始，若使用 r1，則組譯器輸出警告資訊
```

```
.set r1 // 從當前位置開始，若使用 r1，則組譯器不會輸出警告資訊
```

```
.set volatile // 組譯器不對用戶代碼進行最佳化
```

```
.set optimize // 組譯器會根據代碼大小對用戶代碼進行最佳化
```

```
.set nwarn // 當發生資料相依性問題時，組譯器不會輸出警告資訊
```

.space

語法：

```
.space size [, fill]
```

用法：

```
.space 100 // 從當前位置插入 100 個位元組資料 0x00
```

.text

語法：

```
.text
```

用法：

```
.text // 切換到.text 區段
```

.word

語法：

```
.word expression
```

用法：

```
.word 0xdeadbeaf // 插入字資料，存儲順序為 0xaf, 0xbe, 0xad, 0xde
```

2.7.4 區段及其重定位

簡單地說，“區段”是指一個連續的位址空間，所有處於該位址範圍內的資料均會被處理為具有相同性質的一批資料，譬如“唯讀”區段。

鏈接器 **ld** 將從各個目標檔(程式的一部分)讀出的內容鏈接到一起，形成一個可運行程式。組譯器 **as** 輸出的每一個目標檔，其程式的起始位址都會被假定為 0；而鏈接器 **ld** 會為其安排一個最終的運行位址，以保證所有這些目標檔中的程式位址不會重疊。這實際是一種簡化操作，但其足以解釋組合時“區段”的作用。

鏈接器 **ld** 將用戶程式中的位元組塊整體移到運行位址(**run-time address**)上，即位元組塊長度以及塊內位元組順序均不會改變。這個塊整體單元就被稱為“區段”，而為區段安排運行位址的過程就稱為“重定位”，即將目標檔位址調整到正確的運行位址上。

組譯器 **as** 產生的目標檔至少包含 3 個區段：**.text** 區段、**.data** 區段 和**.bss** 區段。其中，每一區段既可以有資料，也可以沒有資料。

除此之外，用戶還可以透過偽指令**.section** 來定義一個區段。假如用戶未使用偽指令來定義**.text** 區段或**.data** 區段，而這些區段依然存在，只是裏面沒有資料。通常，目標檔的**.text** 區段起始在位址 0 處，**.data** 區段則跟在其後，而**.bss** 區段則跟在**.data** 區段之後。

為了鏈接器 **ld** 能執行重定位，組譯器 **as** 需要把有關重定位的一些詳細資訊寫到目標檔中。這些資訊主要包括：

- 目標檔中該引用的起始位址在哪里？
- 該引用的長度為多少位元組？
- 該引用的定義處於另一個目標檔中的哪一區段？相對於那個區段的起始位址的偏移量是多少？
- 該引用是否為 PC 相對位址？

實際上，組譯器組合的每一個位址都可以表示為：位址 = 區段起始位址 + 區段內偏移量

由於多數用於計算位址的運算式都具有這種“區段-區段內相對位址”的屬性，因此，可以使用符號{secname N}來表示區段名為 secname 的區段內的偏移量 N 這樣一個意思。

除了上述**.text**、**.data** 和**.bss** 區段之外，還需要瞭解關於絕對區段(**absolute section**)的概念。當鏈接器 **ld** 在鏈接各目標檔時將部分程式的位址重定位後，處於絕對區段的這些位址就不會改變了。例如，**ld** 會為絕對位址{**absolute 0**}重定位到值為 0 的運行位址。儘管 **ld** 不會為兩個目標檔程式的資料區段分配重疊的位址，但其由絕對區段定義的程式位址卻很可能會重疊，如其中一部分程式的位址（絕對位址為 239）可能總會與另一部分程式要運行的位址（其絕對位址亦為 239）相同。

區段的概念可以延伸到未定義區段(**undefined section**)。任何位址所處的區段在組合時都是未

知的，透過符號{*undefined U*}標記出，其中 **U** 表示在鏈接時要被填充的意思。由於數字總是表示已經定義過的，產生未定義位址的唯一方法就是引用一個未定義符號。**common** 類型的變數就屬於未定義符號：該符號的值在組合階區段是未知的，因此它位於未定義區段。

透過類推可知，用 “**section**” 這個字來表示鏈接程式中相同性質的區段的集合。鏈接器 **ld** 在進行程式鏈接時慣常的做法是將所有目標檔中程式的**.text** 區段連續擺放，集合成一個“大”**.text** 區段。同樣，對**.data** 區段和**.bss** 區段也有類似的處理。

2.8 S+core7 處理器的 GNU 鏈接器

S+core7 處理器的鏈接器的命令行參數如下：

-larchive	添加檔案名為 <i>archive</i> 的歸檔檔案到要鏈接的檔列表中
-Lsearchdir	將名為 <i>searchdir</i> 的路徑添加到路徑列表中，該路徑列表為鏈接器 ld 搜索各程式館用
-M	將鏈接對映檔列印在鏈接器標準輸出中
-o output	指定檔案名為 <i>output</i> 的檔為鏈接器 ld 鏈接操作後的輸出檔
-EB	鏈接大端格式的目標（預設參數）
-EL	鏈接小端格式的目標
Tscriptfile	從名為 <i>scriptfile</i> 的檔裏讀出鏈接命令，這些鏈接命令會取代鏈接器 ld 預設的鏈接參數

舉例：

在命令行視窗中鍵入：`Score-linux-elf-ld -Texec.ld test.o -L . -o test.exec`

其中，**test.o** 檔是一個目標檔，該命令會根據 **exec.ld** 鏈接腳本檔的內容將 **test.o** 檔鏈接並產生一個名為 **test.exec** 的可執行檔。

3 SPCE3200 使用指南

3.1 SPCE3200 簡介

3.1.1 概述

SPCE3200 是一款高度集成的為多媒體應用設計的高性能 32 位晶片。它採用凌陽科技獨立智慧產權的 32 位 S+core7 處理器為內核，內置 MPEG4 硬體編碼譯碼模組，並外擴其他用於多媒體、機器人領域的功能模組。它專長於影像、視頻處理，可以輸出影像、聲音到電視機(NTSC 或 PAL 制式)以及 LCD 上顯示，具備強大的音頻、視頻、影像資料的處理能力。它可以輸出豐富的視頻畫面、聲音，並將這些資料存儲到 SD 卡或 NAND Flash 上。

SPCE3200 的工作電壓範圍為 3.0V~3.6V，CPU 頻率為 27~162MHz。此外，晶片提供 32768Hz 即時時鐘、低電壓檢測、低電壓重設、12 位類比/數位轉換器(ADC)、UART 介面、SPI 介面、SIO 介面、I2C 主設備介面以及其他 I/O 設備介面，例如 TFT LCD、彩色 STN LCD、CMOS 影像感測器(CMOS image sensor)、TVE 控制器、光筆、觸摸屏等。

S+core7 是一個單任務的、具有 7 級管線的高性能、高速的 32 位 RISC 處理器，採用了 Sunplus ISA (Instruction Set Architecture) 指令集，支援 32 位與 16 位混合指令模式以及並行條件執行，從而提高了代碼密度。在 SPCE3200 晶片中，S+core7 運行速度可達 162MHz，為實現 Soc 集成採用了 AMBA 汇流排，並設計了協同處理器以及 Custom Engine 介面從而可以提供靈活的擴展功能介面，並為高效地除錯及線上仿真(ICE)程式採用了 SJTAG 模組。該處理器支援 4KB 的 2 路組相連的 I/D Cache，以及 4KB 的 LIM/LDM(Local Instruction/Data Memory)。S+core7 系列 CPU 支援最多 63 個優先順序的中斷，可以快速回應中斷事件。此外，提供了一些高性能的指令來實現特定功能，如：SUNPLUS 已申請專利的非對齊資料裝載/存儲指令以及前/後增量定址指令，用於實現字串拷貝或記憶體資料傳輸；位元操作指令和分支指令採用迴圈控制計數器，用於有效地控制程式的走向；Sleep 指令為省電系統提供良好的支援。

3.1.2 特性

- 工作電壓：I/O 塊的 VDD 為 3.0V ~ 3.6V，CPU 內核的 VDD 為 1.62V ~ 1.98V
- CPU 工作頻率：27 ~ 162 MHz
- 支援擴展 SDR DRAM 和 DDR DRAM，最大容量可達 16M 位元組
- 具有 32 位/16 位的 SDRAM 資料匯流排
- 支援隔行掃描/逐行掃描的 NTSC/PAL 視頻輸出
- 影像解析度：VGA 模式(640 圖元 x 480 圖元)；CIF 模式(320 圖元 x 240 圖元)
- 支援 65536 色(RGB565 格式)
- 可編程選擇顏色模式：4/16/64/256/32768/65536
- 硬體方式的 MPEG-4/JPEG 編碼譯碼
- MPEG-4 框率(frame rate)：CIF 模式下高達 30 框/秒
- 4 通道 APB DMA 資料傳輸方式：從 APB 設備到 DRAM，或從 DRAM 到 APB 設備

- 硬體的 DRAM DMA 資料傳輸方式：由硬體執行的 DRAM 到 DRAM 的資料傳輸
- 雙通道 16 位高速 DAC，確保身歷聲輸出品質
- 內置 3 個可編程鎖相環(PLL)電路，為系統提供各路時鐘
- 為 NTSC 制/PAL 制系統提供 27 MHz 晶振
- 具備即時時鐘(RTC)
- 共 6 個 16 位計時器/計數器(具可編程自動重載功能)
- 提供 40 個中斷源：分別為計時器、時基、外部輸入以及鍵喚醒類型
- 支援鍵喚醒功能
- 9 通道 12 位 ADC
- USB 功能：支援 USB1.1 主機或 USB1.1 週邊設備
- UART 功能：具有通用非同步接收機和發送機
- 提供串列週邊設備介面(SPI)：具主/從模式
- 提供 Sunplus 同步串列輸入/輸出介面 (SIO)
- 內置 Watchdog 功能
- 提供 LCD 介面：具 TFT 方式/CSTN 方式
- 支援 CCIR-601/656 CMOS 影像感測器/TVE 控制介面
- 支援 SD 卡和 NAND 型 FLASH，用於海量資料存儲

3.2 SPCE3200 插腳信息

3.2.1 插腳分佈

SPCE3200 共有 256 個插腳，封裝形式為 PLCC256，它的插腳如圖 3-1 所示。

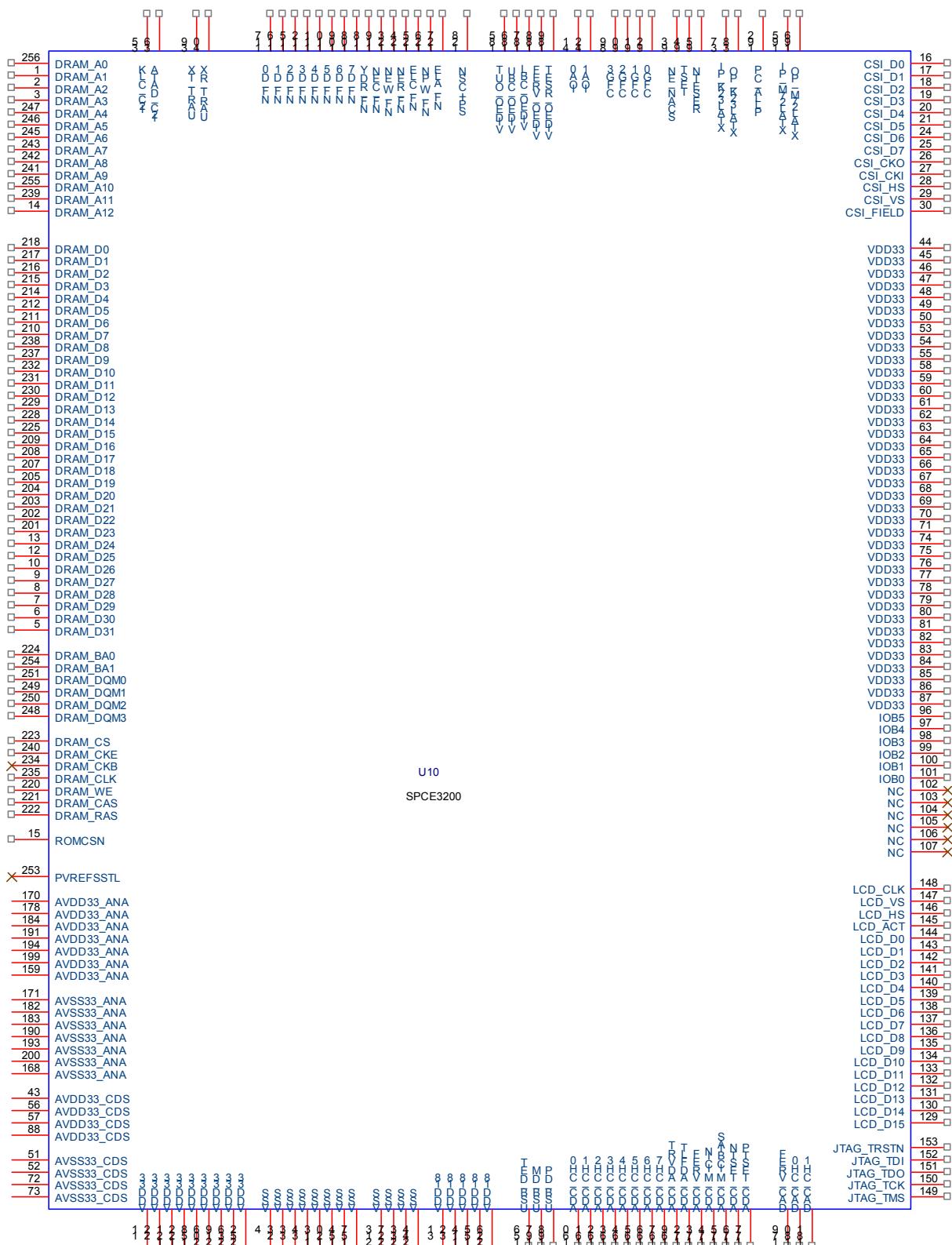


圖 3-1 SPCE3200 晶片插腳圖

3.2.2 插腳描述

SPCE3200 插腳描述如表 3-1 所示。

表 3-1 SPCE3200 插腳描述

插腳名稱	插腳號	插腳屬性	插腳功能
DRAM_A0	256	O	DRAM 位址匯流排
DRAM_A1	1		
DRAM_A2	2		
DRAM_A3	3		
DRAM_A4	247		
DRAM_A5	246		
DRAM_A6	245		
DRAM_A7	243		
DRAM_A8	242		
DRAM_A9	241		
DRAM_A10	255		
DRAM_A11	239		
DRAM_A12	14		
DRAM_D0	218	I/O	DRAM 資料匯流排
DRAM_D1	217		
DRAM_D2	216		
DRAM_D3	215		
DRAM_D4	214		
DRAM_D5	212		
DRAM_D6	211		
DRAM_D7	210		
DRAM_D8	238		
DRAM_D9	237		
DRAM_D10	232		

插腳名稱	插腳號	插腳屬性	插腳功能
DRAM_D11	231		
DRAM_D12	230		
DRAM_D13	229		
DRAM_D14	228		
DRAM_D15	225		
DRAM_D16	209		
DRAM_D17	208		
DRAM_D18	207		
DRAM_D19	205		
DRAM_D20	204		
DRAM_D21	203		
DRAM_D22	202		
DRAM_D23	201		
DRAM_D24	13		
DRAM_D25	12		
DRAM_D26	10		
DRAM_D27	9		
DRAM_D28	8		
DRAM_D29	7		
DRAM_D30	6		
DRAM_D31	5		
ROMCSN	15	O	Flash 片選。用於外接 Nor-type Flash 或 SRAM
CSI_D0	16	I	CMOS 感測器介面資料匯流排
CSI_D1	17		
CSI_D2	18		
CSI_D3	19		
CSI_D4	20		
CSI_D5	21		

插腳名稱	插腳號	插腳屬性	插腳功能
CSI_D6	24		
CSI_D7	25		
CSI_CKO	26	O	CMOS 感測器介面時鐘輸出
CSI_CK1	27	I	CMOS 感測器介面時鐘輸入
CSI_HS	28	I/O	CMOS 感測器介面水平同步信號
CSI_VS	29	I/O	CMOS 感測器介面垂直同步信號
CSI_FIELD	30	I/O	CMOS 感測器介面場掃描信號
I2C_CLK	35	O	I2C 時鐘
I2C_DATA	36	I/O	I2C 資料
XTAL32K_PI	37	I	32768Hz 時鐘輸入
XTAL32K_PO	38	O	32768Hz 時鐘輸出
UART_TX	39	O	UART 發送
UART_RX	40	I	UART 接收
IOA0	41	I/O	A 組通用輸入輸出埠
IOA1	42	I/O	A 組通用輸入輸出埠
VDD33	44~87	I	輸入輸出埠電壓輸入
CFG0	92	I	系統配置埠
CFG1	91		
CFG2	90		
CFG3	89		
SCAN_EN	93	I	掃描使能
TEST	94	I	測試用，接高
RESETN	95	I	重設，低電平有效
IOB5	96	I/O	B 組通用輸入輸出埠
IOB4	97	I/O	B 組通用輸入輸出埠
IOB3	98	I/O	B 組通用輸入輸出埠
IOB2	99	I/O	B 組通用輸入輸出埠
IOB1	100	I/O	B 組通用輸入輸出埠

插腳名稱	插腳號	插腳屬性	插腳功能
IOB0	101	I/O	B 組通用輸入輸出埠
NC	102~107	O	空腳
NF_D0	117		
NF_D1	116		
NF_D2	115		
NF_D3	112		
NF_D4	111		
NF_D5	110		
NF_D6	109		
NF_D7	108		
NF_RDY	118	I	NANDFlash Ready 信號
NF_CEN	119	O	NANDFlash 晶片使能
NF_WEN	123	O	NANDFlash 寫使能
NF_REN	124	O	NANDFlash 讀使能
NF_CLE	125	O	NANDFlash 命令鎖存使能
NF_WPN	126	O	NANDFlash 防寫
NF_ALE	127	O	NANDFlash 位址鎖存
SPI_CSN	128	I/O	SPI 晶片使能
LCD_D0	144		
LCD_D1	143		
LCD_D2	142		
LCD_D3	141		
LCD_D4	140		
LCD_D5	139		
LCD_D6	138		
LCD_D7	137		
LCD_D8	136		
LCD_D9	135		

插腳名稱	插腳號	插腳屬性	插腳功能
LCD_D10	134		
LCD_D11	133		
LCD_D12	132		
LCD_D13	131		
LCD_D14	130		
LCD_D15	129		
LCD_ACT	145	O	LCD 介面使能信號
LCD_HS	146	O	LCD 介面水平同步信號
LCD_VS	147	O	LCD 介面垂直同步信號
LCD_CLK	148	O	LCD 介面時鐘信號
JTAG_TMS	149	I	JTAG 模式選擇信號
JTAG_TCK	150	I	JTAG 時鐘信號
JTAG_TDO	151	O	JTAG 資料輸出信號
JTAG_TDI	152	I	JTAG 資料輸入信號
JTAG_TRSTN	153	I	JTAG 重設，低電平有效
USBDET	156	I	USB 偵測信號
ADC_CH0	160	I/O	ADC 類比輸入通道
ADC_CH1	161		
ADC_CH2	162		
ADC_CH3	163		
ADC_CH4	164		
ADC_CH5	165		
ADC_CH6	166		
ADC_CH7	167		
ADC_ADVRT	169	I	ADC 參考電壓
ADC_ADFLT	172	O	反鋸齒波輸出腳
ADC_VREF	173	O	AFE 參考電壓
ADC_MICIN	174	I	MIC 輸入埠

插腳名稱	插腳號	插腳屬性	插腳功能
ADC_MICBIAS	175	O	緩存適配電壓為駐極體 MIC 提供偏壓。電壓為 AVDD33 的 3/4
ADC_TESTN	176	I/O	測試模式的負輸入或輸出
ADC_TESTP	177	I/O	測試模式的正輸入或輸出
DAC_VREF	179	I	DAC 參考電壓
DAC_CH0	180	O	DAC 通道 0
DAC_CH1	181	O	DAC 通道 1
VIDEO_OUT	185	O	DAC 電流輸出
VIDEO_CBU	186	O	接 0.1uF 到 AVDD
VIDEO_CBL	187	O	接 0.1uF 到 AVDD
VIDEO_VREF	188	O	能帶輸出
VIDEO_RSET	189	O	接 1.4K 到 GND
PLLA_CP	192	I	濾波回路
XTAL27M_PI	195	I	27MHz 晶振輸入腳
XTAL27M_PO	196	O	27MHz 晶振輸出腳
USB_DM	197	I/O	USB 資料負
USB_DP	198	I/O	USB 資料正
DRAM_WE	220	O	DRAM 寫使能
DRAM_CAS	221	O	DRAM 柱位址開關
DRAM_RAS	222	O	DRAM 行位址開關
DRAM_CS	223	O	DRAM 片選
DRAM_BA0	224	O	DRAM Bank 位址
DRAM_BA1	254		
DRAM_CKB	234	O	DRAM 時鐘
DRAM_CLK	235	O	DRAM 時鐘
DRAM_CKE	240	O	DRAM 時鐘使能
DRAM_DQM0	251	O	DRAM 資料遮罩，就是將 DRAM 資料輸入或資料輸出遮罩起來不輸入或輸出
DRAM_DQM1	249		

插腳名稱	插腳號	插腳屬性	插腳功能
DRAM_DQM2	250		
DRAM_DQM3	248		
PVREFSSTL	253	O	SSTL2 參考電壓
AVDD33_ANA	159,170, 178,184, 191,194, 199	I	類比電壓電源
AVSS33_ANA	168,171, 182,183, 190,193, 200	I	類比電壓地
VDD33	11,22, 121,122, 158,206, 219,236, 252	I	I/O 電壓電源
VSS	4,23, 33,34, 113,120, 154,157, 213,227, 233,244	I	I/O 電壓地
VDD18	31,32, 114,155, 226	I	邏輯電壓電源

3.3 結構概述

SPCE3200 的結構如圖 3-2 所示。它包含一個支援使用 SJTAG 進行線上仿真的 S+core7 處理器 CPU、符合 AMBA 標準的與片內高效能模組相連的 AHB 汇流排、連接片內其他低速週邊設備的 APB 汇流排以及 AHB 到 APB 汇流排的橋控制器和 DMA 控制器。

在圖 3-2 中，LDM (Local Data Memory) 指片內的 8KB 的 RAM。

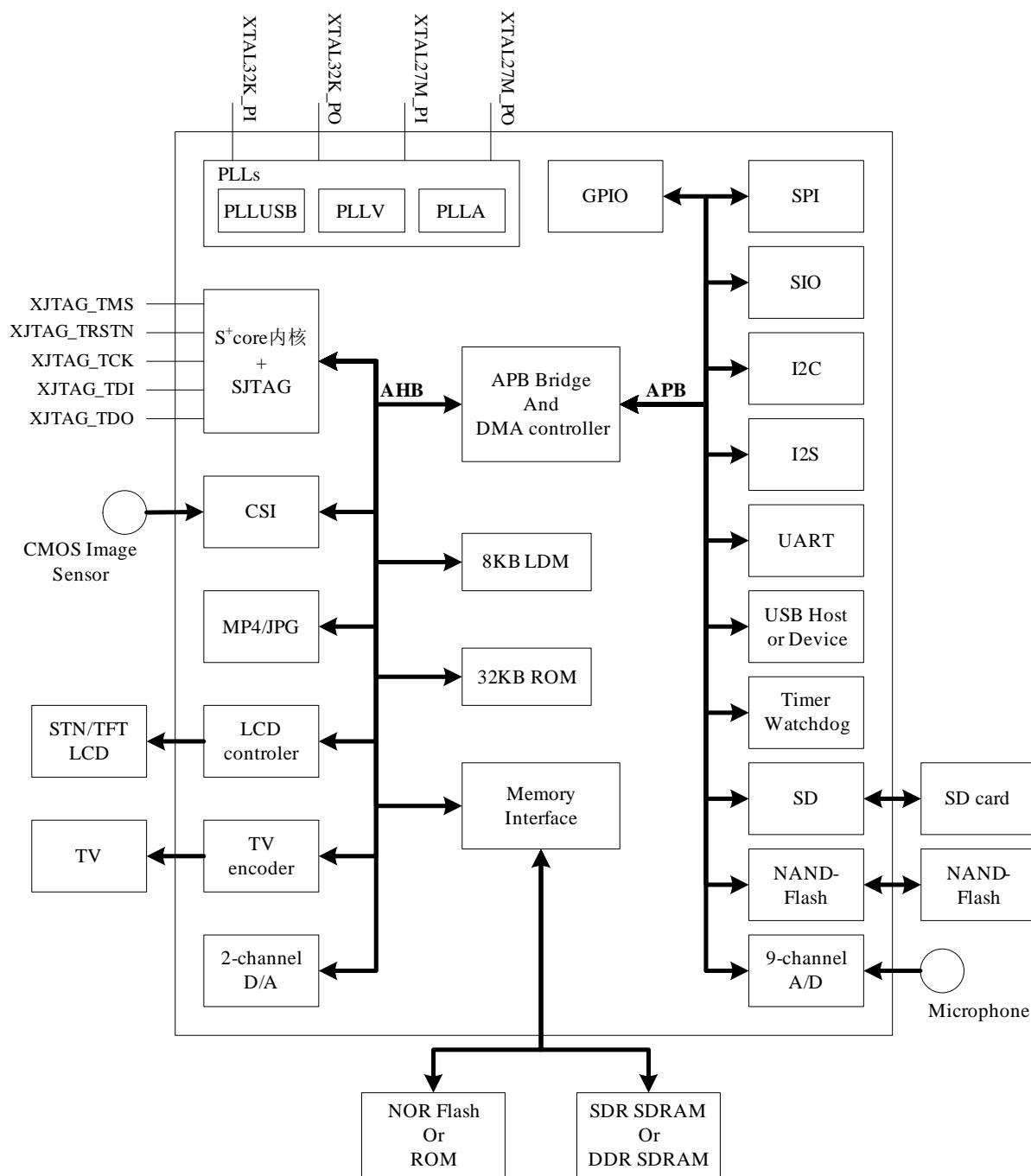


圖 3-2 SPCE3200 結構框圖

掛載在高性能 AHB 汇流排上的所有週邊設備模組包括：

- 記憶體介面控制單元（MIU）
- MPEG4/JPEG 編碼譯碼模組
- CMOS 影像感測器介面（CSI）模組
- LCD 控制器模組
- TV 編碼模組
- 2 通道 16 位高速 D/A 轉換器

掛載在 APB 週邊設備匯流排上的所有模組包括：

- GPIO 控制器模組
- 計時器、即時時鐘和時間基準信號發生模組
- 看門狗模組
- SPI 串列匯流排控制器
- SIO 串列匯流排控制器
- I2C 串列匯流排控制器
- I2S 主/從控制器
- UART 控制器
- USB 主/從控制器
- SD 卡控制器
- Nand 型 Flash 控制器
- 9 通道 12 位 A/D 轉換器

3.4 記憶體對映

SPCE3200 採用基於哈佛結構的 S+core7 處理器內核，處理器內部匯流排實行資料和指令獨立定址，CPU 透過分別的匯流排實現對資料和程式的存取。需要注意的是，雖然處理器內核採用獨立的資料匯流排和指令匯流排，但處理器採用固定記憶體對映（Fixed-MMU）模式對應於週邊設備模組和外部實體記憶體，從而變為週邊設備的統一定址。在晶片上電之後，提供給用戶經過如圖 3-3 所示的記憶體對映之後的虛擬位址空間。

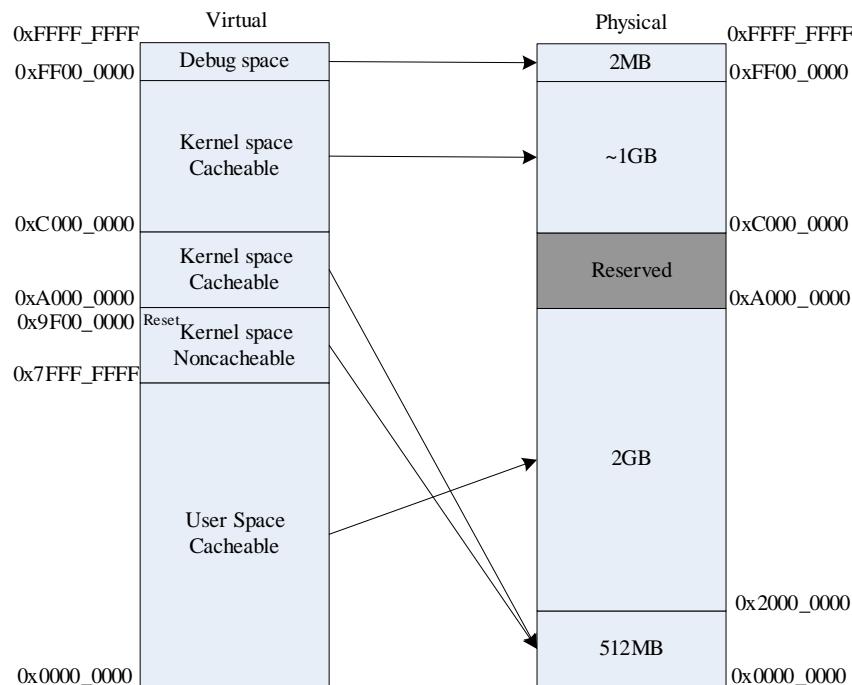


圖 3-3 SPCE3200 的記憶體對映示意圖

SPCE3200 為每個片內週邊設備分配了 32K 的空間，位於虛擬位址 0x88000000 開始的位址空間。這些 32K 的空間包含了各個週邊設備的控制暫存器和資料暫存器等，用戶可以直接存取這些位址來控制各個週邊設備的工作。每個週邊設備分配的位址空間如表 3-2 所示。

表 3-2 SPCE3200 週邊設備位址空間分配表

位址空間	週邊設備名稱	位址空間	週邊設備名稱
0x88000000~0x8800FFFF	CSI	0x88161000~0x88161FFF	TIMER1
0x88030000~0x8803FFFF	TV 編碼器	0x88162000~0x88162FFF	TIMER2
0x88040000~0x8804FFFF	LCD 驅動器	0x88163000~0x88163FFF	TIMER3
0x88070000~0x8807FFFF	MIU	0x88164000~0x88164FFF	TIMER4
0x88080000~0x8808FFFF	APBDMA	0x88165000~0x88165FFF	TIMER5
0x88090000~0x8809FFFF	緩衝區控制器	0x88166000~0x88166FFF	RTC
0x880A0000~0x880AFFFF	IRQ 控制器	0x88170000~0x8817FFFF	看門狗
0x880C0000~0x880CFFFF	LDMDMA	0x88180000~0x8818FFFF	SD 卡控制器
0x880D0000~0x880DFFFF	BLNDMA	0x88190000~0x8819FFFF	Nand 型 Flash 控制器
0x880F0000~0x880FFFFF	AHB 譯碼器	0x881A0000~0x881AFFFF	A/D 轉換器
0x88100000~0x8810FFFF	GPIO 控制器	0x881B0000~0x881BFFFF	USB Device 控制器

位址空間	週邊設備名稱	位址空間	週邊設備名稱
0x88110000~0x8811FFFF	SPI 控制器	0x881C0000~0x881CFFFF	USB Host 控制器
0x88120000~0x8812FFFF	SIO 控制器	0x88200000~0x8820FFFF	軟體配置控制器
0x88130000~0x8813FFFF	I2C 控制器	0x88210000~0x8821FFFF	時鐘控制器
0x88140000~0x8814FFFF	I2S 控制器	0x88220000~0x8822FFFF	MPEG4 編碼譯碼控制器
0x88150000~0x8815FFFF	UART	0x88230000~0x8823FFFF	MIU2
0x88160000~0x88160FFF	TIMER0	0x88240000~0x8824FFFF	ECC 檢查控制器

在 SPCE3200 內部還具有 8KB 的 RAM(Local Data Memory)和 32KB 的 Flash ROM(Local Instruction Memory)。在系統重設後，內部 RAM 被定位在 0xA0000000 開始的 8KB 位址空間內，內部 Flash ROM 被定位在 0x9F000000 開始的 32KB 位址空間內，程式將從內部的 Flash ROM 開始執行。同時，用戶可以使用 cache 指令重新為內部 RAM 和內部 ROM 指定對映位址空間，詳細請參考指令系統一章中關於 cache 指令的描述。

3.5 鎮相環 PLL 與時鐘發生器 CKG

3.5.1 鎮相環 PLL

SPCE3200 依靠外部的一顆 27MHz 的晶振和一顆 32768Hz 的即時時鐘晶振維持整個系統的運轉。其中，27MHz 晶振可以直接為 APB 汇流排及其上的 APB 週邊設備等模組提供工作頻率，32768Hz 的即時時鐘晶振可以為 Watch-Dog、Timer、RTC 和 TimeBase 等模組直接提供工作頻率。

同時，SPCE3200 還內嵌三個 PLL 電路，分別稱之為 PLLU、PLLV 和 PLLA。三個 PLL 電路可以將 27MHz 的晶振頻率倍頻到不同的頻率。如圖 3-4所示。

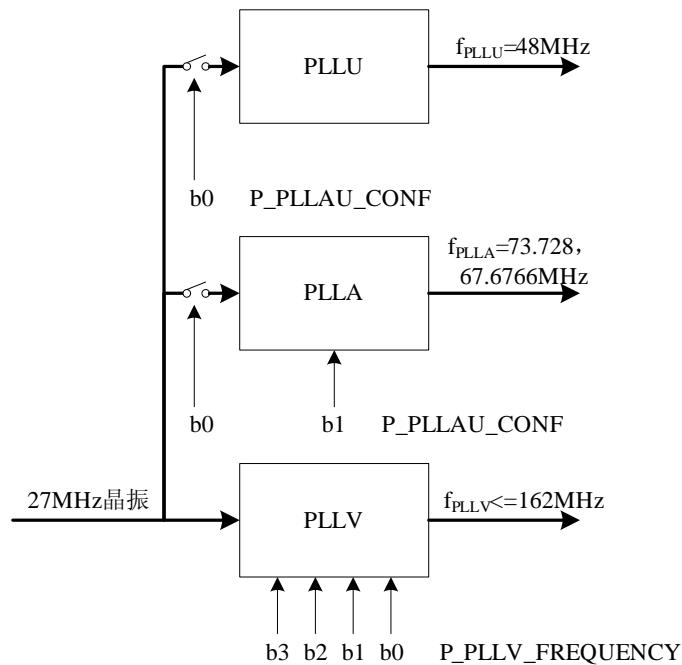


圖 3-4 PLL 電路框圖

經過 PLLV 倍頻之後可以輸出最大至 162MHz 的頻率主要提供給 CPU 及 AHB 汇流排和週邊設備使用。

PLLA 電路將 27MHz 倍頻至 67.6766MHz 或 73.728MHz。

PLLU 電路為 USB 等模組提供 48MHz 的時鐘頻率。

在系統上電後，僅 PLLV 電路處於工作狀態並為片內包括 CPU、MIU (Memory Interface Unit—記憶體介面單元)、APBDMA 和軟體配置模組等在內的主要模組提供工作時鐘。當用戶需要使用某個週邊設備時，必須為該週邊設備接通工作時鐘。SPCE3200 內部的 PLL 控制器控制著所有週邊設備的工作時鐘的通斷，表 3-3 描述了每個片內週邊設備的工作時鐘通斷控制暫存器。

表 3-3 SPCE3200 週邊設備工作時鐘通斷控制暫存器

暫存器位址	週邊設備名稱	暫存器位址	週邊設備名稱
0x88210018	CSI	0x88210070	TIMER1
0x88210020	TV 編碼器	0x88210074	TIMER2
0x88210034	LCD 驅動器	0x88210078	TIMER3
0x882100D4	EPP	0x8821007C	TIMER4
0x88210058	APBDMA	0x88210080	TIMER5
0x882100C8	緩衝區控制器	0x88210088	RTC
0x882100E0	TimeBase	0x88210084	看門狗

暫存器位址	週邊設備名稱	暫存器位址	週邊設備名稱
0x882100CC	LDMDMA	0x882100A4	SD 卡控制器
0x88210050	BLNDMA	0x882100A8	Nand 型 Flash 控制器
0x88210098	SPI 控制器	0x882100AC	A/D 轉換器
0x882100A0	SIO 控制器	0x88210064	USB
0x88210094	I2C 控制器	0x882100FC	軟體配置控制器
0x8821008C	I2S 控制器	0x882100EC	MPEG4 編碼譯碼控制器
0x8821005C	UART	0x882100BC	PLLA&PLLU
0x8821006C	TIMER0	0x88210114	32768 晶振

3.5.2 時鐘發生器 CKG

經過三個 PLL 電路倍頻之後的時鐘信號被送至時鐘發生器單元，並在這裏將時鐘分頻，分配給 SPCE3200 各個片上模組使用。其中，AHB 汇流排及其週邊設備由 PLLV 或 PLLA 經過 CKG 電路分頻提供工作頻率，均工作在相同的頻率下；APB 汇流排及其週邊設備，除 A/D 轉換器和看門狗模組外，均工作於 27MHz 頻率下，計時器模組也可以使用 32768Hz 即時時鐘晶振提供工作頻率；A/D 轉換器可以選擇由 PLLU 或 PLLA 經過 CKG 電路分頻提供工作頻率；看門狗則由 32768Hz 時鐘晶振提供工作頻率。

CKG 時鐘發生器單元為可以選擇工作頻率的某些模組提供了相應的控制暫存器，表 3-4 描述了這些模組的工作頻率選擇暫存器。

表 3-4 SPCE3200 週邊設備工作時鐘選擇暫存器

暫存器位址	週邊設備名稱	暫存器位址	週邊設備名稱
0x88210004	CPU	0x882100B0	ADC
0x8821000C	MIU	0x882100B8	PLLV
0x8821001C	CSI	0x882100BC	PLLAU
0x8821002C	JPEG	0x882100DC	Sleep
0x88210038	LCD	0x882100E4	Timer
0x88210090	I2S	0x882100F0	MPEG4

3.5.3 暫存器描述

與各個模組相關的時鐘通斷控制暫存器和頻率選擇暫存器請參考各個功能部件章節，這裏僅介紹 CPU、AHB 汇流排、PLLV、PLLA、PLLU 以及 32768Hz 即時時鐘晶振等相關的時鐘

配置暫存器。

■ CPU 時鐘選擇暫存器：P_CLK_CPU_SEL(0x88210004)

用戶可以透過此暫存器選擇 CPU 的工作頻率。注意，在系統上電後該暫存器不可直接修改，如果需要更改 CPU 時鐘，請參考3.5.4 節的操作方法。

表 3-5 P_CLK_CPU_SEL(0x88210004)

位	b3-b0
讀/寫	R/W
預設值	0
名稱	CPU_CLOCK

CPU_CLOCK	b3-b0	CPU 時鐘選擇位：
		0000 : PLLV 時鐘頻率輸出
		0001 : PLLV 時鐘頻率輸出 2 分頻
		0010 : PLLV 時鐘頻率輸出 3 分頻
		0011 : PLLV 時鐘頻率輸出 4 分頻
		0100 : PLLV 時鐘頻率輸出 6 分頻
		0101 : PLLV 時鐘頻率輸出 8 分頻
		0110 : PLLA 時鐘頻率輸出
		0111 : PLLA 時鐘頻率輸出 2 分頻
		1000 : PLLA 時鐘頻率輸出 3 分頻
		1001 : PLLA 時鐘頻率輸出 4 分頻
		1010 : PLLA 時鐘頻率輸出 6 分頻
		1011 : PLLA 時鐘頻率輸出 8 分頻

■ AHB 汇流排時鐘配置暫存器：P_CLK_AHB_CONF(0x88210008)

用戶可以透過此暫存器打開或關閉 AHB 汇流排的工作時鐘。注意，在系統上電後該暫存器不可直接修改，如果需要停止 AHB 汇流排時鐘，請參考3.5.4 節的操作方法。

表 3-6 P_CLK_AHB_CONF(0x88210008)

位	b0
讀/寫	W
預設值	1
名稱	CPUAHB_CLK_Stop

CPUAHB_CLK_Stop b0 暫時關閉 AHB 汇流排工作時鐘，以便更改其工作頻率

0：關閉 AHB 時鐘

1：不關閉 AHB 時鐘

■ AHB 汇流排時鐘選擇暫存器：P_CLK_AHB_SEL(0x8821000C)

用戶可以透過此暫存器選擇 AHB 汇流排的工作頻率。注意，在系統上電後該暫存器不可直接修改，如果需要更改 AHB 汇流排時鐘，請參考3.5.4 節的操作方法。

表 3-7 P_CLK_AHB_SEL(0x8821000C)

位	b1~b0
讀/寫	W
預設值	0
名稱	CPUAHB_CLK

CPUAHB_CLK b1~b0 AHB 時鐘選擇位：

00：CPU 工作頻率

01：CPU 工作頻率 2 分頻

10：CPU 工作頻率 3 分頻

11：CPU 工作頻率 4 分頻

■ PLLV 時鐘配置暫存器：P_CLK_PLLV_CONF(0x882100B4)

用戶可以透過此暫存器打開或關閉 PLLV 的輸出。注意，在系統上電後該暫存器不可直接修改，如果需要停止 PLLV 時鐘，請參考3.5.4 節的操作方法。

表 3-8 P_CLK_PLLV_CONF(0x882100B4)

位	b0
讀/寫	W
預設值	1
名稱	PLLV_EN

PLLV_EN	b0	PLLV 頻率輸出使能位：
		0：關閉 PLLV 頻率輸出
		1：打開 PLLV 頻率輸出

■ PLLV 時鐘選擇暫存器：P_CLK_PLLV_SEL(0x882100B8)

用戶可以透過此暫存器選擇 PLLV 的輸出頻率。注意，在系統上電後該暫存器不可直接修改，如果需要更改 PLLV 的時鐘輸出，請參考3.5.4 節的操作方法。

表 3-9 P_CLK_PLLV_SEL(0x882100B8)

位	b3-b0
讀/寫	W
預設值	0
名稱	FREQUENCY

FREQUENCY	b3~b0	PLLV 輸出頻率選擇位：
		0000：保留
		0001：保留
		0010：保留
		0011：81MHz
		0100：87.75MHz
		0101：97.5MHz
		0110：101.25MHz
		0111：108MHz
		1000：114.75MHz

1001 : 121.5MHz

1010 : 128.25MHz

1011 : 135MHz

1100 : 141.75MHz

1101 : 148.5MHz

1110 : 155.25MHz

1111 : 162MHz

■ PLLAU 時鐘配置暫存器：P_CLK_PLLAU_CONF(0x882100BC)

用戶可以透過此暫存器打開或關閉 PLLA 和 PLLU 的輸出，並可以選擇 PLLA 的輸出頻率。

表 3-10 P_CLK_PLLAU_CONF(0x882100BC)

位	b2	b1	b0
讀/寫	W	W	W
預設值	0	0	0
名稱	USBPLL_EN	PLLA_CLK	PLLA_EN

USBPLL_EN b2 PLLU 頻率輸出使能位：

0 : 關閉 PLLU 頻率輸出

1 : 打開 PLLU 頻率輸出

PLLA_CLK b1 PLLA 輸出頻率選擇位：

0 : 73.728MHz

1 : 67.6766MHz

PLLA_EN b0 PLLA 頻率輸出使能位：

0 : 關閉 PLLA 頻率輸出

1 : 打開 PLLA 頻率輸出

■ 32K 即時時鐘配置暫存器：P_CLK_32K_CONF(0x88210114)

用戶可以透過此暫存器使能或禁止 32768Hz 即時時鐘晶振。

表 3-11 P_CLK_32K_CONF(0x88210114)

位	b0
讀/寫	W
預設值	0
名稱	Crystal_En

Crystal_En	b0	32768Hz 即時時鐘晶振使能位：
0：禁止		
1：使能		

3.5.4 系統時鐘調整

■ 停止 CPU/MIU 時鐘以及 PLLV 時鐘

在停止 MIU 時鐘之前，必須首先使 DRAM 進入自刷新模式，否則，在停止 MIU 時鐘之後存儲在 DRAM 中的資料將丟失。

如果希望 CPU 進入睡眠模式，必須檢查中斷請求狀態暫存器 P_INT_REQ_STATUS1 和 P_INT_REQ_STATUS2，清除所有可以喚醒 CPU 的中斷，否則，CPU 進入睡眠模式後將立即被終止。

SPCE3200 內部 ROM 提供了一些程式入口來使 CPU、MIU 或 PLLV 時鐘停止以便降低功耗。這些程式的入口位址如表 3-12 所示。

表 3-12 停止 CPU/MIU/PLLV 時鐘 API 函數列表

程式入口	功能描述
0x9F000010	關閉 CPU 時鐘
0x9F000014	關閉 CPU 和 MIU 時鐘
0x9F000018	關閉 CPU、MIU 和 PLLV 時鐘

用戶需要使用 S+core7 組合語言編寫下面的代碼以便可以調用這些函數更改系統時鐘。

```

@jumpto_extrom_discpu_clk
.global jumpto_extrom_discpu_clk
jumpto_extrom_discpu_clk:
    li r8, 0x9f000010
    br r8

@jumpto_extrom_discpu_miу_clk
.global jumpto_extrom_discpu_miу_clk

```

```
jumpto_extrom_discpu_miу_clk:  
    li r8, 0x9f000014  
    br r8  
  
@jumpto_extrom_discpu_miу_clk_pll  
.global jumpto_extrom_discpu_clk_pll  
jumpto_extrom_discpu_clk_pll:  
    li r8, 0x9f000018  
    br r8
```

■ 更改 CPU/MIU 時鐘以及 PLLV 時鐘

當希望更改 MIU 的時鐘時，程式不能在 DRAM 中運行，因為 SPCE3200 將透過 MIU 模組來存取 DRAM 中的內容，一旦開始更改 MIU 的時鐘，那麼 MIU 中 DRAM 的參數設置將發生錯誤，此時 MIU 將不能正確存取 DRAM 的內容，從而導致程式運行出錯。

用戶必須使用 SPCE3200 內部 ROM 中提供的程式來更改 CPU/MIU 的時鐘，程式入口位址為 0x9F000024。

注意：

- 更改 CPU 時鐘的程式將對 PLLV 和 PLLA 進行操作，所以，在執行更改 CPU 時鐘的程式之前，必須檢查晶片內使用 PLLV 和 PLLA 的模組沒有處於工作狀態。在更改完畢 CPU 時鐘後，PLLA 將處於打開狀態，如果不需要使用它，用戶可以將其關閉。
 - CPU 的時鐘必須是 MIU 時鐘的 1~4 倍。
-

SPCE3200 內部 ROM 提供的更改 CPU 時鐘的程式需要用戶傳遞四個參數以便指定更改後的時鐘頻率。第一個參數含有七個子項，含義如表 3-13 所示。

表 3-13 更改時鐘頻率程式的第一個參數

子項名稱	有效位	影響暫存器	功能描述
	b31~b12	無	保留
PLLA_Enable	b11	P_CLK_PLLAU_CONF	PLLA 使能位： 0：使能 1：禁止
PLLV_Freq_Chg	b10	P_CLK_PLLV_SEL	PLLV 是否改變： 0：不改變 1：改變
AHBSLV_Sel	b9~b8	P_CLK_AHB_SEL	MIU 時鐘選擇位： 00：CPU 工作頻率

子項名稱	有效位	影響暫存器	功能描述
			01 : CPU 工作頻率 2 分頻 10 : CPU 工作頻率 3 分頻 11 : CPU 工作頻率 4 分頻
PLL_DIV_Setting	b7~b4	P_CLK_CPU_SEL	CPU 時鐘選擇位： 0000 : PLLV 時鐘頻率輸出 0001 : PLLV 時鐘頻率輸出 2 分頻 0010 : PLLV 時鐘頻率輸出 3 分頻 0011 : PLLV 時鐘頻率輸出 4 分頻 0100 : PLLV 時鐘頻率輸出 6 分頻 0101 : PLLV 時鐘頻率輸出 8 分頻 0110 : PLLA 時鐘頻率輸出 0111 : PLLA 時鐘頻率輸出 2 分頻 1000 : PLLA 時鐘頻率輸出 3 分頻 1001 : PLLA 時鐘頻率輸出 4 分頻 1010 : PLLA 時鐘頻率輸出 6 分頻 1011 : PLLA 時鐘頻率輸出 8 分頻
PLLV_Freq_Setting	b3~b0	P_CLK_PLLV_SEL	PLLV 輸出頻率選擇位： 0000 : 保留 0001 : 保留 0010 : 保留 0011 : 81MHz 0100 : 87.75MHz 0101 : 97.5MHz 0110 : 101.25MHz 0111 : 108MHz 1000 : 114.75MHz 1001 : 121.5MHz 1010 : 128.25MHz 1011 : 135MHz

子項名稱	有效位	影響暫存器	功能描述
			1100 : 141.75MHz
			1101 : 148.5MHz
			1110 : 155.25MHz
			1111 : 162MHz

第二個參數用來設置 SDRAM 的參數，含義如表 3-14所示。

表 3-14 更改時鐘頻率程式的第二個參數

子項名稱	有效位	影響暫存器	功能描述
	b31~b17	無	保留
MIU_Run108_Sel	b16	P_MIU_SDRAM_SETUP1	MIU 時鐘是否運行在 108MHz： 0：不運行 1：運行
MIU_Timing_Setting	b15~b0	P_MIU_SDRAM_SETUP1	MIU 時鐘頻率選擇位： 0x48C0 : 27MHz 0x4920 : 40MHz 0x4983 : 54MHz 0x4A03 : 67.5MHz 0x4A03 : 81MHz 0x4B04 : 108MHz

第三個和第四個參數用來指定 SPCE3200 變更時鐘頻率後等待 PLLV 變為穩定的時鐘週期。

用戶可以編寫下面所示的 C 代碼和組合代碼以便調用該程式來改變時鐘頻率。

C 代碼：

```
extern void jumpto_extrom_change_cpuclk(unsigned int r4, unsigned int r5,
unsigned int r6, unsigned int r7);
void PLL_change_func(
    int PLLV_Freq_Setting,
    int PLL_DIV_Setting,
    int AHBSLV_SEL,
    int PLLV_Freq_Chg,
    int PLLA_Enable,
    int MIU_Timing_Setting,
```

```
    int MIU_Run108_Sel
) {
    unsigned int r4, r5, r6, r7;
    r4 = PLLV_Freq_Setting + (PLLV_DIV_Setting << 4) + (AHBSLV_SEL << 8) +
        (PLLV_Freq_Chg << 10) + (PLLA_Enable << 11);
    r5 = (MIU_Run108_Sel << 16) + MIU_Timing_Setting;

    r6 = 0xff;
    r7 = 0xff;
    jump_to_extrom_change_cpuclk(r4, r5, r6, r7);
}
```

組合代碼：

```
@jump_to_extrom_change_cpuclk
.global jump_to_extrom_change_cpuclk
jump_to_extrom_change_cpuclk:
    ldis r8, 0x9f00
    ori r8, 0x0024
    br r8
```

3.6 中斷控制器

3.6.1 概述

為了最大效率的使用 CPU 資源，SPCE3200 也引入了中斷機制。SPCE3200 透過中斷控制器向 CPU 發出中斷請求，同時，當有多個中斷發生時，中斷控制器進行中斷優先順序的仲裁。

SPCE3200 的中斷控制器為 40 個中斷源提供了遮罩控制能力，並將這些中斷源進行處理、向處理器發出中斷請求信號。中斷控制器的作用是對來自內部週邊設備和外部中斷請求管腳上發生的多個中斷請求進行仲裁處理(優先權排隊)後，然後依處理結果向 CPU 發出 IRQ 中斷請求。

通常，CPU 只接收來自週邊設備的 IRQ 中斷事件，但這種中斷接收並沒有優先順序機制。因此，中斷處理程式必須借助中斷控制器裏的硬體來處理這種情況。譬如，假設所有中斷源都已設置成 IRQ 中斷，其中有 10 個中斷源向 CPU 同時發出中斷請求，那麼，就可以透過軟體讀出中斷控制器裏的中斷狀態暫存器來獲知都發生了哪些中斷請求，從而確定中斷服務的優先順序。

可想而知，這種中斷處理需要較長的中斷回應時間才能跳越到相應的服務程式的位置上，為此，S+core7 處理器提供了另一種中斷處理機制，稱為向量中斷模式，它是 RISC 型微處理器所共有的特點。當有多個中斷源發出中斷請求時，由硬體優先權邏輯來確定應當先服務於哪一個中斷請求；同時該硬體優先權邏輯將向量表中向量的基底位址與偏移值相配，計算出相應服務程式位置的位址。S+core7 處理器也相應地提供了一些指令，配合硬體來獲取向量表的入口，並實現相應中斷服務程式位置的跳越，從而大大縮短了中斷回應時間。

由此，中斷控制器的功能作用總結起來有三個：

- (1) 40 個中斷源的遮罩控制。

(2) 中斷優先順序進的裁決。

(3) 向 CPU 發出中斷請求。

3.6.2 特性

SPCE3200 中斷控制器的特性如下：

- 為易於實現 Soc 集成，匯流排符合 AMBA 規格(Rev 2.0)
- 支援 40 個中斷源
- IRQ 中斷為高電平有效
- 支援向量中斷模式
- 每一中斷源都可編程設置其優先順序
- 每一個中斷源的中斷都可獨立被遮罩或者使能

3.6.3 中斷源

SPCE3200 有 40 個中斷源，佔用 S+core7 內核中第 24~63 個硬體中斷向量，如表 3-15。

表 3-15 SPCE3200 的中斷源列表

序號	組名	中斷向量號	中斷服務函數	中斷源	助記名	中斷向量位址 = 中斷基底位址 +
1	SLV0	63	IRQ63()	DAC 中斷	DAC_IRQ	63×0x04(0x10)
2		62	IRQ62()	保留	RESERVED	62×0x04(0x10)
3		61	IRQ61()	保留	RESERVED	61×0x04(0x10)
4		60	IRQ60()	保留	RESERVED	60×0x04(0x10)
5		59	IRQ59()	MIC 溢出中斷	MIC_OV	59×0x04(0x10)
6		58	IRQ58()	ADC 中斷	ADC_IRQ	58×0x04(0x10)
7		57	IRQ57()	時基中斷	TMB_IRQ	57×0x04(0x10)
8		56	IRQ56()	計時器中斷	TIMER_IRQ	56×0x04(0x10)
9	SLV1	55	IRQ55()	TV 垂直空白開始中斷	TV_VBS	55×0x04(0x10)
10		54	IRQ54()	LCD 垂直空白開始中斷	LCD_VBS	54×0x04(0x10)
11		53	IRQ53()	保留	RESERVED	53×0x04(0x10)
12		52	IRQ52()	TV 光槍中斷	TV_LIGHT	52×0x04(0x10)
13		51	IRQ51()	CSI 框結束中斷	CSI_FRE	51×0x04(0x10)
14		50	IRQ50()	CSI 座標擊中中斷	CSI_COH	50×0x04(0x10)
15		49	IRQ49()	CSI 運動框結束中斷	CSI_MFRE	49×0x04(0x10)

序號	組名	中斷向量號	中斷服務函數	中斷源	助記名	中斷向量位址 = 中斷基底位址 +
16	SLV2	48	IRQ48()	CSI 摷取完成中斷	CSI_DONE	48×0x04(0x10)
17		47	IRQ47()	TV 座標擊中中斷	TV_COH	47×0x04(0x10)
18		46	IRQ46()	保留	RESERVED	46×0x04(0x10)
19		45	IRQ45()	USB 主從中斷	USB_IRQ	45×0x04(0x10)
20		44	IRQ44()	SIO 中斷	SIO_IRQ	44×0x04(0x10)
21		43	IRQ43()	SPI 中斷	SPI_IRQ	43×0x04(0x10)
22		42	IRQ42()	UART 中斷	UART_IRQ	42×0x04(0x10)
23		41	IRQ41()	NAND 中斷	NAND_IRQ	41×0x04(0x10)
24		40	IRQ40()	SD 卡通訊中斷	SD_IRQ	40×0x04(0x10)
25	SLV3	39	IRQ39()	I2C 主模式中斷	I2C_IRQ	39×0x04(0x10)
26		38	IRQ38()	I2S 從模式中斷	I2S_IRQ	38×0x04(0x10)
27		37	IRQ37()	APB DMA 通道 0 中斷	APBD_CH0	37×0x04(0x10)
28		36	IRQ36()	APB DMA 通道 1 中斷	APBD_CH1	36×0x04(0x10)
29		35	IRQ35()	LDM DMA 中斷	LDM_DMA	35×0x04(0x10)
30		34	IRQ34()	BLN DMA 中斷	BLN_DMA	34×0x04(0x10)
31		33	IRQ33()	APB DMA 通道 2 中斷	APBD_CH2	33×0x04(0x10)
32		32	IRQ32()	APB DMA 通道 3 中斷	APBD_CH3	32×0x04(0x10)
33	SLV4	31	IRQ31()	即時時鐘中斷	RTC_IRQ	31×0x04(0x10)
34		30	IRQ30()	MPEG4 中斷	MPEG4_IRQ	30×0x04(0x10)
35		29	IRQ29()	ECC 中斷	ECC_IRQ	29×0x04(0x10)
36		28	IRQ28()	GPIO 外部中斷	GPIO_IRQ	28×0x04(0x10)
37		27	IRQ27()	TV 垂直空白結束中斷	TV_VBE	27×0x04(0x10)
38		26	IRQ26()	保留	RESERVED	26×0x04(0x10)
39		25	IRQ25()	保留	RESERVED	25×0x04(0x10)
40		24	IRQ24()	保留	RESERVED	24×0x04(0x10)

說明：

- 為了優先順序設置方便，人為把 40 個中斷源分成了 5 組，每組 8 個中斷源，詳見中斷優先順序設定暫存器。
- 中斷向量號是 CPU 處理各中斷源中斷服務程式的標識，根據中斷向量號可以計算出中斷服務程式的入口位址，詳見中斷機制章節。
- 助記名是為了方便各暫存器介紹時約定的符號，詳見暫存器章節。
- 計算中斷向量位址兩種辦法，即偏移量可以是 0x04，也可以是 0x10，用戶可以透過設置 S+core7 內核暫存器 cr3 的 bit0 設置其偏移量，見中斷機制章節。

3.6.4 結構框圖

SPCE3200 中斷控制器的結構如圖 3-5：在使用中斷處理一功能模組的任務時，需要先透過模組的中斷控制器使能該模組中斷；中斷控制器需要先使能（允許）來自該模組中斷源的中斷，當有中斷請求時，中斷控制器把中斷狀態暫存器的相應位置位，並把該請求送給中斷優先順序裁決器，中斷優先順序裁決器根據設定好的優先順序決定向 CPU 發送相應模組的中斷請求；時鐘控制器控制中斷控制器模組的重設。

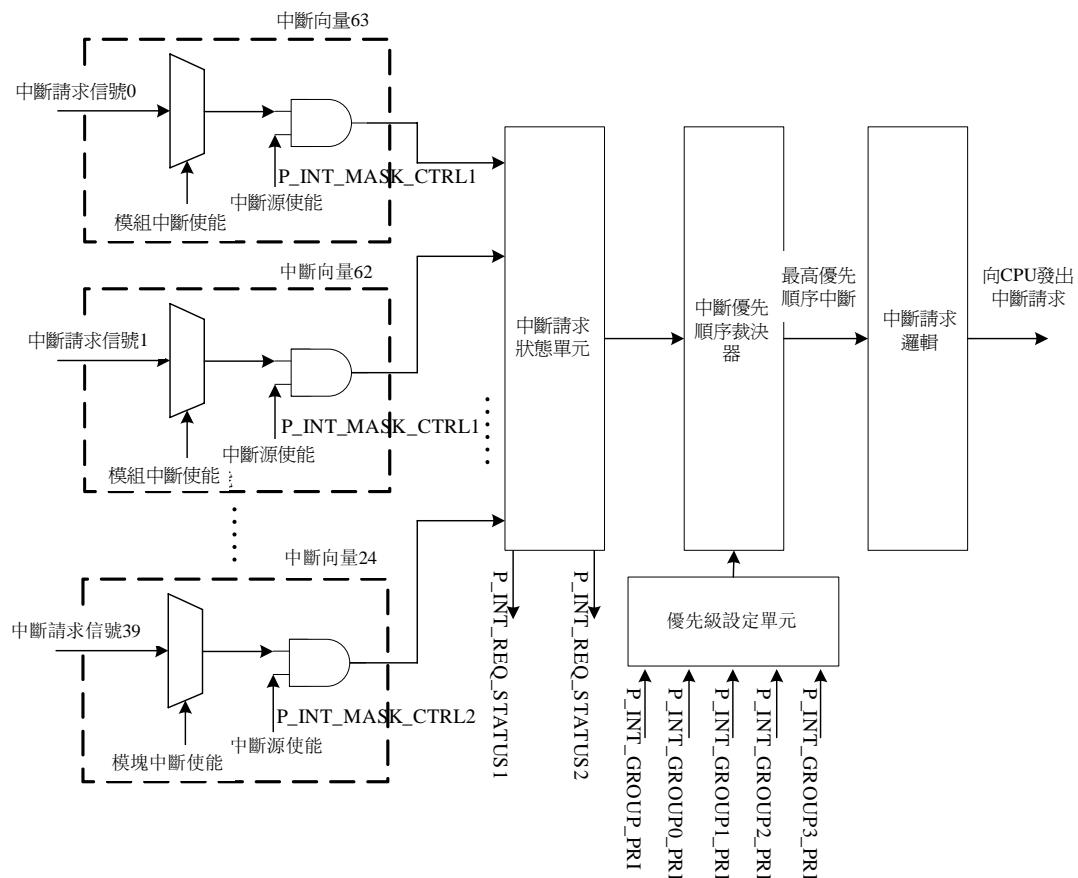


圖 3-5 中斷控制器結構框圖

3.6.5 暫存器描述

根據圖 3-5 中斷控制器結構框圖可以看出，SPCE3200 中斷控制器的硬體暫存器有時鐘控制暫存器、中斷源中斷使能（允許）暫存器、中斷請求狀態暫存器和中斷優先順序設定暫存器四大類共 10 個暫存器，如表 3-16。

表 3-16 中斷控制器相關暫存器列表

暫存器名稱	助記符	位址
中斷時鐘配置暫存器	P_INT_CLK_CONF	0x882100D0
中斷控制暫存器 1	P_INT_MASK_CTRL1	0x880A0020
中斷控制暫存器 2	P_INT_MASK_CTRL2	0x880A0024
中斷請求狀態暫存器 1	P_INT_REQ_STATUS1	0x880A0000
中斷請求狀態暫存器 2	P_INT_REQ_STATUS2	0x880A0004
中斷優先順序暫存器	P_INT_GROUP_PRI	0x880A0008
第 0 組中斷優先順序暫存器	P_INT_GROUP0_PRI	0x880A0010
第 1 組中斷優先順序暫存器	P_INT_GROUP1_PRI	0x880A0014
第 2 組中斷優先順序暫存器	P_INT_GROUP2_PRI	0x880A0018
第 3 組中斷優先順序暫存器	P_INT_GROUP3_PRI	0x880A001C

■ 中斷時鐘配置暫存器：P_INT_CLK_CONF(0x882100D0)

透過中斷時鐘配置暫存器可以重設/不重設中斷控制器模組，正常使用時不需要重設中斷控制器。

表 3-17 P_INT_CLK_CONF(0x882100D0)

位	b31~b1	b0
讀/寫	-	W
預設值	-	1
名稱	-	IRQ_RST

IRQ_RST	b0	中斷控制器模組重設位： 0：中斷控制器模組重設 1：中斷控制器模組不重設
---------	----	--

■ 中斷控制暫存器 1：P_INT_MASK_CTRL1 (0x880A0020)

SPCE3200 有 40 個硬體中斷源（中斷向量 63~24），由兩個 32 位暫存器 P_INT_MASK_CTRL1 和 P_INT_MASK_CTRL2 來控制這些中斷源中斷的使能或者遮罩。

表 3-18 P_INT_MASK_CTRL1 (0x880A0020)

位	b31	b30	b29	b28	b27	b26	b25	b24
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	1	1	1	1	1
名稱	APBD_CH3	APBD_CH2	BLN_DMA	LDM_DM_A	APBD_CH1	APBD_CH0	I2S_IRQ	I2C_IRQ
位	b23	b22	b21	b20	b19	b18	b17	b16
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	1	1	1	1	1
名稱	SD_IRQ	NAND IRQ	UART IRQ	SPI_IRQ	SIO_IRQ	USB_IRQ	RESERVED	TV_COH
位	b15	b14	b13	b12	b11	b10	b9	b8
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	1	1	1	1	1
名稱	CSI_DONE	CSI_MFRE	CSI_COH	CSI_FRE	TV_LIGHT	RESERVED	LCD_VBS	TV_VBS
位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	1	1	1	1	1
名稱	TIMER IRQ	TMB_IRQ	ADC_IRQ	MIC_OV	RESERVED	RESERVED	RESERVED	DAC_IRQ

APBD_CH3	b31	APB DMA 通道 3 中斷遮罩位： 0：使能 APB DMA 通道 3 中斷 1：遮罩 APB DMA 通道 3 中斷
APBD_CH2	b30	APB DMA 通道 2 中斷遮罩位： 0：使能 APB DMA 通道 2 中斷 1：遮罩 APB DMA 通道 2 中斷
.....
ADC_IRQ	b5	ADC 中斷遮罩位： 0：使能 ADC 中斷 1：遮罩 ADC 中斷
MIC_OV	b4	MIC 溢出中斷遮罩位： 0：使能 MIC 溢出中斷 1：遮罩 MIC 溢出中斷
DAC_IRQ	b0	DAC 中斷遮罩位： 0：使能 DAC 中斷 1：遮罩 DAC 中斷

說明：

- 表中暫存器所有 32 位的功能一樣，寫 “1” 遮罩該位對應的中斷，寫 “0” 使能該位對應的中斷。
- 各位對應的中斷名稱請參考表 3-15。

■ 中斷控制暫存器 2：P_INT_MASK_CTRL2(0x880A0024)

透過中斷控制暫存器 2 使能或者遮罩來自中斷向量 31~24 中斷源請求的中斷。

表 3-19 P_INT_MASK_CTRL2(0x880A0024)

位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	1	1	1	1	1
名稱	RESERV ED	RESERV ED	RESERV ED	TV_VBE	GPIO_IR Q	ECC_IRQ	MP4_IRQ	RTC_IRQ

與 P_INT_MASK_CTRL1 類似，P_INT_MASK_CTRL2 的低 8 位中，各位的功能相同：寫 “1” 遷罩該位對應的中斷，寫 “0” 使能該位對應的中斷。各位對應的中斷名稱請參考表 3-15。

SPCE3200 由兩個 32 位硬體暫存器 P_INT_REQ_STATUS1 和 P_INT_REQ_STATUS2 來反映 40 個中斷源的請求狀態。

■ 中斷請求狀態暫存器 1：P_INT_REQ_STATUS1(0x880A0000)

透過中斷請求狀態暫存器 1 來獲得中斷向量 63~32 中斷源的中斷請求狀態。如果中斷控制器收到中斷源的中斷請求，則該中斷源對應的位被置位。

表 3-20 P_INT_REQ_STATUS1(0x880A0000)

位	b31	b30	b29	b28	b27	b26	b25	b24
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	APBD_C H3	APBD_CH 2	BLN_DMA	LDM_DM A	APBD_CH 1	APBD_CH 0	I2S_IRQ	I2C_IRQ
位	b23	b22	b21	b20	b19	b18	b17	b16
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	SD_IRQ	NAND_IR Q	UART_IR Q	SPI_IRQ	SIO_IRQ	USB_IRQ	RESERV ED	TV_COH
位	b15	b14	b13	b12	b11	b10	b9	b8
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	CSI_DON E	CSI_MFR E	CSI_COH	CSI_FRE	TV_LIGH T	RESERV ED	LCD_VBS	TV_VBS
位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	TIMER_IR Q	TMB_IRQ	ADC_IRQ	MIC_OV	RESERV ED	RESERV ED	RESERV ED	DAC_IRQ

APBD_CH3	b31	APB DMA 通道 3 中斷請求狀態位： 0 : APB DMA 通道 3 没有向 CPU 發出中斷請求 1 : APB DMA 通道 3 向 CPU 發出中斷請求
APBD_CH2	b30	APB DMA 通道 2 中斷請求狀態位： 0 : APB DMA 通道 2 没有向 CPU 發出中斷請求 1 : APB DMA 通道 2 向 CPU 發出中斷請求
.....
ADC_IRQ	b5	ADC 中斷請求狀態位： 0 : ADC 没有向 CPU 發出中斷請求 1 : ADC 向 CPU 發出中斷請求
MIC_OV	b4	MIC 溢出中斷請求狀態位： 0 : MIC 溢出沒有向 CPU 發出中斷請求 1 : MIC 溢出向 CPU 發出中斷請求
DAC_IRQ	b0	DAC 中斷請求狀態位： 0 : DAC 没有向 CPU 發出中斷請求 1 : DAC 向 CPU 發出中斷請求

說明：

- 表中暫存器所有 32 位的功能一樣，讀為“1”表示產生了中斷請求，讀為“0”表示沒有產生中斷請求。
- 各位對應的中斷名稱請參考表 3-15。

■ 中斷請求狀態暫存器 2 : P_INT_REQ_STATUS2(0x880A0004)

透過中斷請求狀態暫存器 2 來獲得中斷向量 31~24 中斷源的中斷請求狀態。

表 3-21 P_INT_REQ_STATUS2(0x880A0004)

位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	RESERV ED	RESERV ED	RESERV ED	TV_VBE	GPIO IRQ Q	ECC_IRQ	MP4_IRQ	RTC_IRQ

與 P_INT_REQ_STATUS1 類似，P_INT_REQ_STATUS2 的低 8 位中，各位的功能相同：讀爲“1”表示產生了中斷請求，讀爲“0”表示沒有產生中斷請求。各位對應的中斷名稱請參考表 3-15。

■ 中斷優先順序暫存器：P_INT_GROUP_PRI(0x880A0008)

SPCE3200 的中斷向量 63~32 的中斷優先順序可透過暫存器設置，中斷向量 31~24 的中斷優先順序由中斷控制器硬體控制。由於 32 個中斷源的中斷優先順序不能透過一個暫存器來設定，所以先把這 32 個中斷源分為 4 組，分組情況參考表 3-15，每組有 8 個中斷源，先分別設置各組的優先順序，然後設置各組內部各中斷的優先順序。

表 3-22 P_INT_GROUP_PRI(0x880A0008)

位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	W	W	W	W	W	W	W	W
預設值	1	1	1	0	0	1	0	0
名稱	P_SLV3		P_SLV2		P_SLV1		P_SLV0	

P_SLV3	b7~b6	第 3 組中斷的優先順序設置位： 00：第 1 優先順序 01：第 2 優先順序 10：第 3 優先順序 11：第 4 優先順序
P_SLV2	b5~b4	第 2 組中斷的優先順序設置位： 00：第 1 優先順序 01：第 2 優先順序 10：第 3 優先順序 11：第 4 優先順序
P_SLV1	b3~b2	第 1 組中斷的優先順序設置位： 00：第 1 優先順序 01：第 2 優先順序 10：第 3 優先順序 11：第 4 優先順序

P_SLV0

b1~b0

第 0 組中斷的優先順序設置位：

00：第 1 優先順序

01：第 2 優先順序

10：第 3 優先順序

11：第 4 優先順序

■ 第 0 組中斷優先順序暫存器：P_INT_GROUP0_PRI(0x880A0010)****

透過 P_I_PSLV0 暫存器控制第 0 組中斷中各中斷的優先順序。

表 3-23 P_INT_GROUP0_PRI(0x880A0010)

位	b23~b21	b20~b18	b17~b15	b14~b12	b11~b9	b8~b6	b5~b3	b2~b0
讀/寫	W	W	W	W	W	W	W	W
預設值	111b	110b	101b	100b	011b	010b	001b	000b
名稱	TIMER IRQ	TMB_IRQ	ADC_IRQ	MIC_OV	RESERVED	RESERVED	RESERVED	DAC_IRQ

從 b0~b23 共 24 位控制 8 個中斷的優先順序，每三位控制一個中斷的優先順序，三位從高到低的組合（b23b22b21、b20b19b18、b17b16b15、b14b13b12、b11b10b9、b8b7b6、b5b4b3、b2b1b0）功能如下：

000：第 1 優先順序。

001：第 2 優先順序。

010：第 3 優先順序。

011：第 4 優先順序。

100：第 5 優先順序。

101：第 6 優先順序。

110：第 7 優先順序。

111：第 8 優先順序。

預設最低 3 位（b2~b0）對應的中斷優先順序最高，最高 3 位（b23~b21）對應的中斷優先順序最低。

表中對應的中斷名稱可參考表 3-15。

暫存器 P_INT_GROUP1_PRI、P_INT_GROUP2_PRI、P_INT_GROUP3_PRI 和 P_INT_GROUP0_PRI 類似，只是暫存器中每三位對應的中斷源各不相同而已。

■ 第 1 組中斷優先順序暫存器：**P_INT_GROUP1_PRI(0x880A0014)**

透過 P_INT_GROUP1_PRI 暫存器控制第 1 組各中斷源的優先順序。

表 3-24 P_INT_GROUP1_PRI(0x880A0014)

位	b23~b21	b20~b18	b17~b15	b14~b12	b11~b9	b8~b6	b5~b3	b2~b0
讀/寫	W	W	W	W	W	W	W	W
預設值	111b	110b	101b	100b	011b	010b	001b	000b
名稱	CSI_DONE	CSI_MFRE	CSI_COH	CSI_FRE	TV_LIGH_T	RESERVED	LCD_VBS	TV_VBS

■ 第 2 組中斷優先順序暫存器：**P_INT_GROUP2_PRI(0x880A0018)**

透過 P_INT_GROUP2_PRI 暫存器控制第 2 組各中斷源的優先順序。

表 3-25 P_INT_GROUP2_PRI(0x880A0018)

位	b23~b21	b20~b18	b17~b15	b14~b12	b11~b9	b8~b6	b5~b3	b2~b0
讀/寫	W	W	W	W	W	W	W	W
預設值	111b	110b	101b	100b	011b	010b	001b	000b
名稱	SD_IRQ	NAND IRQ	UART IRQ	SPI IRQ	SIO IRQ	USB IRQ	RESERVED	TV_COH

■ 第 3 組中斷優先順序暫存器：**P_INT_GROUP3_PRI(0x880A001C)**

透過 P_INT_GROUP3_PRI 暫存器控制第 3 組各中斷源的優先順序。

表 3-26 P_INT_GROUP3_PRI(0x880A001C)

位	b23~b21	b20~b18	b17~b15	b14~b12	b11~b9	b8~b6	b5~b3	b2~b0
讀/寫	W	W	W	W	W	W	W	W
預設值	111b	110b	101b	100b	011b	010b	001b	000b
名稱	APBD_CH3	APBD_CH2	BLN_DMA	LDM_DM_A	APBD_CH1	APBD_CH0	I2S_IRQ	I2C_IRQ

3.6.6 中斷機制

SPC3200 的中斷機制是採用中斷向量的模式，每個中斷向量對應一個中斷源（中斷源列表參考表 3-15），但是每個中斷源不一定對應一個中斷，比如 UART 中斷源分別有發送中斷和接收中斷。中斷的處理過程如圖 3-6。中斷的處理過程涉及的硬體單元包括：週邊設備單元（ADC、Timer 等）、中斷控制器和 S+core7 內核。

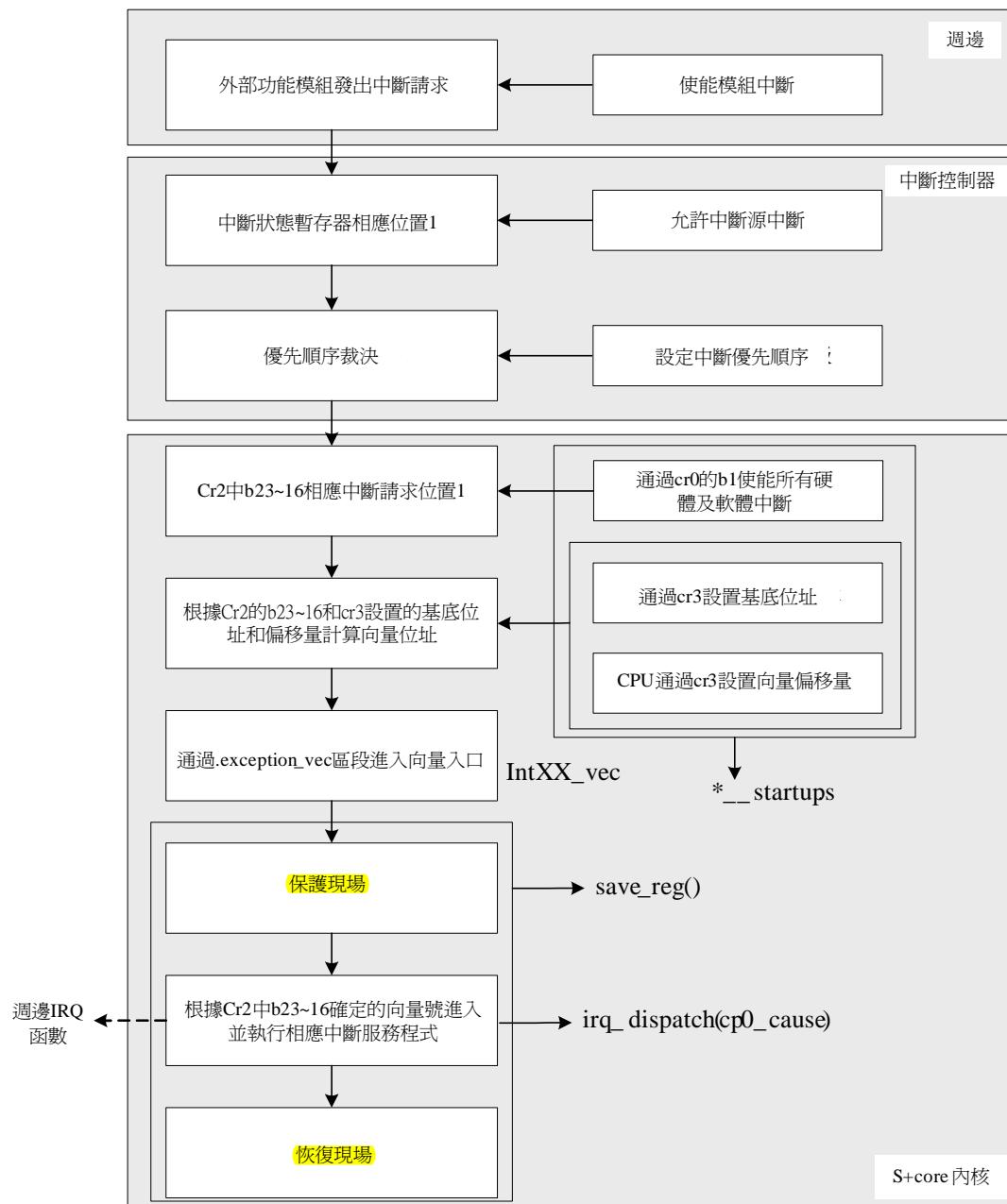


圖 3-6 中斷的處理過程圖

圖 3-6 中週邊設備單元部分主要負責使能硬體模組中斷和發出中斷請求，必須透過相應暫存器使能模組中斷，該模組才會發出中斷請求。比如先要透過 0x88160000 位址使能 Timer0 中斷，Timer 才會在設定定時時間時發出中斷請求。

中斷控制器主要負責中斷源的遮罩和中斷優先順序的裁決，在週邊設備發出中斷請求前，中斷控制器必須要先關閉該模組對應中斷源的中斷遮罩，這樣中斷控制器才會做優先順序裁決並發出中斷請求；如果在週邊設備發出中斷請求前，該中斷源的中斷是被遮罩的，則在週邊設備發出中斷請求時中斷控制器不做任何處理；中斷控制器有專門的暫存器可設置中斷優先順序，中斷控制器按照設置好的優先順序進行優先順序裁決。

S+core7 內核必須允許來自模組中斷源的中斷，才會回應中斷；透過程式狀態暫存器 PSR (cr0) 允許 63 個（含 SPCE3200 的 40 個中斷源）硬體中斷。S+core7 在透過異常原因暫存器 ECR (cr2) 得到中斷請求相應位置 1 時，透過 cr2 的 b23~b18 得到向量號，並透過向量號、基底位址和偏移量得到其向量入口位址；透過 cr3 設置中斷的基底位址，其中最低位設置偏移量，當最低位為 0 時，偏移量為 0x04，當最低位為 1 時，偏移量為 0x10；比如透過異常向量暫存器 EVCPVec (cr3) 設置中斷的基底位址為 0xa0000000，偏移量為 0x04，中斷區段的起始位址為 0xa00001fc，例如 Timer 請求中斷的向量號為 56，那麼其中斷向量位址為 0xa0000200+4×56。在 S+core7 找到中斷向量入口位址後即可轉到相應的中斷服務程式執行，控制週邊設備進行相應操作，如圖 3-6虛線箭頭部分。

圖 3-6 中關於中斷的各個過程分別在 *_startup.s 、 Sys_isr.s 、 Sys_IRQ.c 和 User_IRQ.c 四個檔中進行設置或者定義，這些檔在建立工程時可根據用戶要求自動生成。

_startup.s (“” 為工程名) 定義了 S+core7 內核的所有硬體中斷的使能、中斷基底位址的設置和偏移量的設置。中斷相關程式區段如下：

```
//=====
// 中斷使能
//=====
li r4, 0x1
mtcr r4, cr0          // 使能所有 63 個硬體中斷和 2 個軟體中斷
li r4, 0xa0000000
mtcr r4, cr3          // 指定異常基底位址為 0xa0000000,
                      // 並指定向量位址的偏移模式為偏移 0x4
```

注意：

cr0 和 cr3 各位功能可參考第 1 章 S+core7 內核暫存器。

在 *_startup.s 檔中指定了異常的基底位址，另外 *_Prog.Id 定義了包括可以定義包括 .DATA 、 .TEXT 、異常區段等所有工程中用到的區段， *_Prog.Id 也是在建立工程時會自動生成的檔，異常區段定義程式如下：

```
.exception 0xA00001fc :           // 定義中斷異常區段，起始位址為 0xA00001fc
{
    * (.exception_vec)    // 異常區段名 “.exception_vec” ，該區段在 Sys_isr.S 檔中定義
} = 0
```

Sys_isr.s 定義個各中斷向量的入口 (IntX_vec) 和調用中斷服務函數 (irq_dispatch(cp0_cause))；其中 IntX_vec 中的 X=1,2,3…62, 63 。

參考以下程式區段：由於.exception_vec 異常區段的起始位址為 0xa00001fc，兩個軟體位址

各占去 4 個位元組位址單元(字對齊)，所以 63 個硬體中斷向量的入口位址為 0xa0000200+ (0x04*n)，其中 0x04 為偏移量，n 為 1, 2, 3..., 62, 63。

```
=====
// 文件名: Sys_isr.S
// 功能描述: 中斷向量處理檔案
//           使用時，用戶不需要修改此檔中的內容
=====

#define     SP      r0
.section .exception_vec,"ax"
//定義異常中斷區段，在*_Prog.ld 中已經定義了其起始位址為 0xa00001fc
=====

// 定義兩個軟體中斷向量入口
=====

.align 2                      // 強制 norm_debug_vec 的位址為字對齊，  
norm_debug_vec:                // 即 norm_debug_vec 的位址為 0xa00001fc  
                                // Normal debug 異常向量入口  
        j    norm_debug_service  
.align 2                      // 強制 general_vec 的位址為字對齊，  
general_vec:                  // 即 general_vec 的位址為 0xa0000200  
                                // General 異常向量入口  
        j    general_service  
=====

// 定義 63 個硬體中斷向量入口
=====

.align 2                      // int1_vec 的位址為 0xa0000200+ (0x04*1)  
int1_vec:                     // 中斷向量 IRQ1 入口  
        j    int_service  
.align 2                      // int2_vec 的位址為 0xa0000200+ (0x04*2)  
int2_vec:                     // 中斷向量 IRQ2 入口  
        j    int_service  
.....  
.align 2                      // int24_vec 的位址為 0xa0000200+ (0x04*24)  
int24_vec:                    // (Reserved)中斷向量 IRQ24 入口  
        j    save_reg  
.align 2                      // int25_vec 的位址為 0xa0000200+ (0x04*25)  
int25_vec:                    // (Reserved)中斷向量 IRQ25 入口  
        j    save_reg  
.....  
.align 2                      // int56_vec 的位址為 0xa0000200+ (0x04*56)  
Int56_vec:                    // (Timer)中斷向量 IRQ56 入口  
        j    save_reg  
.....  
.align 2                      // int62_vec 的位址為 0xa0000200+ (0x04*62)  
int62_vec:                    // (Reserved)中斷向量 IRQ62 入口  
        j    save_reg  
.align 2                      // int63_vec 的位址為 0xa0000200+ (0x04*63)  
int63_vec:                    // (DAC)中斷向量 IRQ63 入口  
        j    save_reg  
=====

.extern intmsg                 // 該函數在 Sys_IRQ.c 檔中定義
=====

// 軟體中斷服務函數
=====

norm_debug_service:           // 軟體中斷 1(Normal debug 異常)中斷服務程式
```

```
jl intmsg           // 調用 intmsg 函數

general_service:
    jl intmsg      // 軟體中斷 2(General 異常)中斷服務程式
                    // 調用 intmsg 函數

int_service:
    jl intmsg      // 調用 intmsg 函數

.extern irq_dispatch // 該函數在 Sys_IRQ.c 檔中定義
.set r1
=====
// 組合語言格式: save_reg
// C 語言 格式: void save_reg(void)
// 功能 描述: 中斷服務函數,
//               保護及恢復現場, 調用 irq_dispatch 函數
// 入口 參數: 無
// 出口 參數: 無
=====
save_reg:
    // 保存現場: 保存 r1~r31 暫存器, 保存條件暫存器 cr1 和中斷計數器 cr5
    subi    SP, 38*4
    sw     r1, [SP,1*4]
    sw     r2, [SP,2*4]
    sw     r3, [SP,3*4]
    sw     r4, [SP,4*4]
    sw     r5, [SP,5*4]
    sw     r6, [SP,6*4]
    sw     r7, [SP,7*4]
    sw     r8, [SP,8*4]
    sw     r9, [SP,9*4]
    sw    r10, [SP,10*4]
    sw   r11, [SP,11*4]
    sw   r12, [SP,12*4]
    sw   r13, [SP,13*4]
    sw   r14, [SP,14*4]
    sw   r15, [SP,15*4]
    sw   r16, [SP,16*4]
    sw   r17, [SP,17*4]
    sw   r18, [SP,18*4]
    sw   r19, [SP,19*4]
    sw   r20, [SP,20*4]
    sw   r21, [SP,21*4]
    sw   r22, [SP,22*4]
    sw   r23, [SP,23*4]
    sw   r24, [SP,24*4]
    sw   r25, [SP,25*4]
    sw   r26, [SP,26*4]
    sw   r27, [SP,27*4]
    sw   r28, [SP,28*4]
    sw   r29, [SP,29*4]
    sw   r30, [SP,30*4]
    sw   r31, [SP,31*4]
    mfcr   r13, cr1
    mfcr   r15, cr5
    sw   r13, [SP,33*4]
```

```
sw r15, [SP,35*4]

mfcr r4, cr2
// 透過 cr2 的 b23~b16 讀取中斷請求的向量號，並把此資料作為調用函數的入口參數
jl irq_dispatch // 調用 irq_dispatch 函數，此函數在 Sys_IRQ.c 中定義

// 恢復現場：恢復 r1~r31 暫存器，條件暫存器 cr1 和中斷計數器 cr5 中斷前的資料
lw r1, [SP,1*4]
lw r2, [SP,2*4]
lw r3, [SP,3*4]
lw r4, [SP,4*4]
lw r5, [SP,5*4]
lw r6, [SP,6*4]
lw r7, [SP,7*4]
lw r8, [SP,8*4]
lw r9, [SP,9*4]
lw r10, [SP,10*4]
lw r11, [SP,11*4]
lw r12, [SP,12*4]
lw r13, [SP,13*4]
lw r14, [SP,14*4]
lw r15, [SP,15*4]
lw r16, [SP,16*4]
lw r17, [SP,17*4]
lw r18, [SP,18*4]
lw r19, [SP,19*4]
lw r20, [SP,20*4]
lw r21, [SP,21*4]
lw r22, [SP,22*4]
lw r23, [SP,23*4]
lw r24, [SP,24*4]
lw r25, [SP,25*4]
lw r26, [SP,26*4]
lw r27, [SP,27*4]
lw r28, [SP,28*4]
lw r29, [SP,29*4]
lw r30, [SP,30*4]
lw r31, [SP,31*4]
lw r30, [SP,33*4]
lw r31, [SP,35*4]
mtcr r30, cr1
mtcr r31, cr5
addi SP, 38*4
rte
```

Sys_IRQ.c 定義了 irq_dispatch(cp0_cause)函數和 intmsg()函數：irq_dispatch(cp0_cause)函數在 Sys_isr.S 檔案中的 save_reg 函數中調用，為中斷的分支函數，cp0_cause 為 cr2 的內容，根據 cr2 的內容調用 IRQ63()~IRQ24()中的相應中斷服務程式；intmsg()函數函數在 Sys_isr.S 檔案中的 norm_debug_service 函數、general_service 函數和 int_service 函數中調用，為兩個軟體中斷和除 SPCE3200 的 40 個硬體中斷外的其他硬體中斷的處理程式。程式段如下：

```
//=====
//文件名: Sys_IRQ.c
//功能描述: SPC3200 40 個中斷源(硬體)的中斷向量分支及軟體中斷處理。
//          使用時，用戶不需要修改此檔中的內容
//=====

//=====
//語法格式: void intmsg(void)
//功能描述: 軟體中斷及除 SPC3200 40 個硬體中斷之外的
//          S+core7 內核硬體中斷處理程式
//入口參數: 無
//出口參數: 無
//=====
void intmsg(void)
{
    while(1);
}

//=====
//語法格式: void irq_dispatch(unsigned int cp0_cause)
//功能描述: 取 SPC3200 40 個硬體中斷的中斷向量號,
//          根據中斷向量號調用相應中斷服務程式
//入口參數: cp0_cause 為 S+core7 內核暫存器 cr2 的內容
//          其中第 23~18 位元為 63 個硬體中斷的中斷請求狀態(中斷向量號)
//出口參數: 無
//=====
void irq_dispatch(unsigned int cp0_cause)
{
    int intvec=0;

    intvec = (cp0_cause & 0x00FC0000)>>18; // 取中斷向量號

    switch (intvec)
    {
        case 63:           // DAC 中斷
            IRQ63();       // IRQ63()函數在 User_IRQ.c 檔中定義
            break;
        case 62:           // 保留
            IRQ62();       // IRQ62()函數在 User_IRQ.c 檔中定義
            break;
        .....
        case 56:           // Timer 中斷
            IRQ56();       // IRQ56()函數在 User_IRQ.c 檔中定義
            break;
        .....
        case 25:           // 保留
            IRQ25();       // IRQ25()函數在 User_IRQ.c 檔中定義
            break;
        case 24:           // 保留
            IRQ24();       // IRQ24()函數在 User_IRQ.c 檔中定義
            break;
        default:
            break;
    }
    return;
}
```

User_IRQ.c 中定義了 IRQ63()~IRQ24()四十個中斷服務函數，為一個用戶介面函數檔，用戶可以在其中添加用戶中斷服務函數。

```
=====
//文件名：User_IRQ.c
//功能描述：SPCE3200 40 個中斷源的用戶中斷服務函數。
//          根據中斷向量號，用戶可編寫相應中斷服務函數
=====

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

=====

//語法格式：void IRQ63(void)
//功能描述：DAC 中斷服務函數
//入口參數：無
//出口參數：無
=====

void IRQ63(void)
{
}

=====

//語法格式：void IRQ62(void)
//功能描述：保留中斷向量中斷服務函數
//入口參數：無
//出口參數：無
=====

void IRQ62(void)
{
}

......

=====

//語法格式：void IRQ56(void)
//功能描述：計時器 (Timer) 中斷服務函數
//入口參數：無
//出口參數：無
=====

void IRQ56(void)
{
}

......

