SPCE061A 编程手册 v1.0

2002-3-15

凌阳授权翻译及转载

供大学计划推广专用

北京北阳电子技术有限公司.
北京市海淀区上地信息产业基地中黎科技园 1 号楼 6 层 C 段

TEL : 86-10-62981668 FAX : 86-10-62985972 E-mail:sunnorth@sunplus.com.tw WWW:www.sunnorth.com.cn

0 目录

1 修订记录

修订版本	日期	修订人	备注
V1.0	2002-3-15	王立平	初版

2 SPCE061A 简介

2.1 总述

SPCE061A 是继μ'nSP™系列产品 SPCE500A 等之后凌阳科技推出的又一个 16 位结构的微控制器。与 SPCE500A 不同的是，在存储器资源方面考虑到用户的较少资源的需求以及便于程序调试等功能，SPCE061A 里只内嵌 32K 字的闪存 FLASH ROM。较高的处理速度使μ'nSP™能够非常容易地、快速地处理复杂的数字信号。因此，与 SPCE500A 相同，以μ'nSP™为核心的 SPCE061A 微控制器也适用在数字语音识别应用领域。

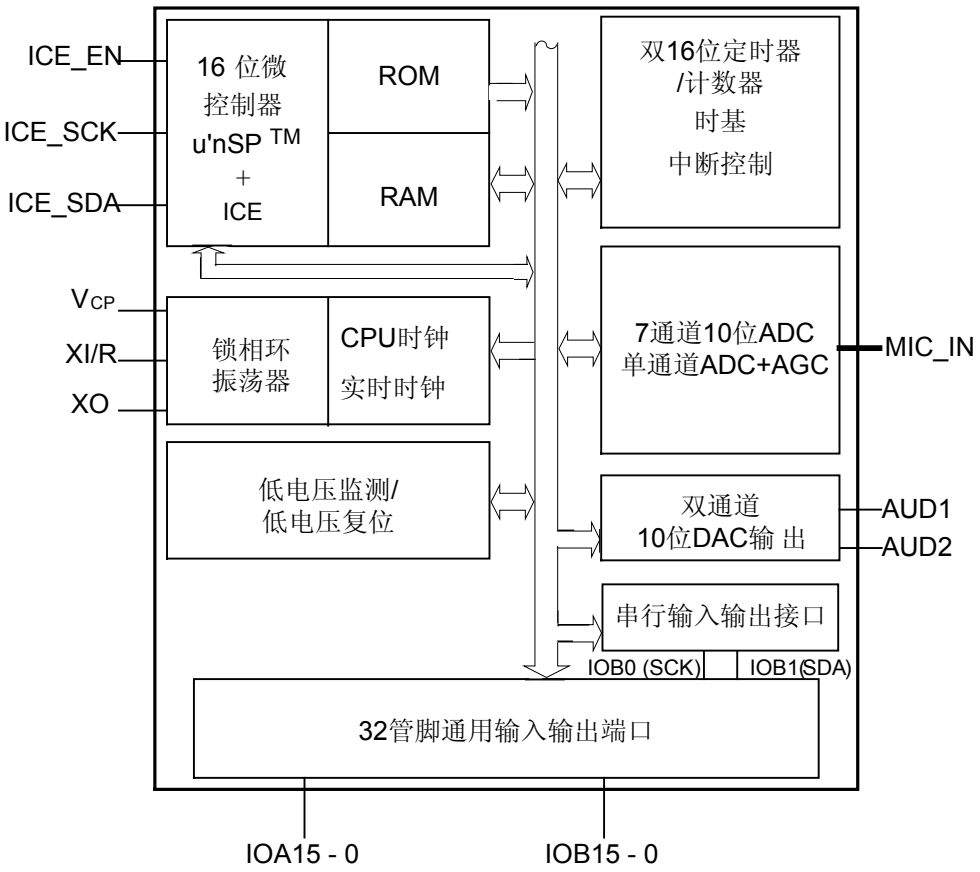
SPCE061A 在 2.6V~3.6V 工作电压范围内的工作速度范围为 0.32MHz~49.152MHz，较高的工作速度使其应用领域更加拓宽。2K 字 SRAM 和 32K 字闪存 ROM 仅占一页存储空间，32 位可编程的多功能 I/O 端口；两个 16 位定时器/计数器；32768Hz 实时时钟；低电压复位/监测功能；8 通道 10 位模-数转换输入功能并具有内置自动增益控制功能的麦克风输入方式；双通道 10 位 DAC 方式的音频输出功能……。SPCE061A 是数字声音和语音识别产品的一种最经济的应用。

2.2 性能

- 16 位μ'nSP™微处理器；
- 工作电压：V_{DD} 为 2.6~3.6V(cpu), V_{DDH} 为 V_{DD}~5.5V(I/O)；
- CPU 时钟：0.32MHz~49.152MHz ；
- 内置 2K 字 SRAM；
- 内置 32K 闪存 ROM；
- 可编程音频处理；
- 晶体振荡器；
- 系统处于备用状态下(时钟处于停止状态)，耗电小于 2μA@3.6V；
- 2 个 16 位可编程定时器/计数器(可自动预置初始计数值)；
- 2 个 10 位 DAC(数-模转换)输出通道；
- 32 位通用可编程输入/输出端口；
- 14 个中断源可来自定时器 A / B，时基，2 个外部时钟源输入，键唤醒；
- 具备触键唤醒的功能；
- 使用凌阳音频编码 SACM_S240 方式(2.4K 位/秒)，能容纳 210 秒的语音数据；
- 锁相环 PLL 振荡器提供系统时钟信号；
- 32768Hz 实时时钟；
- 7 通道 10 位电压模-数转换器(ADC)和单通道声音模-数转换器
- 声音模-数转换器输入通道内置麦克风放大器和自动增益控制(AGC)功能；
- 具备串行设备接口；

- 低电压复位(LVR)功能和低电压监测(LVD)功能；
- 内置在线仿真板(ICE，In- Circuit Emulator)接口。

2.3 结构概览



2.4 特性

	SPCE061A
工作电压	2.6V~3.6V
最大工作速率	49.152MHz
CPU	16 位μ'nSP™
SRAM 容量	2K 字
ROM 容量(字)	32K 闪存 ROM
并行 I/O 端口 A	IOA15~0
并行 I/O 端口 B	IOB15~0
音频输出方式	DAC×2
中断源	TimerA/B、 时基信号发生器 外部中断 触键唤醒
唤醒源	IOA7~0 其它中断源
定时器/计数器	双 16 位加计数定时器/计数器 双通道 PWM 输出
UART	具备
ADC	7 通道 10 位电压模-数转换器(ADC)和 单通道声音模-数转换器(ADC)
串行 SRAM 接口	具备(凌阳格式)
晶振	具备
低电压复位	具备
低电压监测	具备
内置 ICE 接口	具备
上电复位	具备
麦克风放大器 和自动增益控制	单通道
节电功能	具备
中断控制功能	具备
触键唤醒功能	具备

2.5 应用领域

- 语音识别类产品
- 智能语音交互式玩具
- 高级亦教亦乐类玩具
- 儿童电子故事书类产品
- 通用语音合成器类产品
- 需较长语音持续时间类产品

3 CPU

SPCE061A 配备了凌阳科技开发的最新的 16 位微处理器μ'nSP™。它内含有 8 个寄存器：4 个通用寄存器 R1~R4，1 个程序计数器 PC，1 个堆栈指针 SP，1 个基址指针 BP 和 1 个段寄存器 SR。通用寄存器 R3 和 R4 结合形成一个 32 位寄存器 MR，MR 可被用作乘法运算和内积运算的目标寄存器。此外，*SPCE061A* 有 3 个 FIQ 中断和 14 个 IRQ 中断，并且带有一个由指令 BREAK 控制的软中断。

μ'nSP™不仅可以进行加、减等基本算术运算和逻辑运算，还可以完成用于数字信号处理的乘法运算和内积运算。

4 存储器映象

4.1 SRAM

SPCE061A 拥有 2K 字的 SRAM(包括堆栈区)，其地址范围从$000000 到$0007FF。

4.2 闪存 Flash ROM

32K 字的内嵌式闪存用于存放程序和数据。全部 32K 字闪存均可在 ICE 工作方式下被编程写入或被擦除。对闪存设置保密设定后，其内容将不能再通过 ICE 被读写，也就可以使程序不被其他人读取。

5 输入输出端口

输入输出端口是系统与其它设备进行数据交换的接口。SPCE061A 具有两个可编程输入输出端口：A 口和 B 口。A 口既是具有可编程唤醒功能的普通 I/O 口，又可与 ADC 的多路 LINE_IN 输入共用(IOA[6~0]与 LINE_IN[1~7]共用，见P_ADC_MUX_Ctrl (读/写)($702BH)，此时 IOA 必须被设置为悬浮管脚)；B 口除了具有普通 I/O 口的功能外，在特定的管脚上还可以完成一些特殊的功能(参见**B 口的特殊功能**)。

5.1 I/O 端口结构

SPCE061A 提供了位控制结构的 I/O 端口，每一位都可以被单独定义用于输入或输出数据。通常，对某一位的设定包括以下 3 个基本项：数据向量 Data、属性向量 Attribution 和方向控制向量 Direction，它们分别代表 3 个控制端口。3 个端口内每个对应的位组合在一起，形成一个控制字，用来定义相应 I/O 口位的输入输出状态和方式。例如，假设需要 IOA0 是下拉输入管脚，则相应的 Data、Attribution 和 Direction 的值均被置为“0”。如果需要 IOA1 是带唤醒功能的悬浮式输入管脚，则 Data、Attribution 和 Direction 的值被置为“010”。

A 口和 B 口的 Data、Attribution 和 Direction 的设定值均在不同的寄存器里，用户在进行 I/O 口设置时要特别注意这一点。I/O 端口的组合控制设置如下表所示：

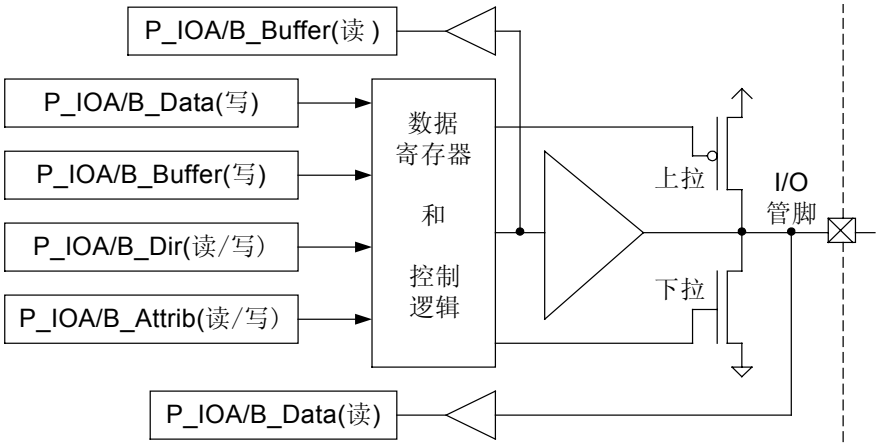
Direction	Attribution	Data	功能	是否带唤醒功能	功能描述
0	0	0	下拉*	是**	带下拉电阻的输入管脚
0	0	1	上拉	是**	带上拉电阻的输入管脚
0	1	0	悬浮	是**	悬浮式输入管脚
0	1	1	悬浮	否	悬浮式输入管脚
1	0	0	高电平输出(带数据反相器)	否	带数据反相器的高电平输出(当向数据位写入“0” 时输出“1”)
1	0	1	低电平输出(带数据反相器)	否	带数据反相器的低电平输出(当向数据位写入“1”时输出“0”)
1	1	0	低电平输出	否	带数据缓存器的低电平输出(无数据反相功能)
1	1	1	高电平输出	否	带数据缓存器的高电平输出(无数据反相功能)

注：

*：口位默认为带下拉电阻的输入管脚；

**：只有当 IOA [7~0]内位的控制字为 000，001 和 010 时， 相应位才具有唤醒的功能。

I/O 结构如下图所示：



5.2 P_IOA_Data(读/写)($7000H)

A 口的数据单元，用于向 A 口写入或从 A 口读出数据。当 A 口处于输入状态时，写入是将 A 口的数据向量写入 A 口的数据寄存器；读出则是从 A 口管脚上读其输入电平状态。当 A 口处于输出状态时，写入输出数据到 A 口的数据寄存器。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.3 P_IOA_Buffer (读/写) ($7001H)

A 口的数据向量单元，用于向数据向量寄存器写入或从该寄存器读出数据。当 A 口处于输入状态时，写入是将 A 口的数据向量写入 A 口的数据寄存器；读出则是从 A 口数据寄存器内读其数值。当 A 口处于输出状态时，写入输出数据到 A 口的数据寄存器。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.4 P_IOA_Dir(读/写)($7002H)

A 口的方向向量单元，用于向方向控制向量寄存器写入或从该寄存器内读出方向控制向量。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.5 P_IOA_Attrib(读/写)($7003H)

A 口的属性向量单元，用于向属性向量寄存器写入或从该寄存器内读出属性向量。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.6 P_IOA_Latch(读)($7004H)

读该单元以锁存 A 口上的输入数据，用于进入备用状态前的触键唤醒功能的启动(参见睡眠/唤醒部分)。

[例 5.1]:

设置 IOA[3~0] 为带下拉电阻的输入口，IOA[7~4]为带上拉电阻的输入口，IOA[11~8]为低电平输出口，IOA[15~12] 为高电平输出口。

```
R1 = 0xF0F0;
[P_IOA_Data] = R1;
R1 = 0xFF00;
[P_IOA_Attrib] = R1;
R1 = 0xFF00;
[P_IOA_Dir] = R1;
```

如上图所示，从 P_IOA_Data(写)和 P_IOA_Buffer(写)单元写入的数据被写向同一个数据寄存器，但从这两个单元(P_IOA_Buffer(读)、P_IOA_Data(读)) 读出的数据却来自不同的位置，见上图。对于某些 I/O 端口的应用，这种读写方式可以节省许多用于存放端口数据的 RAM 空间。

通过相应的设定，可以改变口位的状态、属性值。例如，将管脚的方向控制向量由“0”变为“1”，悬浮管脚(011)可以被更改为高电平输出管脚(111)。

此外，IOA[7~0]可以被用作触键唤醒源。如果需要触键唤醒功能，必须在进入备用状态之前通过读 P_IOA_Latch ($7004H)单元来锁存 A 口的键状态，并以此激活触键唤醒功能。‘睡眠’过程中当有键被按下时，A 口当前的输入状态将不同于被锁存时的状态，从而引起系统的唤醒。

5.7 P_IOB_Data(读/写)($7005H)

B 口的数据单元，用于向 B 口写入或从 B 口读出数据。当 B 口处于输入状态时，写入此单元是将 B 口的数据向量写入 B 口的数据寄存器；读出则是从 B 口管脚上读其输入电平状态。当 B 口处于输出状态时，写入输出数据到 B 口的数据寄存器。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.8 P_IOB_Buffer(读/写)($7006H)

B 口的数据向量单元，用于向数据寄存器写入或从该寄存器内读出数据。当 B 口处于输入状态时，写入是将 B 口的数据向量写入 B 口的数据寄存器；读出则是从 B 口数据寄存器里读其数值。当 B 口处于输出状态时，写入输出数据到 B 口的数据寄存器。

	b15	B14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.9 P_IOB_Dir(读/写)($7007H)

B 口的方向向量单元，用于向方向控制向量寄存器写入或从该寄存器内读出方向控制向量。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

5.10 P_IOB_Attrib(读/写)($7008H)

B 口的属性向量单元，用于向属性向量寄存器写入或从该寄存器内读出属性向量。

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
读/写																

[例 5.2]：

设置 IOB[3~0] 为带上拉电阻的输入口，IOB[7~4]为悬浮式输入口，IOB[11~8]为低电平输出口，IOB[15~12] 为高电平输出口。

```
R1 = 0xF00F;
[P_IOB_Data] = R1;
R1 = 0xFFF0;
[P_IOB_Attrib] = R1;
R1 = 0xFF00;
[P_IOB_Dir] = R1;
```

5.11 B 口的特殊功能

正如前面提到的，B 口除了具有常规的输入/输出端口功能外，还有一些特殊的功能，如下表所示：

口位	特殊功能	功能描述	备注
IOB0	SCK	串行接口 SIO 的时钟信号	参见串行设备接口 SIO 的功能设置内容
IOB1	SDA	串行接口 SIO 的数据传送信号	参见串行设备接口 SIO 的功能设置内容
IOB2	EXT1	外部中断源(下降沿触发)	IOB2 设为输入状态
	Feedback_Output1	与 IOB4 组成一个 RC 反馈电路，以获得振荡信号，作为外部中断源 EXT1	设置 IOB2 为反相输出方式，见如下所示的框图及P_FeedBack(写)($7009H)单元的描述
IOB3	EXT2	外部中断源(下降沿触发)	IOB3 设为输入状态
	Feedback_Output2	与 IOB5 组成一个 RC 反馈电路，以获得一个振荡信号，作为外部中断源 EXT2	设置 IOB3 为反相输出方式，见如下所示的框图及P_FeedBack(写)($7009H)单元的描述
IOB4	Feedback_Input1		见如下所示的框图
IOB5	Feedback_Input2		见如下所示的框图
IOB6	---		
IOB7	Rx	通用异步串行数据接收端口	参见通用异步串行接口部分
*IOB8	APWMO	TimerA 脉宽调制输出	参见定时器/计数器部分，P_Feedback(写) ($7009H) 单元部分
IOB9	BPWMO	TimerB 脉宽调制输出	参见定时器/计数器部分
*IOB10	Tx	通用异步串行数据发送端口	参见通用异步串行端口 UART部分

注：

1. 口位默认为带下拉电阻的输入管脚
2. PWM：脉宽调制(Pulse Width Modulation)
3. *：IOB8 和 IOB10 的特殊功能请参见P_FeedBack(写)($7009H)单元

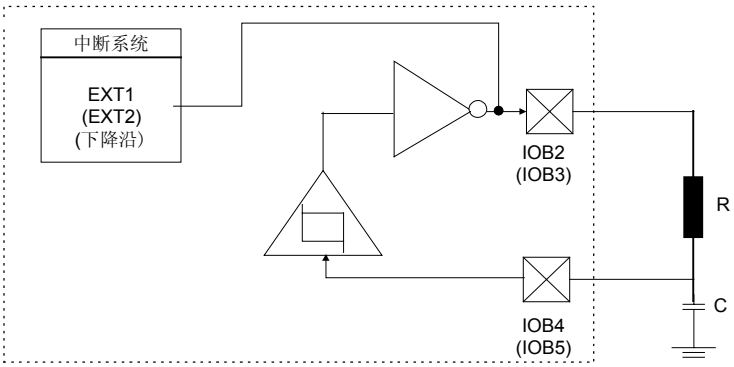
IOB8 的应用方式通过 TimerA 脉宽调制输出 APWMO 的允通信号 TAON 进行控制。当 TAON 为‘0’时，IOB8 为普通并行 I/O 口；而当 TAON 为 ‘1’ 时，IOB8 则为 APWMO 端口。

5.12 P_FeedBack(写)($7009H)

B 口的应用方式控制向量单元，用于控制 B 口的 IOB2 (IOB3)和 IOB4 (IOB5)用作普通 I/O 口，还是在它们之间形成反馈电路以产生振荡信号作为外部中断源输入 EXT1 或 EXT2。

b15 – b4	b3	b2	b1	b0
---	FBKEN3 1：设定 IOB3 和 IOB5 之间形成反馈功能 0：IOB3、IOB5 作为普通的 I/O 口(默认)	FBKEN2 1：设定 IOB2 和 IOB4 之间形成反馈功能 0：IOB2、IOB4 作为普通的 I/O 口(默认)	---	---

下图为 IOB2，IOB3，IOB4 及 IOB5 的反馈结构示意图。通过在 IOB2 (IOB3)和 IOB4 (IOB5)之间增加一个 RC 电路形成反馈回路，即可在 EXT1(EXT2)端得到振荡源频率信号。为使反馈回路正常工作，必须将 IOB2 (IOB3) 设置成反相输出口，且将 IOB4 (IOB5)设置成悬浮式输入口。



当P_FeedBack(写)($7009H)单元的FBKEN2(FBKEN3)位被置为"1"时IOB2、IOB4或IOB3、IOB5之间的反馈结构

[例 5.3]：

以下程序说明如何编程使得通过在 IOB2 和 IOB4 增加一个外部 RC 电路形成反馈回路，以获得振荡源频率。

```
//将 IOB4 设置成悬浮式输入口，IOB2 设置成反相输出口
R1=0x0004；
[P_IOB_Dir]=R1；
R1=0x0010；
[P_IOB_Attrib]=R1；
// 写入 P_FeedBack 口
R1=0x0004；
[P_FeedBack]=R1；
```