=====

//語法格式：void IRQ25(void)
//功能描述：保留中斷向量中斷服務函數
//入口參數：無
//出口參數：無
=====

void IRQ25(void)
{
}

=====
```

```

// 語法格式：void IRQ24(void)
// 功能描述：保留中斷向量中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ24(void)
{
}

```

在使用 SPC3200 的 40 個中斷（63~24）時，各函數之間的調用關係如下：

當有中斷請求時，先按照位址從 IntX_vec 入口，IntX_vec 為 Int63_vec~Int24_vec；在 IntX_vec 中調用 save_reg 函數進行中斷服務的處理；在 save_reg 函數中先保存現場，取中斷向量號並把中斷向量號作為參數，調用 irq_dispatch 函數；save_reg 函數中把中斷向量號作為參數傳遞給 irq_dispatch 函數，irq_dispatch 函數按照中斷向量號調用相應的中斷服務函數 IRQX()，IRQX 為 IRQ24()~IRQ63()。

所以 SPC3200 的中斷處理過程也可看作的處理過程：

的調用關係為中斷從回應入口到處理完中斷服務程式的過程，即左邊粗箭頭的處理過程；處理中斷服務程式後，按照右邊粗箭頭的過程恢復現場，完成一次中斷的處理過程。

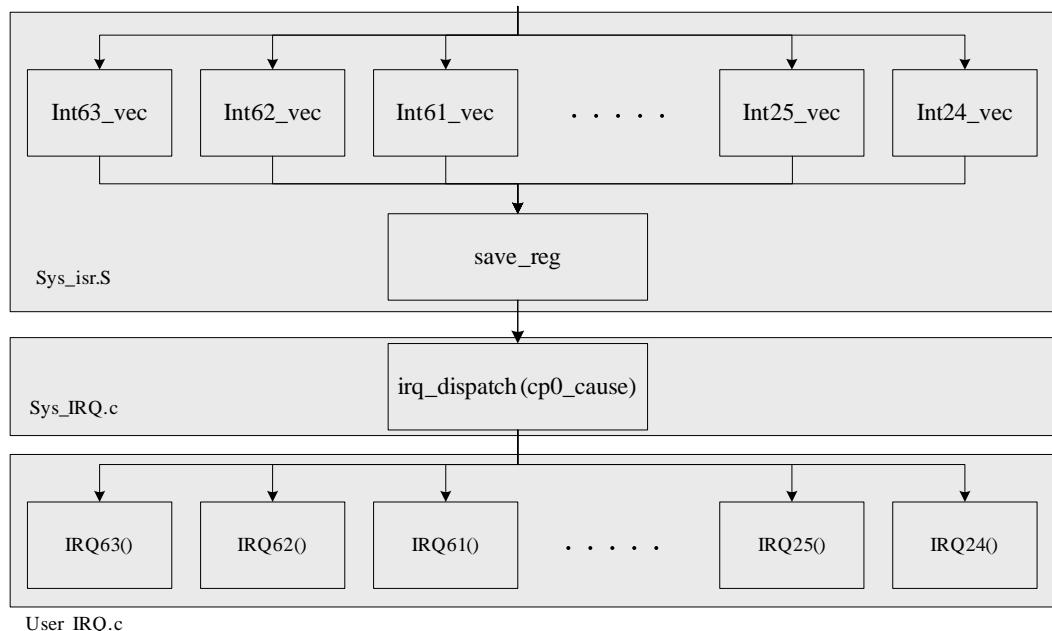


圖 3-7 中斷各函數的調用關係圖

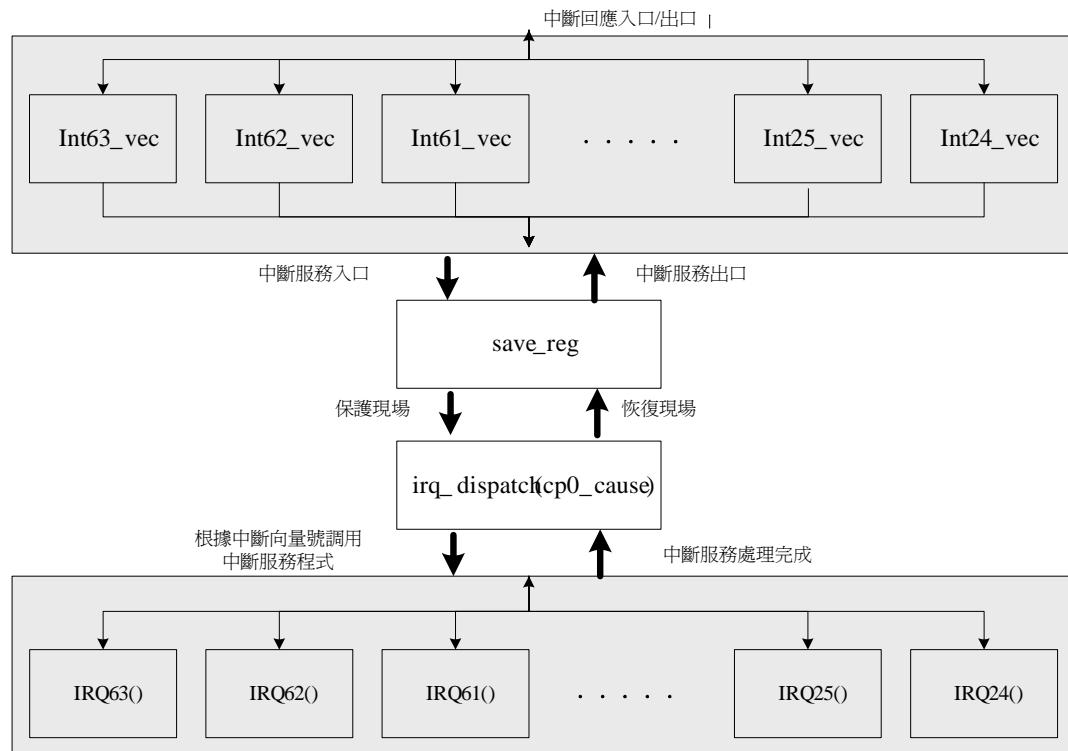


圖 3-8 中斷服務處理過程圖

注意：

如果沒有特別說明，IRQ24()~IRQ63()中斷服務函數中都需要手動清除中斷旗標。

3.6.7 應用舉例

【例 3.1】

利用 Timer 中斷實現三個發光二極體電平翻轉（即閃爍）。

分析：根據圖 3-6，中斷控制器必須關閉來自 Timer 中斷源的中斷遮罩；而且，必須透過相應暫存器使能 Timer 模組中斷，Timer 才會發出中斷請求；所以主程序中主要進行中斷使能的操作。

按照表 3-15，Timer 中斷源對應的中斷向量號為 56，則根據 3.6.6 節中斷機制可指導，中斷服務處理中只需要在 IRQ56()函數中編寫二極體電平翻轉程式即可。

主程序參考程式：

```

int main()
{
    *P_IOB_GPIO_SETUP = 0x00003800;
    *P_INT_MASK_CTRL1 = 0xffffffff7f; // 打開來自 Timer 中斷源的中斷

    // *P_CLK_32K_CTRL = C_32K_CRY_EN; // 如果選擇 32768Hz 時鐘使能晶振
}

```

```

        *P_TIMER0_CLK_CONF = C_TIMER_CLK_EN
            | C_TIMER_RST_DIS; // 配置 Timer0 的時鐘
        *P_CKG_SEL_TIMER = C_TIMER0_CLK_27M
            | 0x8; // 設置計時器源時鐘頻率為 27MHZ / 9
        *P_TIMER0_PRELOAD_DATA = 0x80; // 設置計數初值
        *P_TIMER0_CCP_CTRL = C_TIMER_NOR_MODE; // 工作模式選擇定時計數模式
        *P_TIMER0_MODE_CTRL = C_TIMER_CTRL_EN
            | C_TIMER_INT_EN
            | C_TIMER_INT_FLAG; // 使能計時器、使能中斷、清中斷
    while(1);
}
    
```

中斷服務參考程式：

```

void IRQ56(void)
{
    if(*P_TIMER0_MODE_CTRL & C_TIMER_INT_FLAG)
    {
        *P_TIMER0_MODE_CTRL |= C_TIMER_INT_FLAG; // 清除計時器中斷
        *P_IOB_GPIO_SETUP ^= 0x00000038; // LED1~3 反相,
    }
}
    
```

3.7 記憶體介面單元——MIU

SPCE3200 晶片上具備記憶體介面單元 (MIU—Memory Interface Unit)，可以更好的設計記憶體介面。結構如圖 3-9 所示：

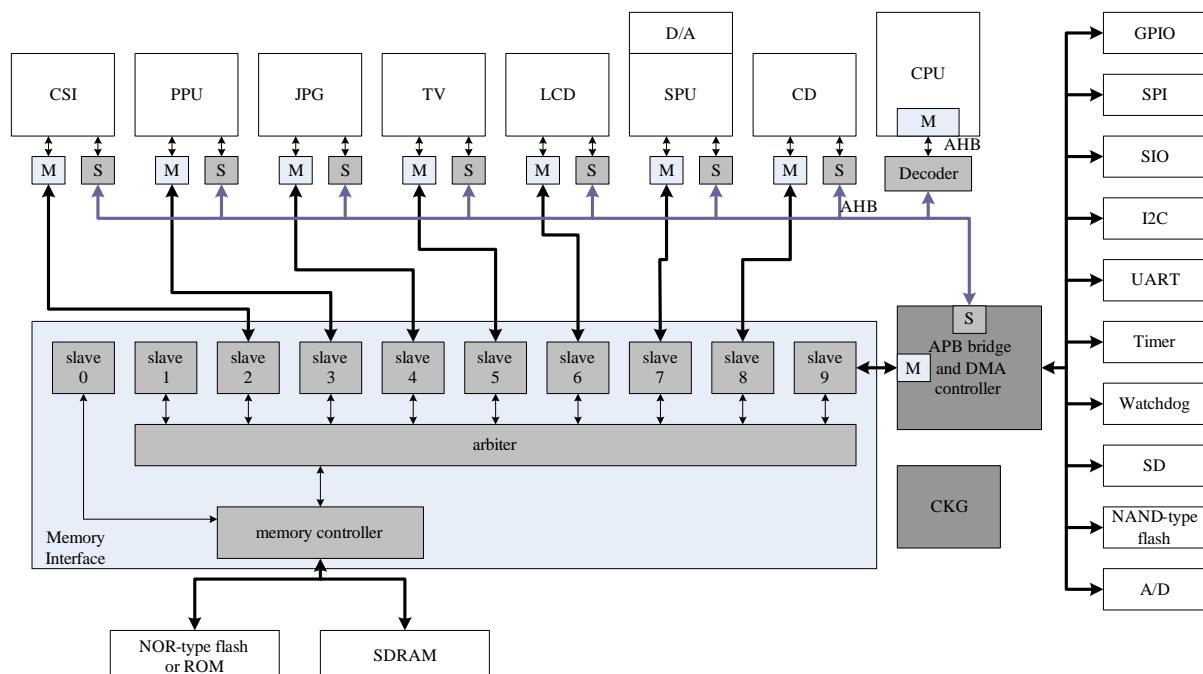


圖 3-9 MIU 結構圖

MIU 存儲介面暫存器主要涉及 TV、LCD、SDRAM、MP4 模組，這裏只介紹與 SDRAM 相關的暫存器，與其他模組相關暫存器只列出，更詳細的介紹請參考相關章節。

表 3-27 MIU 涉及模組相關暫存器

TV 相關暫存器	P_TV_BUFFER_SA1
	P_TV_BUFFER_SA2
	P_TV_BUFFER_SA3
LCD 相關暫存器	P_LCD_BUFFER_SA1
	P_LCD_BUFFER_SA2
	P_LCD_BUFFER_SA3
MPEG4 相關暫存器	P_MPEG4_RAWBUFFER_SA1
	P_MPEG4_RAWBUFFER_SA2
	P_MPEG4_RAWBUFFER_SA3
	P_MPEG4_WRITEBUFFER_SA1
	P_MPEG4_WRITEBUFFER_SA2
	P_MPEG4_WRITEBUFFER_SA3
	P_MPEG4_VLCBUFFER_SA1
	P_MPEG4_VLCBUFFER_SA2
	P_MPEG4_FRAMEBUFFER_HSIZE
SDRAM 相關暫存器	P_MIU_SDRAM_POWER
	P_MIU_SDRAM_SETUP1
	P_MIU_SDRAM_SETUP2
	P_MIU_SDRAM_STATUS
MIU 時鐘暫存器	P_MIU_CLK_CONF

■ MIU 時鐘配置暫存器：P_MIU_CLK_CONF(0x88210010)

MIU 時鐘配置暫存器用於使能或禁止 MIU 模組時鐘，或軟重設 MIU 模組。

表 3-28 P_MIU_CLK_CONF(0x88210010)

位	b31~b3	b2	b1	b0
讀/寫	-	R/W	R/W	R/W
預設值	-	0	0	0
名稱	-	MIU_RST	MIU_DRAM	MIU_STOP

MIU_RST	b2	MIU 模組時鐘重設位： 0 : MIU 模組時鐘重設 1 : MIU 模組時鐘不重設
MIU_DRAM	b1	強制 SDRAM 進入自刷新模式同時停止 MIU 時鐘直到喚醒發生： 0 : 使能進入自刷新模式 1 : 禁止進入自刷新模式
MIU_STOP	b0	MIU 模組時鐘停止位： 0 : MIU 模組時鐘停止 1 : MIU 模組時鐘使能

■ SDRAM 電源暫存器：P_MIU_SDRAM_POWER(0x8807005C)

SDRAM 電源設置。

表 3-29 P_MIU_SDRAM_POWER(0x8807005C)

位	b31~b4	b3	b2	b1	b0
讀/寫	-	R/W	R/W	R/W	R/W
預設值	-	0	0	0	0
名稱	-	DEEP	REDO_MRS	POWER_DW	SELF_REF

DEEP	b3	強制 SDRAM 進入深度省電模式使能位： 0 : 禁止 SDRAM 進入深度省電模式 1 : 使能 SDRAM 進入深度省電模式
------	----	---

REDO_MRS	b2	重新設置模式暫存器使能位： 0：禁止重新設置模式暫存器 1：使能重新設置模式暫存器
POWER_DW	b1	強制 SDRAM 進入省電模式使能位： 0：禁止 SDRAM 進入省電模式 1：使能 SDRAM 進入省電模式
SELF_REF	b0	強制 SDRAM 進入自動刷新模式： 0：禁止 SDRAM 進入自動刷新模式 1：使能 SDRAM 進入自動刷新模式

■ SDRAM 參數設置暫存器 1： P_MIU_SDRAM_SETUP1(0x88070060)

SDRAM 的相關參數設置。

表 3-30 P_MIU_SDRAM_SETUP1(0x88070060)

位	b31	b30~b27	b26~b25	b24~b23	b22~b15
讀/寫	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0
名稱	MIU_EN	SDCLK_SEL	SDRAM_RN	SDRAM_CW	SFRFR_PERIOD
位	b14~b11	b10~b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0
名稱	RFR_CYCLE_NUM	ATRFR_PERIOD	CAS_LA_CYCLE	tRCD_CYCLE	tRP_CYCLE

MIU_EN b31 MIU 使能位：

0：禁止 MIU

1：使能 MIU

SDCLK_SEL b30~b27 SDRAM 時鐘選擇位：

選擇 SDRAM_CLK 的相位延遲

SDRAM_RN	b26~b25	每個邏輯 Bank (L-Bank) 包含的行數選擇位： 00 : 1024 行每邏輯 Bank 01 : 2048 行每邏輯 Bank 10 : 4096 行每邏輯 Bank 11 : 8192 行每邏輯 Bank
SDRAM_CW	b24~b23	SDRAM 每列包含的位數： 00 : 8 位寬度 01 : 16 位寬度 10 : 32 位寬度 11 : 保留
SFRFR_PERIOD	b22~b15	若在 ($SFRFR_PERIOD \times 256 \times ATRFR_PERIOD \times 16$) 個時鐘週期裏沒有發生 ADG_REQ，則會由硬體觸發自動刷新動作
RFR_CYCLE_NUM	b14~b11	確定自動刷新需要的時鐘週期數
ATRFR_PERIOD	b10~b3	每隔 ($ATRFR_PERIOD \times 16$) 個時鐘週期產生自動刷新動作
CAS_LA_CYCLE	b2	Cas latency 時鐘週期設定 0 : 2 個時鐘週期 (默認設置) 1 : 3 個時鐘週期
tRCD_CYCLE	b1	tRCD 時鐘週期設定 0 : 1 個時鐘週期 1 : 2 個時鐘週期 (默認設置)
tRP_CYCLE	b0	tRP 時鐘週期設定 0 : 1 個時鐘週期 1 : 2 個時鐘週期 (默認設置)

■ SDRAM 參數設置暫存器 2： P_MIU_SDRAM_SETUP2(0x88070094)

SDRAM 的相關參數設置。

表 3-31 P_MIU_SDRAM_SETUP2(0x88070094)

位	b1	b0
讀/寫	R/W	R/W
預設值	0	0
名稱	tRCD_CYCLE	tRP_CYCLE

tRCD_CYCLE	b1	tRCD 時鐘週期設置： 0 : 參考 P_MIU1_SDRAM_SETTING[1]位設置 1 : 3 個時鐘週期
tRP_CYCLE	b0	tRP 時鐘週期設置 0 : 參考 P_MIU1_SDRAM_SETTING[0]位設置 1 : 3 個時鐘週期

■ SDRAM 狀態暫存器：P_MIU_SDRAM_STATUS(0x8807006C)

反映 MIU 的工作狀態。

表 3-32 P_MIU_SDRAM_STATUS(0x8807006C)

位	b0
讀/寫	R/W
預設值	0
名稱	MIU_STATUS

MIU_STATUS	b0	SDRAM 的狀態位： 0 : SDRAM 没有状态发生 1 : SDRAM 处于自动刷新模式或省电模式
------------	----	--

3.8 APB 汇流排 DMA

SPCE3200 具有 DMA 功能，透過 DMA 功能可以完成 APB 汇流排讀取 APB 週邊設備模組資料寫到 MIU 記憶體，或從 MIU 記憶體讀取資料寫到 APB 週邊設備模組。APBDMA 結構如圖 3-10 所示：

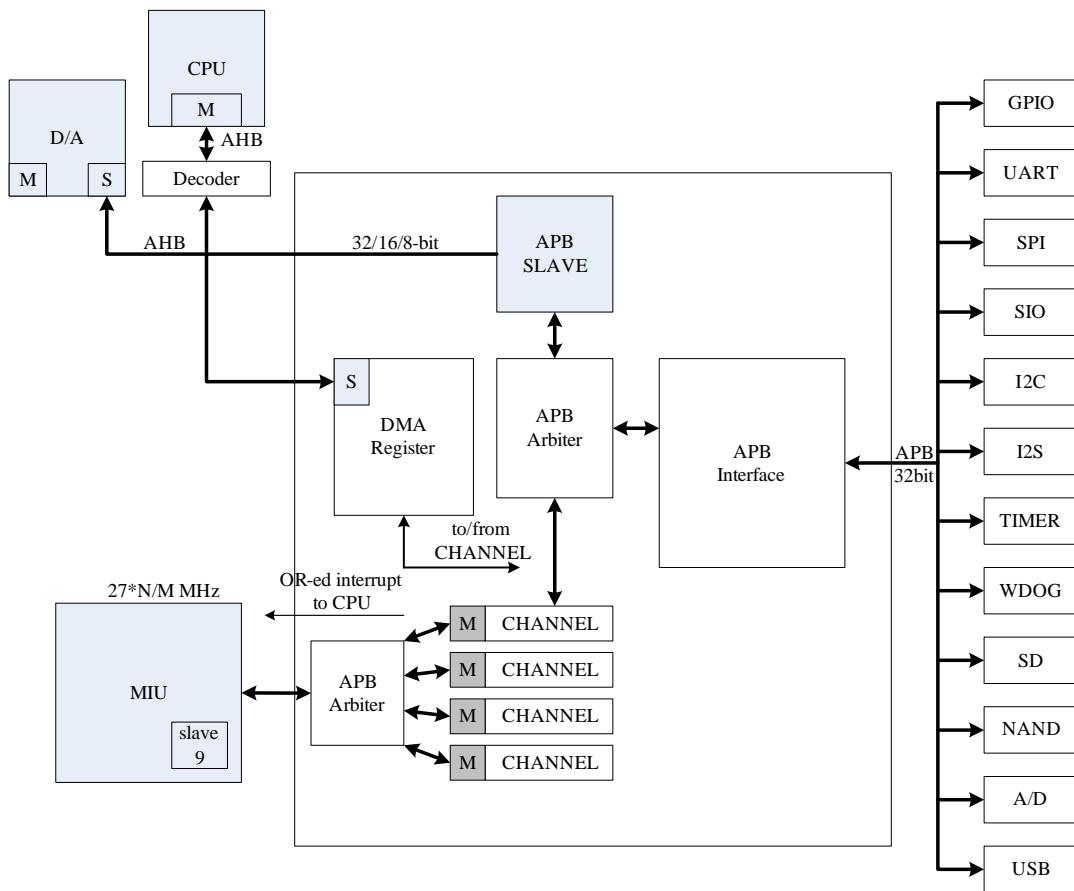


圖 3-10 APBDMA 結構圖

DMA 控制器可為 APB 週邊設備模組同時提供 4 個用於讀/寫 MIU 記憶體的通道，每一通道都可以設置成下列 4 類傳輸方式：

- 8 位單通道傳輸
- 16 位單通道傳輸
- 32 位單通道傳輸
- 32 位突發模式傳輸

DMA 具有 2 種啓動方式：

- (1) 當 DMA 通道被使能（通道使能位元被設為 1）後，DMA 控制器透過通道連續地讀出或寫入資料，並在完成讀寫操作後結束 DMA 操作；
- (2) 當 DMA 通道被使能後，在 APB 週邊設備模組發出 REQ 請求時 DMA 控制器便會進行讀寫操作一次，並當所有 REQ 請求的讀寫操作完成後結束操作。

當 DMA 控制器讀/寫 MIU 時有 2 種記憶體存儲方式：

- (1) 單緩存區方式：即指定 DMA 緩存區的起始、結束位址，則 DMA 控制器僅對這一存儲空間進行讀寫操作，並在讀寫操作完成時透過發出 IRQ 信號來結束操作；
- (2) 雙緩存區方式：即同時指定 BUFFER A 和 BUFFER B 緩存區的起始、結束位址，則 DMA 控制器會交替對 BUFFER A 和 BUFFER B 進行讀寫操作，並在二者有關讀寫操作的傳輸完成後發出 IRQ 信號。將 DMA 通道使能位設回到 ‘0’ 時，會產生 DMA 操作終止請求，則 DMA 控制器會在當前傳輸完成時結束其操作。

當 DMA 向 APB 週邊設備模組進行讀寫操作時，有 2 種定址方式來定位週邊設備埠：

- (1) 常規定址；
- (2) 連續定址。

注意：

當 DMA 操作完成後，用戶需要寫 ‘1’ 到相應的通道 IRQ 狀態位元以清除 DMA 中斷，並需要寫 ‘0’ 到相應的通道 DMA 使能位以結束其 DMA 操作。

DMA 相關暫存器：

DMA 控制器共有 28 個暫存器，如表 3-33 所示。透過對這 28 個暫存器的操作，即可使用 DMA 操作。下麵將對這些暫存器一一進行說明。

表 3-33 DMA 控制器相關暫存器列表

暫存器名稱	助記符	位址
DMA 時鐘配置暫存器	P_DMA_CLK_CONF	0x88210058
DMA 忙狀態暫存器	P_DMA_BUSY_STATUS	0x88080000
DMA 中斷狀態暫存器	P_DMA_INT_STATUS	0x88080004
AHB DMA 第 0 通道緩衝區 A 起始位址	P_DMA_AHB_SA0BA	0x88080008
AHB DMA 第 1 通道緩衝區 A 起始位址	P_DMA_AHB_SA1BA	0x8808000C
AHB DMA 第 2 通道緩衝區 A 起始位址	P_DMA_AHB_SA2BA	0x88080010
AHB DMA 第 3 通道緩衝區 A 起始位址	P_DMA_AHB_SA3BA	0x88080014
AHB DMA 第 0 通道緩衝區 A 結束位址	P_DMA_AHB_EA0BA	0x88080018
AHB DMA 第 1 通道緩衝區 A 結束位址	P_DMA_AHB_EA1BA	0x8808001C
AHB DMA 第 2 通道緩衝區 A 結束位址	P_DMA_AHB_EA2BA	0x88080020
AHB DMA 第 3 通道緩衝區 A 結束位址	P_DMA_AHB_EA3BA	0x88080024
APB DMA 第 0 通道起始位址	P_DMA_APB_SA0	0x88080028

暫存器名稱	助記符	位址
APB DMA 第 1 通道起始位址	P_DMA_APB_SA1	0x8808002C
APB DMA 第 2 通道起始位址	P_DMA_APB_SA2	0x88080030
APB DMA 第 3 通道起始位址	P_DMA_APB_SA3	0x88080034
AHB DMA 第 0 通道緩衝區 B 起始位址	P_DMA_AHB_SA0BB	0x8808004C
AHB DMA 第 1 通道緩衝區 B 起始位址	P_DMA_AHB_SA1BB	0x88080050
AHB DMA 第 2 通道緩衝區 B 起始位址	P_DMA_AHB_SA2BB	0x88080054
AHB DMA 第 3 通道緩衝區 B 起始位址	P_DMA_AHB_SA3BB	0x88080058
AHB DMA 第 0 通道緩衝區 B 結束位址	P_DMA_AHB_EA0BB	0x8808005C
AHB DMA 第 1 通道緩衝區 B 結束位址	P_DMA_AHB_EA1BB	0x88080060
AHB DMA 第 2 通道緩衝區 B 結束位址	P_DMA_AHB_EA2BB	0x88080064
AHB DMA 第 3 通道緩衝區 B 結束位址	P_DMA_AHB_EA3BB	0x88080068
DMA 第 0 通道控制暫存器	P_DMA_CHANNEL0_CTRL	0x8808006C
DMA 第 1 通道控制暫存器	P_DMA_CHANNEL1_CTRL	0x88080070
DMA 第 2 通道控制暫存器	P_DMA_CHANNEL2_CTRL	0x88080074
DMA 第 3 通道控制暫存器	P_DMA_CHANNEL3_CTRL	0x88080078
DMA 通道重設暫存器	P_DMA_CHANNEL_RESET	0x8808007C

■ DMA 時鐘配置暫存器：P_DMA_CLK_CONF(0x88210058)

DMA 時鐘配置暫存器用於使能或禁止 DMA 控制器模組時鐘，或軟重設 DMA 控制器模組。

表 3-34 P_DMA_CLK_CONF(0x88210058)

位	b23~b21	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	APBDMA_RST	APBDMA_STOP

APBDMA_RST b1

APBDMA 模組時鐘重設位：

0 : APBDMA 模組時鐘重設

1 : APBDMA 模組時鐘不重設

APBDMA_STOP	b0	APBDMA 模組時鐘停止位： 0 : APBDMA 模組時鐘停止 1 : APBDMA 模組時鐘使能
-------------	----	---

■ DMA 忙狀態暫存器：**P_DMA_BUSY_STATUS(0x88080000)**

DMA 忙狀態暫存器反映 DMA 的 4 個通道是否出於忙狀態。

表 3-35 P_DMA_BUSY_STATUS(0x88080000)

位	b3	b2	b1	b0
讀/寫	R	R	R	R
預設值	0	0	0	0
名稱	CH3_BUSY	CH2_BUSY	CH1_BUSY	CH0_BUSY

CH3_BUSY	b3	爲 1 時表示通道 3 處於忙狀態
CH2_BUSY	b2	爲 1 時表示通道 2 處於忙狀態
CH1_BUSY	b1	爲 1 時表示通道 1 處於忙狀態
CH0_BUSY	b0	爲 1 時表示通道 0 處於忙狀態

■ DMA 中斷狀態暫存器：**P_DMA_INT_STATUS(0x88080004)**

DMA 中斷狀態暫存器反映 DMA 的 4 個通道的中斷是否發生、中斷旗標清除等操作。

表 3-36 P_DMA_INT_STATUS(0x88080004)

位	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W
預設值	0	0	0	0
名稱	CH3_IRQ	CH2_IRQ	CH1_IRQ	CH0_IRQ

CH3_IRQ	b3	DMA 通道 3 的 IRQ 中斷旗標位： 讀 0：未發生 DMA IRQ 中斷 讀 1：發生 DMA IRQ 中斷 寫 0：無意義 寫 1：清除 DMA IRQ 狀態旗標
CH2_IRQ	b2	DMA 通道 3 的 IRQ 中斷旗標位： 讀 0：未發生 DMA IRQ 中斷 讀 1：發生 DMA IRQ 中斷 寫 0：無意義 寫 1：清除 DMA IRQ 狀態旗標
CH1_IRQ	b1	DMA 通道 3 的 IRQ 中斷旗標位： 讀 0：未發生 DMA IRQ 中斷 讀 1：發生 DMA IRQ 中斷 寫 0：無意義 寫 1：清除 DMA IRQ 狀態旗標
CH0_IRQ	b0	DMA 通道 3 的 IRQ 中斷旗標位： 讀 0：未發生 DMA IRQ 中斷 讀 1：發生 DMA IRQ 中斷 寫 0：無意義 寫 1：清除 DMA IRQ 狀態旗標

- AHB DMA 第 0 通道緩存區 A 起始位址：**P_DMA_AHB_SA0BA(0x88080008)**
- AHB DMA 第 1 通道緩存區 A 起始位址：**P_DMA_AHB_SA1BA(0x8808000C)**
- AHB DMA 第 2 通道緩存區 A 起始位址：**P_DMA_AHB_SA2BA(0x88080010)**
- AHB DMA 第 3 通道緩存區 A 起始位址：**P_DMA_AHB_SA3BA(0x88080014)**

AHB DMA 第*通道緩存區 A 起始位址暫存器設置第*通道緩存區 A 的起始位址。

表 3-37 P_DMA_AHB_SAxBa

暫存器名稱	b31~b0
P_DMA_AHB_SA0BA	第 0 通道緩存區 A 起始位址
P_DMA_AHB_SA1BA	第 1 通道緩存區 A 起始位址
P_DMA_AHB_SA2BA	第 2 通道緩存區 A 起始位址
P_DMA_AHB_SA3BA	第 3 通道緩存區 A 起始位址

- AHB DMA 第 0 通道緩存區 A 結束位址 : **P_DMA_AHB_EA0BA(0x88080018)**
- AHB DMA 第 1 通道緩存區 A 結束位址 : **P_DMA_AHB_EA1BA(0x8808001C)**
- AHB DMA 第 2 通道緩存區 A 結束位址 : **P_DMA_AHB_EA2BA(0x88080020)**
- AHB DMA 第 3 通道緩存區 A 結束位址 : **P_DMA_AHB_EA3BA(0x88080024)**

AHB DMA 第*通道緩存區 A 結束位址暫存器設置第*通道緩存區 A 的結束位址。

表 3-38 P_DMA_AHB_EAxBA

暫存器名稱	b31~b0
P_DMA_AHB_EA0BA	第 0 通道緩存區 A 結束位址
P_DMA_AHB_EA1BA	第 1 通道緩存區 A 結束位址
P_DMA_AHB_EA2BA	第 2 通道緩存區 A 結束位址
P_DMA_AHB_EA3BA	第 3 通道緩存區 A 結束位址

- APB DMA 第 0 通道起始位址 : **P_DMA_APB_SA0(0x88080028)**
- APB DMA 第 1 通道起始位址 : **P_DMA_APB_SA1(0x8808002C)**
- APB DMA 第 2 通道起始位址 : **P_DMA_APB_SA2(0x88080030)**
- APB DMA 第 3 通道起始位址 : **P_DMA_APB_SA3(0x88080034)**

APB DMA 第*通道起始位址暫存器設置第*通道的起始位址。

表 3-39 P_DMA_APB_Sax

暫存器名稱	b31~b0
P_DMA_APB_SA0	第 0 通道 APB 模組存取埠起始位址
P_DMA_APB_SA1	第 1 通道 APB 模組存取埠起始位址

暫存器名稱	b31~b0
P_DMA_APB_SA2	第 2 通道 APB 模組存取埠起始位址
P_DMA_APB_SA3	第 3 通道 APB 模組存取埠起始位址

- AHB DMA 第 0 通道緩存區 B 起始位址 : P_DMA_AHB_SA0BB(0x8808004C)
- AHB DMA 第 1 通道緩存區 B 起始位址 : P_DMA_AHB_SA1BB(0x88080050)
- AHB DMA 第 2 通道緩存區 B 起始位址 : P_DMA_AHB_SA2BB(0x88080054)
- AHB DMA 第 3 通道緩存區 B 起始位址 : P_DMA_AHB_SA3BB(0x88080058)

AHB DMA 第*通道緩存區 B 起始位址暫存器設置第*通道緩存區 B 的起始位址。

表 3-40 P_DMA_AHB_SAxBB

暫存器名稱	b31~b0
P_DMA_AHB_SA0BB	第 0 通道緩存區 B 起始位址
P_DMA_AHB_SA1BB	第 1 通道緩存區 B 起始位址
P_DMA_AHB_SA2BB	第 2 通道緩存區 B 起始位址
P_DMA_AHB_SA3BB	第 3 通道緩存區 B 起始位址

- AHB DMA 第 0 通道緩存區 B 結束位址 : P_DMA_AHB_EA0BB(0x8808005C)
- AHB DMA 第 1 通道緩存區 B 結束位址 : P_DMA_AHB_EA1BB(0x88080060)
- AHB DMA 第 2 通道緩存區 B 結束位址 : P_DMA_AHB_EA2BB(0x88080064)
- AHB DMA 第 3 通道緩存區 B 結束位址 : P_DMA_AHB_EA3BB(0x88080068)

AHB DMA 第*通道緩存區 B 結束位址暫存器設置第*通道緩存區 B 的結束位址。

表 3-41 P_DMA_AHB_EAxBB

暫存器名稱	b31~b0
P_DMA_AHB_EA0BB	第 0 通道緩存區 B 結束位址
P_DMA_AHB_EA1BB	第 1 通道緩存區 B 結束位址
P_DMA_AHB_EA2BB	第 2 通道緩存區 B 結束位址
P_DMA_AHB_EA3BB	第 3 通道緩存區 B 結束位址

- DMA 第 0 通道控制暫存器：P_DMA_CHANNEL0_CTRL(0x8808006C)
- DMA 第 1 通道控制暫存器：P_DMA_CHANNEL1_CTRL(0x88080070)
- DMA 第 2 通道控制暫存器：P_DMA_CHANNEL2_CTRL(0x88080074)
- DMA 第 3 通道控制暫存器：P_DMA_CHANNEL3_CTRL(0x88080078)

DMA 第*通道控制暫存器設置 DMA 控制器的使能、中斷、傳輸方式選擇位、記憶體存儲方式選擇位、APB 週邊設備模組定位定址模式和讀寫方向等。

表 3-42 P_DMA_CHANNELx_CTRL

位	b7	b6	b5~b4	b3	b2	b1	b0
讀/寫	W	W	W	W	W	W	W
預設值	0	0	0	0	0	0	0
名稱	CHx_EN	CHx_IRQ	CHx_TRANS	CHx_MEM	CHx_MODE	CHx_DMA	CHx_DIR

CHx_EN	b7	通道 x DMA 使能位： 0 : 禁止 1 : 使能
CHx_IRQ	b6	通道 x DMA 中斷遮罩： 0 : 遮罩 DMA IRQ 1 : 允許 DMA IRQ
CHx_TRANS	b5~b4	通道 x 傳輸方式選擇位： 00 : 8 位單通道傳輸方式 01 : 16 位單通道傳輸方式 10 : 32 位單通道傳輸方式 11 : 32 位突發模式傳輸
CHx_MEM	b3	通道 x MIU 記憶體存儲方式選擇位： 0 : 單緩存區存儲方式 1 : 雙緩存區存儲方式
CHx_MODE	b2	通道 x DMA 的 APB 週邊設備模組定位定址模式： 0 : 連續模式 1 : 常規模式

CHx_DMA	b1	通道 x DMA 方式選擇位：
---------	----	-----------------

0：自動方式

1：查詢方式

CHx_DIR	b0	通道 x 讀寫方向設置位：
---------	----	---------------

0：MIU 向 APB 的讀寫

1：APB 向 MIU 的讀寫

■ DMA 通道重設暫存器：P_DMA_CHANNEL_RESET(0x8808007C)

DMA 通道重設暫存器重設 DMA 的通道。

表 3-43 P_DMA_CHANNEL_RESET(0x8808007C)

位	b3	b2	b1	b0
讀/寫	W	W	W	W
預設值	0	0	0	0
名稱	CH3_REST	CH2_REST	CH1_REST	CH0_REST

CH3_REST	b3	通道 3 軟體重設位：
----------	----	-------------

0：不重設

1：重設

CH2_REST	b2	通道 2 軟體重設位：
----------	----	-------------

0：不重設

1：重設

CH1_REST	b1	通道 1 軟體重設位：
----------	----	-------------

0：不重設

1：重設

CH0_REST	b0	通道 0 軟體重設位：
----------	----	-------------

0：不重設

1：重設

3.9 啓動代碼

一般在 32 位開發平臺的應用程式中，絕大部分都採用 C 語言編寫，這樣可以大大提高開發效率、增加軟體可讀性和軟體可維護性。在系統初始化，通常會用一個組合語言檔案作為啟動代碼。

3.9.1 檔組成

啟動檔由多個檔組成，包括3.6.6 節講的中斷檔（當工程中要用到中斷時）、*_startup.s 檔案和*.ld 格式的檔，如圖 3-11。

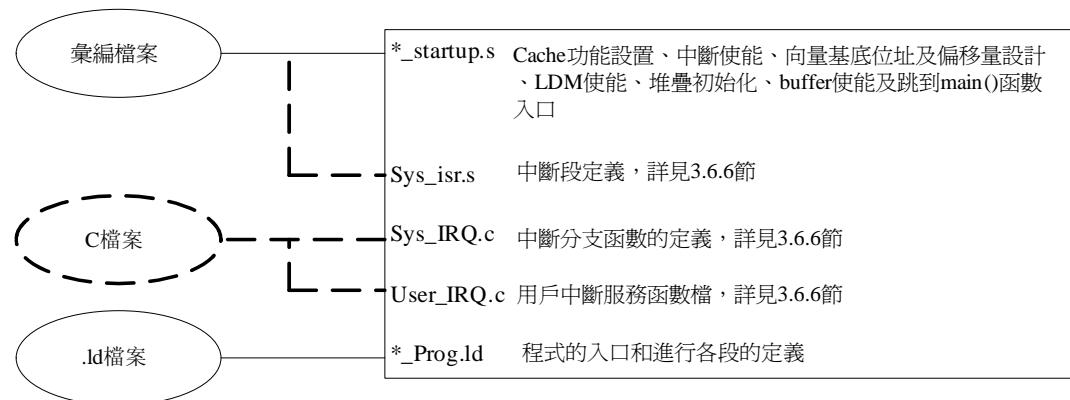


圖 3-11 啓動檔的組成

其中虛線指向的檔只有在中斷才 0 用到，可以看作是中斷的啟動檔，在3.6.6 節已詳細講述，不再贅述。所以，對於一般的工程來說，整 00 個啟動代碼由一個組合語言檔案(*_startup.s) 和一個.ld 檔案 (*_Prog.ld) 組成，“*”為工程名稱。

3.9.2 *_Prog.ld

*_Prog.ld 檔主要有兩大部分的內容：程式入口的定義和指定參加鏈接的區段及區段位址。

程式入口定義主要是指定程式運行的入口位址，利用 ENTRY()的格式來指定，括弧內指定入口的標號。*_Prog.ld 檔案中入口指定如下：

```

ENTRY(_hardware_init)           //指定程式從 hardware_init 標號處入口
                                // _hardware_init 在*_startup.s 檔中定義
    
```

指定參加鏈接的區段包括.text、.data、.bss 等外，還包括用戶定義的區段（例如.hardware_init 區段、.exception_vec 區段等），如果用戶自己利用.section 定義一個區段，要在*_Prog.ld 檔中聲明和控制分配位址。以下是兩個常用的區段的聲明及位址的定義。

堆疊區段：

```

.stack      0xa0fffffc :{ _stack = .; *(.stack) }   //定義堆疊區段基底位址為
                                                    //0xa0fffffc
    
```

中斷異常區段：

```
.exception 0xA00001fc :           // 中斷異常區段，基底位址為 0xA00001fc
{
    *(.exception_vec)          // 中斷區段名.exception_vec
} = 0
```

3.9.3 *_startup.s

*_startup.s 檔首先定義了一個區段.hardware_init（該區段也在*_Prog.Id 檔中進行了鏈接聲明），在.hardware_init 區段中定義了_hwadware_init 函數，且聲明_hwadware_init 函數為程式的入口。_hwadware_init 函數裏禁能了 cache 的寫回功能；使能了 S+core7 內核的 63 個硬體中斷和 2 個軟體中斷，同時指定了中斷向量基底位址和偏移量；使能 LDM，初始化堆疊，啟動 D-Cache 寫緩存功能後進入 main 函數執行用戶程式。

程式模組見圖 3-12。

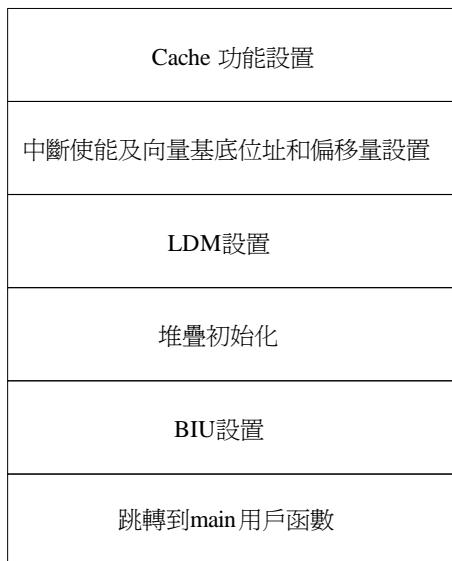


圖 3-12 *_startup.s 檔的程式功能模組示意圖

程式段如下：

```
.extern _start
.section .hardware_init,"ax",@progbits // 定義.hardware_init 區段

.global _hardware_init           // 定義.hardware_init 函數作為程式啟動入口，
                                // 從*.ld 檔中 ENTRY 進入到.hardware_init 函數
.ent _hardware_init
._hardware_init:
//=====
// 禁能 cache 的資料寫回
//=====
    mfcr r5, cr4           // 取出 Cache 控制暫存器 cr4 的資料存在 r5 中
    nop
    li r7, 0x80            // 把 0x80 裝載到 r7 暫存器
```

```
andri r6, r5, 0x80          //取 r5 (cr4) 的 b7 (cache 的寫回使能) 位，其他位置 0，  
                            //存在 r6 中  
cmp.c r7, r6                //Cache 控制暫存器 cr4 的寫回模式是否使能  
bne under_wt               //沒有使能，跳到 under_wt  
nop  
under_wb:                  //使能  
    la r7, tgl_wb            //將 tgl_wb 的位址裝載在 r7 中  
    cache 0x1f, [r7, 0]      //強制將一個被改寫過的有效 Cache 行中內容寫回主記憶體  
                            //並將全部 D-Cache 行改寫為無效  
    nop  
    nop  
    nop  
tgl_wb:                    //禁止寫回 D-Cache 功能  
    cache 0x1d, [r7, 0]  
    nop  
under_wt:                  //處理器進入寫入模式  
  
=====  
// 中斷使能  
=====  
    li r4, 0x1  
    mtcr r4, cr0            //使能所有 63 個硬體中斷和 2 個軟體中斷  
    li r4, 0xa0000000  
    mtcr r4, cr3            //指定異常向量基底位址為 0xa0000000，並指向量位址  
                            //的偏移模式為偏移 0x4  
  
=====  
// LDM 使能  
=====  
    li r5, 0xa1000000  
    cache 0xb, [r5, 0]        //指定 LDM 對映的起始位址  
    mfcr r11, cr4  
    ori r11, 0x8              //允許 LDM 介面  
    bitset.c r11,r11,16       //使能省電模式  
    mtcr r11, cr4  
    nop  
    nop  
  
=====  
// 堆疊基底位址指定  
=====  
    la r0,_stack             //_stack 標號位址在*.ld 檔中指定  
  
=====  
// 匯流排介面單元 (BIU) 設置  
=====  
    la r4, biu_wben          //裝載 biu_wben 位址  
biu_wben:  
    cache 0x1b, [r4, 0]        //啟動 D-Cache 寫緩存功能  
  
=====  
// 啓動執行用戶程式  
=====  
    j _start;                //跳越到_start 運行，該標號在庫中定義，  
                            //進入 main 函數執行用戶程式
```

```
.end _hardware_init //結束.hardware_init 區段
```

3.9.4 啓動代碼工作流程

SPCE3200 晶片在重設後，PC 暫存器指向 0x9F00 0000 位址，然後開始從 ENTRY 入口運行 _hardware_init，對 D-cache、LDM、異常、堆疊等模組進行初始化，初始化完成後進入 main 函數執行用戶程式。啟動代碼工作流程如圖 3-13。

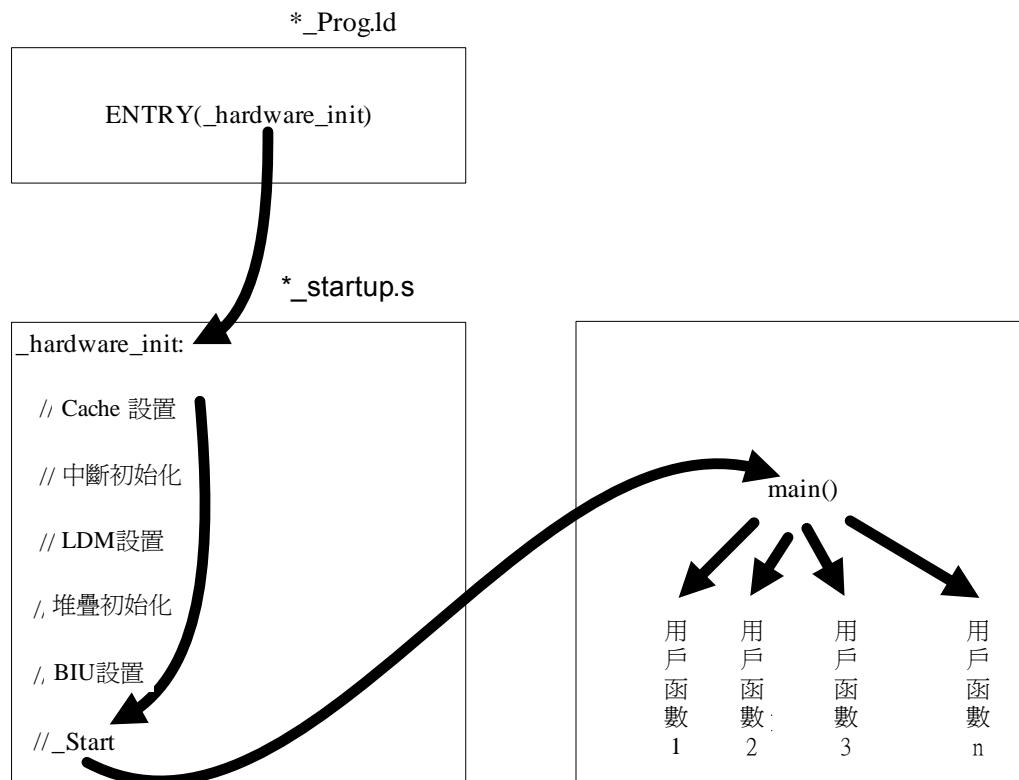


圖 3-13 啓動代碼工作流程圖

4 SPCE3200 功能部件

4.1 通用 I/O 埠——GPIO

4.1.1 概述

SPCE3200 最多擁有 79 個通用 I/O 埠 (GPIO—General Purpose I/O ports)，其中有 8 個 I/O 埠只做為 GPIO 使用，其他 I/O 埠是從其他模組複用為 GPIO 的，各個模組複用為 GPIO 埠的情況參考表 4-1：

表 4-1 各個模組複用為 GPIO 情況

模組 (Module)	可以複用為 GPIO 的數目
TFT	20
CSI	13
NFLASH	16
JTAG	5
DRAM	4
USB	1
UART	2
I2C	2
ADC	8

每個 GPIO 可以單獨編程設置為上拉電阻輸入、下拉電阻輸入、輸出等。SPCE3200 中的部分 GPIO 可以作為外部中斷埠，參考表 4-2：

表 4-2 GPIO 作為外部中斷

模組 (Module)	可以作為外部中斷的數目
TFT	4
CSI	4
NFLASH	4
JTAG	5
UART	2

模組 (Module)	可以作為外部中斷的數目
I2C	2
ADC	4
USB	1

SPCE3200 的 8 個專門用作 GPIO 的埠分為 2 組，一組叫做 IOA 埠，有 IOA0、IOA1 兩個埠，可以作為外部中斷；另一組叫做 IOB 埠，有 IOB0、IOB1、IOB2、IOB3、IOB4、IOB5 六個埠，不可以作為外部中斷。IOA 埠與 IOB 埠都可以編程設置為上拉輸入、下拉輸入、輸出埠等。本節主要介紹 IOA 埠與 IOB 埠，其他模組複用的 GPIO 參考相應模組章節有詳細介紹。表 4-3 為可以作為 GPIO 所有埠的資訊列表：

表 4-3 GPIO 信息表

序號	模組 (Module)	插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
1	TFT	LCD_CLK	148	bit[0]	可以
2		LCD_VS	147	bit[1]	可以
3		LCD_HS	146	bit[2]	可以
4		LCD_D0	144	bit[3]	可以
5		LCD_D1	143	bit[4]	不可以
6		LCD_D2	142	bit[5]	不可以
7		LCD_D3	141	bit[6]	不可以
8		LCD_D4	140	bit[7]	不可以
9		LCD_D5	139	bit[8]	不可以
10		LCD_D6	138	bit[9]	不可以
11		LCD_D7	137	bit[10]	不可以
12		LCD_D8	136	bit[11]	不可以
13		LCD_D9	135	bit[12]	不可以
14		LCD_D10	134	bit[13]	不可以
15		LCD_D11	133	bit[14]	不可以
16		LCD_D12	132	bit[15]	不可以
17		LCD_D13	131	bit[16]	不可以
18		LCD_D14	130	bit[17]	不可以

序號	模組 (Module)	插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
19	CSI	LCD_D15	129	bit[18]	不可以
20		LCD_ACT	145	bit[19]	不可以
21		CSI_D0	16	bit[0]	可以
22		CSI_D1	17	bit[1]	可以
23		CSI_D2	18	bit[2]	可以
24		CSI_D3	19	bit[3]	可以
25		CSI_D4	20	bit[4]	不可以
26		CSI_D5	21	bit[5]	不可以
27		CSI_D6	24	bit[6]	不可以
28		CSI_D7	25	bit[7]	不可以
29		CSI_CKO	26	bit[8]	不可以
30		CSI_CKI	27	bit[9]	不可以
31		CSI_HS	28	bit[10]	不可以
32		CSI_VS	29	bit[11]	不可以
33		CSI_FIELD	30	bit[12]	不可以
34	NFLASH	NF_ALE	127	bit[0]	可以
35		NF_WPN	126	bit[1]	可以
36		NF_CLE	125	bit[2]	可以
37		NF_REN	124	bit[3]	可以
38		NF_WEN	123	bit[4]	不可以
39		NF_CEN	119	bit[5]	不可以
40		NF_RDY	118	bit[6]	不可以
41		NF_D0	117	bit[7]	不可以
42		NF_D1	116	bit[8]	不可以
43		NF_D2	115	bit[9]	不可以
44		NF_D3	112	bit[10]	不可以
45		NF_D4	111	bit[11]	不可以
46		NF_D5	110	bit[12]	不可以

序號	模組 (Module)	插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
47		NF_D6	109	bit[13]	不可以
48		NF_D7	108	bit[14]	不可以
49		SPI_CSN	128	bit[15]	不可以
50	JTAG	JTAG_TRSTN	153	bit[0]	可以
51		JTAG_TDI	152	bit[1]	可以
52		JTAG_TDO	151	bit[2]	可以
53		JTAG_TCK	150	bit[3]	可以
54		JTAG_TMS	149	bit[4]	可以
55	DRAM	ROMCSN	15	bit[0]	不可以
56		DRAM_A11	239	bit[1]	不可以
57		DRAM_A12	14	bit[2]	不可以
58		DRAM_BA1	254	bit[3]	不可以
59	USB	USB_DET	156	bit[0]	可以
60	UART	UART_RX	40	bit[0]	可以
61		UART_TX	39	bit[1]	可以
62	I2C	I2C_CLK	35	bit[0]	可以
63		I2C_DATA	36	bit[1]	可以
64	ADC	ADC_CH0	160	bit[0]	可以
65		ADC_CH1	161	bit[1]	可以
66		ADC_CH2	162	bit[2]	可以
67		ADC_CH3	163	bit[3]	可以
68		ADC_CH4	164	bit[4]	不可以
69		ADC_CH5	165	bit[5]	不可以
70		ADC_CH6	166	bit[6]	不可以
71		ADC_CH7	167	bit[7]	不可以
72	IOA	IOA0	41	bit[0]	可以
73		IOA1	42	bit[1]	可以
74	IOB	IOB0	101	bit[0]	不可以

序號	模組 (Module)	插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
75		IOB1	100	bit[1]	不可以
76		IOB2	99	bit[2]	不可以
77		IOB3	98	bit[3]	不可以
78		IOB4	97	bit[4]	不可以
79		IOB5	96	bit[5]	不可以

4.1.2 插腳描述

SPCE3200 的 8 個 GPIO 插腳描述如表 4-4 所示：

表 4-4 GPIO 插腳描述

插腳名稱	插腳號	插腳屬性	插腳功能
IOA0	41	I/O	GPIO
IOA1	42	I/O	GPIO
IOB5	96	I/O	GPIO
IOB4	97	I/O	GPIO
IOB3	98	I/O	GPIO
IOB2	99	I/O	GPIO
IOB1	100	I/O	GPIO
IOB0	101	I/O	GPIO

4.1.3 結構

SPCE3200 的 IOA 埠的結構如圖 4-1 所示：可以透過暫存器 P_IOA_GPIO_SETUP 設置 IOA 埠的上拉輸入、下拉輸入或輸出埠；透過暫存器 P_IOA_GPIO_INPUT 讀入埠的值；透過設置 P_IOA_GPIO_INT 設置 IOA 埠作為外部中斷。

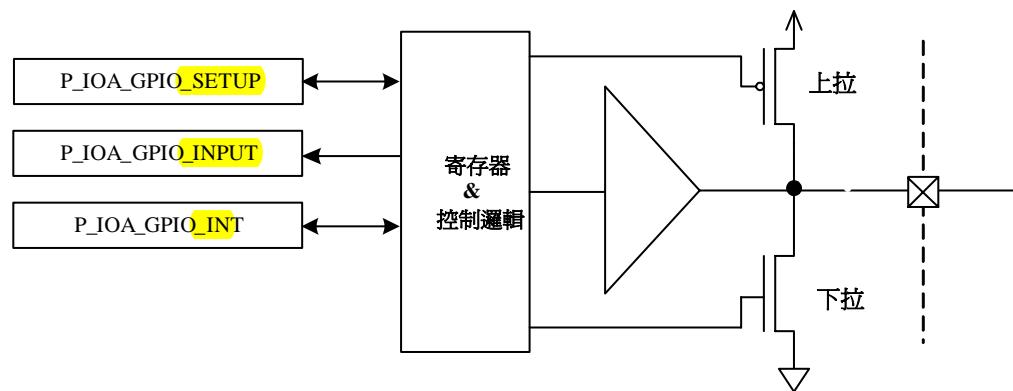


圖 4-1 IOA 的結構圖

SPCE3200 的 IOB 埠的結構圖如圖 4-2 所示：可以透過暫存器 P_IOB_GPIO_SETUP 設置 IOB 埠的上拉輸入、下拉輸入或輸出埠；透過暫存器 P_IOB_GPIO_INPUT 讀出埠的值。

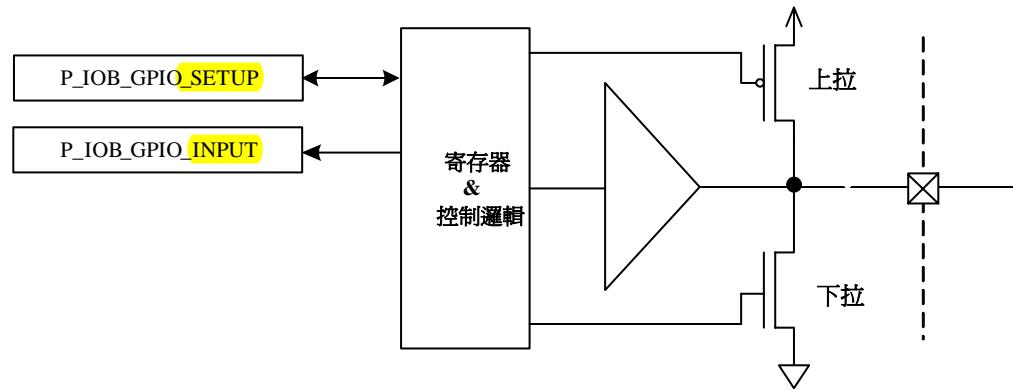


圖 4-2 IOB 的結構圖

4.1.4 暫存器描述

SPCE3200 的 8 個 GPIO 相關的暫存器有 6 個，參考表 4-5：

表 4-5 GPIO 相關暫存器描述

暫存器名稱	助記符	位址
GPIO 時鐘配置暫存器	P_GPIO_CLK_CONF	0x882100FC
IOA 設置暫存器	P_IOA_GPIO_SETUP	0x88200038
IOA 輸入資料暫存器	P_IOA_GPIO_INPUT	0x88200074
IOA 埠外部中斷暫存器	P_IOA_GPIO_INT	0x88200090
IOB 埠設置暫存器	P_IOB_GPIO_SETUP	0x8820004C
IOB 埠輸入資料暫存器	P_IOB_GPIO_INPUT	0x88200070

■ GPIO 時鐘配置暫存器：**P_GPIO_CLK_CONF(0x882100FC)**

設置該暫存器可以使能 GPIO 模組，包括其他模組複用為 GPIO 時的情況，該暫存器預設就是使能的。

表 4-6 P_GPIO_CLK_CONF(0x882100FC)

位	b31~b26	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	1
名稱	-	SFTCFG_RST	SFTCFG_STOP

SFTCFG_RST	b1	GPIO 模組時鐘重設位： 0: GPIO 模組時鐘重設 1: GPIO 模組時鐘不重設
SFTCFG_STOP	b0	GPIO 模組時鐘使能位： 0 : GPIO 模組時鐘停止 1 : GPIO 模組時鐘使能

■ IOA 設置暫存器：**P_IOA_GPIO_SETUP(0x88200038)**

IOA0、IOA1 的埠設置，當需要設置 IOA0、IOA1 為上拉電阻輸入、下拉電阻輸入或輸出使能時需要設置該暫存器，當設置 IOA0、IOA1 為輸出使能時，可以輸出高電平或低電平。注意 IOA 的上拉電阻輸入狀態 0 為使能狀態，1 為禁止狀態。預設為下拉電阻輸入狀態。

表 4-7 P_IOA_GPIO_SETUP(0x88200038)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	1	-	1	-	0	-	0
名稱	-	IOA_PD	-	IOA_PU	-	IOA_OE	-	IOA_O

IOA1_PD	b25	IOA1 的下拉電阻輸入使能位： 0: 禁止下拉電阻輸入 1: 使能下拉電阻輸入
---------	-----	--

IOA0_PD	b24	IOA0 的下拉電阻輸入使能位： 0：禁止下拉電阻輸入 1：使能下拉電阻輸入
IOA1_PU	b17	IOA1 的上拉電阻輸入使能位： 0：使能上拉電阻輸入 1：禁止上拉電阻輸入
IOA0_PU	b16	IOA0 的上拉電阻輸入使能位： 0：使能上拉電阻輸入 1：禁止上拉電阻輸入
IOA1_OE	b9	IOA1 的輸出使能位： 0：禁止輸出 1：使能輸出
IOA0_OE	b8	IOA0 的輸出使能位： 0：禁止輸出 1：使能輸出
IOA1_O	b1	IOA1 的輸出資料
IOA0_O	b0	IOA0 的輸出資料

■ IOA 輸入資料暫存器：P_IOA_GPIO_INPUT(0x88200074)

讀該暫存器可以得到 IOA0~IOA1 的埠輸入資料。

表 4-8 P_IOA_GPIO_INPUT(0x88200074)

位	b31~b2	b1~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	IOA_INPUT

GLOBAL_INPUT	b1~b0	IOA1~IOA0 的埠輸入資料
--------------	-------	------------------

■ IOA 外部中斷暫存器：P_IOA_GPIO_INT(0x88200090)

該暫存器為 IOA0~IOA1 的外部中斷設置暫存器。

表 4-9 P_IOA_GPIO_INT(0x88200090)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	0	-	0	-	0	-	0
名稱	-	IOA_FI	-	IOA_RI	-	IOA_FIEN	-	IOA_RIEN

IOA_FI b25~b24 IOA1~IOA0 埠下降沿中斷旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

IOA_RI b17~b16 IOA1~IOA0 埠上升沿中斷旗標位：

讀 0：沒有發生上升沿中斷

讀 1：發生了上升沿中斷

寫 0：無意義

寫 1：清除中斷旗標

IOA_FIEN b9~b8 IOA1~IOA0 埠下降沿中斷使能位：

0：禁止下降沿中斷

1：使能下降沿中斷

IOA_RIEN b1~b0 IOA1~IOA0 埠上升沿中斷使能位：

0：禁止上升沿中斷

1：使能上升沿中斷

■ IOB 設置暫存器：P_IOB_GPIO_SETUP(0x8820004C)

IOB0~IOB5 的埠設置，當需要設置 IOB0~IOB5 為上拉電阻輸入、下拉電阻輸入或輸出使能時需要設置該暫存器，當設置 IOB0~IOB5 為輸出使能時，可以輸出高電平或低電平。

表 4-10 P_IOB_GPIO_SETUP(0x8820004C)

位	b31~b30	b29~b24	b23~b22	b21~b16	b15~b14	b13~b8	b7~b6	b5~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	1	-	1	-	0	-	0
名稱	-	IOB_PD	-	IOB_PU	-	IOB_OE	-	IOB_O

IOB_PD

b29~b24

IOB5~IOB0 的下拉電阻輸入使能位：

0：禁止下拉電阻輸入

1：使能下拉電阻輸入

IOB_PU

b21~b16

IOB5~IOB0 的上拉電阻輸入使能位：

0：使能上拉電阻輸入

1：禁止上拉電阻輸入

IOB_OE

b13~b8

IOB5~IOB0 的輸出使能位：

0：禁止輸出

1：使能輸出

IOB_O

b5~b0

IOB5~IOB0 的輸出資料

■ IOB 輸入資料暫存器：P_IOB_GPIO_INPUT(0x88200070)

讀該暫存器可以得到 IOB0~IOB5 的輸入資料。

表 4-11 P_IOB_GPIO_INPUT(0x88200070)

位	b31~b14	b13~b8	b7~b0
讀/寫	-	R/W	-
預設值	-	0	-
名稱	-	IOB_INPUT	-

IOB_INPUT

b13~b8

IOB5~IOB0 的輸入資料

4.1.5 基本操作

設置 IOA0、IOA1 為輸入埠，讀取 IOA0、IOA1 的輸入資料，參考程式段如下：

```
unsigned int a; // 定義一個變數  
*P_IOA_GPIO_SETUP = 0x03030000; // 設置 IOA0、IOA1 為輸入埠  
a = *P_IOA_GPIO_INPUT; // 讀取 IOA0、IOA1 的埠值  
a &= 0x00000003; // 得到 IOA0、IOA1 的值
```

設置 IOA0、IOA1 為輸出使能，輸出高電平，參考程式段如下：

```
*P_IOA_GPIO_SETUP = 0x00000300; // IOA0、IOA1 輸出使能  
*P_IOA_GPIO_SETUP |= 0x00000003; // IOA0、IOA1 輸出高電平
```

設置 IOA1 為上升沿外部中斷，查詢該中斷發生，參考程式段如下：

```
*P_INT_MASK_CTRL2 &= ~C_INT_GPIO_DIS; // 使能 IRQ 中斷  
  
*P_IOA_GPIO_INT = C_IOA1_INTRISE_EN // 使能 IOA1 上升沿外部中斷  
| C_IOA1_INTRISE_FLAG;  
while(1)  
{  
    if(*P_IOA_GPIO_INT & C_IOA1_INTRISE_FLAG) // 判斷中斷是否發生  
    {  
        *P_IOA_GPIO_INT |= C_IOA1_INTRISE_FLAG; // 清除中斷旗標位元  
    }  
}
```

4.2 計時器——TIMER

4.2.1 概述

SPCE3200 內嵌 6 個 16 位 CCP (Compare、Capture、PWM，比較、擷取、PWM 輸出) 計時器：Timer0、Timer1、Timer2、Timer3、Timer4 和 Timer5。6 個計時器的功能結構完全相同，2 個時鐘源可供選擇 27MHz/(M+1) 或 32768Hz 即時時鐘。

4.2.2 特性

Timer0~Timer5 這 6 個計時器均可以編程設置工作方式及時鐘源。可以編程設置為 normal 模式，即定時計數模式；CMP 模式，即比較模式；CAP 模式，即擷取模式；PWM 模式。每個計時器也可以單獨設置時鐘源，選擇時鐘源 27MHz/ (M+1) 或 32768Hz 即時時鐘。

4.2.3 插腳描述

當計時器工作在比較輸出模式、擷取輸入模式和 PWM 輸出模式時，均需要透過晶片管腳輸入、輸出資料，表 4-12列出了這些管腳：

表 4-12 計時器 CCP 管腳描述

插腳名稱	插腳號	插腳屬性	插腳功能
USBDET	156	I/O	作為 Timer_CCP0 的輸入輸出
UART_RX	40	I/O	作為 Timer_CCP1 的輸入輸出
UART_TX	39	I/O	作為 Timer_CCP2 的輸入輸出
JTAG_TRSTN	153	I/O	作為 Timer_CCP3 的輸入輸出
IOA0	41	I/O	作為 Timer_CCP4 的輸入輸出
IOA1	42	I/O	作為 Timer_CCP5 的輸入輸出

4.2.4 結構

計時器 Timer 的結構如圖 4-3 所示：由於 6 個計時器的結構完全相同，以 Timer0 介紹計時器的結構。時鐘源可以透過編程選擇其一，16 位計數器在選擇的頻率下從計數初值計數，控制邏輯控制計時器 Timer 的使能、中斷使能等，當工作在 CCP 模式下，CCP 暫存器單元作為計算 PWM 占空比、比較輸出或擷取輸入的存儲單元。

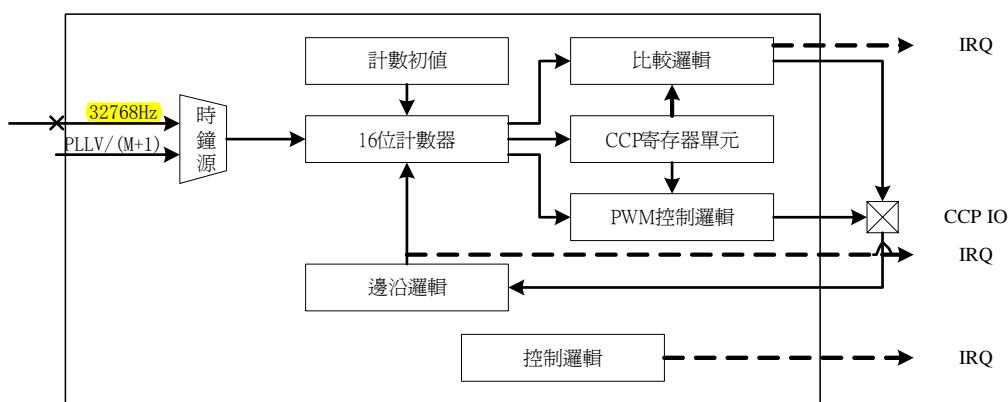


圖 4-3 Timer 的結構

4.2.5 暫存器描述

由於 6 個計時器的結構、使用方法完全相同，這裏以 Timer0 為例介紹。共有 9 個暫存器單元與計時器 Timer0 有關，參考表 4-13：

表 4-13 Timer0 相關的暫存器

暫存器中文名稱	暫存器英文名稱	位址
TIMER 時鐘選擇暫存器	P_TIMER_CLK_SEL	0x882100E4
TIMER 介面選擇暫存器	P_TIMER_INTERFACE_SEL	0x88200010

暫存器中文名稱	暫存器英文名稱	位址
TIMER0 時鐘配置暫存器	P_TIMER0_CLK_CONF	0x8821006C
TIMER0 控制暫存器	P_TIMER0_MODE_CTRL	0x88160000
TIMER0 CCP 控制暫存器	P_TIMER0_CCP_CTRL	0x88160004
TIMER0 計數初值資料暫存器	P_TIMER0_PRELOAD_DATA	0x88160008
TIMER0 計數暫存器	P_TIMER0_COUNT_DATA	0x88160010
TIMER0 CCP 計數初值資料暫存器	P_TIMER0_CCP_DATA	0x8816000C

其中 P_TIMER_CLK_SEL、P_TIMER_INTERFACE_SEL 為 Timer0~Timer5 共用。下麵分別介紹：

■ **TIMER 時鐘選擇暫存器：P_TIMER_CLK_SEL(0x882100E4)**

計時器時鐘源選擇暫存器，選擇計時器使用 27MHz (M+1) 時鐘源或者 32768Hz 時鐘源，當選擇 27MHz (M+1) 時鐘源時，需要透過該暫存器的最低 8 位確定 M 值。當選擇 32768Hz 時鐘源時，需要使能 P_CLK_32K_CONF(0x88210114)暫存器。

表 4-14 P_TIMER_CLK_SEL(0x882100E4)

位	b13	b12	b11	b10	b9	b8	b7~b0
讀/寫	R/W						
預設值	0	0	0	0	0	0	0
名稱	Timer5C	Timer4C	Timer3C	Timer2C	Timer1C	Timer0C	DividedNum

Timer5C	b13	Timer5 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源
Timer4C	b12	Timer4 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源
Timer3C	b11	Timer3 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源

Timer2C	b10	Timer2 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源
Timer1C	b9	Timer1 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源
Timer0C	b8	Timer0 時鐘源選擇位： 0：選擇 27MHz 作為時鐘源 1：選擇 32768Hz 作為時鐘源
DividedNum	b7~b0	時鐘源採用 $27MHz / (\text{DividedNum} + 1)$ ，如果選擇時鐘源 27MHz

■ **TIMER 介面選擇暫存器：P_TIMER_INTERFACE_SEL(0x88200010)**

當計時器工作在 CCP 模式下，需要設置該暫存器使 I/O 埠作為 CCP 埠使用。

表 4-15 P_TIMER_INTERFACE_SEL(0x88200010)

位	b31~b22	b21	b20	b19	b18	b17	b16	b15~b0
讀/寫	-	R/W	R/W	R/W	R/W	R/W	R/W	-
預設值	-	0	0	0	0	0	0	-
名稱	-	TCCP5	TCCP4	TCCP3	TCCP2	TCCP1	TCCP0	-

TCCP5	b21	IOA1 作為 CCP5 埠使能位： 0: 禁止為 CCP5 埠 1: 使能為 CCP5 埠
TCCP4	b20	IOA0 作為 CCP4 埠使能位： 0: 禁止為 CCP4 埠 1: 使能為 CCP4 埠
TCCP3	b19	JTAG_TRSTN 作為 CCP3 埠使能位： 0: 禁止為 CCP3 埠 1: 使能為 CCP3 埠

TCCP2 b18 UART_TX 作為 CCP2 埠使能位：

0: 禁止為 CCP2 埠

1: 使能為 CCP2 埠

TCCP1 b17 UART_RX 作為 CCP1 埠使能位：

0: 禁止為 CCP1 埠

1: 使能為 CCP1 埠

TCCP0 b16 USBDET 作為 CCP0 埠使能位：

0: 禁止為 CCP0 埠

1: 使能為 CCP0 埠

■ TIMER0 時鐘配置暫存器 P_TIMER0_CLK_CONF(0x8821006C)

配置 Timer0 模組的時鐘，當需要使用計時器 Timer0 模組時，需要配置此暫存器。

表 4-16 P_TIMER0_CLK_CONF(0x8821006C)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	TIMER0_RST	TIMER0_STOP

TIMER0_RST b1 Timer0 模組時鐘重設位：

0 : Timer0 模組時鐘重設

1 : Timer0 模組時鐘不重設

TIMER0_STOP b0 Timer0 模組時鐘停止位：

0 : Timer0 模組時鐘停止

1 : Timer0 模組時鐘使能

其他計時器的配置暫存器，參考表 4-17：各位功能與 P_TIMER0_CLK_CONF 相同。

表 4-17 計時器時鐘配置暫存器

計時器 X	暫存器助記符	位址
計時器 0	P_TIMER0_CLK_CONF	0x8821006C
計時器 1	P_TIMER1_CLK_CONF	0x88210070
計時器 2	P_TIMER2_CLK_CONF	0x88210074
計時器 3	P_TIMER3_CLK_CONF	0x88210078
計時器 4	P_TIMER4_CLK_CONF	0x8821007C
計時器 5	P_TIMER5_CLK_CONF	0x88210080

■ TIMER0 控制暫存器：P_TIMER0_MODE_CTRL(0x88160000)

計時器模組 Timer0 的控制暫存器，使能 Timer0，使能 Timer0 中斷及清除中斷旗標位元。

表 4-18 P_TIMER0_MODE_CTRL(0x88160000)

位	b31	b30~b28	b27	b26	b25~b0
讀/寫	R/W	-	R/W	R/W	-
預設值	0	-	0	0	-
名稱	TIMER0_EN	-	TIMER0_IRQ_EN	TIMER0_IRQ_FLAG	-

TIMER0_EN	b31	Timer0 使能位： 0 : 禁止 Timer0 1 : 使能 Timer0
TIMER0_IRQ_EN	b27	Timer0 中斷使能位： 0 : 禁止 Timer0 中斷 1 : 使能 Timer0 中斷
TIMER0_IRQ_FLAG	b26	Timer0 中斷旗標位： 讀 0 : 沒有發生 Timer0 中斷 讀 1 : 發生 Timer0 中斷 寫 0 : 無意義 寫 1 : 清除中斷旗標

其他計時器的控制暫存器參考表 4-19：各位功能與 P_TIMER0_MODE_CTRL 相同。

表 4-19 計時器控制暫存器

計時器 X	暫存器助記符	位址
計時器 0	P_TIMER0_MODE_CTRL	0x88160000
計時器 1	P_TIMER1_MODE_CTRL	0x88161000
計時器 2	P_TIMER2_MODE_CTRL	0x88162000
計時器 3	P_TIMER3_MODE_CTRL	0x88163000
計時器 4	P_TIMER4_MODE_CTRL	0x88164000
計時器 5	P_TIMER5_MODE_CTRL	0x88165000

■ TIMER0 CCP 控制暫存器：P_TIMER0_CCP_CTRL(0x88160004)

該暫存器可以設置計時器 0 的工作方式及在 CCP 工作模式下的選項。

表 4-20 P_TIMER0_CCP_CTRL(0x88160004)

位	b31~b30	b29~b28	b27	b26	b25	b24~b0
讀/寫	R/W	-	R/W	R/W	R/W	R/W
預設值	0	-	0	0	0	0
名稱	CCP_MODE	-	CAP_MODE	COM_MODE	PWM_MODE	-

CCP MODE

b31~b30

Timer0 工作方式選擇位：

00 : 計時器模式

01 : Capture 擷取輸入模式

10 : Compare 比較輸出模式

11 : PWM 輸出模式

CAP MODE

b27

CAP 輸入模式選擇位：

0：下降沿觸發

1：上升沿觸發

COM_MODE	b26	COM 輸出模式選擇位： 0：比較匹配後輸出高脈衝 1：比較匹配後輸出低脈衝
PWM_MODE	b25	PWM 輸出模式選擇位： 0：NRO (Non-Return-One) 輸出模式 1：NRZ (Non-Return-Zero) 輸出模式

其他計時器的 CCP 控制暫存器如表 4-21：各位功能與 P_TIMER0_CCP_CTRL 相同。

表 4-21 計時器 CCP 控制暫存器

計時器 X	暫存器助記符	位址
計時器 0	P_TIMER0_CCP_CTRL	0x88160004
計時器 1	P_TIMER1_CCP_CTRL	0x88161004
計時器 2	P_TIMER2_CCP_CTRL	0x88162004
計時器 3	P_TIMER3_CCP_CTRL	0x88163004
計時器 4	P_TIMER4_CCP_CTRL	0x88164004
計時器 5	P_TIMER5_CCP_CTRL	0x88165004

■ TIMER0 計數初值資料暫存器：P_TIMER0_PRELOAD_DATA(0x88160008)

計時器的計數初值資料暫存器，16 位計數器將以該值計數，當溢出後會將該值重載入 16 位計數器。

表 4-22 P_TIMER0_PRELOAD_DATA(0x88160008)

位	b31~b16	b15~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TIMER0_PRELOAD_REGS

其他計時器的計數初值暫存器如表 4-23：各位功能與 P_TIMER0_PRELOAD_DATA 相同。

表 4-23 計時器計數初值資料暫存器

計時器 X	暫存器助記符	位址
計時器 0	P_TIMER0_PRELOAD_DATA	0x88160008
計時器 1	P_TIMER1_PRELOAD_DATA	0x88161008
計時器 2	P_TIMER2_PRELOAD_DATA	0x88162008
計時器 3	P_TIMER3_PRELOAD_DATA	0x88163008
計時器 4	P_TIMER4_PRELOAD_DATA	0x88164008
計時器 5	P_TIMER5_PRELOAD_DATA	0x88165008

■ **TIMER0 計數暫存器 : P_TIMER0_COUNT_DATA(0x88160010)**

計時器 0 的計數器，當計時器工作在定時/計數模式下時，可以讀該暫存器得到當前計數值。

表 4-24 P_TIMER0_COUNT_DATA(0x88160010)

位	b31~b16	b15~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TIMER0_UPCOUNT

其他計時器的計數器暫存器如表 4-25：各位功能與 P_TIMER0_COUNT_DATA 相同。

表 4-25 計時器計數暫存器

計時器 X	計時器助記符	位址
計時器 0	P_TIMER0_COUNT_DATA	0x88160010
計時器 1	P_TIMER1_COUNT_DATA	0x88161010
計時器 2	P_TIMER2_COUNT_DATA	0x88162010
計時器 3	P_TIMER3_COUNT_DATA	0x88163010
計時器 4	P_TIMER4_COUNT_DATA	0x88164010
計時器 5	P_TIMER5_COUNT_DATA	0x88165010

■ **TIMER0 CCP 計數初值資料暫存器：P_TIMER0_CCP_DATA(0x8816000C)**

當計時器 0 工作在 **Compare** 比較模式下，該暫存器的值與 16 位 **計數器** 的值作比較，當相等時輸出高/低脈衝同時產生 **IRQ 中斷**；當計時器 0 工作在 **Capture** 擷取輸入模式下，當邊沿邏輯擷取到週期信號後，將值保存到該暫存器，**同時產生 IRQ 中斷**；當計時器 0 工作在 **PWM** 模式下，**該暫存器與占空比有關**。

表 4-26 P_TIMER0_CCP_DATA(0x8816000C)

位	b31~b16	b15~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TIMER0_CCP_REGS

其他計時器的 CCP 計數初值資料暫存器參考：各位功能與 P_TIMER0_CCP_DATA 相同。

表 4-27 計時器計數初值資料暫存器

計時器 X	計時器助記符	位址
計時器 0	P_TIMER0_CCP_DATA	0x8816000C
計時器 1	P_TIMER1_CCP_DATA	0x8816100C
計時器 2	P_TIMER2_CCP_DATA	0x8816200C
計時器 3	P_TIMER3_CCP_DATA	0x8816300C
計時器 4	P_TIMER4_CCP_DATA	0x8816400C
計時器 5	P_TIMER5_CCP_DATA	0x8816500C

4.2.6 基本操作

1. 定時/計數工作模式

Timer 用於定時/計數模式時的結構簡化為如圖 4-4 所示：

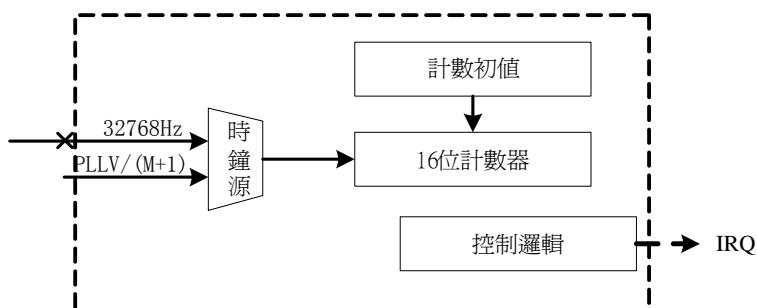


圖 4-4 計時器 Timer 工作在定時/計數模式

Timer0~5 工作在定時/計數模式下的時序如圖 4-5 所示：

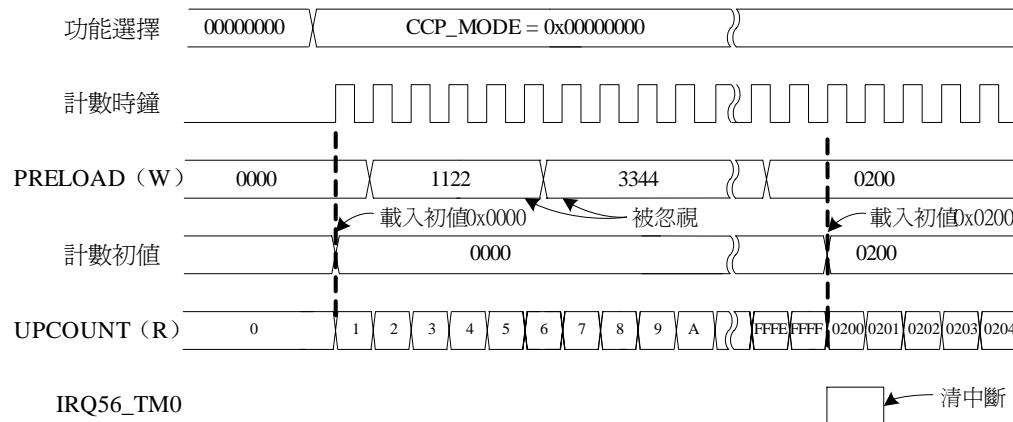


圖 4-5 Timer0~5 定時/計數時序圖

向 P_TIMER0_CC_P_CTRL 暫存器的 b31:b30 中寫入 00，計時器 0 將工作在定時/計數模式下。在使能計時器後，在下一個有效計數時鐘沿（上升沿），計數初值將會被載入，之後在計數時鐘作用下，計數器值遞增，直至計數器溢出，將重載預置的計數初值，開始新一輪的計數。在計時器溢出之前，無法重新裝載初值。

Timer 用於定時，只需選定適當的計數時鐘源和計數初值即可。定時時間 (T) 與計數時鐘頻率 (f)、計數初值 (N) 的關係如下：

$$T = (65536 - N)/f$$

更多的情況是已知定時時間 T 和計數時鐘頻率 f，則計數初值 N 可以由下式算得：

$$N = 65536 - T * f$$

定時/計數器大多是與中斷配合使用的。計時器的溢出信號可作為中斷源，計時器作為定時/計數器的初始化流程如圖 4-6 所示：

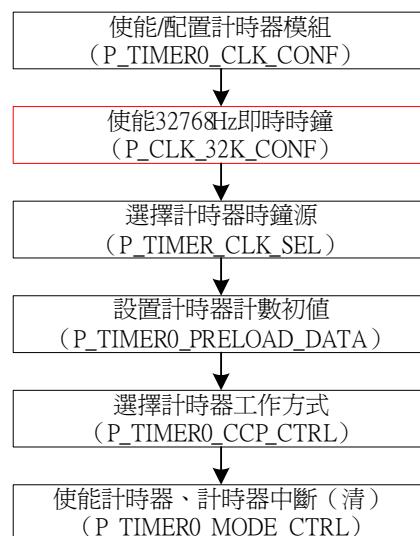
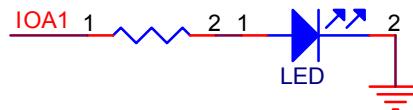


圖 4-6 Timer0~5 定時/計數初始化流程

【例 4.1】

利用 Timer0 實現 0.5s 定時，並控制 IOA1 外接的 LED，使該 LED 閃爍，硬體原理如下圖：



實現 0.5s 延時可以透過多種時鐘源實現，這裏採用 32768Hz，可計算得到計數初值為：

$$N = 65536 - 0.5 * 32768 = 49152(0xC000)$$

參考代碼：

```
#include "SPC3200_Register.h"
#include "SPC3200_Constant.h"

int main(void)
{
    *P_TIMER0_CLK_CONF = C_TIMER_CLK_EN
        | C_TIMER_RST_DIS; // 使能 TIMER0 模組時鐘
    *P_CLK_32K_CONF = C_32K_CRY_EN; // 使能 32K 晶振
    *P_TIMER_CLK_SEL = C_TIMER0_CLK_32K; // 選擇 32K 時鐘
    *P_TIMER0_PRELOAD_DATA = 0xC000; // 計數初值選擇 0.5s
    *P_TIMER0_CCP_CTRL = C_TIMER_NOR_MODE; // 工作模式選擇定時計數模式
    *P_TIMER0_MODE_CTRL = C_TIMER_CTRL_EN
        | C_TIMER_INT_EN // 使能計時器
        | C_TIMER_INT_FLAG; // 使能中斷
    *P_IOA_GPIO_SETUP = 0x00000300; // 清中斷
    // 使能 IOA 埠為輸出埠

    while(1)
    {
        if(*P_TIMER0_MODE_CTRL & C_TIMER_INT_FLAG)
        {
            *P_IOA_GPIO_SETUP ^= 0x00000003;
        }
        *P_TIMER0_MODE_CTRL |= C_TIMER_INT_FLAG; // 清中斷
    }
    return 0;
}
```

2. 比較輸出模式 (Compare)

Timer0~5 用作比較輸出的內部結構如圖 4-7 所示：

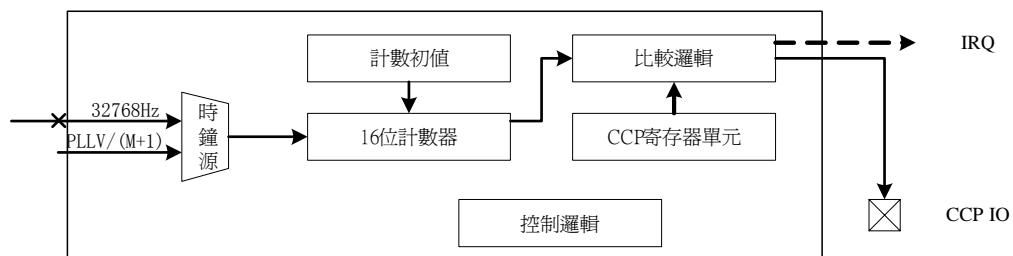


圖 4-7 比較輸出結構圖

Timer0~5 在比較模式下的工作時序如圖 4-8 所示：

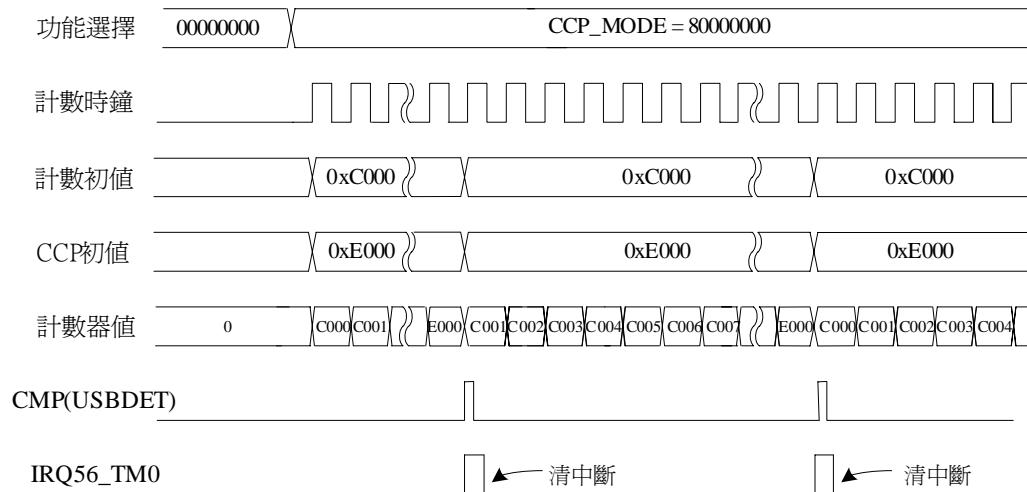


圖 4-8 Timer0~5 比較輸出時序圖

Timer0~5 用作比較輸出模式包括以下操作：配置計時器模組、選擇計時器時鐘源、設置計時器計數初值及 CCP 初值、使能 I/O 作為 CCP 埠使用、選擇計時器工作方式為比較輸出方式、使能計時器及中斷。其操作流程如圖 4-9：

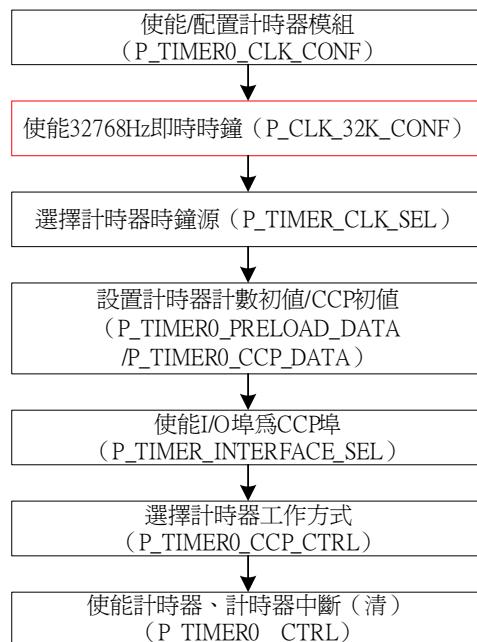


圖 4-9 Timer0~5 比較輸出初始化流程圖

透過設置 P_TIMER0_CCP_CTRL 暫存器的 b26 位可以選擇當比較匹配是輸出高脈衝還是低脈衝。b26 為 0 輸出高脈衝；b26 為 1 輸出低脈衝。

【例 4.2】

設置比較輸出，使用計時器 4 來完成。

參考代碼：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    *P_INT_MASK_CTRL1 = ~C_INT_TIMER_DIS; // 使能 TIMER 中斷
    *P_CLK_32K_CONF = C_32K_CRY_EN; // 使能 32K 晶振
    *P_TIMER4_CLK_CONF = 0x00000000; // 重設 TIMER4 模組
    *P_TIMER4_CLK_CONF = C_TIMER_CLK_EN | C_TIMER_RST_DIS; // 使能 TIMER4 模組
    *P_TIMER_CLK_SEL = C_TIMER4_CLK_32K; // 選擇 32K 時鐘源
    *P_TIMER4_PRELOAD_DATA = 0xF200; // 設置計數初值
    *P_TIMER4_CCP_DATA = 0xF700; // 設置 CCP 初值
    *P_TIMER_INTERFACE_SEL = C_TIMER4_PORT_SEL; // 選擇 CCP 埠
    *P_TIMER4_CCP_CTRL = C_TIMER_CMP_MODE
        | C_TIMER_CMP_HIGH; // 選擇 CMP 工作模式
    *P_TIMER4_MODE_CTRL = C_TIMER_CTRL_EN
        | C_TIMER_INT_EN // 使能中斷
        | C_TIMER_INT_FLAG; // 清中斷

    while(1);
    return 0;
}

// 中斷檔中函數
//=====
// 語法格式：void IRQ56(void)
// 功能描述：計時器（Timer）中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ56(void)
{
    if(*P_TIMER4_MODE_CTRL & C_TIMER_INT_FLAG)
    {
        *P_TIMER4_MODE_CTRL |= C_TIMER_INT_FLAG;
    }
}
```

3. 摷取輸入模式 (Capture)

Timer0~5 用作摷取輸入模式 (Capture) 的結構如圖 4-10所示：當外部信號滿足邊沿邏輯的要求後，觸發邊沿邏輯，邊沿邏輯控制 16 位計數器將摷取到的週期/頻率信號輸入 CCP 暫存器單元。

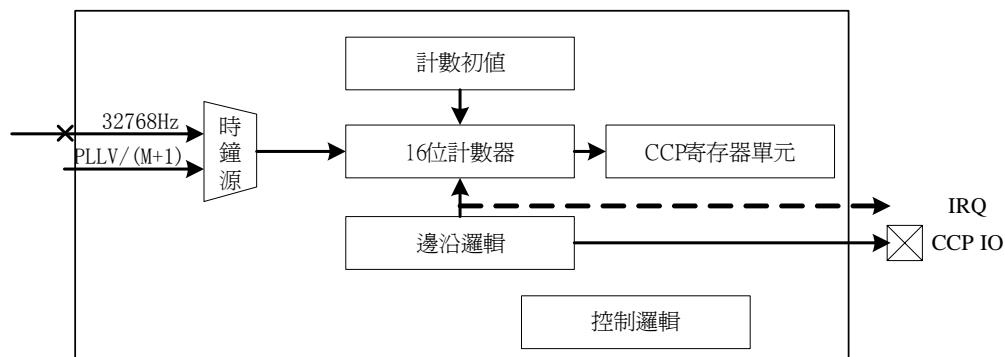


圖 4-10 Timer0~5 摄取輸入結構圖

Capture 的時序圖，這裏設置上升沿觸發，如圖 4-11 所示：

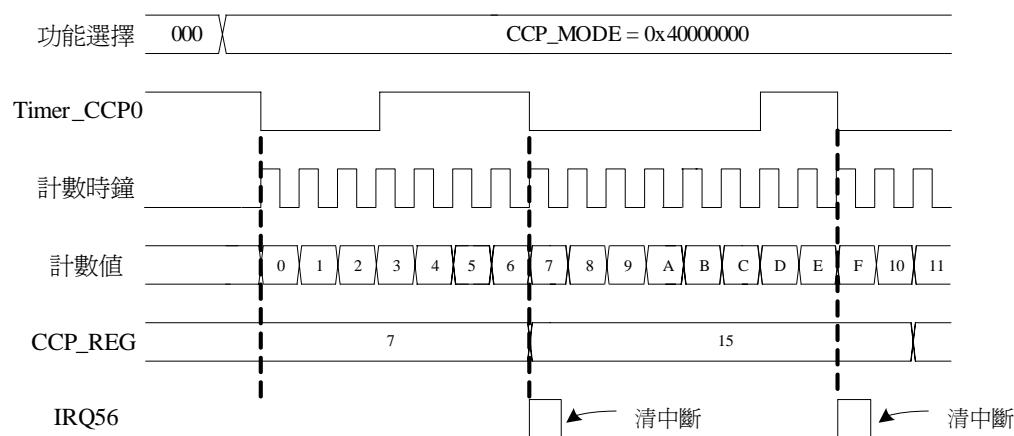


圖 4-11 Timer0~5 摄取輸入時序圖

應用 Timer0~5 的攝取功能，需要以下操作：配置計時器模組、選擇計時器時鐘源、設置計時器計數初值及 CCP 初值、使能 I/O 作為 CCP 埠使用、選擇計時器工作方式為比較攝取方式、使能計時器及中斷。其操作流程如圖 4-12：

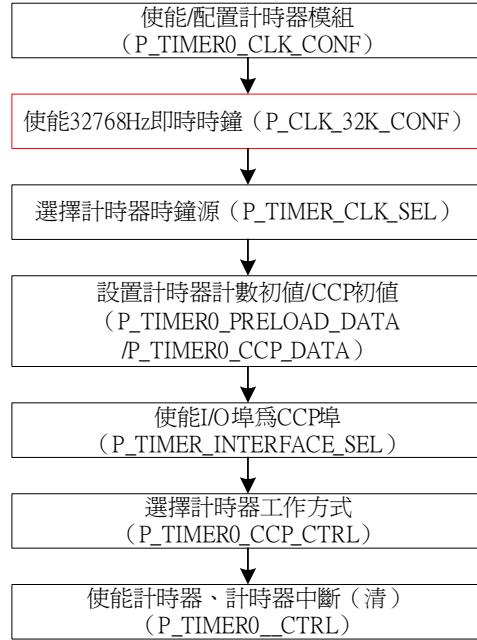


圖 4-12 Timer0~5 摷取輸入初始化流程圖

【例 4.3】

使用計時器 5 作為 PWM 輸出，計時器 4 作為比較擷取輸入，獲得 PWM 的輸出週期。

參考代碼：

```

#include "SPCE3200_Constant.h"
#include "SPCE3200_Register.h"
//-----主函數
-----
int main(void)
{
    *P_INT_MASK_CTRL1 = ~C_INT_TIMER_DIS;           // 打開 TIMER 中斷
    *P_CLK_32K_CONF = C_32K_CRY_EN;                 // 打開 32K 晶振
    *P_TIMER_CLK_SEL = C_TIMER4_CLK_32K
        | C_TIMER5_CLK_32K;                         // 選擇 TIMER4、TIMER5 時鐘源
    *P_TIMER_INTERFACE_SEL = C_TIMER4_PORT_SEL
        | C_TIMER5_PORT_SEL;

    *P_TIMER5_CLK_CONF = 0;                          // 重設 TIMER5 模組
    *P_TIMER5_CLK_CONF = C_TIMER_CLK_EN
        | C_TIMER_RST_DIS;                         // 使能 TIMER5 模組
    *P_TIMER5_PRELOAD_DATA = 0xF200;                // 設置計數初值
    *P_TIMER5_CCP_DATA = 0xF700;                    // 設置 CCP 初值
    *P_TIMER5_CCP_CTRL = C_TIMER_PWM_MODE;          // 選擇 TIMER5 工作在 PWM 模式
    *P_TIMER5_MODE_CTRL = C_TIMER_CTRL_EN
        | C_TIMER_INT_EN | C_TIMER_INT_FLAG;        // 使能 TIMER5

    *P_TIMER4_CLK_CONF = 0;                          // 重設 TIMER4 模組
    *P_TIMER4_CLK_CONF = C_TIMER_CLK_EN
        | C_TIMER_RST_DIS;                         // 使能 TIMER4 模組
    *P_TIMER4_CCP_CTRL = C_TIMER_CAP_MODE         // 選擇 TIMER4 工作在擷取模式
}

```

```

        | C_TIMER_CAP_FALL;
*P_TIMER4_MODE_CTRL = C_TIMER_CTRL_EN           // 使能 TIMER4
        | C_TIMER_INT_EN | C_TIMER_INT_FLAG;

while(1);
return 0;
}
//-----中斷服務函數
-----
IRQ56(void)
{
    unsigned int t;
    unsigned int h;
    if(*P_TIMER5_MODE_CTRL & C_TIMER_INT_FLAG)
    {
        *P_TIMER5_MODE_CTRL |= C_TIMER_INT_FLAG;
    }
    if(*P_TIMER4_MODE_CTRL & C_TIMER_INT_FLAG)
    {
        *P_TIMER4_MODE_CTRL |= C_TIMER_INT_FLAG;
        t = *P_TIMER4_COUNT_DATA;
        h = *P_TIMER4_CCP_DATA;           // 獲得 PWM 週期
    }
}

```

4. PWM 輸出模式

Timer0~5 作 PWM 輸出的結構如圖 4-13 所示：其核心是一個 16 位計數器與 PWM 控制邏輯。透過 P_TIMER0_PRELOAD_DATA 與 P_TIMER0_CCP_DATA 暫存器決定 TIMER0 模組的 PWM 信號的週期和脈寬，PWM 信號透過 CCP IO 輸出。具體對應晶片上管腳參考表 4-4。

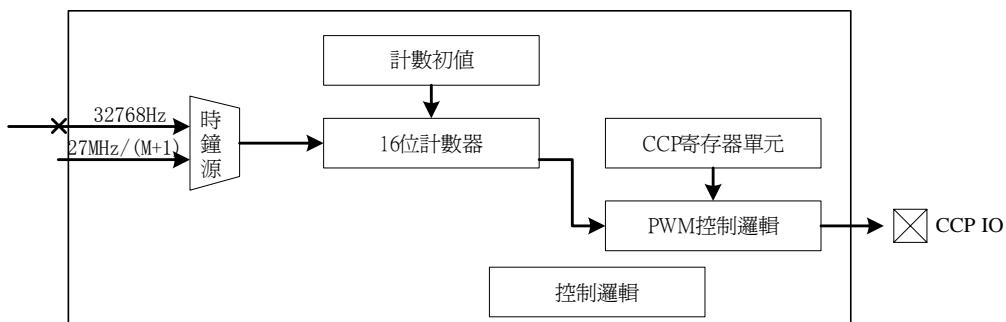


圖 4-13 Timer0~5 PWM 輸出結構圖

Timer0~5 PWM 輸出的時序如圖 4-14 所示：

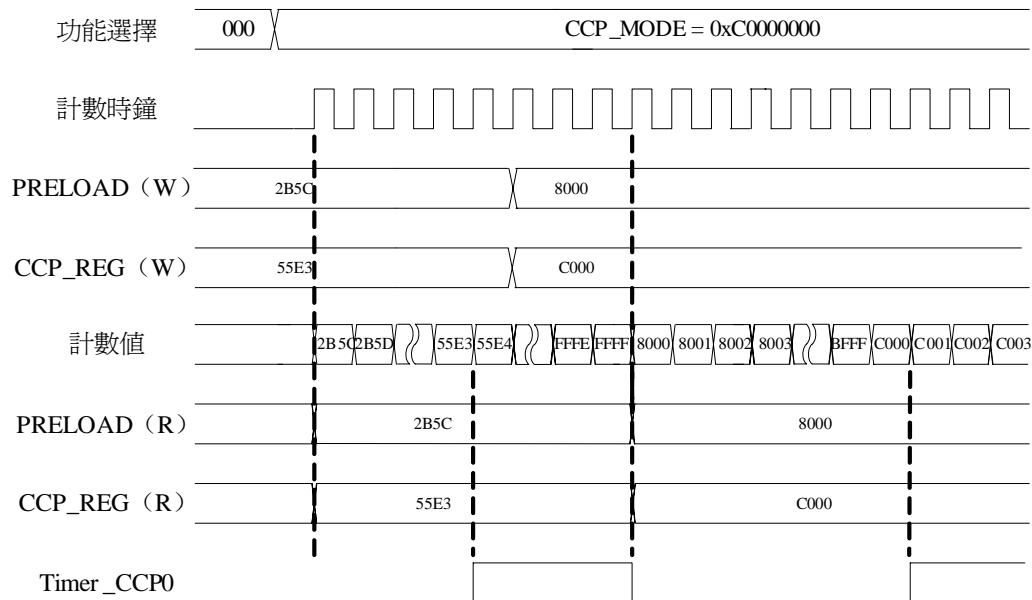


圖 4-14 Timer0~5 PWM 輸入時序圖

向 P_TIMER0_CCP_CTRL 暫存器的 b31:b30 寫入 11，選擇計時器工作在 PWM 輸出模式下，使能計時器，計數器載入 P_TIMER0_PRELOAD_DATA 與 P_TIMER0_CCP_DATA 的值，並開始計數。當計數值 P_TIMER0_COUNT_DATA 與 P_TIMER0_CCP_DATA 的值匹配時置位 PWM 輸出埠，當計數器溢出時（達到 65536）清除 PWM 輸出埠，周而復始。當計數器溢出時，可以重新載入計數初值與 CCP 初值。若計時器的計數時鐘頻率為 f ，P_TIMER0_PRELOAD_DATA 和 P_TIMER0_CCP_DATA 預置的值分別為 PRELOAD 和 CCP_REG，則輸出 PWM 信號的週期和占空比由圖 4-15 公式計算：

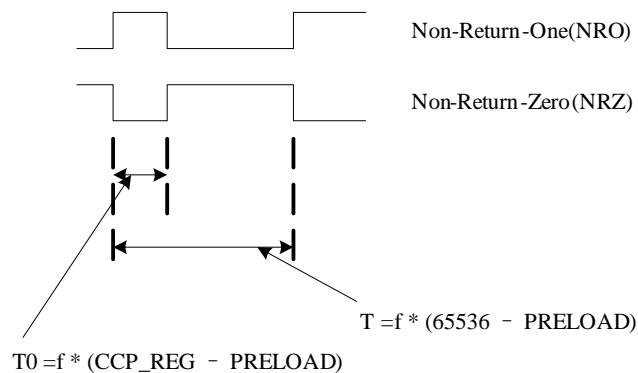


圖 4-15 PWM 輸出信號計算方法

透過設置暫存器 P_TIMER0_CCP_CTRL 的 b25 位選擇 NRO 模式或 NRZ 模式，當 b25 為 0 時選擇 NRO 模式；當 b25 為 1 時選擇 NRZ 模式。

應用 Timer0~5 的 PWM 輸出功能，需要以下操作：配置計時器模組、選擇計時器時鐘源、設置計時器計數初值及 CCP 初值、使能 I/O 作為 CCP 埠使用、選擇計時器工作方式為 PWM 輸出方式、使能計時器及中斷。其操作流程如下：

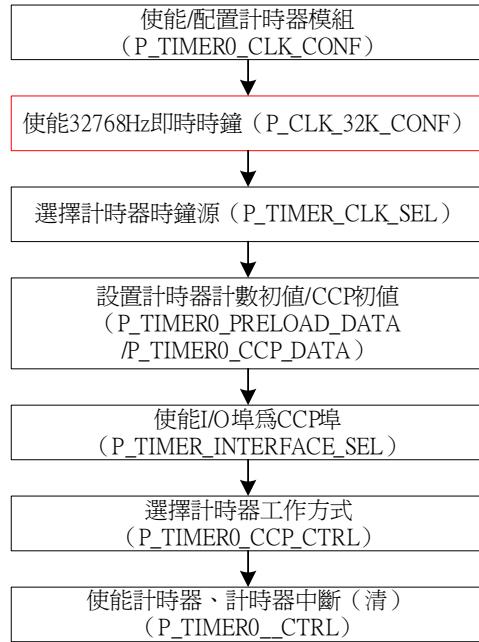
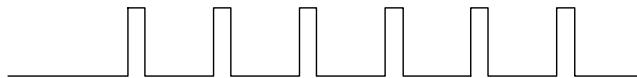


圖 4-16 Timer0~5 PWM 輸出初始化流程圖

【例 4.4】

利用 Timer0 的 PWM 輸出功能，透過 Timer0_CCP 輸出週期為 0.5s，占空比為 20%的波形 (FSYS = 32768Hz)。



根據公式 $T = f * (65536 - PRELOAD)$ 和 $T0 = f * (CCP_REG - PRELOAD)$ 計算得到 PRELOAD = 0xC000，CCP_REG = 0xFFFF。

參考代碼：

```

#include "SPCE3200_Constant.h"
#include "SPCE3200_Register.h"

void PWM_OUT(unsigned int preload, unsigned int valueccp);
void PWM_OUT(unsigned int preload, unsigned int valueccp)
{
    *P_INT_MASK_CTRL1 = ~C_INT_TIMER_DIS;           // 打開 TIMER 中斷
    *P_TIMER0_CLK_CONF = 0;                          // TIMER0 模組重設
    *P_TIMER0_CLK_CONF = C_TIMER_CLK_EN             // TIMER0 模組使能
    | C_TIMER_RST_DIS;
    *P_CLK_32K_CONF = C_32K_CRY_EN;                 // 使能 32K 晶振
    *P_TIMER_CLK_SEL = C_TIMER0_CLK_32K;            // TIMER0 選擇 32k 時鐘源
    *P_TIMER_INTERFACE_SEL = C_TIMER0_PORT_SEL;     // 使能 TIMER0 CCP 埠
    *P_TIMER0_PRELOAD_DATA = preload;                // 載入初值
    *P_TIMER0_CCP_DATA = valueccp;                  // 載入 CCP 初值
    *P_TIMER0_CCP_CTRL = C_TIMER_PWM_MODE;          // TIMER0 選擇 PWM 工作模式
    *P_TIMER0_MODE_CTRL = C_TIMER_CTRL_EN           // TIMER0 使能、中斷使能
    | C_TIMER_INT_EN
    | C_TIMER_INT_FLAG;
}

```

```
}
```

```
int main(void)
```

```
{
```

```
    PWM_OUT(0xC000,0xFFFF);
```

```
    while(1);
```

```
    return 0;
```

```
}
```

4.2.7 注意事項

TIMER 在使用時需要將該模組先重設在使能，然後才能使用。

4.3 即時時鐘——RTC

4.3.1 概述

即時時鐘 RTC (Real Time Clock) 提供一套計時系統，由 32768Hz 的時鐘源經過分頻後得到秒、分、時以及鬧鐘信號，可以方便的得到計時時間。

4.3.2 特性

- 提供 1/2 秒、1 秒、1 分以及 1 小時的中斷請求
- 提供鬧鐘信號中斷請求

4.3.3 暫存器描述

和 RTC 有關的暫存器總共有 10 個，參考表 4-28：

表 4-28 RTC 相關暫存器描述

暫存器英文名稱	暫存器中文名稱	位址
32K 即時時鐘配置暫存器	P_CLK_32K_CONF	0x88210114
RTC 時鐘配置暫存器	P_RTC_CLK_CONF	0x88210088
RTC 秒暫存器	P_RTC_TIME_SEC	0x88166000
RTC 分暫存器	P_RTC_TIME_MIN	0x88166004
RTC 時暫存器	P_RTC_TIME_HOUR	0x88166008
RTC 秒報警暫存器	P_RTC_ALM_SEC	0x8816600C
RTC 分報警暫存器	P_RTC_ALM_MIN	0x88166010
RTC 時報警暫存器	P_RTC_ALM_HOUR	0x88166014
RTC 控制暫存器	P_RTC_MODE_CTRL	0x88166018
RTC 中斷狀態暫存器	P_RTC_INT_STATUS	0x8816601C

■ 32K 即時時鐘配置暫存器：P_CLK_32K_CONF(0x88210114)

請參考第三章設置。

■ RTC 時鐘配置暫存器：P_RTC_CLK_CONF(0x88210088)