5.13 IOB8 的控制向量 TAON、TXPinEn 的设置

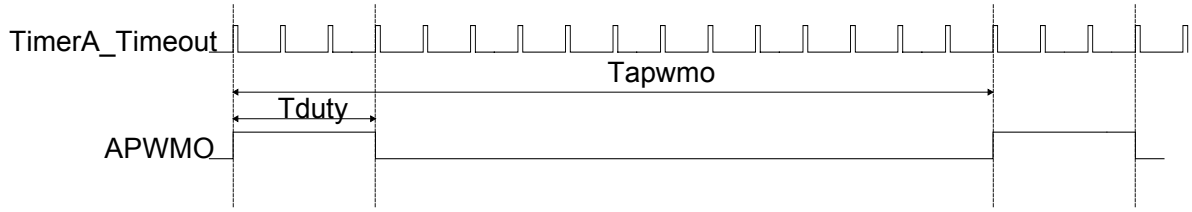
IOB8 的应用由控制向量 TAON、TXPinEN 的组合来进行控制。

TAON	TXPinEn	IOB8
0	0	普通 I/O 端口
0	1	普通 I/O 端口
1	0	APWMO 端口
1	1	UART TX 端口

注：

1. TAON：TimerA 脉宽调制输出 APWMO 的允通信号(详细内容请参见定时器/计数器部分)
2. TXPinEn：通用异步串行接口发送端的允许信号(详细内容请参见通用异步串行接口 UART部分)
3. APWMO：TimerA 脉宽调制的输出信号(详细内容请参见定时器/计数器部分)
4. Tx：通用异步串行数据发送端口 (详细内容请参见通用异步串行接口部分)

[例 5.4]：编写一段程序以实现如下所示的时序图内各信号波形




```
//将 IOB8 设置成反相输出端口
R1=0x0100；
[P_IOB_Dir]=R1；
R1=0x0000；
[P_IOB_Attrib] = R1；
[P_IOB_Data]=R1；
//设置 TimerA 的 APWMO 信号的周期 Tapwmo=(12.288MHz / 512) / 16 = 1.5KHz，设置 APWMO 信号的占空比
//Tduty= (3/16)*Tapwmo。详细内容请参见定时器/计数器部分。
R1=0xFDFF；
[P_TimerA_Data]=R1；
R1=0x00F0；
[P_TimerA_Ctrl]=R1；
```

6 定时器/计数器

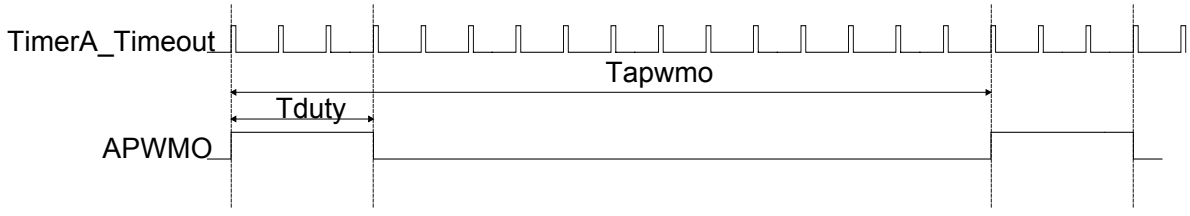
SPCE061A 提供了两个 16 位的定时器/计数器：TimerA 和 TimerB。TimerA 为通用计数器；TimerB 为多功能计数器。TimerA 的时钟源由时钟源 A 和时钟源 B 进行“与”操作而形成；TimerB 的时钟源仅为时钟源 A。

定时器发生溢出后会产生一个溢出信号(TAOUT/TBOUT)。一方面，它会作为定时器中断信号传输给 CPU 中断系统；另一方面，它又会作为 4 位计数器计数的时钟源信号，输出一个具有 4 位可调的脉宽调制占空比输出信号 APWMO 或 BPWMO(分别从 IOB8 和 IOB9 输出)，用来控制马达或其它一些设备的速度。此外，定时器溢出信号还可以用于触发 ADC 输入的自动转换过程和 DAC 输出的数据锁存。

向定时器的 P_TimerA_Data(读/写)($700AH)单元或 P_TimerB_Data(读/写)($700CH)单元写入一个计数值 N 后，选择一个合适的时钟源，定时器/计数器将在所选的时钟频率下开始以递增方式计数 N，N+1，N+2，…0xFFFE，0xFFFF。当计数达到 0xFFFF 后，定时器/计数器溢出，产生中断请求信号，被 CPU 响应后送入中断控制器进行处理。同时，N 值将被重新载入定时器/计数器并重新开始计数。

在 TimerA 内，时钟源 A 是一个高频时钟源，时钟源 B 是一个低频时钟源。时钟源 A 和时钟源 B 的组合，为 TimerA 提供出多种计数速度。若以 ClkA 作为门控信号，‘1’表示允许时钟源 B 信号通过，而‘0’则表示表示禁止时钟源 B 信号通过而停止 TimerA 的计数。例如，如果时钟源 A 为“1”，TimerA 时钟频率将取决于时钟源 B；如果时钟源 A 为“0”，将停止 TimerA 的计数。EXT1 和 EXT2 为外部时钟源。

下图是一个占空比为 3/16 的脉宽调制输出信号的时序。通过写入 P_TimerA_Ctrl($700BH)单元的第 6~9 位，可选择设置 APWMO 输出波形的脉宽占空比；同理，写入 P_TimerB_Ctrl($700DH)单元的第 6~9 位，便可选择设置 BPWMO 输出波形的脉宽占空比。此类 PWM 信号可以用于控制马达及其它设备的速度。



时钟源 A 是高频时钟源，来自带锁相环的晶体振荡器输出 Fosc；时钟源 B 的频率来自 32768Hz 实时时钟系统，也就是说，时钟源 B 可以作为精确的计时器。例如，2Hz 定时器可以作为实时时钟的时钟源。

6.1 P_TimerA_Data(读/写)($700AH)

TimerA 的数据单元，用于向 16 位预置寄存器写入数据(预置计数初值)或从其中读取数据。

6.2 P_TimerA_Ctrl(写)($700BH)

TimerA 的控制单元。用户可以通过设置该单元的第 0~5 位来选择 TimerA 的时钟源(时钟源 A、B)。设置该单元的第 6~9 位，TimerA 将输出不同频率的脉宽调制信号，即对脉宽占空比输出 APWMO 进行控制。

b15 – b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
---	Output_pulse_ctrl				时钟源 B 选择位			时钟源 A 选择位		

b9	b8	b7	b6	脉宽占空比(APWMO)	TAON[1]
0	0	0	0	关断	0
0	0	0	1	1/16	1
0	0	1	0	2/16	1
0	0	1	1	3/16	1
0	1	0	0	4/16	1
0	1	0	1	5/16	1
0	1	1	0	6/16	1
0	1	1	1	7/16	1
1	0	0	0	8/16	1
1	0	0	1	9/16	1
1	0	1	0	10/16	1
1	0	1	1	11/16	1
1	1	0	0	12/16	1
1	1	0	1	13/16	1
1	1	1	0	14/16	1
1	1	1	1	TAOUT[2]触发信号	1

注：

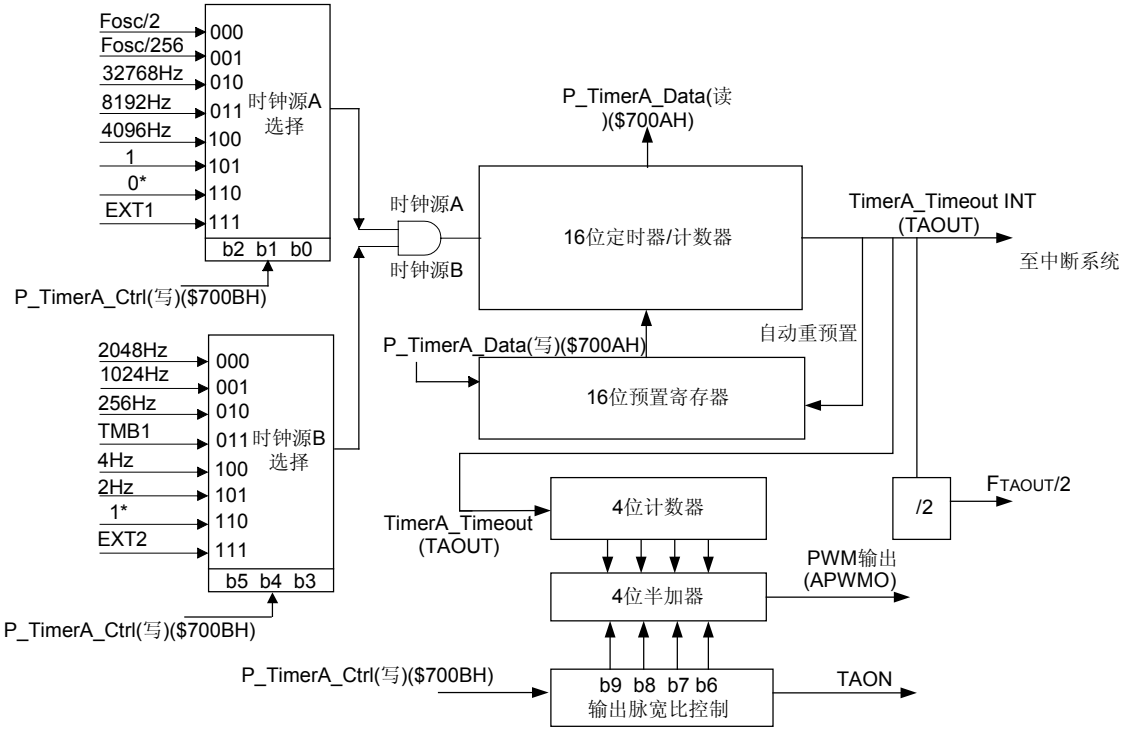
[1]：TAON 是 TimerA(APWMO)的脉宽调制信号输出允许位，默认值为“0”，当 TimerA 的第 6~9 位不全为零时 TAON=1；

[2]：TAOUT 是 TimerA 的溢出信号，当 TimerA 的计数从 N 达到 0xFFFF 后(用户通过设置 P_TimerA_Data (写)($700AH) 单元指定 N 值)，发生计数溢出。产生的溢出信号可以作为 Timer 的中断信号被送至中断控制系统；同时 N 值将被重新载入预置寄存器，使 Timer 重新开始计数。TAOUT 触发信号(TAOUT/2)的占空比为 50%，频率为 $F_{TAOUT}/2$，其它输入信号的频率为 $F_{TAOUT}/16$。请参考 TimerA 结构图。

b2	b1	b0	时钟源 A 的频率
0	0	0	Fosc/2
0	0	1	Fosc/256
0	1	0	32768Hz
0	1	1	8192Hz
1	0	0	4096Hz
1	0	1	1
1	1	0	0*
1	1	1	EXT1

b5	b4	b3	时钟源 B 的频率
0	0	0	2048Hz
0	0	1	1024Hz
0	1	0	256Hz
0	1	1	TMB1
1	0	0	4Hz
1	0	1	2Hz
1	1	0	1*
1	1	1	EXT2

*： 默认值。若以 ClkA 作为门控信号，‘1’表示允许时钟源 B 信号通过，而‘0’则表示禁止时钟源 B 信号通过而停止 TimerA 的计数。如果时钟源 A 为“1”，TimerA 时钟频率将取决于时钟源 B；如果时钟源 A 为“0”，将停止 TimerA 的计数。



TimerA结构

6.3 P_TimerB_Data(读/写)($700CH)

TimerB 的数据单元，用于向 16 位预置寄存器写入数据(预置计数初值)或从其中读取数据。

6.4 P_TimerB_Ctrl(写)($700DH)

TimerB 的控制单元。用户可以通过设置该单元的第 0~2 位来选择 TimerB 的时钟源。设置第 6~9 位，TimerB 将输出不同频率的脉宽调制信号，即对脉宽占空比输出 BPWMO 进行控制。

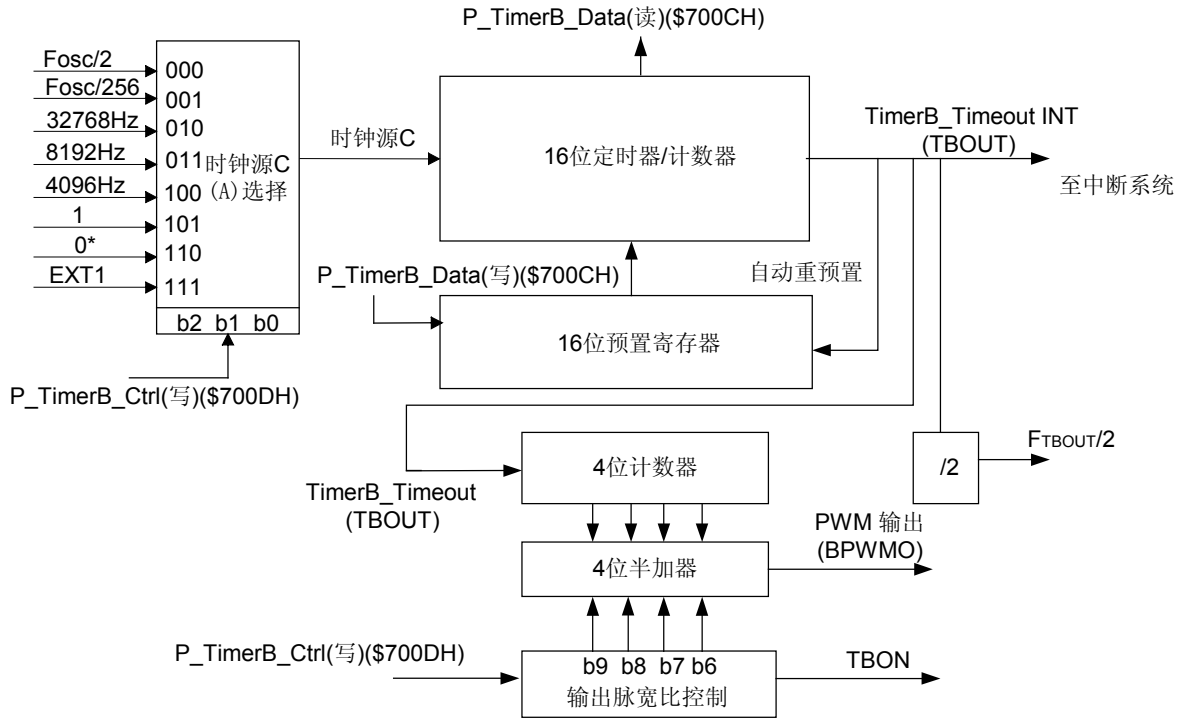
b15 - b10	B9	b8	b7	b6	b5	b4	B3	b2	b1	b0
---	Output_pulse_ctrl				---			时钟源 A 选择位		

b9	b8	b7	b6	脉宽占空比(BPWMO)	TBON[1]
0	0	0	0	关断	0
0	0	0	1	1/16	1
0	0	1	0	2/16	1
0	0	1	1	3/16	1
0	1	0	0	4/16	1
0	1	0	1	5/16	1
0	1	1	0	6/16	1
0	1	1	1	7/16	1
1	0	0	0	8/16	1
1	0	0	1	9/16	1
1	0	1	0	10/16	1
1	0	1	1	11/16	1
1	1	0	0	12/16	1
1	1	0	1	13/16	1
1	1	1	0	14/16	1
1	1	1	1	TBOUT[2] 触发信号	1

注：

[1]：TBON 是 TimerB(BPWMO)的脉宽调制信号输出允许位，默认值为“0”；

[2]：T BOUT 是 TimerB 的溢出信号，当 TimerB 的计数从 N 达到 0xFFFF 后(用户通过设置 P_TimerB_Data (写)($700CH) 单元指定 N 值)，发生计数溢出。产生的溢出信号可以作为 Timer 的中断信号被送至中断控制系统；同时 N 值将被重新载入预置寄存器，使 Timer 重新开始计数。TBOUT 触发信号(TBOUT/2)的占空比为 50%，频率为 $F_{TBOUT}/2$，其它输入信号的频率为 FTBOUT/16。



TimerB结构

6.5 时间基准信号

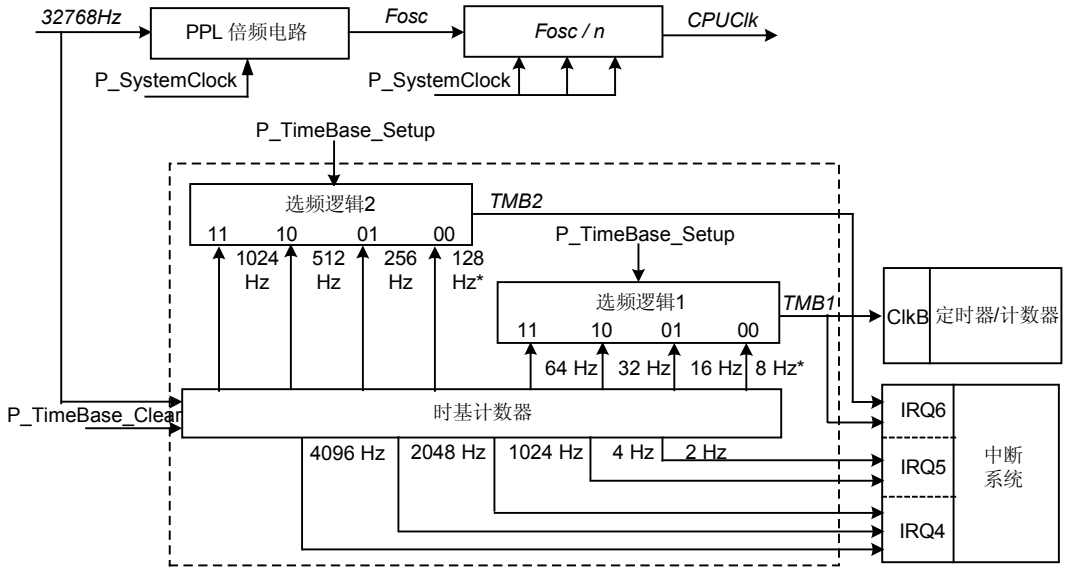
时间基准信号，简称时基信号，来自于 32768Hz 实时时钟，通过频率选择组合而成。时基信号发生器的 2 个选频逻辑 TMB1 和 TMB2 为 TimerA 的时钟源 B 提供各种频率选择信号并为中断系统提供中断源(IRQ6)信号。此外，时基信号发生器还可以直接生成 2Hz、4Hz、1024Hz、2048Hz 以及 4096Hz 的时基信号，为中断系统提供各种实时中断源(IRQ4 和 IRQ5)信号。

6.6 P_Timebase_Setup(写)($700EH)

时基信号发生器通过对 P_Timebase_Setup(写)($700EH)单元的编程写入来进行选频操作。

b15- b4	b3	b2	b1	b0
---	TMB2 选频逻辑		TMB1 选频逻辑	

b3	b2	TMB2	b1	b0	TMB1
0	0	128Hz*	0	0	8Hz**
0	1	256Hz	0	1	16Hz
1	0	512Hz	1	0	32Hz
1	1	1024Hz	1	1	64Hz
*：默认的 TMB2 输出频率为 128Hz			**：默认的 TMB1 输出频率为 8Hz		



时基信号发生器的结构

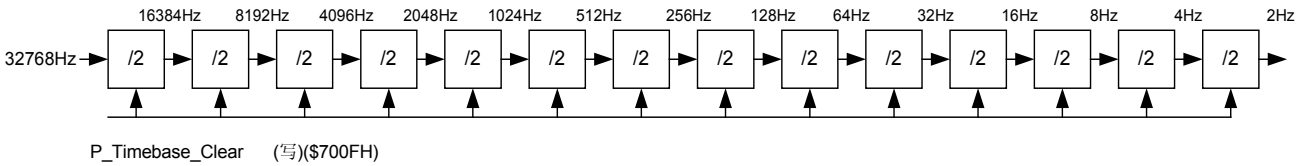
6.7 实时时钟 RTC(Real Time Clock)

32768Hz 实时时钟通常用于钟表、实时时钟延时以及其它与时间相关类产品。SPCE061A 通过对 32768Hz 实时时钟源分频而提供了多种实时时钟中断源。例如，用作唤醒源的中断源 IRQ5_2Hz，表示系统每隔 0.5 秒被唤醒一次，由此可作为精确的计时基准。”

除此之外，SPCE061A 还支持 RTC 振荡器强振模式/自动模式的转换(参考系统时钟部分)。

6.8 P_Timebase_Clear(写)($700FH)

P_Timebase_Clear (写)($700FH)单元是控制端口，设置该单元可以完成时基计数器复位和时间校准。向该单元写入任意数值后，时基计数器将被置为“0”，以此可对时基信号发生器进行精确的时间校准。



7 中断

SPCE061A 具有两种中断方式：快速中断请求 FIQ(Fast Interrupt Request)中断和中断请求 IRQ(Interrupt Request)中断。中断控制器可处理 3 种 FIQ 中断和 11 种 IRQ 中断，以及一个由指令 BREAK 控制的软中断。