配置 RTC 模組時鐘的暫存器，當需要使用 RTC 模組，需要配置此暫存器。

表 4-29 P_RTC_CLK_CONF(0x88210088)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	RTC_RST	RTC_STOP

RTC_RST	b1	RTC 模組時鐘重設位： 0 : RTC 模組時鐘重設 1 : RTC 模組時鐘不重設
RTC_STOP	b0	RTC 模組時鐘使能位： 0 : RTC 模組時鐘停止 1 : RTC 模組時鐘使能

■ RTC 秒暫存器：P_RTC_TIME_SEC(0x88166000)

初始化 RTC 計時系統的秒計時，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。寫入秒初始之後，當啓動 RTC，該值每隔 1s 加一，直到 59，然後會使 P_RTC_TIME_MIN 暫存器的值加一，同時自身恢復為 00，周而復始。讀該暫存器會得到當前的秒計時。

表 4-30 P_RTC_TIME_SEC(0x88166000)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	SECOND_SET

■ RTC 分暫存器：P_RTC_TIME_MIN(0x88166004)

初始化 RTC 計時系統的分計時，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。寫入分初始之後，當啓動 RTC，該值每隔 60s 加一，直到 59，然後會使 P_RTC_TIME_HOUR 暫存器的值加一，同時自身恢復為 00，周而復始。讀該暫存器會得到當前的分計時。

表 4-31 P_RTC_TIME_MIN(0x88166004)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	MINUTE_SET

■ RTC 時暫存器：P_RTC_TIME_HOUR(0x88166008)

初始化 RTC 計時系統的時計時，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。寫入時初始之後，當啓動 RTC，該值每隔 3600s 加一，直到 59，自身恢復為 00，周而復始。讀該暫存器會得到當前的小時。

表 4-32 P_RTC_TIME_HOUR(0x88166008)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	HOUR_SET

■ RTC 秒報警暫存器：P_RTC_ALM_SEC(0x8816600C)

初始化 RTC 定時系統的秒初值，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。當計時系統的小時、分鐘、秒鐘與定時系統的小時、分鐘、秒鐘完全匹配時，產生定時報警中斷。

表 4-33 P_RTC_ALM_SEC(0x8816600C)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	ALM_SECOND_SET

■ RTC 分報警暫存器：**P_RTC_ALM_MIN(0x88166010)**

初始化 RTC 定時系統的分初值，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。當計時系統的小時、分鐘、秒鐘與定時系統的小時、分鐘、秒鐘完全匹配時，產生定時報警中斷。

表 4-34 P_RTC_ALM_MIN(0x88166010)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	ALM_MINUTE_SET

■ RTC 時報警暫存器：**P_RTC_ALM_HOUR(0x88166014)**

初始化 RTC 定時系統的時初值，該值範圍在 0~59，如果寫入其他值，將得到不可預知的結果。當計時系統的小時、分鐘、秒鐘與定時系統的小時、分鐘、秒鐘完全匹配時，產生定時報警中斷。

表 4-35 P_RTC_ALM_HOUR(0x88166014)

位	b5~b0
讀/寫	R/W
預設值	0
名稱	ALM_HOUR_SET

■ RTC 控制暫存器：**P_RTC_MODE_CTRL(0x88166018)**

當使用 RTC 即時時鐘模組時，需要設置此暫存器使能。

表 4-36 P_RTC_ALM_HOUR(0x88166014)

位	b31
讀/寫	R/W
預設值	0
名稱	RTC_EN

RTC_EN	b0	RTC 即時時鐘使能位： 0: 禁止 RTC 即時時鐘 1: 使能 RTC 即時時鐘
--------	----	--

■ RTC 中斷狀態暫存器：P_RTC_INT_STATUS(0x8816601C)

該暫存器可以使能 RTC 中斷、判斷 RTC 中斷是否發生及清除 RTC 中斷。

表 4-37 P_RTC_INT_STATUS(0x8816601C)

位	b31	b30	b29~b28	b27	b26	b25~b24	b23
讀/寫	R/W	R/W	-	R/W	R/W	-	R/W
預設值	0	0	-	0	0	-	0
名稱	ALM_INT	ALM_INTE_N	-	HOUR_INT	HOUR_INT_EN	-	MIN_INT
位	b22	b21~b20	b19	b18	b17~b16	b15	b14
讀/寫	R/W	-	R/W	R/W	-	R/W	R/W
預設值	0	-	0	0	-	0	0
名稱	MIN_INTE_N	-	SEC_INT	SEC_INTE_N	-	HSEC_INT	HSEC_INT_EN

ALM_INT	b31	鬧鐘中斷旗標位：
		讀 0：沒有發生鬧鐘中斷
		讀 1：發生鬧鐘中斷
		寫 0：無意義

寫 1：清除中斷旗標

ALM_INEN	b30	鬧鐘中斷使能位：
		0：禁止鬧鐘中斷
		1：使能鬧鐘中斷

HOUR_INT	b27	時中斷旗標位： 讀 0：沒有發生時中斷 讀 1：發生時中斷 寫 0：無意義 寫 1：清除中斷旗標
HOUR_INTEN	b26	時中斷使能位： 0：禁止時中斷 1：使能時中斷
MIN_INT	b23	分中斷旗標位： 讀 0：沒有發生分中斷 讀 1：發生分中斷 寫 0：無意義 寫 1：清除中斷旗標
MIN_INTEN	b22	分中斷使能位： 0：禁止分中斷 1：使能分中斷
SEC_INT	b19	秒中斷旗標位： 讀 0：沒有發生秒中斷 讀 1：發生秒中斷 寫 0：無意義 寫 1：清除中斷旗標
SEC_INTEN	b18	秒中斷使能位： 0：禁止秒中斷 1：使能秒中斷
HSEC_INT	b15	半秒中斷旗標位： 讀 0：沒有發生半秒中斷 讀 1：發生半秒中斷 寫 0：無意義 寫 1：清除中斷旗標

HSEC_INTEN

b14

半秒中斷使能位：

0：禁止半秒中斷

1：使能半秒中斷

4.3.4 基本操作

當需要使用 RTC 模組時，需要按照下面流程初始化，如圖 4-17 所示：首先需要配置 RTC 模組，然後使能 32768Hz 即時時鐘，當需要採用計時系統時，初始化時間；當需要定時時，初始化定時；當只需要半秒、秒等中斷時，不用初始化時間，預設為 0 時 0 分 0 秒，然後使能中斷及使能 RTC 模組，初始化流程結束。

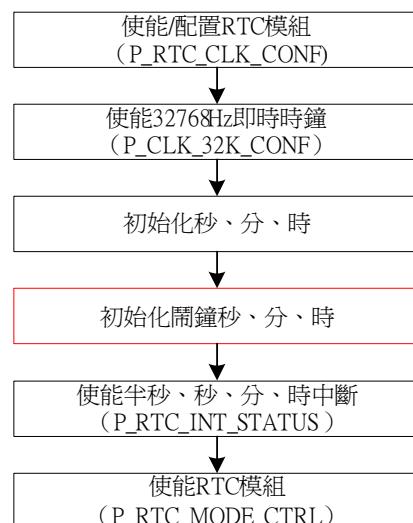
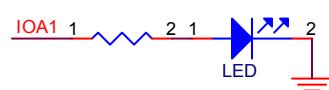


圖 4-17 初始化 RTC 流程

4.3.5 應用舉例

【例 4.5】

使用 RTC 半秒中斷閃爍 LED。硬體連接圖如下：



參考代碼：該代碼使用查詢中斷方式實現。

```

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    *P_IOA_GPIO_SETUP = 0x00000300;
    *P_CLK_32K_CONF = C_32K_CRY_EN; // 使能 32768Hz 晶振
    *P_RTC_CLK_CONF = C_RTC_CLK_EN | C_RTC_RST_DIS; // RTC 時鐘配置
}

```

```

*P_RTC_MODE_CTRL = C_RTC_CTRL_EN;           // 使能 RTC 模組
*P_RTC_INT_STATUS = C_RTC_HSEC_INTEN        // 使能半秒中斷
| C_RTC_HSEC_FLAG;

while(1)
{
    if(*P_RTC_INT_STATUS & C_RTC_HSEC_FLAG)
    {
        *P_IOA_GPIO_SETUP ^= 0x00000003;
        *P_RTC_INT_STATUS |= C_RTC_HSEC_FLAG;      // 清中斷旗標
    }
}
    
```

4.4 時基——Time Base

4.4.1 概述

SPCE3200 內嵌時間基準信號發生器，簡稱時基。在 SPCE3200 內部共有這樣的模組 3 個，TMB0、TMB1、TMB2，它們的結構完全相同，下文將以 TMB0 進行介紹。TMB0 模組透過一個計數器對 32768Hz 時鐘信號進行分頻，一共產生 12 個可操作時基信號：2048Hz、1024Hz、512Hz、256Hz、128Hz、64Hz、32Hz、16Hz、8Hz、4Hz、2Hz、1Hz 信號。

4.4.2 結構

TMB0 時基模組的結構如圖 4-18 所示：設置 P_TMB_MODE_CTRL 暫存器可以選擇時基的頻率，在頻率信號到來時產生 IRQ 中斷。向 P_TMB_RESET_COMMAND 暫存器寫入 0x5xxxxxx5 可以重設時基模組，在系統重設時也可以重設時基模組。

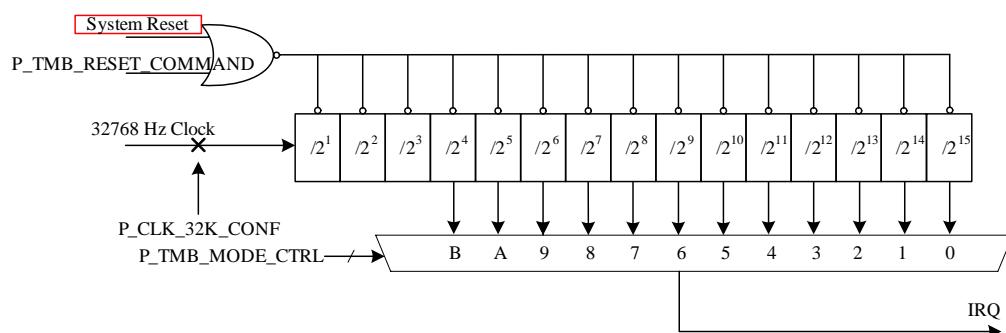


圖 4-18 時基模組的結構

4.4.3 暫存器描述

與 TMB 相關的暫存器共有 5 個，參考表 4-38：

表 4-38 與 TMB 相關的暫存器

暫存器名稱	助記符	位址
32K 即時時鐘配置暫存器	P_CLK_32K_CONF	0x88210114
TMB 時鐘配置暫存器	P_TMB_CLK_CONF	0x882100E0
TMB 控制暫存器	P_TMB_MODE_CTRL	0x88166020
TMB 中斷狀態暫存器	P_TMB_INT_STATUS	0x88166024
TMB 重設命令暫存器	P_TMB_RESET_COMMAND	0x88166028

■ **TMB 時鐘配置暫存器：P_TMB_CLK_CONF(0x882100E0)**

配置 TMB 模組時鐘暫存器，當需要使用 TMB 模組，需要配置此暫存器。

表 4-39 P_TMB_CLK_CONF(0x882100E0)

位	b31~b3	b2	b1	b0
讀/寫	-	R/W	-	R/W
預設值	-	0	-	0
名稱	-	TBASE_RST	-	TBASE_STOP

TBASE_RST b2 TMB 模組時鐘重設位：

0 : TMB 模組時鐘重設

1 : TMB 模組時鐘不重設

TBASE_STOP b0 TMB 模組時鐘使能位：

0 : TMB 模組時鐘停止

1 : TMB 模組時鐘使能

■ **TMB 控制暫存器：P_TMB_MODE_CTRL(0x88166020)**

透過該暫存器可以使能 TMB0、TMB1 及 TMB2 模組，可以選擇 TMB0、TMB1、TMB2 的時基頻率。

表 4-40 P_TMB_MODE_CTRL(0x88166020)

位	b31	b27~b24	b23	b19~b16	b15	b11~b8
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0
名稱	TMB0_EN	TMB0_SEL	TMB1_EN	TMB1_SEL	TMB2_EN	TMB2_SEL

TMB0_EN	b31	TMB0 使能位： 0 : TMB0 使能 1 : TMB0 禁止
TMB0_SEL	b27~b24	TMB0 輸出時基頻率選擇位： 0000 : 1Hz 0001 : 2Hz 0010 : 4Hz 0011 : 8Hz 0100 : 16Hz 0101 : 32Hz 0110 : 64Hz 0111 : 128Hz 1000 : 256Hz 1001 : 512Hz 1010 : 1024Hz 1011 : 2048Hz
TMB1_EN	b23	TMB1 使能位： 0 : TMB1 使能 1 : TMB1 禁止

TMB1_SEL	b19~b16	TMB1 輸出時基頻率選擇位：
		0000 : 1Hz
		0001 : 2Hz
		0010 : 4Hz
		0011 : 8Hz
		0100 : 16Hz
		0101 : 32Hz
		0110 : 64Hz
		0111 : 128Hz
		1000 : 256Hz
		1001 : 512Hz
		1010 : 1024Hz
		1011 : 2048Hz

TMB2_EN	b15	TMB2 使能位：
		0 : TMB2 使能
		1 : TMB2 禁止

TMB2_SEL	b11~b8	TMB2 輸出時基頻率選擇位：
		0000 : 1Hz
		0001 : 2Hz
		0010 : 4Hz
		0011 : 8Hz
		0100 : 16Hz
		0101 : 32Hz
		0110 : 64Hz
		0111 : 128Hz
		1000 : 256Hz
		1001 : 512Hz
		1010 : 1024Hz
		1011 : 2048Hz

■ TMB 中斷狀態暫存器：**P_TMB_INT_STATUS(0x88166024)**

該暫存器使能 TMB0、TMB1、TMB2 的中斷、判斷中斷是否發生及清除中斷旗標位元。

表 4-41 P_TMB_INT_STATUS(0x88166024)

位	b31	b30	b27	b26	b23	b22
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0
名稱	TMB0_INT	TMB0_INTEN	TMB1_INT	TMB1_INTEN	TMB2_INT	TMB2_INTEN

TMB0_INT	b31	TMB0 中斷旗標位： 讀 0：沒有發生 TMB0 中斷 讀 1：發生 TMB0 中斷 寫 0：無意義 寫 1：清除中斷旗標
TMB0_INTEN	b30	TMB0 中斷使能位： 0：禁止 TMB0 中斷 1：使能 TMB0 中斷
TMB1_INT	b27	TMB1 中斷旗標位： 讀 0：沒有發生 TMB1 中斷 讀 1：發生 TMB1 中斷 寫 0：無意義 寫 1：清除中斷旗標
TMB1_INTEN	b26	TMB1 中斷使能位： 0：禁止 TMB1 中斷 1：使能 TMB1 中斷
TMB2_INT	b23	TMB2 中斷旗標位： 讀 0：沒有發生 TMB2 中斷 讀 1：發生 TMB2 中斷 寫 0：無意義 寫 1：清除中斷旗標

TMB2_INTEN	b22	TMB2 中斷使能位： 0：禁止 TMB2 中斷 1：使能 TMB2 中斷
------------	-----	---

■ TMB 重設命令暫存器：P_TMB_RESET_COMMAND(0x88166028)

向該暫存器寫入 0x5xxxxxx5 可以重設 TMB 時基計數器。

表 4-42 P_TMB_RESET_COMMAND(0x88166028)

位	b31~b0
讀/寫	R/W
預設值	0
名稱	RESET_COM

RESET_COM	b31~b0	寫 0x5xxxxxx5 重設時基計數器
-----------	--------	----------------------

4.4.4 基本操作

當選擇 TMB 模組工作時，初始化流程如圖 4-19：首先配置 TMB 模組，然後使能 32768Hz 時鐘，使能 TMB 的中斷，選擇 TMB 的時基頻率，使能 TMB 模組。

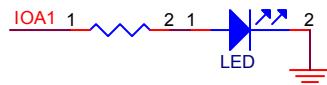


圖 4-19 初始化 TMB 流程

4.4.5 應用舉例

【例 4.6】

使用 TM0 時基模組 2Hz 中斷閃爍 LED。硬體連接圖如下：



參考代碼：該代碼使用查詢中斷方式實現。

```
#include "RTC.h"

int main(void)
{
    *P_CLK_32K_CONF = C_32K_CRY_EN; // 32768Hz 晶振使能
    *P_IOA_GPIO_SETUP = 0x00000202;
    *P_TMB_CLK_CONF = 0x00000000;
    *P_TMB_CLK_CONF = C_TMB_CLK_EN | C_TMB_RST_DIS; // 時基時鐘配置
    *P_TMB_INT_STATUS = C_TMB_TMB0_INTEN; // 使能 TMB0 中斷
    *P_TMB_MODE_CTRL = C_TMB_TMB0_EN; // 使能 TMB0
    *P_TMB_MODE_CTRL |= C_TMB_TMB0_2HZ; // 選擇 2Hz

    while(1)
    {
        while(*P_TMB_INT_STATUS & C_TMB_TMB0_FLAG) // 清中斷旗標
        {
            *P_TMB_INT_STATUS |= C_TMB_TMB0_FLAG;
            *P_IOA_GPIO_SETUP ^= 0x00000002;
        }
    }
}
```

4.5 看門狗——WDOG

4.5.1 概述

為了避免因程式“跑飛”而引起的系統問題，SPCE3200 內部集成了看門狗計數器 WDOG (Watchdog)。

SPCE3200 的 WDOG 實事上就是一個 32 位的計數器，時鐘由 32768Hz 的晶振提供。計數器遞減計數，當計數器計到 0 時計數器溢出，引發系統重設。因此在使能看門狗後，在計數器還沒有溢出前要及時喂狗，否則系統會因看門狗產生重設。

4.5.2 特性

WDOG 的特性如下：

- WDOG 可編程使能：即 WDOG 是否被使能可透過軟體設置
- WDOG 的計數器遞減計數
- WDOG 的計數初值可編程設置：即 WDOG 計數器的計數初值可透過軟體來設置
- WDOG 可透過喂狗來重載計數初值：使能看門狗後，如果執行喂狗操作，WDOG 計數器重載計數初值

4.5.3 結構

WDOG 的結構框圖如圖 4-20：32768Hz 晶振直接給看門狗計數器提供時鐘；可以透過暫存器設置看門狗計數器的計數初值；使能看門狗後，看門狗計數器載入計數初值開始遞減計數；在計數的過程中，如果執行了喂狗操作，看門狗計數器重載計數初值開始遞減計數；一直不喂狗會引發異常並使系統重設。

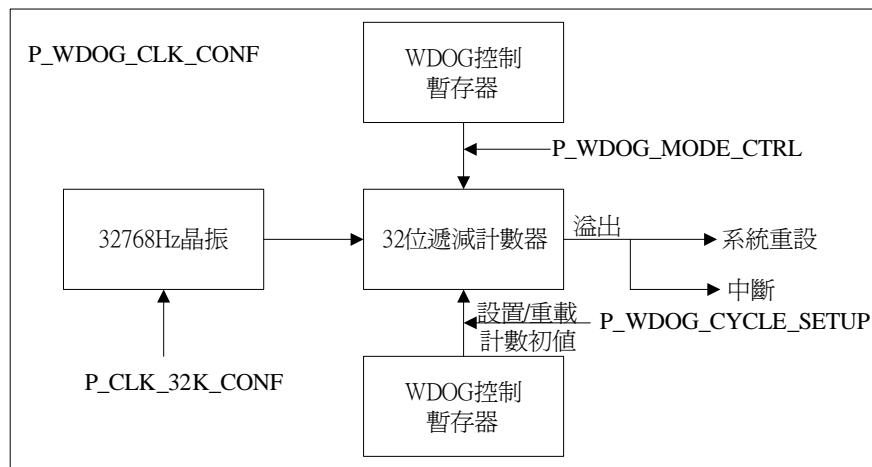


圖 4-20 WDOG 的結構框圖

4.5.4 暫存器描述

看門狗模組相關的暫存器共有 6 個，如表 4-43。透過這些暫存器可以打開或者關閉看門狗功能，同時還可以設置看門狗定時時間。

表 4-43 看門狗相關暫存器列表

暫存器名稱	助記符	位址
32K 即時時鐘配置暫存器	P_CLK_32K_CONF	0x88210114
WDOG 時鐘配置暫存器	P_WDOG_CLK_CONF	0x88210084
WDOG 重設模式暫存器	P_WDOG_RESET_STATUS	0x882100E8
WDOG 控制暫存器	P_WDOG_MODE_CTRL	0x88170000
WDOG 重設週期設置暫存器	P_WDOG_CYCLE_SETUP	0x88170004
WDOG 清狗命令暫存器	P_WDOG_CLR_COMMAND	0x88170008

其中 32K 即時時鐘配置暫存器在鎖相環與時鐘發生器章節已經介紹，這裏只介紹剩餘 5 個暫存器：

■ WDOG 時鐘配置暫存器：P_WDOG_CLK_CONF(0x88210084)

透過 WDOG 時鐘配置暫存器可以停止/打開 WDOG 時鐘或者重設/不重設 WDOG 模組，在

使能 WDOG 前需先打開 WDOG 時鐘。

表 4-44 P_WDOG_CLK_CONF(0x88210084)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	WDOG_RST	WDOG_STOP

WDOG_RST b1 看門狗模組時鐘重設位：

0：看門狗模組時鐘重設

1：看門狗模組時鐘不重設

WDOG_STOP b0 看門狗模組時鐘使能位：

0：看門狗模組時鐘停止

1：看門狗模組時鐘使能

■ WDOG 重設模式暫存器：P WDOG RESET STATUS(0x882100E8)

透過 WDOG 重設模式暫存器可以知道有沒有發生看門狗重設。

表 4-45 P_WDOG_RESET_STATUS(0x882100E8)

位	b31~b3	b1	b0
讀/寫	-	R/W	R
預設值	-	0	0
名稱	-	WDOG_INT	WDOG_ERRINT

WDOG_INT b1 看門狗錯誤重設旗標位：

讀 0：沒有發生看門狗錯誤重設

讀 1：發生看門狗錯誤重設

寫 0：無意義

寫 1：清除重設旗標

WDOG_ERRINT	b0	看門狗計數器溢出重設旗標位： 0：沒有發生看門狗計數器溢出重設 1：發生看門狗計數器溢出重設
-------------	----	--

■ WDOG 控制暫存器：**P_WDOG_MODE_CTRL(0x88170000)**

可以透過 WDOG 控制暫存器使能或者禁止看門狗功能。

表 4-46 P_WDOG_MODE_CTRL(0x88170000)

位	b31~b1	b0
讀/寫	-	R/W
預設值	-	0
名稱	-	WDOG_EN

WDOG_EN	b0	看門狗使能位： 0：禁止看門狗 1：使能看門狗
---------	----	-------------------------------

■ WDOG 重設週期設置暫存器：**P_WDOG_CYCLE_SETUP(0x88170004)**

SPCE3200 的看門狗模組有一個 32 位遞減計數的計數器，可以透過 P_WDOG_CYCLE_SETUP 暫存器設置其計數初值，計數初值的設置範圍為 0x00000001~0xFFFFFFFF。

表 4-47 P_WDOG_CYCLE_SETUP(0x88170004)

位	b31~b0
讀/寫	W
預設值	0x00000000
名稱	RESET_COM

RESET_COM	b31~b0	看門狗計數器計數初值設置位，設置範圍： 0x00000000~0xFFFFFFFF。
-----------	--------	---

當透過 P_WDOG_CYCLE_SETUP 暫存器把看門狗計數器的計數初值設置為 RESET_COM 時，看門狗計數器的重設週期為 RESET_COM/32768 (s)。

■ WDOG 清狗命令暫存器：P_WDOG_CLR_COMMAND(0x88170008)

當使能看門狗後，看門狗計數器按照 P_WDOG_CYCLE_SETUP 暫存器設置的計數初值開始遞減計數，在還沒有計數溢出前，如果透過 P_WDOG_CLR_COMMAND 暫存器清狗，看門狗計數器會重載計數初值並重新開始計數。

表 4-48 P_WDOG_CLR_COMMAND(0x88170008)

位	b31~b0
讀/寫	W
預設值	0
名稱	CLEAR_COM

CLEAR_COM	b31~b0	清狗寫命令：
		寫 0xaaaaaaaa5 清狗
		寫其他值系統重設

4.5.5 基本操作

使用看門狗功能通常有兩個操作：使能看門狗和清看門狗。

使能看門狗的流程如圖 4-21。

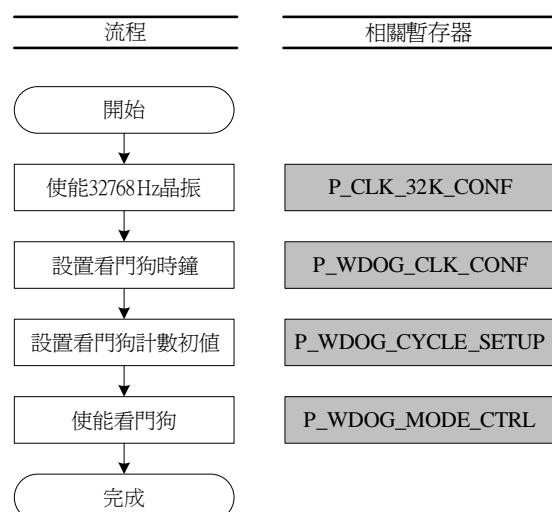


圖 4-21 打開看門狗計數器

打開看門狗計數器後，需在計數器沒有計數溢出前透過 P_WDOG_CLR_COMMAND 清狗，即向 P_WDOG_CLR_COMMAND 暫存器寫 0xAxxxxxx5 即可。

以上兩個操作可參考以下程式段：

```
*P_CLK_32K_CONF = C_32K_CRY_EN; // 32768Hz 晶振打開  
*P_WDOG_CLK_CONF = C_WDOG_CLK_EN | C_WDOG_RST_DIS; // Watch dog 時鐘配置  
*P_WDOG_CYCLE_SETUP = 400000; // 看門狗定時週期設置  
*P_WDOG_MODE_CTRL = C_WDOG_CTRL_EN; // 使能看門狗功能  
  
while(1)  
{  
    *P_WDOG_CLR_COMMAND = C_WDOG_CLR_COMMAND; // 清看門狗  
}
```

4.5.6 注意事項

在使用看門狗時需要注意以下幾點：

- 預設看門狗計數器是關閉的，即看門狗是禁止的，所以在使用前要先使能看門狗
- 在使用看門狗模組時，需要使能 32768Hz 晶振
- 清看門狗時注意寫入暫存器 P_WDOG_CLR_COMMAND 的資料是 0xAxxxxxx5，如果寫入其他資料會導致系統重設（看門狗錯誤重設）

4.6 睡眠與喚醒

4.6.1 睡眠

為了節省能源，SPCE3200 可工作在省電模式，也就是下文所說的睡眠；系統上電後，CPU 一直工作，直到檢測到睡眠命令（sleep）；單片機接收到睡眠信號後關閉系統時鐘（PLL），進入睡眠模式；進入睡眠模式後，程式計數器停在當前指令的下一條指令位址，如果有喚醒信號，系統從這個位址開始運行。

按照 PLL 是否被關閉，SPCE3200 有兩種省電模式（睡眠模式）：

- Wait 模式：不關閉 PLL
- Halt 模式：關閉 PLL

4.6.2 睡眠相關暫存器

與睡眠相關的暫存器有兩個：睡眠控制暫存器和睡眠時鐘選擇暫存器，如表 4-49。

表 4-49 睡眠相關暫存器列表

暫存器名稱	助記符	位址
睡眠控制暫存器	P_SLEEP_MODE_CTRL	0x88210000
睡眠時鐘選擇暫存器	P_SLEEP_CLK_SEL	0x882100DC

■ 睡眠模式控制暫存器：**P_SLEEP_MODE_CTRL(0x88210000)**

透過睡眠模式控制暫存器來控制系統進入 Wait 模式或者 Halt 模式。

表 4-50 P_SLEEP_MODE_CTRL(0x88210000)

位	b31~b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	-	W	W	W	W	W	W	W
預設值	-	0	0	0	0	1	1	1
名稱	-	CPUALLRST ST	CPUWRST T	CPUPRST T	CPUPLLI RQ	CPUCKMI U	CPUCKIR Q	CPUCK

CPUALLRST	b6	暫時重設 CPU (到 warmrst_n 埠) 和所有其他模組位： 0 : 不重設 1 : 重設
CPUWRST	b5	暫時重設 CPU (到 warmrst_n 埠) 位： 0 : 不重設 1 : 重設
CPUPRST	b4	暫時重設 CPU (到 Poweron_n 埠) 位： 0 : 不重設 1 : 重設
CPUPLLIRQ	b3	停止 CPU 和 MIU 時鐘然後停止 PLL 直到 IRQ 發生或者鍵狀態改變位： 0 : 不停止 1 : 停止
CPUCKMIU	b2	停止 CPU 和 MIU 時鐘直到 IRQ 發生或者鍵狀態改變位： 0 : 停止 1 : 不停止
CPUCKIRQ	b1	停止 CPU 時鐘直到 IRQ 發生或者鍵狀態改變位： 0 : 停止 1 : 不停止

CPUCK	b0	暫時停止 CPU 時鐘以改變 CPU 時鐘頻率位：
	0 : 停止	
	1 : 不停止	

說明：

- 透過 P_SLEEP_MODE_CTRL 的 b1 設置為 0 選擇系統省電模式為 Wait 模式；
 - 透過 P_SLEEP_MODE_CTRL 的 b2 設置為 0、b3 設置為 1 選擇系統省電模式為 Halt 模式。
-

■ 睡眠時鐘選擇暫存器：P_SLEEP_CLK_SEL (0x882100DC)

系統睡眠模式有兩種時鐘可以選擇：PLLV/8 和 32768Hz，透過睡眠時鐘選擇暫存器可以選擇睡眠模式利用哪種時鐘。

表 4-51 P_SLEEP_CLK_SEL (0x882100DC)

位	b31~b1	b0
讀/寫	-	W
預設值	-	0
名稱	-	SLEEP_CLK

SLEEP_CLK	b0	選擇睡眠模式的時鐘位
	0 : 選擇 PLLV/8 作為睡眠模式的時鐘	
	1 : 選擇 32768Hz 作為睡眠模式的時鐘	

4.6.3 喚醒

系統從睡眠模式被喚醒需要一個喚醒信號來接通系統時鐘(PLL)電路，並透過產生喚醒 IRQ 中斷信號引導 CPU 完成喚醒處理及系統初始化操作。喚醒中斷操作完成後，程式計數器會指引 CPU 執行睡眠指令的下一條指令。

睡眠模式的喚醒源可來自鍵狀態改變信號（鍵喚醒信號），也可以來自任一模組(該模組的時鐘未被關閉)的中斷信號（中斷喚醒源）。

透過 P_WAKEUP_KEYC_SEL(0x88200008)暫存器來選擇鍵喚醒的埠，用來作為鍵喚醒信號的埠分為 GROUP1、GROUP2、GROUP3、GROUP4、GROUP5 五組，如表 4-52。

表 4-52 鍵喚醒介面列表

序號	0x88200008[b2~b0] 設置	組	介面	對應鍵喚醒埠
1	001	GROUP1	LCD_D0	KEYCHANGE1_0
2			LCD_D1	KEYCHANGE1_1
3			LCD_D2	KEYCHANGE1_2
4			LCD_D3	KEYCHANGE1_3
5			LCD_D4	KEYCHANGE1_4
6			LCD_D5	KEYCHANGE1_5
7			LCD_D6	KEYCHANGE1_6
8			LCD_D7	KEYCHANGE1_7
9			LCD_D8	KEYCHANGE1_8
10			LCD_D9	KEYCHANGE1_9
11			LCD_D10	KEYCHANGE1_10
12			LCD_D11	KEYCHANGE1_11
13			LCD_D12	KEYCHANGE1_12
14			LCD_D13	KEYCHANGE1_13
15			LCD_D14	KEYCHANGE1_14
16			LCD_D15	KEYCHANGE1_15
17	010	GROUP2	CSI_D0	KEYCHANGE2_0
18			CSI_D1	KEYCHANGE2_1
19			CSI_D2	KEYCHANGE2_2
20			CSI_D3	KEYCHANGE2_3
21			CSI_D4	KEYCHANGE2_4
22			CSI_D5	KEYCHANGE2_5
23			CSI_D6	KEYCHANGE2_6
24			CSI_D7	KEYCHANGE2_7
25			CSI_CKO	KEYCHANGE2_8
26			CSI_CKI	KEYCHANGE2_9
27			CSI_HS	KEYCHANGE2_10

序號	0x88200008[b2~b0] 設置	組	介面	對應鍵喚醒埠
28			CSI_VS	KEYCHANGE2_11
29			CSI_FIELD	KEYCHANGE2_12
30			I2C_CLK	KEYCHANGE2_13
31			I2C_DATA	KEYCHANGE2_14
32	011	GROUP3	CSI_HS	KEYCHANGE3_0
33			CSI_VS	KEYCHANGE3_1
34			CSI_FIELD	KEYCHANGE3_2
35			UART_RX	KEYCHANGE3_3
36			UART_TX	KEYCHANGE3_4
37			ADC_CH4	KEYCHANGE3_5
38			ADC_CH5	KEYCHANGE3_6
39			ADC_CH6	KEYCHANGE3_7
40			ADC_CH7	KEYCHANGE3_8
41	100	GROUP4	NFLASH_D6	KEYCHANGE4_0
42			NFLASH_REN	KEYCHANGE4_1
43			NFLASH_WEN	KEYCHANGE4_2
44			NFLASH_CEN	KEYCHANGE4_3
45			NFLASH_RDY	KEYCHANGE4_4
46			ADC_CH4	KEYCHANGE4_5
47			ADC_CH5	KEYCHANGE4_6
48			ADC_CH6	KEYCHANGE4_7
49			ADC_CH7	KEYCHANGE4_8
50			NFLASH_D7	KEYCHANGE4_9
51	101	GROUP5	NFLASH_ALE	KEYCHANGE2_0
52			NFLASH_REN	KEYCHANGE2_1
53			NFLASH_WEN	KEYCHANGE2_2
54			NFLASH_CEN	KEYCHANGE2_3
55			NFLASH_RDY	KEYCHANGE2_4

序號	0x88200008[b2~b0] 設置	組	介面	對應鍵喚醒埠
56			NFLASH_D4	KEYCHANGE2_5
57			NFLASH_D5	KEYCHANGE2_6
58			NFLASH_D6	KEYCHANGE2_7
59			NFLASH_D7	KEYCHANGE2_8
60			SPI_CSN	KEYCHANGE2_9
61			CSI_HS	KEYCHANGE2_10
62			CSI_VS	KEYCHANGE2_11
63			CSI_FIELD	KEYCHANGE2_12

用來作為中斷喚醒信號的中斷喚醒源如表 4-53。

表 4-53 睡眠模式的中斷喚醒源

喚醒中斷源	說明	喚醒中斷源	說明
Mic 中斷	透過 Mic 引入的中斷	Flash 中斷	Nand Flash 中斷
ADC 中斷	透過普通 A/D 通道引入的中斷	SD 中斷	SD 卡中斷
時基中斷	時基中斷	I2C 中斷	I2C 主機中斷
計數器中斷	Timer0~5 引入的中斷	I2S 中斷	I2S 主/從中斷
TV 中斷 1	TV 垂直消隱啓動中斷	APBDMA 中斷 1	APBDMA 通道 1 操作完成中斷
LCD 中斷	LCD 垂直消隱啓動中斷	APBDMA 中斷 2	APBDMA 通道 2 操作完成中斷
TV 中斷 3	TV 光槍擊中中斷	LDMDMA 中斷	LDM DMA 操作完成中斷
CSI 中斷 1	感測器框結束中斷	BLN 中斷	混合 DMA 操作完成中斷
CSI 中斷 3	感測器座標擊中中斷	APBDMA 中斷 3	APBDMA 通道 3 操作完成中斷
CSI 中斷 2	感測器運動框結束中斷	APBDMA 中斷 4	APBDMA 通道 4 操作完成中斷
CSI 中斷 0	擷取完成中斷	RTC 中斷	即時時鐘中斷
TV 中斷 4	TV 座標擊中中斷	MPG 中斷	MPG4 一框編/譯碼完成中斷
USB 中斷	USB 主/從設備中斷	ECC 中斷	ECC 發現錯誤的事件中斷
SIO 中斷	SIO 中斷	SFTCFG 中斷	指定管腳的上升沿/下降沿
SPI 中斷	SPI 中斷	鍵值改變中斷	鍵值改變中斷
UART 中斷	UART 中斷	LVD 中斷	低電壓檢測中斷

4.6.4 鍵喚醒相關暫存器

鍵喚醒有兩個控制暫存器：鍵喚醒控制暫存器和鍵喚醒清除暫存器，如表 4-54。

表 4-54 鍵喚醒相關暫存器列表

暫存器中文名稱	暫存器英文名稱	位址
鍵喚醒選擇暫存器	P_WAKEUP_KEYC_SEL	0x88200008
鍵喚醒清除暫存器	P_WAKEUP_KEYC_CLR	0x882100C0

■ 鍵喚醒控制暫存器：P_WAKEUP_KEYC_SEL(0x88200008)

可以透過 P_WAKEUP_KEYC_SEL 的 b2~b0 選擇 GROUP1、GROUP2、GROUP3、GROUP4、GROUP5 五組中任一組作為鍵喚醒埠。

表 4-55 P_WAKEUP_KEYC_SEL(0x88200008)

位	b31~b3	b2~b0
讀/寫	-	R/W
預設值	-	000
名稱	-	SW_KEYC

SW_KEYC

b1~b0

選擇鍵喚醒介面，參考表 4-52：

001：選擇 GROUP1 組介面為鍵喚醒埠

010：選擇 GROUP2 組介面為鍵喚醒埠

011：選擇 GROUP3 組介面為鍵喚醒埠

100：選擇 GROUP4 組介面為鍵喚醒埠

101：選擇 GROUP5 組介面為鍵喚醒埠

其他：不選擇鍵喚醒埠

■ 鍵喚醒清除暫存器：**P_WAKEUP_KEYC_CLR(0x882100C0)**

鍵喚醒清除暫存器用來清除鍵喚醒。當一鍵喚醒事件發生時，必須透過該暫存器清除鍵喚醒。

表 4-56 P_WAKEUP_KEYC_CLR(0x882100C0)

位	b31~b1	b0
讀/寫	-	R/W
預設值	-	0
名稱	-	KEYCHG_CLR

KEYCHG_CLR b0 鍵喚醒清除位：

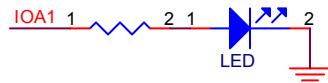
0：清除鍵喚醒

1：使能鍵喚醒

4.6.5 應用舉例

【例 4.7】

利用 Timer 中斷實現 1 個發光二極體電平翻轉（即閃爍），在兩次發光二極體電平翻轉之間的時間間隔裏使系統進入 Wait 節電模式，直到 Timer 中斷。電路連接圖如下所示：



中斷喚醒的一般操作流程如圖 4-22，步驟如下：

- (1) 打開中斷；
- (2) 使能模組時鐘；
- (3) 使能模組；
- (4) 使能模組中斷，清中斷旗標；
- (5) 選擇睡眠模式，給睡眠指令，此時系統進入睡眠模式；注意圖中的等待喚醒中斷信號由硬體操作，同樣軟體不需要做任何操作。
- (6) 喚醒後進入中斷服務程式，進行中斷服務處理。

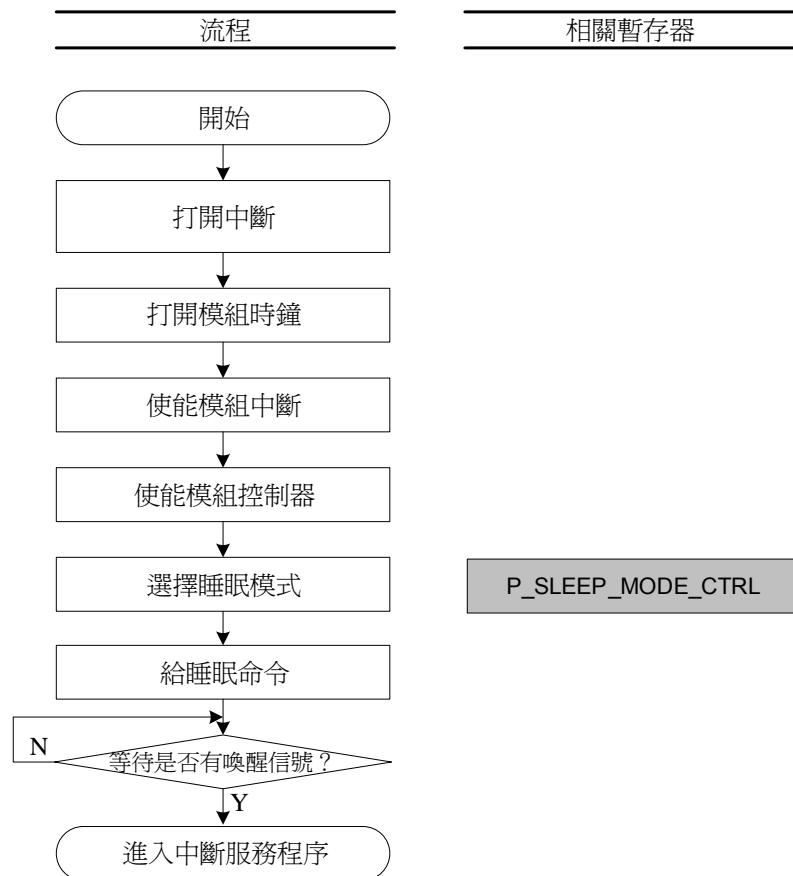


圖 4-22 中斷喚醒的操作流程圖

```

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    *P_INT_MASK_CTRL1 &= ~C_INT_TIMER_DIS;           // 打開 TIMER 中斷
    *P_IOA_GPIO_SETUP = 0x00000202;                   // 使能 IOA1
    *P_TIMER0_CLK_CONF = 0;                          // 重設 TIMER0
    *P_TIMER0_CLK_CONF = C_TIMER_CLK_EN
        | C_TIMER_RST_DIS;                         // 使能 TIMER0
    *P_TIMER_CLK_SEL = C_TIMER0_CLK_27M | 247;       // 選擇 TIMER0 時鐘，分頻 248
    *P_TIMER0_PRELOAD_DATA = 0x80;                  // 載入計數初值
    *P_TIMER0_CCP_CTRL = C_TIMER_NOR_MODE;          // 選擇定時工作模式
    *P_TIMER0_MODE_CTRL = C_TIMER_CTRL_EN
        | C_TIMER_INT_EN | C_TIMER_INT_FLAG;        // 使能中斷
    while(1)
    {
        *P_SLEEP_MODE_CTRL = C_SLEEP_WAIT_MODE;      // 關閉 CPU 時鐘，wait 模式
        __asm("sleep");                            // 進入睡眠模式
    }
    return 0;
}
//=====
// 語法格式：void IRQ56(void)

```

```
// 功能描述：計時器（Timer）中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ56(void)
{
    *P_TIMER0_MODE_CTRL |= C_TIMER_INT_FLAG;           // 清除中斷
    *P_IOA_GPIO_SETUP ^= 0x00000002;                   // IOA 輸出反相
}
```

4.7 ADC

4.7.1 概述

SPCE3200 具有 1 個 12 位 9 路的 ADC (A/D Converter)，該 ADC 不僅具有普通 A/D 轉換功能，而且具有語音錄製專用 A/D 轉換功能，其中：有 1 路 MIC 專用輸入通道，8 路通用 A/D 輸入通道。

當 ADC 作為通用 A/D 使用時，支援可編程取樣頻率的自動取樣方式，同時也支援透過取樣命令控制的手動取樣方式；所有 8 個普通 A/D 輸入通道都透過中斷讀取轉換資料。

MIC 輸入通道支援 PGA，透過設置暫存器控制其增益；不同於通用 A/D 轉換通道，MIC 模式下僅能透過 DMA 傳輸資料。

SPCE3200 的 ADC 時鐘源來自 PLLA 和 PLLU，經過分頻後作為 ADC 的時鐘，無論是時鐘源還是分頻倍數都可以透過軟體編程選擇。

4.7.2 特性

SPCE3200 的 ADC 具有如下特性：

- 1 個 12 位 9 通道類比/數位轉換器
- 具有通用 A/D 和音頻雙重功能
- 測量電壓：0~VRT (參考電壓)
- 片上集成前級放大器和 MIC 增益可編程放大器
- 片上集成抗混濁濾波器和 MIC 偏壓輸出模組
- 集成參考電壓模組和偏壓電流/電壓產生器
- 掉電模式消耗電流：1uA
- 通用 A/D 消耗電流：2.0mA
- 音頻 MIC 消耗電流：4.0mA (不帶 MIC 偏壓輸出)，7.5mA (帶 MIC 偏壓輸出)

4.7.3 插腳描述

SPCE3200 的 A/D 插腳如表 4-57。

表 4-57 SPCE3200 的 A/D 插腳列表

插腳名稱	插腳號	插腳屬性	功能描述
ADC_CH0	160	I	ADC 通道 0 類比輸入腳
ADC_CH1	161	I	ADC 通道 1 類比輸入腳
ADC_CH2	162	I	ADC 通道 2 類比輸入腳
ADC_CH3	163	I	ADC 通道 3 類比輸入腳
ADC_CH4	164	I	ADC 通道 4 類比輸入腳
ADC_CH5	165	I	ADC 通道 5 類比輸入腳
ADC_CH6	166	I	ADC 通道 6 類比輸入腳
ADC_CH7	167	I	ADC 通道 7 類比輸入腳
ADC_ADVRT	169	I	ADC 參考電壓輸入腳
ADC_ADFLT	172	O	反鋸齒波輸出腳
ADC_VREF	173	O	AFE 參考電壓輸出腳
ADC_MICIN	174	I	MIC 輸入埠
ADC_MICBIAS	175	O	緩存適配電壓為 MIC 提供偏壓。電壓為 AVDD33 的 3/4
ADC_TESTN	176	I/O	測試模式的負輸入或輸出
ADC_TESTP	177	I/O	測試模式的正輸入或輸出

4.7.4 結構框圖

SPCE3200 ADC 的結構框圖如圖 4-23。SPCE3200 的 ADC 時鐘源來自 PLLA 和 PLLU，經過分頻後作為 ADC 的時鐘；9 個 ADC 通道可編程選擇；A/D 轉換控制器控制 A/D 轉換；轉換完成後通用 A/D 通道的轉換資料透過中斷控制讀取，而 MIC 通道的轉換資料只能透過 DMA 讀取。

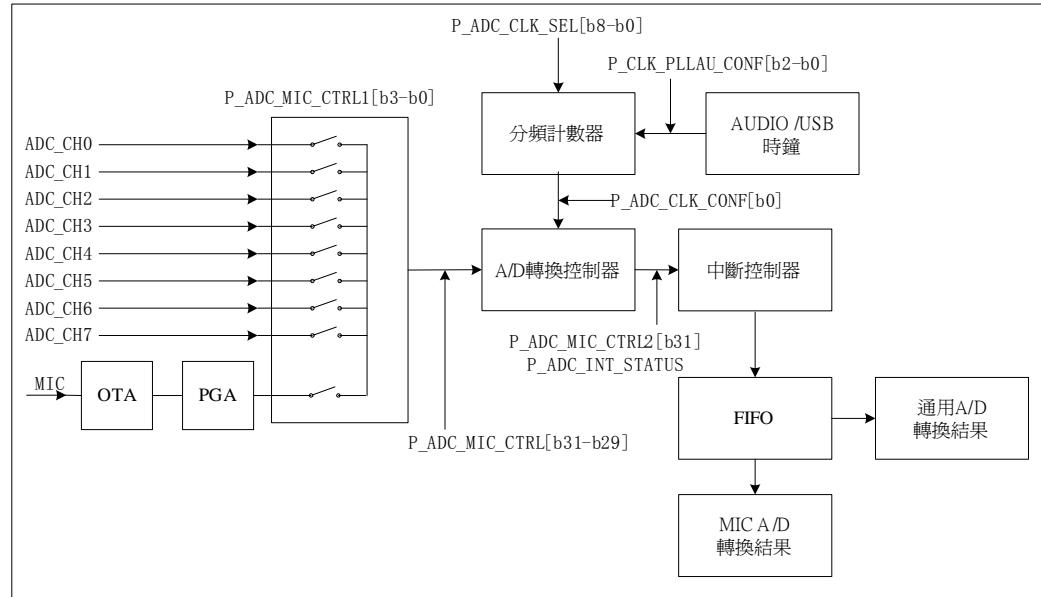


圖 4-23 SPC3200 ADC 的結構框圖

4.7.5 暫存器描述

SPCE3200 的 ADC 相關暫存器共有 16 個，如表 4-58。

表 4-58 ADC 相關暫存器列表

暫存器名稱	助記符	位址
ADC GPIO 設置暫存器	P_ADC_GPIO_SETUP	0x88200048
ADC GPIO 輸入資料暫存器	P_ADC_GPIO_INPUT	0x88200078
ADC GPIO 外部中斷暫存器	P_ADC_GPIO_INT	0x8820009C
ADC 類比輸入控制暫存器	P_ADC_AINPUT_CTRL	0x88200054
PLL AU 時鐘配置暫存器	P_CLK_PLLAU_CONF	0x882100BC
ADC 時鐘配置暫存器	P_ADC_CLK_CONF	0x882100AC
ADC 時鐘選擇暫存器	P_ADC_CLK_SEL	0x882100B0
ADC 控制暫存器 1	P_ADC_MIC_CTRL1	0x881A0000
MIC 增益暫存器	P_ADC_MIC_GAIN	0x881A0004
ADC 取樣時鐘暫存器	P_ADC_SAMPLE_CLK	0x881A0008
ADC 取樣保持暫存器	P_ADC_SAMPLE_HOLD	0x881A000C
ADC 控制暫存器 2	P_ADC_MIC_CTRL2	0x881A0010
ADC 中斷狀態暫存器	P_ADC_INT_STATUS	0x881A0014

暫存器名稱	助記符	位址
ADC 手動方式資料暫存器	P_ADC_MANUAL_DATA	0x881A0018
ADC 自動方式資料暫存器	P_ADC_AUTO_DATA	0x881A001C
MIC 轉換資料暫存器	P_ADC_MIC_DATA	0x881A0020

如果 ADC 的類比輸入埠複用為 GPIO 功能，可以對表 4-58 中前三個暫存器操作。暫存器的各位對應插腳關係如表 4-142：

表 4-59 ADC 類比輸入埠複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
ADC_CH0	160	bit[0]	可以
ADC_CH1	161	bit[1]	可以
ADC_CH2	162	bit[2]	可以
ADC_CH3	163	bit[3]	可以
ADC_CH4	164	bit[4]	不可以
ADC_CH5	165	bit[5]	不可以
ADC_CH6	166	bit[6]	不可以
ADC_CH7	167	bit[7]	不可以

■ ADC GPIO 設置暫存器：P_ADC_GPIO_SETUP(0x88200048)

ADC_CH0~ADC_CH7 除可以用來作為專用的 ADC 類比信號輸入埠外，還可以用來做 GPIO。該暫存器控制 ADC_CH0~ADC_CH7 作為 GPIO 時的輸出使能、上拉電阻輸入/下拉電阻輸入設置和輸出資料。

表 4-60 P_ADC_GPIO_SETUP(0x88200048)

位	b31~b24	b23~b16	b15~b8	b7~b0
讀/寫	R/W	R/W	R/W	R/W
預設值	0xFF	0xFF	0x00	0x00
名稱	ADC_PD	ADC_PU	ADC_OE	ADC_O

ADC_PD	b31~b24	ADC GPIO 下拉電阻輸入埠使能位： 0：禁止為下拉電阻輸入埠 1：使能為下拉電阻輸入埠
ADC_PU	b23~b16	ADC GPIO 上拉電阻輸入埠使能位： 0：使能為上拉電阻輸入埠 1：禁止為上拉電阻輸入埠
ADC_OE	b15~b8	ADC GPIO 輸出使能位： 0：禁止為輸出埠 1：使能為輸出埠
ADC_O	b7~b0	ADC GPIO 輸出資料

說明：

暫存器 P_ADC_GPIO_SETUP 的 bit0、bit8、bit16、bit24 控制 ADC_CH0，按順序 bit7、bit15、bit23、bit31 控制 ADC_CH7。

■ ADC GPIO 輸入資料暫存器 P_ADC_GPIO_INPUT(0x88200078)

ADC_CH0~ADC_CH7 作為通用輸入埠時的輸入資料暫存器。

表 4-61 P_ADC_GPIO_INPUT(0x88200078)

位	b31~b8	b7~b0
讀/寫	-	R
預設值	-	0x00
名稱	-	ADC_INPUT

ADC_INPUT	b7~b0	輸入資料
-----------	-------	------

說明：

b0 對應 ADC_CH0 輸入資料，b7 對應 ADC_CH7 輸入資料。

■ ADC GPIO 外部中斷暫存器：P_ADC_GPIO_INT(0x8820009C)

ADC_CH0~ADC_CH3 作為外部中斷輸入時的控制暫存器，控制外部中斷的使能和中斷旗標位元的清除。

表 4-62 P_ADC_GPIO_INT(0x8820009C)

位	b31~b28	b27~b24	b23~b20	b19~b16	b15~b12	b11~b8	b7~b4	b3~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	0	-	0	-	0	-	0
名稱	-	ADC_FI	-	ADC_RI	-	ADC_FIEN	-	ADC RIEN

ADC_FI b27~b24 ADC GPIO 下降沿中斷事件旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

ADC_RI b19~b16 ADC GPIO 上升沿中斷事件旗標位：

讀 0：沒有發生上升沿中斷

讀 1：發生上升沿中斷

寫 0：無意義

寫 1：清除中斷旗標

ADC_FIEN b11~b8 ADC GPIO 下降沿中斷使能位：

0：禁止下降沿中斷

1：使能下降沿中斷

ADC_RIEN b3~b0 ADC GPIO 上升沿中斷使能位：

0：禁止上升沿中斷

1：使能上升沿中斷

注意：

只有 ADC_CH0~ ADC_CH3 可以作為外部中斷的輸入埠，詳細可參考 GPIO 章節。

■ ADC 類比輸入控制暫存器：**P_ADC_AINPUT_CTRL(0x88200054)**

ADC 的介面 ADC_CH0~ADC_CH7 作為類比信號輸入埠時，需要先透過 P_ADC_AINPUT_CTRL 暫存器使能。

表 4-63 P_ADC_AINPUT_CTRL(0x88200054)

位	b31~b8	b7~b0
讀/寫	-	R/W
預設值	-	0x00
名稱	-	ADC_GPIO_AEN

ADC_GPIO_AEN

b7~b0

ADC 類比輸入埠使能位：

0：禁止為類比輸入埠

1：使能為類比輸入埠

說明：

b0 對應 ADC_CH0，b7 對應 ADC_CH7。

■ PLLAU 設置暫存器：**P_CLK_PLLAU_CONF(0x882100BC)**

P_CLK_PLLAU_CONF 暫存器的介紹請參考鎖相環與時鐘發生器章節。

■ ADC 時鐘配置暫存器：**P_ADC_CLK_CONF(0x882100AC)**

透過 ADC 時鐘配置暫存器可以停止/打開 ADC 時鐘或者重設/不重設 ADC 模組，在使能 ADC 前需先打開 ADC 時鐘。

表 4-64 P_ADC_CLK_CONF(0x882100AC)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	ADC_RST	ADC_STOP

ADC_RST	b1	ADC 模組時鐘重設位： 0 : ADC 模組時鐘重設 1 : ADC 模組時鐘不重設
ADC_STOP	b0	ADC 模組時鐘使能位： 0 : ADC 模組時鐘停止 1 : ADC 模組時鐘使能

■ ADC 時鐘選擇暫存器：P_ADC_CLK_SEL(0x882100B0)

SPCE3200 ADC 的時鐘由 PLLA 或者 PLLU 兩種時鐘源可供選擇，透過 P_ADC_CLK_SEL 暫存器來選擇 PLLA 作為 ADC 的時鐘源，還是 PLLU 作為 ADC 的時鐘源。

表 4-65 P_ADC_CLK_SEL(0x882100B0)

位	b31~b29	b8	b7~b0
讀/寫	-	W	W
預設值	-	0	0x01
名稱	-	ADC_Source	DividedNUM

ADC_Source	b8	ADC 時鐘源選擇位： 0 : 選擇 PLLA 作為 ADC 的時鐘源 1 : 選擇 PLLU 作為 ADC 的時鐘源
DividedNUM	b7~b0	ADC 時鐘設置位，ADC 時鐘為時鐘源的 (DividedNUM+1) 分頻。 DividedNUM!=0

■ ADC 控制暫存器 1 : P_ADC_MIC_CTRL1(0x881A0000)

SPCE3200 的 ADC 有 9 個通道，兩種功能：通用 ADC 和音頻專用 ADC (即 MIC)；透過 ADC 控制暫存器可以使能通用 ADC 或者 MIC、選擇 A/D 通道。

表 4-66 P_ADC_MIC_CTRL1(0x881A0000)

位	b31	b30	b29	b27	b26	b25~b4	b3~b0
讀/寫	R/W	R/W	R/W	R/W	R/W	-	R/W
預設值	0	0	0	0	0	-	0
名稱	ADC_SEL	ADC_EN	MIC_EN	MIC_BOOST	MIC_BIAS	-	CH_SEL

ADC_SEL	b31	通用 ADC 選擇位： 0 : 不選擇通用 ADC 1 : 選擇通用 ADC
ADC_EN	b30	通用 ADC 使能位： 0 : 禁止通用 ADC 1 : 使能通用 ADC
MIC_EN	b29	MIC 使能位： 0 : 禁止 MIC 1 : 使能 MIC
MIC_BOOST	b27	MIC 前級放大器使能位 (僅對 MIC 模式有用)： 0 : 禁止 MIC 前級放大器 1 : 使能 MIC 前級放大器
MIC_BIAS	b26	MIC 偏壓使能位： 0 : 禁止 MIC 偏壓 1 : 使能 MIC 偏壓
CH_SEL	b3~b0	通道選擇位： 0000 : 選擇通用 A/D 通道 0(ADC_CH0) 0001 : 選擇通用 A/D 通道 1(ADC_CH1) 0010 : 選擇通用 A/D 通道 2(ADC_CH2) 0011 : 選擇通用 A/D 通道 3(ADC_CH3) 0100 : 選擇通用 A/D 通道 4(ADC_CH4) 0101 : 選擇通用 A/D 通道 5(ADC_CH5) 0110 : 選擇通用 A/D 通道 6(ADC_CH6)

0111 : 選擇通用 A/D 通道 7(ADC_CH7)

1xxx : 選擇 MIC 通道

■ MIC 增益暫存器：P_ADC_MIC_GAIN(0x881A0004)

SPCE3200 ADC 的 MIC 通道支援 PGA，透過 MIC 增益暫存器來選擇設置其增益。

表 4-67 P_ADC_MIC_GAIN(0x881A0004)

位	b31~b5	b4~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	MIC_GAIN

MIC_GAIN

b4~b0

MIC 增益值選擇：

00000 : 33.0db

00001 : 31.5db

00010 : 30.0db

00011 : 28.5db

.....

10101 : 1.5db

10110 : 0.0db

10111 : -1.5db

.....

11110 : -12.0db

11111 : ∞

■ ADC 取樣時鐘暫存器 P_ADC_SAMPLE_CLK(0x881A0008)

不同於其他的 ADC，SPCE3200 的 ADC 可透過 P_ADC_SAMPLE 編程設置其時鐘週期，和 ADC 取樣保持暫存器結合設置其取樣率。

表 4-68 P_ADC_SAMPLE_CLK(0x881A0008)

位	b31~b24	b15~b0
讀/寫	-	R/W
預設值	-	0x0000
名稱	-	ADC_CLOCK_CYCLE

 ADC_CLOCK_CYCLE b15~b0 ADC 時鐘週期 = ADC_CLOCK_CYCLE+1

注意：

只有自動取樣方式時需要設置其時鐘取樣週期。

■ ADC 取樣保持暫存器：P_ADC_SAMPLE_HOLD(0x881A000C)

SPCE3200 的 ADC 作通用 A/D 使用時，支援可編程取樣頻率的自動取樣方式，同時也支援透過取樣命令控制的手動取樣方式；透過 P_ADC_SAMPLE_HOLD 可設置取樣頻率。

表 4-69 P_ADC_SAMPLE_HOLD(0x881A000C)

位	b31~b28	b27~b16	b15~b0
讀/寫	W	-	R/W
預設值	0	-	0x00
名稱	HOLD_WIDTH	-	HOLD_CYLCE

 HOLD_WIDTH b31~b28 取樣保持帶寬設置：HOLD_WIDTH>=2

 HOLD_CYLCE b15~b0 取樣保持週期設置：HOLD_CYLCE >=15

說明：

使能 ADC 時鐘之後，系統會自動把 P_ADC_SAMPLE_HOLD 設置為 0x200f，即 HOLD_CYLCE 為 15，HOLD_WIDTH 為 2。

注意，只有自動取樣方式時需要設置其時鐘取樣週期。關於設置 ADC 時鐘和取樣保持寬度/週期數的方法舉例來說，假設需要 44.1KHz 的 ADC 取樣率，且 ADC 控制器時鐘為 33.8688 MHz，則取樣週期數即為 $33.8688M / 44.1K = 768$ ，且若需取樣保持週期數為 16，則最終：

ADC 時鐘週期數 = $768 / 16 = 48$;

因此，ACC (ADC_CLOCK_CYCLE) = 47，取樣保持週期 HOLD_CYCLE = 15，取樣保持寬度 HOLD_WIDTH = 2

■ ADC 控制暫存器 2 : P_ADC_MIC_CTRL2(0x881A0010)

透過 ADC 控制暫存器 2 可以使能 ADC 控制器、選擇通用 A/D 的取樣方式等。

表 4-70 P_ADC_MIC_CTRL2(0x881A0010)

位	b31	b27	b26	b25
讀/寫	R/W	R/W	R/W	R/W
預設值	0	0	0	0
名稱	ADC_EN	AUTO_SAMPLE	MIC_MODE	MUTE_SEL
位	b24	b23	b6~b4	b2~b0
讀/寫	R/W	R/W	R/W	R/W
預設值	0	0	0	0
名稱	DMA_SIZE	AUTO_CLR	FIFO_INT	FIFO_RD

ADC_EN	b31	ADC 控制器使能位： 0 : 禁止 ADC 控制器 1 : 使能 ADC 控制器
AUTO_SAMPLE	b27	通用 A/D 取樣方式選擇： 0 : 選擇手動取樣方式 1 : 選擇自動取樣方式
MIC_MODE	b26	MIC 模式資料類型選擇 (無符號數/有符號數)： 0 : 有符號數 1 : 無符號數
MUTE_SEL	b25	MIC 模式的靜音選擇位： 0 : 正常選擇 1 : 選擇靜音

DMA_SIZE	b24	MIC 模式 DMA 資料大小的選擇： 0：32 位 1：16 位
AUTO_CLR	b23	通用 A/D 模式自動清除中斷旗標使能位： 0：禁止通用 A/D 模式自動清除中斷旗標 1：使能通用 A/D 模式自動清除中斷旗標
FIFO_INT	b6~b4	中斷時 FIFO 接收的資料量設置： 000：接收到 1 個資料時中斷 001：接收到 2 個資料時中斷 010：接收到 3 個資料時中斷 011：接收到 4 個資料時中斷 100：接收到 5 個資料時中斷 101：接收到 6 個資料時中斷 110：接收到 7 個資料時中斷 111：接收到 8 個資料時中斷
FIFO_RD	b[2:0]	FIFO 讀接收資料量設置

■ ADC 中斷狀態暫存器：P_ADC_INT_STATUS(0x881A0014)

透過 ADC 中斷狀態暫存器可以使能通用 ADC 中斷、MIC 溢出中斷、FIFO 接收溢出中斷等。

表 4-71 P_ADC_INT_STATUS(0x881A0014)

位	b31	b30	b29	b28	b27
讀/寫	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0
名稱	ADC_INT	ADC_INTEN	ADC_AINT	ADC_AINTEN	MIC_OFINT
位	b26	b23	b22	b21	b15
讀/寫	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0
名稱	MIC_OFINTEN	FIFO_OFE	FIFO_FULL	FIFO_EMPTY	CONVERT_ADC

ADC_INT	b31	通用 A/D 手動取樣方式中斷旗標位： 讀 0：發生通用 A/D 手動取樣方式中斷 讀 1：沒有發生通用 A/D 手動取樣方式中斷 寫 0：無意義 寫 1：清中斷旗標
ADC_INTEN	b30	通用 A/D 手動取樣方式中斷使能位： 0：禁止通用 A/D 手動取樣方式中斷 1：使能通用 A/D 手動取樣方式中斷
ADC_AINT	b29	通用 A/D 自動取樣方式中斷旗標位： 讀 0：發生通用 A/D 自動取樣方式中斷 讀 1：沒有發生通用 A/D 自動取樣方式中斷 寫 0：無意義 寫 1：清中斷旗標
ADC_AINTEN	b28	通用 A/D 自動取樣方式中斷使能位： 0：禁止通用 A/D 自動取樣方式中斷 1：使能通用 A/D 自動取樣方式中斷
MIC_OFINT	b27	MIC 溢出中斷旗標位： 讀 0：發生 MIC 溢出中斷 讀 1：沒有發生 MIC 溢出中斷 寫 0：無意義 寫 1：清中斷旗標
MIC_OFINTEN	b26	MIC 溢出中斷使能位： 0：禁止 MIC 溢出中斷 1：使能 MIC 溢出中斷
FIFO_OFE	b23	FIFO 接收溢出錯誤旗標位： 讀 0：發生溢出錯誤 讀 1：沒有發生溢出錯誤 寫 0：無意義 寫 1：清溢出錯誤旗標
FIFO_FULL	b22	FIFO 滿旗標：

		0 : 非滿
		1 : 滿
FIFO_EMPTY	b21	FIFO 空旗標： 0 : 非空 1 : 空
CONVERT_ADC	b15	通用 A/D 手動方式啓動轉換位： 0 : 不啓動手動轉換 1 : 啓動手動轉換

■ ADC 手動方式資料暫存器：P_ADC_MANUAL_DATA(0x881A0018)

P_ADC_MANUAL_DATA 暫存器用來保存通用 A/D 手動取樣方式轉換的資料，結果保存在暫存器的高 12 位。

表 4-72 P_ADC_MANUAL_DATA(0x881A0018)

位	b31~b20	b19~b0
讀/寫	R	-
預設值	0	-
名稱	ADC_SELF_DATA	-

ADC_SELF_DATA	b31~b20	通用 A/D 手動取樣方式轉換資料，保存在暫存器的高 12 位
---------------	---------	---------------------------------

■ ADC 自動方式資料暫存器：P_ADC_AUTO_DATA(0x881A001C)

P_ADC_AUTO_DATA 暫存器用來保存通用 A/D 自動取樣方式轉換的資料，結果保存在暫存器的高 12 位。

表 4-73 P_ADC_AUTO_DATA(0x881A001C)

位	b31~b20	b19~b0
讀/寫	R	-
預設值	0	-
名稱	ADC_AUTO_DATA	-

ADC_AUTO_DATA	b31~b20	通用 A/D 自動取樣方式轉換資料，保存在暫存器的高 12 位
---------------	---------	---------------------------------

■ MIC 轉換資料暫存器：P_ADC_MIC_DATA(0x881A0020)

P_ADC_MIC_DATA 暫存器用來保存透過 MIC 通道輸入 A/D 轉換的資料。其中低位元組保存 1 個 16 位的 MIC 轉換資料，高位元組保存 1 個 16 位的 MIC 轉換資料。2 個資料按照取樣頻率轉換存儲。其中轉換的資料是 12 位的，經過插值演算法擴展為 16 位的資料。

表 4-74 P_ADC_MIC_DATA(0x881A0020)

位	b31~b0
讀/寫	R
預設值	0x00000000
名稱	MIC_DATA

MIC_DATA	b31~b0	MIC 通道輸入 A/D 轉換的資料
----------	--------	--------------------

4.7.6 基本操作

ADC 的基本操作主要包括 ADC 時鐘的設置和 ADC 的啟動轉換，同時會介紹通用 ADC 的中斷。

1. ADC 時鐘的設置

SPCE3200 ADC 的時鐘源有兩種：PLLA 或者 PLLU，其時鐘是由這兩種時鐘源經過分頻提供的，分頻倍數可軟體設置。

SPCE3200 ADC 時鐘設置流程如圖 4-24。

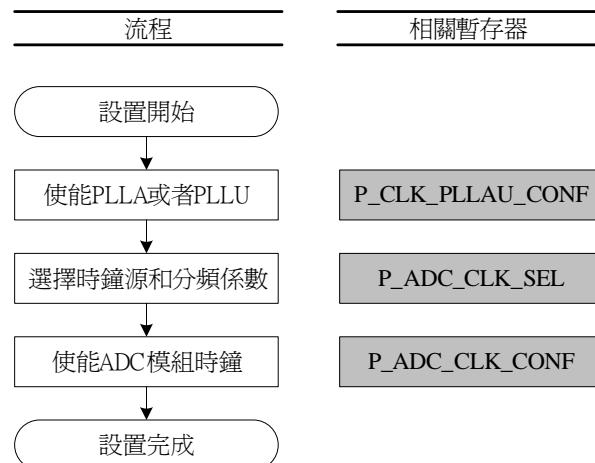


圖 4-24 SPCE3200 ADC 時鐘設置流程圖

程式段參考如下：

```
*P_ADC_CLK_CONF = C_ADC_CLK_EN
                | C_ADC_RST_DIS;           // ADC 時鐘模組設置
*P_CLK_PLLAU_CONF = C_PLLA_CLK_EN
                | C_PLLA_CLK_67M;         // 使能 PLLA
                | C_PLLA_CLK_67M;         // 頻率選擇為 67MHz
*P_ADC_CLK_SEL = C_ADC_SEL_PLLA
                | 0x01;                  // ADC 時鐘源選擇 PLLA
                | 0x01;                  // ADC 時鐘頻率選擇為 PLLA 的 2 分頻
```

2. ADC 的啓動

SPCE3200 的 ADC 有兩種功能：通用 A/D 和專用 MIC 輸入 A/D。

通用 A/D 包括自動取樣方式和手動取樣方式。

通用 A/D 的自動取樣方式啓動流程如圖 4-25。步驟如下：

- (1) 使能 A/D 的資料埠為類比輸入埠；
- (2) 設置為通用 A/D 模式，選擇 A/D 通道；
- (3) 設置取樣保持週期及帶寬；
- (4) 使能 ADC 中斷；
- (5) 使能 ADC 控制器並選擇為自動取樣方式。

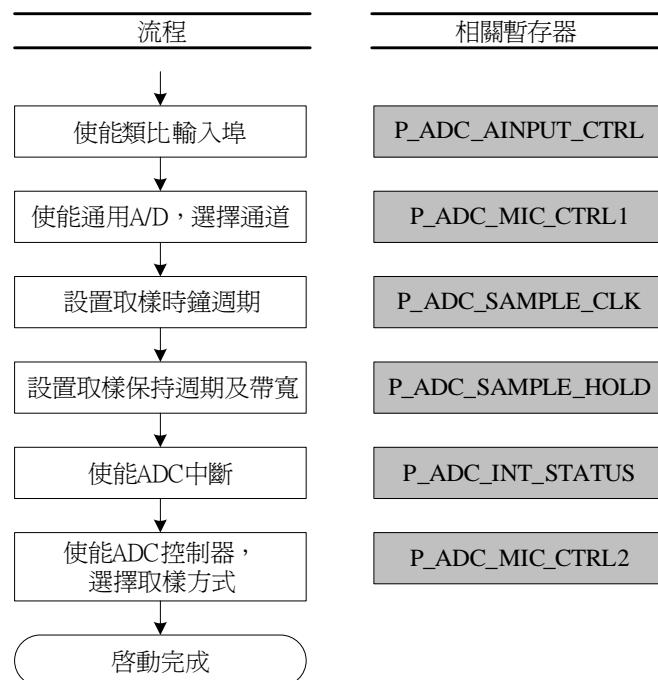


圖 4-25 通用 A/D 的自動取樣方式啓動流程圖

程式段參考如下：

```

*P_ADC_AINPUT_CTRL = C_ADC_CH0_EN;           // ADC 的 channel 0 類比輸入使能
*P_ADC_MIC_CTRL1 = C_ADC_MODE_ADC           // 選擇 ADC 模式
| C_ADC_CH0_SEL                           // 選擇 channel 0
;
*P_ADC_SAMPLE_CLK = 0x00000017;             // 設定 ADC clock cycle 為 24 即(23+1)
*P_ADC_SAMPLE_HOLD = 0x2000000f;            // 設定取樣保持 cycle 為 16, 帶寬為 2
*P_ADC_INT_STATUS = C_ADC_AUTO_INTEN       // 使能 ADC 自動方式的中斷
| C_ADC_AUTO_FLAG;                         // 清中斷旗標
*P_ADC_MIC_CTRL2 = C_ADC_CTRL_EN           // 使能 ADC 控制器
| C_ADC_AUTO_MODE;                         // 選擇自動取樣模式
| C_ADC_AUTO_CLR;                          // 自動清除中斷旗標
;
    
```

通用 A/D 的手動取樣方式啓動流程如圖 4-26。注意手動取樣方式不需要設置 ACC 和取樣保持週期及帶寬；但是要啓動手動轉換後才開始一次 A/D 轉換。



圖 4-26 通用 A/D 的手動取樣方式啓動流程圖

程式段參考如下：

```

*P_ADC_AINPUT_CTRL = C_ADC_CH0_EN;           // ADC 的 channel 0 類比輸入使能
*P_ADC_MIC_CTRL1 = C_ADC_MODE_ADC
| C_ADC_CH0_SEL
;
*P_ADC_INT_STATUS = C_ADC_MANUAL_INTEN      // 使能 ADC 自動方式的中斷
| C_ADC_MANUAL_FLAG;                      // 清中斷旗標
*P_ADC_MIC_CTRL2 = C_ADC_CTRL_EN
| C_ADC_MANUAL_MODE
| C_ADC_AUTO_CLR;                         // 使能 ADC 控制器
                                         // 選擇自動取樣模式
                                         // 自動清除中斷旗標
;
*P_ADC_INT_STATUS |= C_ADC_MANUAL_START;     // 啓動 AD 轉換
    
```

MIC 啓動流程如圖 4-27。步驟如下：

- (1) 使能 MIC，選擇為 MIC 通道；
- (2) 設置 MIC 增益；
- (3) 使能 MIC 中斷；
- (4) 使能 ADC 控制器。

注意這裏只是 ADC 部分的設置，由於 MIC 輸入的轉換資料只能透過 DMA 讀取，所以還要對 DMA 控制器進行設置，詳見 DMA 相關章節。

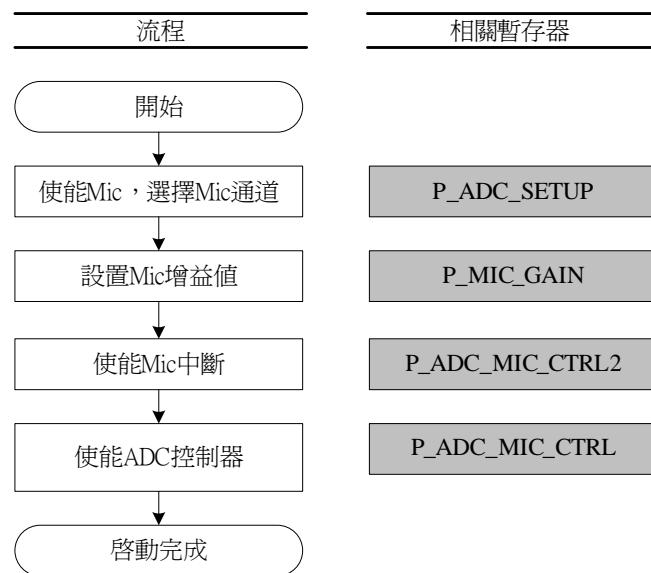


圖 4-27 通用 A/D 的啓動流程圖

程式段參考如下：

```

*P_ADC_MIC_CTRL1 = C_ADC_MODE_MIC           // 使能 MIC
| C_ADC_MIC_SEL          // 選擇 MIC 通道
| C_MIC_BIAS_EN          // MIC 偏壓使能
| C_MIC_BOOST_EN         // MIC 前級放大器使能
;
*P_ADC_MIC_GAIN = C_MIC_GAIN_330;           // 增益值設置為 33.0db
*P_ADC_INT_STATUS = C_ADC_MICOV_INTEN      // 使能 MIC 溢出中斷
| C_ADC_MICOV_FLAG;                  // 清 MIC 溢出中斷旗標
*P_ADC_MIC_CTRL2 = C_ADC_CTRL_EN;           // 使能 ADC 控制器
    
```

3. 中斷

SPCE3200 的 40 個中斷源中，其中有兩個關於 ADC 的中斷源：通用 A/D 中斷和 MIC 溢出 /FIFO 溢出中斷，這兩個中斷都透過 P_ADC_INT_STATUS 暫存器使能，進入中斷後透過 P_ADC_INT_STATUS 暫存器清除中斷旗標。

說明：

- 當 ADC 作為通用 A/D 使用時，如果透過 P_ADC_MIC_CTRL2 暫存器的 bit23 把自動清除中斷旗標位元使能，可不用專門的清除中斷旗標操作。
- 當 ADC 作為通用 A/D 使用時，完成一次 A/D 轉換，引發一次 A/D 中斷，這樣，可以根據 P_ADC_INT_STATUS 的 bit31 或者 bit29 判斷 A/D 轉換是否完成。

中斷方式中斷服務程式段：

```

//=====
// 語法格式：void IRQ58(void)
// 功能描述：通用 ADC 中斷服務函數
// 入口參數：無
    
```

```

// 出口參數：無
//=====
void IRQ58(void)
{
    unsigned int Data = 0;
    float vDataConvert = 0.0000;
    *P_ADC_INT_STATUS |= C_ADC_AUTO_FLAG;           // 清中斷旗標
    Data = *P_ADC_AUTO_DATA;                        // 取出轉換資料
    Data = Data >> 20;                            // 把轉換資料移到低 12 位
    vDataConvert = Data * 3.3 / 0xffff;             // 計算電壓值
}

```

4.7.7 注意事項

使用 SPCE3200 的 ADC 時需要注意下面幾點：

- 通用 A/D 輸入通道輸入的最大電壓不能大於參考電壓
- 參考電壓不能超過 3.3V (I/O 埠電壓選擇為 3.3V 時)

4.8 UART

4.8.1 概述

SPCE3200 具有一個標準的通用非同步串列通訊模組 UART (Universal Asynchronous Receiver/Transmitter)，使其與 MCU 之間具有更強的通訊能力，而且通訊更加靈活。

SPCE3200 的 UART 資料框格式如圖 4-28，可以看出，在 UART 的資料傳輸時，先傳輸低位元後傳輸高位元；資料位元數可以設置；可以編程設置奇偶檢查位。



圖 4-28 UART 資料框格式

4.8.2 特性

SPCE3200 具有如下特性：

- 最大串列傳輸速率可達到 460.8Kbps
- 可編程設置發送等待時間和接收等待（潛伏）時間
- 內嵌 2 位元組發送 FIFO 和 8 位元組接收 FIFO
- 發送、接收和接收超時中斷可獨立遮罩或者使能
- 錯誤啓動位元檢測
- 連接中止產生及檢測

- 完全可編程串列介面：
 - 資料位元可被設置為 5bit、6bit、7bit 或者 8bit
 - 奇、偶、無檢查產生和檢測
 - 1 位或者 2 位的停止位產生

4.8.3 插腳描述

UART 插腳的描述如表 4-75。

表 4-75 UART 插腳描述

插腳名稱	插腳號	插腳屬性	插腳功能
UART_TX	39	O	UART 發送腳
UART_RX	40	I	UART 接收腳

注意：

在與 PC 機通訊時，需要外接電平轉換電路。

4.8.4 結構框圖

UART 的結構框圖如圖 4-29，UART 主要由 UART 時鐘、中斷控制器、FIFO（2 個位元組的發送 FIFO 和 8 位元組的接收 FIFO）、串列傳輸速率發生器和 UART 發送/接收控制器構成。UART 時鐘為 UART 模組的各個單元提供時鐘；中斷控制器控制使能發送或者接收中斷；FIFO 為發送或者接收資料的緩存，當發送資料時，先把發送資料填入 FIFO，經移位暫存器一位一位移入 UART 控制器控制發送，當接收資料時，移位暫存器先把接收資料一位一位的填入 FIFO，再透過中斷控制讀取接收資料；串列傳輸速率發生器提供發送或者接收資料的速率，當 SPCE3200 向其他 MCU 發送資料或者從其他 MCU 接收資料時，必須和其他 MCU 的串列傳輸速率相同；UART 發送/接收控制器主要控制資料的發送和接收，包括奇偶檢查位的設置、資料/停止位的設置等。

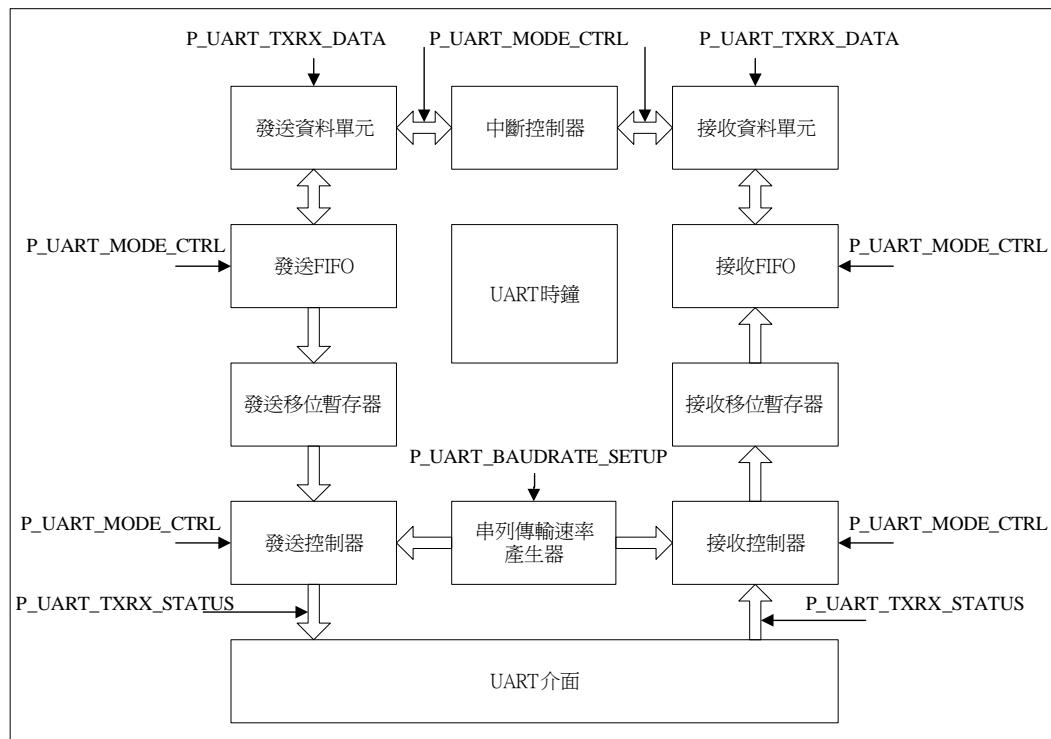


圖 4-29 UART 結構框圖

4.8.5 暫存器描述

UART 模組共有 12 個暫存器，如表 4-76 所示。

表 4-76 UART 相關暫存器列表

暫存器名稱	助記符	位址
UART GPIO 設置暫存器	P_UART_GPIO_SETUP	0x88200040
UART GPIO 輸入資料暫存器	P_UART_GPIO_INPUT	0x88200074
UART GPIO 外部中斷暫存器	P_UART_GPIO_INT	0x88200094
UART 介面選擇暫存器	P_UART_INTERFACE_SEL	0x88200000
UART 時鐘配置暫存器	P_UART_CLK_CONF	0x8821005C
UART 控制暫存器	P_UART_MODE_CTRL	0x88150008
UART 串列傳輸速率設置暫存器	P_UART_BAUDRATE_SETUP	0x8815000C
UART 收發狀態暫存器	P_UART_TXRX_STATUS	0x88150010
UART 錯誤狀態暫存器	P_UART_ERR_STATUS	0x88150004
UART 收發資料暫存器	P_UART_TXRX_DATA	0x88150000
UART 喚醒狀態暫存器	P_UART_WAKEUP_STATUS	0x88210110

如果 UART 介面複用為 GPIO 功能，可以對表 4-76 的前三個暫存器操作。暫存器的各位對應插腳關係如表 4-142：

表 4-77 UART 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
UART_TX	160	bit[0]	可以
UART_RX	161	bit[1]	可以

■ UART GPIO 設置暫存器：P_UART_GPIO_SETUP(0x88200040)

UART GPIO 設置暫存器的功能是：當 UART 的埠作為通用輸入輸出埠（GPIO）使用時設置輸出使能、設置為上/下拉電阻輸入、輸出資料。

表 4-78 P_UART_GPIO_SETUP(0x88200040)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	0x3	-	0	-	0	-	0
名稱	-	XUART_PD	-	XUART_PU	-	UART_OE	-	UART_O

XUART_PD b25~b24 UART GPIO 下拉使能位：

0：禁止為下拉埠

1：使能為下拉埠

XUART_PU b17~b16 UART GPIO 上拉使能位：

0：使能為上拉埠

1：禁止為上拉埠

UART_OE b9~b8 UART GPIO 輸出使能位：

0：禁止位輸出埠

1：使能為輸出埠

UART_O b1~b0 UART GPIO 輸出資料

注意：

b0、b8、b16、b24 控制的是 RX；b1、b9、b17、b25 控制的是 TX。

■ UART GPIO 輸入資料暫存器：P_UART_GPIO_INPUT(0x88200074)

UART GPIO 輸入資料暫存器保存 UART 介面複用為 GPIO 時的外部輸入資料。

表 4-79 P_UART_GPIO_INPUT(0x88200074)

位	b31~b26	b17~b16	b1~b0
讀/寫	-	R	-
預設值	-	0	-
名稱	-	UART_IN	-

UART_IN

b17~b16

UART 作為 GPIO 功能時的輸入資料

■ UART GPIO 外部中斷暫存器：P_UART_GPIO_INT(0x88200094)

當 UART 埠作為 GPIO 時，設置外部中斷使能、中斷觸發沿設置和中斷旗標的清除。

表 4-80 P_UART_GPIO_INT(0x88200094)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	-	W	-	W
預設值	-	0	-	0	-	0	-	0
名稱	-	UART_FI	-	UART_RI	-	UART_FIEN	-	UART RIEN

UART_FI

b25~b24

UART GPIO 下降沿中斷旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

 UART_RI b17~b16 UART GPIO 上升沿中斷旗標位：

讀 0：沒有發生上升沿中斷

讀 1：發生上升沿中斷

寫 0：無意義

寫 1：清除中斷旗標

 UART_FIEN b9~b8 UART GPIO 下降沿中斷使能位：

0：禁止下降沿中斷

1：使能下降沿中斷

 UART_RIEN b1~b0 UART GPIO 上升沿中斷使能位：

0：禁止上升沿中斷

1：使能上升沿中斷

■ UART 時鐘配置暫存器：P_UART_CLK_CONF(0x8821005C)

透過 UART 時鐘配置暫存器可以停止/打開 UART 時鐘或者重設/不重設 UART 模組，在使能 UART 前需先打開 UART 時鐘。

表 4-81 P_UART_CLK_CONF(0x8821005C)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	1
名稱	-	UART_RST	UART_STOP

 UART_RST b1 UART 模組時鐘重設位：

0：UART 模組時鐘重設

1：UART 模組時鐘不重設

 UART_STOP b0 UART 模組時鐘使能位：

0：UART 模組時鐘停止

1：UART 模組時鐘使能

■ UART 介面選擇暫存器：**P_UART_INTERFACE_SEL(0x88200000)**

UART 介面選擇暫存器的功能是：選擇 UART 的埠作為普通輸入輸出埠（GPIO）或者作為 UART 的專用發送/接收埠。

表 4-82 P_UART_INTERFACE_SEL(0x88200000)

位	b31~b25	b24	b23~b0
讀/寫	-	R/W	-
預設值	-	1	-
名稱	-	SW_UART	-

SW_UART b24 UART 的埠功能選擇位：

0：作為 GPIO

1：作為 UART 的發送/接收埠

■ UART 串列傳輸速率設置暫存器：**P_UART_BAUDRATE_SETUP(0x8815000C)**

用來設置 UART 發送或者接收的串列傳輸速率。

表 4-83 P_UART_BAUDRATE_SETUP(0x8815000C)

位	b31~b8	b7~b0
讀/寫	-	W
預設值	-	0xe9(115Kbps)
名稱	-	BAUD_RATE

BAUD_RATE b[7:0] 設置傳輸串列傳輸速率。

串列傳輸速率的計算公式如下：

串列傳輸速率=Fosc/BAUD_RATE -1，其中 Fosc 為 27MHz，BAUD_RATE 為 b[7:0]設置的資料

說明：

由於 SPCE3200 的 UART 最高資料傳輸串列傳輸速率為 460.8Kbps，所以 P_UART_BaudRate 最小可以設置為 Fosc/(460800+1)，最小可以設置為 27000000/(460800+1)=58 (0x3A)。

■ UART 控制暫存器：P_UART_MODE_CTRL(0x88150008)

UART 控制暫存器用來控制 UART 的使能、UART 發送/接收中斷的使能、傳輸資料位元數的選擇、停止位的選擇、奇偶檢查的使能及選擇。

表 4-84 P_UART_MODE_CTRL(0x88150008)

位	b15	b14	b13	b12	b11~b8			
讀/寫	W	W	W	W	-			
預設值	0	0	0	1	-			
名稱	RIEN	TIEN	RT	UEN	-			
位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	-	W		W	W	W	W	W
預設值	-	3		1	0	1	1	0
名稱	-	WLSEL		FEN	SBSEL	PSEL	PEN	SB

RIEN b15 UART 接收中斷使能位：

0 : 遮罩 UART 接收中斷

1 : 使能 UART 接收中斷

TIEN b14 UART 發送中斷使能位：

0 : 遮罩 UART 發送中斷

1 : 使能 UART 發送中斷

RT b13 UART 接收超時中斷使能：

0 : 遮罩 UART 接收超時中斷

1 : 使能 UART 接收超時中斷

UEN b12 UART 使能位：

0 : 禁止

1 : 使能

WLSEL	b6~b5	資料位元長度定義，設置發送/接收一框資料中資料的位元數： 00 : 5bits 01 : 6bits 10 : 7bits 11 : 8bits
FEN	b4	FIFO Buffer 使能位，在發送或者接收的過程中使能 FIFO Buffer，當該位被清零，FIFO 將成為 1 位元組的保持暫存器： 0 : 禁止 1 : 使能
SBSEL	b3	停止位長度選擇。當這一位被設置為 1 時，在發送一框資料的結尾有 2bits 停止位。接收邏輯檢測不到接收資料框中的 2bit 停止位： 0 : 選擇 1bit 停止位 1 : 選擇 2bits 停止位
PSEL	b2	奇偶檢查選擇位： 0 : 選擇奇檢查 1 : 選擇偶檢查 當這一位被設置為 1 時，產生一個偶檢查位並在發送或者接收過程中進行偶檢查檢測； 當這一位被設置為 0 時，產生一個奇檢查位並在發送或者接收過程中進行奇檢查檢測。 該位只有當 P_UART_MODE_CTRL 的 b1 (PEN) 設置為 1 時設置有效。
PEN	b1	奇偶檢查使能位： 0 : 無檢查 1 : 使能檢查

SB

b0

傳輸中止設置位：

0：正常傳輸

1：傳輸中止信號

當這一位被設置為 1 時，完成當前的資料傳輸之後，在發送輸出腳發送一個持續的低電平。這個低電平至少要持續傳輸一框資料所需要的時間，以完成中止過程。在中止的過程中 FIFO 中的內容不被影響。正常使用時，這一位必須被清 0。

■ UART 收發狀態暫存器：P_UART_RX_TX_STATUS(0x88150010)

UART 收發狀態暫存器可以設置 UART 的中斷、FIFO 等狀態。

表 4-85 P_UART_RX_TX_STATUS(0x88150010)

位	b15	b14	b13	b7	b6	b5	b4	b3
讀/寫	R	R	R	R	R	R	R	R
預設值	0	0	0	0	0	0	0	0
名稱	RI	TI	RT	TE	RF	TF	RE	BY

RI

b15

UART 接收中斷旗標位：

0：沒有發生 UART 接收中斷

1：發生 UART 接收中斷

TI

b14

UART 發送中斷旗標位：

0：沒有發生 UART 發送中斷

1：發生 UART 發送中斷

RT

b13

UART 接收超時中斷旗標位：

0：沒有發生 UART 接收超時中斷

1：發生 UART 接收超時中斷

TE	b7	發送 FIFO 空旗標： 0：非空 1：空 該位依賴於 FIFO Buffer 是否使能： <ul style="list-style-type: none">如果 FIFO Buffer 没有使能，當發送保持暫存器為空時該位被置 1；如果 FIFO Buffer 使能，當發送 FIFO 為空時該位被置 1。
RF	b6	接收 FIFO 滿旗標位： 0：非滿 1：滿 該位依賴於 FIFO Buffer 是否使能： <ul style="list-style-type: none">如果 FIFO Buffer 没有使能，當接收保持暫存器為滿時該位被置 1；如果 FIFO Buffer 使能，當接收 FIFO 為滿時該位被置 1。
TF	b5	發送 FIFO 滿旗標位： 0：非滿 1：滿 該位依賴於 FIFO Buffer 是否使能： <ul style="list-style-type: none">如果 FIFO Buffer 没有使能，當發送保持暫存器為慢時該位被置 1；如果 FIFO Buffer 使能，當發送 FIFO 為滿時該位被置 1。
RE	b4	接收 FIFO 空旗標位： 0：非空 1：空 該位依賴於 FIFO Buffer 是否使能： <ul style="list-style-type: none">如果 FIFO Buffer 没有使能，當接收保持暫存器為空時該位被置 1；如果 FIFO Buffer 使能，當接收 FIFO 為空時該位被置 1。

BY

b3

忙旗標位：

0：非忙

1：忙

如果 UART 正在忙於傳輸資料，該位元被置 1，直到所有的資料從移位暫存器傳輸完（包括所有的停止位）。

■ UART 錯誤狀態暫存器 P_UART_ERR_STATUS(0x88150004)

UART 錯誤狀態暫存器用來存放 UART 通訊過程中的各種錯誤旗標，包括溢出錯誤、連接斷開錯誤、奇偶檢查錯誤、框傳輸錯誤等，給該暫存器的相應位元寫 1 可清除錯誤旗標。

表 4-86 P_UART_ERR_STATUS(0x88150004)

位	b31~b16	b15	b14~b4	b3	b2	b1	b0
讀/寫	-	R	-	R/W	R/W	R/W	R/W
預設值	-	1	-	0	0	0	0
名稱	-	RXD	-	OE	BE	PE	FE

RXD

b15

UART 接收信號位元。在接收的過程中，為低電平，接收完成後，被置高電平。

OE

b3

UART 傳輸溢出旗標位元，當 UART 接收到資料且接收 FIFO 已經滿時該旗標被置 1：

讀 0：沒有發生 UART 傳輸溢出

讀 1：發生 UART 傳輸溢出

寫 0：無意義

寫 1：清除溢出錯誤旗標

BE

b2

UART 傳輸中止錯誤旗標位元，當 UART 接收到資料輸入持續低電平一個完整字（起始位+資料位+奇偶檢查位+結束位）傳輸時間，將會檢測到中止並將該旗標置 1：

讀 0：沒有發生傳輸中止錯誤

讀 1：發生傳輸中止錯誤

寫 0：無意義

寫 1：清除中止錯誤旗標

PE	b1	UART 奇偶檢查錯誤旗標位元，當 UART 接收到資料特徵中奇偶檢查位元與透過 P_UART_MODE_CTRL 設置不匹配，該旗標置 1。每次透過資料暫存器讀接收資料時，該位元都會被刷新一次，所以在讀出資料之後必須要檢測該位： 讀 0：沒有發生奇偶檢查錯誤 讀 1：發生奇偶檢查錯誤 寫 0：無意義 寫 1：清除奇偶檢查錯誤旗標
FE	b0	UART 的框傳輸錯誤旗標位元，當 UART 接收到資料沒有一個有效的結束位元，該旗標置 1。每次透過資料暫存器讀接收資料時，該位元都會被刷新一次，所以在讀出資料之後必須要檢測該位： 讀 0：沒有發生框傳輸錯誤 讀 1：發生框傳輸錯誤 寫 0：無意義 寫 1：清除框傳輸錯誤旗標

■ UART 收發資料暫存器：P_UART_TXRX_DATA(0x88150000)

UART 收發資料暫存器用來保存 UART 準備發送的資料或者接收到的資料。

表 4-87 P_UART_TXRX_DATA(0x88150000)

位	b31~b8	b7~b0
讀/寫	-	R/W
預設值	-	0x00
名稱	-	UART_DATA

UART_DATA	b7~b0	讀：UART 接收到的資料 寫：UART 準備發送的資料
-----------	-------	-------------------------------------

■ UART 喚醒狀態暫存器：P_UART_WAKEUP_STATUS(0x88210110)

UART 喚醒狀態暫存器顯示是否發生了 UART 喚醒或者 USB 喚醒。(該暫存器與 USB 喚醒狀態暫存器是同一個位址)。

表 4-88 P_UART_WAKEUP_STATUS(0x88210110)

位	b31~b2	b1	b0
讀/寫	-	R	R
預設值	-	0	0
名稱	-	UART_WAKEUP	USB_WAKEUP

UART_WAKEUP	b1	UART 喚醒狀態位： 0：沒有發生 UART 喚醒 1：發生了 UART 喚醒
USB_WAKEUP	b0	USB 喚醒旗標位： 0：沒有發生 USB 喚醒 1：發生了 USB 喚醒

4.8.6 基本操作

如圖 4-29UART 的結構框圖，UART 包括兩個基本操作：發送和接收。

■ UART 的發送

UART 發送包括查詢和中斷兩種方式。

UART 查詢方式發送流程如圖 4-30。發送前先要打開 UART 時鐘；設置好發送串列傳輸速率；使能 UART 及發送中斷；只要 UART 發送不“忙”，就可以寫發送資料，等待發送，發送完成後“忙”信號被清零。

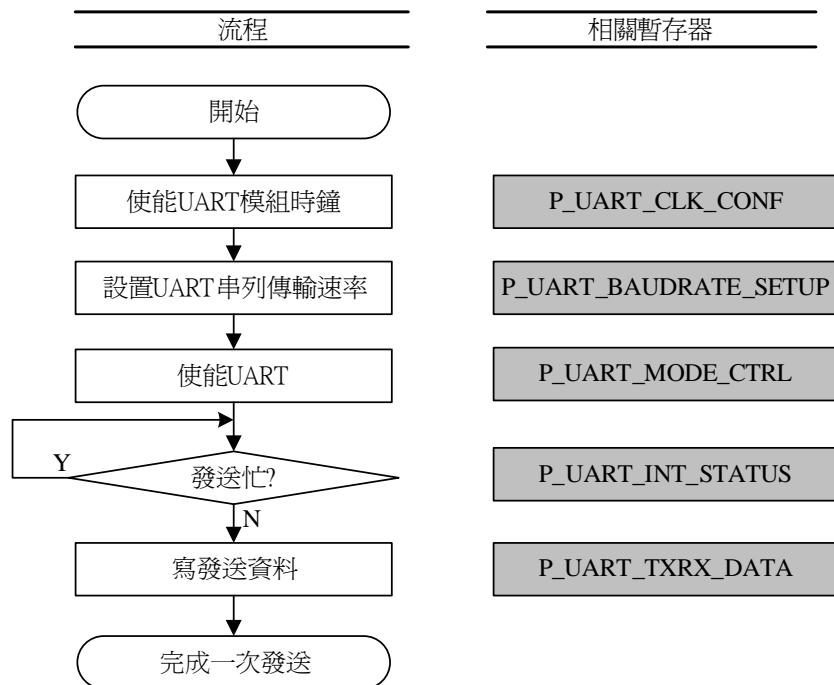


圖 4-30 UART 查詢方式發送流程圖

如果要發送多次資料，發送完成後直接回到判斷發送是否忙迴圈即可。

程式段如下：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    unsigned int uiData = 1; // 將要發送的資料

    *P_UART_CLK_CONF = C_UART_CLK_EN
        | C_UART_RST_DIS; // 使能 UART 模組時鐘
    *P_UART_INTERFACE_SEL = C_UART_PORT_SEL; // 選擇埠作為 UART 使用，預設
    *P_UART_BAUDRATE_SETUP = 0xEA; // 115200
    *P_UART_MODE_CTRL = C_UART_NO_PARITY // 無奇偶檢查
        | C_UART_STOP_1BIT // 1bit 停止位
        | C_UART_DATA_8BIT // 8bit 資料位
        | C_UART_CTRL_EN; // 使能 UART

    while(1)
    {
        if((*P_UART_TXRX_STATUS
            & (C_UART_BUSY_FLAG | C_UART_TXFIFO_FULL)) == 0)
        {
            *P_UART_TXRX_DATA = uiData; // 寫發送資料
            uiData++;
        }
    }
}
```

UART 中斷方式發送包括兩部分：主程序部分和中斷服務程式部分。

主程序流程如圖 4-31，主要進行初始化操作，包括 UART 模組時鐘的使能，串列傳輸速率的設置和使能 UART 及 UART 發送中斷。

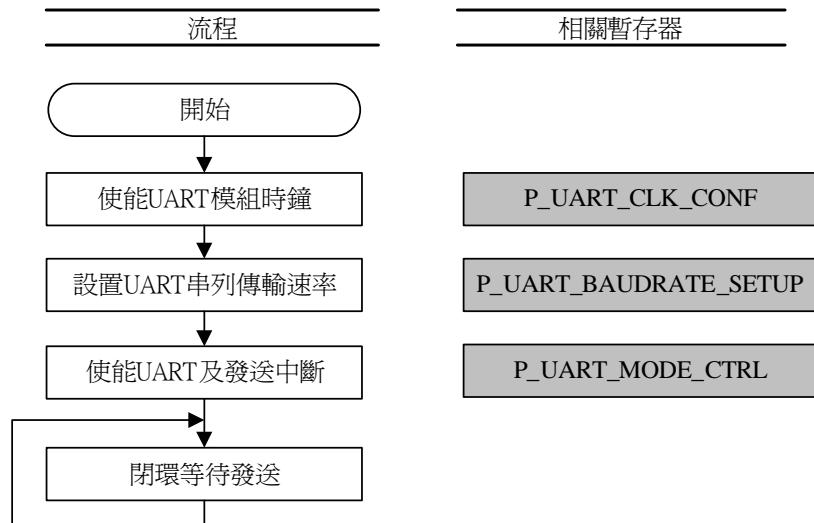


圖 4-31 UART 中斷方式發送主程序流程圖

程式段如下：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    *P_INT_MASK_CTRL1 = ~C_INT_UART_DIS;           // 打開 UART 中斷
    *P_UART_CLK_CONF = C_UART_CLK_EN               // 使能 UART 模組時鐘
    | C_UART_RST_DIS;
    *P_UART_BAUDRATE_SETUP = 0xEA;                 // 選擇串列傳輸速率 115200bps
    *P_UART_MODE_CTRL = C_UART_NO_PARITY          // 無奇偶檢查
    | C_UART_STOP_1BIT;                           // 1bit 停止位
    | C_UART_DATA_8BIT;                          // 8bit 資料位
    | C_UART_TX_INTEN;                          // 發送中斷使能
    | C_UART_CTRL_EN;                           // UART 使能
    while(1);                                     // 等待中斷
}
```

中斷服務程式流程如圖 4-32：UART 的中斷服務程式不需要手動清除中斷旗標，而是在處理完中斷服務程式時會自動清除中斷旗標。

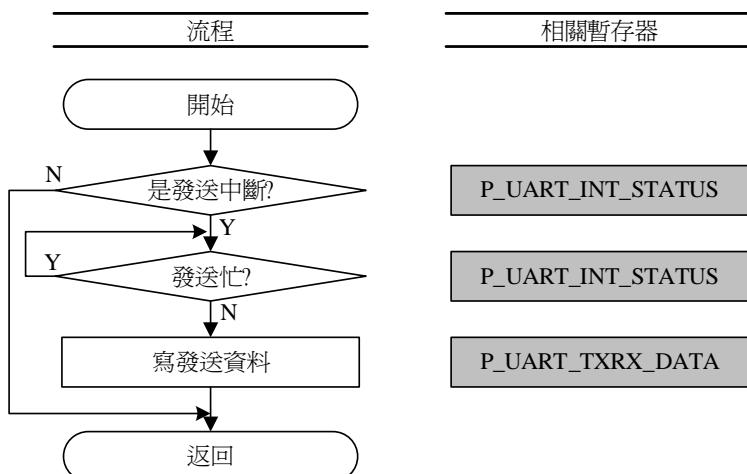


圖 4-32 UART 中斷方式發送中斷服務程式流程圖

程式段如下：

```

//=====
// 語法格式：void IRQ42(void)
// 功能描述：UART 中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ42(void)
{
    if(*P_UART_TXRX_STATUS & C_UART_TX_FLAG)
    {
        if((*P_UART_TXRX_STATUS
            & (C_UART_BUSY_FLAG | C_UART_TXFIFO_FULL)) == 0)
        {
            *P_UART_TXRX_DATA = 0xAA;           // 寫發送資料
        }
    }
}
    
```

■ UART 的接收

UART 接收也包括查詢和中斷兩種方式，分別介紹。

UART 查詢方式發送流程如圖 4-33。接收前先打開 UART 時鐘；設置好接收串列傳輸速率；使能 UART 及接收中斷；如果接收中斷來，透過 P_UART_RX_RX_DATA 單元讀出資料。

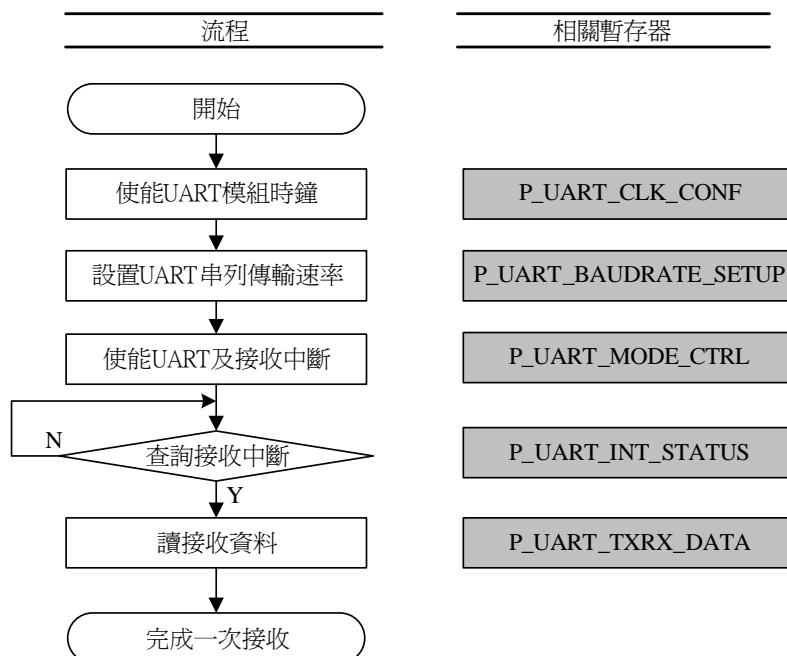


圖 4-33 UART 接收流程圖

如果要接收多次資料，接收完成讀出資料後直接回到判斷接收 FIFO 是否滿迴圈即可。

程式段如下：

```

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
int main(void)
{
    unsigned int uiData[10]; // 定義陣列接收資料
    unsigned int i = 0;

    *P_UART_CLK_CONF = C_UART_CLK_EN
        | C_UART_RST_DIS; // 使能 UART 模組時鐘
    *P_UART_INTERFACE_SEL = C_UART_PORT_SEL; // 選擇埠作為 UART 收發埠
    *P_UART_BAUDRATE_SETUP = 0xEA; // 設置串列傳輸速率為 115200bps
    *P_UART_MODE_CTRL = C_UART_NO_PARITY
        | C_UART_STOP_1BIT // 1bit 停止位
        | C_UART_DATA_8BIT // 8bit 資料位
        | C_UART_CTRL_EN // 使能 UART
        | C_UART_RX_INTEN // 使能 UART 接收中斷
        ;
    while(1)
    {
        if(*P_UART_TXRX_STATUS & C_UART_RX_FLAG) // 有資料？
        {
            uiData[i] = *P_UART_TXRX_DATA; // 讀接收資料
            i++;
            if(i == 9)
                i = 0;
        }
    }
}
    
```

UART 中斷方式接收同樣包括兩部分：主程序部分和中斷服務程式部分。

主程序流程如圖 4-34，主要進行初始化包括 UART 模組時鐘的使能，串列傳輸速率的設置和打開 UART 及 UART 接收中斷。

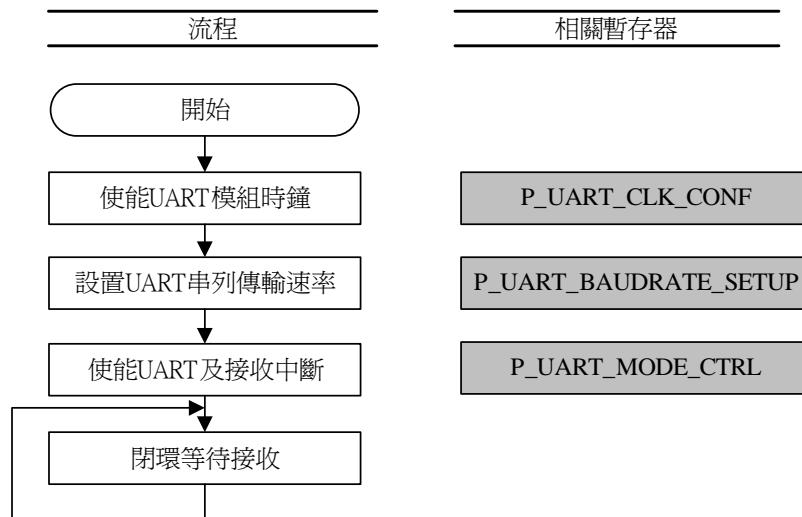


圖 4-34 UART 中斷方式接收主程序流程圖

程式段如下：

```

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    *P_INT_MASK_CTRL1 = ~C_INT_UART_DIS;           // 打開 UART 中斷
    *P_UART_CLK_CONF = C_UART_CLK_EN
        | C_UART_RST_DIS;                         // 使能 UART 模組時鐘
    *P_UART_INTERFACE_SEL = C_UART_PORT_SEL;       // 選擇埠作為 UART 埠
    *P_UART_BAUDRATE_SETUP = 0xEA;                 // 設置串列傳輸速率為 115200bps
    *P_UART_MODE_CTRL = C_UART_NO_PARITY
        | C_UART_STOP_1BIT                        // 1bit 停止位
        | C_UART_DATA_8BIT                         // 8bit 資料位
        | C_UART_CTRL_EN                           // UART 使能
        | C_UART_RX_INTEN                         // UART 接收中斷使能
    ;
    while(1);                                     // 等待接受中斷
}
    
```

中斷服務程式流程如圖 4-32：同樣，UART 的接收中斷服務程式也不需要手動清除中斷旗標，而是在處理完中斷服務程式時會自動清除中斷旗標。

程式段如下：

```

//=====
// 語法格式：void IRQ42(void)
    
```

```
// 功能描述：UART 中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ42(void)
{
    unsigned int uiData;

    if(*P_UART_TXRX_STATUS & C_UART_RX_FLAG) // 是否是接收中斷？
    {
        uiData = *P_UART_TXRX_DATA;           // 讀接收資料
    }
}
```

4.8.7 注意事項

使用 SPCE3200 的 UART 通訊時需要注意下面幾點：

- 通訊雙方的串列傳輸速率必須設置一致；
- 通訊雙方的奇偶檢查方式要設置一致；
- 通訊雙方接收/發送的資料位要設置一致，即如果 SPCE3200 的資料位設置為 6bits，通訊另一方的資料位必須也能設置為 6bits 且必須設置為 6bits；
- 計算串列傳輸速率時需要注意，串列傳輸速率 = $27\text{MHz}/\text{BAUD_RATE} - 1$ ，其中的 BAUD_RATE 為 P_UART_BAUDRATE_SETUP 暫存器設置的資料。

4.9 SPI

4.9.1 概述

SPCE3200 內嵌一個 SPI (Serial Peripheral Interface) 介面控制器，該介面是一個同步、全雙工串列介面，可以便利地與其他週邊設備或者元器件通訊。

4.9.2 特性

SPCE3200 的 SPI 特性如下：

- 支援主/從模式的單字節和連續多位元組資料傳輸
- 支援資料接收溢出錯誤檢測
- 支援發送/接收中斷請求
- 可編程主模式時鐘的相位和極性
- 可選擇資料取樣時間
- 可編程主模式的時鐘頻率，該時鐘頻率可設置為 SPI 系統時鐘 (27MHz) 的 1/2、1/4、1/8、1/16、1/32、1/64、1/128
- 內嵌 8Byte 的接收 FIFO 和發送 FIFO

4.9.3 插腳描述

SPCE3200 有四個 SPI 相關插腳，其中 SPI_CLK、SPI_TX、SPI_RX 為複用插腳，如表 4-89。

表 4-89 SPI 插腳列表

插腳名稱	插腳號	插腳屬性	插腳功能
NF_D4	111	I	SPI_RX 複用插腳，SPI 接收
NF_D5	110	O	SPI_TX 複用插腳，SPI 發送
NF_ALE	127	I/O	SPI_CLK 複用插腳，SPI 時鐘信號
SPI_CSN	128	I/O	SPI 片選信號

SPCE3200 透過 SPI 與其他 SPI 設備或者器件通訊時，只需要把 SPCE3200 的 SPI_RX (NF_D4) 與其他設備的發送腳 (TX) 連接，SPI_TX (NF_D5) 與其他設備的接收腳 (RX) 連接，其他兩個管腳分別對應連接即可。

4.9.4 結構框圖

SPCE3200 的 SPI 模組結構框圖如圖 4-35，SPI 模組包括時鐘產生電路，發送/接收的串/並轉換電路及 FIFO、中斷控制器和 SPI 控制器。時鐘產生電路包括時鐘分頻器、極性/相位產生器，系統時鐘透過分頻器進行分頻後，透過極性、相位產生器產生一個一定相位、一定極性和頻率的時鐘提供給 SPI 模組；並行發送資料的並/串轉換電路和 FIFO 轉換為串列資料後透過控制器控制發送，接收到的串列資料經過接收 FIFO 和串/並轉換電路轉換成並行資料，以便讀取；中斷控制器使能發送/接收中斷，當有資料發送或者接收到資料時，向 CPU 發出中斷請求；SPI 控制器控制選擇為主模式或者從模式，並控制資料的發送和接收。

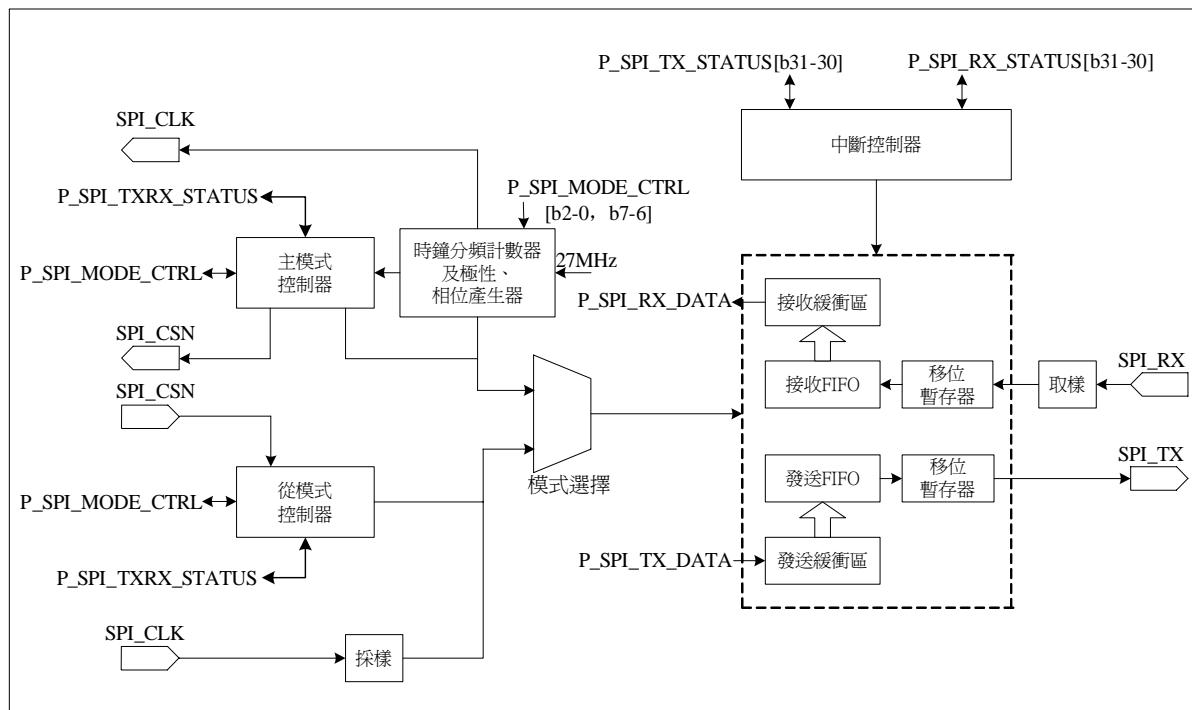


圖 4-35 SPI 結構框圖

4.9.5 SPI 描述

1. SPI 汇流排配置

SPI 是一個全雙工的同步串列通訊介面，一個 SPI 汇流排可以連接多個主機和多個從機，但是在同一時刻只允許有一個主機操作匯流排。在一次資料傳輸中，主機向從機發送一位元組的資料，同時，從機也向主機發送一位元組資料。如圖 4-36為以 SPCE3200 為主機的 SPI 汇流排配置示例，SPI 汇流排的時鐘是由主機 SPCE3200 產生的。

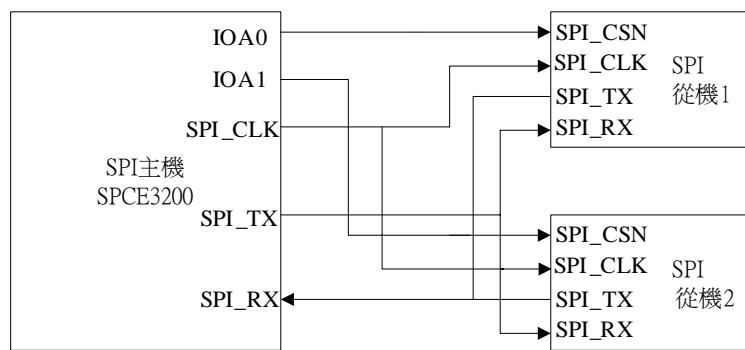


圖 4-36 SPI 汇流排配置

2. SPI 資料傳輸

SPI 的資料傳輸根據時鐘相位 (SPH) 和極性 (SPO) 的設置不同而不同，有下面四種情況：

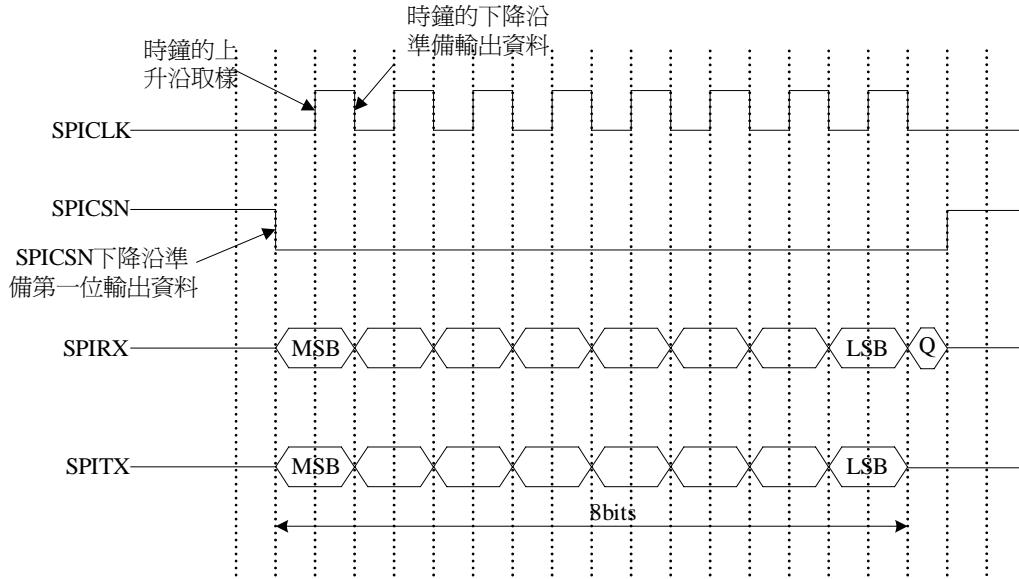
(1) SPO=0 , SPH=0


圖 4-37 SPO=0 , SPH=0

如圖 4-37，當 SPO=0 時，時鐘高電平有效，第一個時鐘沿為上升沿；SPH=0，第一位資料在時鐘的第一個時鐘沿被取樣，所以在時鐘的第一個上升沿到來前必須把第一位傳輸資料輸出到資料線上，其他位元資料在時鐘的下降沿被輸出到資料線上，在時鐘的上升沿被取樣。

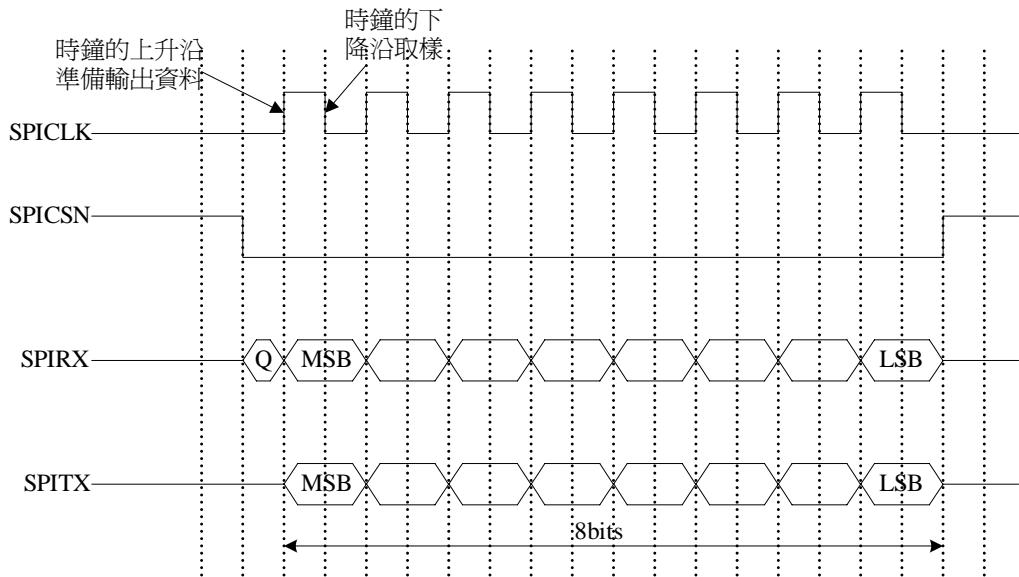
(2) SPO=0 , SPH=1


圖 4-38 SPO=0 , SPH=1

如圖 4-38，當 SPO=0 時，時鐘高電平有效，第一個時鐘沿為上升沿；SPH=1，第一位資料在時鐘的第二個時鐘沿被取樣，所以在資料傳輸過程中，所有位元資料在時鐘的上升沿輸出到資料線上，在時鐘的下降沿被取樣。

(3) SPO=1 , SPH=0

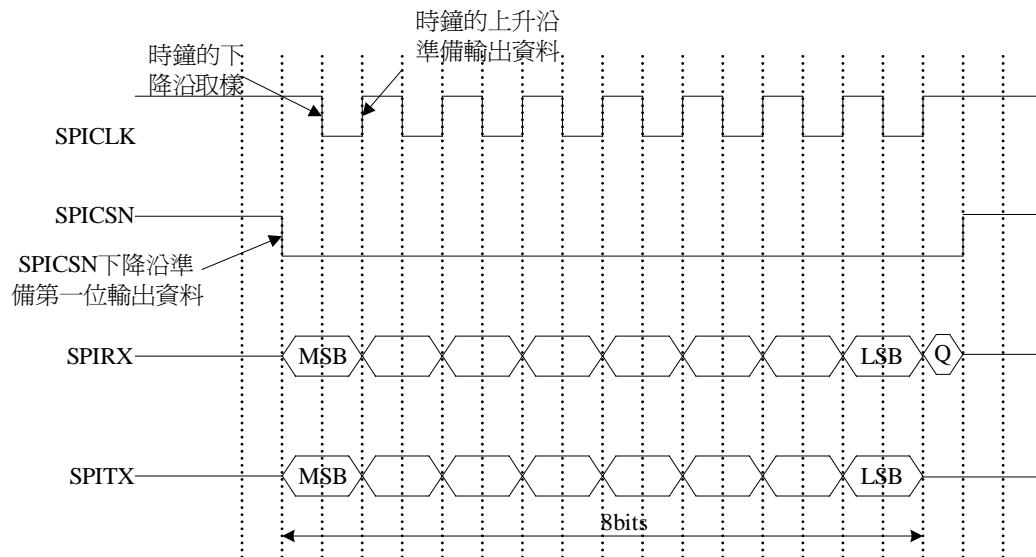


圖 4-39 SPO=1, SPH=0

如圖 4-39，當 $SPO=1$ 時，時鐘低電平有效，第一個時鐘沿為下降沿； $SPH=0$ ，第一位資料在時鐘的第一個時鐘沿被取樣，所以在資料傳輸過程中，第一位資料在第一個下降沿到來之前輸出到資料線上，其他位元資料在時鐘的上升沿輸出到資料線上，在時鐘的下降沿被取樣。

(4) **SPO=1 , SPH=1**

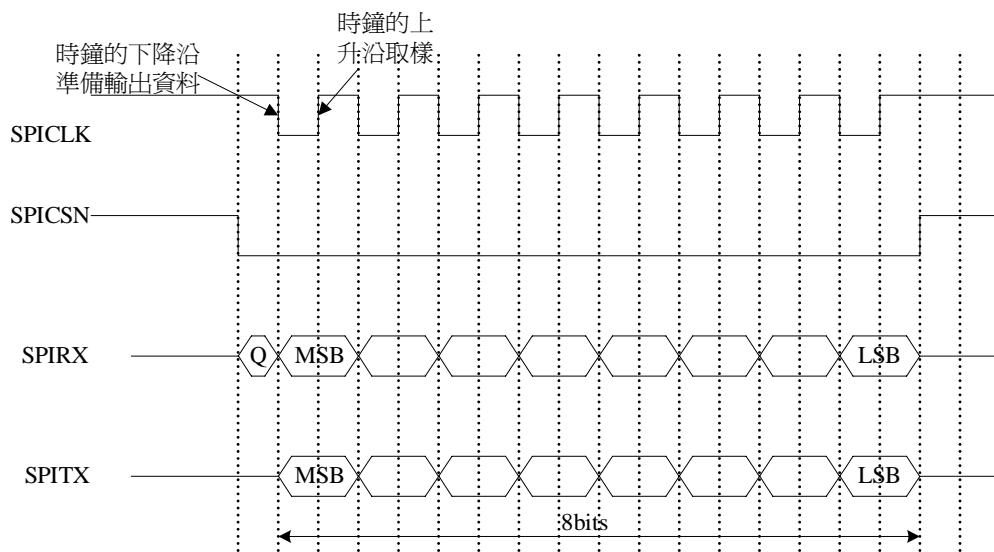


圖 4-40 SPO=1, SPH=1

如圖 4-40，當 $SPO=1$ 時，時鐘低電平有效，第一個時鐘沿為下降沿； $SPH=1$ ，第一位資料在時鐘的第二個時鐘沿被取樣，所有位元資料在時鐘的下降沿輸出到資料線上，在時鐘的上升沿被取樣。

3. SPI 的應用

SPI 可應用於：

- 串列記憶體，如 93C46 記憶體；
- 串列週邊設備，如 ADC、DAC、LCD 控制器、CAN 控制器和感測器等；
- 外部協同處理器等。

4.9.6 暫存器描述

SPI 模組共有 8 個暫存器，如表 4-90。

表 4-90 SPI 相關暫存器列表

暫存器名稱	助記符	位址
SPI 時鐘配置暫存器	P_SPI_CLK_CONF	0x88210098
SPI 介面選擇暫存器	P_SPI_INTERFACE_SEL	0x882000A4
SPI 控制暫存器	P_SPI_MODE_CTRL	0x88110000
SPI 發送狀態暫存器	P_SPI_TX_STATUS	0x88110004
SPI 發送資料暫存器	P_SPI_TX_DATA	0x88110008
SPI 接收狀態暫存器	P_SPI_RX_STATUS	0x8811000C
SPI 接收資料暫存器	P_SPI_RX_DATA	0x88110010
SPI 收發狀態暫存器	P_SPI_TXRX_STATUS	0x88110014

■ SPI 時鐘配置暫存器：P_SPI_CLK_CONF(0x88210098)

透過 SPI 時鐘配置暫存器可以停止/打開 SPI 時鐘或者重設/不重設 SPI 模組，在使能 SPI 前需先打開 SPI 模組時鐘。

表 4-91 P_SPI_CLK_CONF(0x88210098)

位	b31~b2	b1	b0
讀/寫	-	W	W
預設值	-	1	0
名稱	-	SPI_RST	SPI_STOP

SPI_RST	b1	SPI 模組時鐘重設位： 0 : SPI 模組時鐘重設 1 : SPI 模組時鐘不重設
SPI_STOP	b0	SPI 模組時鐘使能位元。 0 : SPI 模組時鐘停止 1 : SPI 模組時鐘使能

■ SPI 介面選擇暫存器：P_SPI_INTERFACE_SEL(0x882000A4)

由於 SPI 的 SPI_Tx、SPI_Rx 和 SPI_CLK 與 Nand Flash 的介面複用，所以在使用這些埠前先要透過 SPI 介面選擇暫存器設置這些複用介面為 SPI 功能。

表 4-92 P_SPI_INTERFACE_SEL(0x882000A4)

位	b31~b9	b8	b7~b0
讀/寫	-	R/W	-
預設值	-	0	-
名稱	-	SPI_EN	-

SPI_EN	b8	SPI 介面使能位： 0 : 複用介面不為 SPI 介面 1 : 複用介面作為 SPI 介面
--------	----	--

■ SPI 控制暫存器：P_SPI_MODE_CTRL(0x88110000)

透過 P_SPI_MODE_CTRL 可以設置 SPI 的時鐘頻率、時鐘的極性和相位、SPI 的主/從模式，進行 SPI 的使能等。

表 4-93 P_SPI_MODE_CTRL(0x88110000)

位	b31	b27	b26	b25	b7	b6	b2~b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0	0
名稱	SPI_EN	SPI_LB	SOFT_REST	SPI_MODE	SPI_CK_PH	SPI_CK_PO	SPI_CLOCK

SPI_EN	b31	SPI 使能位： 0：禁止 SPI 1：使能 SPI
SPI_LB	b27	SPI 迴圈送回模式選擇位： 0：正常模式 1：SPIRX=SPITX
SOFT_REST	b26	SPI 軟重設使能位： 0：不進行軟重設 1：軟重設
SPI_MODE	b25	SPI 主/從模式選擇位： 0：主模式 1：從模式
SPI_CK_PH	b7	SPI 時鐘相位選擇位 (SPH)： 0：資料在第一個時鐘沿被取樣 1：資料在第二個時鐘沿被取樣 詳細可參考圖 4-37~圖 4-40。
SPI_CK_PO	b6	SPI 時鐘極性選擇位 (SPO)： 0：時鐘高電平有效 1：時鐘低電平有效 詳細可參考圖 4-37~圖 4-40。
SPI_CLOCK	b[2:0]	SPI 主模式的時鐘選擇位： 000 : PCLK/2 001 : PCLK/4 010 : PCLK/8 011 : PCLK/16 100 : PCLK/32 101 : PCLK/64 110 : PCLK/128 注：PCLK 是 APB 汇流排的主頻，為 27MHz。

■ SPI 收發狀態暫存器：**P_SPI_TXRX_STATUS(0x88110014)**

透過 P_SPI_TXRX_STATUS 可以設置 SPI 的清除中斷旗標方式、資料溢出模式，並可以得到 SPI 汇流排是否忙、發送/接收 FIFO 的狀態資訊等。

表 4-94 P_SPI_TXRX_STATUS(0x88110014)

位	b31	b30	b4	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0	0
名稱	SPI_OW_MODE	SPI_SMART	SPI_BUSY	RX_FULL	RX_EMPTY	TX_FULL	TX_EMPTY

SPI_OW_MODE	b31	SPI 溢出模式設置位： 0：當發生溢出時跳過資料 1：當發生溢出時重寫資料
SPI_SMART	b30	SPI 清中斷旗標方式選擇位： 0：手動清中斷旗標 1：自動清中斷旗標
SPI_BUSY	b4	SPI 汇流排狀態位： 0：空閒 1：忙
RX_FULL	b3	SPI 接收 FIFO 滿旗標位： 0：非滿 1：滿
RX_EMPTY	b2	SPI 接收 FIFO 空旗標位： 0：空 1：非空
TX_FULL	b1	SPI 發送 FIFO 滿旗標位： 0：滿 1：非滿

TX_EMPTY	b0	SPI 發送 FIFO 空旗標位： 0：非空 1：空
----------	----	----------------------------------

■ SPI 發送狀態暫存器：P_SPI_TX_STATUS(0x88110004)

透過 SPI 發送狀態暫存器可以設置 SPI 的發送中斷等。

表 4-95 P_SPI_TX_STATUS(0x88110004)

位	b31	b30	b29~b28	b27	b26~b0
讀/寫	R/W	W	-	R	-
預設值	0	0	-	0	-
名稱	TX_FLAG	TX_EN	-	TX_EMPTY_FLAG	-

TX_FLAG	b31	SPI 發送中斷旗標位： 讀 0：沒有發生 SPI 發送中斷 讀 1：發生 SPI 發送中斷 寫 0：無意義 寫 1：清除中斷旗標
---------	-----	---

TX_EN	b30	SPI 發送中斷使能位： 0：遮罩 SPI 發送中斷 1：使能 SPI 發送中斷
-------	-----	--

TX_EMPTY_FLAG	b27	SPI 發送 FIFO 空旗標位： 0：非空 1：空
---------------	-----	----------------------------------

■ SPI 接收狀態暫存器：P_SPI_RX_STATUS(0x8811000C)

SPI 接收狀態暫存器可以設置 SPI 的接收中斷等。

表 4-96 P_SPI_RX_STATUS(0x8811000C)

位	b31	b30	b29~b28	b27	b26	b25~b0
讀/寫	R/W	W	-	R	R/W	-
預設值	0	0	-	0	0	-
名稱	RX_FLAG	RX_EN	-	RX_FULL_FLAG	RX_OR_ERR	-

RX_FLAG	b31	SPI 接收中斷旗標位： 讀 0：沒有發生 SPI 接收中斷 讀 1：發生了 SPI 接收中斷 寫 0：無意義 寫 1：清除該中斷旗標
RX_EN	b30	SPI 接收中斷使能位： 0：遮罩 SPI 接收中斷 1：使能 SPI 接收中斷
RX_FULL_FLAG	b27	SPI 接收 FIFO 滿旗標位： 0：非滿 1：滿
RX_OR_ERR	b26	SPI 接收溢出錯誤旗標位： 讀 0：沒有發生 SPI 接收溢出 讀 1：發生了 SPI 接收溢出 寫 0：無意義 寫 1：清除該旗標

■ SPI 發送資料暫存器：P_SPI_TX_DATA(0x88110008)

SPI 發送資料暫存器用來保存 SPI 準備發送的資料。

表 4-97 P_SPI_TX_DATA(0x88110008)

位	b31~b8	b7~b0
讀/寫	-	W
預設值	-	0x00

位	b31~b8	b7~b0
名稱	-	SPI_TX_DATA

SPI_TX_DATA	b7~b0	SPI 準備發送的資料
-------------	-------	-------------

■ SPI 接收資料暫存器：**P_SPI_RX_DATA(0x88110010)**

SPI 接收資料暫存器用來保存 SPI 接收到的資料。

表 4-98 P_SPI_RX_DATA(0x88110010)

位	b31~b8	b7~b0
讀/寫	-	R
預設值	-	0x00
名稱	-	SPI_RX_DATA

SPI_RX_DATA	b7~b0	SPI 接收到的資料
-------------	-------	------------

4.9.7 基本操作

SPCE3200 的 SPI 是一個同步、全雙工串列介面，可以透過 P_SPI_MODE_CTRL 暫存器的 bit25 位設置為主機模式或從機模式，作為主機時輸出的時鐘速率、極性、相位可以透過 P_SPI_MODE_CTRL 的 bit0~bit2、bit6 和 bit7 編程設置。

不管 SPCE3200 工作在主機模式還是從機模式，分別有兩種操作方式：查詢方式和中斷方式。

■ 查詢方式

以主機為例，SPCE3200 作為主機查詢方式傳輸資料的操作流程如圖 4-41，操作可分為 SPI 初始化、發送和接收三個步驟。其中初始化包括使能 SPI 模組時鐘；由於 SPI 的介面與 Nand Flash 及 SD 介面複用，所以需要設置為 SPI 介面；使能發送及接收中斷；使能 SPI 準備發送或者接收資料；如圖 4-41。如果 SPI 汇流排不忙，且發送 FIFO 非滿，填充發送資料準備發送；等待發送完成。如果接收 FIFO 非空，讀取接收資料。

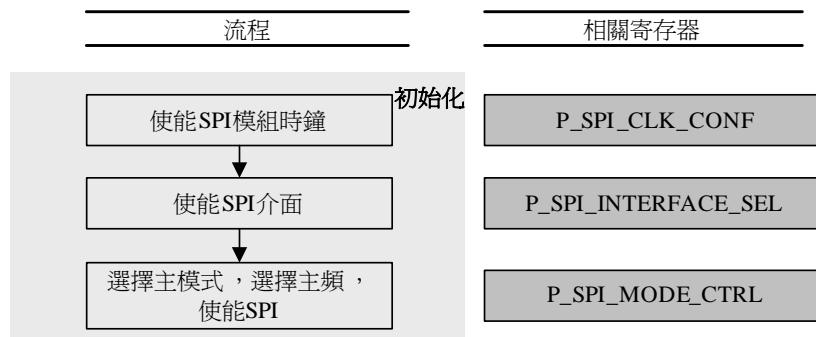


圖 4-41 SPI 主機查詢方式初始化

自發自收，即將 SPI_RX 與 SPI_TX 管腳相連，使用查詢接收中斷方式，參考程式：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    unsigned int uiT_data[10] = {1,2,3,4,5,6,7,8,9,10};
    unsigned int uiR_data[10] = {0};
    unsigned int i = 0;

    *P_SPI_CLK_CONF = C_SPI_CLK_EN
        | C_SPI_RST_DIS;                                // 使能 SPI 模組時鐘
    *P_SPI_INTERFACE_SEL = C_SPI_PORT_SEL;           // SPI 埠選擇
    *P_SPI_MODE_CTRL = C_SPI_CLK_27MDIV128
        | C_SPI_MASTER_MODE;                          // 選擇系統時鐘
        | C_SPI_NORMAL_MODE;                         // 主模式
        | C_SPI_CTRL_EN;                            // 正常傳輸模式，相對 test 模式
        | C_SPI_RX_INTEN;                           // SPI 使能
    *P_SPI_RX_STATUS = C_SPI_RX_INTEN
        | C_SPI_RX_FLAG;                            // 使能 SPI 接受中斷

    while(1)
    {
        *P_SPI_TX_DATA = uiT_data[i];                // 發送
        while(!(P_SPI_RX_STATUS & C_SPI_RX_FLAG)); // 是否有接收中斷
        *P_SPI_RX_STATUS |= C_SPI_RX_FLAG;           // 清除接收中斷
        uiR_data[i] = *P_SPI_RX_DATA;                 // 接收資料
        i++;
        if(i == 10)
            i = 0;
    }
}
```

使用查詢 FIFO 方式，參考程式：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    unsigned int uiT_data[10] = {1,2,3,4,5,6,7,8,9,10};
    unsigned int uiR_data[10] = {0};
```

```
unsigned int i = 0;
unsigned int j = 0;

*p_SPI_CLK_CONF = C_SPI_CLK_EN
    | C_SPI_RST_DIS;           // 使能 SPI 模組時鐘
*p_SPI_INTERFACE_SEL = C_SPI_PORT_SEL; // SPI 埠選擇
*p_SPI_MODE_CTRL = C_SPI_CLK_27MDIV128
    | C_SPI_MASTER_MODE      // 主模式
    | C_SPI_NORMAL_MODE      // 正常傳輸模式，相對 test 模式
    | C_SPI_CTRL_EN;          // SPI 使能

while(1)
{
    *p_SPI_TXRX_STATUS = C_SPI_MANNUAL_CLR;
    while(*p_SPI_TXRX_STATUS & C_SPI_TXFIFO_NOTFULL)
    {
        *p_SPI_TX_DATA = uiT_data[i]; // 發送
        i++;
        if(i == 10)
            i = 0;
    }
    while(*p_SPI_TXRX_STATUS & C_SPI_RXFIFO_NOTEEMPTY)
    {
        uiR_data[j] = *p_SPI_RX_DATA;
        j++;
        if(j == 10)
            j = 0;
    }
}
```

■ 中斷方式

中斷方式和查詢方式類似，在主程序裏主要進行 SPI 的初始化操作，在中斷服務程式裏填寫發送資料或者讀取接收資料。

自發自收，即將 SPI_RX 與 SPI_TX 管腳相連，使用中斷方式，參考程式：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"

int main(void)
{
    unsigned int uiT_data[10] = {1,2,3,4,5,6,7,8,9,10};
    unsigned int i = 0;

    *p_INT_MASK_CTRL1 = ~C_INT_SPI_DIS; // 使能 SPI 中斷
    *p_SPI_CLK_CONF = C_SPI_CLK_EN
        | C_SPI_RST_DIS;           // 使能 SPI 模組時鐘
    *p_SPI_INTERFACE_SEL = C_SPI_PORT_SEL; // SPI 埠選擇
    *p_SPI_MODE_CTRL = C_SPI_CLK_27MDIV128
        | C_SPI_MASTER_MODE      // 主模式
        | C_SPI_NORMAL_MODE      // 正常傳輸模式，相對 test 模式
        | C_SPI_CTRL_EN;          // SPI 使能
    *p_SPI_RX_STATUS = C_SPI_RX_INTEN // 使能 SPI 接受中斷
```

```
| C_SPI_RX_FLAG;
while(1)
{
    while(*P_SPI_TXRX_STATUS & C_SPI_BUSY_FLAG);
    *P_SPI_TX_DATA = uiT_data[i];           // 發送
    i++;
    if(i == 10)
        i = 0;
}
//=====
// 語法格式：void IRQ43(void)
// 功能描述：SPI 中斷服務函數
// 入口參數：無
// 出口參數：無
//=====
void IRQ43(void)
{
    unsigned int uiR_data;

    if(*P_SPI_RX_STATUS & C_SPI_RX_FLAG)      // 是否有接收中斷
    {
        uiR_data = *P_SPI_RX_DATA;            // 接收資料
    }
    *P_SPI_RX_STATUS |= C_SPI_RX_FLAG;         // 清除接收中斷
}
```

4.9.8 注意事項

使用 SPCE3200 的 SPI 介面時需要注意：SPCE3200 作為主機/從機時 SPI_CS_N 和 SPI_CLK 的輸入/輸出方向不同，作為主機時這兩個管腳都為輸出，作為從機時這兩個管腳都為輸入。

4.10 I²C

4.10.1 概述

SPCE3200 有一個標準的硬體 I²C (Inter-Integrated Circuit)，另外，為了描述方便，本書中所有的 I²C 都用 I²C 表示) 介面，可以方便的與帶有 I²C 汇流排的晶片通訊，可調整匯流排的通訊時鐘頻率。

4.10.2 特性

I²C 的特性如下：

- 可作為標準的 I²C 介面；
- 可配置為主機；
- 可編程時鐘頻率來控制通訊速率；
- 主機、從機之間可實現雙向資料傳輸；
- 握手機制使得主從機之間的通訊更加靈活；
- 支援單字節 (8bit)、半字 (16bit)、多位元組 (8bit×n) 的資料傳輸；
- 傳輸時序中資料位址的傳輸，使得 SPCE3200 和外部串列記憶體之間的讀寫更加便利。

4.10.3 插腳描述

SPCE3200 有兩個 I2C 插腳，如表 4-99 所示：

表 4-99 I2C 插腳列表

插腳名稱	插腳號	插腳屬性	插腳功能
I2C_CLK	35	O	I2C 時鐘插腳
I2C_DATA	36	I/O	I2C 資料插腳

使用 SPCE3200 作為 I2C 匯流排的主機時，在 SPCE3200 的 I2C_CLK 和 I2C_DATA 接入匯流排時需要各自接一個 1~10KΩ 的上拉電阻。

SPCE3200 的 I2C 介面結構框圖如圖 4-42：主要包括時鐘發生器、控制器和移位暫存器三大部分。

時鐘發生器即分頻計數器，I2C 的系統時鐘 (27MHz) 經 4 分頻，透過分頻計數器後提供 I2C 的資料傳輸時鐘頻率。

控制器包括開啓/傳輸模式等控制模組、資料傳輸位元數仲裁器和中斷控制器，開啓/傳輸模式等控制模組（圖中的 I2C 控制器）負責開始或者停止資料傳輸，選擇傳輸模式；資料傳輸位元數仲裁器控制進入/移出移位暫存器的資料位元數；中斷控制器負責使能 I2C 中斷，並在主機向匯流排發送起始信號和位址時發出中斷請求，發出後中斷旗標位置 1。

移位暫存器主要負責按照時序控制邏輯單元的控制邏輯把位址、資料及應答信號按位移出到介面發送或者給接收資料單元保存接收資料。

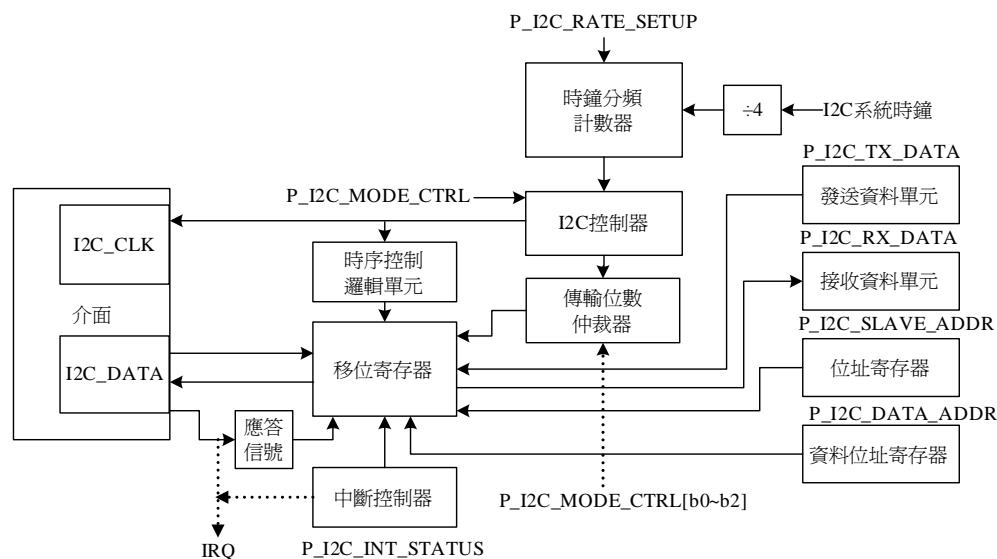


圖 4-42 I2C 結構框圖

4.10.4 I2C 描述

1. 汇流排配置

I2C 是一個半雙工的串列通訊匯流排，是兩線制（資料線和時鐘線）匯流排；一個 I2C 汇流排上可以連接多個主機和多個從機，由握手信號決定主從機之間的通訊；主機產生所有串列時鐘脈衝及起始和結束條件。

SPCE3200 只可以作為 I2C 汇流排的主機。對於主機來說，按照資料傳輸的方向可以分為以下兩類匯流排配置模式：

- 主發送器模式：即主機向從機發送資料。
- 主接收器模式：即從機向主機發送資料。

圖 4-43 為 I2C 汇流排的配置電路。其中 R_p 為上拉電阻，阻值在 $1\sim10K\Omega$ 之間即可。

圖 4-44 為 SPCE3200 作為 I2C 汇流排主機的典型應用電路。

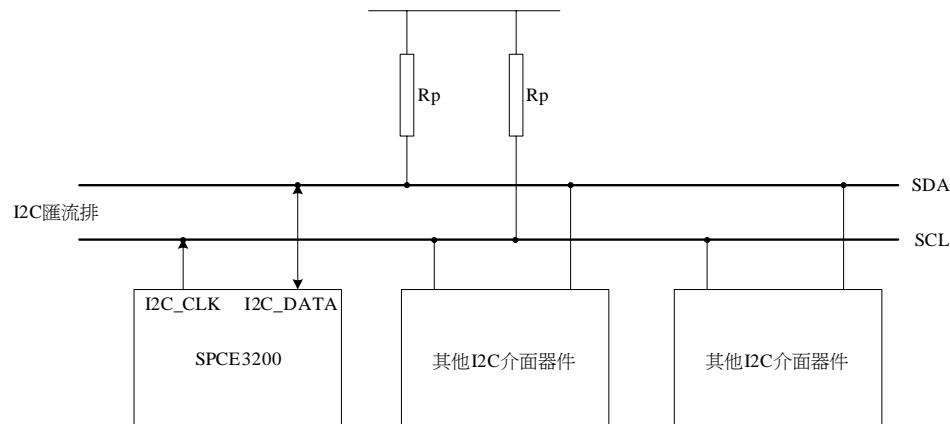


圖 4-43 I2C 汇流排配置

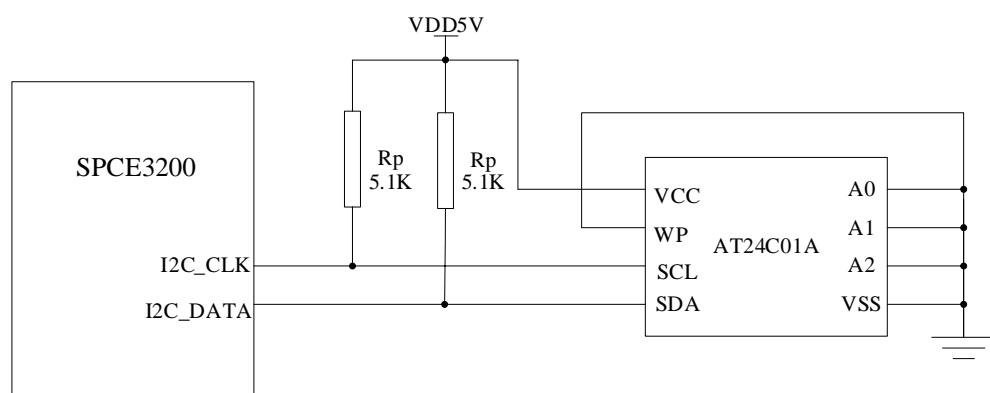


圖 4-44 SPCE3200 作為 I2C 汇流排主機典型電路

2. I2C 資料傳輸

SPCE3200 的 I2C 資料傳輸包括主發送和主接收：

■ 主發送

SPCE3200 的主發送是指 SPCE3200 作為主機向 I2C 汇流排上的從機發送資料，此時 SPCE3200 稱作主發送器，因此主發送也被稱作主發送器模式。由於 SPCE3200 的 I2C 傳輸支援單字節、半字和多位元組傳輸，所以主發送器模式又可分別：單字節發送、半字發送和多位元組發送模式。

(1) 單字節主發送模式

單字節主發送模式的資料格式如圖 4-45，起始 (S) 和結束 (P) 條件均由主機 SPCE3200 產生，分別用於指示串列資料傳輸開始和結束。在此模式下資料傳輸模式為發送，所以暫存器 P_I2C_MODE_CTRL (詳見第 4.10.5 節) 的 bit6 應該被設置為 0，表示執行發送操作。完成發送後，I2C 中斷旗標置 1。

注意圖中從機 ID 位址 (8 位) 為 7 位從機 ID 位址 +1 位 0。

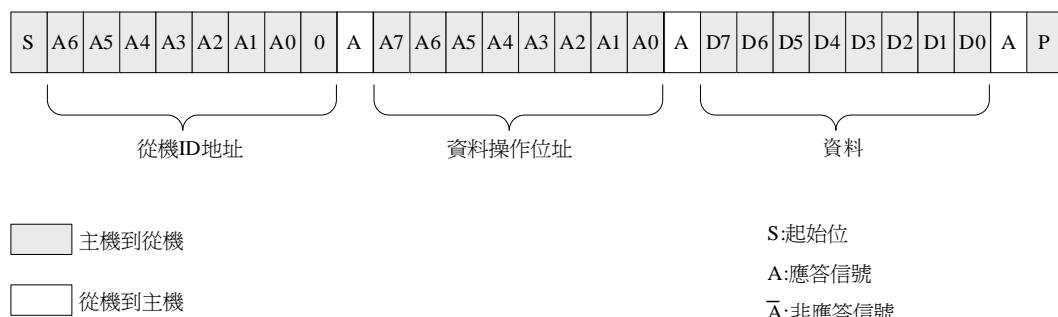


圖 4-45 單字節主發送模式的資料格式

單字節主發送模式的時序圖如圖 4-46：在 SPCE3200 向 I2C 汇流排上某一從機發送資料時，SPCE3200 (主機) 需先向 I2C 汇流排發送一個起始位元 (起始條件)，接著發送對方 (從機) 的 ID 位址和對從機操作的位址 (資料操作位址)，如果向一個 I2C 介面記憶體 0x00 位址寫一個資料，該位址即為 0x00；如果從機收到這些資料且確認準備好通訊，向主機發送一個應答信號；主機收到應答信號後引發中斷開始向從機發送一個位元組資料；從機收到一位元組資料後向主機發送應答信號；I2C 中斷旗標置 1；主機發送停止位元，一個位元組的資料發送完成。

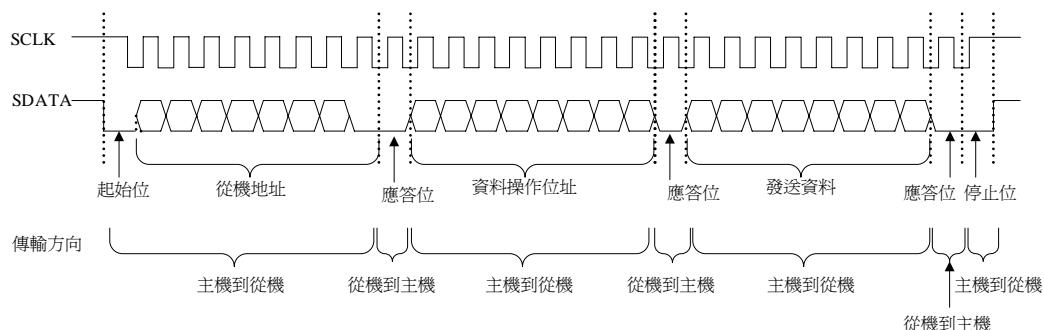


圖 4-46 單字節主發送模式的時序圖

(2) 半字主發送模式

半字主發送模式的資料格式如圖 4-47，和單字節發送模式類似，所不同的是半字主發送模式的資料位為 16 位（圖中 D15~D0）。時序圖如圖 4-48。

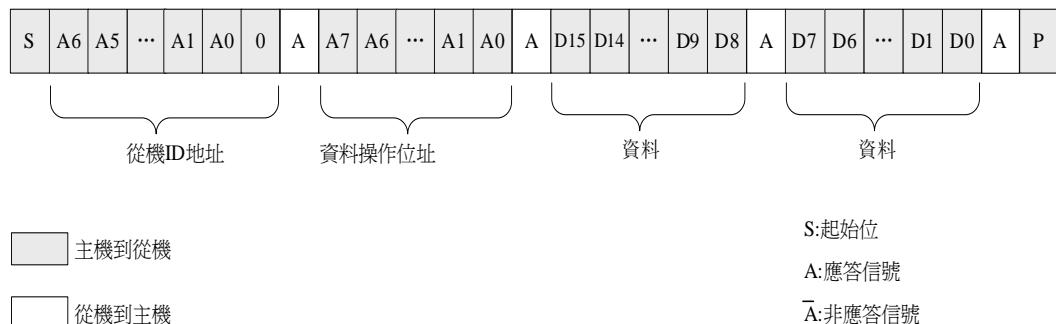


圖 4-47 半字主發送模式的資料格式

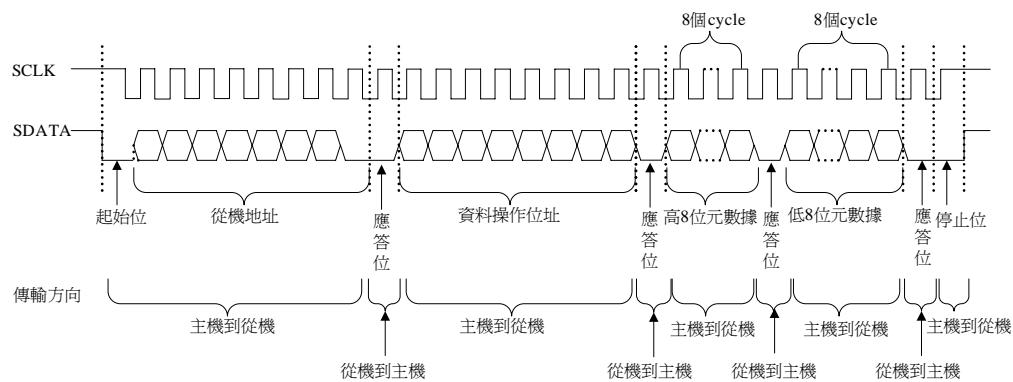


圖 4-48 半字主發送模式的時序圖

(3) 多位元組發送模式

多位元組 ($8 \times N$) 發送模式的資料格式如圖 4-49，多位元組發送和單字節或者半字發送的過程不同，單字節、半字是在完成所有資料（所有一次傳輸中包含的 ID 位址、資料操作位址、資料和應答）的傳輸後觸發中斷，而多位元組傳輸時每傳輸完 8 位元資料就會觸發一次中斷，例如，發送完從機 ID 位址，觸發一次中斷，發送完資料操作位址，又會觸發一次中斷，依次類推。所以，多位元組傳輸模式的最小傳輸單位是 8 位元，也稱為一個 Phase。

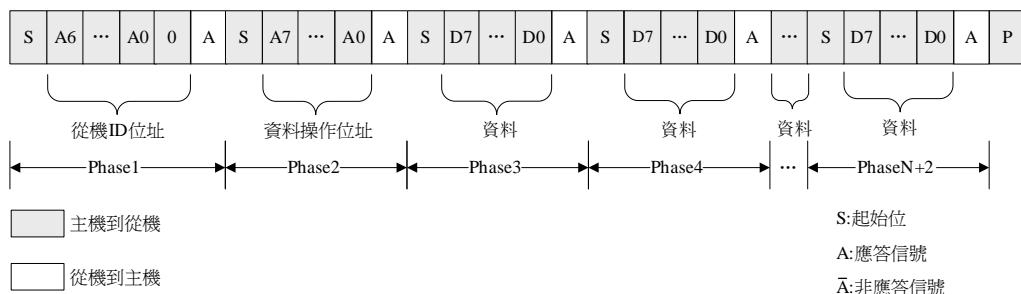


圖 4-49 多位元組主發送模式的資料格式

一個 Phase 的資料發送時序圖如圖 4-50。圖 4-49 中每一個 Phase 都可以看作一次獨立的資料發送，N 個位元組的資料的發送可以看作發送 1 個 Phase 的從機位址、發送 1 個 Phase 的資料操作位址和 N 個 Phase 的資料的過程。

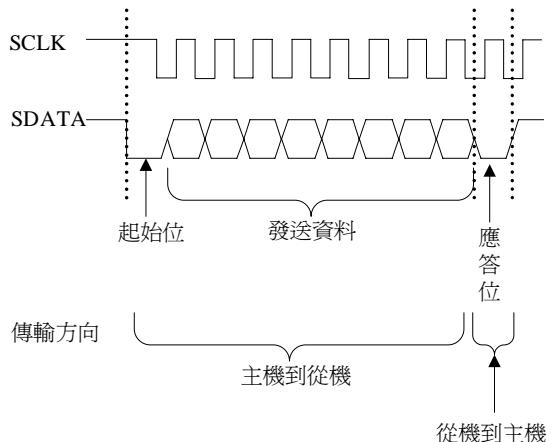


圖 4-50 一個 Phase 的資料發送時序圖

■ 主接收

SPCE3200 的主接收是指 SPCE3200 作為主機接收來自 I2C 汇流排上從機的資料，此時 SPCE3200 稱作主接收器，因此主接收也被稱作主接收器模式。同樣由於 SPCE3200 的 I2C 傳輸支援單字節、半字和多位元組傳輸，所以主接收器模式又可分別：單字節接收、半字接收和多位元組接收模式。

(1) 單字節主接收模式

單字節主接收模式的資料格式如圖 4-51，起始 (S) 和結束 (P) 條件均由主機 SPCE3200 產生，分別用於指示串列資料傳輸開始和結束。在此模式下資料傳輸模式為接收，所以暫存器 P_I2C_MODE_CTRL (詳見第4.10.5 節) 的 bit6 應該被設置為 1，表示執行接收操作。

注意圖中出現了兩個從機 ID 位址(8 位)，第一個從機 ID 位址為 7 位從機 ID 位址+1 位“0”；第二個從機 ID 位址為 7 位從機 ID 位址+1 位“1”。兩個從機 ID 位址的功能不同：第一個從機 ID 位址為一個握手信號，說明和位址 A7~A0 的從機進行通訊；第二個從機 ID 位址之後帶一位“1”，表示要進行的操作為讀操作。

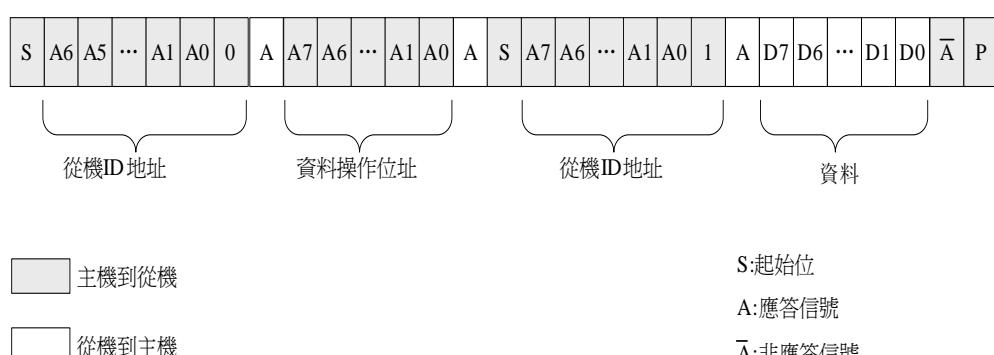


圖 4-51 單字節主接收模式的資料格式

單字節主接收模式的時序圖如圖 4-52：SPCE3200（主機）需先向 I2C 決定排發送一個起始位元（起始條件），接著發送對方（從機）的 ID 位址和對從機操作的位址（資料操作位址，如果從一個 I2C 介面記憶體 0x00 位址讀一個資料，該位址即為 0x00），並發送讀寫條件（低電平表示寫，高電平表示讀）；如果從機收到這些資訊且確認準備好狀態，向主機發送一個應答信號；主機收到應答信號後發送從機位址，注意從機位址的最低位為 1；從機接收到該資訊後，給主機發送應答信號，並發送一位元組的資料，發送完成主機接收到資料後，主機返回一非應答信號，表示一位元組資料接收完成；I2C 中斷旗標置 1；主機發送停止位元，一個位元組的資料接收結束。

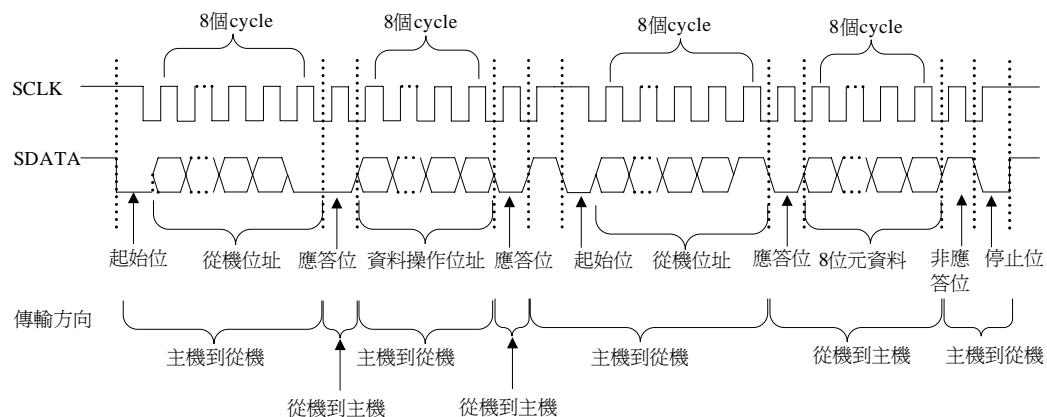


圖 4-52 單字節主接收模式的時序圖

(2) 半字主接收模式

半字主接收模式的資料格式如圖 4-53，和單字節接收模式類似，所不同的是半字主接收模式的資料位為 16 位（圖中 D15~D0）。時序圖如圖 4-54。

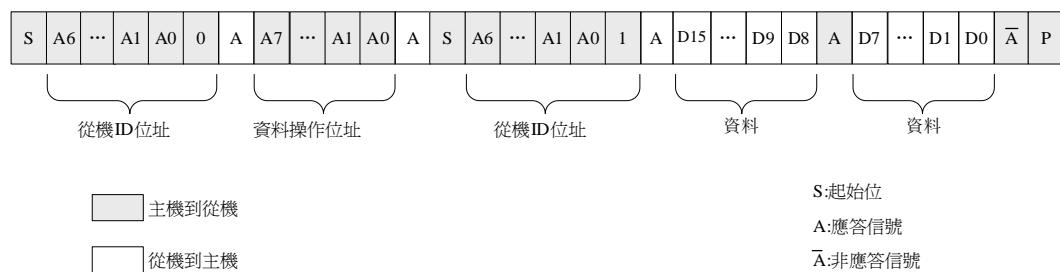


圖 4-53 半字主接收模式的資料格式

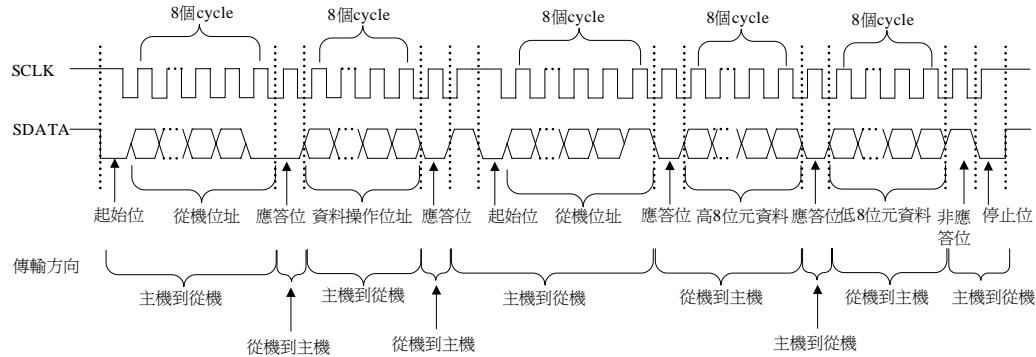


圖 4-54 半字主接收模式的時序圖

(3) 多位元組主接收模式

多位元組 ($8 \times N$) 主接收模式的資料格式如圖 4-55，多位元組接收和單字節或者半字接收的過程不同，後兩者是在完成所有資料（所有一次傳輸中包含的 ID 位址、資料操作位址、資料和應答）的傳輸後觸發中斷，而多位元組傳輸時每傳輸完 8 位元資料就會觸發一次中斷，例如，發送完從機 ID 位址，觸發一次中斷，發送完資料操作位址，又會觸發一次中斷，依次類推。所以，多位元組傳輸模式的最小傳輸單位是 8 位元，同樣稱為一個 Phase。

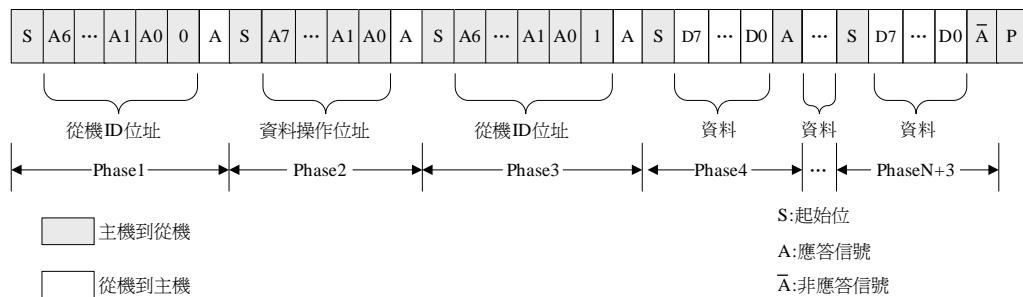


圖 4-55 多位元組主接收模式的資料格式

一個 Phase 的資料接收時序圖如圖 4-56，一個 Phase 的資料發送時序圖如圖 4-50。圖 4-55 中每一個 Phase 都可以看作一次獨立的資料傳輸， N 個位元組的資料的接收可以看作發送 1 個 Phase 的從機位址、發送 1 個 Phase 的資料操作位址、再發送 1 個 Phase 的從機位址（最低位為 1）和接收 N 個 Phase 的資料的過程。

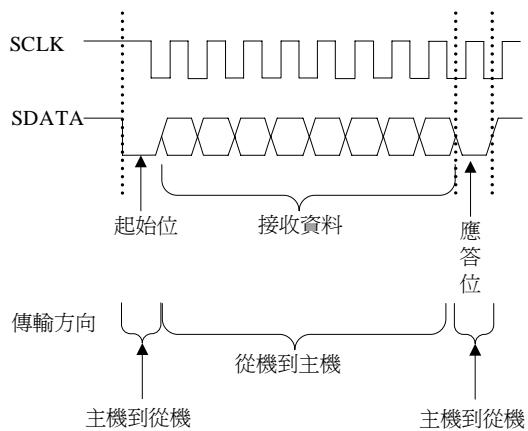


圖 4-56 一個 Phase 的資料接收時序圖

3. I2C 的應用

I2C 汇流排具有連接線少（只有兩線），連接簡單和通訊靈活、方便等特點，主要用於多設備、短距離通訊的場合，例如可應用於：

- 串列記憶體，如 E2PROM、RAM 等；
- 串列週邊設備，如 LCD 控制器、時鐘晶片和音調發生器等。

4.10.5 暫存器描述

I2C 模組相關的暫存器共有 12 個，如表 4-100。

表 4-100 I2C 相關暫存器列表

暫存器名稱	助記符	位址
I2C GPIO 設置暫存器	P_I2C_GPIO_SETUP	0x88200044
I2C GPIO 輸入資料暫存器	P_I2C_GPIO_INPUT	0x88200074
I2C GPIO 外部中斷暫存器	P_I2C_GPIO_INT	0x88200098
I2C 介面選擇暫存器	P_I2C_INTERFACE_SEL	0x88200004
I2C 時鐘配置暫存器	P_I2C_CLK_CONF	0x88210094
I2C 控制暫存器	P_I2C_MODE_CTRL	0x88130020
I2C 中斷狀態暫存器	P_I2C_INT_STATUS	0x88130024
I2C 傳輸速率設置暫存器	P_I2C_RATE_SETUP	0x88130028
I2C 從機位址暫存器	P_I2C_SLAVE_ADDR	0x8813002C
I2C 資料位址暫存器	P_I2C_DATA_ADDR	0x88130030
I2C 發送資料暫存器	P_I2C_TX_DATA	0x88130034
I2C 接收資料暫存器	P_I2C_RX_DATA	0x88130038

如果 I2C 介面複用為 GPIO 功能，可以對表 4-100 的前三個暫存器操作。暫存器的各位對應插腳關係如表 4-142：

表 4-101 I2C 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
I2C_CLK	35	bit[0]	可以
I2C_DATA	36	bit[1]	可以

■ I2C GPIO 設置暫存器：P_I2C_GPIO_SETUP(0x88200044)

I2C GPIO 設置暫存器的功能是：I2C 複用 GPIO 使用時，設置輸出使能、上/下拉電阻輸入和輸出資料。

表 4-102 P_I2C_GPIO_SETUP(0x88200044)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	W	-	W	-	W	-	W
預設值	-	0x3	-	0	-	0	-	0
名稱	-	XI2C_PD	-	XI2C_PU	-	I2C_OE	-	I2C_O

XI2C_PD b25~b24 I2C GPIO 下拉使能位：

0：禁止為下拉埠

1：使能為下拉埠

XI2C_PU b17~b16 I2C GPIO 上拉使能位：

0：使能為上拉埠

1：禁止為上拉埠

I2C_OE b9~b8 I2C GPIO 輸出使能位：

0：禁止位輸出埠

1：使能為輸出埠

I2C_O b1~b0 I2C GPIO 輸出資料

注意：

b0、b8、b16、b24 控制 I2C_CLK 管腳；b1、b9、b17、b25 控制 I2C_DATA 管腳。

■ I2C GPIO 輸入資料暫存器：P_I2C_GPIO_INPUT(0x88200074)

I2C GPIO 輸入資料暫存器各位功能如下：

表 4-103 P_I2C_GPIO_INPUT(0x88200074)

位	b31~b26	b25~b24	b23~b0
讀/寫	-	R	-
預設值	-	0	-
名稱	-	I2C_IN	-

I2C_IN

b25~b24

I2C 埠作為 GPIO 的輸入資料

■ I2C GPIO 外部中斷暫存器：P_I2C_GPIO_INT(0x88200098)

I2C GPIO 外部中斷暫存器各位功能如下：

表 4-104 P_I2C_GPIO_INPUT(0x88200074)

位	b31~b26	b25~b24	b23~b18	b17~b16	b15~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	-	W	-	W
預設值	-	0	-	0	-	0	-	0
名稱	-	I2C_FI	-	I2C_RI	-	I2C_FIEN	-	I2C_RIEN

I2C_FI

b25~b24

I2C GPIO 下降沿中斷旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

I2C_RI	b17~b16	I2C GPIO 上升沿中斷旗標位： 讀 0：沒有發生上升沿中斷 讀 1：發生上升沿中斷 寫 0：無意義 寫 1：清除中斷旗標
I2C_FIEN	b9~b8	I2C GPIO 下降沿中斷使能位： 0：禁止下降沿中斷 1：使能下降沿中斷
I2C_RIEN	b1~b0	I2C GPIO 上升沿中斷使能位： 0：禁止上升沿中斷 1：使能上升沿中斷

■ I2C 介面選擇暫存器：P_I2C_INTERFACE_SEL(0x88200004)

透過 I2C 介面選擇暫存器選擇 I2C 介面的複用功能。

表 4-105 P_I2C_INTERFACE_SEL(0x88200004)

位	b31~b1	b0
讀/寫	-	R/W
預設值	-	0
名稱	-	SW_I2C

SW_I2C	b0	I2C 複用埠功能選擇位： 0：作為 GPIO 1：作為 I2C 介面
--------	----	---

■ I2C 時鐘配置暫存器：P_I2C_CLK_CONF(0x88210094)

透過 I2C 時鐘配置暫存器可以停止/打開 I2C 時鐘或者重設/不重設 I2C 模組，在使能 I2C 前需先打開 I2C 時鐘。

表 4-106 P_I2C_CLK_CONF(0x88210094)

位	b31~b2	b1	b0
讀/寫	-	W	W
預設值	-	1	0
名稱	-	I2C_RST	I2C_STOP

I2C_RST	b1	I2C 模組時鐘重設位： 0 : I2C 模組時鐘重設 1 : I2C 模組時鐘不重設
I2C_STOP	b0	I2C 模組時鐘使能位： 0 : I2C 模組時鐘停止 1 : I2C 模組時鐘使能

■ I2C 傳輸速率設置暫存器：P_I2C_RATE_SETUP(0x88130028)

透過 I2C 傳輸速率設置暫存器可以設置其傳輸速率。

表 4-107 P_I2C_RATE_SETUP(0x88130028)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	CLK_SET

CLK_SET	b9~b0	I2C 時鐘頻率設置位。I2C 時鐘頻率 =27M/4/CLK_SET。
---------	-------	---

■ I2C 控制暫存器：P_I2C_MODE_CTRL(0x88130020)

I2C 控制暫存器用來控制開始/停止資料傳輸，選擇讀（接收）/寫（發送）模式，選擇傳輸的資料位元數等。

表 4-108 P_I2C_MODE_CTRL(0x88130020)

位	b8	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	W	W	W	R	R	R	W	W	W
預設值	0	0	0	0	0	0	0	0	0
名稱	STOPN	GAK	MODE	ACKN	ACK16	ACK8	STARTN	START16	START8

STOPN	b8	停止 $8 \times n$ 位資料傳輸使能位： 0 : 不停止 $8 \times n$ 位資料傳輸 1 : 停止 $8 \times n$ 位資料傳輸
GAK	b7	傳輸結束時 ACK/NAK 應答信號位： 0 : 以 NAK (非 ACK) 結束 1 : 以 ACK 結束
MODE	b6	發送器/接收器模式選擇位： 0 : 選擇為發送器模式 1 : 選擇為接收器模式
ACKN	b5	普通資料傳輸應答位： 0 : 無應答 1 : $8 \times n$ 位資料傳輸應答
ACK16	b4	16 位 (半字) 資料傳輸應答位： 0 : 無應答 1 : 16 位 (半字) 資料傳輸應答
ACK8	b3	8 位 (單字節) 資料傳輸應答位： 0 : 無應答 1 : 8 位 (單字節) 資料傳輸應答
STARTN	b2	普通資料傳輸起始位： 0 : 無操作 1 : 開始 $8 \times n$ 位資料傳輸

START16	b1	16 位（半字）資料傳輸起始位： 0：無操作 1：開始 16 位（半字）的資料傳輸
START8	b0	8 位（單字節）資料傳輸起始位： 0：無操作 1：開始 8 位（單字節）的資料傳輸

■ I2C 中斷狀態暫存器：P_I2C_INT_STATUS(0x88130024)

I2C 中斷狀態暫存器各位功能如下：

表 4-109 P_I2C_INT_STATUS(0x88130024)

位	b1	b0
讀/寫	W	R/W
預設值	0	0
名稱	INT_EN	INT_FLAG

INT_EN	b1	I2C 發送/接收中斷使能位： 0：遮罩發送/接收中斷 1：使能發送/接收中斷
INT_FLAG	b0	I2C 發送/接收中斷旗標位： 讀 0：沒有發生 I2C 發送/接收中斷 讀 1：發生 I2C 發送/接收中斷 寫 0：無意義 寫 1：清中斷旗標

■ I2C 從機位址暫存器 P_I2C_SLAVE_ADDR(0x8813002C)

I2C 從機位址暫存器各位功能如下：

表 4-110 P_I2C_SLAVE_ADDR(0x8813002C)

位	b31~b8	b7~b1	b0
讀/寫	-	W	-
預設值	-	0	-
名稱	-	ID_REG	-

ID_REG

b7~b1

I2C 從機位址，共 7 位

■ I2C 資料位址暫存器：P_I2C_DATA_ADDR(0x88130030)

I2C 資料位址暫存器用來指定運算元據的源位址，即源資料在從機設備中的位址，例如外部 RAM 中的存儲資料位址等。

表 4-111 P_I2C_DATA_ADDR(0x88130030)

位	b31~b8	b7~b0
讀/寫	-	W
預設值	-	0
名稱	-	ADDR

ID_REG

b7~b0

I2C 目標位址

■ I2C 發送資料暫存器：P_I2C_TX_DATA(0x88130034)

I2C 發送資料暫存器各位功能如下：

表 4-112 P_I2C_TX_DATA(0x88130034)

位	b31~b16	b15~b0
讀/寫	-	W
預設值	-	0
名稱	-	I2C_WDATA

I2C_WDATA

b15~b0

I2C 準備發送的資料。

■ I2C 接收資料暫存器：**P_I2C_RX_DATA(0x88130038)**

I2C 接收資料暫存器各位功能如下：

表 4-113 P_I2C_RX_DATA(0x88130038)

位	b15~b0
讀/寫	R
預設值	0
名稱	I2C_RDATA

I2C_RDATA	b15~b0	I2C 接收到的資料。
-----------	--------	-------------

4.10.6 基本操作

SPCE3200 可以作為 I2C 汇流排的主機，可以和 I2C 汇流排上的從機進行資料傳輸，包括資料發送和接收，因此 SPCE3200 的 I2C 通訊包括主發送和主接收。

1. 8bit 模式發送與接收

參考程式段如下：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
// 8bit 發送與接收
int main(void)
{
    unsigned int i = 0;                                // 軟體延時使用
    unsigned int j = 0;                                // 發送的資料與位址
    unsigned int uiData = 0;                            // 接收資料

    *P_I2C_CLK_CONF = C_I2C_CLK_EN
        | C_I2C_RST_DIS;                           // 使能 I2C 模組時鐘
    *P_I2C_INTERFACE_SEL = C_I2C_PORT_SEL;           // 埠選擇 I2C
    *P_I2C_RATE_SETUP = 67;                          // 傳輸速率在 100K，27M/4/67 約為 100K
    *P_I2C_INT_STATUS = C_I2C_INT_EN;                // I2C 中斷使能
    while(1)
    {
        j++;
        *P_I2C_SLAVE_ADDR = 0xa0;                    // 器件 ID
        *P_I2C_DATA_ADDR = j;                        // 存儲位址
        *P_I2C_TX_DATA = j;                         // 寫入資料
        *P_I2C_MODE_CTRL = C_I2C_TX_MODE
            | C_I2C_8BIT_START;                     // I2C 選擇發送模式
            // 8bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 發送
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;          // 清除中斷
        while(!(*P_I2C_MODE_CTRL & C_I2C_8BIT_ACK)); // 等待應答信號
    }
}
```

```

        *P_I2C_MODE_CTRL = C_I2C_END_ACK;           // 發送停止位 P , 發送結束
        for(i=0;i<10000;i++);                      // 軟體延時

        *P_I2C_MODE_CTRL = C_I2C_RX_MODE           // I2C 選擇接收模式
            | C_I2C_8BIT_START;                     // 8bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 接收
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;        // 清除中斷
        uiData = *P_I2C_RX_DATA;                   // 接收資料
        *P_I2C_MODE_CTRL = C_I2C_8BIT_ACK          // 發送應答信號
            | C_I2C_END_ACK;                      // 發送停止位 P , 接收結束
    }
    return 0;
}

```

2. 16bit 模式發送與接收

參考程式如下：

```

#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
// 16bit 發送與接收
int main(void)
{
    unsigned int i = 0;                           // 軟體延時使用
    unsigned int j = 0;                           // 發送的資料與位址
    unsigned int uiData = 0;                      // 接收資料
    unsigned int uiTdata = 0x1111;                // 發送的資料

    *P_I2C_CLK_CONF = C_I2C_CLK_EN
        | C_I2C_RST_DIS;                         // 使能 I2C 模組時鐘
    *P_I2C_INTERFACE_SEL = C_I2C_PORT_SEL;       // 埠選擇 I2C
    *P_I2C_RATE_SETUP = 67;                      // 傳輸速率在 100K , 27M/4/67 約為 100K
    *P_I2C_INT_STATUS = C_I2C_INT_EN;             // I2C 中斷使能
    while(1)
    {
        j += 2;
        uiTdata += 1;
        *P_I2C_SLAVE_ADDR = 0xa0;                 // 器件 ID
        *P_I2C_DATA_ADDR = j;                     // 存儲位址
        *P_I2C_TX_DATA = uiTdata;                // 寫入資料
        *P_I2C_MODE_CTRL = C_I2C_TX_MODE
            | C_I2C_16BIT_START;                  // I2C 選擇發送模式
            // 16bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 發送
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;        // 清除中斷
        while(!(*P_I2C_MODE_CTRL & C_I2C_16BIT_ACK)); // 等待應答信號
        *P_I2C_MODE_CTRL = C_I2C_END_ACK;           // 發送停止位 P , 發送結束
        for(i=0;i<10000;i++);                      // 軟體延時

        *P_I2C_MODE_CTRL = C_I2C_RX_MODE           // I2C 選擇接收模式
            | C_I2C_16BIT_START;                  // 16bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 接收
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;        // 清除中斷
        uiData = *P_I2C_RX_DATA;                   // 接收資料
        *P_I2C_MODE_CTRL = C_I2C_16BIT_ACK          // 發送應答信號
            | C_I2C_END_ACK;                      // 發送停止位 P , 接收結束
    }
}

```

```
    }
    return 0;
}
```

■ 多位元組資料發送與接收

多位元組資料發送的操作相對比較複雜，中斷方式和查詢方式比較類似，所不同的是中斷方式可以在中斷服務程式中判斷是否返回應答信號，從而決定是否傳輸下一個資料。這裏只介紹比較常用的查詢方式。

在介紹多位元組資料發送模式時介紹到，多位元組資料發送可以看作多個獨立 Phase 的資料發送過程，按照圖 4-50單個 Phase 資料發送的時序圖，1 個 Phase 的資料發送流程如圖 4-49。

使用 8N bit 模式發送接收 1 個位元組，參考程式如下：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
// 8N bit 模式發送與接收 1 個位元組
int main(void)
{
    unsigned int i = 0;                                // 軟體延時使用
    unsigned int j = 0;                                // 發送的資料與位址
    unsigned int uiData = 0;                            // 接收資料
    unsigned int uiTdata = 0x11;                         // 發送的資料

    *P_I2C_CLK_CONF = C_I2C_CLK_EN
        | C_I2C_RST_DIS;                             // 使能 I2C 模組時鐘
    *P_I2C_INTERFACE_SEL = C_I2C_PORT_SEL;           // 埠選擇 I2C
    *P_I2C_RATE_SETUP = 67;                           // 傳輸速率在 100K ,
    27M/4/27                                         // 約為 100K
    *P_I2C_INT_STATUS = C_I2C_INT_EN;                // I2C 中斷使能
    while(1)
    {
        j += 1;                                      // 選擇器件位址
        uiTdata += 1;                                // 改變寫入資料
        *P_I2C_TX_DATA = 0xa0;                        // 器件 ID
        *P_I2C_MODE_CTRL = C_I2C_TX_MODE
            | C_I2C_8NBIT_START;                     // I2C 選擇發送模式
            // 8N bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;          // 清除中斷
        while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號

        *P_I2C_TX_DATA = j;                          // 資料位址
        *P_I2C_MODE_CTRL = C_I2C_TX_MODE
            | C_I2C_8NBIT_START;                     // I2C 選擇發送模式
            // 8N bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
        *P_I2C_INT_STATUS |= C_I2C_INT_FLAG;          // 清除中斷
        while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號

        *P_I2C_TX_DATA = uiTdata;                    // 發送的資料
        *P_I2C_MODE_CTRL = C_I2C_TX_MODE
            | C_I2C_8NBIT_START;                     // I2C 選擇發送模式
            // 8N bit 模式
        while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
```

```
*P_I2C_INT_STATUS |= C_I2C_INT_FLAG; // 清除中斷
while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號

*P_I2C_MODE_CTRL = C_I2C_8NBIT_STOP; // 發送停止位 P，發送結束
for(i=0;i<10000;i++); // 軟體延時

*P_I2C_TX_DATA = 0xa0; // 器件 ID
*P_I2C_MODE_CTRL = C_I2C_TX_MODE // I2C 選擇發送模式
| C_I2C_8NBIT_START; // 8N bit 模式
while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
*P_I2C_INT_STATUS |= C_I2C_INT_FLAG; // 清除中斷
while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號

*P_I2C_TX_DATA = j; // 資料位址
*P_I2C_MODE_CTRL = C_I2C_TX_MODE // I2C 選擇發送模式
| C_I2C_8NBIT_START; // 8N bit 模式
while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
*P_I2C_INT_STATUS |= C_I2C_INT_FLAG; // 清除中斷
while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號
*P_I2C_MODE_CTRL = C_I2C_8NBIT_STOP; // 發送停止位 P，發送結束

*P_I2C_TX_DATA = 0xa0 | 1; // 器件 ID，最低位為 1 表示讀操作
*P_I2C_MODE_CTRL = C_I2C_TX_MODE // I2C 選擇發送模式
| C_I2C_8NBIT_START; // 8N bit 模式
while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 等待發送
*P_I2C_INT_STATUS |= C_I2C_INT_FLAG; // 清除中斷
while(!(*P_I2C_MODE_CTRL & C_I2C_8NBIT_ACK)); // 等待應答信號

*P_I2C_MODE_CTRL = C_I2C_RX_MODE // I2C 選擇接收模式
| C_I2C_8NBIT_START; // 8N bit 模式
while(!(*P_I2C_INT_STATUS & C_I2C_INT_FLAG)); // 接收
*P_I2C_INT_STATUS |= C_I2C_INT_FLAG; // 清除中斷
uiData = *P_I2C_RX_DATA; // 接收資料
*P_I2C_MODE_CTRL = C_I2C_8NBIT_STOP; // 發送停止位 P，發送結束

}
}
```

4.10.7 注意事項

使用 SPCE3200 的 I2C 模組時要注意下面幾點：

- 設置 I2C 時鐘頻率要考慮通訊另一方（從機）支援的時鐘頻率；
- SPCE3200 只能作為 I2C 的主機使用。

4.11 SIO 控制器

4.11.1 概述

串列 I/O 介面 (SIO) 是凌陽科技公司自主產權所有的一種串列介面，用於與其他設備進行通訊。這種串聯埠透過 2 個管腳 (SCK 和 SDA) 就可以進行資料的發送與接收操作。SIO 可以

方便的與凌陽其他具有 SIO 介面的晶片進行通訊，如 SPR 系列存儲晶片、帶 SIO 介面的其他凌陽單片機等。

SIO 協議由起始位元、讀/寫控制位元、位址字、資料字以及停止位元組成。圖 4-57、圖 4-58 分別為 SIO 協定的寫和讀方式的時序，圖中 SCK 為時鐘信號，而 SDA 則為雙向傳輸的串列資料信號。若在 SCK 為高電平期間 SDA 完成了由 1 到 0 的跳變，這就是 SIO 協議規定的傳輸的“起始位”；相反，在 SCK 為高電平期間 SDA 完成了由 0 到 1 的跳變，即為協議規定的傳輸的“停止位”。除了 SIO 的“起始位/停止位”以外，SDA 只能在 SCK 為低電平時改變其邏輯電平。另外，位址字和資料字的傳輸是以最高有效位（MSB）開頭的；而且，若資料字為 16 位元的，則 SIO 首先發送/接收的是低位元組資料，然後才是高位元組資料。

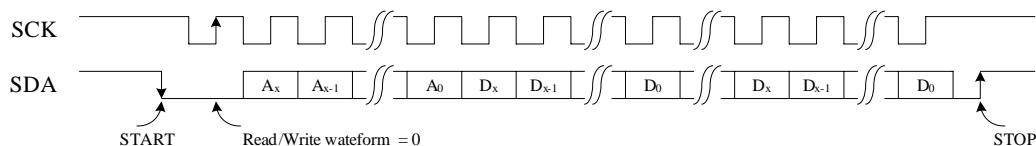


圖 4-57 SIO 寫操作方式時序

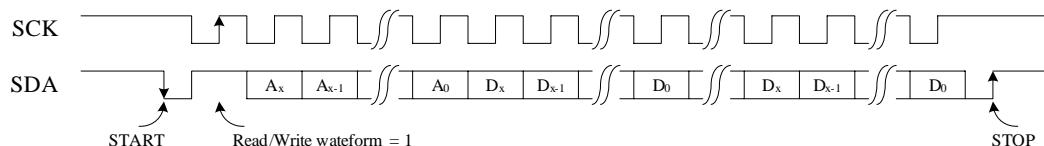


圖 4-58 SIO 讀操作方式時序

4.11.2 特性

SIO 控制器具有如下一些技術特性：

- 透過 SCK 和 SDA 插腳提供主模讀/寫功能
- 支援 4 種位址方式：無位址、8 位位址、16 位位址以及 24 位位址
- 支援脈衝讀/寫信號功能
- 支援 8 位/16 位資料寬度
- 可透過中斷/查詢兩種方式實現資料發送/接收（SIO 控制器使用 IRQ44 中斷）
- 支援 4 種 SIO 串列傳輸速率
- 提供與 SPDS301 晶片通訊的自動位元流傳輸方式

4.11.3 插腳描述

SPCE3200 的 SIO 介面插腳與 SJTAG 和 I2S 介面插腳複用，在使用時需要設置相關暫存器以便使其工作在 SIO 介面方式。SIO 介面的插腳描述如表 4-114 所示。

表 4-114 SIO 介面插腳對應表

插腳名稱	插腳號	插腳屬性	插腳功能
SIO_RDY	149	I	SIO 介面的狀態檢測插腳
SIO_SCK	150	O	SIO 介面的時鐘輸出插腳
SIO_SDA	151	I/O	SIO 介面的資料插腳

4.11.4 結構

SIO 控制器的結構如圖 4-59 所示。

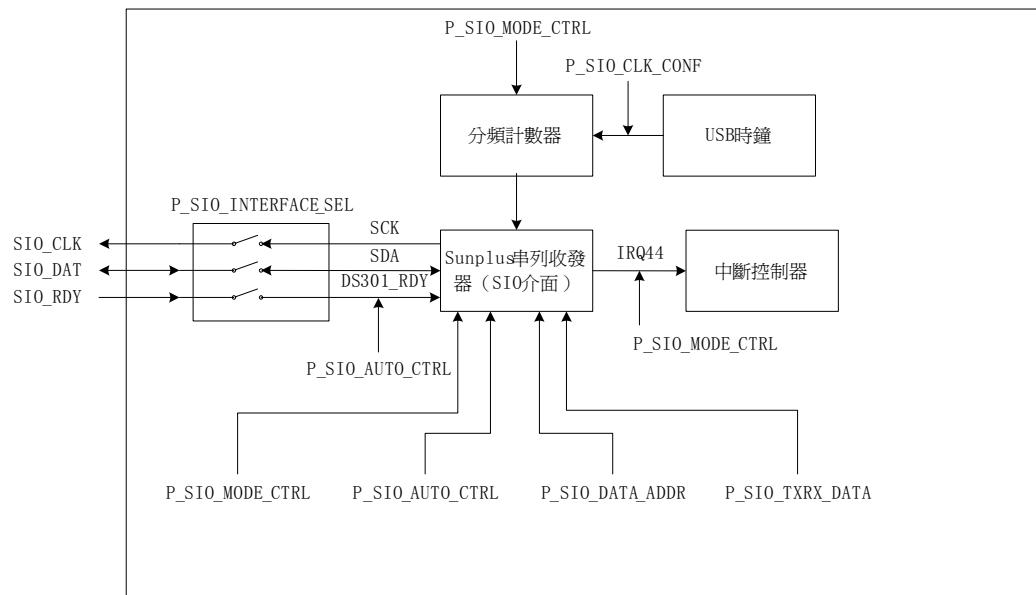


圖 4-59 SIO 控制器結構

4.11.5 暫存器描述

SIO 控制器共有 9 個控制暫存器，如表 4-115 所示。透過對這 9 個控制暫存器的操作，即可實現 SIO 模組的所有功能。

表 4-115 SIO 相關暫存器列表

暫存器名稱	助記符	位址
SIO 介面選擇暫存器	P_SIO_INTERFACE_SEL	0x88200004
JTAG GPIO 設置暫存器	P_JTAG_GPIO_SETUP	0x88200034
JTAG GPIO 輸入資料暫存器	P_JTAG_GPIO_INPUT	0x88200070

暫存器名稱	助記符	位址
JTAG GPIO 外部中斷暫存器	P_JTAG_GPIO_INT	0x8820008C
SIO 時鐘配置暫存器	P_SIO_CLK_CONF	0x882100A0
SIO 控制暫存器	P_SIO_MODE_CTRL	0X88120000
SIO 自動傳輸控制暫存器	P_SIO_AUTO_CTRL	0x88120004
SIO 資料位址暫存器	P_SIO_DATA_ADDR	0x88120008
SIO 收發資料暫存器	P_SIO_TXRX_DATA	0x8821000C

如果使用 JTAG 介面（JTAG 模組與 SIO 模組複用插腳）的 GPIO 功能，需要對下面的暫存器操作。暫存器的各位對應插腳關係如：

表 4-116 SIO 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
JTAG_TRSTN	153	bit[0]	可以
JTAG_TDI	152	bit[1]	可以
JTAG_TDO	151	bit[2]	可以
JTAG_TCK	150	bit[3]	可以
JTAG_TMS	149	bit[4]	可以

■ JTAG GPIO 設置暫存器：P_JTAG_GPIO_SETUP(0x88200034)

JTAG GPIO 設置暫存器設置 JTAG 介面複用為 GPIO 時的插腳輸出使能、輸出資料、上拉電阻輸入使能和下拉電阻輸入使能。

表 4-117 P_JTAG_GPIO_SETUP(0x88200034)

位	b28~b24	b23~b21	b20~b16	b15~b13	b12~b8	b7~b5	b4~b0
讀/寫	R/W	-	R/W	-	R/W	-	R/W
預設值	1		1		0	-	0
名稱	JTAG_PD		JTAG_PU		JTAG_OE	-	JTAG_O

JTAG_PD	b28~b24	JTAG 介面作為 GPIO 使用時的下拉電阻輸入使能位： 0：禁止下拉電阻輸入 1：使能下拉電阻輸入
JTAG_PU	b20~b16	JTAG 介面作為 GPIO 使用時的上拉電阻輸入使能位： 0：使能上拉電阻輸入 1：禁止上拉電阻輸入
JTAG_OE	b12~b8	JTAG 介面作為 GPIO 使用時的輸出使能位： 0：禁止輸出 1：使能輸出
JTAG_O	b4~b0	JTAG 介面作為 GPIO 使用時的輸出資料

■ JTAG GPIO 輸入資料暫存器：**P_JTAG_GPIO_INPUT(0x88200070)**

表 4-118 P_JTAG_GPIO_INPUT(0x88200070)

位	b31~b5	b4~b0
讀/寫	-	R
預設值	-	0
名稱	-	JTAG_INPUT

JTAG_INPUT	b4~b0	JTAG 介面作為 GPIO 使用時的輸入資料
------------	-------	-------------------------

■ JTAG GPIO 外部中斷暫存器：**P_JTAG_GPIO_INT(0x8820008C)**

JTAG GPIO 外部中斷暫存器可進行中斷使能、中斷觸發沿設置、中斷旗標清除等操作。

表 4-119 P_JTAG_GPIO_INT(0x8820008C)

位	b28~b24	b23~b21	b20~b16	b15~b13	b12~b8	b7~b5	b4~b0
讀/寫	R/W	-	R/W	-	R/W	-	R/W
預設值	0	-	0	-	0	-	0
名稱	JTAG_FI	-	JTAG_RI	-	JTAG_FIEN	-	JTAG RIEN

JTAG_FI	b28~b24	JTAG GPIO 下降沿中斷旗標位： 讀 0：沒有發生下降沿中斷 讀 1：發生下降沿中斷 寫 0：無意義 寫 1：清除中斷旗標
JTAG_RI	b20~b16	JTAG GPIO 上升沿中斷旗標位： 讀 0：沒有發生上升沿中斷 讀 1：發生上升沿中斷 寫 0：無意義 寫 1：清除中斷旗標
JTAG_FIEN	b12~b8	JTAG GPIO 下降沿中斷使能位： 0：禁止下降沿中斷 1：使能下降沿中斷
JTAG RIEN	b4~b0	JTAG GPIO 上升沿中斷使能位： 0：禁止上升沿中斷 1：使能上升沿中斷

作為 SIO 控制器需要設置以下暫存器：

■ **SIO 介面選擇暫存器：P_SIO_INTERFACE_SEL(0x88200004)**

SPCE3200 的 SIO 介面插腳與 SJTAG 和 I2S 介面插腳複用，在使用時需要設置 SIO 介面選擇暫存器以便使其工作在 SIO 介面方式。

表 4-120 P_SIO_INTERFACE_SEL(0x88200004)

位	b31~b11	b10~b8	b7~b0
讀/寫	-	W	-
預設值	-	0	-
名稱	-	SW_PERI	-

SW_PERI	b10~b8	SIO、JTAG 和 I2S 複用介面選擇位：
		000：選擇做為 JTAG 埠使用
		001：選擇做為 SIO 埠使用
		010：選擇做為 I2S 主模組埠使用
		011：選擇做為 I2S 從模組埠使用
		其他：不作為週邊設備埠使用

■ **SIO 時鐘配置暫存器：P_SIO_CLK_CONF(0x882100A0)**

表 4-121 P_SIO_CLK_CONF(0x882100A0)

位	b31-b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	SIO_RST	SIO_STOP

SIO_RST	b1	SIO 模組時鐘使能位：
---------	----	--------------

0：SIO 模組時鐘重設

1：SIO 模組時鐘使能

SIO_STOP	b0	SIO 模組時鐘停止位：
----------	----	--------------

0：SIO 模組時鐘停止

1：SIO 模組時鐘使能

■ **SIO 控制暫存器：P_SIO_MODE_CTRL(0x88120000)**

SIO 控制暫存器用於控制 SIO 收發器的工作模式等。

表 4-122 P_SIO_MODE_CTRL(0x88120000)

位	b31	b30	b15	b14	b13	b12
讀/寫	R	R	W	W	W	W
預設值	0	0	0	0	0	0
名稱	SIO_REQUEST	SIO_IRQ_STS	SIO_DATA_WIDTH	APBDMA_EN	IRQ_EN	PROTOCOL

位	b11~b10	b9~b8	b7	b2	b1	b0
讀/寫	W	W	W	W	W	W
預設值	0	0	0	0	0	0
名稱	BAUDRATE	ADDR_MOD_E	SIO_IRQ_CL_R	SIO_TRANS_FER	SIO_RW_CTL	SIO_START

SIO_REQUEST	b31	SIO 資料傳輸請求位（唯讀），當採用查詢方式進行資料傳輸時，讀取該位為 1 時： 在寫操作下，表示允許寫入資料； 在讀操作下，表示接收資料完畢，可以從 P_SIO_TXRX_DATA 暫存器讀取接收資料
SIO_IRQ_STS	b30	SIO 中斷狀態位（唯讀）： 讀 0：沒有發生 SIO 中斷 讀 1：發生 SIO 中斷
SIO_DATA_WIDTH	b15	SIO 資料位元寬控制位： 0：8 位元資料寬度 1：16 位資料寬度
APBDMA_EN	b14	APBDMA 通道使能位： 0：由 CPU 傳輸資料 1：由 APBDMA 傳輸資料（詳細請參考 APBDMA）
IRQ_EN	b13	SIO 中斷使能位： 0：禁止 SIO 中斷 1：使能 SIO 中斷（SIO 使用 IRQ44 中斷向量）
PROTOCOL	b12	SIO 協議讀/寫位元波形控制位： 0：寫操作下，讀/寫位波形為低電平；讀操作下，讀/寫位波形為高電平 1：無論寫模式還是讀模式，讀/寫位波形均為低電平
BAUDRATE	b11~b10	SIO 汇流排串列傳輸速率選擇位： 00：27/16MHz 01：27/4MHz 10：27/8MHz

11 : 27/32MHz

ADDR_MODE	b9~b8	SIO 位址寬度選擇位： 00 : 16 位位址寬度 01 : 無位址位 10 : 8 位位址寬度 11 : 24 位位址寬度
SIO_IRQ_CLR	b7	SIO 中斷旗標清除位： 寫入 1 即可將 SIO 中斷請求旗標清除
SIO_TRANSFER	b2	首先傳輸高/低位元組資料選擇位： 0 : 首先傳輸 SIO 收發資料暫存器的高位元組 1 : 首先傳輸 SIO 收發資料暫存器的低位元組
SIO_RW_CTL	b1	SIO 讀/寫操作選擇位： 0 : SIO 讀操作 1 : SIO 寫操作
SIO_START	b0	SIO 啓動/停止傳輸控制位： 0 : 停止 SIO 資料傳輸 1 : 啓動 SIO 資料傳輸

■ SIO 自動傳輸控制暫存器 : P_SIO_AUTO_CTRL(0x88120004)

SIO 自動傳輸控制暫存器可以控制 SIO 模組使用中斷方式自動傳輸一定數量的資料而不需要人工查詢幹預。

表 4-123 P_SIO_AUTO_CTRL(0x88120004)

位	b7~b4	b3~b2	b1	b0
讀/寫	W	-	R	W
預設值	15	-	1	0
名稱	SIO_WORD_NUM	-	DS301_READY	AUTOMATIC_EN

SIO_WORD_NUM	b7~b4	自動傳輸的字數： 以封包形式傳遞給 SPDS301 的資料字的字數，其預設值為 15，該值不能被設置成 0，否則將產生不可預知的錯誤，該參數值僅當 AUTOMATIC_EN 位元使能的狀態下有效
DS301_READY	b1	DS301_Ready 管腳檢測使能位： 0：禁止（不使用 DS301_Ready 管腳） 1：使能（使用 DS301_Ready 管腳）
AUTOMATIC_EN	b0	自動傳輸方式（與 SPDS301 通訊）使能控制位： 0：關閉自動傳輸 1：打開自動傳輸 打開自動傳輸方式後，SIO 模組將反復查詢反映 SPDS301 狀態的暫存器，直至讀到該暫存器裏 “decode work”位和 “request ready”位均為 1，此時會產生 SIO 中斷，通知用戶寫一個資料到 P_SIO_TXRX_DATA 暫存器中。寫完此資料字後，中斷會被自動清除。 在自動傳輸方式下每發生一次 SIO 中斷都要寫一個資料到 P_SIO_TXRX_DATA 暫存器中。關於 SPDS301 晶片的詳細資料，請參見 SPDS301 編程指南檔。

■ SIO 資料位址暫存器：P_SIO_DATA_ADDR(0x88120008)

SIO 資料位址暫存器可以設置 SIO 模組對外部設備的定址位址。

表 4-124 P_SIO_DATA_ADDR(0x88120008)

位	b23~b16	b15~b8	b7~b0
讀/寫	W	W	W
預設值	0	0	0
名稱	SIO_ADDRH	SIO_ADDRM	SIO_ADDRL

SIO_ADDRH	b23~b16	SIO 起始位址的高位元組部分，僅當 24 位元位址模式時使用
SIO_ADDRM	b15~b8	SIO 起始位址的中位元組部分，24 位元和 16 位元位址模式時使用
SIO_ADDRL	b7~b0	SIO 起始位址的低位元組部分，24 位元、16 位元和 8 位元位址模式時使用

■ SIO 收發資料暫存器：P_SIO_RXDATA(0x8812000C)

SIO 收發資料暫存器為 SIO 模組發送/接收資料的緩衝單元。向該單元寫入或讀出資料，SIO 模組將按照串列方式發送或接收資料。

表 4-125 P_SIO_RXDATA(0x8812000C)

位	b15~b8	b7~b0
讀/寫	R/W	R/W
預設值	0	0
名稱	SIO_RXDATAH	SIO_RXDATAL

SIO_RXDATAH	b15~b8	SIO 資料高位元組，僅當資料位元寬設置為 16 位時有效
SIO_RXDATAL	b7~b0	SIO 資料低位元組

4.11.6 基本操作

SIO 模組的操作有三種模式：

- 不使用“自動傳輸”模式；
- 使用自動傳輸模式，但不使用 DS301_Ready 插腳；
- 使用自動傳輸模式，並且使用 DS301_Ready 插腳。

下麵分別介紹：

■ 不使用“自動傳輸”模式：

在不使用“自動傳輸”模式下，用戶可以控制 SIO 模組進行資料的串列發送/接收操作。圖 4-60 所示的是 SIO 模組運行於“非自動傳輸”模式下的運行流程。

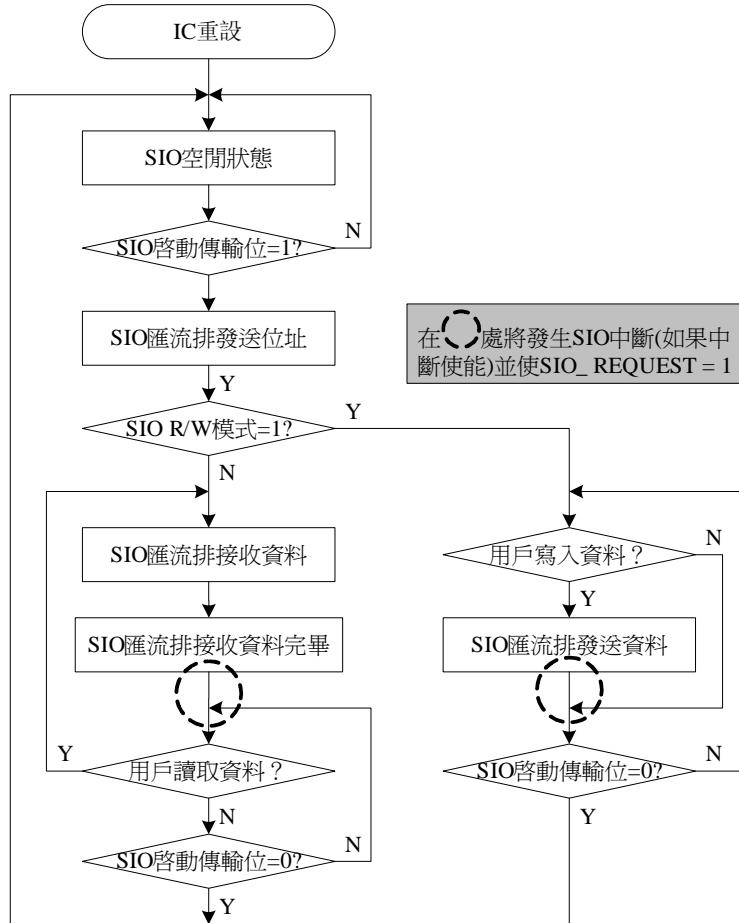


圖 4-60 不使用“自動傳輸”模式傳輸資料的操作流程

在該模式下使用 SIO 向週邊設備寫入資料的操作步驟如圖 4-61 所示。

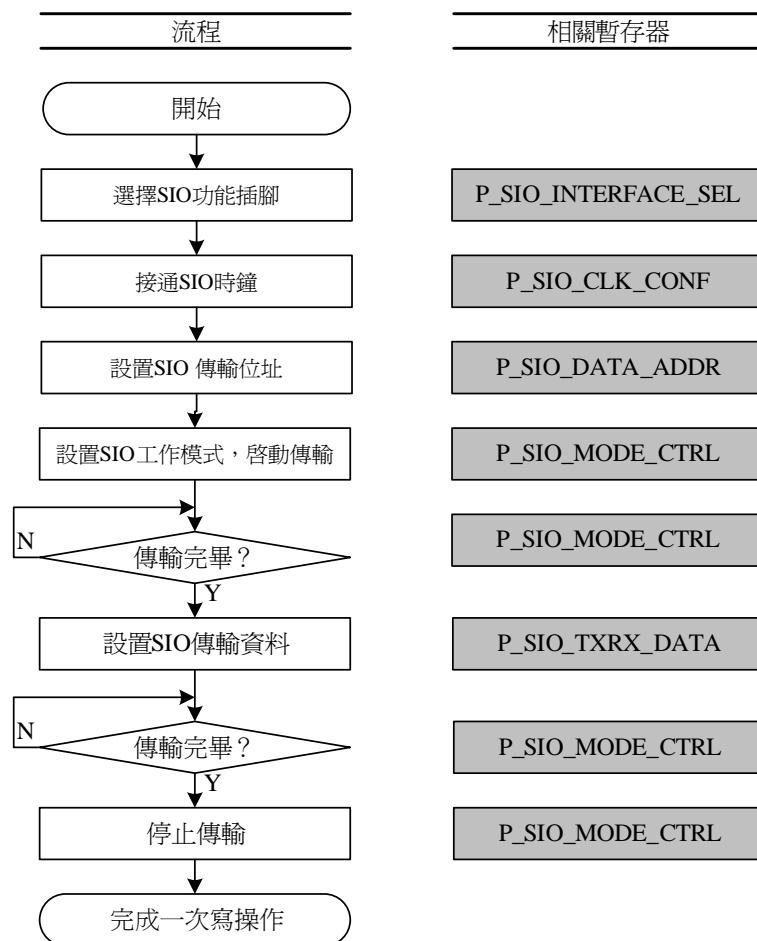


圖 4-61 SIO 寫操作流程圖

- (1) 設置 P_SIO_INTERFACE_SEL 暫存器，選擇管腳功能為 SIO 介面；
- (2) 設置 P_SIO_CLK_CONF 暫存器，使能 SIO 模組的時鐘；
- (3) 設置 P_SIO_TXRX_ADDR 暫存器，選擇週邊設備的定址位址（如果 SIO 設置了無位址模式，則可以跳過此步）；
- (4) 設置 P_SIO_MODE_CTRL 暫存器，根據需要設置 SIO 的工作模式，同時，使 SIO_START 位為 1，啓動 SIO 匯流排的傳輸；
- (5) 反複查詢 P_SIO_MODE_CTRL 的 b31 位 SIO_REQUEST 直至該位為 1，等待 SIO 傳輸就緒；
- (6) 向 P_SIO_TXRX_DATA 暫存器寫入資料；
- (7) 反複查詢 P_SIO_MODE_CTRL 的 b31 位 SIO_REQUEST 直至該位為 1，等待 SIO 傳輸就緒；
- (8) 傳輸完畢，設置 P_SIO_MODE_CTRL 暫存器的 SIO_START 位為 0。

參考代碼如下：