相比之下，FIQ 中断的优先级较高而 IRQ 中断的优先级较低。也就是说，FIQ 中断可以中断 IRQ 中断服务子程序的执行，而 CPU 执行相应的 FIQ 中断服务子程序的过程不能被任何中断源的中断请求中断。

中断源	中断优先级	中断向量	保留字
Fosc/1024 溢出信号	FIQ/IRQ0	FFF8H/FFF6H	_FIQ/_IRQ0
TimerA 溢出信号	FIQ /IRQ1	FFF9H/FFF6H	_FIQ/_IRQ1
TimerB 溢出信号	FIQ /IRQ2	FFFAH/FFF6H	_FIQ/_IRQ2
外部时钟源输入信号 EXT2	IRQ3	FFFBH	_IRQ3
外部时钟源输入信号 EXT1			
触键唤醒信号			
4096Hz 时基信号	IRQ4	FFFCH	_IRQ4
2048Hz 时基信号			
1024Hz 时基信号			
4Hz 时基信号	IRQ5	FFFDH	_IRQ5
2Hz 时基信号			
频选信号 TMB1	IRQ6	FFFEH	_IRQ6
频选信号 TMB2			
UART 传输中断	IRQ7	FFFFH	_IRQ7
BREAK	软中断		

所谓 IRQ 的中断优先级只有在两种以上 IRQ 中断同时发生时才起作用。当多个 IRQ 中断同时发生时，中断请求的响应顺序为：IRQ0 ，IRQ1，IRQ2，…. ，IRQ6，IRQ7。当多个 FIQ 中断同时发生时，中断事件的响应顺序为：FIQ_TMA，FIQ_TMB。

通过编程写入 P_INT_Ctrl(写)($7010H)单元，可以设置系统的中断源。向该单元某位写入“1”允许相应中断源的中断请求，反之，写入“0”将禁止相应中断源之中断请求，但此并不能清除 P_INT_Ctrl(读)($7010)单元内相应的中断标志位。

当有中断事件发生时，程序会读出 P_INT_Ctrl(读)($7010H)单元的数据以判断哪一个中断源被激活，然后，CPU 根据中断优先级执行相应的中断服务子程序(当同时发生多个中断事件时)。当前的中断服务子程序执行完毕，系统将清除相应的中断向量并开始按优先级处理其它的 IRQ 中断，直到完成所

有的中断的处理。此时，CPU 跳出 IRQ 中断子程序，从断点处继续执行其后的指令。

在某个中断服务子程序的结尾，向 P_INT_Clear(写)($7011H) 单元写入数值可清除该中断(注：因为不是写入任何数值都可清除的)。通过这种方式可以直接清除被处理过的中断标志且不会影响到其它已被激活的中断源。例如，P_INT_Ctrl (写)($7010H)单元的第 5 位被置位，即允许 IRQ4_2KHz 中断源提出中断请求。当 IRQ4_2KHz 中断发生后，该中断的状态标志位被置位，CPU 进入相应的中断服务子程序。中断服务子程序执行完毕，向 P_INT_Clear (W)($7011H)写入 0000000000100000 以清除 IRQ4_2KHz 中断的状态标志，但不会影响其它中断的状态标志。

μ'nSP™提供了如下保留字作为中断服务程序的入口点：_FIQ、_IRQ0、_IRQ1、_IRQ2、_IRQ3、_IRQ4、_IRQ5、_IRQ6、_IRQ7(_IRQ7 用于 UART IRQ)、_BREAK (中断向量地址：0xFFF5，用于软件中断)。

注：UART IRQ 中断的控制及其描述可参考***UART***部分的内容。

7.1 P_INT_Ctrl(读)($7010H)

该单元用于中断事件标志的读出或中断/唤醒功能的设置及取消。

b7	b6	b5	b4	b3	b2	b1	b0
IRQ3_KEY	IRQ4_4KHz	IRQ4_2KHz	IRQ4_1KHz	IRQ5_4Hz	IRQ5_2Hz	IRQ6_TMB1	IRQ6_TMB2

b15	b14	b13	b12	b11	b10	b9	b8
FIQ_Fosc/1024	IRQ0_Fosc/1024	FIQ_TMA	IRQ1_TMA	FIQ_TMB	IRQ2_TMB	IRQ3_EXT2	IRQ3_EXT1

7.2 P_INT_Clear(写)($7011H)

该单元用于清除中断的状态标志。将 P_INT_Clear 单元的某位置为‘1’，可以清除相应位的中断事件；置为‘0’时，则不会影响该位中断的标志。

b7	b6	b5	b4	b3	b2	b1	b0
IRQ3_KEY	IRQ4_4KHz	IRQ4_2KHz	IRQ4_1KHz	IRQ5_4Hz	IRQ5_2Hz	IRQ6_TMB1	IRQ6_TMB2

b15	b14	b13	b12	b11	b10	b9	b8
FIQ_Fosc/1024	IRQ0_Fosc/1024	FIQ_TMA	IRQ1_TMA	FIQ_TMB	IRQ2_TMB	IRQ3_EXT2	IRQ3_EXT1

7.3 P_INT_Ctrl_New(读/写)($702DH)

该单元用于激活和屏蔽中断。

b7	b6	b5	b4	b3	b2	b1	b0
IRQ3	IRQ4	IRQ4	IRQ4	IRQ5	IRQ5	IRQ6	IRQ6

b15	b14	b13	b12	b11	b10	b9	b8
FIQ	IRQ0	FIQ	IRQ1	FIQ	IRQ2	IRQ3	IRQ3

7.4 中断控制配置端口

P_INT_Ctrl_New(写)	P_INT_Ctrl (读)	P_INT_Clear(写)	功能
1	—	—	允许中断/唤醒功能
0	—	—	屏蔽中断/唤醒功能，但不清除 P_INT_Ctrl (读)单元相应的中断标志位
—	1	—	有中断事件发生
—	0	—	没有中断事件发生
—	—	1	清除中断事件
—	—		不改变中断源的状态

[例 7.1]：

允许 TimerA 中断 FIQ_TMA，并从 IOB 口输出一个触发信号。

```
//将 IOB 设置成输出口
        R1=0x0FFFF；
        [P_IOB_Attrib]=R1；
        [P_IOB_Dir]=R1；
        R1=0x0000；
        [P_IOB_Data]=R1；
//设置 TimerA=8KHz
       R1=0xFA00；
       [P_TimerA_Data]=R1；
       R1=0x0030；
       [P_TimerA_Ctrl]=R1；
// 设置 FIQ_TMA 中断源
       R1=0x2000；
       [P_INT_Ctrl_New]=R1；
       INT FIQ；
// 设置 TimerA 中断后，SPCE061A 将根据用户设定的 TimerA 的溢出频率进入相应的 FIQ 中断服务子程序入口
    L_loop_fiq：
    JMP L_loop_fiq；
……
// FIQ 中断处理子程序
.TEXT
.INCLUDE HARDWARE.INC.
```

```
.public _FIQ;
_FIQ：
        PUSH R1，R4 to [sp]；
        R1 = C_FIQ_TMA；
        Test R1，[P_INT_Ctrl]；                    //是否是定时器 A 中断
        JNE  L_FIQ_TimerA；
        R1 = C_FIQ_TMB；
        Test R1，[P_INT_Ctrl]；                    //是否是定时器 B 中断
        JNE L_FIQ_TimerB；
L_FIQ_PWM:                                         //PWM 中断处理
        R1 = C_FIQ_PWM；
        [P_INT_Clear] = R1；
        POP R1，R4 from [sp]；
        RETI；
L_FIQ_TimerA:                                      //定时器 A 中断处理
        [P_INT_Clear] = r1；
        R1 = [P_IOB_Buffer]
        R1 = R1 XOR 0xFFFF                         //当发生 FIQ_TMA 中断时触发 IOB 信号
        [P_IOB_Buffer] = R1
        POP R1，R4 from [sp]；
        RETI；
L_FIQ_TimerB:                                      //定时器 B 中断处理
        [P_INT_Clear] = r1；
        POP R1，R4 from [sp]；
         RETI；
```

8 睡眠与唤醒

8.1 睡眠

IC 在上电复位开始工作，直到接收到睡眠信号后，才关闭系统时钟(PLL 振荡器)，进入睡眠状态。用户可以通过对 P_SystemClock(读)($7013H)单元写入 *CPUClk STOP* 控制字(CPU 睡眠信号)使系统从运行状态转入备用状态。系统进入睡眠状态后，程序计数器(PC)会停在程序的下一条指令计数上，当有任一唤醒事件发生后开始由此继续执行程序。

8.2 唤醒

系统接收到唤醒信号后接通系统时钟(PLL 振荡器)，同时 CPU 会响应唤醒事件的处理并进行初始化。IRQ3_KEY 为触键唤醒源(IOA7~0)，其它中断源(FIQ、IRQ1~IRQ6 及 UART IRQ)都可以作为唤醒源。唤醒操作完成后，程序将会从进入睡眠后指令计数的断点处开始被继续执行。关于触键唤醒源，请参考**端口 A 的结构** 。

[例 8.1]：

说明如何编程实现系统从工作状态进入睡眠状态后，由触键引起唤醒。在进入睡眠状态之前，首先要将 IOA[7~0]设置为输入状态且允许 RQ3_KEY 中断来实现触键唤醒。

```
//IOA[7~0]设置(关于其它 I/O 端口的结构，请参考I/O 端口结构)。
R1=0x0000；
[P_IOA_Attrib]=R1；
R1=0x0000；
[P_IOA_Dir]=R1；
[P_IOA_Data]=R1；

//中断设置(允许 IRQ3_KEY 触键中断，关于其它的中断设置，请参考中断部分)
INT OFF；
R1=0x0080；
[P_Int_Ctrl]=R1；
INT IRQ；

//读 P_IOA_Latch 单元，以锁存 IOA[0～7]的数据，用于触键唤醒
R1=[P_IOA_Latch]；

//将 P_SystemClock(写)7013H($)单元的第 0~2 位置为“111”，使系统进入睡眠状态，参见系统时钟部分
R1=0x0007；
```

```
    [P_SystemClock]=R1；
      …(端口 A 的触键唤醒源被触发后，调用 IRQ3  中断服务子程序)

IRQ3 子程序：
 _IRQ3:
      R1 = 0x0100；
      TEST R1,[P_INT_Ctrl]；
      JNZ L_IRQ3_Ext1；
      R1 = 0x0200；
      TEST R1,[P_INT_Ctrl]；
      JNZ L_IRQ3_Ext2；
L_IRQ3_KeyChange_WakeUp:
      R1 = 0x0080；                          //清除 IRQ3 触键中断请求
      [P_INT_Clear]= R1；
        :
      (处理系统被唤醒后的任务)
        :
      RETI；

L_IRQ3_Ext2:
      [P_INT_Clear] = R1；                   //清除 IRQ3_EXT2 中断请求
      RETI；

L_IRQ3_Ext1:
      [P_INT_Clear] = R1；                   //清除 IRQ3_EXT1 中断请求
      RETI；
```