```
#include "SPCE3200_Register.h"
```

```

#include "SPC3200_Constant.h"
// 向 SPR4096 的 0x40 單元寫入資料 0x55
int main(void)
{
    int i;
    *P_CLK_PLLAU_CONF |= C_PLLU_CLK_EN; // 使能 PLLU
    *P_SIO_INTERFACE_SEL |= C_SIO_PORT_SEL; // 選擇 SIO 介面
    *P_SIO_CLK_CONF = C_SIO_CLK_EN + C_SIO_RST_DIS; // 使能 SIO 模組

    *P_SIO_DATA_ADDR = 0x40; // 設置操作位址
    *P_SIO_MODE_CTRL = C_SIO_DATA_8BIT // 選擇 8 位元資料寬度
        | C_SIO_RWBIT_NORMAL // 讀寫控制位選擇普通模式
        | C_SIO_CLK_27MDIV32 // 串列傳輸速率選擇 27MHz/32
        | C_SIO_ADDR_24BIT // 選擇 24 位址寬度
        | C_SIO_WRITE_MODE // 選擇寫操作方式
        | C_SIO_TXRX_START; // 啓動傳輸

    while((*P_SIO_MODE_CTRL & C_SIO_DATA_REQ) == 0); // 等待傳輸結束
    *P_SIO_TXRX_DATA = 0x55; // 設置待寫入的資料
    while((*P_SIO_MODE_CTRL & C_SIO_DATA_REQ) == 0); // 等待傳輸結束
    *P_SIO_MODE_CTRL = C_SIO_TXRX_STOP; // 停止傳輸
    for(i = 0; i < 10000; i++); // 等待 SPR4096 內部燒寫
    while(1);
    return 0;
}

```

使用 SIO 從週邊設備讀取資料的操作步驟如圖 4-62 所示。

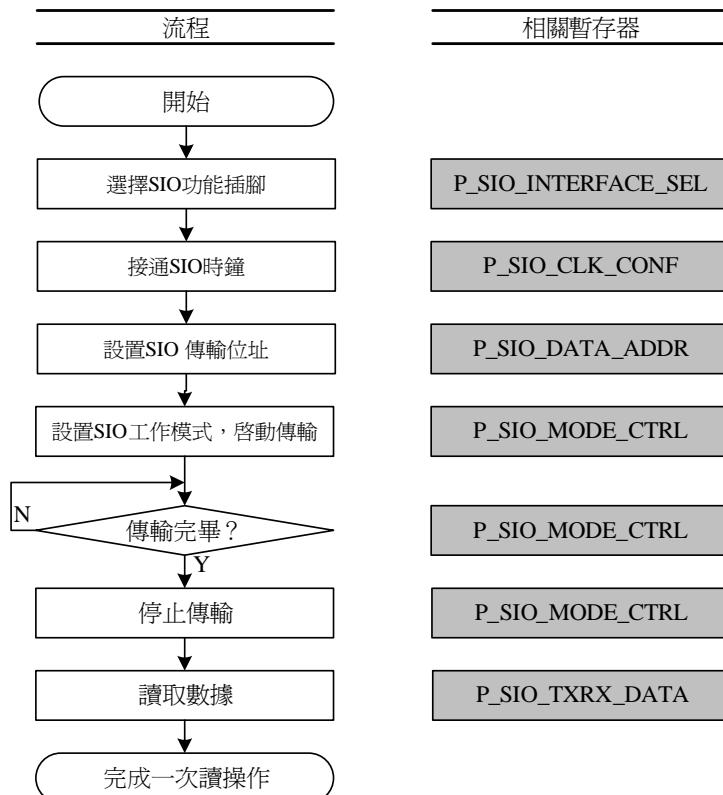


圖 4-62 SIO 讀操作流程圖

- (1) 設置 P_SIO_INTERFACE_SEL 暫存器，選擇管腳功能為 SIO 介面；
- (2) 設置 P_SIO_CLK_CONF 暫存器，使能 SIO 模組的時鐘；
- (3) 設置 P_SIO_TXRX_ADDR 暫存器，選擇週邊設備的定址位址（如果 SIO 設置了無位址模式，則可以跳過此步）；
- (4) 設置 P_SIO_MODE_CTRL 暫存器，根據需要設置 SIO 的工作模式，同時，使 SIO_START 位為 1，啓動 SIO 匯流排的傳輸；
- (5) 反復查詢 P_SIO_MODE_CTRL 的 b31 位 SIO_REQUEST 直至該位為 1，等待 SIO 傳輸就緒；
- (6) 設置 P_SIO_MODE_CTRL 暫存器的 SIO_START 位為 0，停止傳輸；
- (7) 從 P_SIO_TXRX_DATA 暫存器讀取資料；

參考代碼如下：

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
// 從 SPR4096 的 0x40 單元讀取資料
int main(void)
{
    unsigned int Data;
    *P_CLK_PLLAU_CONF |= C_PLLU_CLK_EN; // 使能 PLLU
    *P_SIO_INTERFACE_SEL |= C_SIO_PORT_SEL; // 選擇 SIO 介面
    *P_SIO_CLK_CONF = C_SIO_CLK_EN + C_SIO_RST_DIS; // 使能 SIO 模組

    *P_SIO_DATA_ADDR = 0x40; // 設置操作位址
    *P_SIO_MODE_CTRL = C_SIO_DATA_8BIT // 選擇 8 位元資料寬度
        | C_SIO_RWBIT_NORMAL // 讀寫控制位選擇普通模式
        | C_SIO_CLK_27MDIV32 // 串列傳輸速率選擇 27MHz/32
        | C_SIO_ADDR_24BIT // 選擇 24 位位址寬度
        | C_SIO_READ_MODE // 選擇讀操作方式
        | C_SIO_TXRX_START; // 啓動傳輸
    while((*P_SIO_MODE_CTRL & C_SIO_DATA_REQ) == 0); // 等待傳輸結束
    *P_SIO_MODE_CTRL = C_SIO_TXRX_STOP; // 停止傳輸
    Data = *P_SIO_TXRX_DATA; // 讀取資料
    while(1);
    return 0;
}
```

■ 使用“自動傳輸”模式，但不使用 DS301_Ready 管腳：

在該模式下，SIO 模組可以自動查詢 SPDS301 的狀態暫存器，以確定 SPDS301 是否處於空閒狀態，並自動重複發送過程，直至向 SPDS301 發送完畢用戶設置的 AutoTx_Num 個資料。在該模式下不能接收資料。圖 4-63 所示的是 SIO 模組運行於“自動傳輸”模式但不使用 DS301_Ready 管腳檢測的運行流程。

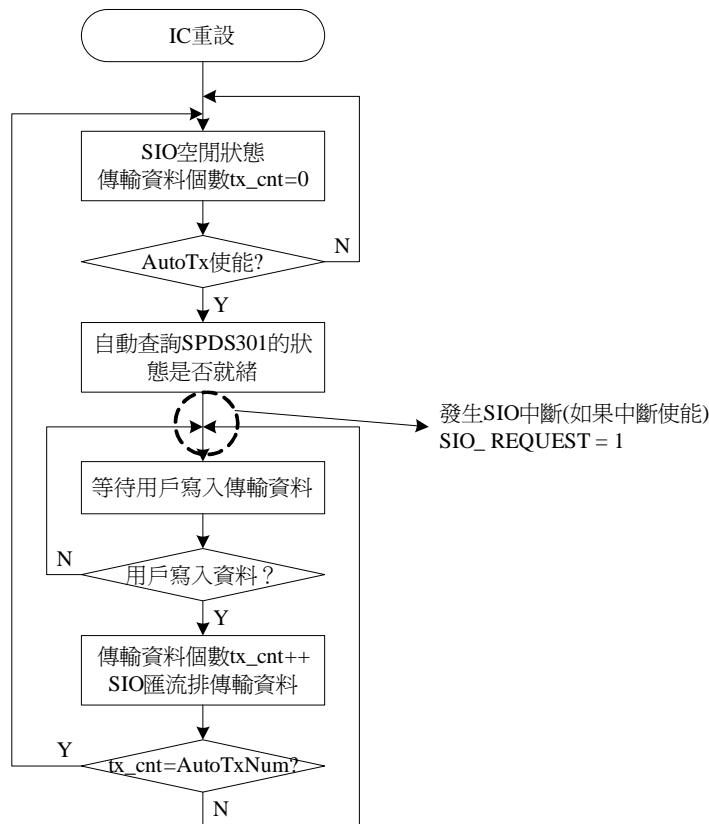


圖 4-63 使用“自動傳輸”模式但不使用 DS301_Ready 管腳傳輸資料的操作流程

■ 使用“自動傳輸”模式，並且使用 DS301_Ready 管腳：

在該模式下，SIO 模組查詢 DS301_Ready 管腳的狀態，以確定 SPDS301 是否處於空閒狀態，並自動重複發送過程，直至向 SPDS301 發送完畢用戶設置的 AutoTx_Num 個資料。在該模式下不能接收資料。圖 4-64 所示的是 SIO 模組運行於“自動傳輸”模式並且使用 DS301_Ready 管腳檢測的運行流程。

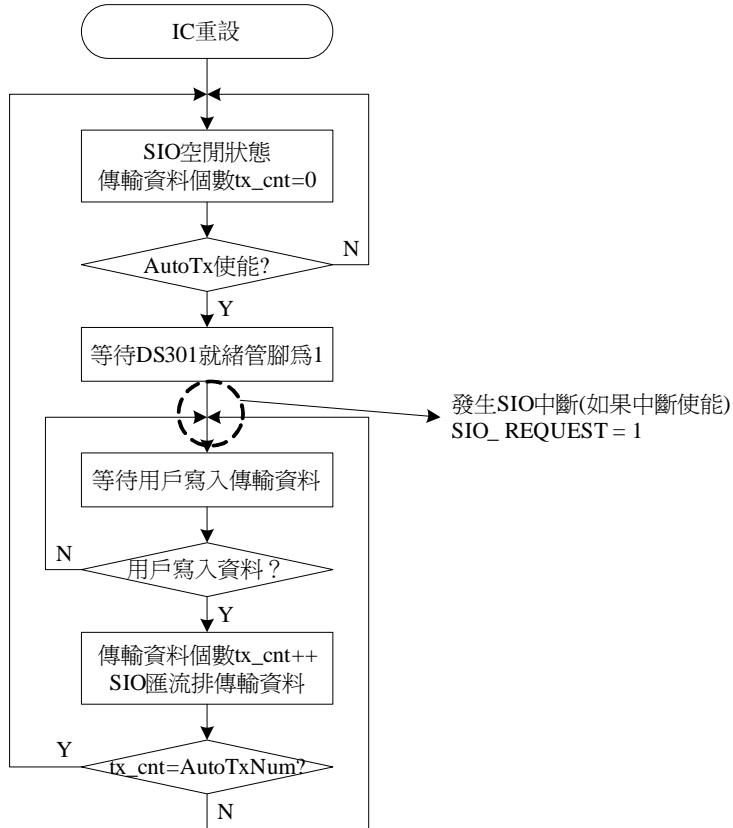


圖 4-64 使用“自動傳輸”模式同時使用 DS301_Ready 管腳傳輸資料的操作流程

“自動傳輸”模式下進行資料傳輸的操作流程在這裏不再詳述。

4.11.7 注意事項

- 由於 SIO 控制器由 PLLU 提供工作頻率，在使用 SIO 控制器之前，首先需要保證 P_CLK_PLLAU_CONF 暫存器的 b2 位為 1，使能 PLLU。
- SIO 中斷使能由 P_SIO_MODE_CTRL 的 b13 位控制，設置該位為 1 後將使能 SIO 中斷，用戶需要注意必須透過向該暫存器的 b7 位寫入 1 來清除中斷狀態。關於 SIO 中斷的正確操作如表 4-126 所示。

表 4-126 SIO 中斷操作說明

SIO 工作模式	中斷發生時 SIO 模組的狀態	正確的操作
不使用“自動傳輸”模式	當 SIO 中斷發生在讀操作下，說明 SIO 接收到 1 個位元組/字資料	讀出 P_SIO_TXRX_DATA 暫存器內接收到的資料，然後向 P_SIO_MODE_CTRL 暫存器的 b7 位元寫入 1 清除中斷旗標，以便 SIO 可以繼續接收資料。注意：即使清除了 SIO 的啟動/停止傳輸控制位，同樣需要清除中斷旗標

SIO 工作模式	中斷發生時 SIO 模組的狀態	正確的操作
	當 SIO 中斷發生在寫操作下，說明 SIO 已經完成上一個寫操作，等待下一個待寫入的資料	向 P_SIO_TXRX_DATA 暫存器內寫入資料後，當 SIO 模組完成資料傳輸後，將產生中斷，此時，應向 P_SIO_MODE_CTRL 暫存器的 b7 位寫入 1 以便清除中斷狀態。注意：即使清除了 SIO 的啓動/停止傳輸控制位，同樣需要清除中斷旗標
使用“自動傳輸”模式	SIO 中斷產生說明 SIO 模組完成了上一次的資料傳輸，等待用戶給出新的傳輸資料	向 P_SIO_TXRX_DATA 暫存器內寫入資料後，當 SIO 模組完成資料傳輸後，將產生中斷，此時，應向 P_SIO_MODE_CTRL 暫存器的 b7 位寫入 1 以便清除中斷旗標

4.12 Nor 型 Flash 控制器

4.12.1 概述

閃速記憶體（Flash Memory）具有非易失性，並且可輕易擦寫。Flash 技術結合了 OTP 記憶體的成本優勢和 EEPROM 的可再編程性能，因此，得到越來越廣泛的使用。

NOR 型 Flash 記憶體是一種採用標準匯流排介面與處理器交互的 Flash ROM，其特點是高速隨機位元組存取能力，以及晶片內執行（XIP, eXecute In Place），這樣應用程式可以直接在 Flash 快閃記憶體內運行，不必再把代碼讀到系統 RAM 中。NOR Flash 帶有 SRAM 介面，有足夠的位址插腳來定址，可以很容易地存取其內部的每一個位元組。

NOR 的傳輸效率很高，但是很低的寫入和擦除速度大大影響了它的性能。在 1~4MB 小容量時具有很高的成本效益，更加安全，不容易出現資料故障，因此，主要應用以代碼存儲為主，多與運算相關。

對 Nor 型 Flash 的操作包括讀操作、字寫入操作、扇區/塊/整片擦除操作和內部操作狀態檢測操作等。下面以 ST 公司的 SST39VF160 晶片為例，詳細介紹以上幾種操作。

SST39VF160 晶片具有 48 腳 TSOP 封裝和 48 腳 TFBGA 封裝兩種形式。TSOP 封裝的晶片插腳分佈如圖 4-65 所示。

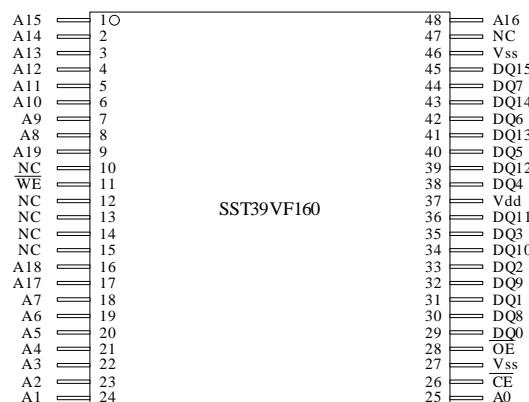


圖 4-65 SST39VF160 插腳分佈圖

插腳描述如表 4-127 所示。

表 4-127 SST39VF160 插腳功能描述

插腳符號	插腳名稱	功能描述
A19~A0	位址輸入	提供記憶體定址位址。在扇區擦除中 A19~A11 位址線用來選擇擦除哪一個扇區。在塊擦除操作中 A19~A15 位址線用來選擇擦除哪一個塊
DQ15~DQ0	資料輸入/輸出	在讀週期輸出資料，在寫週期接收寫入的資料並內部鎖存。在 \overline{CE} 或 \overline{OE} 為高電平時，資料線輸入為高阻態
\overline{CE}	片選使能	晶片選擇線，低電平有效
\overline{OE}	輸出使能	資料輸出使能線，低電平有效
\overline{WE}	寫使能	寫操作控制線
VDD	電源	為 SST39VF160 提供 2.7~3.6V 電源
VSS	地	
NC	不連接的插腳	

■ 讀操作

SST39VF160 的讀操作是由 \overline{CE} 和 \overline{OE} 信號線控制的。當兩者都為低時，處理器就可以從 SST39VF160 的輸出埠讀取資料。 \overline{CE} 是 SST39VF160 的片選線，當 \overline{CE} 為高，晶片未被選中。 \overline{OE} 是輸出使能信號線。當 \overline{CE} 或 \overline{OE} 中某一個為高時，SST39VF160 的資料線為高阻態。

SST39VF160 的讀操作的時序如圖 4-66 所示。

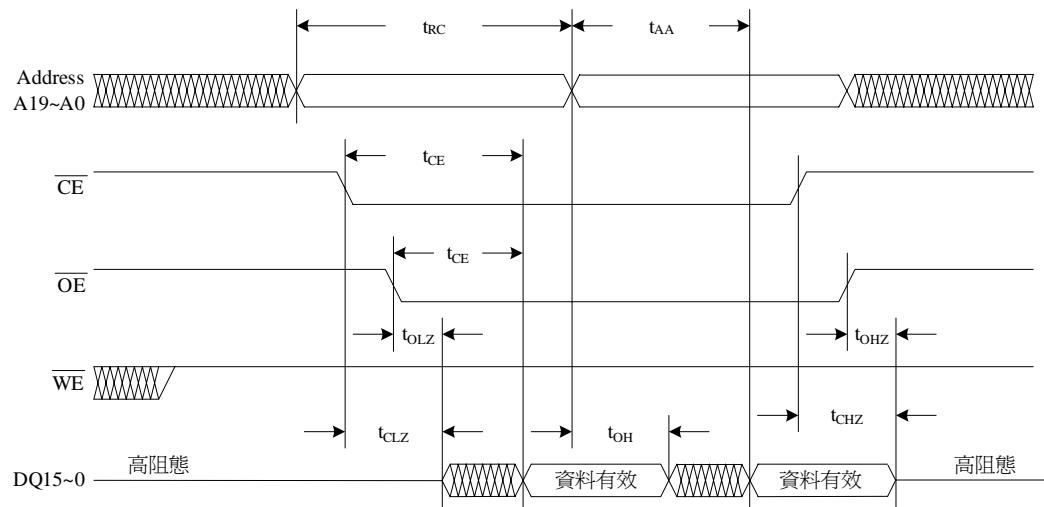


圖 4-66 SST39VF160 的讀時序

讀時序的參數如表 4-128 所示。

表 4-128 讀時序參數表 (單位: ns)

符號	描述	SST39VF160-70		SST39VF160-90	
		最小值	最大值	最小值	最大值
t_{RC}	整個讀週期時間	70		90	
t_{CE}	讀週期中晶片使能時間		70		90
t_{AA}	位址操作時間		70		90
t_{OE}	輸出資料使能時間		35		45
t_{CLZ}	從 \overline{CE} 變低到動態輸出時間	0		0	
t_{OLZ}	從 \overline{OE} 變低到動態輸出時間	0		0	
t_{CHZ}	從 \overline{CE} 變高到資料線高阻態時間		20		30
t_{OHZ}	從 \overline{OE} 變高到資料線高阻態時間		20		30
t_{OH}	位址改變時輸出資料保持時間	0		0	

■ 字寫入操作

SST39VF160 的寫操作主要是以一個字接一個字的方式進行寫入的。在寫入之前，扇區中如果有資料（非 0xFF），則必須首先進行充分地擦除。寫操作分三步進行：

第一步，送出“軟體資料保護”的 3 位元組；

第二步，送出位址和資料；

第三步，內部寫入處理階段，這個階段在第 4 個 \overline{WE} 或 \overline{CE} 的上升沿時被初始化。被初始化後，內部寫入處理就將在 $20 \mu s$ 時間內完成。在內部寫入階段，任何指令都將被忽略。字寫入操作流程如圖 4-67 所示。

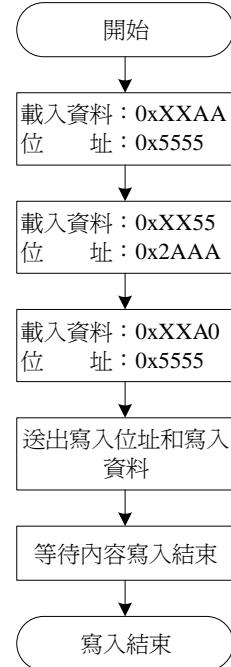


圖 4-67 SST39VF160 的字寫入流程圖

SST39VF160 的字寫入時序如圖 4-68 和圖 4-69 所示。

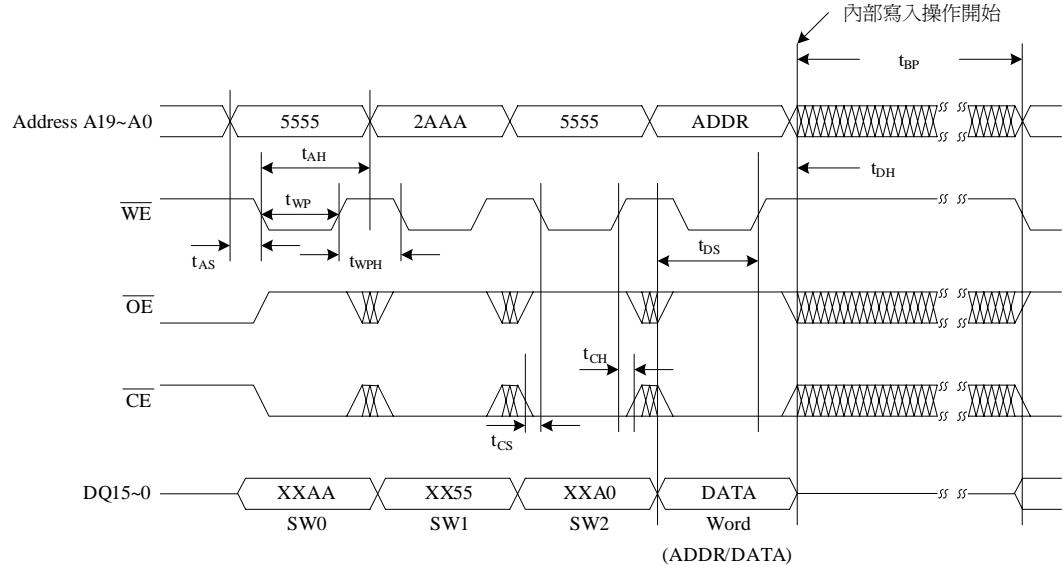
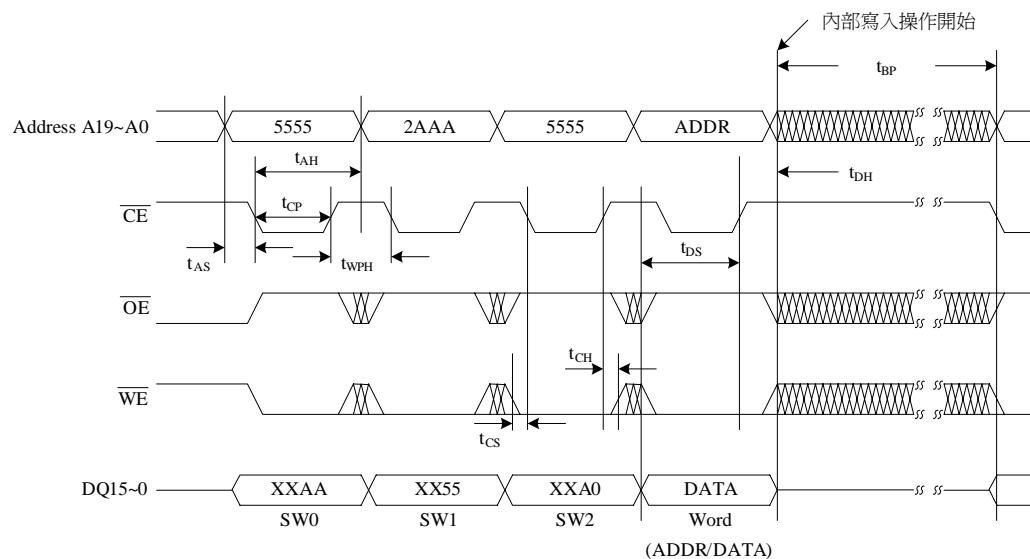


圖 4-68 透過 \overline{WE} 控制的字寫入時序圖


 圖 4-69 透過 \overline{CE} 控制的字寫入時序圖

SST39VF160 的寫入及擦除時序參數表如表 4-129 所示。

表 4-129 寫入及擦除時序參數表

符號	描述	最小值	最大值	單位
t_{BP}	字寫入時間		20	μs
t_{AS}	位址建立時間	0		ns
t_{AH}	位址保持時間	30		ns
t_{CS}	\overline{WE} 和 \overline{CE} 建立時間	0		ns
t_{CH}	\overline{WE} 和 \overline{CE} 保持時間	0		ns
t_{OES}	\overline{OE} 高電平建立時間	0		ns
t_{OEH}	\overline{OE} 高電平保持時間	10		ns
t_{CP}	\overline{CE} 脈寬	40		ns
t_{WP}	\overline{WE} 脈寬	40		ns
t_{WPH}	\overline{WE} 高電平脈寬	30		ns
t_{CPH}	\overline{CE} 高電平脈寬	30		ns
t_{DS}	資料建立時間	30		ns
t_{DH}	資料保持時間	0		ns
t_{IDA}	軟體 ID 操作和退出時間		150	ns

符號	描述	最小值	最大值	單位
t_{SE}	扇區擦除		25	ms
t_{BE}	塊擦除		25	ms
t_{SCE}	片擦除		100	ms

■ 扇區/塊/整片擦除操作

扇區或塊擦除操作允許 SST39VF 160 以一個扇區接一個扇區，或一個塊接一個塊地進行擦除。扇區是統一的 2K Word (16 位) 大小。塊是統一的 32K Word 大小。扇區操作透過執行 6 位元組的指令序列來進行，這個指令序列中包括扇區擦除指令 (0x30) 和扇區位址 (SectorAddress)。塊操作也透過 6 位元組的指令序列來進行，這個指令序列中包括塊擦除指令 (0x50) 和塊位址 (BlockAddress)。扇區或塊的位址在 \overline{WE} 的第 6 個下降沿處鎖存，指令位元組 (0x30 或 0x50) 在 \overline{WE} 的第 6 個上升沿處鎖存。之後，開始內部擦除操作。除了可以使用內部操作狀態檢測方法判斷內部擦除是否結束之外，在內部擦除階段其他的指令都將被忽略。

SST39VF160 還提供一個整片擦除的功能，允許使用者一次性快速擦除整個記憶體（存儲陣列的每個單位都為 1）。整片擦除同樣透過執行 6 位元組的指令序列來進行，6 位元組指令序列中包括片擦除指令 (0x10) 和位元組序列最後的位址 0x5555。圖 4-70 所示為擦除操作的流程圖。

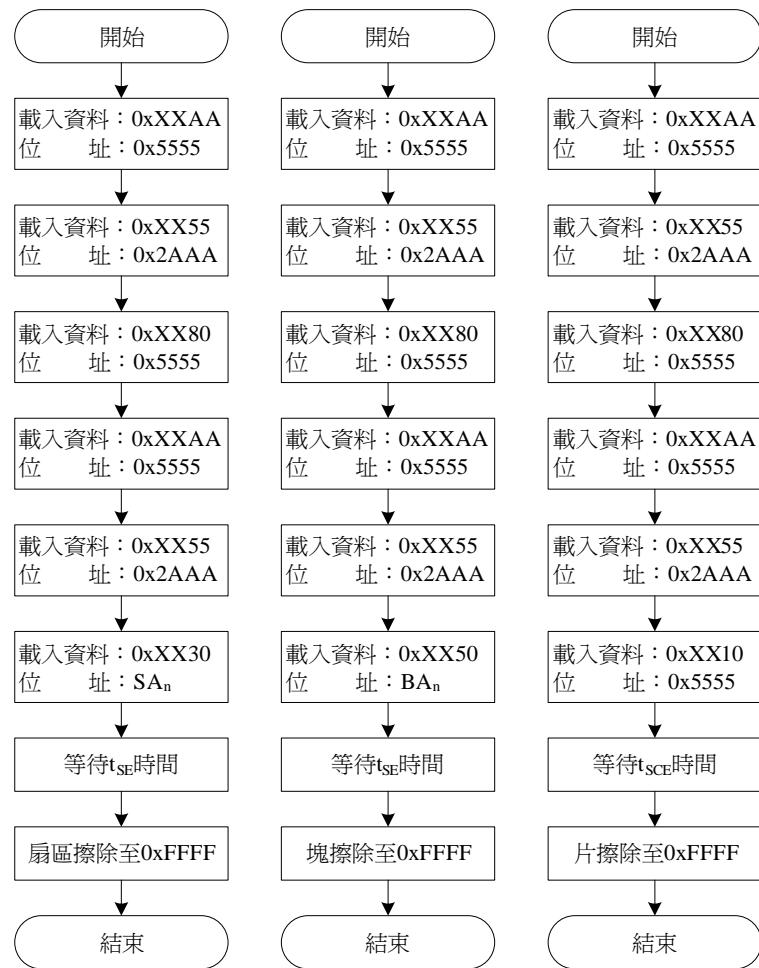


圖 4-70 SST39VF160 的擦除操作流程

圖 4-71 所示為塊擦除時序圖，該器件支援 \overline{CE} 信號控制的塊擦除操作。 BA_n 為塊位址。

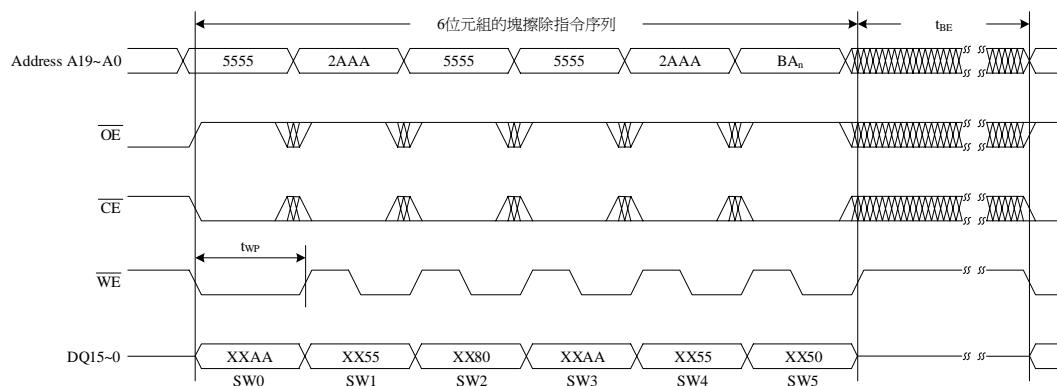


圖 4-71 SST39VF160 的塊擦除時序圖

圖 4-72 所示為扇區擦除時序圖，該器件支援 \overline{CE} 信號控制的扇區擦除操作。 SA_n 為塊位址。

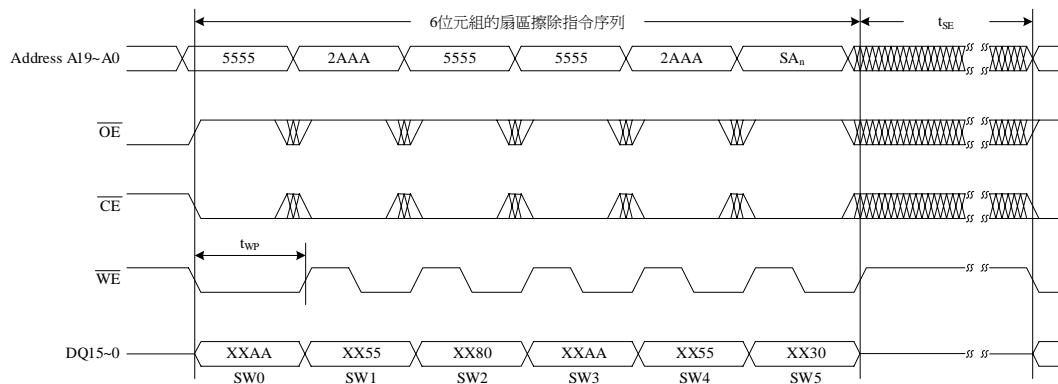


圖 4-72 SST39VF160 的扇區擦除時序圖

圖 4-73 所示為整片擦除時序圖，該器件支援 \overline{CE} 信號控制的整片擦除操作。

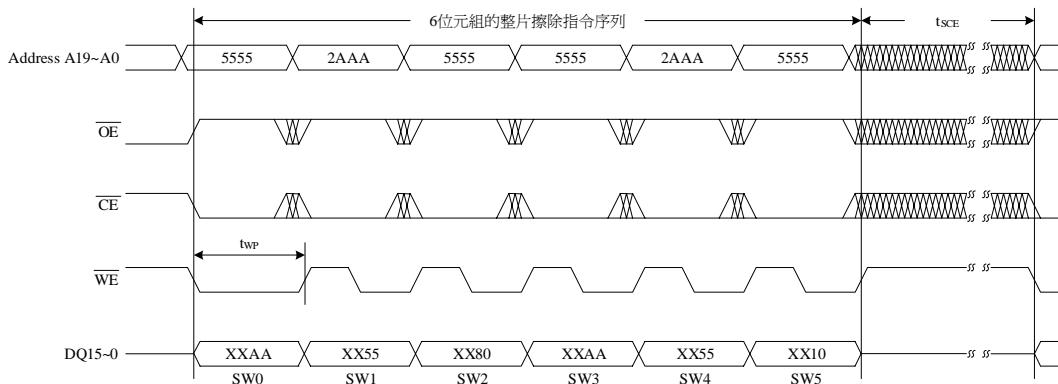


圖 4-73 SST39VF160 的整片擦除時序圖

■ 內部操作狀態檢測

SST39VF160 提供兩種軟體方式來檢測內部操作是否完成。軟體檢測方式涉及兩個狀態位：Data Polling Bit (DQ7) 和 Toggle bit (DQ6)。在送完寫命令序列或擦除命令序列之後的 \overline{WE} 的上升沿，寫入結束檢測功能被使能，同時內部寫入或擦除操作也被初始化，此後可以透過寫入結束檢測操作查詢內部操作是否完成。這裏，只介紹 Toggle bit 方式的檢測操作。

在內部寫入或擦除的過程中，任何對 DQ6 連續的讀操作，晶片都會產生一個不斷翻轉的 1 和 0 輸出。當內部寫入或擦除完成時，DQ6 將停止翻轉。其時序圖和操作流程圖分別如圖 4-74 和圖 4-75 所示。

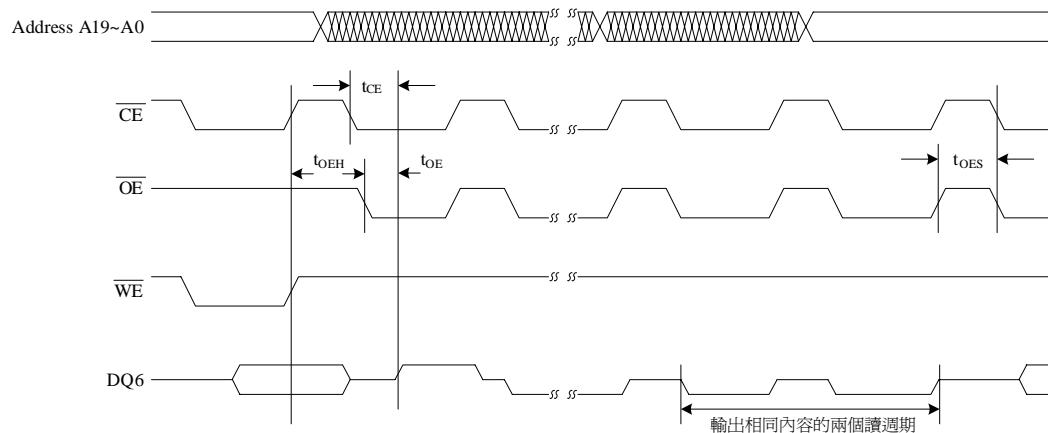


圖 4-74 Toggle bit 時序圖

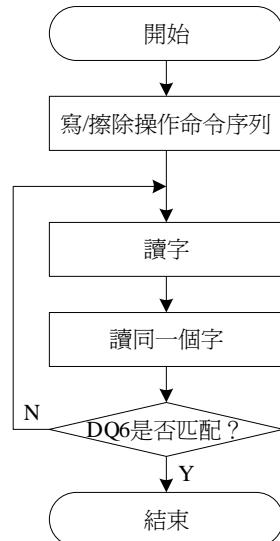


圖 4-75 Toggle Bit 方式檢測操作流程圖

SPCE3200 內部內置 Nor 型 Flash 控制器，可以透過 DRAM 介面實現對外部 Nor 型 Flash 或 ROM 的存取，從而實現 boot 程式或應用程式的存儲。為滿足 ROM 或 Flash 的要求，可以對存取速度進行編程。

4.12.2 特性

SPCE3200 內部的 Nor 型 Flash 介面隸屬於 Memory Interface Unit 部件管轄，具有以下特點：

- 汇流排介面與 DRAM 和 SRAM 複用
- 資料匯流排支援 16bit 寬度
- 可線上對 Flash 進行擦寫

4.12.3 插腳描述

SPCE3200 的 Nor 型 Flash 的介面與 DRAM 公用匯流排。其插腳描述如表 4-130 所示。

表 4-130 Nor 型 Flash 介面插腳對應表

插腳名稱	插腳號	插腳屬性	插腳功能	複用功能
CE	15	O	Flash 片選	ROMCSN
OE	221	O	輸出使能	DRAM_CAS
WE	220	O	寫使能	DRAM_WE
DQ0	218	I/O	資料匯流排	DRAM_D0
DQ1	217	I/O		DRAM_D1
DQ2	216	I/O		DRAM_D2
DQ3	215	I/O		DRAM_D3
DQ4	214	I/O		DRAM_D4
DQ5	212	I/O		DRAM_D5
DQ6	211	I/O		DRAM_D6
DQ7	210	I/O		DRAM_D7
DQ8	209	I/O		DRAM_D16
DQ9	208	I/O		DRAM_D17
DQ10	207	I/O		DRAM_D18
DQ11	205	I/O		DRAM_D19
DQ12	204	I/O		DRAM_D20
DQ13	203	I/O		DRAM_D21
DQ14	202	I/O		DRAM_D22
DQ15	201	I/O		DRAM_D23
A0	256	O	位址匯流排	DRAM_A0
A1	1	O		DRAM_A1
A2	2	O		DRAM_A2
A3	3	O		DRAM_A3
A4	247	O		DRAM_A4
A5	246	O		DRAM_A5
A6	245	O		DRAM_A6
A7	243	O		DRAM_A7

插腳名稱	插腳號	插腳屬性	插腳功能	複用功能
A8	242	O		DRAM_A8
A9	241	O		DRAM_A9
A10	255	O		DRAM_A10
A11	239	O		DRAM_A11
A12	14	O		DRAM_A12
A13	224	O		DRAM_BA0
A14	254	O		DRAM_BA1
A15	251	O		DRAM_DQM0
A16	249	O		DRAM_DQM1
A17	250	O		DRAM_DQM2
A18	248	O		DRAM_DQM3
A19	238	O		DRAM_D8
A20	237	O		DRAM_D9
A21	232	O		DRAM_D10
A22	231	O		DRAM_D11
A23	230	O		DRAM_D12

SPCE3200 外接的 Nor 型 Flash 的虛擬位址被分配在 0x9E000000~0x9FFFFFF 範圍內。在使用時，可以直接存取該位址範圍內的存儲單元。

4.12.4 暫存器描述

SPCE3200 內部的 Nor 型 Flash 控制器共有 2 個控制暫存器，如表 4-131 所示。這兩個暫存器主要控制著對 Nor 型 Flash 的擦除或寫操作。

表 4-131 Nor 型 Flash 控制器相關暫存器列表

暫存器名稱	助記符	位址
NOR 命令控制暫存器	P_NOR_COMMAND_CTRL	0x880700BC
NOR 塊設置暫存器	P_NOR_BANK_SETUP	0x880700C4
DRAM 介面選擇暫存器	P_DRAM_INTERFACE_SEL	0x88200008
DRAM GPIO 設置暫存器	P_DRAM_GPIO_SETUP	0x88200050

暫存器名稱	助記符	位址
DRAM GPIO 輸入資料暫存器	P_DRAM_GPIO_INPUT	0x88200070

如果使用 DRAM 介面（DRAM 介面與 NOR 介面複用）介面複用為 GPIO 功能，可以對下面的暫存器操作。暫存器的各位對應得插腳關係如表 4-132：

表 4-132 DRAM 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
ROMCSN	15	bit[0]	不可以
DRAM_A11	239	bit[1]	不可以
DRAM_A12	14	bit[2]	不可以
DRAM_BA1	254	bit[3]	不可以

■ DRAM 介面選擇暫存器：P_DRAM_INTERFACE_SEL(0x88200008)

SPCE3200 的 NOR 型 Flash 介面插腳與 DRAM 介面插腳複用，而且其中部分插腳可以作為 GPIO 使用，在使用時需要設置 DRAM 介面選擇暫存器以便使其工作在需要的介面模式。

表 4-133 P_DRAM_INTERFACE_SEL(0x88200008)

位	b31~b25	b24	b23~b17	b16	b15~b9	b8	b7~b0
讀/寫	-	R/W	-	R/W	-	R/W	-
預設值	-	0	-	0	-	0	-
名稱	-	SW_DRAM_BA1	-	SW_DRAM_A12	-	SW_DRAM_A11	-

SW_DRAM_BA1

b24

DRAM_BA1 插腳功能選擇位：

0：作為 GPIO 使用

1：作為 DRAM BA1 塊選擇插腳使用

SW_DRAM_A12

b16

DRAM_A12 插腳功能選擇位：

0：作為 GPIO 使用

1：作為 DRAM A12 位址選擇插腳使用

 SW_DRAM_A11 b8 DRAM_A11 插腳功能選擇位：

0：作為 GPIO 使用

1：作為 DRAM A11 位址選擇插腳使用

■ DRAM GPIO 設置暫存器：P_DRAM_GPIO_SETUP(0x88200050)

DRAM GPIO 設置暫存器設置 DRAM 介面複用為 GPIO 時的插腳輸出使能、輸出資料、上拉電阻輸入、下拉電阻輸入。

表 4-134 P_DRAM_GPIO_SETUP(0x88200050)

位	b27~b24	b23~b20	b19~b16	b15~b12	b11~b8	b7~b4	b3~b0
讀/寫	R/W	-	R/W	-	R/W	-	R/W
預設值	0	-	0	-	0	-	0
名稱	DRAM_PD	-	DRAM_PU	-	DRAM_OE	-	DRAM_O

 DRAM_PD b27~b24 DRAM 介面作為 GPIO 使用時的下拉電阻輸入使能位：

0: 禁止下拉電阻輸入

1: 使用下拉電阻輸入

 DRAM_PU b19~b16 DRAM 介面作為 GPIO 使用時的上拉電阻輸入使能位：

0 : 使能上拉電阻輸入

1 : 禁止上拉電阻輸入

 DRAM_OE b11~b8 DRAM 介面作為 GPIO 使用時的輸出使能位：

0 : 不輸出使能

1 : 輸出使能

 DRAM_O b3~b0 DRAM 介面作為 GPIO 使用時的輸出資料

■ DRAM GPIO 輸入資料暫存器：P_DRAM_GPIO_INPUT(0x88200070)

DRAM GPIO 輸入暫存器保存 DRAM 介面複用為 GPIO 時的外部輸入資料。

表 4-135 P_DRAM_GPIO_INPUT(0x88200070)

位	b31~b20	b19~b16	b15~b0
讀/寫	-	R/W	-
預設值	-	0	-
名稱	-	DRAM_INPUT	-

DRAM_INPUT	b19~b16	DRAM 介面作為 GPIO 使用時的輸入資料
------------	---------	-------------------------

■ NOR 命令控制暫存器：P_NOR_COMMAND_CTRL(0x880700BC)

NOR 命令控制暫存器，可以打開或關閉對 Nor 型 Flash 的命令暫存器的操作。

表 4-136 P_NOR_COMMAND_CTRL(0x880700BC)

位	b31~b1	b0
讀/寫	-	W
預設值	-	0
名稱	-	COMMAND_EN

COMMAND_EN	b0	NOR 型 FLASH 命令寫使能位：
0：普通模式，該模式下向 NOR 型 FLASH 寫入的資料均為普通資料		
1：命令模式，該模式下向 NOR 型 FLASH 寫入的資料均為命令資料		

■ NOR 塊設置暫存器：P_NOR_BANK_SETUP(0x880700C4)

SPCE3200 內的 NOR 型 Flash 控制器在執行擦除或者編程操作時，需要以 2MB 大小的塊來進行。在進行寫或擦除之前，需將塊號寫入該暫存器。

表 4-137 P_NOR_BANK_SETUP(0x880700C4)

位	b31~b8	b7~b0
讀/寫	-	W
預設值	-	0

位	b31~b8	b7~b0
名稱	-	BANK_ADDR

BANK_ADDR	b7~b0	用戶需要以 2MB 為單位，計算當前寫入位址或擦除位址屬於哪個塊，並將塊號寫入該暫存器
-----------	-------	---

4.12.5 基本操作

■ 讀操作

對 NOR 型 Flash 的讀操作比較簡單，直接存取 NOR 型 Flash 所在的位址空間的位址即可。例如：欲讀取 0x9E000000 位址單元的資料，則可使用如下代碼進行：

```
int *addr;
int temp;
addr = 0x9e000000;
temp = *addr;
```

■ 擦除操作

對 NOR 型 Flash 的擦除操作需要遵循晶片本身規定的命令序列來進行。同時，需要設置 P_NOR_COMMAND_CTRL 暫存器和 P_NOR_BANK_SETUP 暫存器，以便控制器可以準確對擦除區域操作。

以 NOR 型 Flash 的扇區擦除操作為例，說明擦除操作的典型步驟。圖 4-76 所示的使用 SPCE3200 開發板對 NOR 型 Flash 進行扇區擦除的編程步驟。

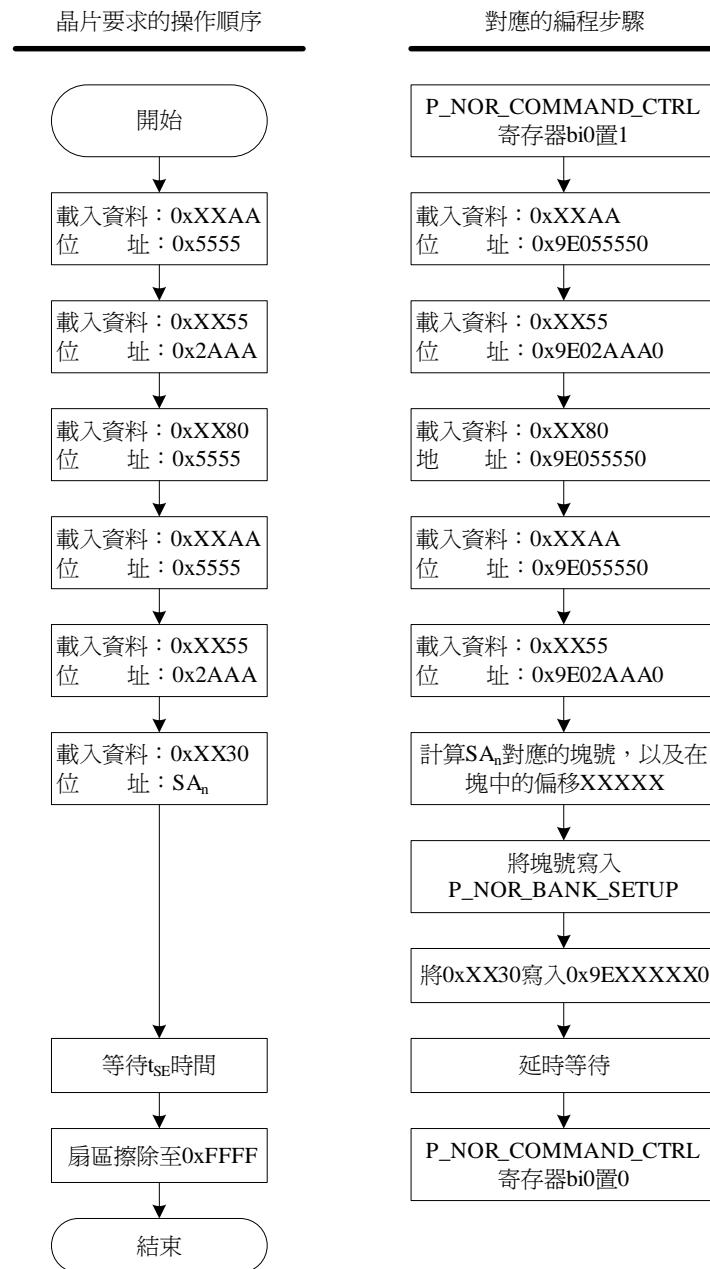


圖 4-76 扇區擦除操作步驟

參考代碼如下：

```

#define P_NOR_COMMAND_CTRL      (volatile unsigned int *)0x880700bc
#define P_NOR_BANK_SETUP        (volatile unsigned int *)0x880700c4

#define C_Flash_BaseAddr        0x9e000000
#define C_CMD1_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD2_Addr             (volatile unsigned int *)0x9e02aaa0
#define C_CMD3_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD4_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD5_Addr             (volatile unsigned int *)0x9e02aaa0
#define C_CMD6_Addr             (volatile unsigned int *)0x9e055550

```

```
#define C_CMD_Data1          0xaa           // 晶片要求的命令
#define C_CMD_Data2          0x55           // 晶片要求的命令
#define C_CMD_Data3          0x80           // 晶片要求的命令
#define C_CMD_Data4          0xaa           // 晶片要求的命令
#define C_CMD_Data5          0x55           // 晶片要求的命令
#define C_SECTOR_ERASE        0x30           // 擦除命令
#define C_SECTOR_ERASE_TIME   500000         // 擦除扇區延時時間
#define C_SECTOR_SIZE         2048          // 2048 字/扇區，1 word = 16 bits

extern void NorFlash_SectorErase(unsigned int addr);
//=====
// 語法格式： void NorFlash_SectorErase(unsigned int addr)
// 實現功能： 扇區擦除(4KB/Sector)
// 參 數：    addr      - 擦除位址(低 12bit 將被忽略)
// 返回值： 無
//=====
void NorFlash_SectorErase(unsigned int addr)
{
    addr -= C_Flash_BaseAddr;           // 去掉基底位址 0x9e000000
    addr = addr / 2;                   // 位址轉換成位元組位址
    addr = addr & (~(C_SECTOR_SIZE - 1)); // 將低 12bit 置為 0
    addr = NorFlash_SetAddress(addr);   // 以 2MB 為單位，計算屬於哪個塊，
                                         // 並將該塊寫入 NOR 塊設置暫存器
    addr = C_Flash_BaseAddr + (addr << 4); // 根據硬體連接，轉換操作位址
    *P_NOR_COMMAND_CTRL = 1;           // 作為命令寫入資料

    *C_CMD1_Addr = C_CMD_Data1;        // 根據晶片要求，寫入命令 1，0xAA
    *C_CMD2_Addr = C_CMD_Data2;        // 根據晶片要求，寫入命令 2，0x55
    *C_CMD3_Addr = C_CMD_Data3;        // 根據晶片要求，寫入命令 3，0x80
    *C_CMD4_Addr = C_CMD_Data4;        // 根據晶片要求，寫入命令 4，0xaa
    *C_CMD5_Addr = C_CMD_Data5;        // 根據晶片要求，寫入命令 5，0x55
    *(unsigned int *)addr = C_SECTOR_ERASE; // 擦除扇區位址

    delay(C_SECTOR_ERASE_TIME);       // 延時晶片規定時間
    *P_NOR_COMMAND_CTRL = 0;           // 切換回普通模式
}

//=====
// 語法格式： unsigned int NorFlash_SetAddress(unsigned int addr)
// 實現功能： 為控制器設置操作位址(內部調用)
// 參 數：    addr      - 原始操作位址
// 返回值： 控制器操作位址
//=====
unsigned int NorFlash_SetAddress(unsigned int addr)
{
    unsigned i = 0;
    while(addr > 0xfffff)           // 如果位址大於 2MB
    {
        addr -= 0x100000;            // 減去 2MB 位址
        i++;                         // 塊號加 1
    }
    *P_NOR_BANK_SETUP = i;           // 設置塊號
    return(addr);
}
```

```
//=====
// 語法格式： void delay(unsigned int clk)
// 實現功能： 延時一段時間(內部調用)
// 參 數： clk - 延時時間控制
// 返回值： 無
//=====
void delay(unsigned int clk)
{
    unsigned int i;
    for(i = 0; i < clk; i++);
}
```

■ 編程操作（寫操作）

對 NOR 型 FLASH 的寫操作與擦除操作類似，按照 NOR 型 Flash 晶片規定的命令順序操作即可，這裏不再細述操作流程。

參考代碼如下：

```
#define P_NOR_COMMAND_CTRL      (volatile unsigned int *)0x880700bc
#define P_NOR_BANK_SETUP         (volatile unsigned int *)0x880700c4

#define C_Flash_BaseAddr        0x9e000000
#define C_CMD1_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD2_Addr             (volatile unsigned int *)0x9e02aaa0
#define C_CMD3_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD4_Addr             (volatile unsigned int *)0x9e055550
#define C_CMD5_Addr             (volatile unsigned int *)0x9e02aaa0
#define C_CMD6_Addr             (volatile unsigned int *)0x9e055550

#define C_CMD_Data1              0xaa          // 晶片要求的命令
#define C_CMD_Data2              0x55          // 晶片要求的命令
#define C_CMD_Data3              0x80          // 晶片要求的命令
#define C_CMD_Data4              0xaa          // 晶片要求的命令
#define C_CMD_Data5              0x55          // 晶片要求的命令
#define C_WORD_WRITE             0xa0          // 寫入命令

#define C_WRITE_DELAY_TIME      500           // 寫入延時時間
#define C_SECTOR_SIZE            2048          // 2048 字/扇區，1 word = 16 bits

//=====
// 語法格式： void NorFlash_Write32(unsigned int addr, unsigned int data)
// 實現功能： 寫入 32bit 資料
// 參 數： addr - 操作位址
//          data - 寫入資料
// 返回值： 無
//=====
void NorFlash_Write32(unsigned int addr, unsigned int data)
{
    unsigned int uiTemp;

    if(addr >= C_Flash_BaseAddr)           // 是否大於基底位址
    {
```

```

        addr -= C_Flash_BaseAddr;           // 減掉基底位址
        uiTemp = addr;                   // 臨時保存寫入位址
        //write low 16 bits
        addr >>= 1;                     // 轉換成晶片位元組位址，字的低位址
        addr = NorFlash_SetAddress(addr); // 設置 NOR 塊設置暫存器
        addr = C_Flash_BaseAddr + (addr << 4); // 根據硬體連線轉換為實際操作位址
        NorFlash_WordWrite(addr, data & 0x0000ffff); // 寫入資料低 16 位
        //write high 16 bits
        addr = uiTemp;                  // 恢復位址
        addr >>= 1;                     // 轉換成晶片位元組位址
        addr++;                         // 字的高位址
        addr = NorFlash_SetAddress(addr); // 設置 NOR 塊設置暫存器
        addr = C_Flash_BaseAddr + (addr << 4); // 根據硬體連線轉換為實際操作位址
        NorFlash_WordWrite(addr, (data >> 16) & 0x0000ffff); // 寫入資料高 16 位
    }
}

//=====
// 語法格式： void NorFlash_WordWrite(unsigned int addr, unsigned data)
// 實現功能： Nor Flash 寫入操作流程(內部調用)
// 參 數：   addr      - 操作位址
//          data     - 寫入資料
// 返回值： 無
//=====

void NorFlash_WordWrite(unsigned int addr, unsigned data)
{
    *P_NOR_COMMAND_CTRL = 1;           // 作為命令寫入資料
    data &= 0x0000ffff;
    *C_CMD1_Addr = C_CMD_Data1;       // 根據晶片要求，寫入命令 1，0xAA
    *C_CMD2_Addr = C_CMD_Data2;       // 根據晶片要求，寫入命令 2，0x55
    *C_CMD3_Addr = C_WORD_WRITE;      // 根據晶片要求，寫入命令 3，0xA0
    *(unsigned int *)addr = data;      // 寫入資料
    delay(C_WRITE_DELAY_TIME);        // 延時 20us
    *P_NOR_COMMAND_CTRL = 0;          // 切換回普通模式
}

//=====
// 語法格式： unsigned int NorFlash_SetAddress(unsigned int addr)
// 實現功能： 為控制器設置操作位址(內部調用)
// 參 數：   addr      - 原始操作位址
// 返回值： 控制器操作位址
//=====

unsigned int NorFlash_SetAddress(unsigned int addr)
{
    unsigned i = 0;
    while(addr > 0xfffff)           // 如果位址大於 2MB
    {
        addr -= 0x100000;            // 減去 2MB 位址
        i++;                         // 塊號加 1
    }
    *P_NOR_BANK_SETUP = i;           // 設置塊號
    return(addr);
}

//=====

```

```

// 語法格式： void delay(unsigned int clk)
// 實現功能： 延時一段時間(內部調用)
// 參數： clk - 延時時間控制
// 返回值： 無
//=====
void delay(unsigned int clk)
{
    unsigned int i;
    for(i = 0; i < clk; i++);
}

```

4.13 Nand 型 Flash 控制器

4.13.1 概述

對於許多消費類音視頻產品而言，Nand 型 Flash 存儲設備是一種比硬碟驅動器更好的存儲方案，這在不超過 4GB 的低容量應用中表現得猶為明顯。隨著人們持續追求功耗更低、重量更輕和性能更佳的產品，NAND 正被證明極具吸引力。

Nand 型 Flash 結構能提供極高的單元密度，可以達到高存儲密度，並且寫入和擦除的速度也很快。NAND 器件使用複雜的 I/O 埠來串列地存取資料，對於 16 位的器件，並行快閃記憶體大約需要 40 多個 I/O 插腳，相對而言，NAND 器件僅需 24 個插腳。

Nand 型 Flash 使用 8 個插腳用來傳送控制、位址和資料資訊，同時配合 ALE、CLE、CEN、WEN、REN 等控制管腳，實現 Flash 的讀寫管理。

典型的 Nand 型 Flash 器件的插腳分佈如圖 4-77 所示。

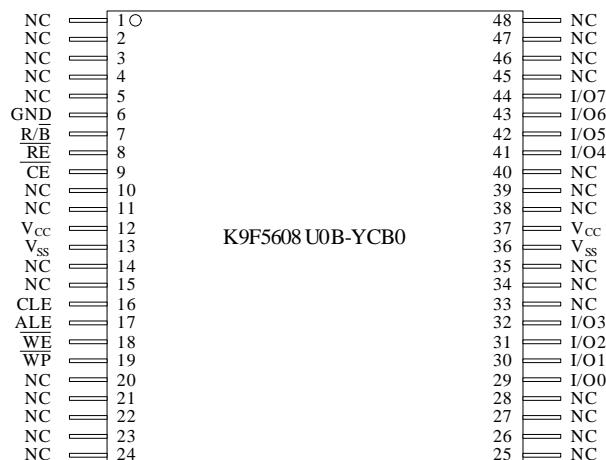


圖 4-77 K9F5608U0b 插腳分佈圖

插腳描述如表 4-138 所示。

表 4-138 Nand 型 Flash 插腳功能描述

插腳符號	插腳名稱	功能描述
I/O7~I/O0	資料輸入/輸出	I/O 埠是用來輸入指令、位址和資料，並在讀週期時輸出資料的。當晶片未被選中或輸出禁能時，I/O 埠呈高阻態。
CLE	命令鎖存控制	CLE 插腳上的輸入電平用來控制打開/關閉指令送入指令暫存器的通路。當 CLE 為高時，I/O 埠在 \overline{WE} 信號的上升沿將指令鎖存至指令暫存器。
ALE	位址鎖存控制	ALE 插腳上的輸入電平用來控制打開/關閉位址送入位址暫存器的通路。當 ALE 為高時，I/O 埠在 \overline{WE} 信號的上升沿將位址鎖存至位址暫存器。
\overline{CE}	晶片使能	低電平使能的片選控制線。 當晶片處於忙狀態時，該插腳即使變高也將被忽略。
\overline{RE}	讀使能	該插腳為串列資料輸出控制線。當它為低電平時，內部資料將輸出至 I/O 埠。輸出資料在該插腳下降後 t_{REA} 時間內有效，同時，內部列位址計數器將加 1。
\overline{WE}	寫使能	該插腳對 I/O 埠的寫入進行控制。指令、位址和資料都會在該插腳的上升沿被鎖存。
\overline{WP}	防寫	該插腳提供在電源波動情況下，對器件不可預料的寫入或擦除的保護。當該插腳為低電平時，內部高電壓發生器被重設。
R/\overline{B}	讀/忙輸出	該插腳的輸出說明瞭器件目前的操作狀態。當它為低電平時，表明某個寫入、擦除或隨機讀操作正在進行，當這個操作完成，該插腳才會重新回到高電平狀態。它是一個漏極開路輸出，而且在晶片未選中或輸出未使能時，不會進入高阻態。
V_{CC}	電源	器件的供電電源。
V_{SS}	地	
NC	不連接的插腳	
GND	GND 輸入使能 備用區	在進行連續行讀操作時，當要讀取包括備用區在內的資料時，要把 GND 接 V_{SS} 或拉低；如果不要讀取包括備用區在內的資料時，則將該插腳接 V_{CC} 或置高。

Nand 型 Flash 器件的存儲空間的組織結構由若干個塊（Block）組成，每個塊又由 32 或 64 個頁（Page）組成，每個頁則包含了 528 或 2112 個位元組（列）。其中，具有 528 位元組的頁容量的器件，每個塊由 32 個頁組成；具有 2112 位元組的頁容量的器件，每個塊由 64 個頁組成。

在 528 位元組或 2112 位元組的頁中，分別包含了 16 位元組或 64 位元組的備用區域，該區域與其他的 512 位元組或 2048 位元組沒有區別，但是，一般用備用區域來做壞塊標識或損耗均衡等工作。

Nand 型 Flash 器件的存儲空間的組織結構如圖 4-78 所示。

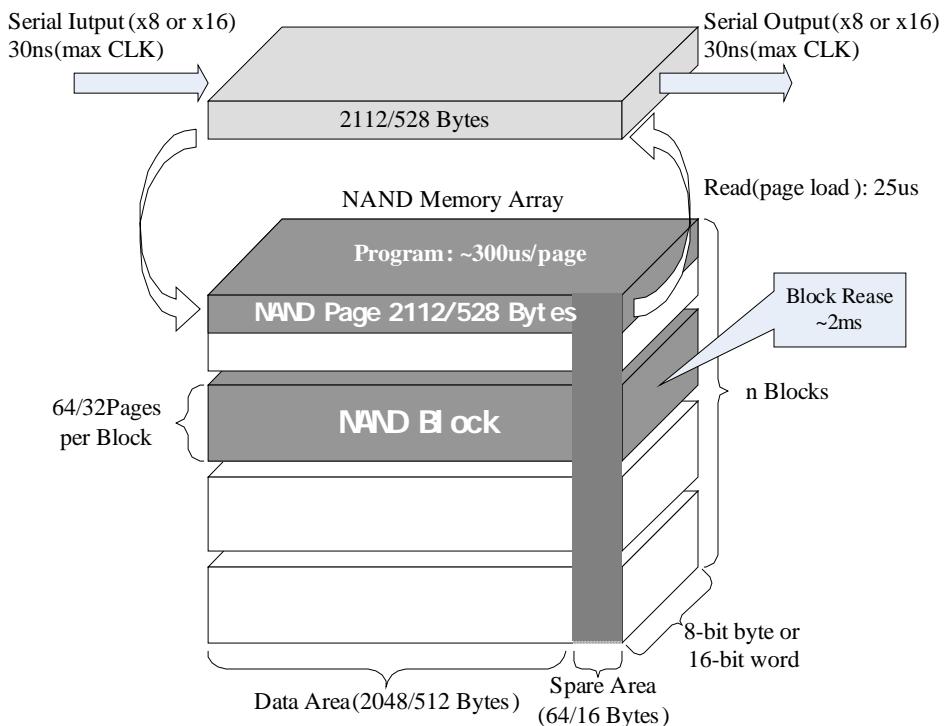


圖 4-78 Nand 型 Flash 存儲空間的組織結構

在對 Nand 型 Flash 進行讀或寫操作時，都以頁為單位進行；擦除操作則以塊為單位進行。

對 Nand 型 Flash 器件內部存儲單元的定址位址透過 8 位埠傳送，這樣，可以有效的節省插腳的數量，並能保持不同密度器件插腳的一致性，允許系統可以在電路不做改動的情況下升級為高容量的記憶體件。

Nand 型 Flash 器件透過 CLE 和 ALE 信號線實現 I/O 埠上指令和位址的複用。指令、位址和資料都透過拉低 \overline{WE} 和 \overline{CE} 從 I/O 埠寫入器件中。CLE 和 ALE 是否為高決定了當前寫入的資料是命令還是位址，或者是資料。一些命令只需要一個匯流排週期完成，例如，重設命令、讀命令和讀狀態命令等；另外一些命令，例如頁寫入和塊擦除命令等，則需要 2 個週期，即在 CLE 有效的情況下啟動兩次 \overline{WE} 信號來寫入命令。表 4-139 列舉出了以 K9F5608U08 為例的 Nand 型 Flash 具備的命令。

表 4-139 K9F5608U0B 具備的指令和功能

功能	第一個週期	第二個週期	器件忙時是否接收
讀方式 1	0x00/0x01	-	否
讀方式 2	0x50	-	否
讀晶片 ID 號	0x90	-	否
重設	0xff	-	是

功能	第一個週期	第二個週期	器件忙時是否接收
頁寫入	0x80	0x10	否
回拷貝	0x00	0x8a	否
塊擦除	0x60	0xd0	否
讀當前狀態	0x70	-	是

SPCE3200 內嵌了 Nand 型 Flash 控制器，以便可以方便的連接 Nand 型 Flash 存儲單元，提供大容量存儲解決方案。該控制器提供標準的帶有 ECC 改錯碼計算電路的 8 位元 Nand 型 Flash 匯流排介面，同時，還支援連接智慧媒體卡。

4.13.2 特性

SPCE3200 內部的 Nand 型 Flash 控制器具有以下特性：

- 完全可編程的 CLE 和 ALE 時序，支援多種 Nand 型 Flash
- 提供查詢/中斷/DMA 的存取方式
- 支援 528 位元組~2112 位元組的頁容量
- 可編程的讀/寫時鐘週期
- 可編程的 ALE 時鐘週期

4.13.3 插腳描述

SPCE3200 的 Nand 型 Flash 的介面插腳與 SPI 和 SD 卡介面複用，在使用時需要設置相關的暫存器以便使其工作在 Nand 型 Flash 介面模式。Nand 型 Flash 介面的插腳描述如表 4-140 所示。

表 4-140 Nand 型 Flash 介面插腳對應表

插腳名稱	插腳號	插腳屬性	插腳功能
NF_ALE	127	O	位址鎖存使能，高電平有效
NF_WPN	126	O	防寫使能，低電平有效
NF_CLE	125	O	命令鎖存使能，高電平有效
NF_REN	124	O	讀資料使能，低電平有效
NF_WEN	123	O	寫資料使能，低電平有效
NF_CEN	119	O	晶片片選，低電平有效
NF_RDY	118	I	Nand 型 Flash 就緒輸入，高電平有效
NF_D0	117	I/O	資料匯流排
NF_D1	116	I/O	

插腳名稱	插腳號	插腳屬性	插腳功能
NF_D2	115	I/O	
NF_D3	112	I/O	
NF_D4	111	I/O	
NF_D5	110	I/O	
NF_D6	109	I/O	
NF_D7	108	I/O	

4.13.4 結構

Nand 型 Flash 控制器的結構如圖 4-79所示。

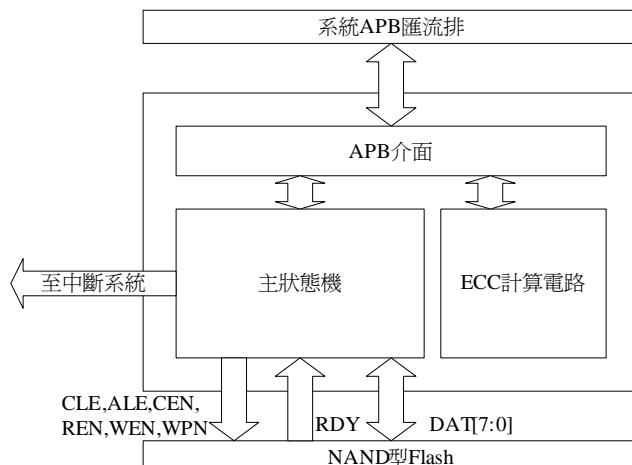


圖 4-79 Nand 型 Flash 控制器結構

4.13.5 暫存器描述

Nand 型 Flash 控制器共有 19 個暫存器，如表 4-141所示。透過對這 19 個暫存器的操作，即可透過 Nand 型 Flash 介面擴展 Nand 型 Flash 存儲設備。下麵將對這些暫存器一一進行說明。

表 4-141 Nand 型 Flash 控制器相關暫存器列表

暫存器名稱	助記符	位址
NAND 介面選擇暫存器	P_NAND_INTERFACE_SEL	0x882000A4
NAND GPIO 設置暫存器	P_NAND_GPIO_SETUP	0x8820002C
NAND GPIO 上下拉暫存器	P_NAND_GPIO_PULL	0x88200030

暫存器名稱	助記符	位址
NAND GPIO 輸入資料暫存器	P_NAND_GPIO_INPUT	0x8820006C
NAND GPIO 外部中斷暫存器	P_NAND_GPIO_INT	0x88200088
NAND 時鐘配置暫存器	P_NAND_CLK_CONF	0x882100A8
NAND 中斷控制暫存器	P_NAND_INT_CTRL	0x88190014
NAND 控制暫存器	P_NAND_MODE_CTRL	0x88190000
NAND 命令暫存器	P_NAND_CLE_COMMAND	0x88190004
NAND 位址暫存器	P_NAND_ALE_ADDR	0x88190008
NAND 發送資料暫存器	P_NAND_TX_DATA	0x8819000C
NAND 接收資料暫存器	P_NAND_RX_DATA	0x88190010
NAND 中斷狀態暫存器	P_NAND_INT_STATUS	0x88190018
NAND 真實行檢查碼暫存器	P_NAND_ECC_TRUELP	0x8819001C
NAND 真實列檢查碼暫存器	P_NAND_ECC_TRUECP	0x88190020
NAND 對比行檢查碼暫存器	P_NAND_ECC_CMPLP	0x88190024
NAND 對比列檢查碼暫存器	P_NAND_ECC_CMPCP	0x88190028
NAND 檢查狀態暫存器	P_NAND_ECC_STATUS	0x8819002C
NAND 檢查控制暫存器	P_NAND_ECC_CTRL	0x88190030

如果使用 Nand 型 Flash 介面複用為 GPIO 功能，可以對下面的暫存器操作。暫存器的各位對應插腳關係如表 4-142：

表 4-142 Nand 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
NF_ALE	127	bit[0]	可以
NF_WPN	126	bit[1]	可以
NF_CLE	125	bit[2]	可以
NF_REN	124	bit[3]	可以
NF_WEN	123	bit[4]	不可以
NF_CEN	119	bit[5]	不可以
NF_RDY	118	bit[6]	不可以

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
NF_D0	117	bit[7]	不可以
NF_D1	116	bit[8]	不可以
NF_D2	115	bit[9]	不可以
NF_D3	112	bit[10]	不可以
NF_D4	111	bit[11]	不可以
NF_D5	110	bit[12]	不可以
NF_D6	109	bit[13]	不可以
NF_D7	108	bit[14]	不可以
SPI_CSN	128	bit[15]	不可以

■ NAND GPIO 設置暫存器 : P_NAND_GPIO_SETUP(0x8820002C)

NAND GPIO 設置暫存器設置 NAND 介面複用為 GPIO 時的插腳輸出使能和輸出資料。

表 4-143 P_NAND_GPIO_SETUP(0x8820002C)

位	b31~b16	b15~b0
讀/寫	R/W	R/W
預設值	0	0
名稱	NFLASH_GPIO_OE	NFLASH_GPIO_O

NFLASH_GPIO_OE b31-b16 NFLASH 介面作為 GPIO 使用時的輸出使能位：

0 : 禁止輸出

1 : 使能輸出

NFLASH_GPIO_O b15-b0 NFLASH 介面作為 GPIO 使用時的輸出資料

■ NAND GPIO 上/下拉暫存器 : P_NAND_GPIO_PULL(0x88200030)

NAND GPIO 上/下拉暫存器設置 NAND 介面複用為 GPIO 時的上拉電阻輸入使能和下拉電阻輸入使能功能。

表 4-144 P_GPIO_NFLASH_PULL (0x88200030)

位	b31~b16	b15~b0
讀/寫	R/W	R/W
預設值	1	1
名稱	NFLASH_GPIO_PD	NFLASH_GPIO_PU

NFLASH_GPIO_PD b31~b16 NFlash 介面作為 GPIO 使用時的下拉電阻輸入使能位：

0：禁止下拉電阻輸入

1：使能下拉電阻輸入

NFLASH_GPIO_PU b15~b0 NFlash 介面作為 GPIO 使用時的上拉電阻輸入使能位：

0：使能上拉電阻輸入

1：禁止上拉電阻輸入

■ NAND GPIO 輸入資料暫存器：P_NAND_GPIO_INPUT(0x8820006C)

NAND GPIO 輸入暫存器保存 NAND 介面複用為 GPIO 時的外部輸入資料。

表 4-145 P_NAND_GPIO_INPUT(0x8820006C)

位	b15~b0
讀/寫	R
預設值	0
名稱	NFLASH_INPUT

NFLASH_INPUT b15~b0 NFlash 介面作為 GPIO 使用時的輸入資料

■ NAND GPIO 外部中斷暫存器：P_NAND_GPIO_INT(0x88200088)

NAND GPIO 外部中斷暫存器可進行中斷使能、中斷觸發沿設置、中斷旗標清除等操作。

表 4-146 P_NAND_GPIO_INT(0x88200088)

位	b27~b24	b23~b20	b19~b16	b15~b12	b11~b8	b7~b4	b3~b0
讀/寫	R/W	-	R/W	-	R/W	-	R/W
預設值	0	-	0	-	0	-	0
名稱	NFLASH_FI	-	NFLASH_RI	-	NFLASH_FIEN	-	NFLASH RIEN

NFLASH_FI

b27-b24

NFlash GPIO 下降沿中斷旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

NFLASH _RI

b19-b16

NFlash GPIO 上升沿中斷旗標位：

讀 0：沒有發生上升沿中斷

讀 1：發生上升沿中斷

寫 0：無意義

寫 1：清除中斷旗標

NFLASH _FIEN

b11-b8

NFlash GPIO 下降沿中斷使能位：

0：禁止下降沿中斷

1：使能下降沿中斷

NFLASH _RIEN

b3-b0

NFlash GPIO 上升沿中斷使能位：

0：禁止上升沿中斷

1：使能上升沿中斷

作為 NAND Flash 控制器需要設置以下暫存器：

■ NAND 介面選擇暫存器：**P_NAND_INTERFACE_SEL(0x882000A4)**

SPCE3200 的 Nand 型 Flash 的介面插腳與 SPI 和 SD 卡介面複用，在使用時需要設置 NAND 介面選擇暫存器以便使其工作在 Nand 型 Flash 介面模式。

表 4-147 P_NAND_INTERFACE_SEL(0x882000A4)

位	b31~b1	b0
讀/寫	-	R/W
預設值	-	0
名稱	-	NFLASH_EN

NFLASH_EN	b0	Nand 型 Flash 介面使能位： 0：禁止為 NAND Flash 介面 1：使能為 NAND Flash 介面
-----------	----	---

■ NAND 時鐘配置暫存器：P_NAND_CLK_CONF(0x882100A8)

NAND 時鐘配置暫存器用於使能或禁止 Nand 型 Flash 控制器模組時鐘，或軟重設 Nand 型 Flash 控制器模組。

表 4-148 P_NAND_CLK_CONF(0x882100A8)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	FLASH_RST	FLASH_STOP

FLASH_RST	b1	NAND FLASH 模組時鐘重設位： 0：NAND Flash 模組時鐘重設 1：NAND Flash 模組時鐘不重設
FLASH_STOP	b0	NAND FLASH 模組時鐘使能位： 0：NAND Flash 模組時鐘停止 1：NAND Flash 模組時鐘使能

■ NAND 中斷控制暫存器：P_NAND_INT_CTRL(0x88190014)

NAND 中斷控制暫存器用於使能或禁止 Nand 控制器的中斷。Nand 型 Flash 控制器使用 IRQ41 中斷。

表 4-149 P_NAND_INT_CTRL(0x88190014)

位	b31~b6	b5	b4	b3	b2	b1	b0
讀/寫		R/W	R/W	R/W	R/W	R/W	R/W
預設值		0	0	0	0	0	0
名稱		INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0

INTEN5	b5	命令丟失 (CMDLOSS) 中斷使能位： 0：禁止 1：使能	當 Flash 控制器不忙的時候向該暫存器寫入資料時，Flash 控制器將啓動 CLE 信號輸出，並將寫入 NAND 命令暫存器的命令送至八位元資料匯流排，同時會啓動 WEN 信號的輸出，以便將命令寫入 Nand 型 Flash。如果當控制器忙的時候向該暫存器寫入資料，將引起命令丟失 (CMDLOSS) 旗標位置 1，同時觸發命令丟失中斷。
INTEN4	b4	讀資料錯誤 (RDMISS) 中斷使能位： 0：禁止 1：使能	當讀緩衝區滿時，超過一定時間未讀取資料，則發生讀資料錯誤中斷
INTEN3	b3	寫資料丟失 (WRLOSS) 中斷使能位： 0：禁止 1：使能	當寫緩衝區空時，超過一定時間未寫入資料，則發生寫資料丟失中斷
INTEN2	b2	RDY 信號中斷使能位： 0：禁止 1：使能	當 RDY 信號有上升沿輸入時觸發該中斷
INTEN1	b1	讀緩衝區滿中斷使能位： 0：禁止 1：使能	當讀緩衝區滿時，產生該中斷

INTEN0	b0	寫緩衝區空中斷使能位： 0：禁止 1：使能 當寫緩衝區空時，產生該中斷
--------	----	--

■ NAND 控制暫存器：P_NAND_MODE_CTRL(0x88190000)

NAND 控制暫存器用於對 Nand 型 Flash 記憶體存取時序的設定。

表 4-150 P_NAND_MODE_CTRL(0x88190000)

位	b31-b28	b27-b16	b15-b14	b13-b12	b11-b10	b9-b8
讀/寫	-	R/W	R/W	R/W	R/W	R/W
預設值	-	0	0	0	0	0
名稱	-	TBYTES	RDHIGH	RDLOW	WRHIGH	WRLOW
位	b7-b6	b5-b4	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	1	1	0
名稱	ALECYC	RDTYPE	WRCMD	FLWPN	FLCEN	FLEN

TBYTES	b27-b16	本次傳輸的資料量： 如果本次需要傳輸的資料為 N，在設置的時候寫 N - 1
--------	---------	---

RDHIGH	b15-b14	REN（讀操作使能）高電平持續時鐘週期數量： 00 : 1 個時鐘週期 01 : 2 個時鐘週期 10 : 3 個時鐘週期 11 : 4 個時鐘週期
--------	---------	--

這裏的時鐘週期來自於 27MHz 晶振

RDLOW	b13-b12	REN (讀操作使能) 低電平持續時鐘週期數量：
-------	---------	--------------------------

00 : 1 個時鐘週期

01 : 2 個時鐘週期

10 : 3 個時鐘週期

11 : 4 個時鐘週期

這裏的時鐘週期來自於 27MHz 晶振

WRHIGH	b11-b10	WEN (寫操作使能) 高電平持續時鐘週期數量：
--------	---------	--------------------------

00 : 1 個時鐘週期

01 : 2 個時鐘週期

10 : 3 個時鐘週期

11 : 4 個時鐘週期

這裏的時鐘週期來自於 27MHz 晶振

WRLOW	b9-b8	WEN (寫操作使能) 低電平持續時鐘週期數量：
-------	-------	--------------------------

00 : 1 個時鐘週期

01 : 2 個時鐘週期

10 : 3 個時鐘週期

11 : 4 個時鐘週期

這裏的時鐘週期來自於 27MHz 晶振

ALECYC	b7-b6	當 P_NAND_ALE_ADDR 單元被寫入時觸發多少個寫信號：
--------	-------	-----------------------------------

00 : 1 個時鐘週期

01 : 2 個時鐘週期

10 : 3 個時鐘週期

11 : 4 個時鐘週期

該暫存器主要用來設置位址週期數量。即，在 ALE 有效期間位址被分為多少個位元組被寫入

RDTYPE	b5-b4	設置讀操作的啓動時刻：
--------	-------	-------------

00 : 在 RDY 信號上升沿之後讀取

01 : 在 CLE 下降沿之後讀取

10 : 在 ALE 下降沿之後讀取

11 : 保留

WRCMD	b3	讀/寫命令控制位： 0：讀命令 1：寫命令
FLWPN	b2	WPN 信號線控制位： 0：WPN 輸出低電平 1：WPN 輸出高電平
FLCEN	b1	CEN 信號線控制位： 0：CEN 輸出低電平 1：CEN 輸出高電平
FLEN	b0	Flash 介面使能位： 0：Flash 介面禁止 1：Flash 介面使能 如需連接 Flash 記憶體，該位必須設置為 1

■ NAND 命令暫存器：P_NAND_CLE_COMMAND(0x88190004)

當 Flash 控制器不忙的時候向該暫存器寫入資料時，Flash 控制器將啓動 CLE 信號輸出，並將寫入 NAND 命令暫存器的命令送至八位元資料匯流排，同時會啓動 WEN 信號的輸出，以便將命令寫入 Nand 型 Flash。如果當控制器忙的時候向該暫存器寫入資料，將引起命令丟失 (CMDLOSS) 旗標位置 1，同時觸發命令丟失中斷。

表 4-151 P_NAND_CLE_COMMAND(0x88190004)

位	b7~b0
讀/寫	R/W
預設值	0
名稱	FL_CLE

FL_CLE	b7-b0	Flash 控制命令： 向該暫存器寫入資料將觸發 CLE 信號以及 WEN 信號的輸出
--------	-------	--

■ NAND 位址暫存器：P_NAND_ALE_ADDR(0x88190008)

當 Flash 控制器不忙的時候向該暫存器寫入資料時，Flash 控制器將啓動 ALE 信號輸出，並

將寫入該暫存器的位址根據 P_NAND_MODE_CTRL 暫存器的 ALECYC 設置從最低位元組開始依次寫入資料匯流排，同時啓動 WEN 信號的輸出，以便寫入 Nand 型 Flash。

表 4-152 P_NAND_ALE_ADDR(0x88190008)

位	b31~b0
讀/寫	R/W
預設值	0
名稱	FL_ALE

FL_ALE	b31-b0	Flash 操作位址： 向該暫存器寫入資料將觸發 ALE 信號以及 WEN 信號的輸出。 WEN 信號的輸出數量由 P_NAND_MODE_CTRL 暫存器的 ALECYC 確定
--------	--------	--

■ NAND 發送資料暫存器：P_NAND_TX_DATA(0x8819000C)

當 P_NAND_INT_STATUS 暫存器的 WREMPY 位為 1 的時候必須向該暫存器寫入資料，否則，將引起寫資料丟失 (WRLOSS) 位元將被置 1，同時觸發寫資料丟失中斷，此時，Nand 型 Flash 會認為控制器已經完成寫操作，從而拉低自己的 RDY 插腳，開始編程操作，從而導致用戶此後無法寫入資料。

表 4-153 P_NAND_TX_DATA(0x8819000C)

位	b31~b0
讀/寫	R/W
預設值	0
名稱	FL_WD

FL_WD	b31-b0	向 NAND Flash 寫入的資料
-------	--------	--------------------

■ NAND 接收資料暫存器：P_NAND_RX_DATA(0x88190010)

當 P_NAND_INT_STATUS 暫存器的 RDFULL 位為 1 的時候必須從該暫存器讀出資料，否則，將引起讀資料錯誤 (RDMISS) 位元將被置 1，同時觸發讀資料錯誤中斷，此時，控制器停止 REN 信號的輸出，導致 Nand 型 Flash 認為讀取操作結束，從而使此後讀出操作無效。

表 4-154 P_NAND_RX_DATA(0x88190010)

位	b31~b0
讀/寫	R/W
預設值	0
名稱	FL_RD

FL_RD	b31-b0	從 NAND Flash 讀出的資料
-------	--------	--------------------

■ NAND 中斷狀態暫存器：P_NAND_INT_STATUS(0x88190018)

NAND 中斷狀態暫存器用於查詢控制器的中斷狀態。

表 4-155 P_NAND_INT_STATUS(0x88190018)

位	b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
預設值	0	0	0	0	0	0	0	0
名稱	RDY	BUSY	CMDLOSS	RDMISS	WRLOSS	RDYRISE	RDFULL	WREMPTRY

RDY	b7	Nand Flash 就緒旗標位： 0 : Nand Flash 忙 1 : Nand Flash 就緒 該位反應了對 Nand Flash 的 RDY 插腳的檢測結果
-----	----	---

BUSY	b6	控制器忙碌旗標位： 0 : 控制器空閒，此時 ALE/CLE 暫存器可以被寫入 1 : 控制器忙，此時 ALE/CLE 暫存器不可被寫入，否則，CMDLOSS 旗標將被置位，寫入的資料將丟失
------	----	---

CMDLOSS	b5	命令丟失中斷旗標位： 讀 0 : 沒有發生命令丟失中斷 讀 1 : 發生命令丟失中斷 寫 0 : 無意義 寫 1 : 清除中斷旗標
---------	----	---

RDMINSS	b4	讀資料錯誤中斷旗標位： 讀 0：沒有發生讀資料錯誤中斷 讀 1：發生讀資料錯誤中斷 寫 0：無意義 寫 1：清除中斷旗標
WRLOSS	b3	寫資料丟失中斷旗標位： 讀 0：沒有發生寫資料丟失中斷 讀 1：發生寫資料丟失中斷 寫 0：無意義 寫 1：清除中斷旗標
RDYRISE	b2	RDY 信號中斷旗標位： 讀 0：RDY 信號沒有上升沿輸入 讀 1：RDY 信號有上升沿輸入 寫 0：無意義 寫 1：清除中斷旗標
RD_FULL	b1	讀緩衝區滿中斷旗標位： 讀 0：讀緩衝區未滿 讀 1：讀緩衝區滿 該位將在讀取 P_NAND_RX_DATA 暫存器後自動清除
WREMPTRY	b0	寫緩衝區空中斷旗標位： 讀 0：寫緩衝區未空 讀 1：寫緩衝區空 該位元將在向 P_NAND_TX_DATA 暫存器寫入資料後自動清除

所有 Flash 都存在位翻轉的現象，Nand 型 Flash 也不例外。在某些情況下，一個位的資料被發現發生了翻轉錯誤。在大多數應用中，用戶需要使用錯誤校正（ECC）演算法來儘量避免錯誤的發生。SPCE3200 的 Nand 型 Flash 控制器內部具有兩個 ECC 計算電路，可以由硬體完成錯誤校正。下面是對 ECC 電路的控制暫存器的描述。

需要注意的是，兩個 ECC 計算電路可以配合針對 512 個位元組的資料進行錯誤校正，對於 2KB 大小頁容量的 Nand 型 Flash 晶片，需要將 2KB 的頁分為四次來進行計算。

■ NAND 實際行檢查碼暫存器：P_NAND_ECC_TRUELP(0x8819001C)

在透過控制器向 Nand Flash 寫入資料或從 Nand Flash 讀出資料的時候，控制器內部的兩個 ECC 計算電路可以計算 512 個位元組的資料的第 0~255 位元組以及第 256~511 位元組的行奇偶檢查碼，並將計算結果保存至該暫存器內。

在執行寫入操作的時候編程者應將該暫存器內容寫入 Nand Flash 對應頁的備用區域內，以便將來讀取該頁資料時可以進行糾錯檢查。

在執行讀取操作的時候，利用該暫存器內的實際資料的奇偶檢查值，配合 P_NAND_ECC_TRUECP、P_NAND_ECC_CMPLP 以及 P_NAND_ECC_CMPCP 暫存器的值，ECC 電路可以對資料進行錯誤檢查和糾正。

表 4-156 P_NAND_ECC_TRUELP(0x8819001C)

位	b31~b16	b15~b0
讀/寫	R/W	R/W
預設值	0	0
名稱	TRUELP1	TRUELP0

TRUELP1	b31~b16	第 256~511 位元組的實際行奇偶檢查碼
TRUELP0	b15~b0	第 0~255 位元組的實際行奇偶檢查碼

■ NAND 實際列檢查碼暫存器：P_NAND_ECC_TRUECP(0x88190020)

表 4-157 P_NAND_ECC_TRUECP(0x88190020)

位	b11~b6	b5~b0
讀/寫	R/W	R/W
預設值	111111	111111
名稱	TRUECP1	TRUECP0

TRUECP1	b11~b6	第 256~511 位元組的實際列奇偶檢查碼
TRUECP0	b5~b0	第 0~255 位元組的實際列奇偶檢查碼

■ NAND 對比行檢查碼暫存器：**P_NAND_ECC_CMPLP(0x88190024)**

在從 Nand Flash 讀取資料時，編程者應該將之前寫入備用區域的行奇偶檢查碼讀出並保存至該暫存器，控制器內部的 ECC 電路可以利用該暫存器的值，並配合實際資料的行和列奇偶檢查值暫存器 P_NAND_ECC_TRUELP 和 P_NAND_ECC_TRUECP，以及 P_NAND_ECC_CMPCP 暫存器來對資料進行錯誤檢查和糾正。

表 4-158 P_NAND_ECC_CMPLP(0x88190024)

位	b31~b16	b15~b0
讀/寫	R/W	R/W
預設值	0	0
名稱	CALLP1	CALLP0

CALLP1	b31~b16	第 256~511 位元組的行奇偶檢查碼
CALLP0	b15~b0	第 0~255 位元組的行奇偶檢查碼

■ NAND 對比列檢查碼暫存器：**P_NAND_ECC_CMPCP(0x88190028)**

表 4-159 P_NAND_ECC_CMPCP(0x88190028)

位	b11~b6	b5~b0
讀/寫	R/W	R/W
預設值	111111	111111
名稱	TRUECP1	TRUECP0

CALCP1	b11~b6	第 256~511 位元組的列奇偶檢查碼
CALCP0	b5~b0	第 0~255 位元組的列奇偶檢查碼

■ NAND 檢查控制暫存器：P_NAND_ECC_CTRL(0x88190030)

表 4-160 P_NAND_ECC_CTRL(0x88190030)

位	b2	b1	b0
讀/寫	W	W	W
預設值	0	0	0
名稱	CLRECC1	CLRECC0	CALECC

CLRECC1	b2	ECC 電路的第 1 部分重設控制位： 寫 0：無意義 寫 1：重設第 256~511 位元組對應的 ECC 計算電路
CLRECC0	b1	ECC 電路的第 0 部分重設控制位： 寫 0：無意義 寫 1：重設第 0~255 位元組對應的 ECC 計算電路
CALECC	b0	ECC 檢查的啓動控制位： 寫 0：無意義 寫 1：ECC 電路將根據 P_NAND_ECC_TRUELP、P_NAND_ECC_TRUECP、P_NAND_ECC_CMPLP 和 P_NAND_ECC_CMPCP 暫存器的值對資料進行錯誤檢查，計算完成後該位將置 0

■ NAND 檢查狀態暫存器：P_NAND_ECC_STATUS(0x8819002C)

該暫存器保存了對 0~255 位元組和 256~511 位元組兩部分資料的 ECC 檢查結果，用戶可以讀取該暫存器內容獲得錯誤情況，並根據錯誤情況做錯誤校正。

表 4-161 P_NAND_ECC_STATUS(0x8819002C)

位	b31~b30	b29~b27	b26~b24	b23~b16	b15~b14	b13~b11	b10~b8	b7~b0
讀/寫	R	-	R	R	R	-	R	R
預設值	0	-	0	0	0	-	0	0
名稱	ERR1	-	ERRBIT1	ERRBYTE1	ERR0	-	ERRBIT0	ERRBYTE0

ERR1	b31~b30	第 256~511 位元組的檢查結果： 00：沒有錯誤 01：有一處錯誤，可以糾正（錯誤位置由 ERRBIT1 和 ERRBYTE1 指出） 10：保留 11：有兩處或以上的錯誤，不可糾正
ERRBIT1	b26~b24	ECC 電路檢測出的錯誤位元組中的錯誤位： 000 : ERRBYTE1 的第 0 位元錯誤 001 : ERRBYTE1 的第 1 位元錯誤 111 : ERRBYTE1 的第 7 位元錯誤
ERRBYTE1	b23~b16	ECC 電路檢測出的第 256~511 位元組中的錯誤位元組
ERR0	b15~b14	第 0~255 位元組的檢查結果： 00：沒有錯誤 01：有一處錯誤，可以糾正（錯誤位置由 ERRBIT0 和 ERRBYTE0 指出） 10：保留 11：有兩處或以上的錯誤，不可糾正
ERRBIT0	b10~b8	ECC 電路檢測出的錯誤位元組中的錯誤位： 000 : ERRBYTE0 的第 0 位元錯誤 001 : ERRBYTE0 的第 1 位元錯誤 111 : ERRBYTE0 的第 7 位元錯誤
ERRBYTE0	b7~b0	ECC 電路檢測出的第 0~255 位元組中的錯誤位元組

4.13.6 基本操作

1. Nand 型 Flash 控制器的初始化：

在使用 Nand 型 Flash 控制器之前，必須首先進行初始化。Nand 型 Flash 控制器的初始化比較簡單，只需要設置三個暫存器即可：

- (1) 設置 P_NAND_INTERFACE_SEL 暫存器，選擇對應插腳做為 Nand 型 Flash 介面使用；
- (2) 設置 P_NAND_CLK_CONF 暫存器，使能 Nand 型 Flash 控制器模組時鐘；

(3) 設置 P_NAND_INT_CTRL 暫存器，根據需要使能中斷。

參考代碼如下：

```
* P_NAND_INTERFACE_SEL = C_NAND_PORT_SEL;           // 選擇插腳為 NAND FLASH
* P_NAND_CLK_CONF = C_NAND_CLK_EN
| C_NAND_RST_DIS;                                     // 使能 NAND FLASH 模組時鐘
* P_NAND_INT_CTRL = 0x00000000;                      // 禁止 NAND 所有中斷
```

對 Nand 型 Flash 的操作大致包括發送命令、發送位址、寫/讀資料等。下麵針對每種操作敘述暫存器的操作流程。

2. 發送命令

(1) 設置 P_NAND_MODE_CTRL 暫存器，確定 Nand 型 Flash 操作時序。主要考慮的設置項包括：

- Flash 模組使能位 (FLEN) 必須設置為 1；
- CEN 插腳輸出設置位 (FLCEN) 一般設置為 0，選中 Nand 型 Flash 晶片；
- 根據需要設置防寫位 (FLWPN)、讀寫命令位 (WRCMD)；
- 根據實際連接的 Nand 型 Flash 晶片的容量和位址週期數確定 ALECYC；
- 根據實際連接的 Nand 型 Flash 晶片的時序要求設置 RDTYPE、WRLOW、WRHIGH、RDLOW、RDHIGH 等位；
- 根據需要設置本次需要傳輸的資料量 TBYTES。這一步設置是與後面對 Flash 的讀寫操作相關的，控制器將根據此設定值控制資料寫傳輸數量，同時，ECC 電路也將根據此設定值工作。

(2) 向 P_NAND_CLE_COMMAND 暫存器中寫入命令；

(3) 判斷 P_NAND_INT_STATUS 暫存器的 BUSY 位，等待控制器完成操作；

注意：

在發送命令的時候由於將同時設置位址週期數量，所以請首先根據實際連接的 Nand 型 Flash 晶片確定其需要的位址週期數。

參考代碼如下：

```
* P_NAND_MODE_CTRL = 0x00000001           // NAND 模組使能
| 0                                         // 片選，CEN 插腳輸出低電平
| 0x00000004                                // WPN 為 1，不保護
| 0                                         // WRCMD 為 0，讀命令
| 0x000000C0                                // WRLOW 和 RDLOW 設置為 2 個週期
| 0x00001100                                // WRHIGH 和 RDHIGH 設置為 1 個週期
| (4 - 1) << 16;                            // 本次傳輸 4 個位元組
* P_NAND_CLE_COMMAND = 0x90;                // 寫入命令 90
while(*P_NAND_INT_STATUS & 0x80);          // 等待控制器完成操作
```

3. 發送位址

通常根據 Nand 型 Flash 晶片的容量不同，定址需要的位址週期數也不同。對於不超過四個

位址週期的晶片，位址長度不超過 32 位，可以一次寫入 P_NAND_ALE_ADDR 暫存器並發送：

- (1) 設置 P_NAND_MODE_CTRL 暫存器的 ALECYC，選擇合適的週期模式，注意不要改變該暫存器其他位的值；
- (2) 向 P_NAND_ALE_ADDR 暫存器寫入 32 位位址(包括 8 位列位址和 10~13 位行位址)；
- (3) 判斷 P_NAND_INT_STATUS 暫存器的 BUSY 位，等待控制器完成操作；

由於在發送命令操作的時候已經設置過 ALECYC 位址週期，如果位址模式沒有發生變化，第一步可以省略掉。

參考代碼如下：

```
*P_NAND_MODE_CTRL &= 0xFFFFFFFF3F;           // 將 ALECYC 設置為 0
*P_NAND_MODE_CTRL |= 0x000000C0;             // 選擇 4 位址週期
// 上面兩步如果與發送命令時的設置一致則可以省略
*P_NAND_ALE_ADDR = address;                  // 寫入位址
while(*P_NAND_INT_STATUS & 0x80);            // 等待控制器完成操作
```

對於 5 位址週期的定址模式，定址位址長度超過 32 位，不能透過一次操作完成。一般可以透過下面的流程實現：

- (1) 設置 P_NAND_MODE_CTRL 暫存器的 ALECYC，選擇兩位址週期，注意不要改變暫存器其他位的值；
- (2) 向 P_NAND_ALE_ADDR 暫存器寫入列位址；
- (3) 判斷 P_NAND_INT_STATUS 暫存器的 BUSY 位，等待控制器完成操作；
- (4) 設置 P_NAND_MODE_CTRL 暫存器的 ALECYC，選擇三位址週期，注意不要改變暫存器其他位的值；
- (5) 向 P_NAND_ALE_ADDR 暫存器寫入行位址；
- (6) 判斷 P_NAND_INT_STATUS 暫存器的 BUSY 位，等待控制器完成操作；

參考代碼如下：

```
*P_NAND_MODE_CTRL &= 0xFFFFFFFF3F;           // 將 ALECYC 設置為 0
*P_NAND_MODE_CTRL |= 0x00000040;             // 選擇 2 位址週期
*P_NAND_ALE_ADDR = column_address;          // 寫入列位址
while(*P_NAND_INT_STATUS & 0x80);            // 等待控制器完成操作
*P_NAND_MODE_CTRL &= 0xFFFFFFFF3F;           // 將 ALECYC 設置為 0
*P_NAND_MODE_CTRL |= 0x00000080;             // 選擇 3 位址週期
*P_NAND_ALE_ADDR = row_address;              // 寫入行位址
while(*P_NAND_INT_STATUS & 0x80);            // 等待控制器完成操作
```

4. 寫資料

向 Flash 寫入資料的過程比較簡單：

- (1) 查詢 P_NAND_INT_STATUS 暫存器的 WREMPYTY 位，等待寫緩衝區空；

- (2) 向 P_NAND_TX_DATA 暫存器寫入 32 位資料；
- (3) 重複這個過程，直至發送完畢 P_NAND_MODE_CTRL 暫存器的 TBYTES 位元設置的資料個數；

參考代碼如下：512 位元組資料，512/4 字資料。

```
for(i=0;i<512/4;i++)  
{  
    while((*P_NAND_INT_STATUS & 0x01) == 0);           // 等待寫緩衝區空  
    *P_NAND_TX_DATA = data[i];                          // 寫入資料  
}
```

5. 讀資料

控制器可以根據 P_NAND_MODE_CTRL 暫存器中的 RDTYPE 以及 TBYTES 的設置，自動獲取由 Nand 型 Flash 發送過來的資料。從 Flash 讀取資料的過程如下：

- (1) 查詢 P_NAND_INT_STATUS 暫存器的 RDFULL 位，等待讀緩衝區滿；
- (2) 從 P_NAND_RX_DATA 暫存器讀取接收到的資料；
- (3) 重複這個過程，直至接收完畢 P_NAND_MODE_CTRL 暫存器的 TBYTES 位元設置的資料個數。

參考代碼如下：

```
for(i=0; i<512/4; i++)  
{  
    while((*P_NAND_INT_STATUS & 0x02) == 0);           // 等待讀緩衝區空  
    data[i] = *P_NAND_RX_DATA;                           // 讀取資料  
}
```

6. ECC 檢查

SPCE3200 內置的 ECC 檢查碼計算電路可以對 512 位元組資料進行 ECC 檢查碼的計算以及糾錯比對，如果用戶使用的 Nand 型 Flash 晶片的頁容量並非 512 位元組，則需要以 512 位元組為一組分別做 ECC 檢查。

使用 ECC 檢查，首先需要用戶在向 Flash 寫入資料的時候計算這些資料的 ECC 檢查碼，並將這些資料一併保存至 Flash 內，之後需要重新讀回資料時，同時也將先前寫入的檢查碼讀出，這樣，ECC 電路便可以計算實際讀取的資料的 ECC 檢查碼，並與先前寫入的檢查碼比較，以確認讀取的資料是否存在錯誤，並做錯誤糾正。

寫入資料時 ECC 檢查碼的生成是 ECC 電路自動完成的，用戶在使用時需要遵循如下流程：

- (1) 啓動寫操作之前，首先向 P_NAND_ECC_CTRL 暫存器的 CLRECC1 和 CLRECC0 位寫入 1，重設內部的兩個 ECC 電路；
- (2) 寫操作完成後，讀取 P_NAND_ECC_TRUELP 和 P_NAND_ECC_TRUECP 暫存器，得到 ECC 檢查值；
- (3) 根據需要保存 ECC 檢查值到 Flash。

參考代碼如下：

```
*P_NAND_ECC_CTRL = 0x00000006;           // 寫操作前重設 ECC 電路
.....
.....
uiEcc_Truelp = *P_NAND_ECC_TRUELP;        // 省略 FLASH 寫操作過程
uiEcc_Truecp = *P_NAND_ECC_TRUECP;        // 讀真實行檢查碼
uiEcc_Cmpelp = *P_NAND_ECC_CMPELP;        // 讀真實列檢查碼
.....
.....
// 省略 ECC 檢查碼保存過程，一般保存在 FLASH 備用區域
```

讀取資料時 ECC 檢查工作需要用戶手動控制 ECC 電路進行，應遵循如下流程：

- (1) 啓動讀操作之前，首先向 P_NAND_ECC_CTRL 暫存器的 CLRECC1 和 CLRECC0 位寫入 1，重設內部的兩個 ECC 電路；
- (2) 讀操作完成之後，同時將之前保存在 Flash 內的 ECC 檢查值讀出；
- (3) 將讀出的 ECC 檢查值寫入 P_NAND_ECC_CMPLP 和 P_NAND_ECC_CMPCP 暫存器；
- (4) 向 P_NAND_ECC_CTRL 暫存器的 CALECC 位寫入 1，啓動 ECC 電路進行檢查；
- (5) 讀取 P_NAND_ECC_STATUS 暫存器，判斷 512 位元組中是否存在錯誤，並進行糾錯工作。

參考代碼如下：

```
*P_NAND_ECC_CTRL = 0x00000006;           // 讀操作前重設 ECC 電路
.....
.....
// 需要將之前保存的 ECC 檢查碼重新讀取，一般從 flash 備用區域讀取，省略
*P_NAND_ECC_CMPLP = uiEcc_Cmplp;         // 寫入對比行檢查碼
*P_NAND_ECC_CMPCP = uiEcc_Cmpcp;         // 寫入對比列檢查碼
*P_NAND_ECC_CTRL = 1;                     // 啓動 ECC 電路進行檢查
uiEcc_Result = *P_NAND_ECC_STATUS;        // 讀取檢查結果
// 這裏省略根據檢查結果進行糾錯的過程
.....
```

4.14 SD 卡控制器

4.14.1 概述

安全數字 (SD, Secure Digital) 記憶體卡實際是一種快閃記憶體 (Flash) 卡，專門為滿足安全使用、大容量、高性能以及作為時尚音視頻電子產品內部需要的存儲配件而設計的。SD 存儲卡的通訊透過一種新式 9 針串列介面實現，可工作在一個低電壓範圍內。SD 卡使用的 9 針介面，除去電源和地，還包括以下信號線：

- CLK：從主機到 SD 卡的時鐘信號，用於實現同步串列通信和資料傳輸
- CMD：雙向的命令/回應信號線。主機透過此線向 SD 卡串列發送命令或透過此線串列接收 SD 卡的回應
- DAT0 ~ DAT3 : 4 位元雙向資料匯流排。主機透過該匯流排向 SD 卡寫入資料或從 SD 卡讀取資料

SD 卡內部的存儲區域是以扇區為單位管理的。在對存儲區域的資料進行擦除、寫和讀操作時，必須以扇區為單位進行。通常 SD 卡的每個扇區為 512 位元組。

主機並不能直接讀寫 SD 卡內部的資料，而是依靠命令控制 SD 卡進行操作。所以，主機與 SD 卡之間的通信分為命令位元流和資料位元流兩種形式，使用起始位元和結束位元來標識一個完整的位元流。

- 命令：由主機透過 CMD 線發送給 SD 卡，用以告知 SD 卡啟動什麼樣的操作
- 回應：在接到主機的命令後由 SD 卡回送給主機的應答資訊。SD 卡在接到主機的命令後大多數情況下會返回應答資訊給主機。回應資料也是透過 CMD 線傳輸的
- 資料：向 SD 卡寫入或從 SD 卡讀出的用戶資料，透過 4 位元寬度的資料匯流排傳輸在主機和 SD 卡之間傳輸

主機與 SD 卡之間最基本的通信是命令/回應傳輸（如圖 4-80 所示）。這種通信方式可以直接在主機與 SD 卡之間傳遞資訊。

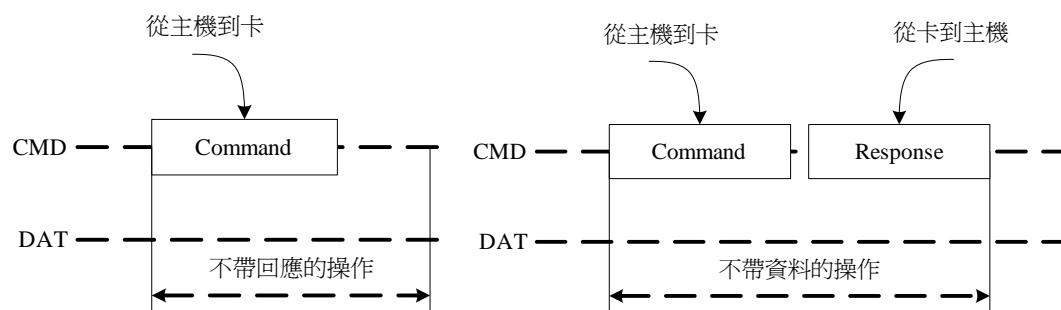


圖 4-80 命令/回應傳輸操作

有些命令將會帶有資料的傳輸，比如讀/寫扇區命令等。這些資料在 SD 卡接到命令之後開始透過 4 位元寬度的資料匯流排傳輸。圖 4-81 所示的是主機讀取 SD 卡內部的扇區資料的操作，圖 4-82 所示的是主機向 SD 卡寫入扇區資料的操作。可以看到，在基本的命令/回應傳輸的基礎上，配合 4 位元寬度的資料匯流排附加進行資料流程的傳輸。

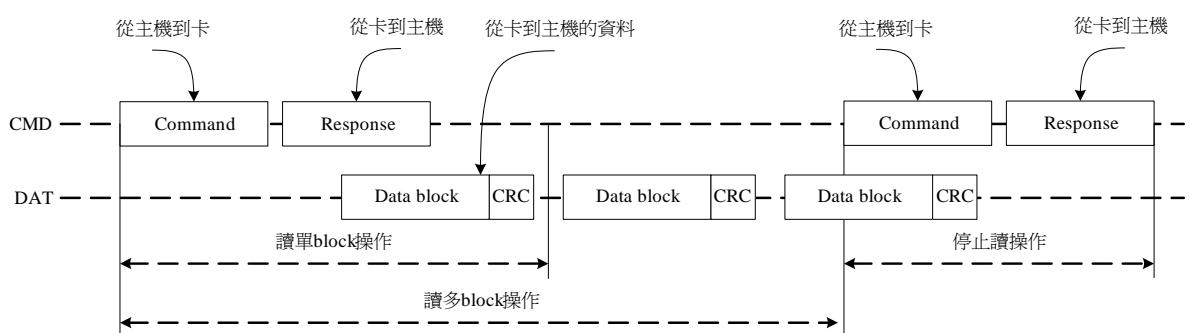


圖 4-81 讀扇區操作

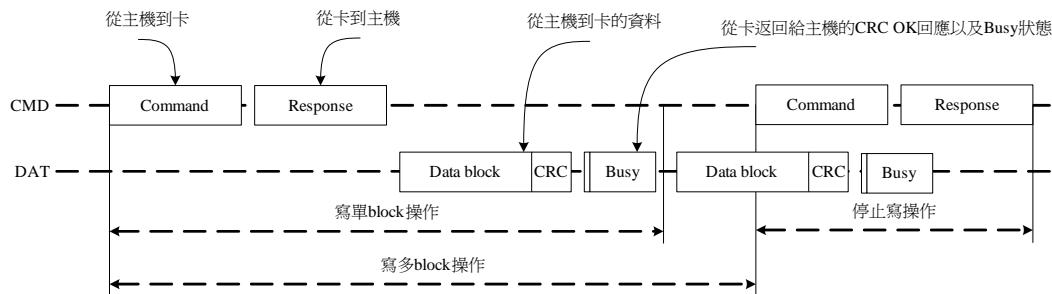


圖 4-82 寫扇區操作

SPCE3200 內部的 SD 卡控制器可以透過編程式控制制完成以上所有四種基本通信功能，並能夠提供透過 DMA 存取的高速傳輸性能，從而獲得產品的最佳成本表現比。

4.14.2 特性

SPCE3200 的 SD 卡控制器具有如下特點：

- 完全相容 SD 存儲卡的規格
- 能夠直接接收 SD 命令，提高了相容性
- 可編程選擇 SD 汇流排的時鐘速度
- 提供在緩存區滿狀態下 SD 汇流排的時鐘控制
- 提供中斷配置
- 支援 DMA 讀/寫操作
- 支援 1 位 / 4 位 SD 工作模式
- 支援對 SD I/O 卡的中斷檢測。SD I/O 卡是設計在 SD 存儲卡的基礎上並能與其相容的，其設計的目的是為了給那些移動電子設備提供高速且低功耗的資料輸入/輸出功能。

4.14.3 插腳描述

SPCE3200 的 SD 卡介面與 SPI 和 NAND 型 FLASH 介面複用，使用時需要設置相關暫存器以便使其工作在 SD 介面模式。SD 卡介面的插腳描述如表 4-162 所示。

表 4-162 SD 卡介面插腳對應表

插腳名稱	插腳號	插腳屬性	插腳功能
SD_CLK	126	O	SD 介面的 CLK 輸出插腳
SD_CMD	125	I/O	SD 介面的 CMD 插腳
SD_D[0]	117	I/O	SD 介面的資料線的 Bit0
SD_D[1]	116	I/O	SD 介面的資料線的 Bit1
SD_D[2]	115	I/O	SD 介面的資料線的 Bit2
SD_D[3]	112	I/O	SD 介面的資料線的 Bit3

4.14.4 結構

SD 卡控制器的結構如圖 4-83 所示。

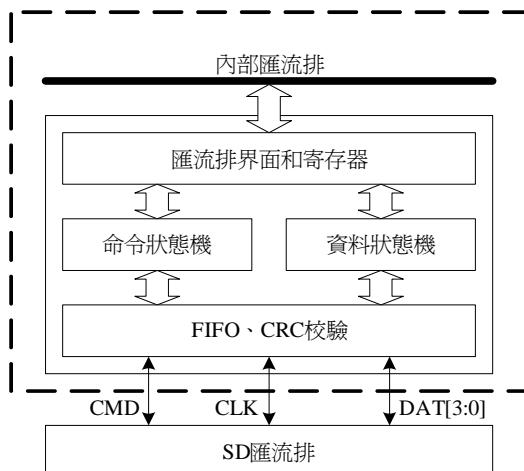


圖 4-83 SD 卡控制器結構

4.14.5 暫存器描述

SD 卡控制器共有 10 個暫存器，如表 4-163 所示。透過對這 10 個暫存器的操作，即可使用 SD 卡控制器擴展 SD 卡存儲設備。

表 4-163 SD 卡控制器相關暫存器列表

暫存器名稱	助記符	位址
SD 介面選擇暫存器	P_SD_INTERFACE_SEL	0x882000A4
SD 時鐘配置暫存器	P_SD_CLK_CONF	0x882100A4
SD 控制暫存器	P_SD_MODE_CTRL	0x88180018
SD 中斷控制暫存器	P_SD_INT_CTRL	0x8818001C
SD 中斷狀態暫存器	P_SD_INT_STATUS	0x88180014
SD 命令設置暫存器	P_SD_COMMAND_SETUP	0x88180008
SD 參數資料暫存器	P_SD_ARGUMENT_DATA	0x8818000C
SD 回應資料暫存器	P_SD_RESPONSE_DATA	0x88180010
SD 發送資料暫存器	P_SD_TX_DATA	0x88180000
SD 接收資料暫存器	P_SD_RX_DATA	0x88180004

■ SD 介面選擇暫存器：**P_SD_INTERFACE_SEL(0x882000A4)**

SPCE3200 的 SD 卡介面與 SPI 和 NAND 型 FLASH 介面複用，使用時需要設置 SD 介面選擇暫存器以便使其工作在 SD 介面模式。

表 4-164 P_SD_INTERFACE_SEL(0x882000A4)

位	b16	b15-b0
讀/寫	R/W	-
預設值	0	-
名稱	SD_EN	-

SD_EN

b16

SD 卡介面使能位：

0：禁止為 SD 卡介面

1：使能為 SD 卡介面

■ SD 時鐘配置暫存器：**P_SD_CLK_CONF(0x882100A4)**

SD 時鐘配置暫存器用於使能或禁止 SD 控制器模組時鐘，或軟重設 SD 卡控制器模組。

表 4-165 P_SD_CLK_CONF(0x882100A4)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	SD_RST	SD_STOP

SD_RST

b1

SD 模組時鐘重設位：

0：SD 模組時鐘重設

1：SD 模組時鐘不重設

SD_STOP

b0

SD 模組時鐘停止位：

0：SD 模組時鐘停止

1：SD 模組時鐘使能

■ SD 控制暫存器 : P_SD_MODE_CTRL(0x88180018)

SD 控制暫存器用於控制 SD 汇流排的時鐘頻率、扇區大小等工作特性。需要注意的是，該暫存器僅當 SD 卡控制器的 Status 暫存器的 BUSY 位為 0 時才可以被改變。

表 4-166 P_SD_MODE_CTRL(0x88180018)

位	b31~b28			b27~b16		
讀/寫	-			W		
預設值	-			0		
名稱	-			BLKLEN		
位	b15~b12	b11	b10	b9	b8	b7~b0
讀/寫	-	W	W	W	W	W
預設值	-	0	0	0	0	0x54
名稱	-	EN_SD	IOEN	DMAMODE	BUSWIDTH	CLKDIV

BLKLEN b27~b16 SD 卡的扇區長度。以位元組為單位。該值應該與實際連接的 SD 卡的扇區長度相等

EN_SD b11 SD 汇流排使能控制。該位元控制 I/O 埠是否做為 SD 汇流排使用還是做為 GPIO 使用。如果希望使用 SD 卡控制器，該位必須設置為 1。

0 : SD_CLK、SD_CMD、SD_DAT0 做為 GPIO

1 : SD_CLK、SD_CMD、SD_DAT0 做為 SD 汇流排

另外，如果 BUSWIDTH 位設置為 1，當 EN_SD 位為 1 時 SD_DAT1、SD_DAT2、SD_DAT3 也做為 SD 汇流排使用

IOEN b10 I/O 卡中斷使能：

0 : 禁止 I/O 卡中斷

1 : 使能 I/O 卡中斷

DMAMODE b9 DMA 傳輸方式選擇位：

0 : 不使用 DMA 傳輸方式

1 : 使用 DMA 傳輸方式

BUSWIDTH	b8	SD 資料匯流排寬度設置位： 0：1 位寬度 1：4 位寬度
CLKDIV	b7~b0	SD 匯流排的時鐘頻率設置，SD 匯流排的時鐘頻率計算如下： $f_{SDCLK} = f_{SYSCLK} / 2(CLKDIV + 1)$ 。CLKDIV 的預設值為 0x54

■ SD 中斷控制暫存器：**P_SD_INT_CTRL(0x8818001C)**

SD 中斷控制暫存器用於使能或禁止 SD 卡控制器的中斷。SD 卡控制器使用 IRQ40 中斷。

表 4-167 P_SD_INT_CTRL(0x8818001C)

位	b31~b7	b6	b5	b4	b3	b2	b1	b0
讀/寫	-	R/W						
預設值	-	0	0	0	0	0	0	0
名稱	-	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0

INTEN6	b6	I/O 卡中斷使能位： 0：禁止 I/O 卡中斷 1：使能 I/O 卡中斷
INTEN5	b5	卡插入/拔出中斷使能位： 0：禁止卡插入/拔出中斷 1：使能卡插入/拔出中斷
INTEN4	b4	資料緩衝區清空中斷使能位： 0：禁止資料緩衝區清空中斷 1：使能資料緩衝區清空中斷
INTEN3	b3	資料緩衝區溢出中斷使能位： 0：禁止資料緩衝區溢出中斷 1：使能資料緩衝區溢出中斷

 INTEN2 b2 命令緩衝區溢出中斷使能位：

0：禁止命令緩衝區溢出中斷

1：使能命令緩衝區溢出中斷

 INTEN1 b1 資料傳輸完成中斷使能位：

0：禁止資料傳輸完成中斷

1：使能資料傳輸完成中斷

 INTEN0 b0 命令發送完成中斷使能位：

0：禁止命令發送完成中斷

1：使能命令發送完成中斷

■ SD 中斷狀態暫存器：P_SD_INT_STATUS(0x88180014)

讀取 SD 中斷狀態暫存器的內容可以判斷 SD 卡控制器的狀態。

表 4-168 P_SD_INT_STATUS(0x88180014)

位	b31~b14	b13	b12	b11	b10		b9	b8	
讀/寫	-	R	R	-	R		R	R	
預設值	-	0	0	-	0		0	0	
名稱	-	CARDINT	CARDPRE	-	DATCRCERR		TIMEOUT	DATBUFEMPTY	
位	b7	b6	b5	b4	b3	b2	b1	b0	
讀/寫	R	R	R	R	R	R	R	R	
預設值	0	0	0	0	0	0	0	0	
名稱	DATBUF FULL	CMDBUF FULL	RSPCRCER R	RSPIDXERR	DATCOM	CMDCOM	CARDBUSY	BUSY	

 CARDINT b13 I/O 卡中斷掛起旗標位：

0：沒有發生 I/O 卡掛起中斷

1：發生 I/O 卡掛起中斷

在 P_SD_MODE_CTRL 暫存器中的 IOEN 位為 1 時這一位才會被置位。主機需要透過設備特殊命令來清除該中斷旗標，將這一位元寫為 1 沒有意義

CARDPRE	b12	SD 卡是否插入： 0：SD 卡未插入 1：SD 卡插入 這一位僅在 SD 卡控制器空閒時檢測 SD 介面的 DAT3 管腳，控制器的工作不會受此位元的影響，且主機不論這一位元為何值都可以啓動資料傳輸。 將此位寫 1 會清除卡插入掛起的中斷狀態旗標。
DATCRCERR	b10	接收資料的 CRC 檢查碼錯誤，或發送資料後 SD 卡返回 CRC 錯誤回應 0：沒有發生 CRC 錯誤 1：發生 CRC 錯誤
TIMEOUT	b9	命令人回應超時、讀資料回應超時或發送完寫命令後一段時間內沒有資料寫入 P_SD_TX_DATA 而引起的超時。 0：沒有超時 1：發生超時
DATBUFEMPTY	b8	當存儲在 FIFO 中的資料數目未超過特定數目 (trigger level) 時這一位會置 1，表示資料緩衝區空。 向 P_SD_TX_DATA 暫存器寫入合適數目的資料，或將 P_SD_COMMAND_SETUP 暫存器中的 STPCMD 位置 1 時會將這一位清 0。
DATBUFFULL	b7	當存儲在 FIFO 中的資料數目超過特定數目 (trigger level) 時這一位會置 1，表示資料緩衝區滿。 清除該旗標的方法：從 P_SD_RX_DATA 暫存器讀出合適數目的資料；將 P_SD_COMMAND_SETUP 暫存器中的 STPCMD 位置 1 時會將這一位清 0。
CMDBUFFULL	b6	命令人回應暫存器 P_SD_RESPONSE_DATA 滿則該位被置 1。 清除該旗標的方法：讀取 P_SD_RESPONSE_DATA 的值 啓動一個新的傳輸；將 P_SD_COMMAND_SETUP 暫存器中的 STPCMD 位置 1。
RSPCRCERR	b5	回應資料的 CRC 檢查碼錯誤則該位元被置 1。 在 R3 型回應情況下接收到的 CRC 檢查碼不為 6b'111111 則該位被置 1
RSPIDXERR	b4	回應中的索引是否出錯： 0：正常 1：出錯

DATCOM	b3	資料傳輸完畢旗標位： 0：資料傳輸未完成 1：資料傳輸完畢
CMDCOM	b2	命令傳輸完畢旗標位： 0：命令傳輸未完成 1：命令傳輸完畢
CARDBUSY	b1	SD 卡是否 BUSY (DAT0 是否為低電平)： 0：SD 卡不忙 1：SD 卡忙 注意：主機在發送寫命令後需要查詢此位
BUSY	b0	SD 卡控制器忙碌旗標位： 0：控制器空閒 1：控制器忙碌

■ SD 命令設置暫存器：P_SD_COMMAND_SETUP(0x88180008)

主機透過 SD 命令設置暫存器控制 SD 卡控制器的操作，SD 卡控制器透過讀取這個暫存器的值決定是否向 SD 卡發送命令。

表 4-169 P_SD_COMMAND_SETUP(0x88180008)

位	b14~b12	b11	b10	b9	b8	b7	b6	b5~b0
讀/寫	W	W	W	W	W	W	W	W
預設值	0	0	0	0	0	0	0	0
名稱	RESPTYPE	INICARD	MULBLK	TxDATA	CMDWD	RUNCMD	STPCMD	CMDCODE

RESPTYPE	b14~b12	命令的回應類型： 000：沒有回應 001：R1 類型的回應 010：R2 類型的回應 011：R3 類型的回應 110：R6 類型的回應 111：R1b 類型的回應 目前只有 R2 類型具有 128bit 的響應資料長度，其他的回應類型均為 32bit 回應資料長度
INICARD	b11	向該位寫 1 將使 SD 卡控制器透過 SD 汱流排的時鐘線發送 74 個脈衝
MULBLK	b10	當前命令是否為多扇區操作命令： 0：單扇區操作命令 1：多扇區操作命令
TxDATA	b9	當前命令是否需要發送/接收資料： 0：需要接收資料（Read 操作） 1：需要發送資料（Write 操作） 注意：該位僅當下麵的 CMDWD 位設置為 1 時有效
CMDWD	b8	當前命令是否伴隨有資料傳輸： 0：沒有資料傳輸 1：有資料傳輸
RUNCMD	b7	將該位寫為 1 將啟動 SD 卡控制器按照當前的配置發送命令給 SD 卡。 該位將在 SD 卡控制器啟動命令傳輸後自動清 0，但是，用戶必須等待 P_SD_INT_STATUS 暫存器的 BUSY 位為 0 時啟動下一次命令傳輸。
STPCMD	b6	向該位寫 1 將迫使 SD 卡控制器回到 IDLE 狀態。當 SD 卡控制器回到 IDLE 狀態後，該位元自動清 0。
CMDCODE	b5~b0	主機要發送的命令序號

■ SD 參數資料暫存器：P_SD_ARGUMENT_DATA(0x8818000C)

主機把需要傳遞給 SD 卡的參數寫入 SD 參數資料暫存器。SD 卡控制器將此暫存器內容做為命令參數發送給 SD 卡。

表 4-170 P_SD_ARGUMENT_DATA(0x8818000C)

位	b31~b0
讀/寫	W
預設值	0
名稱	Argument

Argument	b31~b0	傳輸給 SD 卡的命令參數，有效位為 b15~b0
----------	--------	---------------------------

■ SD 回應資料暫存器：P_SD_RESPONSE_DATA(0x88180010)

由 SD 卡返回的回應將被保存在該暫存器內。注意，SD 卡控制器並不關心該暫存器內的回應資料的內容和意義，但是主機必須對此關注。

R1、R1b、R3、R6 型回應具有 6 位的命令索引和 32 位的回應資料，而 R2 型回應具有 128 位的回應資料。主機需要查詢 P_SD_INT_STATUS 暫存器的 CMDBUFFULL 位，當該位為 1 時才可以讀取該暫存器內的回應資料。

表 4-171 P_SD_RESPONSE_DATA(0x88180010)

位	b31~b0
讀/寫	R
預設值	0
名稱	Response

Response	b31~b0	來自 SD 卡的回應資料
----------	--------	--------------

■ SD 發送資料暫存器：P_SD_TX_DATA(0x88180000)

當主機寫入 32 位資料到此暫存器後，SD 卡控制器就會將該資料發送給 SD 卡。而當資料發送完畢後，狀態暫存器 P_SD_INT_STATUS 中的 DATBUFEMPTY 位元會置 1，或產生 DMA 傳輸請求。而向此暫存器寫入資料，也必須等待 P_SD_INT_STATUS 暫存器中的 DATBUFEMPTY 位為 1。

表 4-172 P_SD_TX_DATA(0x88180000)

位	b31-b0
讀/寫	W
預設值	0
名稱	DataTx

DataTx b31~b0 需要發送給 SD 卡的資料，需等待 P_SD_INT_STATUS 暫存器的 DATBUFEMPTY 位為 1 時向該暫存器寫入資料

■ SD 接收資料暫存器：P_SD_RX_DATA(0x88180004)

從 SD 卡讀出的資料要存放在這個暫存器裏。當由 SD 卡接收到 32 位資料時，狀態暫存器 P_SD_INT_STATUS 中的 DATBUFFULL 位元會置 1，或會產生 DMA 請求。注意，只有在 DATBUFFULL 位為 1 時才能從此暫存器裏讀出資料。

表 4-173 P_SD_RX_DATA(0x88180004)

位	b31-b0
讀/寫	R
預設值	0
名稱	DataRx

DataRx b31-b0 從 SD 卡接收到的資料，需等待 P_SD_INT_STATUS 暫存器的 DATBUFFULL 位為 1 時才能從該暫存器讀取資料

4.14.6 基本操作

1. SD 卡控制器的初始化：

在使用 SD 卡控制器之前，必須對其進行初始化，設置其工作模式。

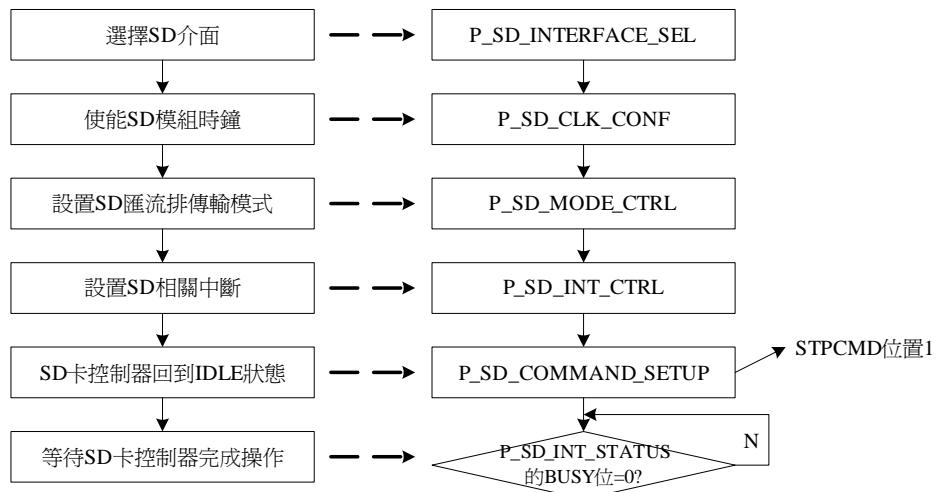


圖 4-84 SD 卡控制器初始化步驟

- (1) 設置 P_SD_INTERFACE_SEL 暫存器，選擇 SD 模組控制器介面；
- (2) 設置 P_SD_CLK_CONF 暫存器，使能 SD 卡控制器模組時鐘；
- (3) 設置 P_SD_MODE_CTRL 暫存器，設定 SD 卡的扇區大小、使能 SD 卡控制器、根據需要打開或者關閉 DMA 傳輸方式、根據需要設置 SD 資料匯流排寬度；
- (4) 設置 P_SD_INT_CTRL 暫存器，根據需要打開或關閉 SD 卡相關中斷；
- (5) 設置 P_SD_COMMAND_SETUP 暫存器的 STPCMD 位為 1，使 SD 卡控制器回到 IDLE 狀態；
- (6) 讀取 P_SD_INT_STATUS 暫存器，等待 BUSY 位為 0。

參考代碼如下：

```

*P_SD_INTERFACE_SEL = C_SD_PORT_SEL;           // 選擇複用埠為 SD 介面
*P_SD_CLK_CONF = C_SD_CLK_EN | C_SD_RST_DIS;   // 使能 SD 模組時鐘
*P_SD_MODE_CTRL = (512 << 16)                 // 設置 SD 卡扇區為 512 位元組
        | C_SD_BUS_4BIT                         // 設置 SD 資料匯流排為 4bit
        | C_SD_PORT_EN;                        // 使能 SD 控制器
*P_SD_INT_CTRL = C_SD_CARDDETECT_INTEN;         // 使能 SD 卡插入檢測中斷
*P_SD_COMMAND_SETUP = C_SD_74CLK_START;          // 初始化
while(*P_SD_INT_STATUS & C_SD_CTRL_BUSY);       // 等待 SD 控制器完成操作
    
```

接下來，便可以使用 SD 卡控制器向 SD 卡發送命令，控制 SD 卡的讀寫操作。下面以發送命令為例，說明控制 SD 卡控制器發送命令和資料傳輸的一般步驟。

主機向 SD 卡發送的命令大致有如下四種類型：不帶回應的命令（基本操作，用於控制 SD 卡的動作）、不帶資料的命令（基本操作，用於控制 SD 卡的動作並從 SD 卡讀取基本資訊）、帶資料的讀操作命令（用於從 SD 卡讀取資料）、帶資料的寫操作命令（用於向 SD 卡寫入資料）。分別說明使用 SD 卡控制器實現四種操作的過程：

2. 不帶回應的命令

SD 卡的命令集中有些命令，SD 卡是不需要返回回應給主機的，如命令序號為 0 的迫使 SD 卡回到 IDLE 狀態命令。使用這類命令對應的暫存器操作順序為：

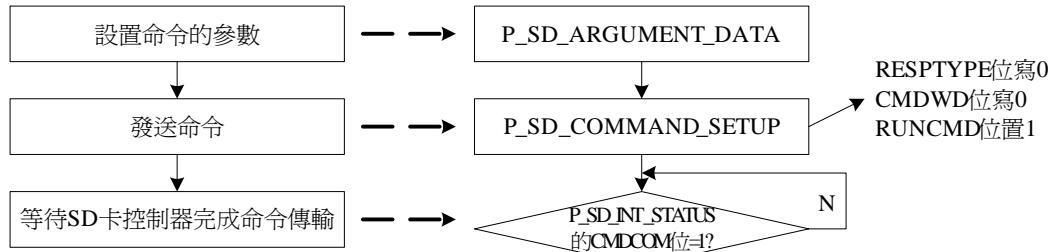


圖 4-85 使用不帶回應的命令的操作步驟

- (1) 首先將命令參數寫入 P_SD_ARGUMENT_DATA 暫存器；
- (2) 向 P_SD_COMMAND_SETUP 暫存器寫入命令序號，同時，
 - 將 RESPTYPE 對應位設置為 0，表示該命令不帶有回應資料；
 - 將 CMDWD 位設置為 0，表示該命令不帶有資料傳輸；
 - 將 RUNCMD 位設置為 1，以便在寫入命令序號後立即啟動 SD 卡控制器的命令傳輸；
- (3) 查詢 P_SD_INT_STATUS 暫存器的 CMDCOM 位，等待該位為 1，此時 SD 卡控制器的命令傳輸操作完成。

參考代碼如下：

```

// 以發送 CMD0 命令為例
*P_SD_ARGUMENT_DATA = 0;                                // 該命令沒有參數
*P_SD_COMMAND_SETUP = C_SD_RSP_R0                      // 沒有回應
| C_SD_CMD_WITHOUTDATA                                // 沒有參數
| C_SD_CMD_START                                     // 立即發送
| 0;                                                 // SD 命令 0，CMD0
while((*P_SD_INT_STATUS & C_SD_CMD_COMPLETE) == 0); // 等待命令傳輸完成
    
```

3. 不帶資料的命令

SD 卡的命令集中有一些命令需要 SD 卡返回回應給主機，但是不需要有其他資料傳輸的過程。比如命令序號為 9 的讀取 SD 卡 CSD 暫存器的命令。使用這類命令對應的暫存器操作順序為：

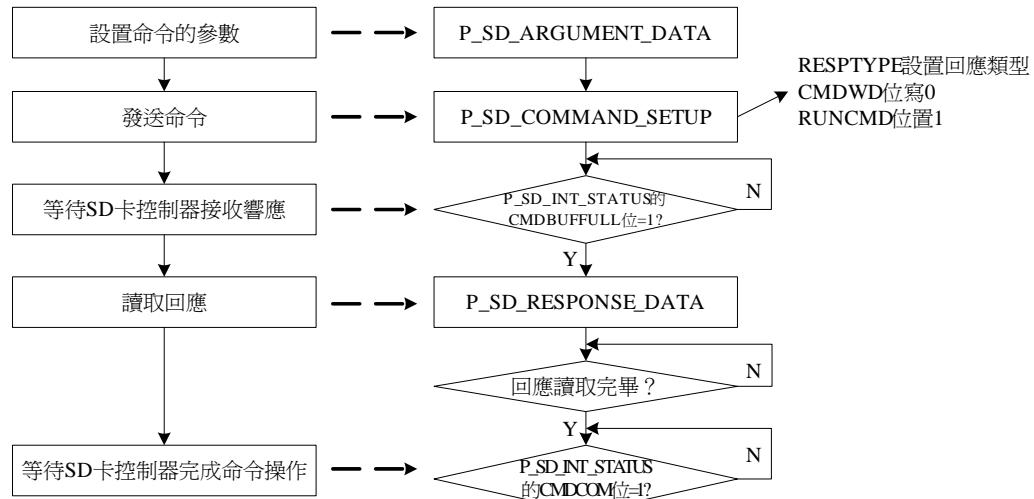


圖 4-86 使用不帶資料的命令的操作步驟

- (1) 首先將命令參數寫入 P_SD_ARGUMENT_DATA 暫存器；
- (2) 向 P_SD_COMMAND_SETUP 暫存器寫入命令序號，同時，
 - 將 RESPTYPE 對應位設置為該命令對應的回應類型；
 - 將 CMDWD 位設置為 0，表示該命令不帶有資料傳輸；
 - 將 RUNCMD 位設置為 1，以便在寫入命令序號後立即啟動 SD 卡控制器的命令傳輸；
- (3) 查詢 P_SD_INT_STATUS 暫存器的 CMDBUFFULL 位，等待該位為 1，此時 SD 卡控制器接收到 32 位長度的回應資料；
- (4) 讀取 P_SD_RESPONSE_DATA 暫存器，得到 SD 卡發送回來的回應；
- (5) 如果回應資料不止 32 位，則跳至第 3 步，重複讀取過程，直至所有回應讀取完畢；
- (6) 查詢 P_SD_INT_STATUS 暫存器的 CMDCOM 位，等待該位為 1，此時 SD 卡控制器的命令傳輸操作完成。

參考代碼如下：

```

// 以發送 CMD9 查詢 SD 卡的 CSD 暫存器命令為例
*P_SD_COMMAND_SETUP = arg;                                // 寫入命令參數
*P_SD_COMMAND_SETUP = C_SD_CMD_WITHOUTDATA                // 沒有資料
| C_SD_RSP_R2                                              // 回應類型為 R2
| C_SD_CMD_START                                            // 立即發送
| 9;                                                       // SD 命令 9，CMD9
for(i=0; i<4; i++)
{
    while((*P_SD_INT_STATUS & C_SD_CMDBUF_FULL) == 0);   // 等待響應暫存器滿
    response[i] = *P_SD_RESPONSE_DATA;                      // 讀取回應
}
while((*P_SD_INT_STATUS & C_SD_CMD_COMPLETE) == 0);      // 等待命令傳輸完成
    
```

4. 帶資料的讀操作命令

當主機需要讀取 SD 卡的扇區時，需要使用帶資料的讀操作命令。使用這類命令對應的暫存器操作順序為：

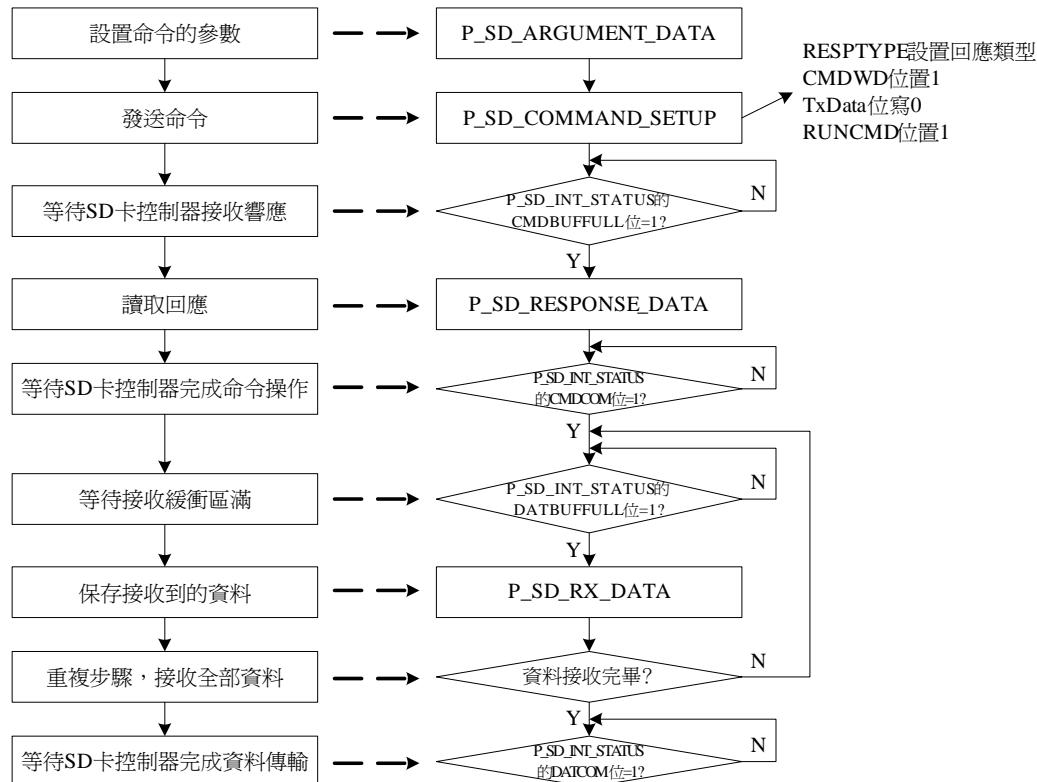


圖 4-87 使用帶資料的讀操作命令的操作步驟

- (1) 首先將命令參數寫入 **P_SD_ARGUMENT_DATA** 暫存器；
- (2) 向 **P_SD_COMMAND_SETUP** 暫存器寫入命令序號，同時，
 - 將 RESPTYPE 對應位設置為該命令對應的回應類型；
 - 將 CMDWD 位設置為 1，表示該命令帶有資料傳輸；
 - 將 TxData 位設置為 0，表示該命令需要讀取 SD 卡的資料；
 - 將 RUNCMD 位設置為 1，以便在寫入命令序號後立即啟動 SD 卡控制器的命令傳輸；
- (3) 查詢 **P_SD_INT_STATUS** 暫存器的 CMDBUFFULL 位，等待該位為 1，此時 SD 卡控制器接收到 32 位長度的回應資料；
- (4) 讀取 **P_SD_RESPONSE_DATA** 暫存器，保存 SD 卡發送回來的回應（一般這類命令只有 32 位的回應）；
- (5) 查詢 **P_SD_INT_STATUS** 暫存器的 CMDCOM 位，等待該位為 1，此時 SD 卡控制器的命令傳輸操作完成；
- (6) 查詢 **P_SD_INT_STATUS** 暫存器的 DATBUFFULL 位，等待該位為 1，此時 SD 卡控制器接收到 4 位元組的資料；

- (7) 從 P_SD_RX_DATA 暫存器讀取接收到的資料並保存；
- (8) 跳至第 6 步，重複接收資料過程，直至接收完畢整個扇區的資料；
- (9) 查詢 P_SD_INT_STATUS 暫存器的 DATCOM 位，等待該位為 1，此時 SD 卡控制器完成資料接收工作。

參考代碼如下：

```
// 以發送 CMD17 讀扇區命令為例
*P_SD_ARGUMENT_DATA = blockaddress;                                // 寫入命令參數，扇區位址
*P_SD_COMMAND_SETUP = C_SD_CMD_WITHDATA
    | C_SD_DATA_RECEIVE          // 帶資料
    | C_SD_RSP_R1                // 資料接收
    | C_SD_CMD_START              // 回應類型為 R1
    | C_SD_CMD_START              // 立即發送
    | 17;                         // SD 命令 17，CMD17
while((*P_SD_INT_STATUS & C_SD_CMDBUF_FULL) == 0);           // 等待響應暫存器滿
response = *P_SD_RESPONSE_DATA;                                // 保存回應資料
while((*P_SD_INT_STATUS & C_SD_CMD_COMPLETE) == 0);           // 等待命令傳輸完畢
for(i=0; i<512/4; i++)                                         // 接收一個扇區 512 位元組資料
{
    while((*P_SD_INT_STATUS & C_SD_DATABUF_FULL) == 0); // 等待接收緩衝區滿
    BlockData[i] = *P_SD_RX_DATA;                          // 以字為單位保存收到的資料
}
while((*P_SD_INT_STATUS & C_SD_DATA_COMPLETE) == 0); // 等待資料傳送完畢
```

5. 帶資料的寫操作命令

當主機需要向 SD 卡的扇區寫入資料時，需要使用帶資料的寫操作命令。使用這類命令對應的暫存器操作順序為：

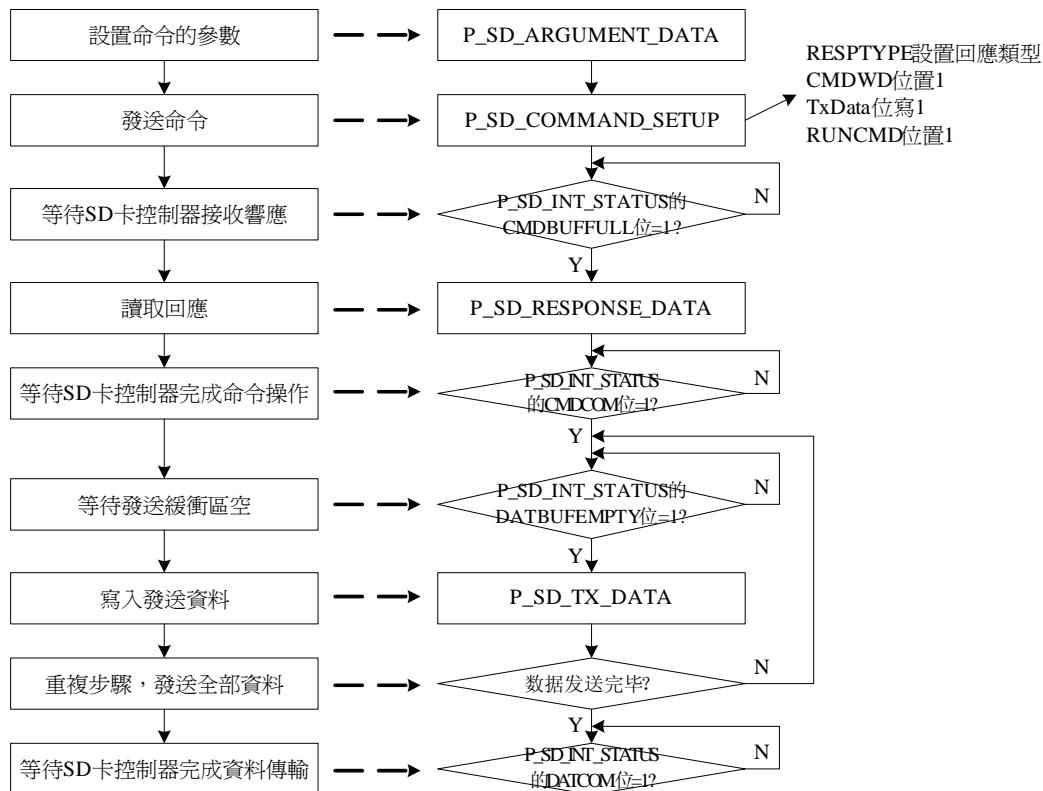


圖 4-88 使用帶資料的寫操作命令的操作步驟

- (1) 首先將命令參數寫入 P_SD_ARGUMENT_DATA 暫存器；
- (2) 向 P_SD_COMMAND_SETUP 暫存器寫入命令序號，同時，
 - 將 RESPTYPE 對應位設置為該命令對應的回應類型；
 - 將 CMDWD 位設置為 1，表示該命令帶有資料傳輸；
 - 將 TxData 位設置為 1，表示該命令需要向 SD 卡寫入資料；
 - 將 RUNCMD 位設置為 1，以便在寫入命令序號後立即啟動 SD 卡控制器的命令傳輸；
- (3) 查詢 P_SD_INT_STATUS 暫存器的 CMDBUFFULL 位，等待該位為 1，此時 SD 卡控制器接收到 32 位長度的回應資料；
- (4) 讀取 P_SD_RESPONSE_DATA 暫存器，保存 SD 卡發送回來的回應（一般這類命令只有 32 位的回應）；
- (5) 查詢 P_SD_INT_STATUS 暫存器的 CMDCOM 位，等待該位為 1，此時 SD 卡控制器的命令傳輸操作完成；
- (6) 查詢 P_SD_INT_STATUS 暫存器的 DATBUFEEMPTY 位，等待該位為 1，此時 SD 卡發送緩存區為空；
- (7) 向 P_SD_TX_DATA 暫存器寫入資料；
- (8) 跳至第 6 步，重複發送資料過程，直至將整個扇區的資料發送完畢；

(9) 查詢 P_SD_INT_STATUS 暫存器的 DATCOM 位，等待該位為 1，此時 SD 卡控制器完成資料接收工作。

參考代碼如下：

```
// 以發送 CMD24 寫單扇區命令為例
*p_SD_ARGUMENT_DATA = blockaddress; // 寫入命令參數，扇區位址
*p_SD_COMMAND_SETUP = C_SD_CMD_WITHDATA // 帶資料
| C_SD_DATA_TRANSFER // 資料發送
| C_SD_RSP_R1 // 回應類型為 R1
| C_SD_CMD_START // 立即發送
| 24; // SD 命令 24，CMD24
while((*P_SD_INT_STATUS & C_SD_CMDBUF_FULL) == 0); // 等待響應暫存器滿
response = *P_SD_RESPONSE_DATA; // 保存回應資料
while((*P_SD_INT_STATUS & C_SD_CMD_COMPLETE) == 0); // 等待命令傳輸完畢
for(i=0; i<512/4; i++) // 發送一個扇區 512 位元組
{
    while((*P_SD_INT_STATUS & C_SD_DATABUF_EMPTY) == 0); // 等待發送緩衝區空
    *P_SD_RX_DATA = BlockData[i]; // 以字為單位發送的資料
}
while((*P_SD_INT_STATUS & C_SD_DATA_COMPLETE) == 0); // 等待資料傳送完畢
```

4.15 TFT LCD 控制器

4.15.1 TFT LCD 概述

1. 發展現狀

LCD 是 Liquid Crystal Display 的縮寫，是一種顯示器件。當前 LCD 種類眾多 TN (Twisted Nematic)、STN(Super TN)、DSTN(Double STN)、CSTN(Color STN)、FSTN(Film STN)、UFB (CSTN) 及 TFT LCD。當前 LCD 發展迅速，應用在各種顯示場合，尤其以 TFT-LCD 發展更為迅猛。

TFT LCD 是 Thin Film Transistor-Liquid Crystal Display 的縮寫，即薄膜電晶體液晶顯示器。在驅動方式上，TFT LCD 與無源 TN-LCD、STN-LCD 的簡單矩陣不同，它在液晶顯示幕的每一個圖元上都設置有一個薄膜電晶體 (TFT)，可有效地克服非選通時的串擾，使顯示液晶屏的靜態特性與掃描線數無關，大大提高了影像品質 (尺寸、色彩)，控制起來也比較容易。

當前生產 TFT-LCD 的主要廠商有日本的夏普、韓國的三星、LG、臺灣的友達光電、奇美電子、中華映管、翰宇彩晶、廣輝電子等。

2. 結構

TFT LCD 的面板排布如圖 4-89所示：

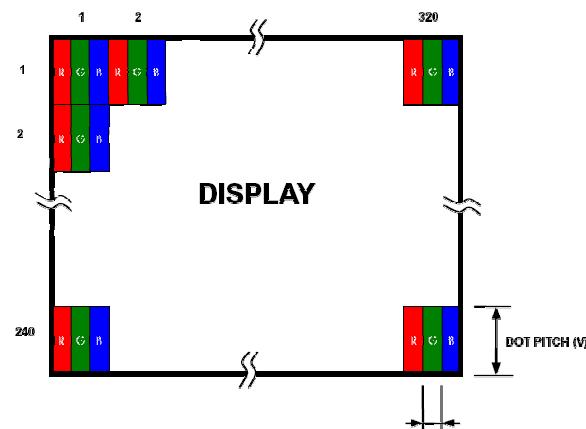


圖 4-89 TFT LCD 的面板排布

TFT LCD 的結構如圖 4-90 所示：主要由偏振片、濾色器基板、液晶、TFT 基板、偏振片、背光源組成。

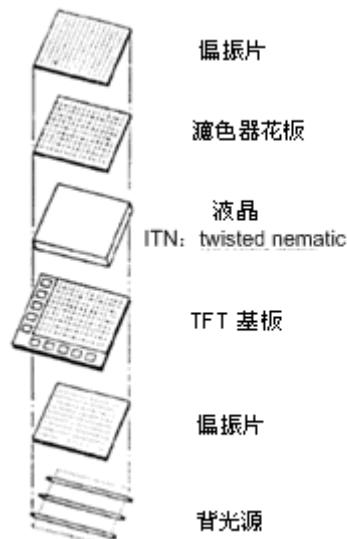


圖 4-90 TFT LCD 的結構

一個 TFT 單元的結構如圖 4-91 所示：

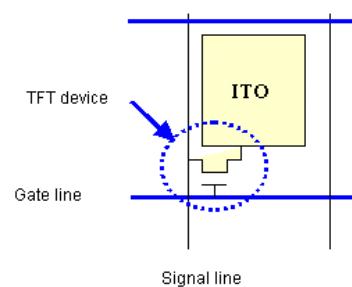


圖 4-91 TFT 的一個控制單元

3. 顯示原理

透過改變 R、G、B 信號線上的電壓，可以控制 LCD 的透光，使每一個圖元呈現不同的顏色。按照時序，控制整屏的所有圖元點就可以控制 LCD 顯示。

內置於 SPCE3200 的 TFT LCD 介面能夠支援多種輸入格式的 TFT 型 LCD 顯示幕的控制，譬如 DataEnable (DE)、Hsync/Vsync、16 位並行 RGB、8 位 delta RGB 以及 CCIR601/656 等輸入。LCD 顯示幕支援的解析度為 320 (水平圖元) × 240 (垂直圖元)，並支援 NTSC/PAL 顯示格式。由於同步信號的位置及寬度都是可配置的，故可以滿足各種規格的 TFT LCD 屏的要求。

4.15.2 特性

SPCE3200 的 TFT-LCD 控制器支援下列資料格式：並行 RGB 格式、串列 RGB 格式、串列 RGBDm 格式、CCIR-601/CCIR-656 格式。支援多種時鐘頻率，系統時鐘 (27MHz)、系統時鐘/2 (13.5MHz)、系統時鐘/4 (6.75MHz)、系統時鐘/8 (3.375MHz)。

4.15.3 插腳描述

SPCE3200 的 TFT-LCD 控制器相關 I/O 管腳：LCD_CLK (時鐘信號)、LCD_ACT (資料使能信號)、LCD_VS (垂直同步信號)、LCD_HS (水平同步信號)、LCD_Data[15:0] (資料匯流排)，具體參考下表：

表 4-174 TFT 管腳描述表

插腳名稱	插腳號	插腳屬性	插腳功能
LCD_CLK	148	O	LCD 的時鐘信號
LCD_VS	147	O	LCD 的垂直輸出信號
LCD_HS	146	O	LCD 的水平輸出信號
LCD_D0	144	O	LCD 的資料輸出匯流排
LCD_D1	143	O	
LCD_D2	142	O	
LCD_D3	141	O	
LCD_D4	140	O	
LCD_D5	139	O	
LCD_D6	138	O	
LCD_D7	137	O	
LCD_D8	136	O	
LCD_D9	135	O	
LCD_D10	134	O	

插腳名稱	插腳號	插腳屬性	插腳功能
LCD_D11	133	O	
LCD_D12	132	O	
LCD_D13	131	O	
LCD_D14	130	O	
LCD_D15	129	O	
LCD_ACT	145	O	LCD 的資料使能信號

4.15.4 暫存器描述

TFT LCD 控制器共有 32 個暫存器，如表 4-175 所示：透過對這 32 個暫存器的操作，即可透過 TFT LCD 控制器控制顯示。

表 4-175 TFT 控制器相關暫存器列表

暫存器名稱	助記符	位址
TFT 控制暫存器	P_TFT_MODE_CTRL	0x88040000
TFT 輸出資料格式暫存器	P_TFT_DATA_FMT	0x88040004
TFT 水平顯示掃描暫存器	P_TFT_HOR_ACT	0x88040008
TFT 水平空白前掃暫存器	P_TFT_HOR_FRONT	0x8804000C
TFT 水平空白後掃暫存器	P_TFT_HOR_BACK	0x88040010
TFT 水平同步掃描暫存器	P_TFT_HOR_SYNC	0x88040014
TFT 垂直顯示掃描暫存器	P_TFT_VER_ACT	0x88040018
TFT 垂直空白前掃暫存器	P_TFT_VER_FRONT	0x8804001C
TFT 垂直空白後掃暫存器	P_TFT_VER_BACK	0x88040020
TFT 垂直同步掃描暫存器	P_TFT_VER_SYNC	0x88040024
TFT 框資料格式暫存器 1	P_TFT_FRAME_FMT1	0x88040028
TFT 起始行暫存器	P_TFT_ROW_START	0x8804002C
TFT 起始列暫存器	P_TFT_COL_START	0x88040030
TFT 列寬度暫存器	P_TFT_COL_WIDTH	0x88040034
TFT 餘量寬度暫存器	P_TFT_DUMMY_WIDTH	0x88040038
TFT 狀態暫存器	P_TFT_BUFFER_STATUS	0x8804003C

暫存器名稱	助記符	位址
TFT 輸出資料順序暫存器	P_TFT_DATA_SEQ	0x88040040
TFT 中斷狀態暫存器	P_TFT_INT_STATUS	0x88040050
TFT 框資料格式暫存器 2	P_TFT_FRAME_FMT2	0x880400A0
LCD 時鐘配置暫存器	P_LCD_CLK_CONF	0x88210034
LCD 時鐘選擇暫存器	P_LCD_CLK_SEL	0x88210038
LCD 介面選擇暫存器	P_LCD_INTERFACE_SEL	0x88200000
LCD 顯示緩衝區起始位址暫存器 1	P_LCD_BUFFER_SA1	0x8807000C
LCD 顯示緩衝區起始位址暫存器 2	P_LCD_BUFFER_SA2	0x88070010
LCD 顯示緩衝區起始位址暫存器 3	P_LCD_BUFFER_SA3	0x88070014
LCD 顯示緩衝區選擇暫存器	P_LCD_BUFFER_SEL	0x88090024
TFT GPIO 輸出資料暫存器	P_TFT_GPIO_DATA	0x88200014
TFT GPIO 輸出使能暫存器	P_TFT_GPIO_OUTPUTEN	0x88200018
TFT GPIO 上拉暫存器	P_TFT_GPIO_PULLUP	0x8820001C
TFT GPIO 下拉暫存器	P_TFT_GPIO_PULLDOWN	0x88200020
TFT GPIO 輸入資料暫存器	P_TFT_GPIO_INPUT	0x88200064
TFT GPIO 外部中斷暫存器	P_TFT_GPIO_INT	0x88200080

如果使用 TFT LCD 介面複用為 GPIO 功能，可以對下面的暫存器操作。暫存器的各位對應插腳關係如表 4-176：

表 4-176 TFT 介面複用為 GPIO 插腳與暫存器位對應關係

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
LCD_CLK	148	bit[0]	可以
LCD_VS	147	bit[1]	可以
LCD_HS	146	bit[2]	可以
LCD_D0	144	bit[3]	可以
LCD_D1	143	bit[4]	不可以
LCD_D2	142	bit[5]	不可以
LCD_D3	141	bit[6]	不可以

插腳名稱	插腳號	控制暫存器位	是否可以作為外部中斷
LCD_D4	140	bit[7]	不可以
LCD_D5	139	bit[8]	不可以
LCD_D6	138	bit[9]	不可以
LCD_D7	137	bit[10]	不可以
LCD_D8	136	bit[11]	不可以
LCD_D9	135	bit[12]	不可以
LCD_D10	134	bit[13]	不可以
LCD_D11	133	bit[14]	不可以
LCD_D12	132	bit[15]	不可以
LCD_D13	131	bit[16]	不可以
LCD_D14	130	bit[17]	不可以
LCD_D15	129	bit[18]	不可以
LCD_ACT	145	bit[19]	不可以

■ TFT GPIO 輸出資料暫存器：**P_TFT_GPIO_DATA(0x88200014)**

TFT GPIO 輸出資料暫存器設置 TFT 控制器介面複用為 GPIO 時的輸出資料。

表 4-177 P_TFT_GPIO_DATA(0x88200014)

位	b31~b20	b19~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TFT_GPIO_O

TFT_GPIO_O	b19~b0	TFT_GPIO_O:TFT 介面作為 GPIO 使用時的輸出資料
------------	--------	-----------------------------------

■ TFT GPIO 輸出使能暫存器：**P_TFT_GPIO_OUTPUTEN(0x88200018)**

TFT GPIO 輸出使能暫存器設置 TFT 介面複用為 GPIO 時插腳輸出使能。

表 4-178 P_TFT_GPIO_OUTPUTEN(0x88200018)

位	b31~b20	b19~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TFT_GPIO_OE

TFT_GPIO_OE	b19~b0	TFT_GPIO_OE:TFT 介面作為 GPIO 使用時的輸出使能位： 0 : 禁止輸出 1 : 使能輸出
-------------	--------	--

■ TFT GPIO 上拉暫存器：P_TFT_GPIO_PULLUP(0x8820001C)

TFT GPIO 上拉暫存器設置 TFT 介面複用為 GPIO 時的上拉電阻輸入使能。

表 4-179 P_TFT_GPIO_PULLUP(0x8820001C)

位	b31~b20	b19~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TFT_GPIO_PU

TFT_GPIO_PU	b19~b0	TFT_GPIO_PU:TFT 介面作為 GPIO 使用時的上拉電阻輸入使能位： 0 : 使能上拉電阻輸入 1 : 禁止上拉電阻輸入
-------------	--------	--

■ TFT GPIO 下拉暫存器：P_TFT_GPIO_PULLDOWN(0x88200020)

TFT GPIO 下拉暫存器設置 TFT 介面複用為 GPIO 時的下拉電阻輸入使能。

表 4-180 P_TFT_GPIO_PULLDOWN(0x88200020)

位	b31~b20	b19~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TFT_GPIO_PD

TFT_GPIO_PD b19~b0

TFT_GPIO_PD : TFT 介面作為 GPIO 使用時的下拉電阻輸入使能位：

0 : 禁止下拉電阻輸入

1 : 使能下拉電阻輸入

■ TFT GPIO 輸入資料暫存器：P_TFT_GPIO_INPUT(0x88200064)

TFT GPIO 輸入資料暫存器保存 TFT 介面複用為 GPIO 時的外部輸入資料。

表 4-181 P_TFT_GPIO_INPUT(0x88200064)

位	b31~b20	b19~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	TFT_INPUT

TFT_INPUT

b19~b0

TFT 介面作為 GPIO 使用時的輸入資料

■ TFT GPIO 外部中斷暫存器：P_TFT_GPIO_INT(0x88200080)

TFT GPIO 外部中斷暫存器可進行中斷使能、中斷觸發沿設置、中斷旗標清除等操作。

表 4-182 P_TFT_GPIO_INT(0x88200080)

位	b31~b28	b27~b24	b23~b20	b19~b16	b15~b12	b11~b8	b7~b4	b3~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	0	-	0	-	0	-	0
名稱	-	TFT_FI	-	TFT_RI	-	TFT_FIEN	-	TFT RIEN

 TFT_FI b27~b24 TFT_FI : TFT 介面複用為 GPIO 下降沿中斷旗標位：

讀 0：沒有發生下降沿中斷

讀 1：發生下降沿中斷

寫 0：無意義

寫 1：清除中斷旗標

 TFT_RI b19~b16 TFT_RI : TFT 介面複用為 GPIO 上升沿中斷旗標位：

讀 0：沒有發生上升沿中斷

讀 1：發生上升沿中斷

寫 0：無意義

寫 1：清除中斷旗標

 TFT_FIEN b11~b8 TFT_FIEN : TFT 介面複用為 GPIO 下降沿中斷使能位：

0：禁止下降沿中斷

1：使能下降沿中斷

 TFT_RIEN b3~b0 TFT_RIEN : TFT 介面複用為 GPIO 上升沿中斷使能位：

0：禁止上升沿中斷

1：使能上升沿中斷

 作為 TFT 控制器介面需要設置以下暫存器：

■ LCD 時鐘配置暫存器：**P_LCD_CLK_CONF(0x88210034)**

LCD 時鐘配置暫存器用於使能或禁止 TFT 控制器模組時鐘，或軟重設 TFT 控制器模組。

表 4-183 P_LCD_CLK_CONF(0x88210034)

位	b31~b2	b1	b0
讀/寫	-	R/W	R/W
預設值	-	1	0
名稱	-	LCD_RST	LCD_STOP

LCD_RST	b1	LCD 模組時鐘重設位： 0 : LCD 模組時鐘重設 1 : LCD 模組時鐘不重設
LCD_STOP	b0	LCD 模組時鐘使能位： 0 : LCD 模組時鐘停止 1 : LCD 模組時鐘使能

■ LCD 時鐘選擇暫存器 : P_LCD_CLK_SEL(0x88210038)

為滿足不同 TFT LCD 的時鐘頻率，SPCE3200 控制器提供用戶選擇配置時鐘頻率，TFT LCD 預設時鐘使用 APB 匯流排時鐘 27MHz。如果用戶選擇 PLLU 時鐘作為頻率源，需要先透過 P_CLK_PLLAU_CONF 暫存器使能 PLLU。

表 4-184 P_LCD_CLK_SEL(0x88210038)

位	b31~b6	b5~b3	b2~b0
讀/寫	-	R/W	R/W
預設值	-	0	0
名稱	-	TFT_CLK	STN_CLK

TFT_CLK	b5~b3	選擇 TFT LCD 時鐘頻率： 000 : 27MHz 001 : USB PLL 輸出頻率 2 分頻 010 : USB PLL 輸出頻率 3 分頻 011 : USB PLL 輸出頻率 4 分頻 100 : USB PLL 輸出頻率 6 分頻 101 : USB PLL 輸出頻率 8 分頻
STN_CLK	b2~b0	選擇 STN LCD 時鐘頻率： 000 : 視頻 PLL 輸出頻率 001 : 視頻 PLL 輸出頻率 2 分頻 010 : 視頻 PLL 輸出頻率 3 分頻 011 : 視頻 PLL 輸出頻率 4 分頻

 100：視頻 PLL 輸出頻率 6 分頻

 101：視頻 PLL 輸出頻率 8 分頻

■ LCD 介面選擇暫存器：**P_LCD_INTERFACE_SEL(0x88200000)**

LCD 介面選擇暫存器設置控制器作為 TFT 介面、STN 介面等。

表 4-185 P_LCD_INTERFACE_SEL(0x88200000)

位	b31~b2	b1~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	SW_LCD

 SW_LCD

b1~b0

SW_LCD : LCD 埠配置

00：不做為 LCD 介面

01：TFT_AUO 介面

10：TFT_TOPPOLY 介面

11：STN 介面

■ TFT 控制暫存器：**P_TFT_MODE_CTRL(0x88040000)**

TFT 控制暫存器選擇 TFT 輸出時鐘、影像水平、垂直的縮放比例、使能 TFT LCD 控制器等。

表 4-186 P_TFT_MODE_CTRL(0x88040000)

位	b31~b25	b24	b23~b12	b11~b10	b9~b8	b7~b2	b1~b0
讀/寫	-	R/W	-	R/W	R/W	-	R/W
預設值	-	0	-	0	0	-	0
名稱	-	TFT_EN	-	VER_SCALING	HOR_SCALING	-	TFT_CLK_SEL

 TFT_EN

b24

TFT_EN : TFT-LCD 使能位：

0：禁止

1：使能

VER_SCALING	b11~b10	VER_SCALING : 影像垂直縮放位： 00 : 不縮放 01 : 顯示影像垂直方向放大一倍 10 : 顯示影像垂直方向縮小一倍 11 : 不縮放
HOR_SCALING	b9~b8	HOR_SCALING : 影像水平縮放位： 00 : 不縮放 01 : 顯示影像水平方向放大一倍 10 : 顯示影像水平方向縮小一倍 11 : 不縮放
TFT_CLK_SEL	b1~b0	TFT_CLK_SEL : TFT-LCD 輸出時鐘選擇位： 0 : 系統時鐘 (27MHz) 01 : 系統時鐘/2 (13.5MHz) 10 : 系統時鐘/4 (6.75MHz) 11 : 系統時鐘/8 (3.875MHz)

■ TFT 中斷狀態暫存器 : P_TFT_INT_STATUS(0x88040050)

TFT 中斷狀態暫存器控制垂直空白中斷允許及清除垂直空白中斷旗標。

表 4-187 P_TFT_INT_STATUS(0x88040050)

位	b31~b25	b24	b23~b17	b16	b15~b0
讀/寫	-	R/W	-	R/W	-
預設值	-	0	-	0	-
名稱	-	INT_EN	-	VBLK_INT	-

INT_EN	b24	INT_EN : 垂直空白中斷使能位： 0 : 禁止 1 : 使能
--------	-----	---

VBLK_INT	b16	VBLK_INT : 垂直空白中斷旗標位： 讀 0 : 沒有發生垂直空白中斷 讀 1 : 發生垂直空白中斷 寫 0 : 無意義 寫 1 : 清除中斷旗標
----------	-----	--

■ TFT 狀態暫存器 : P_TFT_BUFFER_STATUS(0x8804003C)

TFT 狀態暫存器反映 TFT 緩存區與 TFT 控制器的狀態。

表 4-188 P_TFT_BUFFER_STATUS(0x8804003C)

位	b31~b25	b24	b23~b17	b16	b15~b0
讀/寫	-	R/W	-	R/W	-
預設值	-	0	-	0	-
名稱	-	BUF_ERR	-	TFT_FINISH	-

BUF_ERR	b24	BUF_ERR : TFT-LCD 緩存區下溢旗標，當 LCD 緩存區的讀取速度慢於寫入速度時，TFT-LCD 控制器會將這一位置 1 讀 0 : 未發生下溢 讀 1 : 發生下溢 寫 0 : 無有意義 寫 1 : 清除旗標
---------	-----	--

TFT_FINISH	b16	TFT_FINISH : 這一位置 1 表示 TFT 控制器完成了內部操作，用戶此時可以關閉 TFT 時鐘
------------	-----	---

TFT 顯示緩存區相關暫存器：

- LCD 顯示緩衝區起始位址暫存器 1 : P_LCD_BUFFER_SA1(0x8807000C)
- LCD 顯示緩衝區起始位址暫存器 2 : P_LCD_BUFFER_SA2(0x88070010)
- LCD 顯示緩衝區起始位址暫存器 3 : P_LCD_BUFFER_SA3(0x88070014)

這 3 個暫存器將存儲 LCD 顯示緩衝區的起始位址，注意起始位址的低 8 位必須為 0。

表 4-189 P_LCD_BUFFER_SA1/2/3

位	b31~b24	b23~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	PPU_LCD_ADR1/2/3

PPU_LCD_ADRx

b23~b0

LCD 框緩衝區 x (1、2、3) 的起始位址

b7~b0 必須為 0

■ LCD 顯示緩衝區選擇暫存器：P_LCD_BUFFER_SEL(0x88090024)

LCD 顯示緩衝區選擇暫存器選擇當前具體使用哪一個框緩衝區。

表 4-190 P_LCD_BUFFER_SEL(0x88090024)

位	b31~b2	b1~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	LCD_PTR

LCD_PTR

b1~b0

LCD_PTR: 寫入該暫存器是將當前框緩存區指向 LCD 框緩存區 0~2 中之一；讀出該暫存器是為了獲知當前框緩存區指向 LCD 框緩存區 0~2 中哪一個

0 : LCD frame buffer0

1 : LCD frame buffer1

2 : LCD frame buffer2

TFT 資料格式相關暫存器：

■ TFT 輸出資料格式暫存器：P_TFT_DATA_FMT(0x88040004)

TFT 輸出資料格式暫存器選擇輸出資料格式是並行 RGB 格式、串列 RGB 格式、串列 RGBDM 格式、CCIR601 格式還是 CCIR656 格式，在選擇 CCIR656 格式後，選擇輸出有效資料是 640 個圖元點，還是 720 個圖元點。

表 4-191 P_TFT_DATA_FMT(0x88040004)

位	b31~b9	b8	b7~b3	b2~b0
讀/寫	-	R/W	-	R/W
預設值	-	0	-	0
名稱	-	CCIR656_M	-	TFT_FMT

CCIR656_M

b8

CCIR656_M：標準 CCIR656 格式

0：控制器會輸出 640 個有效圖元的顯示

 1：控制器會輸出 720 個有效圖元的顯示，但在影像兩端有
80 個圖元的黑屏

TFT_FMT

b2~b0

TFT_FMT : TFT-LCD 輸出資料格式

000：並行 RGB 格式

001：串列 RGB 格式

010：串列 RGBDm 格式

011：CCIR601 格式

100：CCIR656 格式

■ TFT 框資料格式暫存器 1 : P_TFT_FRAME_FMT1(0x88040028)

TFT 框資料格式暫存器選擇框緩衝區資料格式。

表 4-192 P_TFT_FRAME_FMT1(0x88040028)

位	b31~b2	b1~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	FB_FMT

FB_FMT	b1~b0	FB_FMT : 框緩衝區資料格式
		00 : RGB565 格式 (小端模式)
		01 : RGB1555 格式 (大端模式)
		10 : YUYV 格式 (小端模式)
		11 : 4Y4U4Y4V 格式 (小端模式)

■ TFT 輸出資料順序暫存器 : P_TFT_DATA_SEQ(0x88040040)

TFT 輸出資料順序暫存器選擇對外輸出資料的順序。

表 4-193 P_TFT_DATA_SEQ(0x88040040)

位	b31~b25	b24	b23~b19	b18~b16	b15~b11	b10~b8	b7~b3	b2~b0
讀/寫	-	R/W	-	R/W	-	R/W	-	R/W
預設值	-	0	-	0	-	0	-	0
名稱	-	YUV_FMT	-	YUV_SEQ	-	RGB_OLESEQ	-	RGB_ELSSEQ

YUV_FMT	b24	YUV_FMT : 輸出資料格式選擇
		0 : YCbCr 格式
		1 : YUV 格式

YUV_SEQ	b18~b16	YUV_SEQ : YUV 資料輸出序列選擇 (針對 CCIR601 及 CCIR656 輸入格式)
		000 : Y0U0Y1V0/Y0Cb0Y1Cr0
		001 : Y0V0Y1U0/Y0Cr0Y1Cb0
		010 : U0Y0V0Y1/Cb0Y0Cr0Y1
		011 : V0Y0U0Y1/Cr0Y0Cb0Y1
		100 : Y1U0Y0V0/Y1Cb0Y0Cr0 (為測試保留)
		101 : Y1V0Y0U0/Y1Cr0Y0Cb0 (為測試保留)
		110 : U0Y1V0Y0/Cb0Y1Cr0Y0 (為測試保留)
		111 : V0Y1U0Y0/Cr0Y1Cb0Y0 (為測試保留)

RGB_OLSEQ	b10~b8	RGB_OLSEQ : RGB 資料輸出序列（針對串列 RGB 和串列 RGBDm 輸入資料的奇數行） 000 : RGB (RGBDm) 001 : GBR (GBRDm) 010 : BRG (BRGDm) 011 : RBG (RBGDm) 100 : BGR (BGRDm) 101 : GRB (GRBDm)
RGB_ElseQ	b2~b0	RGB_ElseQ : RGB 資料輸出序列（針對串列 RGB 和串列 RGBDm 輸入資料的偶數行） 000 : RGB (RGBDm) 001 : GBR (GBRDm) 010 : BRG (BRGDm) 011 : RBG (RBGDm) 100 : BGR (BGRDm) 101 : GRB (GRBDm)

■ TFT 框資料格式暫存器 2 : P_TFT_FRAME_FMT2(0x880400A0)

TFT 框資料格式暫存器選擇框緩衝區資料格式。

表 4-194 P_TFT_FRAME_FMT2(0x880400A0)

位	b31~b1	b0
讀/寫	-	R/W
預設值	-	0
名稱	-	FB_YCbCr

FB_YCbCr	b0	FB_YCbCr : 框緩存區資料格式選擇 0 : YUV 格式 1 : YCbCr 格式
----------	----	---

TFT 行設置相關暫存器：

■ TFT 水平顯示掃描暫存器：P_TFT_HOR_ACT(0x88040008)

根據 TFT LCD 顯示器的資料手冊，填寫水平顯示時間。

表 4-195 P_TFT_HOR_ACT(0x88040008)

位	b31~b12	b11~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	HOR_ACT

HOR_ACT

b11~b0

HOR_ACT：水平活動區

單位：系統時鐘週期。

注意，時鐘週期是指 27MHz

■ TFT 水平空白前掃暫存器：P_TFT_HOR_FRONT(0x8804000C)

根據 TFT LCD 顯示器的資料手冊，填寫水平空白前掃時間。

表 4-196 P_TFT_HOR_FRONT(0x8804000C)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	HOR_FBLK

HOR_FBLK

b9~b0

HOR_FBLK：前水平空白區

單位：系統時鐘週期

■ TFT 水平空白後掃暫存器：P_TFT_HOR_BACK(0x88040010)

根據 TFT LCD 顯示器的資料手冊，填寫水平空白後掃時間。

表 4-197 P_TFT_HOR_BACK(0x88040010)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	HOR_BBLK

HOR_BBLK

b9~b0

HOR_BBLK：後水平空白區

單位：系統時鐘週期

■ TFT 水平同步掃描暫存器：P_TFT_HOR_SYNC(0x88040014)

根據 TFT LCD 顯示器的資料手冊，填寫水平同步掃描時間。另外該暫存器還可以選擇水平同步脈衝的極性。

表 4-198 P_TFT_HOR_SYNC(0x88040014)

位	b31~b25	b24	b23~b8	b7~b0
讀/寫	-	R/W	R/W	R/W
預設值	-	0	0	0
名稱	-	HS_P	-	HOR_SYNCW

HS_P

b24

HS_P：水平同步脈衝的極性

0：負極性

1：正極性

HOR_SYNCW

b7~b0

HOR_SYNCW：水平同步脈衝寬度

單位：系統時鐘週期

TFT 場設置相關暫存器：

■ TFT 垂直顯示掃描暫存器：P_TFT_VER_ACT(0x88040018)

根據 TFT LCD 顯示器的資料手冊，填寫垂直顯示掃描時間。

表 4-199 P_TFT_VER_ACT(0x88040018)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	VER_ACT

VER_ACT

b9~b0

VER_ACT：垂直活動

單位：行

■ TFT 垂直空白前掃暫存器：P_TFT_VER_FRONT(0x8804001C)

根據 TFT LCD 顯示器的資料手冊，填寫垂直空白前掃時間。

表 4-200 P_TFT_VER_FRONT(0x8804001C)

位	b31~b8	b7~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	VER_FBLK

VER_FBLK

b7~b0

VER_FBLK：前垂直空白區

單位：行

■ TFT 垂直空白後掃暫存器：P_TFT_VER_BACK(0x88040020)

根據 TFT LCD 顯示器的資料手冊，填寫垂直空白後掃時間。

表 4-201 P_TFT_VER_BACK(0x88040020)

位	b31~b8	b7~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	VER_BBLK

VER_BBLK

b7~b0

VER_BBLK：後垂直空白區

單位：行

■ TFT 垂直同步掃描暫存器：P_TFT_VER_SYNC(0x88040024)

根據 TFT LCD 顯示器的資料手冊，填寫垂直同步掃描時間。另外該暫存器還可以選擇垂直同步脈衝的極性。

表 4-202 P_TFT_VER_SYNC(0x88040024)

位	b31~b25	b24	b23~b5	b4~b0
讀/寫	-	R/W	R/W	R/W
預設值	-	0	0	0
名稱	-	VS_P	-	VER_SYNCW

VS_P

b24

VS_P：垂直同步脈衝的極性

0：負極性

1：正極性

VER_SYNCW

b4~b0

VER_SYNCW：垂直同步脈衝寬度

設定範圍：1~31

TFT 顯示位置相關暫存器：

■ TFT 起始行暫存器：P_TFT_ROW_START(0x8804002C)

TFT 起始行暫存器設置起始行開始位置，參考圖 4-92。

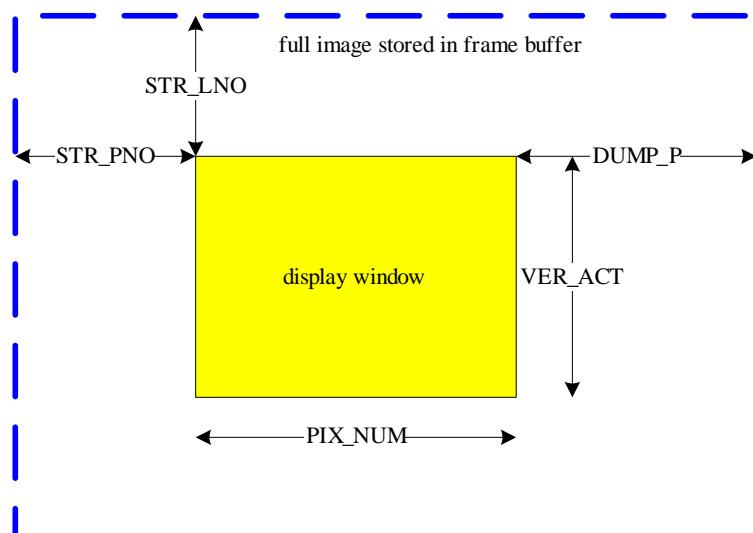


圖 4-92 TFT LCD 顯示結構圖

表 4-203 P_TFT_ROW_START(0x8804002C)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	STR_LNO

STR_LNO

b9~b0

STN_LNO：框緩衝區起始行的行數

單位：行

■ TFT 起始列暫存器：P_TFT_COL_SRART(0x88040030)

TFT 起始列暫存器設置起始列開始位置，參考表 4-204。

表 4-204 P_TFT_COL_START(0x88040030)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	STR_PNO

STR_PNO	b9~b0	STN_PNO : 框緩衝區起始圖元數，須為 16 的整數倍 單位：圖元
---------	-------	---

■ TFT 列寬度暫存器：**P_TFT_COL_WIDTH(0x88040034)**

TFT 列寬度暫存器設置 TFT 螢幕列顯示寬度，參考表 4-205。

表 4-205 P_TFT_COL_WIDTH(0x88040034)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	PIX_NUM

PIX_NUM	b9~b0	PIX_NUM : 框緩存區內一行的影像圖元數，須為 16 的整數倍 單位：圖元（16 位）
---------	-------	---

■ TFT 餘量寬度暫存器：**P_TFT_DUMMY_WIDTH(0x88040038)**

TFT 餘量寬度暫存器設置 TFT 顯示螢幕與框緩衝區之間寬度。

表 4-206 P_TFT_DUMMY_WIDTH(0x88040038)

位	b31~b10	b9~b0
讀/寫	-	R/W
預設值	-	0
名稱	-	DUMP_PIX

DUMP_PIX	b9~b0	DUMP_PIX : 框緩存區內一行的影像圖元數，須為 16 的整數倍 單位：圖元（16 位）
----------	-------	--

4.15.5 基本操作

使用 SPCE3200 TFT 控制器點亮台盛公司的 TS35ND5B01 屏，完成初始化程式。

```
#include "SPCE3200_Register.h"
#include "SPCE3200_Constant.h"
```

```
void LCD_Buffer_Setting(int FB_ADDR0, int FB_ADDR1, int FB_ADDR2)
{
    *P_LCD_BUFFER_SA1 = FB_ADDR0;
    *P_LCD_BUFFER_SA2 = FB_ADDR1;
    *P_LCD_BUFFER_SA3 = FB_ADDR2;
}

void Init_TFTLcd(void)
{
    *P_LCD_CLK_CONF = C_LCD_RST_DIS
                      | C_LCD_CLK_EN;           // LCD 模式時鐘使能
    *P_LCD_INTERFACE_SEL = C_LCD_PORT_AUO;      // 管腳複用，選擇 TFT_AUO 模式

    // 設置 TFT-LCD 資料模式
    *P_TFT_FRAME_FMT1 = C_TFT_BUF_4Y4U4Y4V;      // 資料框格式選擇 4Y4U4Y4V
    *P_TFT_DATA_FMT = C_TFT_PARALLEL_RGB;          // TFT-LCD 管腳輸出資料格式為並行
                                                    // RGB 格式
    *P_TFT_DATA_SEQ = C_TFT_OUTPUT_YCBCR;          // TFT-LCD 輸出資料格式為 YCbCr
    *P_TFT_FRAME_FMT2 = C_TFT_BUF_YCBCR;            // TFT-LCD 框資料格式為 YCbCr

    // 設置 TFT-LCD 的行信號
    *P_TFT_HOR_ACT = 1280;
    *P_TFT_HOR_FRONT = 80;
    *P_TFT_HOR_BACK = 152;
    *P_TFT_HOR_SYNC = 120;

    // 設置 TFT-LCD 的場信號
    *P_TFT_VER_ACT = 240;
    *P_TFT_VER_FRONT = 4;
    *P_TFT_VER_BACK = 15;
    *P_TFT_VER_SYNC = 3;

    // 設置 TFT-LCD 顯示的起始位置
    *P_TFT_ROW_START = 0;
    *P_TFT_COL_START = 0;
    *P_TFT_COL_WIDTH = 320;
    *P_TFT_DUMMY_WIDTH = 0;

    // 設置 TFT-LCD 的控制暫存器
    *P_TFT_INT_SEL = ~C_TFT_INT_EN;                // 中斷禁止
    // 使能 TFT 模組，影像不放大，選擇 6.75MHz 時鐘
    *P_TFT_CTRL_SEL = C_TFT_CTRL_EN | C_TFT_CLK_27MDIV4;
}
```

5 SPCE3200 開發系統介紹

通常利用 SPCE3200 晶片進行開發時，硬體上需要一套 SPCE3200 開發板作為開發平臺，需要 PC 機進行軟體程式開發，同時需要一套下載除錯程式的工具（本書中特指 SJPROBE）把 PC 機上編寫好的程式下載到開發板上；軟體上需要凌陽科技 S+core 集成開發環境（S+core IDE）支援程式的開發。

目前 SPCE3200 系統開發板有兩種：SPCE3200 實驗儀和 SPCE3200 實驗箱。本書只介紹實驗儀。

5.1.1 SPCE3200 實驗儀

SPCE3200 實驗儀是凌陽科技推出的 32 位嵌入式開發板，是為大專院校作為嵌入式作業系統、嵌入式硬體設計教學設備，供學生學習和研究量身打造的，同時也適合研究開發人員和電子愛好者進行評估和開發。該開發板設計基於開發考慮，配置簡單，佈局簡潔明瞭，週邊設備齊全，介面標準，擴展方便。採用四層板設計，線路穩定。SPCE3200 晶片是開發板上的核心器件，採用凌陽自主設計的 S+core7 內核，並且集成了 MPEG4 硬體編碼譯碼以及 CMOS 感測器 / TV 譯碼介面。該晶片具備低功耗、高性能的特性，可適用於 PDA、便攜媒體播放器以及機器人等設備與終端。

5.1.2 功能特點

SPCE3200 實驗儀具有板上資源豐富、介面方便等特點，具體如下：

- 以 SPCE3200 晶片為核心，SPCE3200 採用 Sunplus S+core7® 處理器，頻率高達 162MHz
- 提供 DRAM 介面：可以透過 DRAM 介面存取外部 ROM 或 NOR Flash，來實現 boot 程式或應用程式的存儲
- 128M bit SDRAM
- 64M bit NOR Flash
- 提供 Nand Flash 介面：支援 64M Byte Nand Flash
- 提供 10M 乙太網介面。採用 10Mbps 低功耗嵌入式專用乙太網晶片 ENC28J60，介面為標準 RJ45 插座，集成網路變壓器，安全可靠
- 提供 UART 介面及 JOY STICK 介面
- 提供 USB 1.1 主從通訊介面。透過跳線選擇 USB 介面作為主機介面或者設備介面
- 提供 SD 卡介面
- 提供 I²C 介面。支援 I²C (主模式)，有 3 種傳輸類型：8 位、16 位和 8 位×n
- 提供 SPI 主從通訊介面
- 提供 SIO 介面
- 提供 LCD 介面。支援多種 TFT LCD 的輸入格式，該介面支援 320(H) × 240(V) 的解析度以及 NTSC/PAL 制式；同步信號的寬度及位置是可以配置的，以適應不同規格的 TFT LCD 板。支援 STN LCD
- 提供四線觸摸屏介面

- 提供音視頻介面：既支援標準 TV 介面，也支援類似耳機、音響等設備的輸出
- 提供硬體 MPEG-4/JPEG 編碼譯碼。MPEG-4 框率(frame rate)：CIF 模式下，高達 30 框/秒
- 提供 CSI I/F 攝像頭介面（凌陽 CMOS 影像感測器介面），支援 CCIR-601/656 CMOS 影像感測器
- 提供通用 A/D 介面和 MIC 輸入和 Line in 輸入介面：既支援板上提供的 MIC 輸入語音，又支援具有 Line in 介面的 MIC 輸入語音
- 提供 GPS/GPRS 介面
- 提供 IPOD KEY 介面
- 提供 SJTAG 介面，可直接與凌陽科技提供的 SJ Probe 連接進行程式燒錄和線上除錯
- 提供三個按鍵
- 提供三個 LED

5.1.3 硬體原理

1. 硬體佈局

SPCE3200 實驗儀的佈局結構如圖 5-1：SPCE3200 及其周邊電路分佈在實驗儀的中央，作為實驗儀的控制核心；實驗儀的左端為電源電路，為整個實驗儀提供工作電壓；在 SPCE3200 的四周，分佈了 SDRAM、Nor Flash、Nand Flash 等，作為 SPCE3200 的片外存儲設備，實驗儀上有兩片 64M bit 的 SDRAM 並聯，構成 128M bit；在實驗儀的四個邊上，分佈了乙太網、UART、USB、SD 卡、音視頻等介面，可以方便地與週邊各功能設備連接。