9 系统时钟

系统时钟的信号源为 PLL 振荡器。系统时钟频率(Fosc)和 CPU 时钟频率(CPUCLK)可通过对 P_SystemClock(写)($7013H)单元编程来控制。默认的 Fosc、CPUCLK 分别为 24.576MHz 和 Fosc/8。用户可以通过对 P_SystemClock 单元编程完成对系统时钟和 CPU 时钟频率的定义。

此外，32768Hz RTC 振荡器有两种工作方式：强振模式和自动弱振模式。处于强振模式时，RTC 振荡器始终运行在高耗能的状态下。处于自动弱振模式时，系统在上电复位后的前 7.5s 内处于强振模式，然后自动切换到弱振模式以降低功耗。CPU 被唤醒后默认的时钟频率为 Fosc/8，用户可以根据需要调整该值。这样可以避免在系统被唤醒后造成 ROM 读取错误。

在 SPCE061A 内，P_SystemClock(写)($7013H)单元控制着系统时钟和 CPU 时钟。通过设置该单元的第 5 位可以改变系统时钟的频率(Fosc=20/24MHz)；将第 0~2 位置为“111”可以使 CPU 时钟停止工作，系统切换至低功耗的备用状态。此外，在备用状态下，通过设置该单元的第 4 位可以接通或关闭 32KHz 实时时钟。

b15-b8	b7~b5	b4[1]	b3	b2	b1	b0
---	PLL 频率选择	32KHz 睡眠状态	32KHz 方式选择	CPU 时钟选择		
		1：在备用状态下，32768Hz 时钟仍处于工作状态(默认) 0：在备用状态下，32768H 时钟被关闭	1：32768Hz 时钟处强振模式 0：32768Hz 时钟处自动弱振模式(默认)			

b2	b1	b0	CPU 时钟频率
0	0	0	Fosc
0	0	1	Fosc/2
0	1	0	Fosc/4
0	1	1	Fosc/8[2]
1	0	0	Fosc/16
1	0	1	Fosc/32
1	1	0	Fosc/64
1	1	1	停止(睡眠状态)

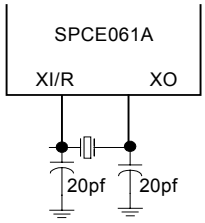
b7	b6	b5	CPU 时钟频率
0	0	0	24.576MHz
0	0	1	20.48MHz
0	1	0	32.768MHz
0	1	1	40.96MHz
1	-	-	49.152MHz

注：

[1]：只有当 b 0~b2 同时被置为“1”时(即睡眠状态)b4 设置才有效；

[2]：上电复位或系统从备用状态(睡眠状态)被唤醒后，默认的 CPU 时钟频率为 Fosc/8。

下图为 SPCE061A 与晶体振荡器振荡器的连接电路原理图。

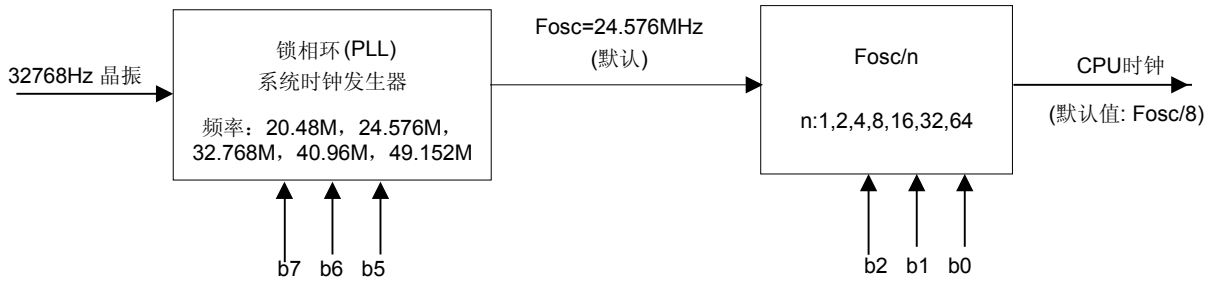


SPCE061A 与振荡器的连接

10 锁相环 PLL (Phase Lock Loop)振荡器

PLL 的作用是为系统提供一个实时时钟的基频(32768Hz)，然后将基频进行倍频，调整至 49.152MHz、40.96MHz、32.768MHz、24.576MHz 或 20.480MHz。系统默认的 PLL 自激振荡频率为 24.576MHz。

PLL 的结构如下图所示：



系统时钟选频P_SystemClock(写)($7013H)的第7、6、5位　　系统时钟选频P_SystemClock(写)($7013H)的第2、1、0位

系统时钟

11 模-数转换器 ADC

SPCE061A 有 8 个 10 位模-数转换器，其中 7 个通道用于将模拟量信号 (例如电压信号) 转换为数字量信号，可以直接通过引线(IOA[0~6])输入。另外有一个通道只作为语音输入通道，通过内置有自动增益控制放大器的麦克风通道(MIC_IN)输入。实际上可以把模数转换器(ADC，Analog to Digital Converter)看作是一个实现模/数信号转换的编码器。在 ADC 内，由数模转换器 DAC0 和逐次逼近寄存器 SAR 组成逐次逼近式模-数转换器。向 P_ADC_Ctrl(写)($7015H)单元第 0 位(ADE)写入“1”用以激活 ADC。系统默认的设置为屏蔽 ADC(ADE=0)。

ADC 采用自动方式工作。硬件 ADC 的最高速率限定为(Fosc/32/12)Hz，如果速率超过此值，当从 P_ADC(读)($7014H)单元读出数据时会发生错误。

下表列出了 ADC 在各种系统时钟频率下的响应速率。

FOSC(MHz)	20.48	24.576	32.768	40.96	49.152
ADC 响应率(KHz)	640	768	1024	1280	1536

在 ADC 自动方式被启用后，会产生出一个启动信号，即 RDY=0。此时，DAC0 的电压模拟量输出值与外部的电压模拟量输入值进行比较，以尽快找出外部电压模拟量的数字量输出值。逐次逼近式控制首先将 SAR 中数据的最高有效位试设为‘1’，而其它位则全设为‘0’，即 10 0000 0000B。这时，DAC0 输出电压 V_{DAC0}(1/2 满量程)就会与输入电压 V_{in} 进行比较。如果 Vin>V_{DAC0}，则保持原先设置为‘1’的位(最高有效位)仍为‘1’；否则，该位会被清‘0’。接着，逐次逼近式控制又将下一位试设为‘1’，其余低位依旧设为‘0’，即 110000 0000B，V_{DAC0} 与 V_{in} 进行比较的结果若 Vin>V_{DAC0}，则仍保持原先设置位的值，否则便清‘0’该位。这个逐次逼近的过程一直会延续到 10 位中的所有位都被测试之后，A/D 转换的结果保存在 SAR 内。

当 10 位 A/D 转换完成时，RDY 会被置‘1’。此时，用户通过读取 P_ADC ($7014H)或 P_ADC_MUX_Data($702BH)单元可以获得 10 位 A/D 转换的数据。而从该单元读取数据后，又会使 RDY 自动清‘0’来重新开始进行 A/D 转换。若未读取 P_ADC ($7014H) 或 P_ADC_MUX_Data($702BH) 单元中的数据，RDY 仍保持为‘1’，则不会启动下一次的 A/D 转换。外部信号由 LIN_IN[1~7]即 IOA[0~6] 或通道 MIC_IN 输入。从 LIN_IN[1~7]输入的模拟信号直接被送入缓冲器 P_ADC_MUX_Data($702BH)；从 MIC_IN 输入的模拟信号则要经过缓冲器和放大器。AGC 功能将通过 MIC_IN 通道输入的模拟信号的放大值控制在一定范围内，然后放大信号经采样-保持模块被送至比较器参与 A/D 转换值的确定，最后送入 P_ADC ($7014H)。

11.1 P_ADC(读/写)($7014H)

该单元储存 MIC 输入的 A/D 转换的数据。逐次逼近式的 ADC 由一个 10 位 DAC(DAC0) 、一个 10 位缓存器 DAR0、一个逐次逼近寄存器 SAR 和一个比较器 COMP 组成。

b15 – b6	b5 – b0
DAR0(读/写)	---

P_ADC(读)：读出本单元实际为 A/D 转换输出的 10 位数字量。而且，如果 P_DAC_Ctrl ($702A) 单元第 3、4 位被设为‘00’，那么在转换过程里读出本单元($7014H)亦会触发 A/D 转换重新开始。

11.2 P_ADC_Ctrl(读/写)($7015H)

该单元为 ADC 的控制端口。

b15[2]	b6	b2	b0	控制功能描述
RDY (读)	DAC_I (写)	AGCE (写)	ADE (写)	
0	-	-	-	10 位模/数转换未完成
1	-	-	-	10 位模/数转换完成，输出 10 位数字量
	0			DAC 电流= 3mA @V_{DD}=3V[1]
	1			DAC 电流= 2mA @V_{DD}=3V
-	-	0	-	取消自动增益控制功能[3]
-	-	1	-	设置自动增益控制功能
-	-	-	0	禁止模/数转换工作
-	-	-	1	允许模/数转换工作