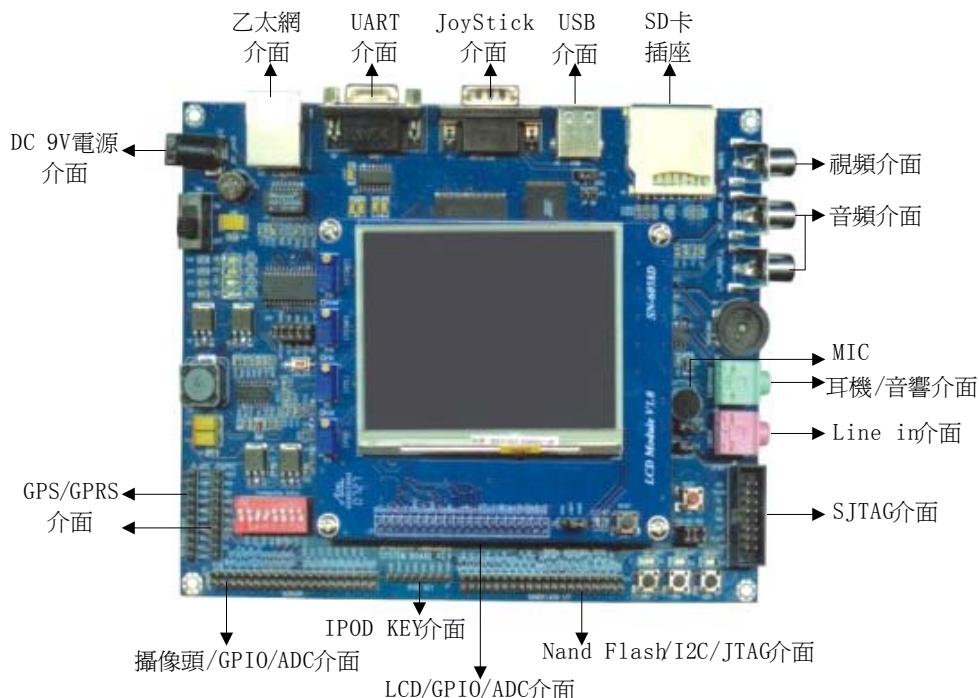


圖 5-1 SPCE3200 實驗儀佈局結構圖

為了使得 SPCE3200 實驗儀各模組能更好地協調工作，在實驗儀上設計了 8 組跳線和一個配

置開關，如表 5-1。

表 5-1 SPCE3200 實驗儀跳線及配置開關功能列表

跳線名稱	跳線功能	跳線連接	功能說明
JP1	選擇乙太網模組工作介面	NF_ALE 接 CLK	NF_ALE 作為具有 SPI 介面的乙太網控制晶片時鐘信號
		NF_D5 接 TXD	NF_D5 作為具有 SPI 介面的乙太網控制晶片資料發送插腳
		NF_D4 接 RXD	NF_D4 作為具有 SPI 介面的乙太網控制晶片資料接收插腳
		SPI_CSN 連接 CSN	SPI_CSN 作為具有 SPI 介面的乙太網控制晶片片選信號
		GND 接 LANCLKOUT	乙太網控制晶片的 LANCLKOUT 插腳接地
		3.3V 接 3.3V	-
JP2	選擇 LED 燈工作介面	IOB0 與右邊介面短接	IOB0 作為 LED 燈 D11 的控制埠
		IOB1 與右邊介面短接	IOB0 作為 LED 燈 D12 的控制埠
		IOB2 與右邊介面短接	IOB0 作為 LED 燈 D13 的控制埠
JP3	選擇 MIC 輸入或者 Line in 輸入	MIC 與中間介面短接	選擇 MIC 輸入方式
		LINEIN 與中間介面短接	選擇 Line in 輸入方式
JP4	外部 Nor Flash 的片選選擇	FLASH_CSN 連接 VDD33	外部 Nor Flash 的片選直接與 3.3V 電源連接
		FLASH_CSN 連接 ROMCSN	外部 Nor Flash 與 SPCE3200 晶片的 ROMCSN 連接
JP5	外部 SRAM 的片選選擇	SRAM_CSN 連接 VDD33	外部 SRAM 的片選直接與 3.3V 電源連接
		SRAM_CSN 連接 ROMCSN	外部 SRAM 與 SPCE3200 晶片的 ROMCSN 連接
JP6	USB	DP	DP 介面
		DM	DM 介面
JP7	USB 主從選擇	USB_DET 接 VDD5	SPCE3200 作為 USB 主機

跳線名稱	跳線功能	跳線連接	功能說明
		USB_DET 懸浮	SPCE3200 作為 USB 設備
JP8	ADC 的外部參考電壓選擇	ADC_ADVRT 與 AV33_ADC 連接	選擇 ADC 的外部參考電壓為 3.3V
Configuration Switch	JTAG、UART 及乙太網配置	CFG0	閉合：禁止 ICE 斷開：使能 ICE
		CFG2	閉合：從內部 ROM 啓動 斷開：從外部 ROM (Nor Flash、Nand Flash) 啓動
		IOA0	閉合：IOA0 作為乙太網控制晶片 INT 的控制插腳 斷開：IOA0 作為通用 I/O 使用
		IOA1	閉合：IOA1 作為乙太網控制晶片 WOL 的控制插腳 斷開：IOA1 作為通用 I/O 使用
		UARTTX	閉合：UART 介面可發送資料 斷開：UART 介面不可發送資料
		UARTRX	閉合：UART 介面可接收資料 斷開：UART 介面不可接收資料
		CFG1RST	兩者都閉合：CFG1RST 與 ICERST 連通
		ICERST	其他：CFG1RST 與 ICERST 不連通

2. 使用說明

SPCE3200 實驗儀為用戶提供了方便的介面，使用時，只需要用相應的連接線和外部設備連接即可。

SPCE3200 實驗儀透過除錯器和 PC 機相連，可以完成程式的下載和線上除錯，如圖 5-2 所示：用 20pin 的排線連接 SPCE3200 實驗儀和 SJPROBE，再透過平行埠線連接 SJPROBE 與 PC 機。SPCE3200 實驗儀的所有跳線和撥碼開關都可按照用戶自己的需求進行設置。

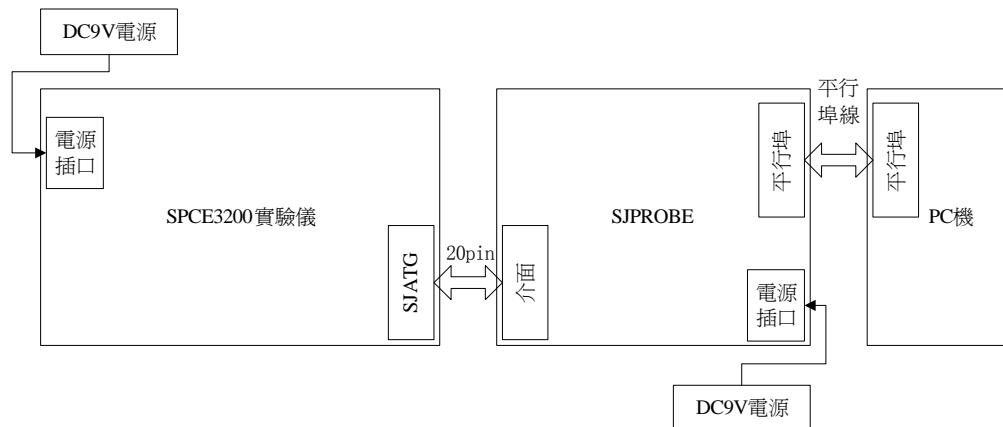


圖 5-2 SPC3200 實驗儀開發硬體連接

5.2 S+core IDE 集成開發環境

S+core IDE 是凌陽公司針對 SPC3200 開發平臺推出的一款功能強大的集成開發環境，支援工作區(Workspace)的多工程管理模式，支援 GDB 除錯方式，為用戶利用 SPC3200 開發提供了強有力的支援。

S+core IDE 可以安裝在 Windows98®、Windows2000® 及 WindowsXP® 作業系統上。圖 5-3 則展示了 S+core IDE 的用戶介面。該用戶介面主要包括：菜單欄、工具欄、Workspace 視窗、代碼編輯區以及 Output 視窗。其中，菜單欄為用戶提供了編輯、除錯工程常用的命令選項；工具欄裏除了 Standard 的工具外，還提供了一些常用的除錯工具；Workspace 視窗顯示工程資訊，包括工程包含的源碼檔案、資源檔案、全局函數等資訊；用戶可以在編輯區編寫程式；透過 Output 視窗可以看到工程的編譯、鏈接、查詢變數或者函數等資訊。

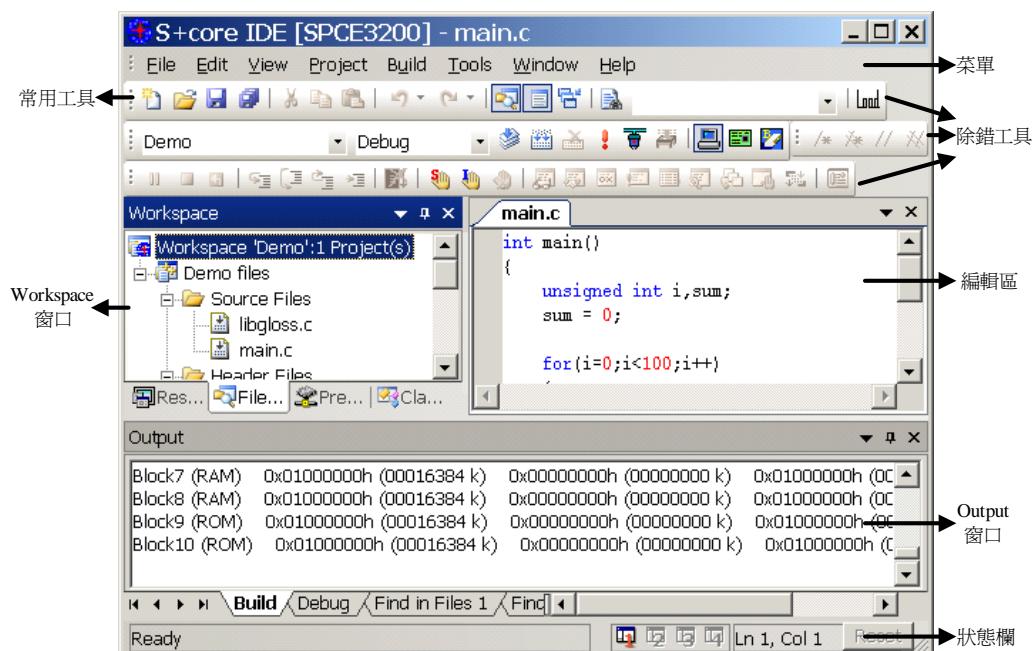


圖 5-3 S+core IDE 用戶介面

下面介紹 S+core IDE 的常用操作和常見設置。

5.2.1 工程的編輯

S+core IDE 中，所有檔都是透過工作區來組織的。所以，在使用 S+core IDE 進行開發時，首先要做的通常是建立或打開一個工作區，然後再向其中添加工程、檔。

1. 建立工作區

在建立工程時 S+core IDE 會自動新建一個工作區。當然，也可以在建立工程前先建立一個工作區，具體操作如下。

- (1) 如圖 5-4所示，選中菜單的 File->New...(或者按快捷鍵 Ctrl+N)打開如圖 5-5的對話方塊。
- (2) 切換到 Workspace 標籤，在“Location”欄選擇工作區的保存路徑；在“Workspace name”欄輸入工作區名稱；然後，點擊“OK”按鈕即可。

這樣，一個工作區就建立好了。介面顯示如圖 5-6。

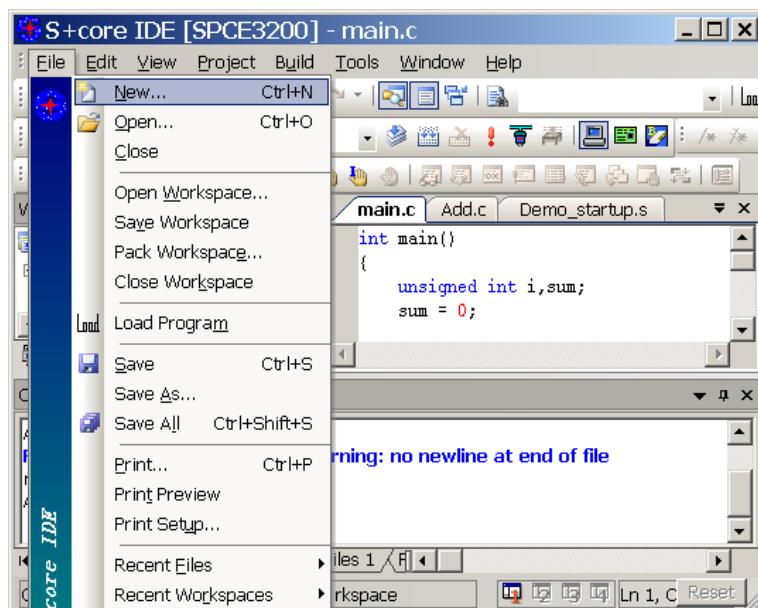


圖 5-4 新建工作區/工程/檔

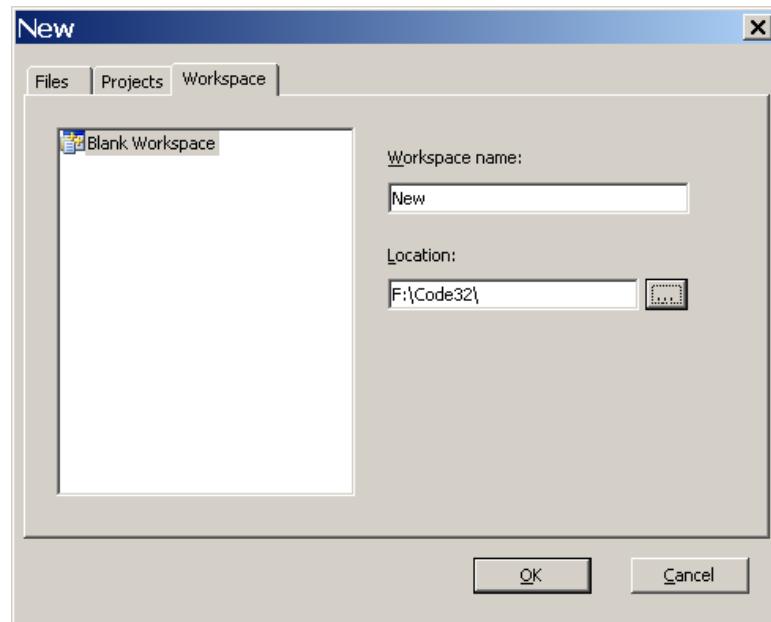


圖 5-5 新建工作區

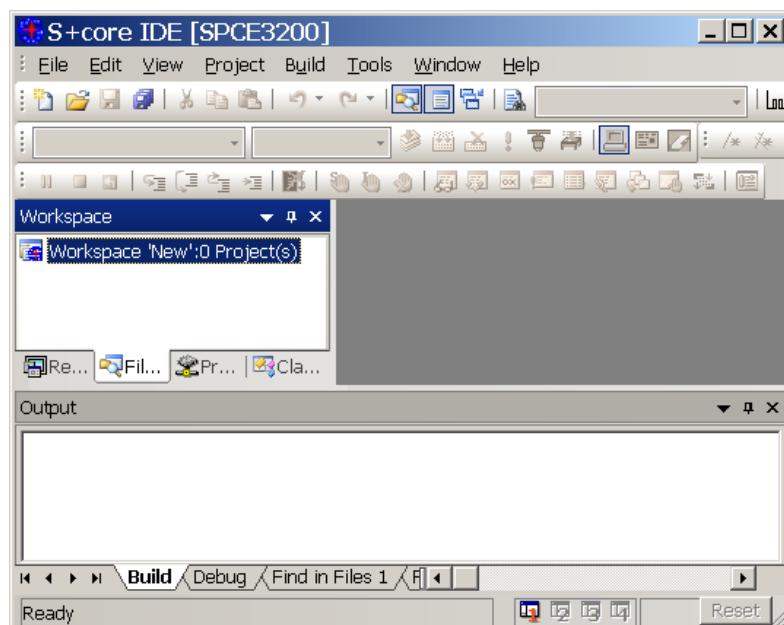


圖 5-6 工作區建立後的介面

2. 建立工程

- (1) 如圖 5-4所示，選中菜單的 File->New...(或者按快捷鍵 Ctrl+N)打開如圖 5-7的對話方塊。
- (2) 切換到 Projects 標籤，然後在 “Location” 欄選擇工程的保存路徑；在 “Project Name” 欄輸入工程名稱。
- (3) 如果在建立工程前已經建立工作區，選擇 “Add to current workspace” ，否則選擇 “Create new workspace” 。

(4) 最後，點擊“OK”即可。

這樣，就建立好了一個工程，介面如圖 5-8。

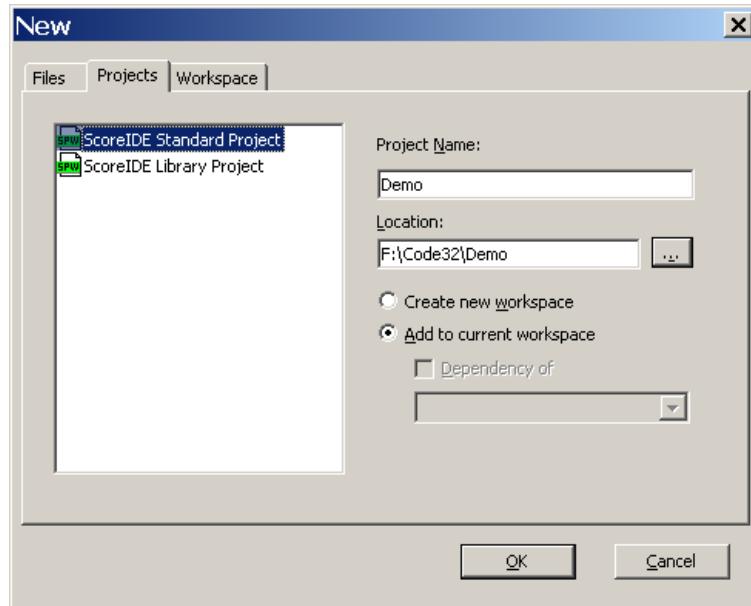


圖 5-7 新建工程

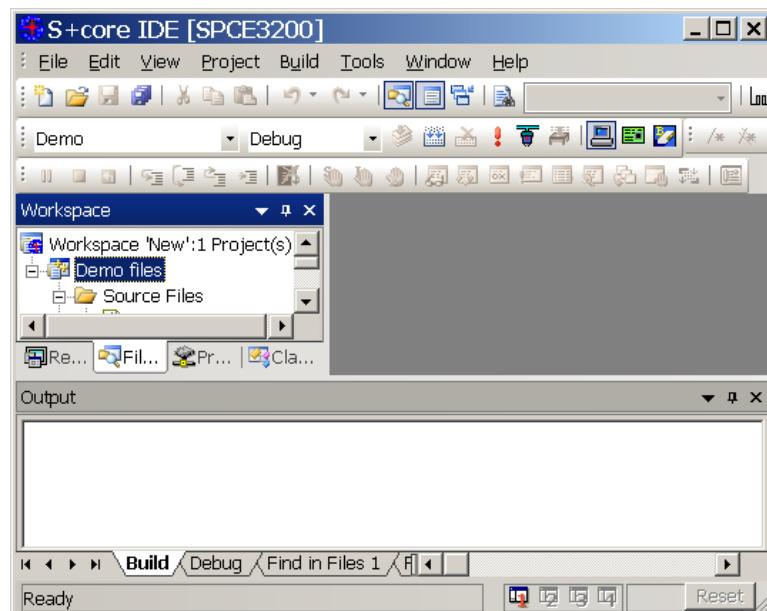


圖 5-8 工程建立後的介面

3. 建立檔案

- (1) 如圖 5-4所示，選中菜單的 File->New..(或者按快捷鍵 Ctrl+N)打開如圖 5-9的對話方塊。
- (2) 切換到 Files 標籤，然後勾選 “Add to project” 選項，以在特定工程下建立檔案。接下來，在 “Add to project” 下面的下拉清單中選擇同一個工作區內已經建立的工程的名稱。

- (3) 在 “Location” 欄選擇檔案的保存路徑。
- (4) 在 “File” 欄輸入檔案名稱。
- (5) 在左邊的檔類型選擇視窗中選擇要建立檔案的類型 (C 檔案、C++ 檔案、組合語言檔案或者標頭檔案等)。
- (6) 然後，點擊 “OK” 即可。

介面將如圖 5-10 所示，用戶可以在編輯區中為建立好的檔案編寫程式。

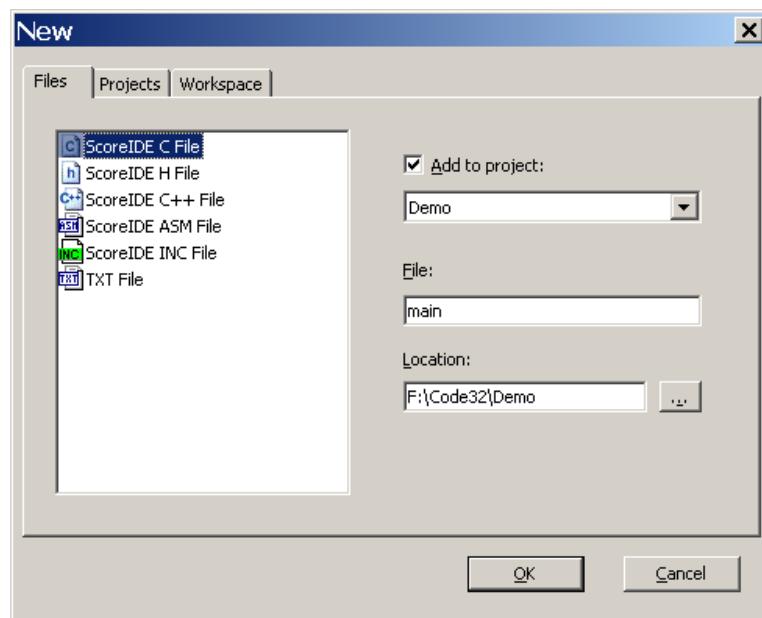


圖 5-9 新建檔案

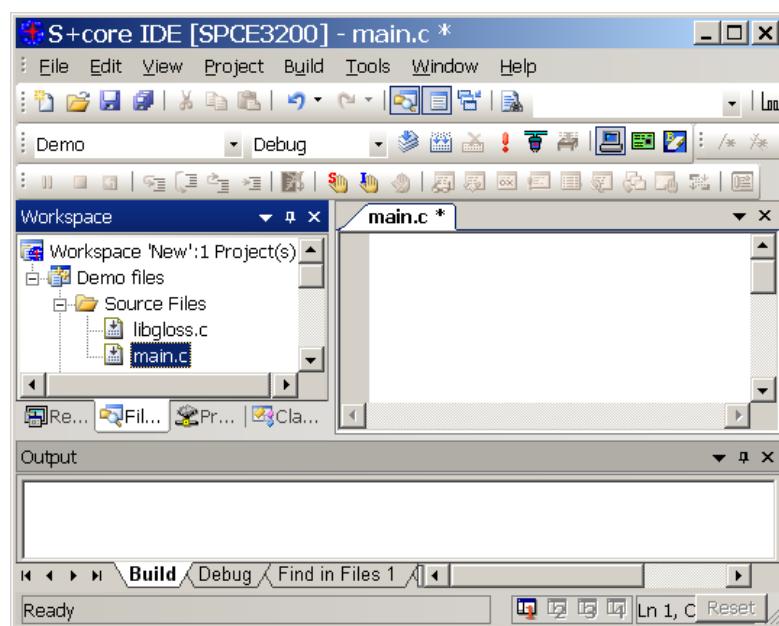


圖 5-10 新建檔介面

4. 添加工程到工作區

- (1) 如圖 5-11所示，選中菜單的 Project->Insert Project To Workspace...，打開如圖 5-12 對話方塊。
- (2) 在對話方塊中，選擇好要添加的工程檔後，然後點擊“打開”按鈕。

添加工程後的介面如圖 5-13。從 Workspace 視窗可看到，工作區中的工程數量增加了 1 個。

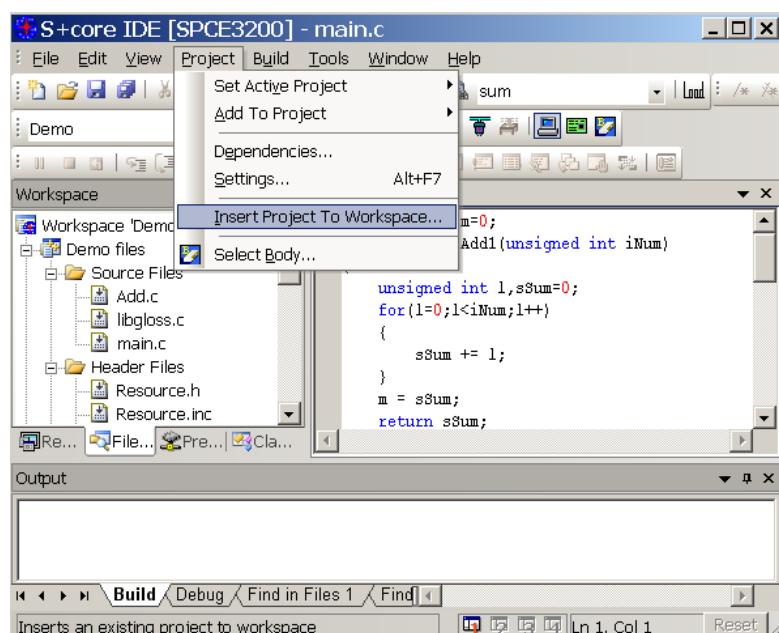


圖 5-11 添加工程到工作區

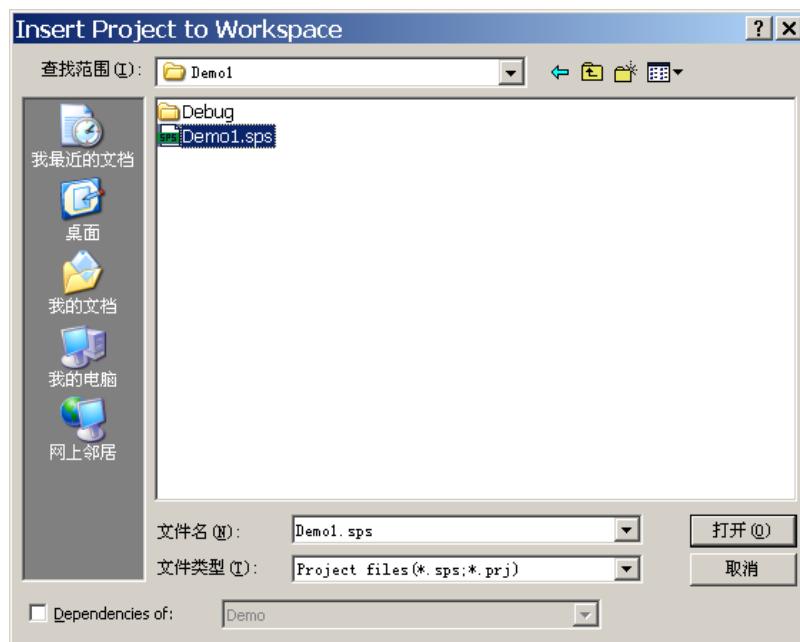


圖 5-12 添加工程檔對話方塊

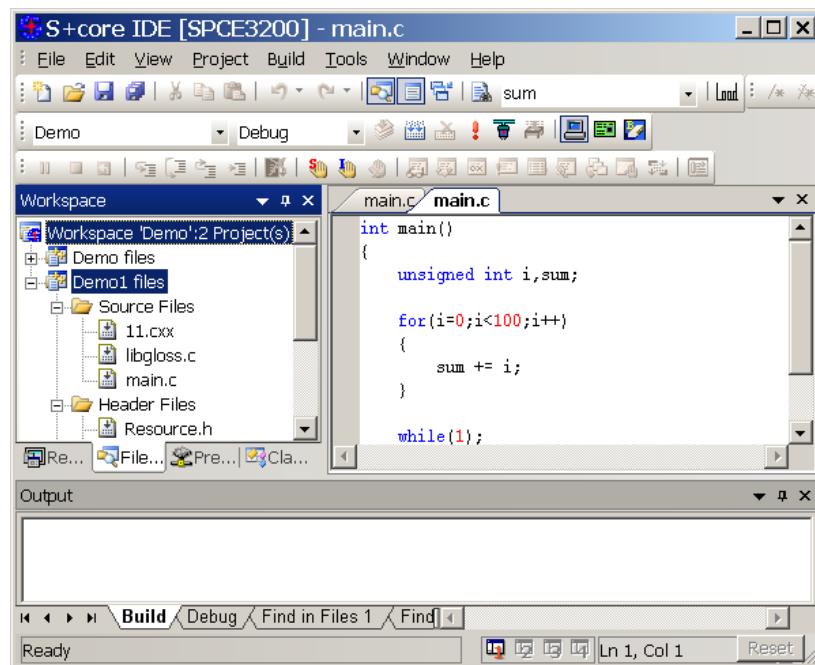


圖 5-13 添加工程後的介面

5. 添加檔到工程

- (1) 如圖 5-14，選中菜單的 Project->Add To Project。會彈出一個子菜單，其中有兩種選擇“New...”和“Files...”：

選擇“New...”可添加一個新的檔到工程，和新建檔到工程的功能一樣，這裏不再贅述；

選擇“Files...”可添加一個或者多個已有的檔到當前工程。

- (2) 選擇“Files...”。會出現如圖 5-15的對話方塊。

- (3) 透過該對話方塊，選擇好要添加的檔，然後點擊“打開”按鈕即可。

添加檔後的介面如圖 5-16。

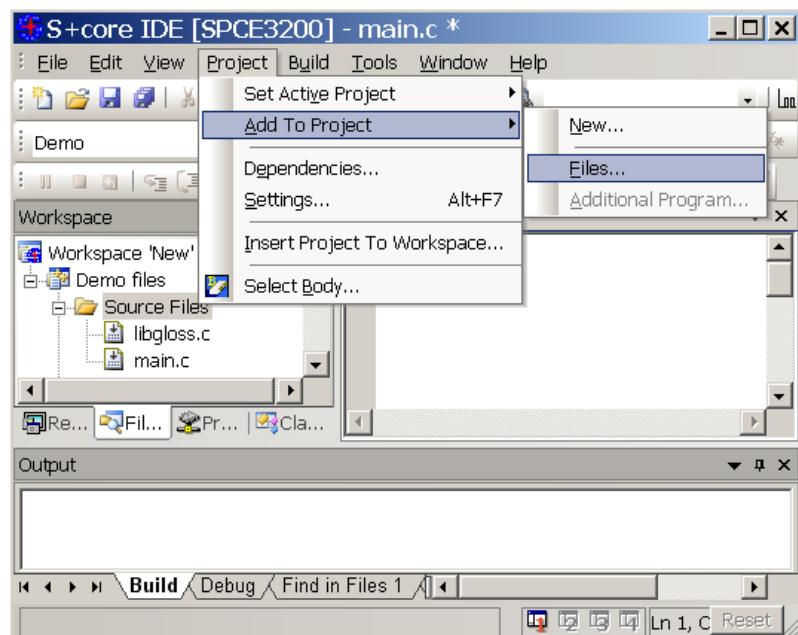


圖 5-14 添加檔到工程

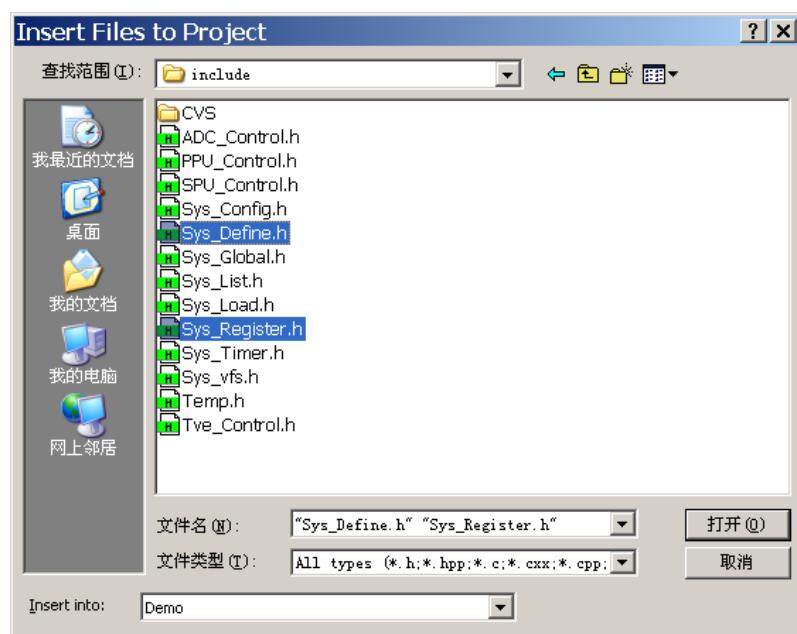


圖 5-15 選擇添加檔

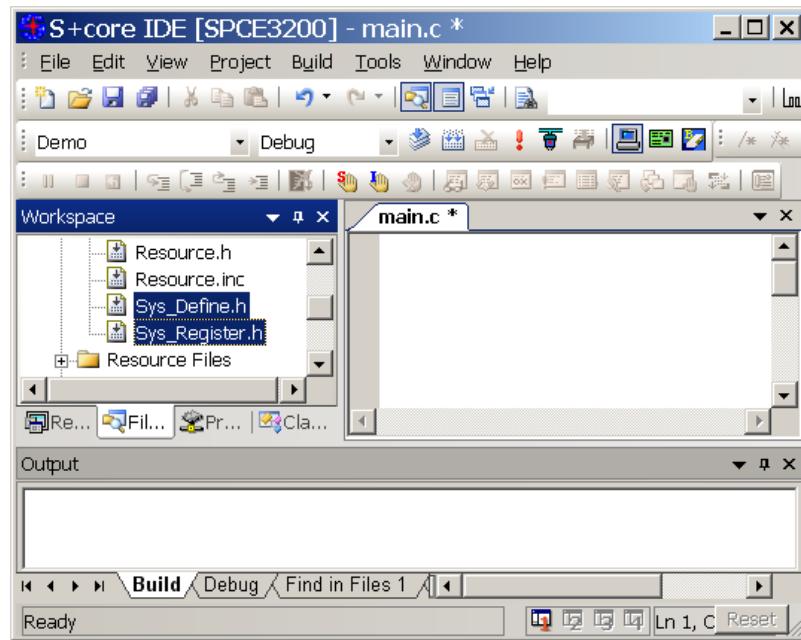


圖 5-16 添加檔後的介面

6. 打開已有工作區

- (1) 如圖 5-17，選中菜單的 File->Open Workspace...。如圖 5-18的對話方塊將被打開。
 - (2) 透過此對話方塊，選擇要打開的工作區檔案 (*.spw) ，然後點擊“打開”按鈕即可。
- 這樣，所選擇的工作區及其包含的所有工程均被打開，介面如圖 5-19。

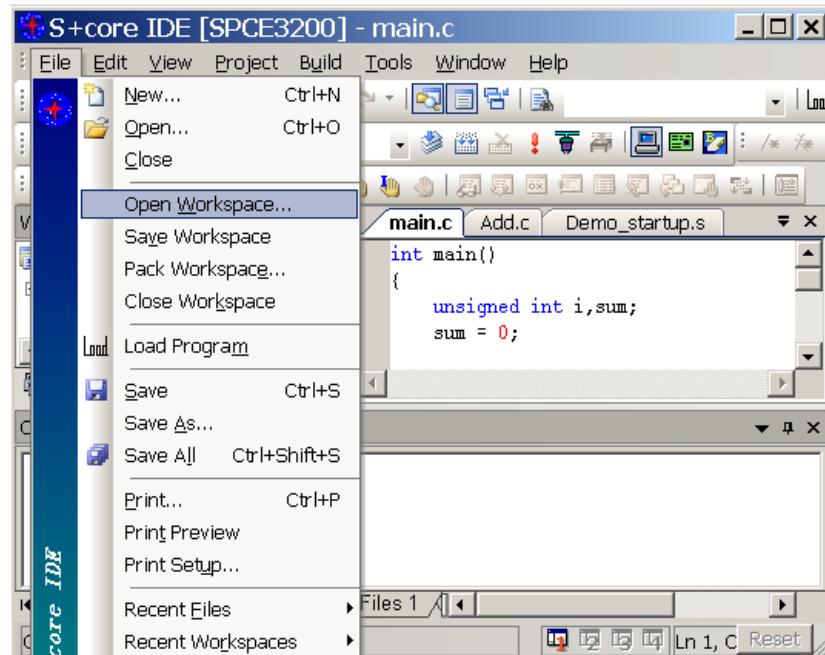


圖 5-17 打開工作區

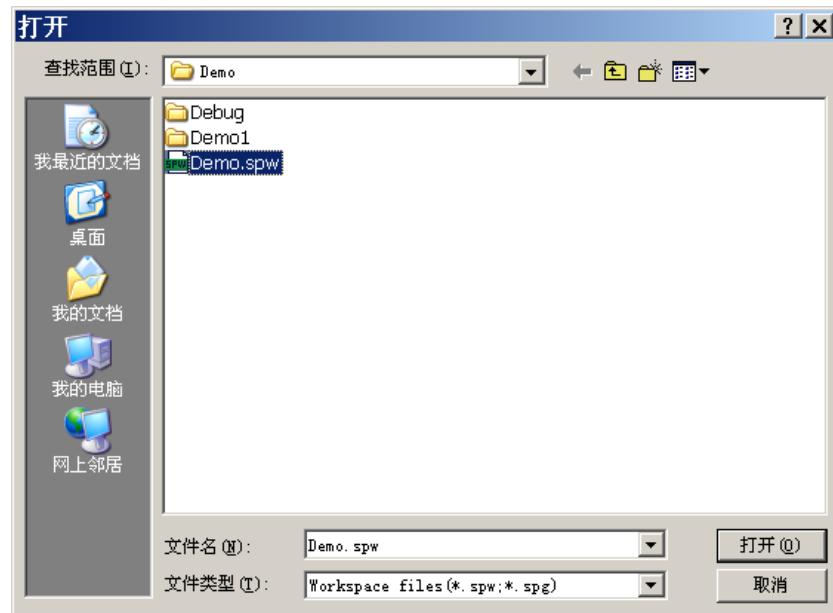


圖 5-18 選擇要打開的工作區

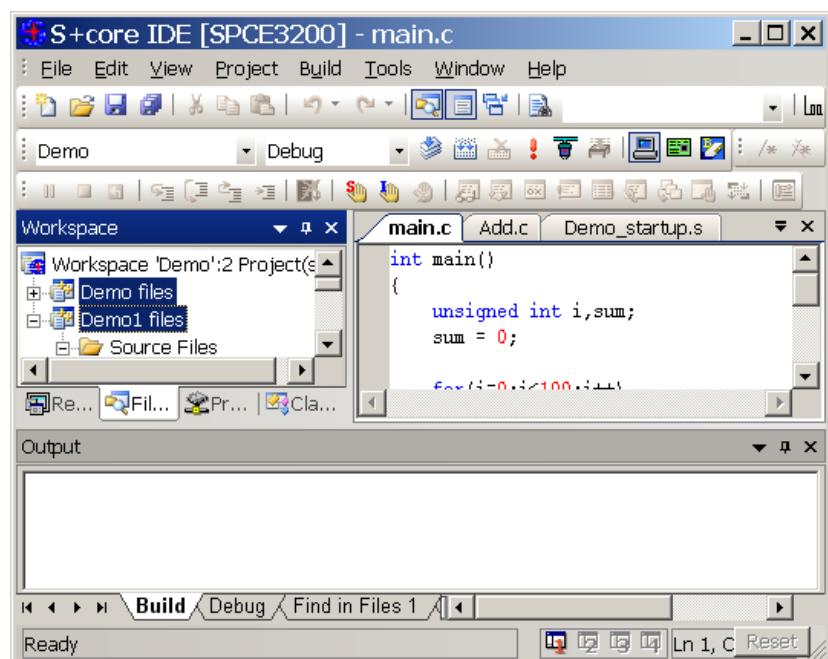


圖 5-19 打開工作區後的介面

7. 打開已有工程

- (1) 如圖 5-20，選中菜單的 File->Open (或者按快捷鍵 Ctrl+O)。如圖 5-21的對話方塊將被打開。打開已有工程或者檔；
- (2) 透過此對話方塊，選擇要打開的工程檔 (*.spw)，然後點擊“打開”按鈕即可。

打開工程後的介面如圖 5-22，打開時會自動生成一個和工程名相同的工作區，並在此工作區打開該工程。

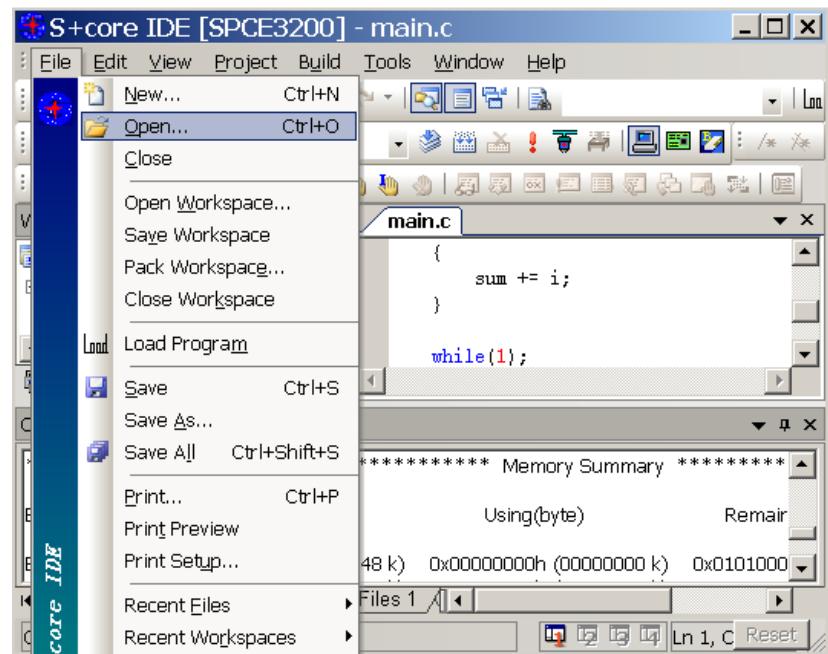


圖 5-20 打開工程

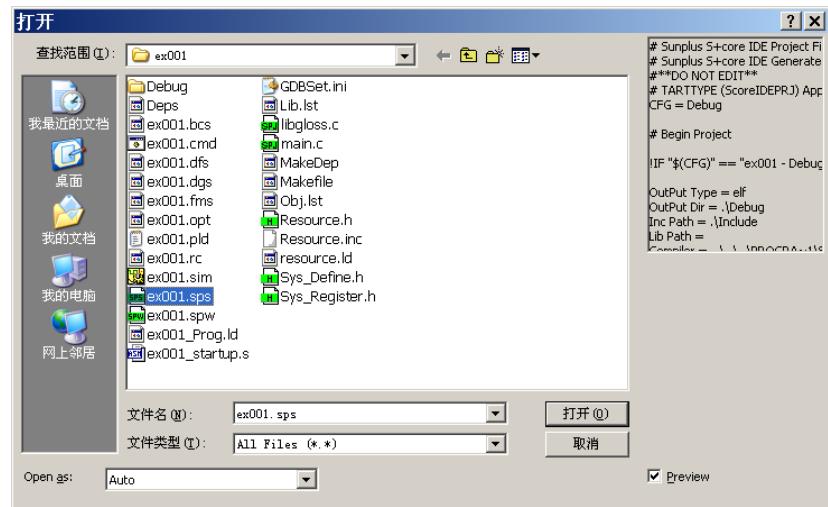


圖 5-21 選擇要打開的工程

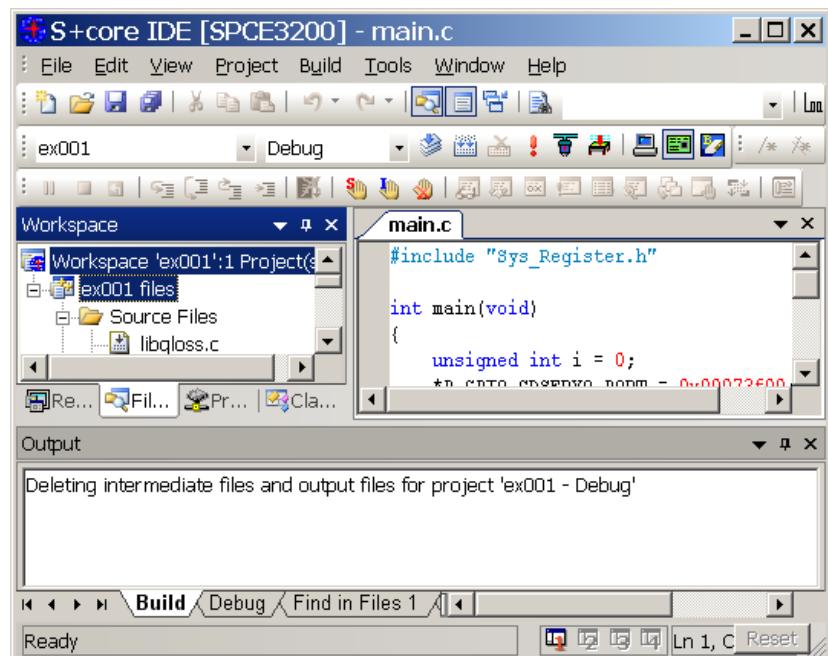


圖 5-22 打開工程後的介面

8. 選擇 Body

由於 S+core IDE 支援不同 S+core 內核的不同型號的晶片開發，所以在建立工程後要選擇工程基於的晶片型號，即選擇 Body，具體操作步驟為：

- (1) 點擊工具欄上的  圖示或者選擇菜單 Project->Select Body... 打開如圖 5-23 的對話方塊。
- (2) 選擇 Body Name：基於 SPCE3200 開發板開發時 Body Name 要選擇 SPCE3200。
- (3) 選擇 Probe：通常選擇 Auto 即可，也可以根據自己使用的 Probe 的類型選擇為平行埠型或者 USB 型。
- (4) 點擊 “OK” 按鈕。

若已給工程設置了 Body，也可以透過此操作來改變。

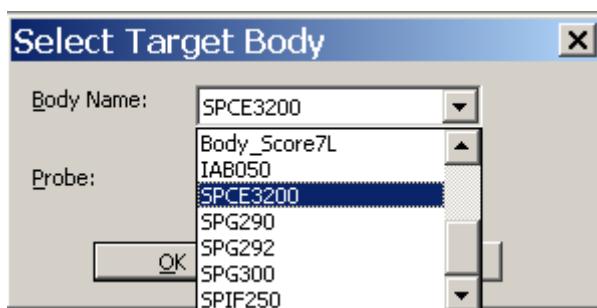


圖 5-23 Body 選擇

5.2.2 工程的設置

如圖 5-24，選擇菜單 Project-> Setting... 可以對工程進行設置。

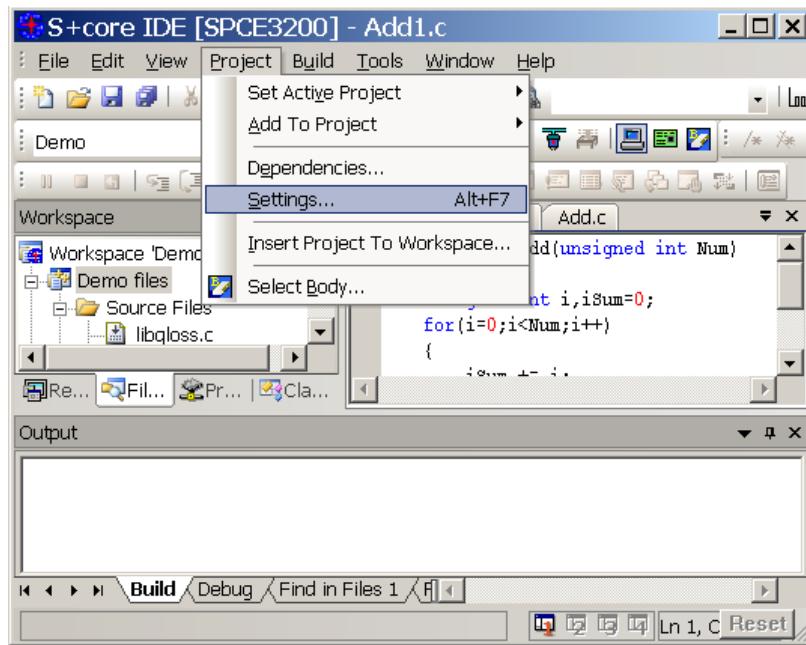
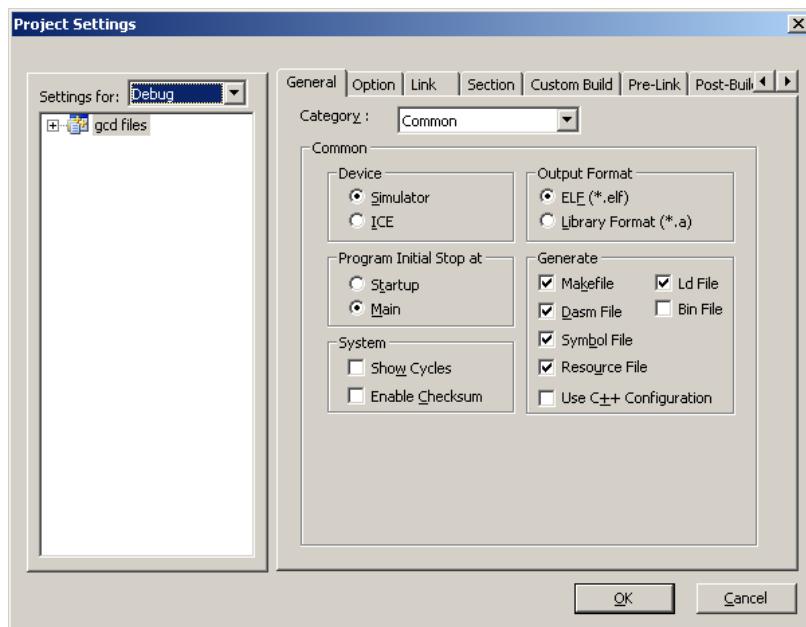


圖 5-24 打開 “Project Setting”

Project Settings 對話方塊如下：



從上圖可看到，Project Settings 對話方塊中包含 General、Option、以及 Link 等多個標籤，下面將分別進行介紹。

1. General 標籤

- (1) 介面如圖 5-25。透過此標籤可以對選中工程的一般選項進行設置。圖中“Category”有三種選擇：選擇“Common”為普通設置頁面；選擇“Advanced”為高級設置頁面；選擇“Exception Break”為異常中斷設置頁面。

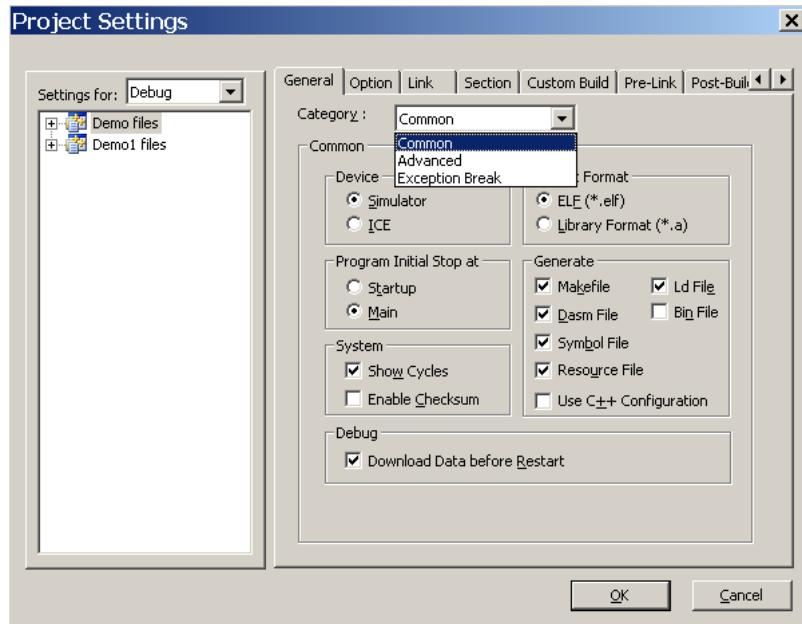


圖 5-25 Project 的“General”設置

- (2) Common 頁面中，可設置些通用選項，如圖 5-25。

其中：

- “Device”用於選擇除錯方式：“Simulator”軟體仿真；“ICE”為線上除錯。
- “Output Format”用於選擇輸出檔的格式：*.elf 為輸出可執行檔；*.a 為輸出庫檔。
- “Program Initial Stop at”用於選擇程式下載後停止的位置：“Main”為停止在 main 檔案中；“Startup”為停止在*_startup.asm 檔案中。
- “Generate”用於選擇是否要生成 Makefile、Symbol file、Dasm file、Resource file、Ld file 等檔案。
- “System”用於選擇軟體仿真時是否在狀態欄裏顯示時鐘週期（Show Cycles），是否要檢查下載資料的正確性（Enable Checksum）。通常，在軟體仿真時，如果選中 Show Cycles，則會在狀態欄看到每一條指令的執行週期，如圖 5-26。
- “Debug”用於選擇程式重新開始運行時是否重新下載。

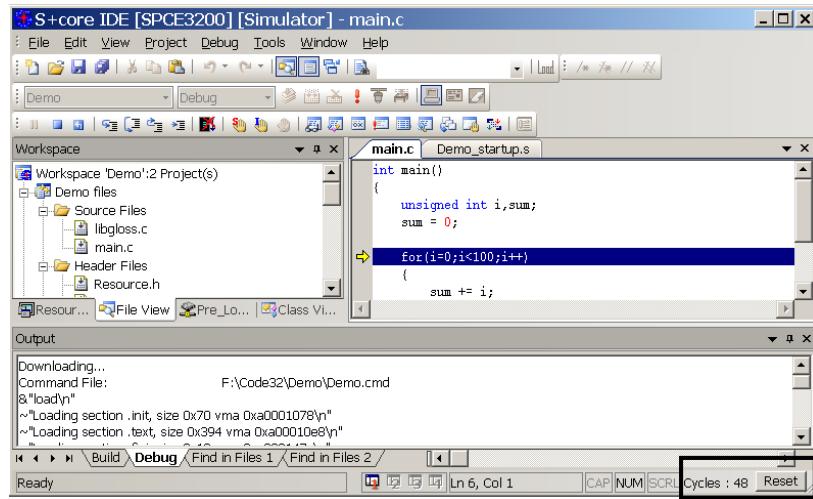


圖 5-26 Show Cycles 介面

(3) Advanced 頁面中，可進行原始檔案對映設置等高級設置，如圖 5-27。

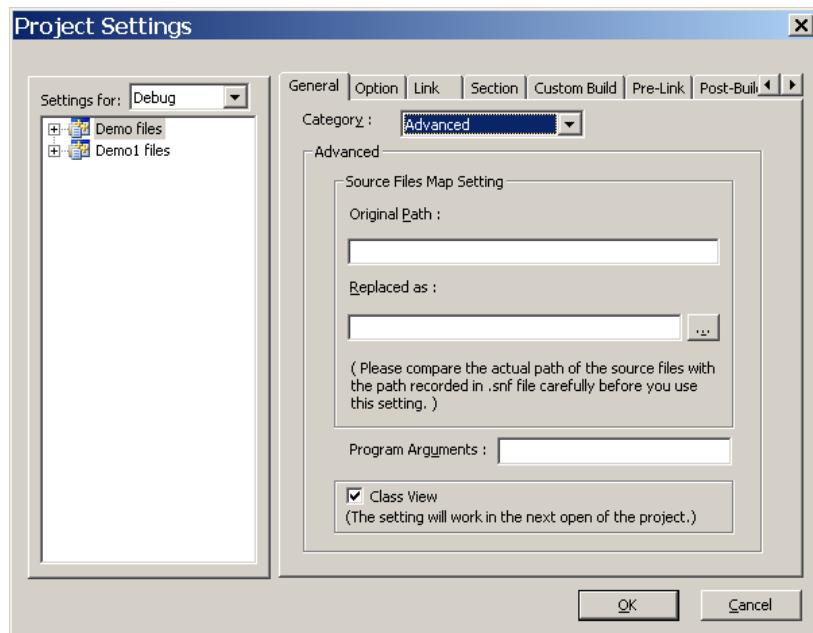


圖 5-27 Advanced 對話方塊

其中：

- “Original Path” 用來指定改變路徑前的工作區路徑。
- “Replaced as” 用來指定改變路徑後的工作區路徑。
- 若在本機上移動了工程或者工作區，透過這兩欄設置好路徑後不需要重新 Build 就可以下載程式。
- “Class View” 用來選擇是否在 Workspace 視窗的 Class View 區顯示資訊。該功能只有在選擇該選項後關閉工程，然後再重新打開工程時才起作用。

(4) Exception Break 頁面如圖 5-28。該頁面中，可設置異常的基底位址，預設 Exception Base Address 為 0x00000000。

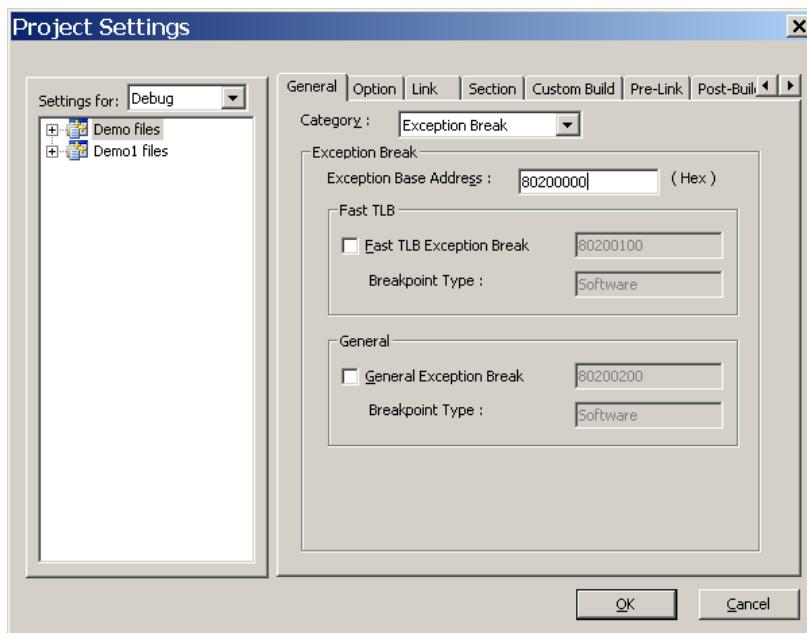


圖 5-28 Exception Break 對話方塊

2. Option 標籤

介面如圖 5-29。透過此標籤可以輸入命令行參數（有關 compiler、assembler 和 linker 的命令行參數請參考第 2.6 、2.7 、2.8 章）對 C compiler、assembler 和 linker 進行設置。

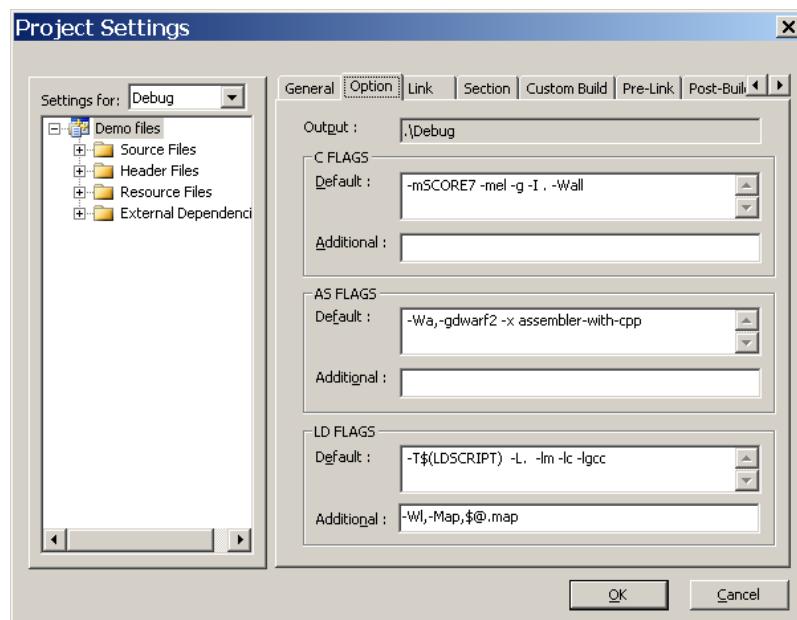


圖 5-29 Project 的“Option” 設置

3. Link 標籤

介面如圖 5-30。透過此介面可以鏈接外部檔、庫函數。

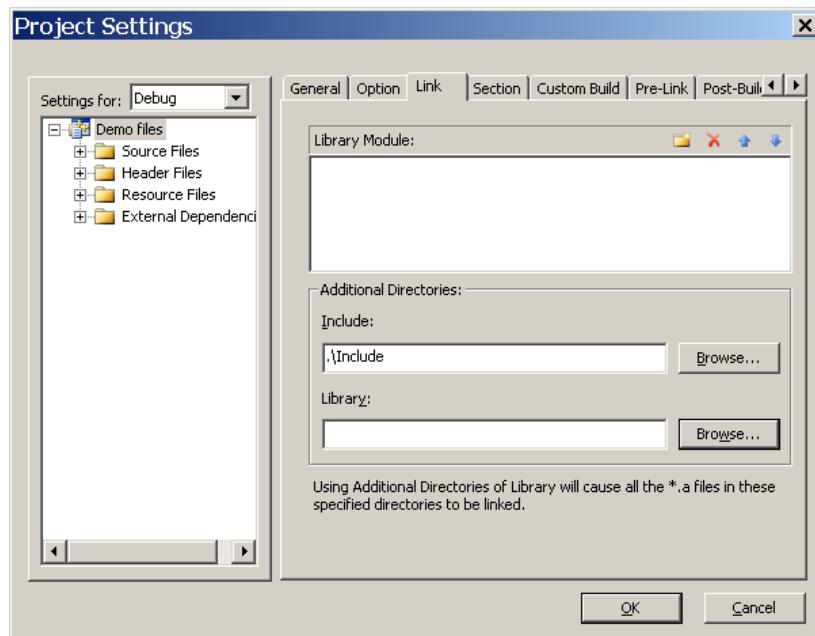


圖 5-30 Project 的“Link”設置

其中各個欄的功能為：

- “Library Module”：添加庫函數檔。
- “Include”：包含外部檔，透過“Browse...”按鈕選擇要包含的檔（夾）。
- “Library”：程式庫函數檔，透過“Browse...”按鈕選擇要鏈接的庫檔路徑，可以包含多個檔。

4. 其他標籤

除了 General、Option、Link 三個常用標籤外，還有六個標籤：Section、Custom Build、Pre-Link、Post-Build、Simulator Setting、Simulator Memory。

其中：

- Section 顯示庫/目標檔列表 (Obj&Lib modules)、預設區段 (Default sections)、用戶定義區段資訊 (User defined sections) 及其它資訊 (Others)；
- Custom Build 用於設置用戶自定義 Build 命令；
- Pre-Link 用於設置 Link 前編譯命令；
- Post-Build 用於設置 Build 後命令；
- Simulator Setting 用於設置 Body、Cache、斷點、LCD\UART\TIMER 外部介面等。比如 Body 要選擇為 Little Endian 還是 Big Endian，指令斷點的最多數量設置為幾個等。一般使用默認設置即可；
- Simulator Memory 用於觀察 Simulator Memory 及 CPU Memory。

5.2.3 工程的除錯

■ 除錯工具條

S+core IDE 除了提供類似新建、打開、保存、複製、剪貼、粘貼、撤銷、前進等常用工具外，還提供了開發除錯程式過程中常用到的除錯工具，如圖 5-31：

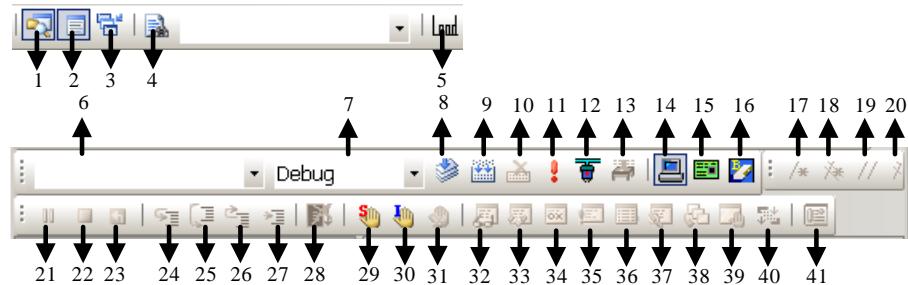


圖 5-31 常用除錯工具

表 5-2 為圖 5-31 中各工具的功能和快捷鍵。

表 5-2 除錯工具列表

序號	圖標	快捷鍵	功能及使用方法
1		Alt+0	打開 Workspace 視窗；點擊該圖標或者按 Alt+0
2		Alt+5	打開 Output 視窗；點擊該圖標或者按 Alt+5
3		-	打開 Windows 視窗；點擊該圖標
4		Ctrl+Shift+F	查找函數、變數、字串等；點擊該圖標或者按 Ctrl+Shift+F
5		-	Load 工程（可執行檔）；點擊該圖標
6	-	-	工程選擇對話方塊；可以選擇一個工作區的不同工程進行除錯
7	-	-	選擇為 Debug 模式或者 Release 模式
8		Ctrl+F7	Compile 當前檔案；點擊該圖標或者按 Ctrl+F7
9		F7	Build 當前工程；點擊該圖標或者按 F7
10		-	停止 Build 當前工程；點擊該圖標
11		F5	全速運行程式；點擊該圖標或者按 F5
12		F8	下載程式；點擊該圖標或者按 F8
13		-	Debug 除錯；點擊該圖標
14		-	選擇軟體仿真；點擊該圖標

序號	圖標	快捷鍵	功能及使用方法
15		-	選擇 ICE；點擊該圖標
16		-	選擇 Body (*)；點擊該圖標
17		-	注釋一段程式；選中要注釋的程式，點擊該圖標
18		-	去掉一段程式的注釋；選中被注釋的程式，點擊該圖標
19		-	注釋一行程式；選中要注釋的程式，點擊該圖標
20		-	去掉一行程式的注釋；選中被注釋的程式，點擊該圖標
21		-	停止下載；點擊該圖標
22		Shift+F5	停止 Debug；點擊該圖標或者按 Shift+F5
23		Ctrl+Shift+F5	重新開始運行；點擊該圖標或者按 Ctrl+Shift+F5
24		F11	單步運行，進入副程式；點擊該圖標或者按 F11
25		F10	單步運行，不進入副程式；點擊該圖標或者按 F10
26		Shift+F11	單步運行時，跳出副程式；點擊該圖標或者按 Shift+F11
27		Ctrl+F10	運行到遊標處；點擊該圖標或者按 Ctrl+F10
28		-	停止 GDB 除錯；點擊該圖標
29		F9	在指定位置設置軟體斷點；點擊該圖標或者按 F9
30		F6	在指定位置設置指令斷點；點擊該圖標或者按 F6
31		-	取消所有斷點；點擊該圖標
32		Alt+6	打開 Watch 視窗；點擊該圖標或者按 Alt+6
33		Alt+L	打開 Local Variable 視窗；點擊該圖標或者按 Alt+L
34		Alt+7	打開 Registers 視窗；點擊該圖標或者按 Alt+7
35		Alt+8	打開 Command 視窗；點擊該圖標或者按 Alt+8
36		Alt+9	打開 Memory 視窗；點擊該圖標或者按 Alt+9
37		Alt+A	打開 Disassembly 視窗；點擊該圖標或者按 Alt+A
38		Alt+C	打開 Call Stack 視窗；點擊該圖標或者按 Alt+C
39		Alt+B	打開 Breakpoints 視窗；點擊該圖標或者按 Alt+B
40		Alt+R	打開 Function Browser 視窗；點擊該圖標或者按 Alt+R
41		Alt+O	打開 OS Information 視窗；點擊該圖標或者按 Alt+O

■ 除錯視窗

1. Workspace 視窗

(1) 打開 Workspace 視窗

有三種方法可以打開 Workspace 視窗：

- 點擊  圖示；
- 按 Alt+0 快捷鍵；
- 選擇菜單 View-> Workspace。

(2) Workspace 視窗介紹

該視窗主要包含以下四個部分：

- Resource View：如圖 5-32 (A)，該視窗顯示工作區包含工程的資源檔案。
- File View：如圖 5-32 (B)，該視窗顯示工作區包含的工程、檔資訊；其中第一行顯示工作區名稱及該工作區包含的工程數量；從第二行開始顯示工程名及工程中包含的檔案名。
- Pro_Load View：如圖 5-32 (C)，該視窗顯示預下載工程資訊，包含可執行檔等。
- Class View：如圖 5-32 (D)，該視窗顯示工程中定義的總體變數、全局函數、巨集等資訊。

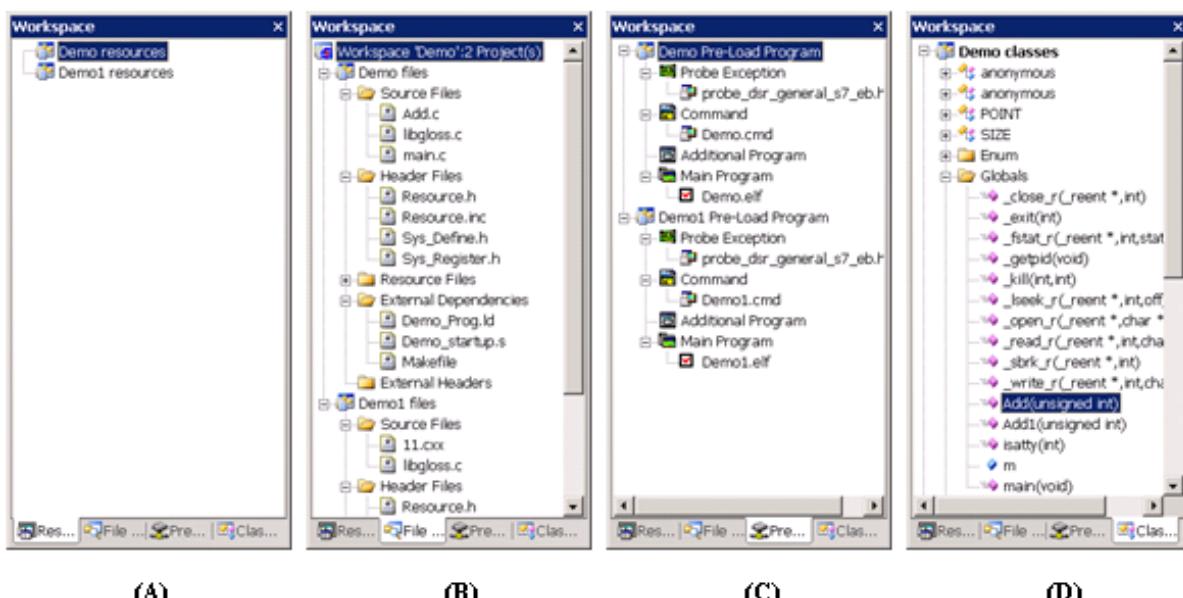


圖 5-32 Workspace 視窗

2. Output 視窗

(1) 打開 Output 視窗

有三種方法可以打開 Output 視窗：

- 點擊  圖示；

- 按 Alt+5 快捷鍵；
- 選擇菜單 View-> Output。

(2) Output 視窗介紹

該視窗主要包含以下四個部分：

- Build**：如圖 5-33，該視窗顯示工程編譯或者鏈接的相關資訊，這些資訊包括程式編譯/鏈接過程資訊、程式是否有錯誤、生成的可執行檔案名、Memory 的佔用情況等。
提示：如果在編譯或者鏈接時顯示錯誤資訊，雙擊錯誤資訊便可以定位程式中的錯誤。
- Debug**：如圖 5-34，該視窗顯示工程下載及除錯的相關資訊。
- Find in Files 1**：如圖 5-35，該視窗顯示在工程中查找變數、函數、字串等時的查找結果資訊，這些資訊包括變數、函數、字串等所在的路徑、檔案名、在檔案中的行位置、源程式語句、查找結果的總數等。
提示：雙擊 Output 視窗中的查找結果便可以定位到要查找的程式語句。
- Find in Files 2**：功能與 Find in Files 1 相同。

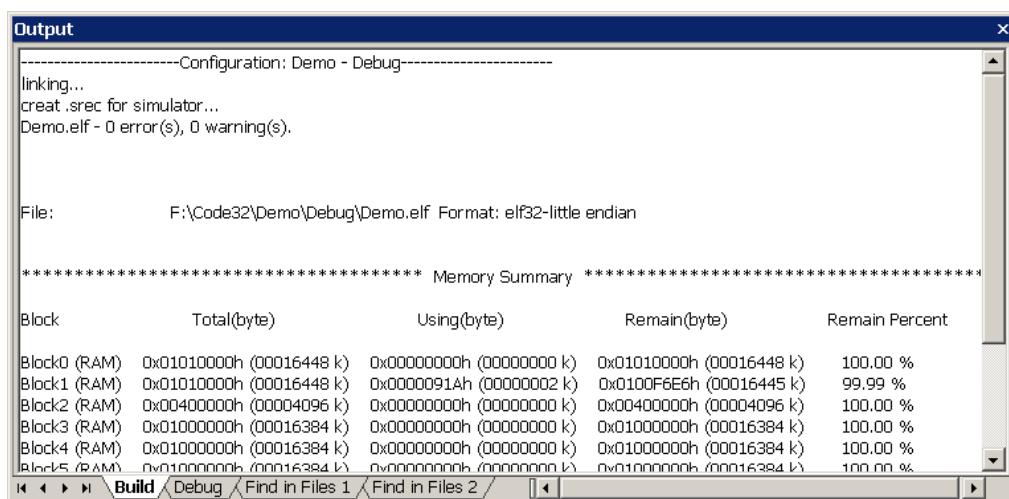


圖 5-33 Output 視窗(Build)



圖 5-34 Output 視窗(Debug)

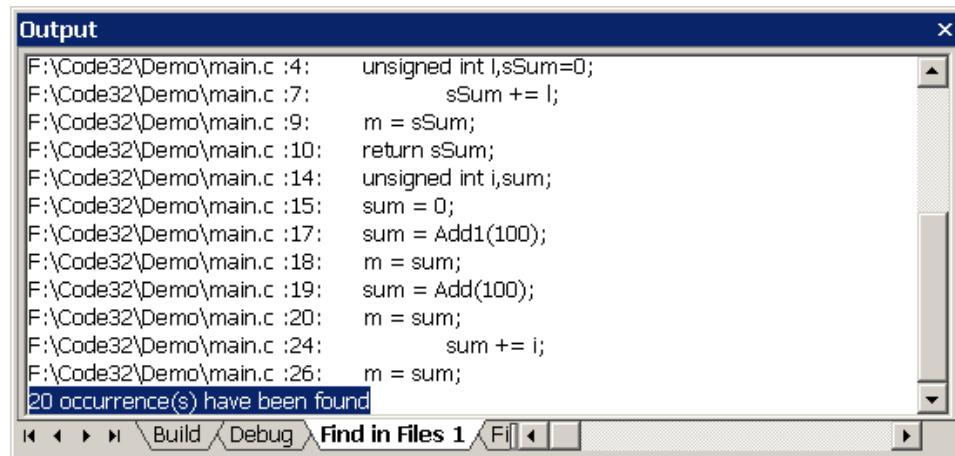


圖 5-35 Output 視窗(Find in Files)

3. Windows 視窗

- 點擊  圖示即可打開 Windows 視窗，如圖 5-36。
- 該視窗顯示當前打開並在編輯區顯示的檔。
- 該視窗中各按鈕功能分別為：

“Activate”	啓動檔
“Save”	保存檔
“Close Window(s)”	從編輯區關閉檔
“OK”	退出 Windows 視窗

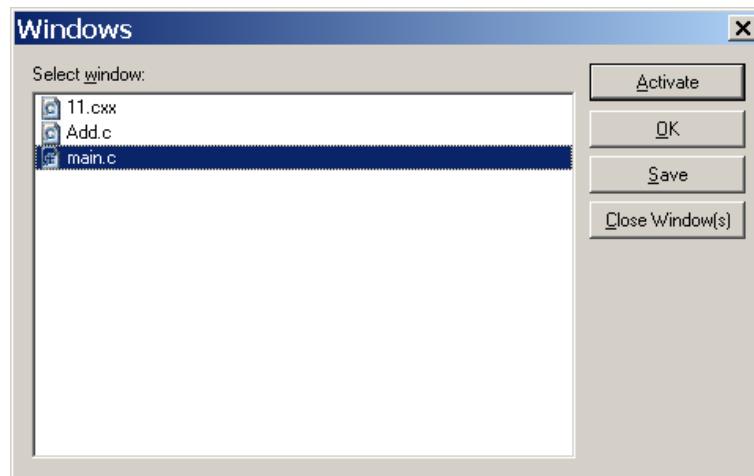


圖 5-36 Windows 視窗

4. Watch 視窗

Watch 視窗為變數觀察視窗。如圖 5-37。

(1) 打開 Watch 視窗

有三種方法可以打開 Watch 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+6 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Watch。

(2) Watch 視窗介紹

該視窗主要包含以下欄位：

- Name：變數名稱。在除錯狀態下，可在 Name 欄裏輸入變數名。
- Value：變數值。當在 Name 欄輸入變數名並按回車後，顯示該變數的值。在除錯的過程中，可以觀察該變數值的變化；同時，還可以透過 Value 欄修改該變數的值，以便用戶跟蹤除錯程式。
- Address：變數位址。當在 Name 欄輸入變數名並按回車後，系統會自動給該變數分配一個位址，Address 欄顯示該位址值。
- Type：變數類型。當在 Name 欄輸入變數名並按回車後，顯示該變數的定義類型。

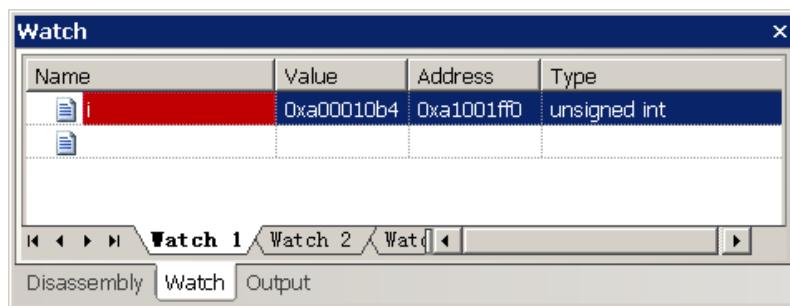


圖 5-37 Watch 視窗

(3) 說明

對於局部變數，只有程式運行在定義該變數的函數中時，才能顯示該變數的值、位址和類型；如果程式運行到其他函數，該變數的 Value、Address 和 Type 都顯示為“not find”，如圖 5-38。

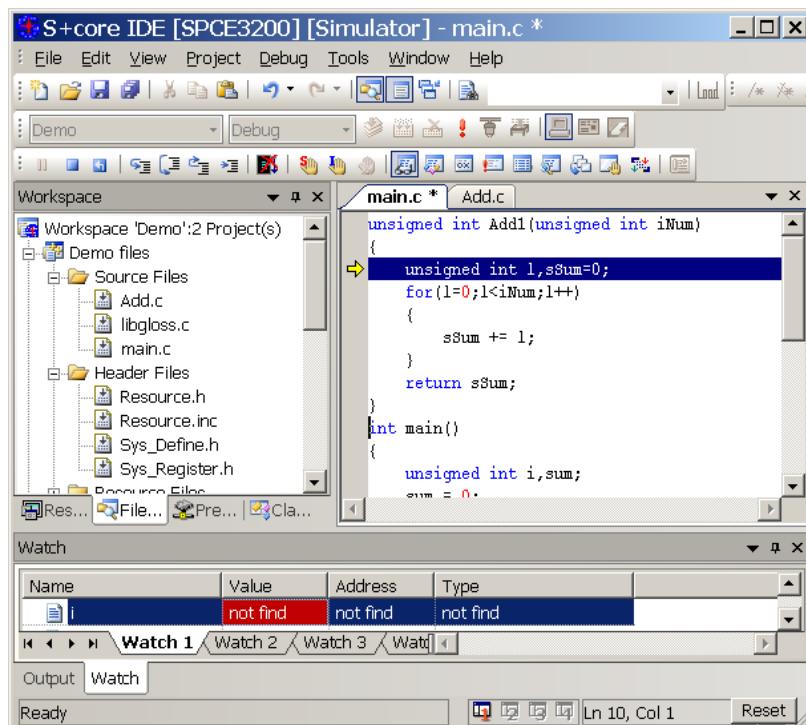


圖 5-38 “i” 變數觀察介面

5. Local Variable 視窗

Local Variable 視窗為局部變數觀察視窗。如圖 5-39。

(1) 打開 Local Variable 視窗

有三種方法可以打開 Local Variable 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+L 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Local Variable。

(2) Local Variable 視窗介紹

該視窗主要包含以下欄位：

- Name：局部變數名稱。
- Value：局部變數值。
- Address：局部變數位址。
- Type：局部變數類型。

(3) 說明

- Local Variable 視窗的資料不可以修改。在除錯狀態下，當程式運行到一個函數時，系統會自動把函數中定義的所有變數及變數的值、位址、類型顯示在 Local Variable 視窗。
- Local Variable 視窗不會顯示總體變數，如果要跟蹤總體變數，只能透過 Watch 視窗來觀察。

Local Variable			
Name	Value	Address	Type
i	0xa00010b4	0xa1001ff0	unsigned int
sum	0x00000000	0xa1001ff4	unsigned int

圖 5-39 Local Variable 視窗

6. Registers 視窗

Registers 視窗為暫存器觀察視窗。Registers 視窗包含兩個視窗：General 視窗和 TLB Entry 視窗，如圖 5-40 和圖 5-41。

(1) 打開 Registers 視窗

有三種方法可以打開 Registers 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+7 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Registers。

(2) 說明

- Registers 視窗顯示各模式下暫存器的資料，在程式運行過程中，以紅色字表示該暫存器的資料被修改。
- 在除錯過程中，這些暫存器的資料也可以修改。

Registers											
r0 A0FFFFE8	r10 00000000	r20 p0000000	r30 00000000	TLBPT 00000000	DSAVE 00000000						
r1 A00019EC	r11 00010008	r21 p0000000	r31 00000000	PEADDR 00000000	COUNTER 00000000						
r2 A0FFF8	r12 00000000	r22 p0000000	PSR 00000001	TLBRPT 0000001F	LDCR 00000000						
r3 A00010C0	r13 00000000	r23 p0000000	COND 00000006	PEVN 00000000	STCR 00000000						
r4 00000000	r14 00000000	r24 p0000000	ECR 00000000	PECTX 00000000	CEH 00000000						
r5 00000000	r15 00000000	r25 p0000000	EXCPVEC A0000000	LIMPFN 00000000	CEL 00000000						
r6 00000000	r16 00000000	r26 p0000000	CCR 00010009	LDMPFN 01000000							
r7 00000080	r17 00000000	r27 p0000000	EPC 00000000	PREV FF000000							
r8 A0001A1C	r18 00000000	r28 A00059F0	EMA 00000000	DREG 1C000002							
r9 00000000	r19 00000000	r29 p0000000	TLBLOCK 00000000	PC A0001242							

圖 5-40 Registers(暫存器)觀察視窗

Registers											
PEVN 0 80000000	PECTX 0 00000000	VPN 0 80000	ASID 0 00	PFN 0 00000	B 0 0	C 0 0	W 0 0	V 0 0	G 0 0		

圖 5-41 Registers(暫存器)觀察視窗

7. Command 視窗

Command 視窗為 S+core IDE 命令視窗。如圖 5-42。

(1) 打開 Command 視窗

有三種方法可以打開 Command 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+8 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Command。

(2) 說明

除錯過程中，透過這個視窗可以執行一些 GDB 命令，例如圖 5-42 中提示的 “cls” ，當在 “>” 後鍵入 “cls” 後回車，就可以看到 Command 視窗內容被清除，重新顯示剛打開時的資訊。

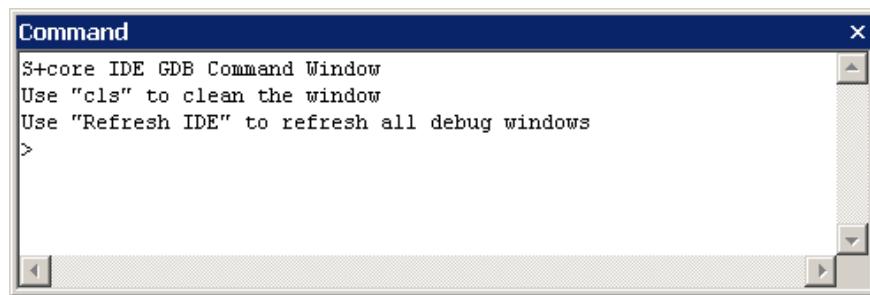


圖 5-42 Command 視窗

8. Memory 視窗

Memory 視窗為記憶體觀察視窗。如圖 5-43。

(1) 打開 Memory 視窗

有三種方法可以打開 Memory 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+9 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Memory。

(2) Memory 視窗介紹

該視窗主要包含以下三個部分：

- Memory 下拉菜單：該菜單可以選擇 RAM 或者 ROM 的位址段。系統按照記憶體的對映情況把記憶體分為不連續的 16 個不同大小的 Block，其中第 0~8Block 為 RAM 位址，第 9、10Block 為 ROM 位址，第 11~15Block 為保留區。在 Block 選擇框後面的框分別顯示選擇 Block 是屬於 RAM 還是 ROM，同時顯示該 Block 的起始和結束位址。當透過 Memory 下拉菜單選擇好 Block 後，位址內容區就會顯示從該 Block 起始位址開始的存儲資料。

- **Address** 位址輸入欄：透過該欄可以輸入一個位址，按回車後，位址內容區顯示從該位址開始的資料，同時 **Memory** 下拉菜單自動顯示當前位址所在的 **Block**，該 **Block**所在的存儲區（**RAM** 或 **ROM**）、起始位址和結束位址。
注意：**Address** 位址輸入欄只能輸入 **Block0~10** 中的任意一個位址，否則系統會提示輸入了一個無效的位址。
- 位址內容：位址內容區最左邊一列顯示位址；中間顯示這些位址單元的資料（16 進制表示的）；最右邊顯示這些位址單元資料的 **ASCII** 碼。

(3) 說明

在除錯過程中，**Memory** 視窗位址內容區的資料可以修改。

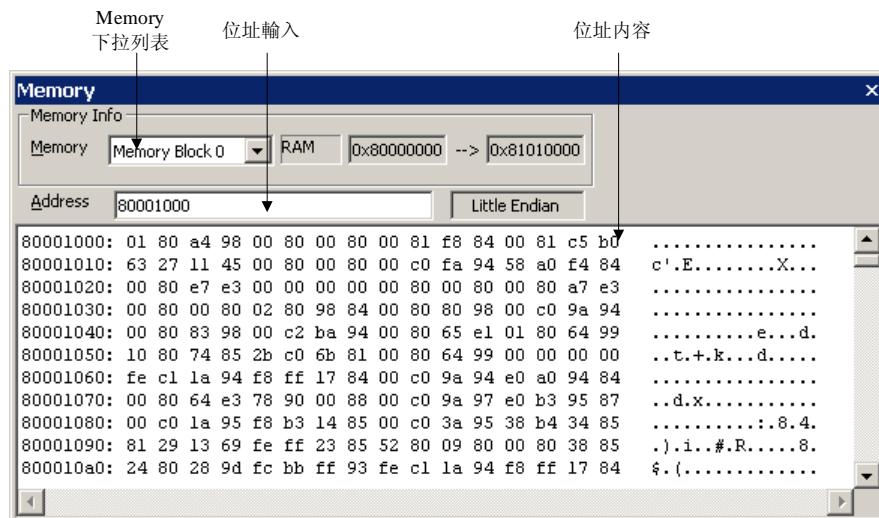


圖 5-43 Memory 視窗

9. Disassembly 視窗

Disassembly 視窗為反組譯觀察視窗。如圖 5-44。

(1) 打開 Disassembly 視窗

有三種方法可以打開 Disassembly 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 **Alt+A** 快捷鍵；
- 在除錯狀態下，選擇菜單 **View->Debug Windows->Disassembly**。

(2) 說明

除錯過程中，透過 Disassembly 視窗可以看到程式的反組譯代碼，如圖 5-44。黑色字體的部分為源代碼所在的路徑、程式在檔中的行位置及源程式語句；灰色的部分為對應的反組譯代碼。

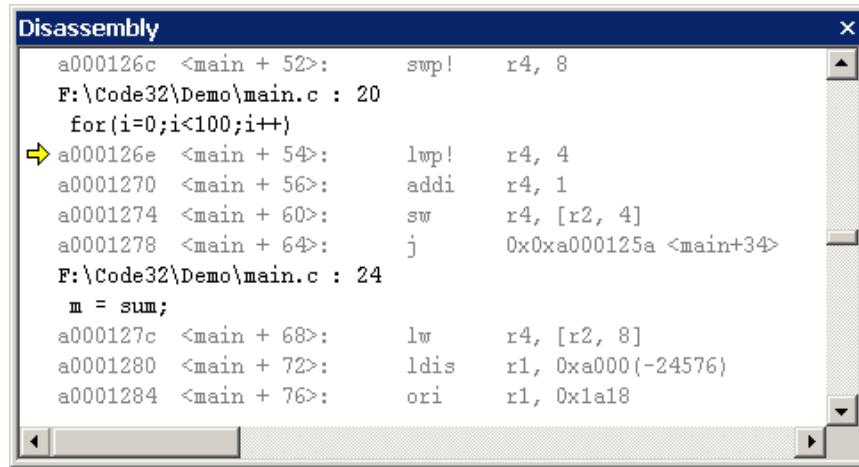


圖 5-44 Disassembly 視窗

10. Call Stack 視窗

Call Stack 視窗為堆疊調用情況觀察視窗。如圖 5-45。

(1) 打開 Call Stack 視窗

有三種方法可以打開 Call Stack 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+C 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Call Stack。

(2) 說明

- 如圖 5-45，在除錯過程中，透過 Call Stack 視窗可以看到當前運行程式的堆疊調用情況。調用函數時，先把該函數壓入堆疊，等到函數返回時，再把該函數彈出堆疊。
- 視窗顯示的內容中左邊的部分為函數名及其參數，右邊的部分為程式運行的行位置。

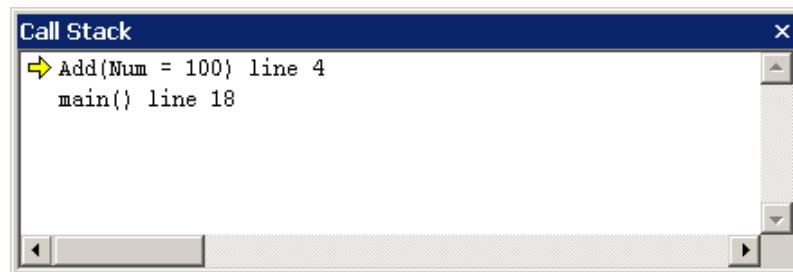


圖 5-45 Call Stack 視窗

11. Breakpoints 視窗

Breakpoints 視窗為斷點設置及觀察視窗，包含三個視窗：軟體斷點、指令斷點和指定變數斷點（資料斷點）視窗，如圖 5-46、圖 5-47、圖 5-48。

(1) 打開 Breakpoints 視窗

有三種方法可以打開 Breakpoints 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 **Alt+B** 快捷鍵；
- 在除錯狀態下，選擇菜單 **View->Debug Windows->Breakpoints**。

(2) Breakpoints 視窗介紹

該視窗主要包含以下欄位、按鈕：

- **Address**：斷點設置位址。
- **Condition**：斷點條件。
- **Enable**：使能斷點。
- **Line**：斷點所在程式檔中的行位置。
- **File Name**：斷點所在檔案名稱及路徑。
- **Variable**：變數斷點。
- “**ADD**” 按鈕：增加斷點。
- “**Modify**” 按鈕：修改斷點。
- “**Remove**” 按鈕：移除選中斷點。
- “**Remove All**” 按鈕：移除所有斷點。

(3) 說明

- 如圖 5-46，在透過 Breakpoints 視窗設置軟體斷點時，先在 **Address (Hex)** 欄裏填寫需要設置斷點的程式位址（如果有條件，再在 **Condition** 欄裏填寫條件），點擊 “**Add**” 按鈕，在右邊的框內會顯示該斷點的資訊，在 **Enable** 欄選中，就可以完成軟體斷點設置。同時，透過其他方式設置的軟體斷點也在 Breakpoints 視窗顯示，也可以透過該視窗進行使能、禁止、修改、移除等操作。

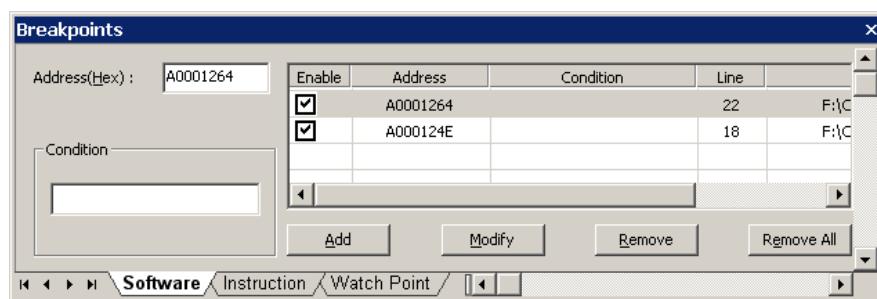


圖 5-46 Breakpoints 視窗(Software)

- 如圖 5-47，透過 Breakpoints 視窗設置指令斷點的方法和設置軟體斷點的方法相同。另外，透過其他方式設置的指令斷點也在 Breakpoints 視窗顯示，也可以透過該視窗進行使能、禁止、修改、移除等操作。

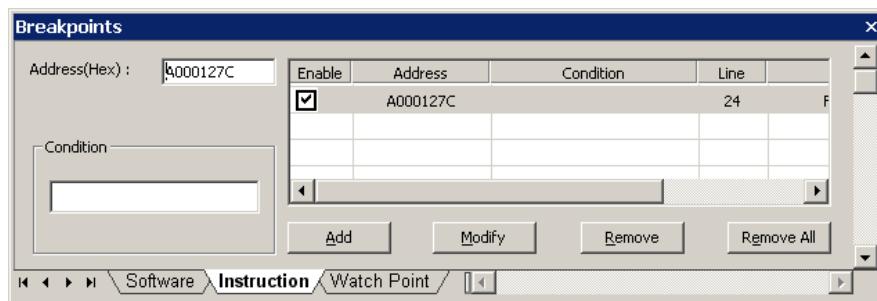


圖 5-47 Breakpoints 視窗(Instruction)

- 如圖 5-48，透過 Breakpoints 視窗設置變數斷點的方法和上面兩種類似：先在 Variable 欄裏填寫變數名，點擊 “Add” 按鈕，在右邊的框內會顯示該斷點的資訊，在 Enable 欄選中，就可以完成變數斷點設置；此時全速運行時系統會在有該變數的地方停下。如果在設置斷點的同時在 Condition 欄加上條件 ($= =$ 、 $!=$ 、 $>$ 、 $>=$ 、 $<$ 、 $<=$)，比如變數等於 ($= =$) 一個資料，此時全速運行時系統會在有該變數等於設置資料的時候停下，這對於一些迴圈來說，是非常有用的。

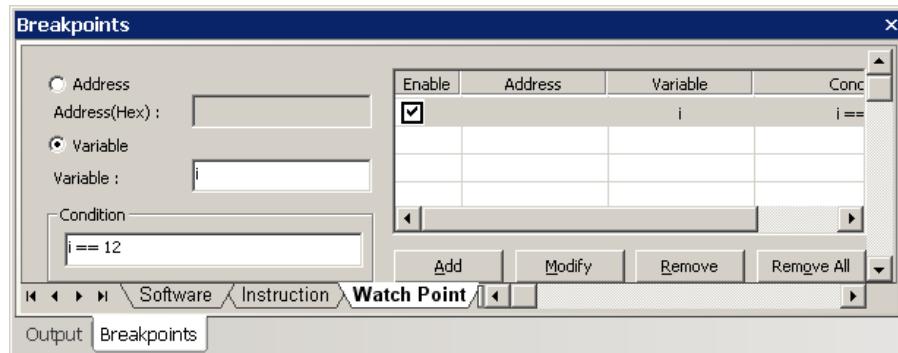


圖 5-48 Breakpoints 視窗(Watch Point)

12. Function Browser 視窗

Function Browser 視窗為功能函數觀察視窗，如圖 5-49。

(1) 打開 Function Browser 視窗

有三種方法可以打開 Function Browser 視窗：

- 在除錯狀態下，點擊  圖示；
- 在除錯狀態下，按 Alt+R 快捷鍵；
- 在除錯狀態下，選擇菜單 View->Debug Windows->Function Browser。

(2) Function Breakpoints 視窗介紹

該視窗主要包含以下兩個部分：

- Files**：顯示當前工程所有的 C 語言檔列表，透過 Select 欄選擇或者不選擇 C 檔，如果要選擇，在方框上點擊選中；Index 為序號；Files 欄顯示 C 語言檔案的名稱及路徑。

- **Function**：顯示被選中的 C 檔定義的所有功能函數列表，透過“Fast Input”欄輸入並查找被選中的 C 檔中的功能函數；Index 為序號；Name 欄顯示功能函數名及其參數；File Name 欄顯示功能函數所在的檔案的名稱及路徑。

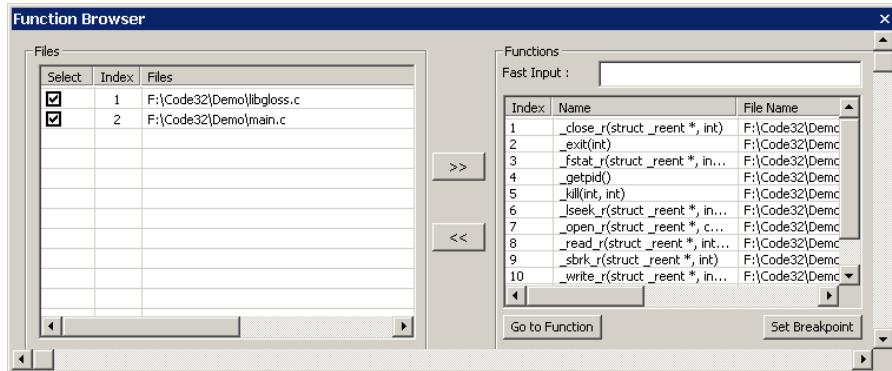


圖 5-49 Function Browser 視窗

13. OS Information 視窗

OS 資訊視窗，在 OS 的相關章節會有詳細介紹，故在此不做闡述。

5.3 應用舉例

【例 5.1】

在 S+core IDE 下建立一個工作區 MyCode，並在其中建立兩個工程 Code1 和 Code2，Code1 實現的功能是從 1 加到 100，計算出累加結果；Code2 實現的功能是使 SPC-E3200 開發板的 LED1 燈閃爍。

1. 要求：

軟體仿真除錯工程 Code1：

- (1) 單步運行，觀察執行各條程式語句需要的週期數，並分別透過變數觀察視窗觀察各定義變數的變化、暫存器觀察視窗觀察各暫存器的變化；
- (2) 全速運行，分別添加一個軟體斷點、一個指令斷點和一個變數斷點；透過添加的變數斷點觀察從 0 加到 50 時的結果。
- (3) 全速運行，透過變數觀察視窗觀察最後的運行結果。

利用 ICE 進行線上除錯工程 Code2：

- (1) 單步運行，透過 Memory 視窗觀察 0x8820004C~0x8820004F 位址單元的內容的變化和暫存器視窗觀察暫存器的變化；
- (2) 全速運行，觀察其現象。