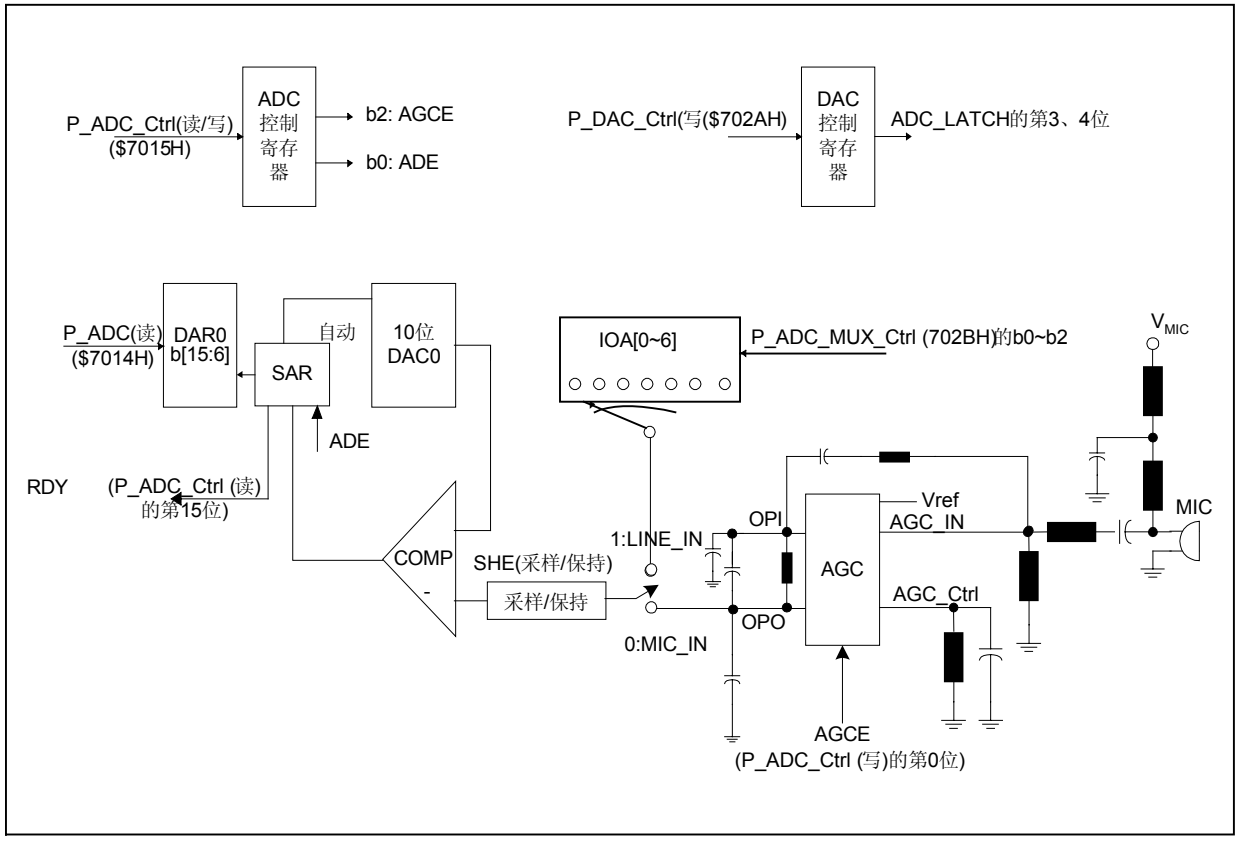
注：

[1] 此为 DAC_I 的缺省选择。

[2] b15 只用于 MIC_IN 通道输入

[3] 当模拟信号通过麦克风的 MIC_IN 通道输入时，可选择 AGCE 为‘1’，即运算放大器的增益可在其线性区域内自动调整。AGCE 缺省选择为‘0’，即取消自动增益控制功能。

[4] 写入时需注意 b5 = 1, b4=1, b3 =1 and b1=0.



11.3 P_ADC_MUX_Ctrl (读/写)($702BH)

ADC 多通道控制是通过对 P_ADC_MUX_Ctrl (702BH)单元编程实现的，见下表。

b15	b14	b13-b3	b2	b1	b0	控制功能描述
Ready_MUX (读)[1]	FAIL(读) [2]	-	Channel_sel(读/写)			
0		-	-	-	-	10 位模/数转换未完成
1	0	-	-	-	-	10 位模/数转换完成
-		-	0	0	0	模拟电压信号通过 MIC_IN 输入
-		-	0	0	1	模拟电压信号通过 LINE_IN1 输入
-		-	0	1	0	模拟电压信号通过 LINE_IN2 输入
-		-	0	1	1	模拟电压信号通过 LINE_IN3 输入
-		-	1	0	0	模拟电压信号通过 LINE_IN4 输入
-		-	1	0	1	模拟电压信号通过 LINE_IN5 输入
-		-	1	1	0	模拟电压信号通过 LINE_IN6 输入
-		-	1	1	1	模拟电压信号通过 LINE_IN7 输入

1. Ready_MUX 只用于 Line_in[7:1].
2. 一般情况下，该位总为‘0’。以下情况除外：由于 MIC_IN 的优先级高于 AD LINE_IN，所以在 LIN_IN AD 转换过程里又有 MIC_IN 时，若 AD 切换到 MIC 输入，原 LINE_IN 的数据会出现问题，此时 fail 被置为’1’。 MIC AD 完成之后，该位被清为’0’。

ADC 的多路 LINE_IN 输入将与 IOA[6~0]共用，即：

IOA6	IOA5	IOA4	IOA3	IOA2	IOA1	IOA0
LIN_IN 7	LIN_IN 6	LIN_IN 5	LIN_IN 4	LIN_IN 3	LIN_IN 2	LIN_IN 1

11.4 P_ADC_MUX_Data(读) ($702BH)

P_ADC_MUX_Data 单元用于读出 LINE_IN[7:1]10 位 ADC 转换的数字数据，即：

b15	b14	b13	B12	b11	b10	b9	b8	b7	b6
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

11.5 ADC 直流电气特性

ADC 直流电气项目	项目符号	最小值	典型值	最大值	单位
ADC 分辨率	RESO		10		bit
ADC 有效位数	ENOB	8			bit
ADC 信噪比	SNR	50			dB
ADC 积分非线性	INL			±4	LSB[1]
ADC 差分非线性	DNL			±2	LSB
ADC 转换率	F_{CONV}			96K[2]	Hz
电源电流 @Vdd=3V	I_{ADC}		3.4		mA
功耗 @Vdd=3V	P_{ADC}		10.2		mW

注：

[1] LSB 表示为最小有效单位，在 VRT=3V 的情况下，1LSB 为 2.93 mv。

[2] 此由最大采样率(Samplerate_max)得来，即 Samplerate_max =ADC 响应率/16=1536KHz/16=96KHz。

11.6 ADC 的交流电气特性

ADC 交流电气项目	项目符号	最小值	典型值	最大值	单位
ADC 建立时间	Tint	0.33			Sec.
ADC 采样时间	Tsh	2			μs
ADC 通道选择时间	Tsel	1			μs
VRT 开关时间	Tsw	1			μs
ADC 转换时间	Tconv	16			CLK

11.7 ADC 的功能管脚

ADC 管脚	I/O	功能描述
Vref2	O	2V 输出参考电压
MICP	I	MIC_IN 的正相输入
MICN	I	MIC_IN 的负相输入
AGC	I/O	自动增益控制端
MICOUT	O	麦克风的第一运放输出
VIN[7~1]	I	模拟信号源输入[7~1]，源自通用 I/O 口
OPI	I	麦克风的第二运放输入
VCM	O	ADC 参考电压
VRTPAD	I	外部 A/D 最高参考电压
V_{MIC}	I	麦克风的电源

12 DAC 方式音频输出

SPCE061A 提供的音频输出方式为双 DAC 方式。在此方式下，DAC1、DAC2 转换输出的模拟量电流信号分别通过 AUD1 和 AUD2 管脚输出，输入的数字量分别写入 P_DAC1(写) ($7017)和 P_DAC2(写) ($7016)单元。

12.1 P_DAC2(读/写)($7016H)

在 DAC 方式下，该单元是一个带 10 位缓冲寄存器 DAR2 的 10 位 D/A 转换单元(DAC2)。

b15 – b6	b5 – b0
DA2_Data(读/写)	---

P_DAC2(写)：通过此单元直接写入 10 位数据到 10 位缓存器 DAR2，来锁存 DAC2 的输入数字量值(无符号数)。

P_DAC2(读)：从 DAR2 内读出 10 位数据。

12.2 P_DAC1(读/写)($7017H)

该单元为一个带 10 位缓存器(DAR1)的 10 位 D/A 转换单元(DAC1)。用于向 DAR1 写入或从其中读出 10 位数据。

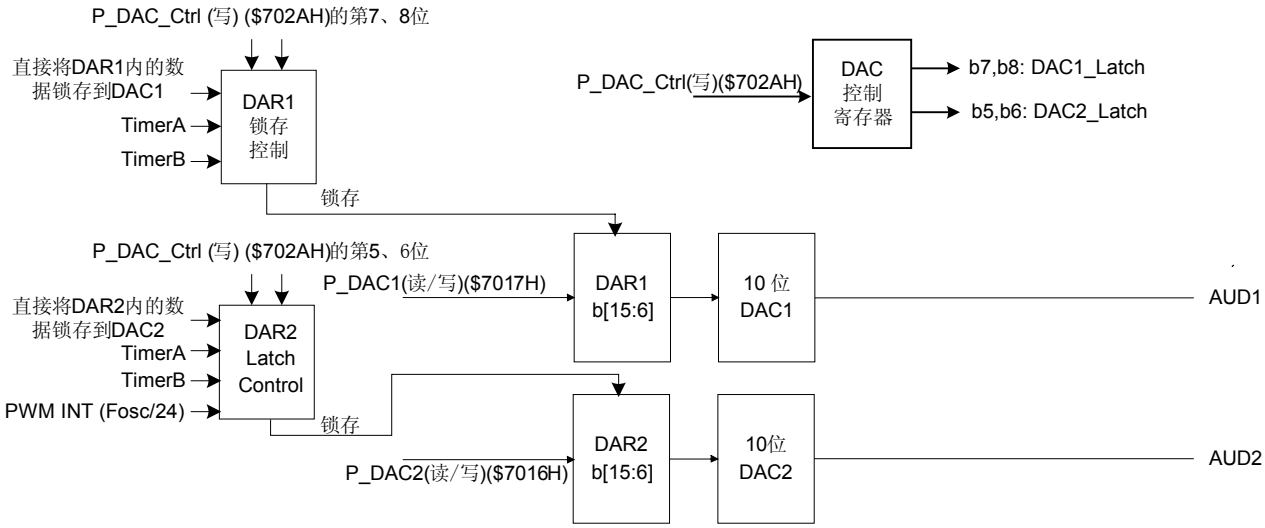
b15 – b6	b5 – b0
DA1_Data(读/写)	---

12.3 P_DAC_Ctrl(写) ($702AH)

DAC 音频输出方式的控制单元。第 5~8 位用于选择 DAC 输出方式下的数据锁存方式；第 3、4 位用来控制 A/D 转换方式。第 1 位总为'0'，用于双 DAC 音频输出。

b9~b15 为保留位。下表详细地列出了 P_DAC_Ctrl 单元的 b3~b8 的控制功能。

b8	b7	b6	b5	b4	b3
DAC1_Latch(写)		DAC2_Latch(写)		AD_Latch(写)	
00：直接将 DAR1 内数据锁存到 DAC1 内(缺省设置) 01：通过 TimerA 溢出将 DAR1 内的数据锁存到 DAC1 内 10：通过 TimerB 溢出将 DAR1 内的数据锁存到 DAC1 内 11：通过 TimerA 或 TimerB 的溢出将 DAR1 内的数据锁存到 DAC1 内		00：直接将 DAR2 内的数据锁存到 DAC2 内(缺省设置) 01：通过 TimerA 溢出将 DAR2 内的数据锁存到 DAC2 内 10：通过 TimerB 溢出将 DAR2 内的数据锁存到 DAC2 内 11：通过 TimerA 或 TimerB 的溢出将 DAR2 内的数据锁存到 DAC2 内		00：通过读 P_ADC(读)($7014H)触发 ADC 自动转换(缺省设置) 01：通过 TimerA 溢出触发 A/D 转换 10：通过 TimerB 溢出触发 A/D 转换 11：通过 TimerA 或 TimerB 的溢出触发 A/D 转换	