2. 解析及步驟：

- 第一步：新建工作區 MyCode，如圖 5-50。

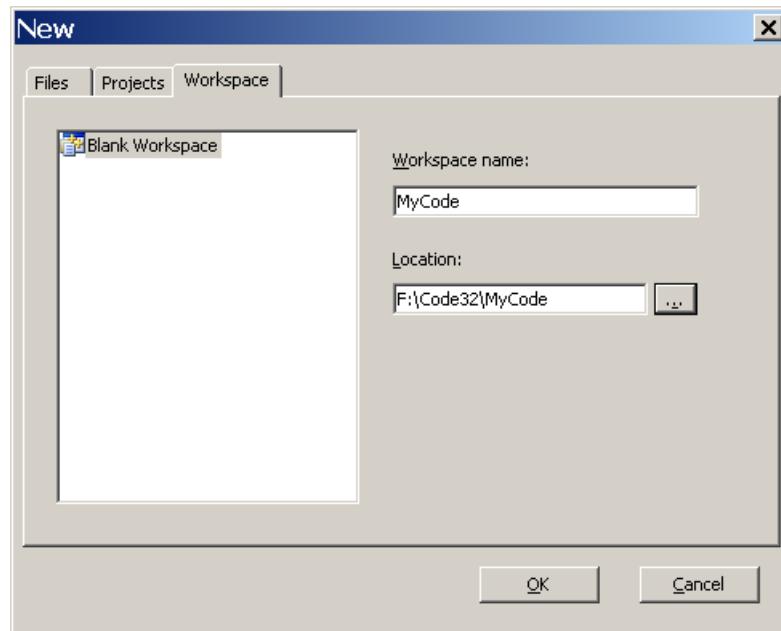


圖 5-50 新建工作區 MyCode

- 第二步：新建工程 Code1 和 Code2，如圖 4-41 和 圖 5-52。建立工程後的介面如圖 5-53。

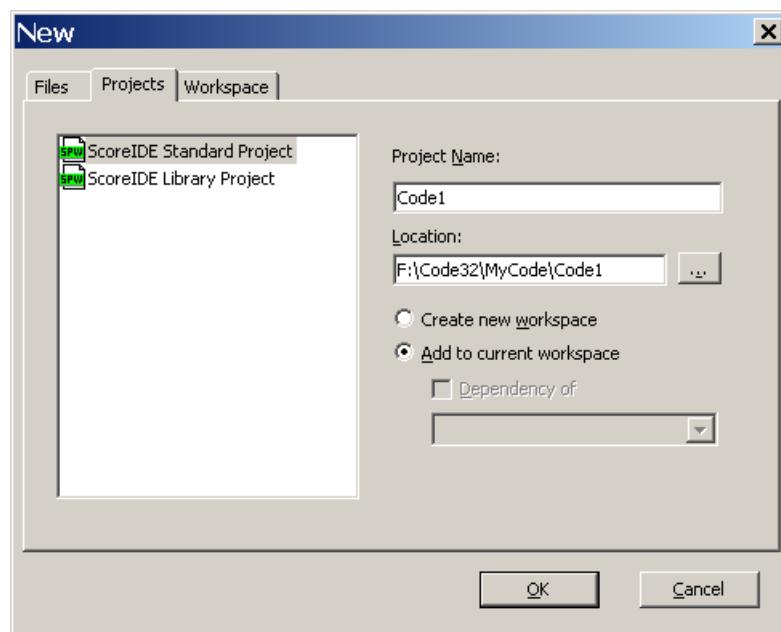


圖 5-51 新建工程 Code1

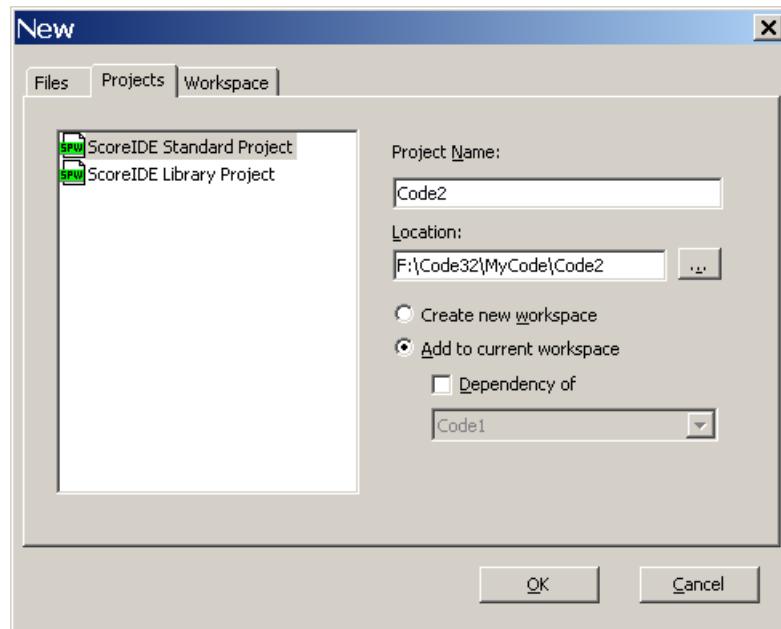


圖 5-52 新建工程 Code2

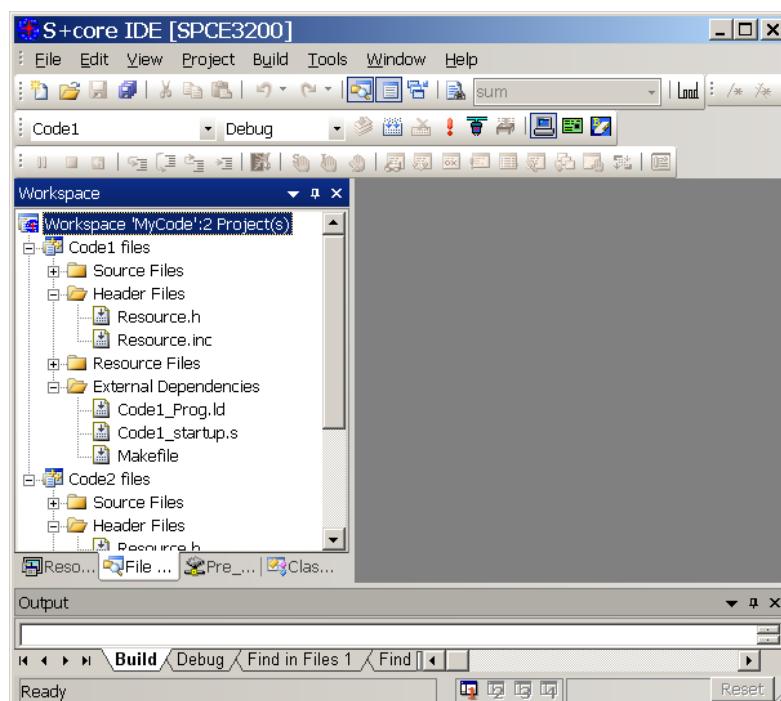


圖 5-53 建立工程後的介面

- 第三步：點擊  圖示選擇 Body。Code1 和 Code2 的 Body 都選擇為 SPCE3200，如圖 5-54。



圖 5-54 Body 選擇為 SPCE3200

■ 第四步：在工程 Code1 和工程 Code2 裏各建一個新檔 main.c。

(1) 在 Code1 裏新建一個 main.c 檔案，如圖 5-55。注意，Add to Project 選擇 Code1。

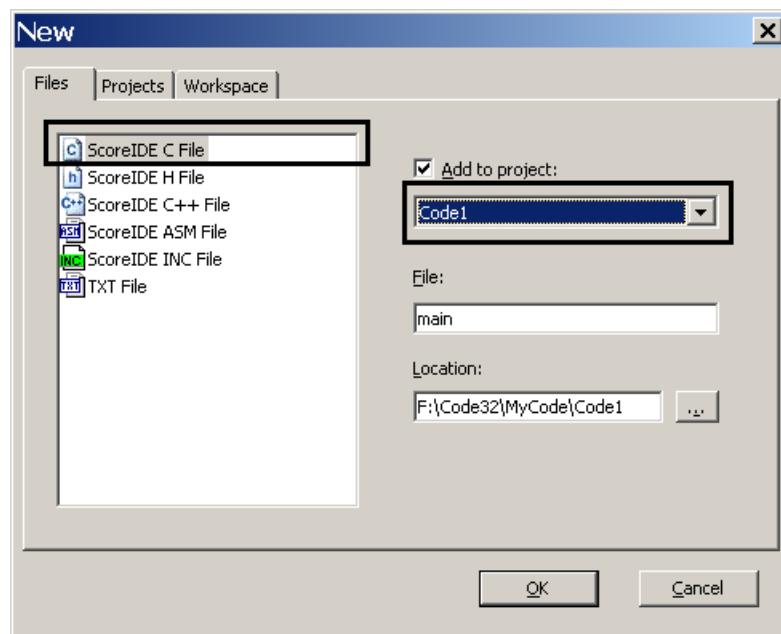


圖 5-55 在 Code1 工程裏建立檔 main.c

(2) 在 Code2 裏新建一個 main.c 檔案，如圖 5-56。注意，Add to Project 選擇 Code2。建立檔後的介面如圖 5-57。

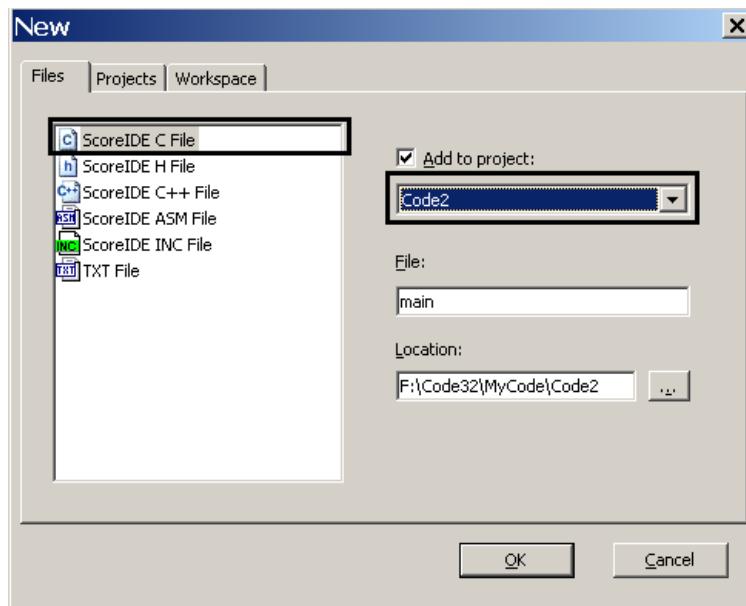


圖 5-56 在 Code2 工程裏建立檔 main.c

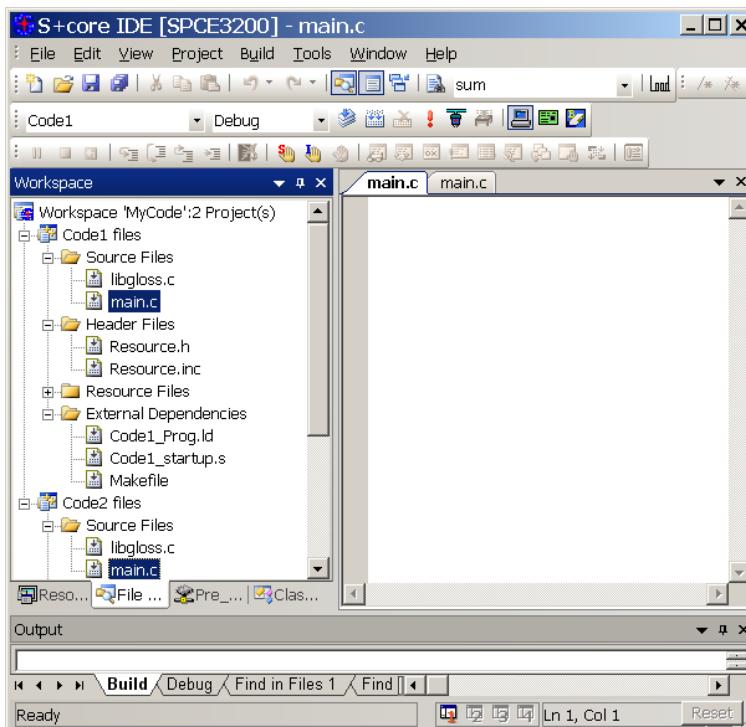


圖 5-57 建立檔後的介面

- 第五步：在 Code1 的 main.c 檔中編寫 1 到 100 的累加程式，在 Code2 的 main.c 檔中編寫 LED1 燈閃爍的程式。

Code1 的 main.c 程式段：累加結果保存在 sum 中。

```
int main(void)
{
    int i,sum=0;
```

```

for(i=0;i<=100;i++)
{
    sum += i;
}
while(1);
return 0;
}
    
```

Code2 的 main.c 程式段：

```

#define P_IOB_GPIO_SETUP (volatile unsigned int*)0x8820004C
int main(void)
{
    int i;
    *P_IOB_GPIO_SETUP = 0x00000100;
    while(1)
    {
        for(i=0;i<0x0027ffff;i++);
            *P_IOB_GPIO_SETUP ^= 0x00000001;
    }
    return 0;
}
    
```

- 第六步：在工具欄裏選擇 Code1 工程，如圖 5-58，選擇軟體仿真；按 F7 或者點擊  編譯鏈接生成可執行檔，透過 Output 視窗觀察鏈接資訊；按 F8 或者點擊  下載程式。

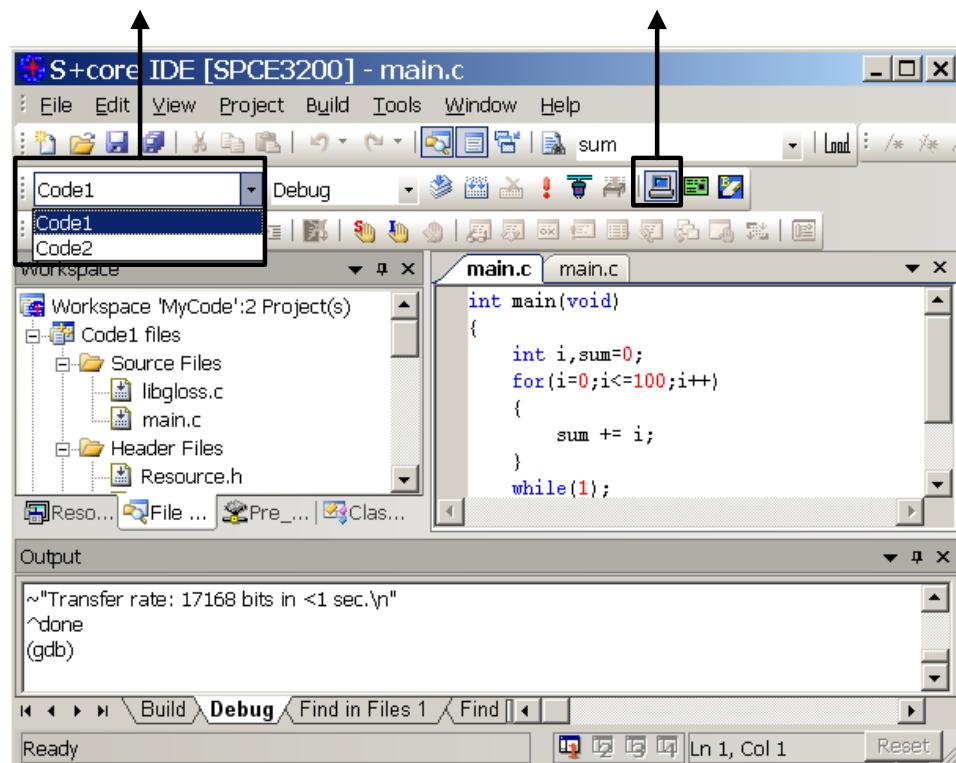


圖 5-58 選擇 Code1 工程

- 第七步：按 Alt+6 快捷鍵或者點擊  圖示打開 Watch 視窗，在 Name 欄依次輸入 i、sum；按 Alt+L 快捷鍵或者點擊  圖示打開 Local Variable 視窗；按 Alt+7 快捷鍵或者點擊  圖示打開 Registers 視窗。
- 第八步：如圖 5-59，按 F11 或者點擊  圖示單步運行，同時觀察 Watch 視窗和 Local Variable 視窗中 i 和 sum 的變化以及狀態欄裏 Cycles 的變化。

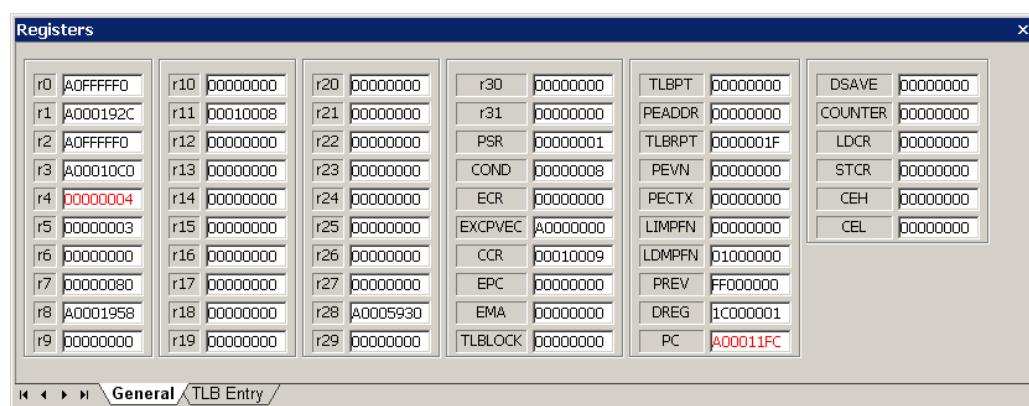
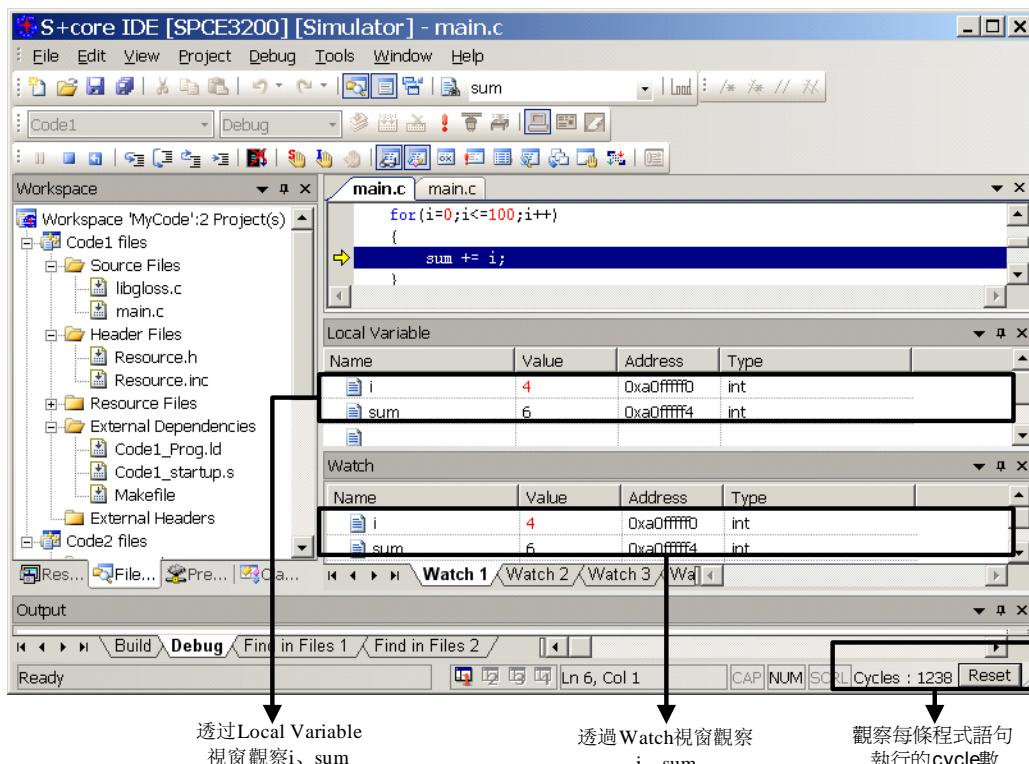


圖 5-59 除錯介面

- 第九步：如圖 5-60，按 F9 或者點擊  圖示在 while 語句加一個軟體斷點；按 F6 或者點擊  圖示在第一條語句加一個指令斷點；按 Alt+B 或者點擊  圖示打開 Breakpoints 視窗，如圖 5-60，在 Variable 欄填寫變數 i，再在 Condition 欄填寫 i=51；全速運行，觀察變數觀察視窗此時 sum 的值。

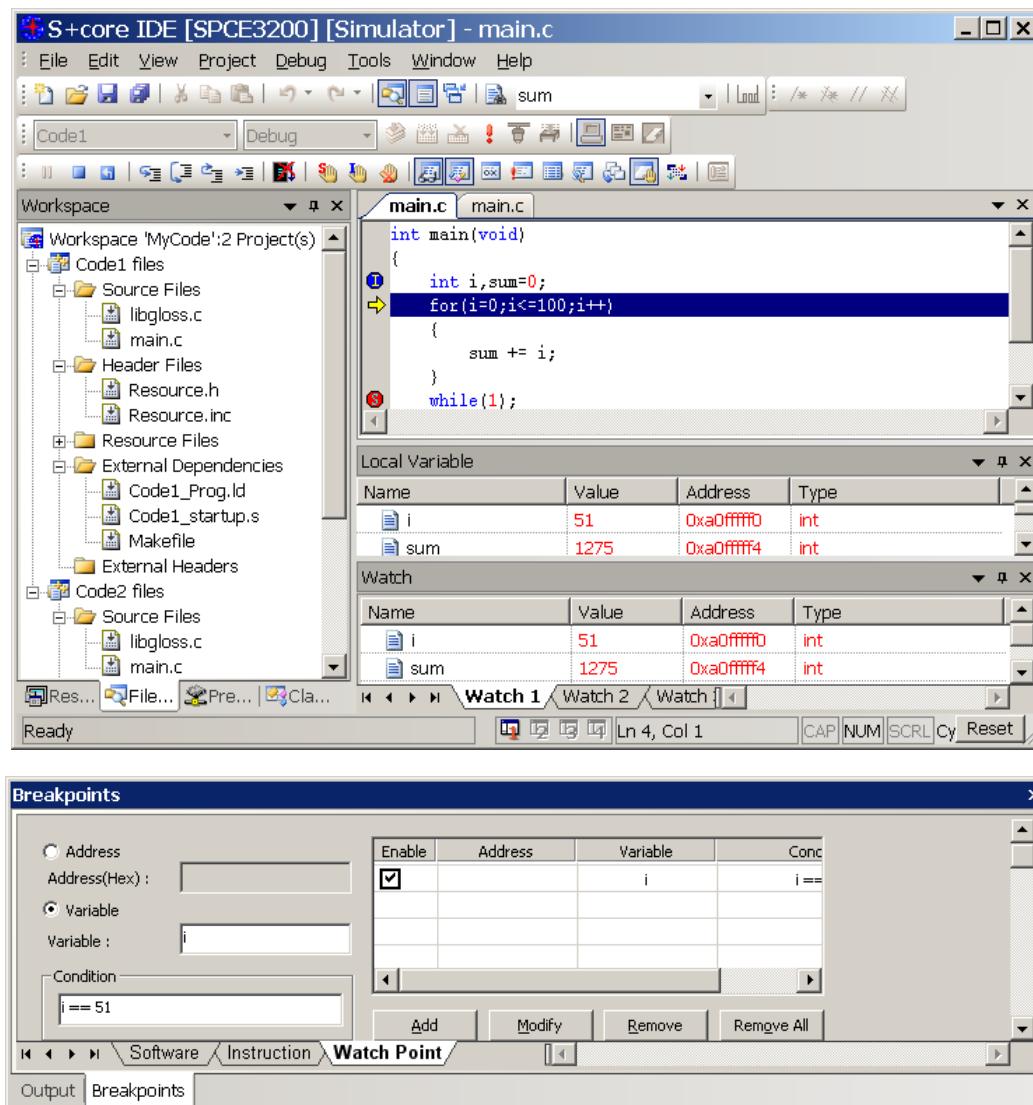


圖 5-60 添加斷點介面

- 第十步：如圖 5-2 連接硬體；如圖 5-61 連接 JP2 中 GPIO2 與 LED1；SPCE3200 的所有跳線和配置開關均為默認設置。

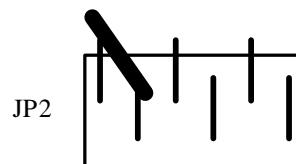


圖 5-61 GPIO2 與 LED1 連接

- 第十一步：同以上步驟類似的方法，選擇 `Code2` 工程，選擇 ICE。編譯鏈接沒有錯誤後，下載到 SPCE3200 開發板上，打開 Memory 視窗，單步運行，觀察 Memory 視窗 `0x8820004C~0x8820004F` 位址單元的資料變化；全速運行，觀察 LED1 的現象。

6 SPCE3200 應用實例

SPCE3200 在多媒體影像處理設備、手持 PDA 設備等領域應用優勢明顯。下麵舉一個基礎應用範例，透過這個範例，讀者可以對 SPCE3200 的應用及開發過程有所瞭解。

6.1 原理概述

本例所設計的是一個透過 UART 遠端控制的具有時間和鬧鈴功能的電子記事本系統。PC 端透過 UART 串列通信匯流排向 SPCE3200 發送命令，SPCE3200 接收命令並完成對內部的時間、鬧鈴和日曆系統的控制。

PC 端透過串聯埠發送的命令支援如下幾種：

- time [年-月-日 時:分:秒]
- alarm [時:分] | [on|off|stop]
- diary [-w|-d]

time 命令可以使 SPCE3200 將其內部時間透過 UART 傳送給 PC 端顯示。當 time 命令後面帶有日期和時間做為參數時，SPCE3200 可以讀取這些參數，並根據參數調整其內部時鐘和日曆系統。

alarm 命令可以查看 SPCE3200 內部的鬧鐘系統的狀態。如果 alarm 命令後面帶有時間做為參數，則 SPCE3200 可以讀取這些參數，並更改其內部的鬧鐘設置。當鬧鐘時間到時，SPCE3200 控制 LED 燈閃爍，以提醒用戶；alarm on 命令可以打開鬧鐘功能；alarm off 命令可以關閉鬧鈴功能；alarm stop 命令可以停止正在閃爍的 LED。

diary 命令可以使 SPCE3200 將非易失性記憶體中的一段文本發送給 PC 端顯示。如果 diary 命令後面帶有 “-w” 參數，系統允許用戶輸入一段文字，並將這段文字存儲到非易失性記憶體中；如果帶有 “-d” 參數，則可以將之前寫入的文字內容刪除。

6.2 應用分析

分析本例的設計要求，可以知道，使用 SPCE3200 內部的 UART 控制器可以完成與 PC 機的通信和交互。而這部分也是整個系統的基礎，其他所有功能，包括接收並解析 PC 端發送過來的命令、向 PC 機回傳資訊燈，都是以 UART 為基礎的。所以，在編寫程式時需要首先初始化 UART 控制器，以便搭建成整個系統的基礎。

時鐘功能可以利用 SPCE3200 內部的 RTC 模組來實現。但是由於 RTC 電路只能產生半秒、秒、分、時等信號，不能完成對日期的記錄和計算，所以，同時需要透過軟體編程來實現日曆功能。

SPCE3200 內部的 RTC 模組還內置了一個時間比較功能，使用該功能，即可完成鬧鐘功能的實現。

根據設計要求，系統需要具備將文本內容保存到非易失性記憶體，並可以重新讀出然後發送給 PC 機的功能。這裏，選用 Nor 型 Flash 記憶體做為保存文本內容的載體，所以，在該系

系統中需要對 Nor 型 Flash 進行編程。

綜上，使用上面提到的幾個模組：UART、RTC、Nor 型 Flash，就可以完成本例要求的功能。

6.3 硬體電路

硬體電路由 LED 控制電路、RS232 電平轉換電路構成。

系統透過使用 GPIO 外接三個 LED 燈做為鬧鐘提示顯示，電路如圖 6-1 所示。IOB3～IOB5 三個 GPIO 透過電阻 R 接三個 LED 發光二極體，鬧鐘時間到時，則使發光二極體閃爍示意。

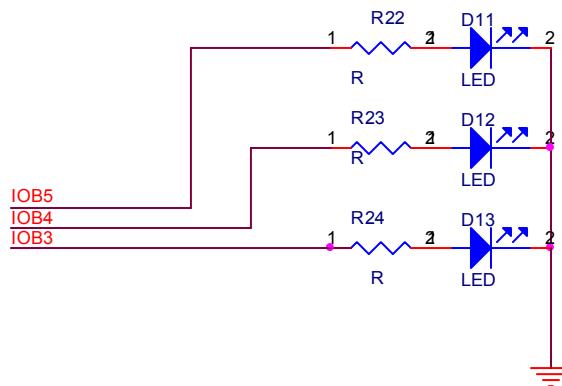


圖 6-1 LED 控制電路原理圖

RS232 電平轉換電路由一片 MAX3232 及其典型應用電路構成，完成 TTL 電平向 232 電平的轉換，以便可以直接和 PC 的串聯埠相連。電路如圖 6-2 所示。

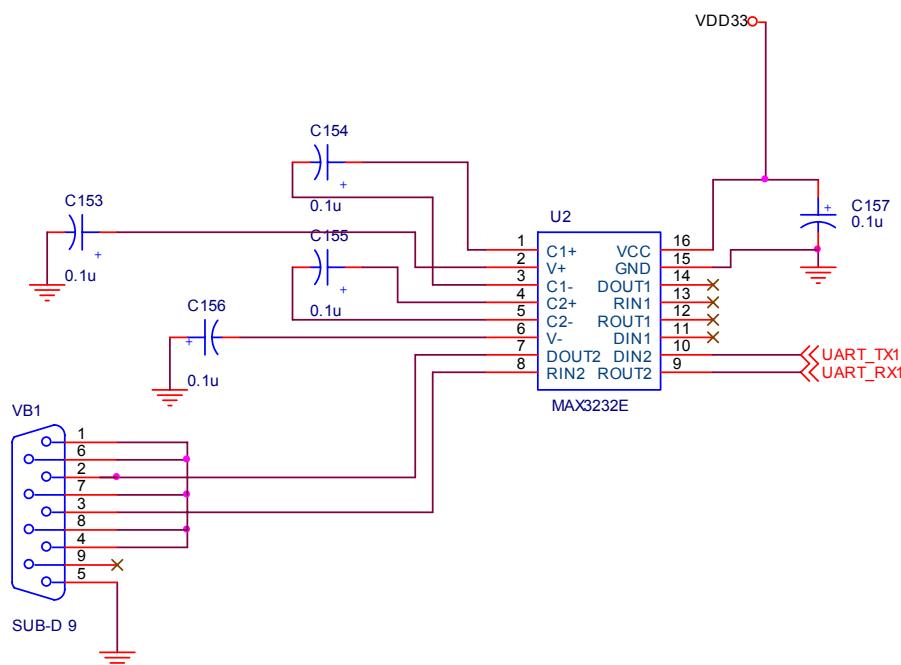


圖 6-2 RS232 電平轉換電路原理圖

用戶在使用 SPCE3200 開發板除錯時，可以將 JP2 的三組跳線短接，並將 SW3 撥碼開關的 7 和 8 兩位置為打開狀態，然後使用串聯埠線將開發板的串聯埠與 PC 機相連，即可完成硬體連接。

6.4 程式設計

整個程式分為主程式、軟體 FIFO 管理程式、UART 收發程式、RTC 控制及日期計算程式、Nor 型 Flash 操作程式、命令獲取和分配程式、命令處理程式等。

6.4.1 主程式

程式按照模組化程式設計，所有功能均透過調用子函數實現，主函數比較簡單，主要完成系統初始化工作，然後便等待接收 PC 端命令並做處理。其流程圖如圖 6-3 所示。

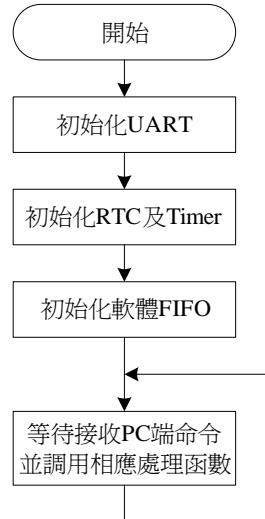


圖 6-3 主程序流程圖

6.4.2 軟體 FIFO 管理程式

PC 端傳遞過來的命令透過 UART 串列接收，此時要求在接收到一定數量的資料之後判斷當前是否是一條有效的命令，而 UART 自帶的緩衝 FIFO 不足以緩存一條命令，所以，有必要使用軟體編程實現一個 FIFO 緩衝區，用來暫時保存 UART 接收到的資料，這樣可以方便命令獲取程式判斷當前是否已經是一條有效的命令。

軟體 FIFO 的基本原理是，首先事先在存儲空間中分配一定數量的存儲單元，然後定義兩個位置計數器，以便記錄當前緩衝區內有效資料的範圍。當向緩衝區填入或者從緩衝區取出資料時，同時移動位置計數器，並判斷是否溢出或清空。

關於軟體 FIFO 的具體實現請參考應用例配套代碼，這裏不再細述。在下面的編程中直接調用軟體 FIFO 模組的 API 介面函數即可，常用軟件 FIFO 函數如表 6-1 所示。

表 6-1 常用软件 FIFO 函數

FIFO_Init()	
語法格式	void FIFO_Init(void)
參 數	無
返 回 值	無
功 能	初始化軟體 FIFO
FIFO_CheckEmpty()	
語法格式	int FIFO_CheckEmpty(void)
參 數	無
返 回 值	0 : FIFO 非空 1 : FIFO 空
功 能	檢查 FIFO 是否為空
FIFO_Pop()	
語法格式	char FIFO_Pop(void)
參 數	無
返 回 值	FIFO 隊首元素
功 能	得到 FIFO 隊首資料，並將其在 FIFO 中刪除
FIFO_Push()	
語法格式	void FIFO_Push(char item)
參 數	char item : 待填入的資料
返 回 值	無
功 能	向 FIFO 中填入一個資料

6.4.3 UART 收發程式

在前面的章節中，介紹了 SPCE3200 內部的 UART 收發器的使用方法。在本例中，將使用中斷方式來接收 PC 端的資料，並使用查詢方式向 PC 機發送回饋資訊。

在 UART 的接收中斷中，需要將接收到的資料保存至軟體 FIFO，以便於此後的處理，如圖 6-4 所示。

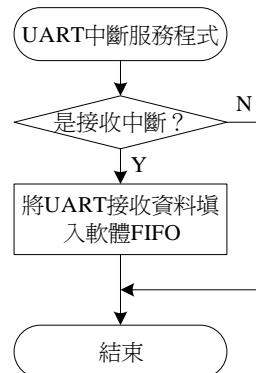


圖 6-4 UART 中斷接收程式流程圖

在後面的程式中可以從 FIFO 中將資料取出，並判斷接收到的命令是否有效。

使用查詢方式發送資料的代碼如下：

```

void out(unsigned char ch)
{
    while(( *P_UART_RX_STATUS & 0x4000) == 0);
    *P_UART_RX_DATA = ch;
}

```

6.4.4 RTC 控制及日期計算程式

該部分主要完成時間日期的計算和管理，以及鬧鐘功能的實現。

SPCE3200 內部的 RTC 模組可以實現時-分-秒的計時和計算，用戶只需要設置與三個時間量相關的三個暫存器 (P_RTC_TIME_HOUR、P_RTC_TIME_MIN、P_RTC_TIME_SEC)，即可為系統設置好初始時間；讀取這三個暫存器，即可獲取當前的系統時間。關於時間的操作相對比較簡單，參考代碼片段如下：

```

typedef struct {                                // 時間結構體
    char Hour;                                 // 時
    char Minute;                               // 分
    char Second;                               // 秒
} TIME;

//=====
// 語法格式： void RTC_GetTime(TIME *tp)
// 實現功能： 得到系統時間
// 參 數： tp - 時間結構體指標
// 返回值： 無
//=====
void RTC_GetTime(TIME *tp)
{
    tp->Hour = *P_RTC_TIME_HOUR;
    tp->Minute = *P_RTC_TIME_MIN;
    tp->Second = *P_RTC_TIME_SEC;
}

```

```

//=====
// 語法格式： void RTC_SetTime(TIME *tp)
// 實現功能： 設置系統時間
// 參 數： tp - 時間結構體指標
// 返回值： 無
//=====

void RTC_SetTime(TIME *tp)
{
    *P_RTC_TIME_HOUR = tp->Hour;
    *P_RTC_TIME_MIN = tp->Minute;
    *P_RTC_TIME_SEC = tp->Second;
}

```

RTC 模組並沒有提供日期計算功能，所以，需要另外編程實現。這裏，首先定義了用於存儲年月日的變數，然後使用了 RTC 模組提供的小時中斷，以便可以在小時數更新的時候判斷當前的日期是否需要更新。RTC 中斷服務程式中關於日期更新的流程如圖 6-5 所示。

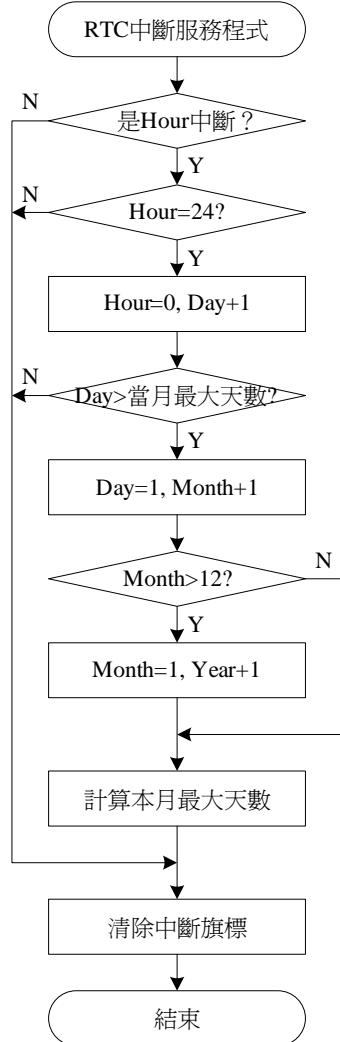


圖 6-5 RTC 中斷服務程式流程圖 (1)

在 RTC 模組中同時提供時間比對（Alarm）功能，當 RTC 的時分秒與預設的 Alarm 時間的時分秒相等時，可以觸發 Alarm 中斷，以便用戶可以處理定時事件。這裏，將使用它實現鬧鐘功能。RTC 模組的 Alarm 中斷服務程式流程如圖 6-6 所示。

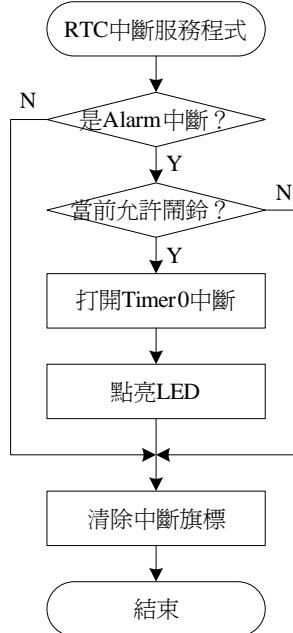


圖 6-6 RTC 中斷服務程式流程圖 (2)

其中，Timer0 的溢出頻率被初始化為 4Hz。在 Timer0 的中斷服務程式中，使 IOB3~IOB5 的輸出資料取反，從而實現 LED 燈的閃爍。同時，在 Timer0 的中斷服務程式中對閃爍次數計數，用來控制 LED 閃爍的總時間。Timer0 的中斷服務程式流程如圖 6-7 所示。

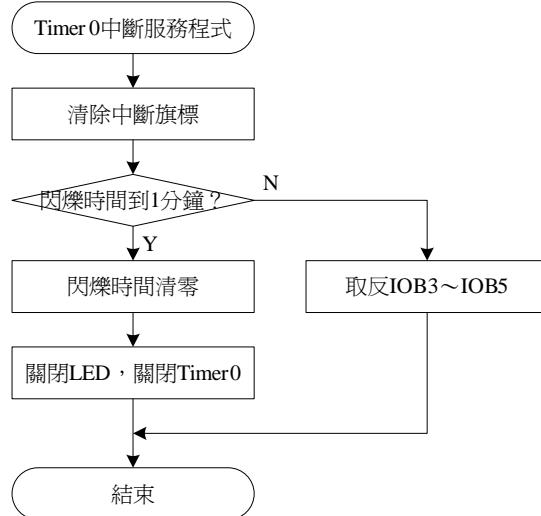


圖 6-7 Timer0 中斷服務程式流程圖

6.4.5 Nor 型 Flash 操作程式

Nor 型 Flash 主要用來保存 PC 端發送記事命令之後發送過來的文本內容。對 Nor 型 Flash

的操作在前面的章節（第 4.12）中已經介紹過，這裏不再贅述。

6.4.6 命令獲取和分配程式

命令獲取程式是與 PC 端交互的重要核心。如何識別當前是否是一條有效的命令是程式設計的關鍵。在本例中，為了簡便起見（同時也可以方便其他擴展），規定從 PC 端發送過來的命令（包括參數）以回車換行符（\r\n）做為結束字元。如，PC 端發送 “alarm on” 命令，實際透過 UART 發送的資料為：0x61,0x6c,0x61,0x72,0x6d,0x20,0x6f,0x6e,0x0d,0x0a。這樣，在程式中就可以接收到一行字元為基準處理資料。接收一行字元的程式流程如圖 6-8 所示。

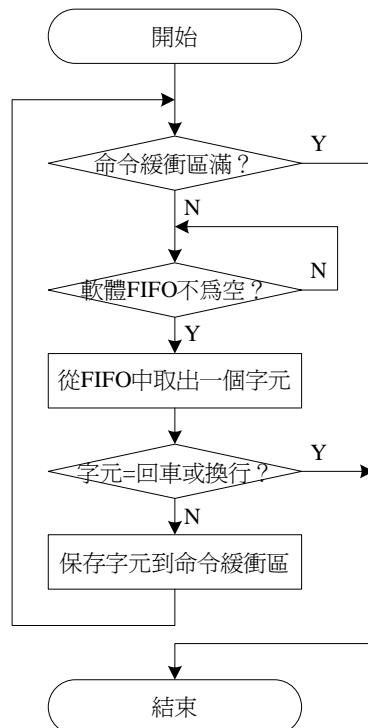


圖 6-8 接收一行字元的程式流程圖

接收到一行字元後，就可以分析其中是否包含有有效的命令，並調用相應的處理函數進行處理。程式流程如圖 6-9 所示。

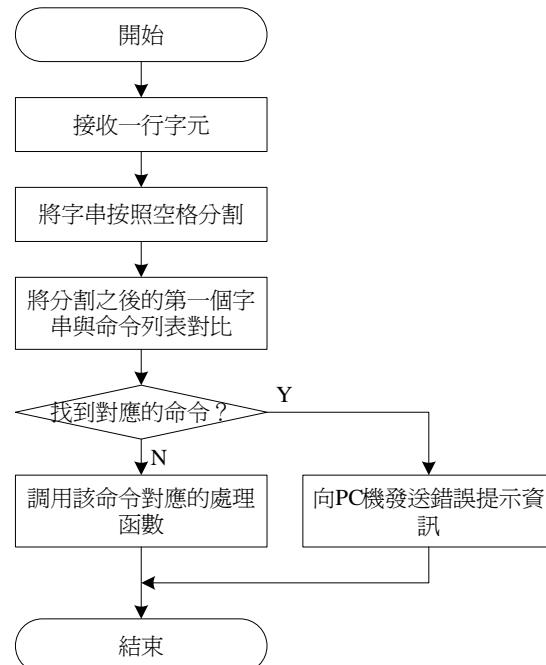


圖 6-9 命令解析分配程式流程圖

6.4.7 命令處理程式

命令處理程式與系統支援的命令一一對應。按照本例的要求，系統支援“time”、“alarm”、“diary”三條命令，所以，對應的處理程式也有三個：時間命令處理程式、鬧鐘命令處理程式和記事本命令處理程式。下麵分別介紹三個處理程式的實現。

1. 時間命令處理程式

在時間處理程式中，首先需要分析 PC 端發送的“time”命令後面是否帶有參數，如果帶有參數，則需要將參數分別轉換為日期資料和時間資料，並分析日期和時間是否合法，如果合法才能對系統時間進行更新，否則，不能接受給出的設置參數。這部分程式流程如圖 6-10 所示。

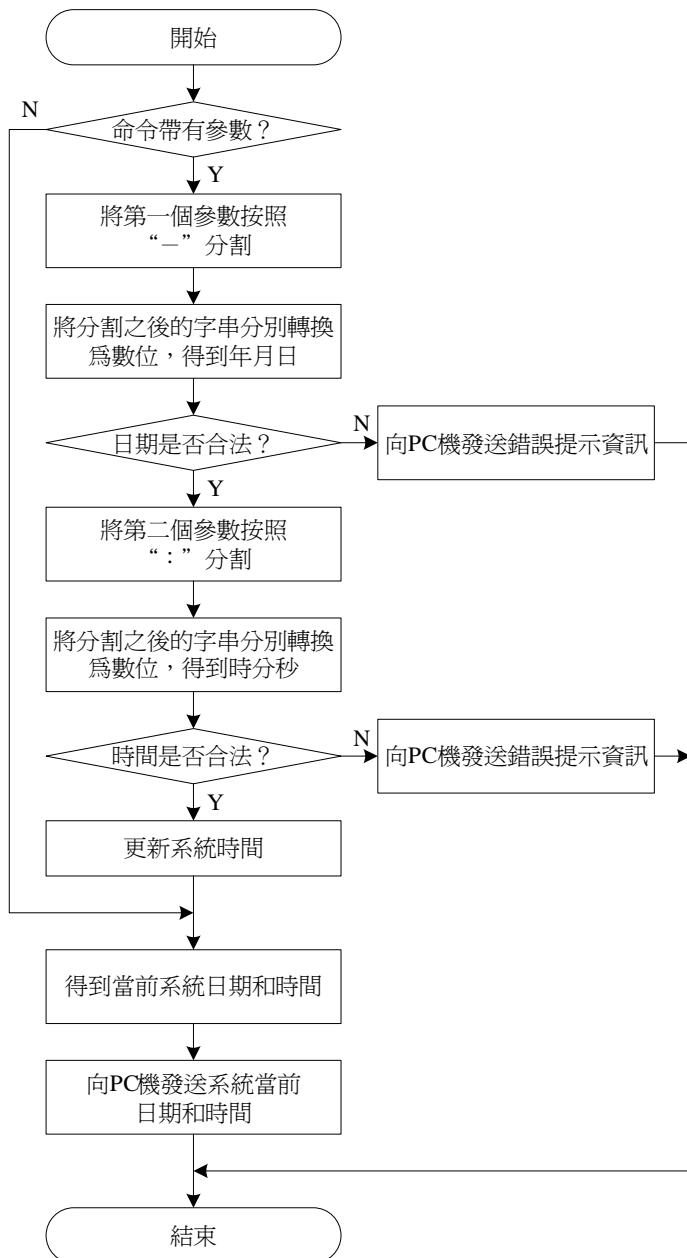


圖 6-10 時間命令處理程式流程圖

2. 鬧鐘命令處理程式

鬧鐘命令可以實現鬧鐘狀態的查詢、對鬧鐘時間的設置、鬧鈴的打開或關閉等，在這個命令的處理程式中，需要分析命令後面所帶的參數的意義，並根據不同的參數做出不同的處理。這部分程式的流程如圖 6-11所示。

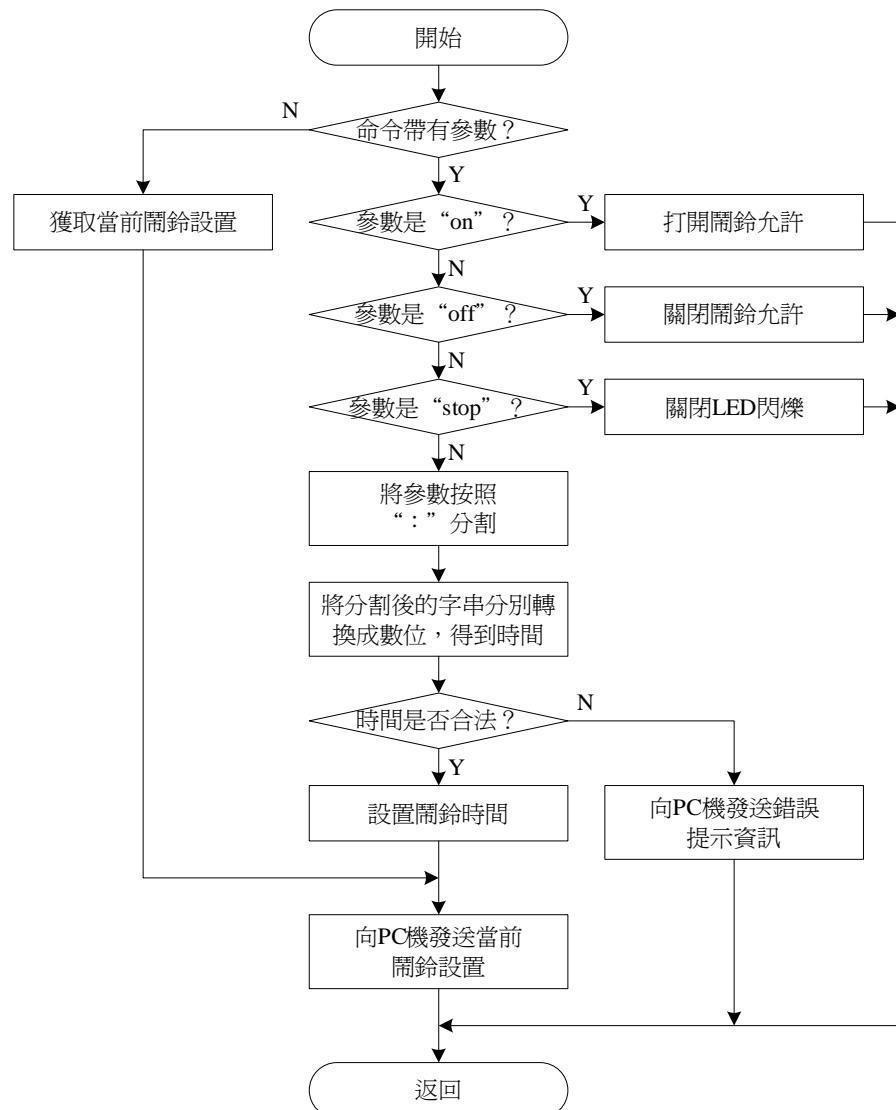


圖 6-11 鬧鐘命令處理程式流程圖

3. 記事本命令處理程式

記事本命令處理程式需要完成記事本內容的顯示或記事本內容的保存。在本例中，記事本的內容從 Nor 型 Flash 的某個扇區開始記錄，為了便於記錄文本資訊同時簡化編程，在本應用例的實現中規定，在扇區的起始 4 個位元組中寫入 “SUN” 表示當前有記事文本記錄，並在接下來的 4 個位元組中保存文本內容的長度，同時規定，文本內容不能超過一個扇區（4KB）大小。

記事本命令處理程式的流程如圖 6-12 所示。

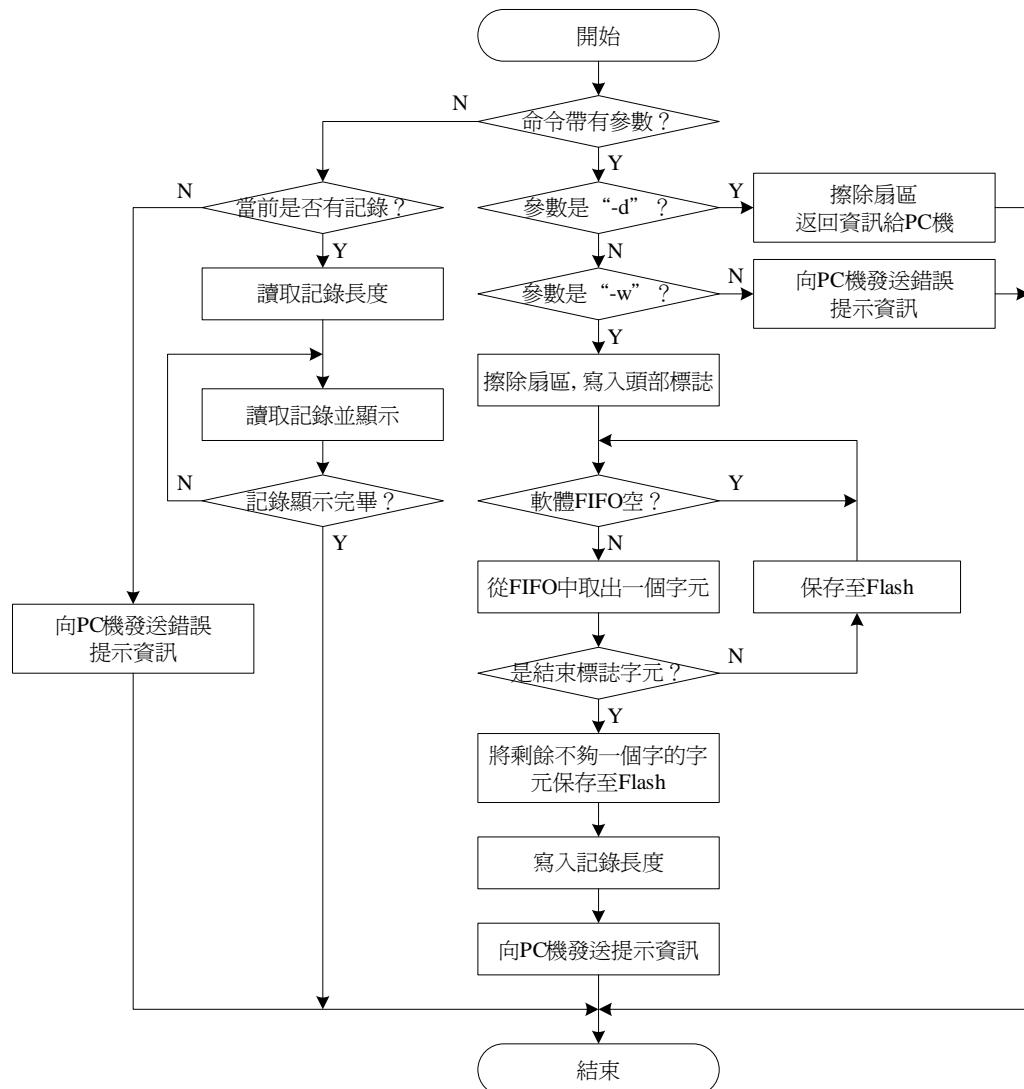


圖 6-12 記事本命令處理程式流程圖

7 附錄

7.1 常用術語、縮寫和約定解釋

7.1.1 術語

表 7-1 書中用到的術語表

序號	術語	解釋
1	RISC	Reduced Instruction Set Computing 的縮寫，精簡指令集。
2	ISA	Instruction Set Architecture 的縮寫，指令集構架。 SPCE3200 採用了凌陽自身的指令構架，支援 32 位與 16 位混合指令模式以及並行條件執行。
3	AMBA	Advanced Microcontroller Bus Architecture 的縮寫，增強型微控制器匯流排構架。 AMBA 協議為設計高性能嵌入式微控制器提供了片上通信的標準，主要包括三種匯流排：AHB、ASB、APB。
4	AHB	Advanced High-performance Bus 的縮寫，增強型高性能匯流排。 AHB 是高性能系統的主幹匯流排，主要連接控制器與片上記憶體、片外記憶體介面或者其他高速低功耗高性能週邊設備單元。
5	ASB	Advanced System Bus 的縮寫，高性能系統匯流排。 ASB 是介於 AHB 與 APB 之間的一種匯流排。
6	APB	Advanced Peripheral Bus 的縮寫，增強型週邊設備匯流排。 APB 主要用來連接一些低功耗週邊設備單元。
7	SOC	System On a Chip 的縮寫，片上系統。
8	SJTAG	Sunplus JTAG 的縮寫。 JTAG 是 Joint Test Action Group 的縮寫，最初用來對晶片進行測試，現在大部分用來實現線上編程。
9	Harvard	哈佛結構，是一種將程式記憶體和資料記憶體分開的結構形式。
10	Byte	位元組，8 位元。
11	Halfword	半字，16 位。
12	Word	字，32 位。

序號	術語	解釋
13	NTSC 制 /PAL 制	NTSC 是 National Television Standards Committee 的縮寫，翻譯為（美國）國家電視標準委員會；PAL 是 Phase Alternating Line 的縮寫，翻譯為逐行倒相。 NTSC 和 PAL 是屬於全球兩大主要的電視廣播制式，其解析度不同，PAL 制式使用的是 720*576，而 NTSC 制式使用的是 760*480。
14	I-Cache	Instruction Cache 的縮寫，指令 cache。
15	D-Cache	Data Cache 的縮寫，資料 cache。
16	DRAM	Dynamic Random Access Memory 的縮寫，動態隨機記憶體。
17	SDRAM	Synchronous DRAM 的縮寫，同步動態隨機記憶體。
18	MMU	Memory Management Unit 的縮寫，記憶體管理單元。
19	LDM	Local Data Memory 的縮寫，本地資料記憶體。
20	LIM	Local Instruction Memory 的縮寫，本地指令記憶體。
21	DMA	Direct Memory Access 的縮寫，直接記憶體存取。 DMA 是實現高速週邊設備與記憶體之間自動成批交換資料而儘量減少 CPU 幫助的輸入輸出操作方式。
22	BLNDMA	Blending and DMA controller 的縮寫，具有混合功能的 DMA 控制器。
23	APBDMA	APB Bridge and DMA Controller 的縮寫，APB 橋和 DMA 控制器。
24	FIFO	First In First Out 的縮寫，先入先出。是實現資料先入先出的記憶體件。
25	PLL	Phase Locked Logic 的縮寫，鎖相環邏輯電路。
26	CKG	Clock Generation 的縮寫，時鐘產生邏輯。
27	MIU	Memory Interface Unit 的縮寫，記憶體介面單元。
28	BUFCTL	Buffer Controller 的縮寫，緩存控制器。
29	GPIO	General-Port for Input/Output 的縮寫，通用輸入輸出埠。
30	Timer	計時器。
31	RTC	Real Time Clock 的縮寫，即時時鐘。
32	TMB	Time Base 的縮寫，時基。
33	WDOG	Watch Dog 的縮寫，看門狗。
34	ADC	Analog to Digital Converter 的縮寫，類比/數位轉換器。
35	MIC	Microphone 的縮寫。
36	DAC	Digital to Analog Converter 的縮寫，數位/類比轉換器。

序號	術語	解釋
37	UART	Universal Asynchronous Receiver/Transmitter 的縮寫，通用非同步收發傳輸器。
38	SPI	Serial Peripheral Interface 的縮寫，串列週邊介面。
39	I2C	Inter-Integrated Circuit 的縮寫。
40	SIO	Serial Interface I/O 的縮寫，串列輸入輸出介面。
41	USB	Universal Serial Bus 的縮寫，通用串列匯流排。
42	SD	Secure Digital Card 的縮寫，安全數碼卡，是一種基於半導體快閃記憶器的新一代記憶設備。
43	TFT	Thin Film Transistor 的縮寫，薄膜場效應電晶體，液晶顯示器上的每一液晶圖元點都是由集成在其後的薄膜電晶體來驅動。
44	STN	Super Twisted Nematic 的縮寫，超扭曲向列。
45	ECC	Error Checking and Correcting 的縮寫，錯誤檢查糾正。
46	CSI	CMOS Sensor Interface 的縮寫，場效應管感測器介面。
47	JPEG	Joint Photographic Experts Group 的縮寫，聯合影像專家組。
48	MPEG4	Moving Picture Experts Group 4 的縮寫，移動影像專家組 4。
49	LVR	Low Voltage Reset 的縮寫，低電壓重設。

7.1.2 縮寫

在 SPCE3200 晶片的暫存器中使用了助記符，有些助記符使用了縮寫，表 7-2列出了縮寫的全稱及解釋：

表 7-2 助記符縮寫、全稱及解釋

序號	縮寫	全稱	解釋
1	CLK	Clock	時鐘
2	PLLV	Video PLL	音頻鎖相環邏輯
3	PLLA	Audio PLL	視頻鎖相環邏輯
4	PLLU	USB PLL	USB 鎖相環邏輯
5	PLLAU	PLLA&PLLU	-
6	32K	32768	32768Hz 晶振
7	CONF	Config	配置

序號	縮寫	全稱	解釋
8	ADDR	Address	位址
9	SA	Start Address	起始位址
10	EA	End Address	結束位址
11	BA	Base Address/Buffer A	基底位址/緩存 A
12	OA	Offset Address	偏移位址
13	BB	Buffer B	緩存 B
14	CTRL	Control	控制
15	SEL	Select	選擇
16	CLR	Clear	清除
17	ERR	Error	錯誤
18	EN	Enable	使能
19	DIS	Disable	禁能
20	RST	Reset	重設
21	TX	Transmit	發送
22	RX	Receive	接收
23	TXRX	Transmit& Receive	收發
24	LP	Line Parity	行檢查
25	CP	Column Parity	列檢查
26	CMP	Compare	比較
27	AINPUT	Analog Input	類比輸入埠
28	REQ	Request	請求
29	PRI	Priority	優先順序
30	KEYC	Key Change	鍵值改變，主要用於鍵喚醒
31	FMT	Format	格式
32	HOR	Horizontal	水平
33	SYNC	Synchronization	同步
34	VER	Vertical	垂直
35	ACT	Active	啟動

序號	縮寫	全稱	解釋
36	COL	Column	列
37	SEQ	Sequence	順序
38	NUM	Number	數字
39	CCP	Capture/Comparison/PWM	擷取/比較/脈寬調製輸出
40	SEC	Second	秒
41	MIN	Minute/Minimal	分/最小
42	ALM	Alarm	報警
43	INT	Interrupt	中斷
44	NAND	Nand Flash	Nand 型 Flash

7.1.3 約定

在 SPCE3200 晶片的暫存器中使用了助記符，並對助記符的欄位進行了約定。下表列出了這些欄位及其約定說明：

表 7-3 助記符的一些約定

序號	英文	約定
1	CONF	Config，配置：主要用於一個時鐘的打開/關閉配置。
2	STATUS	狀態：對於一些既有使能又有狀態或者只有狀態的描述。
3	CTRL	Control，控制：模組使能或者模式的選擇控制。
4	SEL	Select，選擇：主要是對於給定的一些條件的選擇，比如頻率或者介面的選擇。
5	SETUP	設置：用於一些頻率的設置和 GPIO 的設置。
6	INPUT	輸入資料。
7	ADDR	位址。
8	DATA	資料。
9	PULL	上拉或者下拉。
10	COMMAND	命令，給一個暫存器寫一個具體的命令。
11	MODE	模式，如輸入輸出模式等。
12	INTERFACE	介面，主要是一些複用介面。

序號	英文	約定
13	INT	中斷，包括模組的中斷、中斷控制器等。
14	CLR	清除，一般在用在只寫的暫存器。

7.2 CPU 內核暫存器速查表

序號	暫存器名	內核暫存器功能	函數調用時是否被保存
1	r0	通用暫存器 0 用作堆疊指標	是
2	r1	通用暫存器 1 用作暫存器，通常用於組譯器	否
3	r2	通用暫存器 2 用作框（頁）指標	是
4	r3	通用暫存器 3 用作鏈接暫存器。函數調用時，此暫存器保存返回位址	是
5	r4	通用暫存器 4 用於傳遞第 1 個參數及返回值	否
6	r5	通用暫存器 5 用於傳遞第 2 個參數	否
7	r6	通用暫存器 6 用於傳遞第 3 個參數	否
8	r7	通用暫存器 7 用於傳遞第 4 個參數	否
9	r8	通用暫存器 8 用作調用者保存的暫存器	否
10	r9	通用暫存器 9 用作調用者保存的暫存器	否
11	r10	通用暫存器 10 用作調用者保存的暫存器	否
12	r11	通用暫存器 11 用作調用者保存的暫存器	否
13	r12	通用暫存器 12 用作被調用者保存的暫存器，或用於保存框指標	是
14	r13	通用暫存器 13 用作被調用者保存的暫存器，或用於保存框指標	是
15	r14	通用暫存器 14 用作被調用者保存的暫存器，或用於保存框指標	是
16	r15	通用暫存器 15 用作被調用者保存的暫存器，或用於保存框指標	是
17	r16	通用暫存器 16 用作被調用者保存的暫存器，或用於保存框指標	是
18	r17	通用暫存器 17 用作被調用者保存的暫存器，或用於保存框指標	是

序號	暫存器名	內核暫存器功能	函數調用時是否被保存
19	r18	通用暫存器 18 用作被調用者保存的暫存器，或用於保存框指標	是
20	r19	通用暫存器 19 用作被調用者保存的暫存器，或用於保存框指標	是
21	r20	通用暫存器 20 用作被調用者保存的暫存器，或用於保存框指標	是
22	r21	通用暫存器 21 用作被調用者保存的暫存器，或用於保存框指標	是
23	r22	通用暫存器 22 用作調用者保存的暫存器	否
24	r23	通用暫存器 23 用作調用者保存的暫存器	否
25	r24	通用暫存器 24 用作調用者保存的暫存器	否
26	r25	通用暫存器 25 用作調用者保存的暫存器	否
27	r26	通用暫存器 26 用作調用者保存的暫存器	否
28	r27	通用暫存器 27 用作調用者保存的暫存器	否
29	r28	通用暫存器 28 用作全局指標	是
30	r29	通用暫存器 29，為編譯器保留	是
31	r30	通用暫存器 30 僅為作業系統使用	是
32	r31	通用暫存器 31 僅為作業系統使用	是
33	CEH	乘/除法運算特殊暫存器，保存乘法運算結果的高 32 位或除法結果的餘數	否
34	CEL	乘/除法運算特殊暫存器，保存乘法運算結果的低 32 位或除法結果的商	否
35	Sr0(CNT)	特殊功能暫存器 0 用於計數器操作	否
36	Sr1(LCR)	特殊功能暫存器 1 用於裝載合併指令操作	否
37	Sr2(SCR)	特殊功能暫存器 2 用於存儲合併指令操作	否
38	Cr0(PSR)	程式狀態暫存器，主要用來進行異常使能，Endian 的選擇等。	-
39	Cr1(Condition)	條件暫存器，N、Z、C 等條件旗標位元。	-
40	Cr2(ECR)	異常原因暫存器	-
41	Cr3(EXCPVec)	異常向量暫存器	-
42	Cr4(CCR)	Cache 控制暫存器	-
43	Cr5(EPC)	異常程式計數器	-

序號	暫存器名	內核暫存器功能	函數調用時是否被保存
44	Cr6(EMA)	異常記憶體位址暫存器	-
45	Cr7(TLBLOCK)	TLB 時鐘暫存器	-
46	Cr8(TLBPT)	TLB 指標暫存器	-
47	Cr9(PEADDR)	頁入口位址暫存器	-
48	Cr10(TLBRPT)	TLB 隨機指標暫存器	-
49	Cr11(PEVN)	頁入口的實際頁數暫存器	-
50	Cr12(PECTX)	頁入口內容暫存器	-
51	Cr13	-	-
52	Cr14	-	-
53	Cr15	LIM 實體框號	-
54	Cr16	LDM 實體框號暫存器	-
55	Cr17	-	-
56	Cr18(Prev)	Prev 暫存器，保存處理器的版本及修訂資訊	-
57	Cr29(DREG)	DREG 暫存器，保留 Debug 的一些資訊	-
58	Cr30(DEPC)	Debug 異常程式計數器暫存器	-
59	Cr31(DSAVE)	Debug 異常內容保存暫存器	-

7.3 硬體模組暫存器速查表

表 7-4 硬體模組暫存器速查表

序號	暫存器名稱	助記符	位址
時鐘模組			
1	CPU 時鐘選擇暫存器	P_CLK_CPU_SEL	0x88210004
2	AHB 汇流排時鐘配置暫存器	P_CLK_AHB_CONF	0x88210008
3	AHB 汇流排時鐘選擇暫存器	P_CLK_AHB_SEL	0x8821000C
4	PLLV 時鐘配置暫存器	P_CLK_PLLV_CONF	0x882100B4
5	PLLV 時鐘選擇暫存器	P_CLK_PLLV_SEL	0x882100B8
6	PLLAU 時鐘配置暫存器	P_CLK_PLLAU_CONF	0x882100bc

序號	暫存器名稱	助記符	位址
7	32K 即時時鐘配置暫存器	P_CLK_32K_CONF	0x88210114
8	PLLV 穩定輸出時間暫存器	P_PLLV_STABLE_TIME	0x88210104
中斷控制器模組			
9	中斷時鐘配置暫存器	P_INT_CLK_CONF	0x882100D0
10	中斷請求狀態暫存器 1	P_INT_REQ_STATUS1	0x880a0000
11	中斷請求狀態暫存器 2	P_INT_REQ_STATUS2	0x880a0004
12	中斷優先順序暫存器	P_INT_GROUP_PRI	0x880a0008
13	第 0 組中斷優先順序暫存器	P_INT_GROUP0_PRI	0x880a0010
14	第 1 組中斷優先順序暫存器	P_INT_GROUP1_PRI	0x880a0014
15	第 2 組中斷優先順序暫存器	P_INT_GROUP2_PRI	0x880a0018
16	第 3 組中斷優先順序暫存器	P_INT_GROUP3_PRI	0x880a001C
17	中斷控制暫存器 1	P_INT_MASK_CTRL1	0x880a0020
18	中斷控制暫存器 2	P_INT_MASK_CTRL2	0x880a0024
MIU 模組			
19	MIU 時鐘配置暫存器	P_MIU_CLK_CONF	0x88210010
20	SDRAM 電源暫存器	P_MIU_SDRAM_POWER	0x8807005C
21	SDRAM 參數設置暫存器 1	P_MIU_SDRAM_SETUP1	0x88070060
22	SDRAM 狀態暫存器	P_MIU_SDRAM_STATUS	0x8807006C
23	SDRAM 參數設置暫存器 2	P_MIU_SDRAM_SETUP2	0x88070094
BUFFER CONTROL 模組			
24	BUFCTRL 時鐘配置暫存器	P_BUFCtrl_CLK_CONF	0x882100C8
25	BUFCTRL 控制暫存器 1	P_BUFCtrl_MODE_CTRL1	0x88090000
26	BUFCTRL 緩衝區選擇暫存器	P_BUFCtrl_BUFFER_SEL	0x88090004
27	BUFCTRL 中斷控制暫存器	P_BUFCtrl_INT_CTRL	0x8809004C
28	BUFCTRL 框丟失中斷控制暫存器	P_BUFCtrl_FRAMELOSS_IN	0x88090028
29	BUFCTRL 控制暫存器 2	P_BUFCtrl_MODE_CTRL2	0x8809003C
30	BUFCTRL 控制暫存器 3	P_BUFCtrl_MODE_CTRL3	0x88090040

序號	暫存器名稱	助記符	位址
APB DMA 模組			
31	DMA 時鐘配置及存器	P_DMA_CLK_CONF	0x88210058
32	DMA 忙狀態暫存器	P_DMA_BUSY_STATUS	0x88080000
33	DMA 中斷狀態暫存器	P_DMA_INT_STATUS	0x88080004
34	AHB DMA 第 0 通道緩衝區 A 起始位址	P_DMA_AHB_SA0BA	0x88080008
35	AHB DMA 第 1 通道緩衝區 A 起始位址	P_DMA_AHB_SA1BA	0x8808000C
36	AHB DMA 第 2 通道緩衝區 A 起始位址	P_DMA_AHB_SA2BA	0x88080010
37	AHB DMA 第 3 通道緩衝區 A 起始位址	P_DMA_AHB_SA3BA	0x88080014
38	AHB DMA 第 0 通道緩衝區 A 結束位址	P_DMA_AHB_EA0BA	0x88080018
39	AHB DMA 第 1 通道緩衝區 A 結束位址	P_DMA_AHB_EA1BA	0x8808001C
40	AHB DMA 第 2 通道緩衝區 A 結束位址	P_DMA_AHB_EA2BA	0x88080020
41	AHB DMA 第 3 通道緩衝區 A 結束位址	P_DMA_AHB_EA3BA	0x88080024
42	APB DMA 第 0 通道起始位址	P_DMA_APB_SA0	0x88080028
43	APB DMA 第 1 通道起始位址	P_DMA_APB_SA1	0x8808002C
44	APB DMA 第 2 通道起始位址	P_DMA_APB_SA2	0x88080030
45	APB DMA 第 3 通道起始位址	P_DMA_APB_SA3	0x88080034
46	AHB DMA 第 0 通道緩衝區 B 起始位址	P_DMA_AHB_SA0BB	0x8808004C
47	AHB DMA 第 1 通道緩衝區 B 起始位址	P_DMA_AHB_SA1BB	0x88080050
48	AHB DMA 第 2 通道緩衝區 B 起始位址	P_DMA_AHB_SA2BB	0x88080054
49	AHB DMA 第 3 通道緩衝區 B 起始位址	P_DMA_AHB_SA3BB	0x88080058
50	AHB DMA 第 0 通道緩衝區 B 結束位址	P_DMA_AHB_EA0BB	0x8808005C
51	AHB DMA 第 1 通道緩衝區 B 結束位址	P_DMA_AHB_EA1BB	0x88080060
52	AHB DMA 第 2 通道緩衝區 B 結束位址	P_DMA_AHB_EA2BB	0x88080064
53	AHB DMA 第 3 通道緩衝區 B 結束位址	P_DMA_AHB_EA3BB	0x88080068
54	DMA 第 0 通道控制暫存器	P_DMA_CHANNEL0_CTRL	0x8808006C
55	DMA 第 1 通道控制暫存器	P_DMA_CHANNEL1_CTRL	0x88080070
56	DMA 第 2 通道控制暫存器	P_DMA_CHANNEL2_CTRL	0x88080074
57	DMA 第 3 通道控制暫存器	P_DMA_CHANNEL3_CTRL	0x88080078

序號	暫存器名稱	助記符	位址
58	DMA 通道重設暫存器	P_DMA_CHANNEL_RESET	0x8808007C
GPIO			
59	GPIO 時鐘配置暫存器	P_GPIO_CLK_CONF	0x882100FC
60	IOA 設置暫存器	P_IOA_GPIO_SETUP	0x88200038
61	IOB 設置暫存器	P_IOB_GPIO_SETUP	0x8820004C
62	IOB 輸入資料暫存器	P_IOB_GPIO_INPUT	0x88200070
63	IOA 輸入資料暫存器	P_IOA_GPIO_INPUT	0x88200074
64	IOA 外部中斷暫存器	P_IOA_GPIO_INT	0x88200090
Timer 模組			
65	TIMER 時鐘選擇暫存器	P_TIMER_CLK_SEL	0x882100E4
66	TIMER 介面選擇暫存器	P_TIMER_INTERFACE_SEL	0x88200010
67	TIMER0 時鐘配置暫存器	P_TIMER0_CLK_CONF	0x8821006C
68	TIMER0 控制暫存器	P_TIMER0_MODE_CTRL	0x88160000
69	TIMER0 CCP 控制暫存器	P_TIMER0_CCP_CTRL	0x88160004
70	TIMER0 計數初值資料暫存器	P_TIMER0_PRELOAD_DATA	0x88160008
71	TIMER0 CCP 計數初值資料暫存器	P_TIMER0_CCP_DATA	0x8816000C
72	TIMER0 計數暫存器	P_TIMER0_COUNT_DATA	0x88160010
73	TIMER1 時鐘配置暫存器	P_TIMER1_CLK_CONF	0x88210070
74	TIMER1 控制暫存器	P_TIMER1_MODE_CTRL	0x88161000
75	TIMER1 CCP 控制暫存器	P_TIMER1_CCP_CTRL	0x88161004
76	TIMER1 計數初值資料暫存器	P_TIMER1_PRELOAD_DATA	0x88161008
77	TIMER1 CCP 計數初值資料暫存器	P_TIMER1_CCP_DATA	0x8816100C
78	TIMER1 計數暫存器	P_TIMER1_COUNT_DATA	0x88161010
79	TIMER2 時鐘配置暫存器	P_TIMER2_CLK_CONF	0x88210074
80	TIMER2 控制暫存器	P_TIMER2_MODE_CTRL	0x88162000
81	TIMER2 CCP 控制暫存器	P_TIMER2_CCP_CTRL	0x88162004
82	TIMER2 計數初值資料暫存器	P_TIMER2_PRELOAD_DATA	0x88162008
83	TIMER2 CCP 計數初值資料暫存器	P_TIMER2_CCP_DATA	0x8816200C

序號	暫存器名稱	助記符	位址
84	TIMER2 計數暫存器	P_TIMER2_COUNT_DATA	0x88162010
85	TIMER3 時鐘配置暫存器	P_TIMER3_CLK_CONF	0x88200078
86	TIMER3 控制暫存器	P_TIMER3_MODE_CTRL	0x88163000
87	TIMER3 CCP 控制暫存器	P_TIMER3_CCP_CTRL	0x88163004
88	TIMER3 計數初值資料暫存器	P_TIMER3_PRELOAD_DATA	0x88163008
89	TIMER3 CCP 計數初值資料暫存器	P_TIMER3_CCP_DATA	0x8816300C
90	TIMER3 計數暫存器	P_TIMER3_COUNT_DATA	0x88163010
91	TIMER4 時鐘配置暫存器	P_TIMER4_CLK_CONF	0x8821007C
92	TIMER4 控制暫存器	P_TIMER4_MODE_CTRL	0x88164000
92	TIMER4 CCP 控制暫存器	P_TIMER4_CCP_CTRL	0x88164004
94	TIMER4 計數初值資料暫存器	P_TIMER4_PRELOAD_DATA	0x88164008
95	TIMER4 CCP 計數初值資料暫存器	P_TIMER4_CCP_DATA	0x8816400C
96	TIMER4 計數暫存器	P_TIMER4_COUNT_DATA	0x88164010
97	TIMER5 時鐘配置暫存器	P_TIMER5_CLK_CONF	0x88210080
98	TIMER5 控制暫存器	P_TIMER5_MODE_CTRL	0x88165000
99	TIMER5 CCP 控制暫存器	P_TIMER5_CCP_CTRL	0x88165004
100	TIMER5 計數初值資料暫存器	P_TIMER5_PRELOAD_DATA	0x88165008
101	TIMER5 CCP 計數初值資料暫存器	P_TIMER5_CCP_DATA	0x8816500C
102	TIMER5 計數暫存器	P_TIMER5_COUNT_DATA	0x88165010

即時時鐘模組

103	RTC 時鐘配置暫存器	P_RTC_CLK_CONF	0x88210088
104	RTC 秒暫存器	P_RTC_TIME_SEC	0x88166000
105	RTC 分暫存器	P_RTC_TIME_MIN	0x88166004
106	RTC 時暫存器	P_RTC_TIME_HOUR	0x88166008
107	RTC 秒報警暫存器	P_RTC_ALM_SEC	0x8816600C
108	RTC 分報警暫存器	P_RTC_ALM_MIN	0x88166010
109	RTC 時報警暫存器	P_RTC_ALM_HOUR	0x88166014
110	RTC 控制暫存器	P_RTC_MODE_CTRL	0x88166018

序號	暫存器名稱	助記符	位址
111	RTC 中斷狀態暫存器	P_RTC_INT_STATUS	0x8816601C
時基模組			
112	TMB 時鐘配置暫存器	P_TMB_CLK_CONF	0x882100E0
113	TMB 控制暫存器	P_TMB_MODE_CTRL	0x88166020
114	TMB 中斷狀態暫存器	P_TMB_INT_STATUS	0x88166024
115	TMB 重設命令暫存器	P_TMB_RESET_COMMAND	0x88166028
WDOG 模組			
116	WDOG 時鐘配置暫存器	P_WDOG_CLK_CONF	0x88210084
117	WDOG 重設模式暫存器	P_WDOG_RESET_STATUS	0x882100E8
118	WDOG 控制暫存器	P_WDOG_MODE_CTRL	0x88170000
119	WDOG 重設週期設置暫存器	P_WDOG_CYCLE_SETUP	0x88170004
120	WDOG 清狗命令暫存器	P_WDOG_CLR_COMMAND	0x88170008
ADC 模組			
121	ADC 時鐘配置暫存器	P_ADC_CLK_CONF	0x882100AC
122	ADC 時鐘選擇暫存器	P_ADC_CLK_SEL	0x882100B0
123	ADC GPIO 設置暫存器	P_ADC_GPIO_SETUP	0x88200048
124	ADC GPIO 輸入資料暫存器	P_ADC_GPIO_INPUT	0x88200078
125	ADC GPIO 外部中斷暫存器	P_ADC_GPIO_INT	0x8820009C
126	ADC 類比輸入控制暫存器	P_ADC_AINPUT_CTRL	0x88200054
127	ADC 控制暫存器 1	P_ADC_MIC_CTRL1	0x881a0000
128	MIC 增益暫存器	P_ADC_MIC_GAIN	0x881a0004
129	ADC 取樣時鐘暫存器	P_ADC_SAMPLE_CLK	0x881a0008
130	ADC 取樣保持暫存器	P_ADC_SAMPLE_HOLD	0x881a000c
131	ADC 控制暫存器 2	P_ADC_MIC_CTRL2	0x881a0010
132	ADC 中斷狀態暫存器	P_ADC_INT_STATUS	0x881a0014
133	ADC 手動方式資料暫存器	P_ADC_MANUAL_DATA	0x881a0018
134	ADC 自動方式資料暫存器	P_ADC_AUTO_DATA	0x881a001c
135	MIC 轉換資料暫存器	P_ADC_MIC_DATA	0x881a0020

序號	暫存器名稱	助記符	位址
UART 模組			
136	UART 時鐘配置暫存器	P_UART_CLK_CONF	0x8821005C
137	UART 介面選擇暫存器	P_UART_INTERFACE_SEL	0x88200000
138	UART GPIO 設置暫存器	P_UART_GPIO_SETUP	0x88200040
139	UART GPIO 輸入資料暫存器	P_UART_GPIO_INPUT	0x88200074
140	UART GPIO 外部中斷暫存器	P_UART_GPIO_INT	0x88200094
141	UART 控制暫存器	P_UART_MODE_CTRL	0x88150008
142	UART 串列傳輸速率設置暫存器	P_UART_BAUDRATE_SETUP	0x8815000C
143	UART 發收狀態暫存器	P_UART_TXRX_STATUS	0x88150010
144	UART 錯誤狀態暫存器	P_UART_ERR_STATUS	0x88150004
145	UART 發收資料暫存器	P_UART_TXRX_DATA	0x88150000
146	UART 哨兵狀態暫存器	P_UART_WAKEUP_STATUS	0x88210110
SPI 模組			
147	SPI 時鐘配置暫存器	P_SPI_CLK_CONF	0x88210098
148	SPI 介面選擇暫存器	P_SPI_INTERFACE_SEL	0x882000A4
149	SPI 控制暫存器	P_SPI_MODE_CTRL	0x88110000
150	SPI 發送狀態暫存器	P_SPI_TX_STATUS	0x88110004
151	SPI 發送資料暫存器	P_SPI_TX_DATA	0x88110008
152	SPI 接收狀態暫存器	P_SPI_RX_STATUS	0x8811000C
153	SPI 接收資料暫存器	P_SPI_RX_DATA	0x88110010
154	SPI 發收狀態暫存器	P_SPI_TXRX_STATUS	0x88110014
I2C 模組			
155	I2C 時鐘配置暫存器	P_I2C_CLK_CONF	0x88210094
156	I2C 介面選擇暫存器	P_I2C_INTERFACE_SEL	0x88200004
157	I2C GPIO 設置暫存器	P_I2C_GPIO_SETUP	0x88200044
158	I2C GPIO 輸入資料暫存器	P_I2C_GPIO_INPUT	0x88200074
159	I2C GPIO 外部中斷暫存器	P_I2C_GPIO_INT	0x88200098
160	I2C 控制暫存器	P_I2C_MODE_CTRL	0x88130020

序號	暫存器名稱	助記符	位址
161	I2C 中斷狀態暫存器	P_I2C_INT_STATUS	0x88130024
162	I2C 傳輸速率設置暫存器	P_I2C_RATE_SETUP	0x88130028
163	I2C 從機位址暫存器	P_I2C_SLAVE_ADDR	0x8813002C
164	I2C 資料位址暫存器	P_I2C_DATA_ADDR	0x88130030
165	I2C 發送資料暫存器	P_I2C_TX_DATA	0x88130034
166	I2C 接收資料暫存器	P_I2C_RX_DATA	0x88130038
SIO 模組			
167	SIO 介面選擇暫存器	P_SIO_INTERFACE_SEL	0x88200004
168	JTAG GPIO 設置暫存器	P_JTAG_GPIO_SETUP	0x88200034
169	JTAG GPIO 輸入資料暫存器	P_JTAG_GPIO_INPUT	0x88200070
170	JTAG GPIO 外部中斷暫存器	P_JTAG_GPIO_INT	0x8820008C
171	SIO 時鐘配置暫存器	P_SIO_CLK_CONF	0x882100A0
172	SIO 控制暫存器	P_SIO_MODE_CTRL	0X88120000
173	SIO 自動傳輸控制暫存器	P_SIO_AUTO_CTRL	0x88210004
174	SIO 資料位址暫存器	P_SIO_DATA_ADDR	0x88210008
175	SIO 收發資料暫存器	P_SIO_TXRX_DATA	0x8821000C
NAND Flash 模組			
176	NAND 介面選擇暫存器	P_NAND_INTERFACE_SEL	0x882000A4
177	NAND GPIO 設置暫存器	P_NAND_GPIO_SETUP	0x8820002C
178	NAND GPIO 上下拉暫存器	P_NAND_GPIO_PULL	0x88200030
179	NAND GPIO 輸入資料暫存器	P_NAND_GPIO_INPUT	0x8820006C
180	NAND GPIO 外部中斷暫存器	P_NAND_GPIO_INT	0x88200088
181	NAND 時鐘配置暫存器	P_NAND_CLK_CONF	0x882100A8
182	NAND 中斷控制暫存器	P_NAND_INT_CTRL	0x88190014
183	NAND 控制暫存器	P_NAND_MODE_CTRL	0x88190000
184	NAND 命令暫存器	P_NAND_CLE_COMMAND	0x88190004
185	NAND 位址暫存器	P_NAND_ALE_ADDR	0x88190008
186	NAND 發送資料暫存器	P_NAND_TX_DATA	0x8819000C

序號	暫存器名稱	助記符	位址
187	NAND 接收資料暫存器	P_NAND_RX_DATA	0x88190010
188	NAND 中斷狀態暫存器	P_NAND_INT_STATUS	0x88190018
189	NAND 真實行檢查碼暫存器	P_NAND_ECC_TRUELP	0x8819001C
190	NAND 真實列檢查碼暫存器	P_NAND_ECC_TRUECP	0x88190020
191	NAND 對比行檢查碼暫存器	P_NAND_ECC_CMPLP	0x88190024
192	NAND 對比列檢查碼暫存器	P_NAND_ECC_CMPCP	0x88190028
192	NAND 檢查狀態暫存器	P_NAND_ECC_STATUS	0x8819002C
194	NAND 檢查控制暫存器	P_NAND_ECC_CTRL	0x88190030
SD Card 模組			
195	SD 介面選擇暫存器	P_SD_INTERFACE_SEL	0x882000A4
196	SD 時鐘配置暫存器	P_SD_CLK_CONF	0x882100A4
197	SD 控制暫存器	P_SD_MODE_CTRL	0x88180018
198	SD 中斷控制暫存器	P_SD_INT_CTRL	0x8818001C
199	SD 中斷狀態暫存器	P_SD_INT_STATUS	0x88180014
200	SD 命令設置暫存器	P_SD_COMMAND_SETUP	0x88180008
201	SD 參數資料暫存器	P_SD_ARGUMENT_DATA	0x8818000C
202	SD 回應資料暫存器	P_SD_RESPONSE_DATA	0x88180010
203	SD 發送資料暫存器	P_SD_TX_DATA	0x88180000
204	SD 接收資料暫存器	P_SD_RX_DATA	0x88180004
TFT LCD 模組			
205	TFT 控制暫存器	P_TFT_MODE_CTRL	0x88040000
206	TFT 輸出資料格式暫存器	P_TFT_DATA_FMT	0x88040004
207	TFT 水平顯示掃描暫存器	P_TFT_HOR_ACT	0x88040008
208	TFT 水平空白前掃暫存器	P_TFT_HOR_FRONT	0x8804000C
209	TFT 水平空白後掃暫存器	P_TFT_HOR_BACK	0x88040010
210	TFT 水平同步掃描暫存器	P_TFT_HOR_SYNC	0x88040014
211	TFT 垂直顯示掃描暫存器	P_TFT_VER_ACT	0x88040018
212	TFT 垂直空白前掃暫存器	P_TFT_VER_FRONT	0x8804001C

序號	暫存器名稱	助記符	位址
213	TFT 垂直空白後掃暫存器	P_TFT_VER_BACK	0x88040020
214	TFT 垂直同步掃描暫存器	P_TFT_VER_SYNC	0x88040024
215	TFT 框資料格式暫存器 1	P_TFT_FRAME_FMT1	0x88040028
216	TFT 起始行暫存器	P_TFT_ROW_START	0x8804002C
217	TFT 起始列暫存器	P_TFT_COL_START	0x88040030
218	TFT 列寬度暫存器	P_TFT_COL_WIDTH	0x88040034
219	TFT 餘量寬度暫存器	P_TFT_DUMMY_WIDTH	0x88040038
220	TFT 狀態暫存器	P_TFT_BUFFER_STATUS	0x8804003C
221	TFT 輸出資料順序暫存器	P_TFT_DATA_SEQ	0x88040040
222	TFT 中斷狀態暫存器	P_TFT_INT_STATUS	0x88040050
223	TFT 框資料格式暫存器 2	P_TFT_FRAME_FMT2	0x880400A0
224	LCD 時鐘配置暫存器	P_LCD_CLK_CONF	0x88210034
225	LCD 時鐘選擇暫存器	P_LCD_CLK_SEL	0x88210038
226	LCD 介面選擇暫存器	P_LCD_INTERFACE_SEL	0x88200000
227	LCD 顯示緩衝區起始位址暫存器 1	P_LCD_BUFFER_SA1	0x8807000C
228	LCD 顯示緩衝區起始位址暫存器 2	P_LCD_BUFFER_SA2	0x88070010
229	LCD 顯示緩衝區起始位址暫存器 3	P_LCD_BUFFER_SA3	0x88070014
230	LCD 顯示緩衝區選擇暫存器	P_LCD_BUFFER_SEL	0x88090024
231	TFT GPIO 輸出資料暫存器	P_TFT_GPIO_DATA	0x88200014
232	TFT GPIO 輸出使能暫存器	P_TFT_GPIO_OUTPUTEN	0x88200018
233	TFT GPIO 上拉暫存器	P_TFT_GPIO_PULLUP	0x8820001C
234	TFT GPIO 下拉暫存器	P_TFT_GPIO_PULLDOWN	0x88200020
235	TFT GPIO 輸入資料暫存器	P_TFT_GPIO_INPUT	0x88200064
236	TFT GPIO 外部中斷暫存器	P_TFT_GPIO_INT	0x88200080

7.4 S+core7 指令速查表

7.4.1 32 位指令速查表

表 7-5 32 位指令速查表

序號	指令	功能說明	語法格式
1	lb	裝載位元組資料	lb rD, [rA, SIimm15]
		裝載位元組資料(後變址定址)	lb rD, [rA]+, SIimm12
		裝載位元組資料(前變址定址)	lb rD, [rA, SIimm12]+
2	lbu	裝載無符號位元組資料	lbu rD, [rA, SIimm15]
		裝載無符號位元組資料(後變址定址)	lbu rD, [rA]+, SIimm12
		裝載無符號位元組資料(前變址定址)	lbu rD, [rA, SIimm12]+
3	lh	裝載半字資料	lh rD, [rA, SIimm15]
		裝載半字資料(後變址定址)	lh rD, [rA]+, SIimm12
		裝載半字資料(前變址定址)	lh rD, [rA, SIimm12]+
4	lhu	裝載無符號半字資料	lhu rD, [rA, SIimm15]
		裝載無符號半字資料(後變址定址)	lhu rD, [rA]+, SIimm12
		裝載無符號半字資料(前變址定址)	lhu rD, [rA, SIimm12]+
5	lw	裝載字資料	lw rD, [rA, SIimm15]
		裝載字資料(後變址定址)	lw rD, [rA]+, SIimm12
		裝載字資料(前變址定址)	lw rD, [rA, SIimm12]+
6	sb	存儲位元組資料	sb rD, [rA, SIimm15]
		存儲位元組資料(後變址定址)	sb rD, [rA]+, SIimm12
		存儲位元組資料(前變址定址)	sb rD, [rA, SIimm12]+
7	sh	存儲半字資料	sh rD, [rA, SIimm15]
		存儲半字資料(後變址定址)	sh rD, [rA]+, SIimm12
		存儲半字資料(前變址定址)	sh rD, [rA, SIimm12]+
8	sw	存儲字資料	sw rD, [rA, SIimm15]
		存儲字資料(後變址定址)	sw rD, [rA]+, SIimm12
		存儲字資料(前變址定址)	sw rD, [rA, SIimm12]+
9	ldi	裝載立即數	ldi rD, SIimm16

序號	指令	功能說明	語法格式
10	ldis	裝載移位立即數	ldis rD, SImm16
11	lcb	裝載合併字資料開始	lcb [rA]+
12	lcw	裝載合併字資料	lcw rD, [rA]+
13	lce	裝載合併字資料結束	lce rD, [rA]+
14	scb	存儲合併字資料開始	scb rD, [rA]+
15	scw	存儲合併字資料	scw rD, [rA]+
16	sce	存儲合併字資料結束	sce [rA]+
17	add{.c}	加法運算	add{.c} rD, rA, rB
18	addc{.c}	帶進位加法運算	addc{.c} rD, rA, rB
19	addi{.c}	立即數加法運算	addi{.c} rD, SImm16
20	addis{.c}	移位立即數加法運算	addis{.c} rD, SImm16
21	addr{.c}	暫存器立即數加法運算	addr{.c} rD, rA, SImm14
22	sub{.c}	減法運算	sub{.c} rD, rA, rB
23	subc{.c}	帶借位減法運算	subc{.c} rD, rA, rB
24	subi{.c}	立即數減法運算	subi{.c} rD, SImm16
25	subis{.c}	移位立即數減法運算	subis{.c} rD, SImm16
26	subri{.c}	暫存器立即數減法運算	subri{.c} rD, rA, SImm14
27	neg{.c}	取負	neg{.c} rD, rB
28	cmp{TCS}.c	比較	cmp{TCS}.c rA, rB
29	cmpi.c	立即數比較	cmpi.c rA, SImm16
30	cmpz{TCS}.c	零比較	cmpz{TCS}.c rA
31	mul	乘法運算	mul rA, rB
32	mulu	無符號數乘法運算	mulu rA, rB
33	div	除法運算	div rA, rB
34	divu	無符號數除法運算	divu rA, rB
35	and{.c}	邏輯與	and{.c} rD, rA, rB
36	andi{.c}	立即數邏輯與	andi{.c} rD, Imm16
37	andis{.c}	移位立即數邏輯與	andis{.c} rD, Imm16

序號	指令	功能說明	語法格式
38	andri{.c}	暫存器立即數邏輯與	andri{.c} rD, rA, Imm14
39	or{.c}	邏輯或	or{.c} rD, rA, rB
40	ori{.c}	立即數邏輯或	ori{.c} rD, Imm16
41	oris{.c}	移位立即數邏輯或	oris{.c} rD, Imm16
42	orri{.c}	暫存器立即數邏輯或	orri{.c} rD, rA, Imm14
43	xor{.c}	邏輯異或	xor{.c} rD, rA, rB
44	not{.c}	邏輯異非	not{.c} rD, rA
45	t{cond}	測試並置 T 條件旗標	t{cond}
46	bittst.c	位測試	bittst.c rD, BN
47	bitset.c	位置位	bitset.c rD, rA, BN
48	bitclr.c	位清零	bitclr.c rD, rA, BN
49	bittgl.c	位翻轉	bittgl.c rD, rA, BN
50	mv{cond}	資料傳送	mv{cond} rD, rA
51	mtcr	傳送資料到控制暫存器	mtcr rD, Crn
52	mfcr	從控制暫存器傳送資料	mfcr rD, Crn
53	rol{.c}	迴圈左移	rol{.c} rD, rA, rB
54	rolc.c	帶進位迴圈左移	rolc.c rD, rA, rB
55	rol{i}.c	立即數迴圈左移	rol{i}.c rD, rA, Imm5
56	rolic.c	立即數帶進位迴圈左移	rolic.c rD, rA, Imm5
57	ror{.c}	迴圈右移	ror{.c} rD, rA, rB
58	rorc.c	帶進位迴圈右移	rorc.c rD, rA, rB
59	rori{.c}	立即數迴圈右移	rori{.c} rD, rA, Imm5
60	roric.c	立即數帶進位迴圈右移	roric.c rD, rA, Imm5
61	sll{.c}	邏輯左移	sll{.c} rD, rA, rB
62	slli{.c}	立即數邏輯左移	slli{.c} rD, rA, Imm5
63	sra{.c}	算術右移	sra{.c} rD, rA, rB
64	srai{.c}	立即數算術右移	srai{.c} rD, rA, Imm5
65	srl{.c}	邏輯右移	srl{.c} rD, rA, rB

序號	指令	功能說明	語法格式
66	srl{i}.c{}	立即數邏輯右移	srl{i}.c{ rD, rA, Imm5
67	extsb{i}.c{}	擴展有符號位元組資料	extsb{i}.c{ rD, rA
68	extzb{i}.c{}	擴展無符號位元組資料	extzb{i}.c{ rD, rA
69	extsh{i}.c{}	擴展有符號半字資料	extsh{i}.c{ rD, rA
70	extzh{i}.c{}	擴展無符號半字資料	extzh{i}.c{ rD, rA
71	j	無條件跳越	j{l} label
72	b	條件分支	b{cond}{l} label
73	br	條件暫存器分支	br{cond}{l} rA
74	ceinst	Custom Engine 用戶定義	ceinst func5, rA, rB, USD1, USD2
75	mtcex	傳送資料到 Custom Engine 暫存器	mtcel rD mtceh rD mtcehl rD, rA
76	mfcecx	從 Custom Engine 暫存器傳送資料到通用暫存器	mfcel rD mfceh rD mfcehl rD, rA
77	mtsrx	傳送資料到特殊暫存器	mtsrx rA, Srx
78	mfsrx	從特殊暫存器傳送資料到通用暫存器	mfsrx rD, Srx
79	cache	cache 控制指令	cache cache_op, [rA, SImm15]
80	trap	條件陷阱	trap{cond} Software_Parameter(Imm5)
81	syscall	系統調用	syscall Software_Parameter(Imm15)
82	sleep	睡眠	sleep
83	sdbbp	軟體 Debug 斷點	sdbbp code(Imm5)
84	pflush	清空管線	pflush
85	rte	從異常返回	rte
86	drte	從 debug 異常返回	drte
87	COPn	協同處理器用戶定義 (n 為協同處理器編號)	COPn cop_code20 或 COPn CrD, CrA, CrB, cop_code5

序號	指令	功能說明	語法格式
88	MTCn	傳送資料到協同處理器資料暫存器	MTCn rD, CrA
89	MTCCn	傳送資料到協同處理器控制暫存器	MTCCn rD, CrA
90	MFCn	從協同處理器資料暫存器傳送資料	MFCn rD, CrA
91	MFCCn	從協同處理器控制暫存器傳送資料	MFCCn rD, CrA
92	LDCn	裝載資料到協同處理器資料暫存器	LDCn CrA, [rD, SImm12]
93	STCn	從協同處理器資料暫存器存儲資料	STCn CrA, [rD, SImm12]

7.4.2 16 位指令速查表

表 7-6 16 位指令速查表

序號	指令	功能說明	格式
1	lbu{p}!	裝載無符號位元組資料	lbu! rDg0, [rAg0] 基底位址定址 lbus! rDg0, Imm5 基底位址變址定址
2	lh{p}!	裝載半字資料	lh! rDg0, [rAg0] 基底位址定址 lhps! rDg0, Imm6 基底位址變址定址
3	lw{p}!	裝載字資料	lw! rDg0, [rAg0] 基底位址定址 lwps! rDg0, Imm7 基底位址變址定址
4	sb{p}!	存儲位元組資料	sb! rDg0, [rAg0] 基底位址定址 sbps! rDg0, Imm5 基底位址變址定址
5	sh{p}!	存儲半字資料	sh! rDg0, [rAg0] 基底位址定址 shps! rDg0, Imm6 基底位址變址定址
6	sw{p}!	存儲字資料	sw! rDg0, [rAg0] 基底位址定址 swps! rDg0, Imm7 基底位址變址定址
7	ldiu!	裝載無符號立即數	ldiu! rDg0, Imm8
8	push!	存儲字資料（前減量變址定址）	push! rDgh, [rAg0]

序號	指令	功能說明	格式
9	pop!	裝載字資料（後增量變址定址）	pop! rDg0, [rAg0]
10	add!	16 位加法運算	add! rDg0, rAg0
11	addc!	16 位帶進位加法運算	addc! rDg0, rAg0
12	adde!	16 位立即數加法運算	adde! rDg0, Imm4
13	sub!	16 位減法運算	sub! rDg0, rAg0
14	subei!	16 位立即數減法運算	subei! rDg0, Imm4
15	neg!	16 位取負	neg! rDg0, rAg0
16	cmp!	16 位比較	cmp! rDg0, rAg0
17	and!	16 位邏輯與	and! rDg0, rAg0
18	or!	16 位邏輯或	or! rDg0, rAg0
19	xor!	16 位邏輯異或	xor! rDg0, rAg0
20	not!	16 位邏輯異非	not! rDg0, rAg0
21	t{cond}!	16 位測試並置 T 條件旗標	t{cond}!
22	bittst!	16 位位測試	bittst! rDg0, BN(Imm5)
23	bitset!	位置位	bitset! rDg0, BN
24	bitclr.c	位清零	bitclr! rDg0, BN
25	bittgl.c	位翻轉	bittgl! rDg0, BN
26	mv!	16 位暫存器資料傳送(低組->低組)	mv! rDg0, rAg0
27	mlfh!	16 位暫存器資料傳送(高組->低組)	mlfh! rDg0, rAg1
28	mhfl!	16 位暫存器資料傳送(低組->高組)	mhfl! rDg1, rAg0
29	sll!	16 位邏輯左移	sll! rDg0, rAg0
30	slli!	16 位立即數邏輯左移	slli! rDg0, Imm5
31	sra!	16 位算術右移	sra! rDg0, rAg0
32	srl!	16 位邏輯右移	srl! rDg0, rAg0
33	srlil!	16 位立即數邏輯右移	srlil! rDg0, Imm5

7.5 假指令速查表

表 7-7 假指令速查表

序號	假指令	功能說明	格式
1	.align	將位置計數器 PC 對齊到值為 alignment 的整數倍的位址	.align alignment[, [fill][, max]]
2	.ascii	插入字串	.ascii strings
3	.asciz	插入包含有字串結束旗標的字串	.asciz strings
5	.byte	插入位元組資料	.byte expressions
7	. data	切換到.data 區段	.data [subsection]
8	. equ	全局符號 symbol 的位址賦值為 expression	.equ symbol, expression
9	. extern	申明外部函數或者標號	.extern symbol
10	. global	申明標號為全局符號	.global symbol
11	. hword	插入半字資料	.hword expression
12	. include	引入其他檔案到當前檔案內	.include filename
13	. int	在未初始化的區段裏插入 4 個位元組資料	.int expressions
14	. org	將當前區段的位置計數器 PC 後移到新位址	.org new-loc [, fill]
15	. section	定義一個區段	.section NAME [, "FLAGS"[, @TYPE[, @ENTSIZE]]]
16	. set	禁止編譯器產生使用某暫存器帶來的警告	.set symbol, expression
17	. space	從當前位置插入 size 個位元組資料	.space size [, fill]
18	. text	切換到.text 區段	.text
19	. word	插入字資料	.word expression