P_DAC_Ctrl (写) ($702AH)的第7、8位
直接将DAR1内的数据锁存到DAC1
TimerA
TimerB
DAR1 锁存控制
锁存
P_DAC_Ctrl(写)($702AH)
DAC 控制寄存器
b7,b8: DAC1_Latch
b5,b6: DAC2_Latch
P_DAC_Ctrl (写) ($702AH)的第5、6位
P_DAC1(读/写)($7017H)
直接将DAR2内的数据锁存到DAC2
TimerA
TimerB
PWM INT (Fosc/24)
DAR2 Latch Control
锁存
DAR1 b[15:6]
10 位 DAC1
AUD1
P_DAC2(读/写)($7016H)
DAR2 b[15:6]
10位 DAC2
AUD2

音频输出的结构

13 低电压监测/低电压复位 (LVD/LVR)

SPCE061A 可通过屏蔽选项设置或取消低电压监测和低电压复位功能，目的是为了通过对系统的电源电压进行监控，而使系统运行在一个正常、可靠的工作环境，并在一旦出现电源异常的情况下能立即采取相应的措施，使系统及时恢复正常。

13.1 低电压监测(LVD)

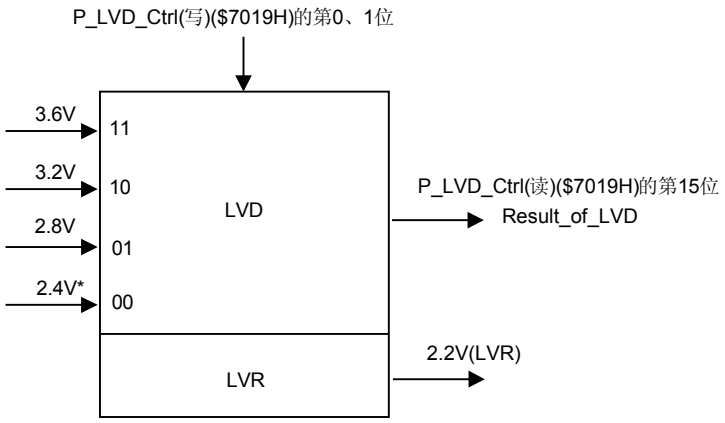
低电压监测功能可以提供系统内电源电压的使用情况。如果系统电压 V_{cc} 低于用户设定的电压监测低限电压 V_{LVD}，P_LVD_Ctrl 单元的第 15 位(LVD 监测标志位)将被置为“1”；反之，当 $V_{cc} > V_{LVD}$ 时，该位被置为“0”。

SPCE061A 具有 4 级电压监测低限：2.4V、2.8V、3.2 和 3.6V，可通过对 P_LVD_Ctrl 单元编程进行控制。假定 V_{LVD}=3.2V，当系统电压 V_{cc} 低于 3.2V 时，P_LVD_Ctrl 单元的第 15 位返回值为“1”，这样，CPU 可以通过可编程电压监测低限来完成低电压监测。系统默认的电压监测低限为 2.4V。

13.2 P_LVD_Ctrl(读/写)($7019H)

b15	b14 - b2	b1	b0
Result_of_LVD	---	LVD_level_define(写)	
0：$V_{DD} > V_{LVD}$ 1： $V_{DD} < V_{LVD}$			

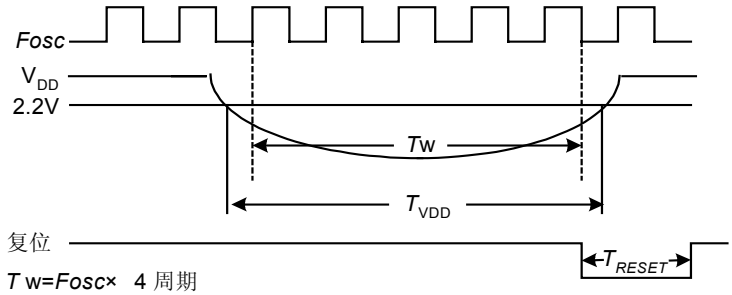
b1	b0	电压监测低限 (V_{LVD})
0	0	2.4V* (默认)
0	1	2.8V
1	0	3.2V
1	1	3.6V



13.3 低电压复位(LVR)

当电源电压低于 2.2V 时，系统会变得不稳定且易出故障。导致电源电压过低的原因很多，如电压的反跳、负载过重、电池能量不足……。如果系统设置了低电压复位(LVR)功能，当电源电压低于该值时，会在 4 个时钟周期之后产生一个复位信号，使系统复位。

当工作电压在 4 个时钟周期内降至 2.2V 以下，LVR 就会产生一个复位信号使系统复位，从而使系统恢复到初始工作状态。LVR 时序如下所示。



$T_w = Fosc \times$ 4 周期
$T_{VDD} > T_w$
$T_{RESET} = Fosc \times$ 512 周期

14 串行设备输入输出端口 SIO

串行输入输出端口 SIO 提供了一个 1 位的串行接口，用于与其它设备进行数据通讯。在 SPCE061A 内通过 IOB0 和 IOB1 这 2 个端口实现与设备进行串行数据交换功能。其中，IOB0 用来作为时钟端口(SCK)，IOB1 则用来作为数据端口(SDA)，用于串行数据的接收或发送。参见IOB 口的特殊功能。

14.1 P_SIO_Ctrl(读/写)($701EH)

用户必须通过设置 P_SIO_Ctrl(读/写)($701EH)单元的第 7 位，将 IOB7、IOB1 分别设置为 SCK 管脚和 SDA 管脚。如果该单元的第 6 位被设置为“0”，串行输入/输出接口可以从用户指定的地址读出数据。该单元的第 3、4 位的作用是让用户自行指定数据传输速度；而通过设置第 0、1 位，可以指定串行设备的寻址位数。

SIO 的控制选择在 P_SIO_Ctrl 单元内进行，见下表：

b7	b6	b5	b4	b3	b2	b1	b0	设置功能说明
IO	R/W	R/W_EN	Speed_Sel		—	Addr_Sel		
X	X	X	X	X	—	0	0	串行设备地址(缺省)设置为 16 位(A0~A15)
X	X	X	X	X	—	0	1	无地址设置
X	X	X	X	X	—	1	0	串行设备地址设置为 8 位(A0~A7)
X	X	X	X	X	—	1	1	串行设备地址设置为 24 位(A0~A23)
X	X	X	0	0	—	X	X	数据传输速率设为 *CPUClk*/16(缺省设置)
X	X	X	0	1	—	X	X	无用
X	X	X	1	0	—	X	X	数据传输速率设为 *CPUClk*/8
X	X	X	1	1	—	X	X	数据传输速率设为 *CPUClk*/32
1	X	X	X	X	—	X	X	设置 IOB0=SCK(串行接口时钟端口)，IOB1=SDA(串行接口数据端口) 。用户不必设置 IOB0 和 IOB1 的输入输出状态
0	X	X	X	X	—	X	X	用作普通的 I/O 口(默认)
X	1	X	X	X	—	X	X	设置数据帧的写传输
X	0	X	X	X	—	X	X	设置数据帧的读传输(默认)
X	X	1	X	X	—	X	X	关断读/写帧的传输
X	X	0	X	X	—	X	X	接通读/写帧的传输(默认)

关于 CPU 时钟 CPU_CLK 请参考系统时钟部分。

14.2 P_SIO_Data(读/写)($701AH)

该单元为接收/发送串行数据的缓冲单元。向该单元写入或读出数据，可按串行方式发送或接收数据字节。用户须通过写入 P_SIO_Start ($701FH)单元来启动 P_SIO_Data ($701AH)单元与串行设备数据交换的过程。传输是从串行设备的起始地址(由 P_SIO_Addr_Low, P_SIO_Addr_Mid 和 P_SIO_Addr_High3 个单元指定)开始，然后是数据。

进行写操作时，第一次向 P_SIO_Data (写) ($701AH)单元写入数值是在写入 P_SI_Start(写)单元任意一个数值之后，随后，SIO 将从串行设备的起始地址开始传送，后面接着传送写入 P_SIO_Data 单元中的 8 位数据。

进行读操作时，第一次读 P_SIO_Data (读) ($701AH)单元数据是在向 P_SIO_Start (写) ($701FH)单元写入任一数值后，SIO 将首先传送串行设备的起始地址。

b7	b6	b5	b4	b3	b2	b1	b0
D7	D6	D5	D4	D3	D2	D1	D0

14.3 P_SIO_Addr_Low(读/写)($701BH)

串行设备起始地址的低字节(默认值为 00H)。

b7	b6	b5	b4	b3	b2	b1	b0
A7	A6	A5	A4	A3	A2	A1	A0

14.4 P_SIO_Addr_Mid(读/写)($701CH)

串行设备起始地址的中字节(默认值为 00H)。

b7	b6	b5	b4	b3	b2	b1	b0
A15	A14	A13	A12	A11	A10	A9	A8

14.5 P_SIO_Addr_High(读/写)($701DH)

串行设备起始地址的高字节(默认值为 00H)。

b7	b6	b5	b4	b3	b2	b1	b0
A23	A22	A21	A20	A19	A18	A17	A16

14.6 P_SIO_Start(读/写)($701FH)

向 P_SIO_Start(写)($701FH)单元写入任意一个数值，可以启动数据传输过程。接着，当对 P_SIO_Data ($701AH)单元读写操作时会使 SIO 根据 P_SIO_Addr_Low、 P_SIO_Addr_Mid 和 P_SIO_Addr_High 的内容传输读/写操作的起始地址，之后再读写 P_SIO_Data 单元时 SIO 将不再传输此起始地址。

如果需要重新指定一个起始地址进行数据传输，用户可以向 P_SIO_Stop ($7020H)单元写入任意一个数值以停止 SIO 操作，然后向 P_SIO_Addr_Low、 P_SIO_Addr_Mid 和 P_SIO_Addr_High 写入新的地址；最后，向 P_SIO_Start(写)($701FH)单元写入任意一个数值重新启动 SIO 操作。

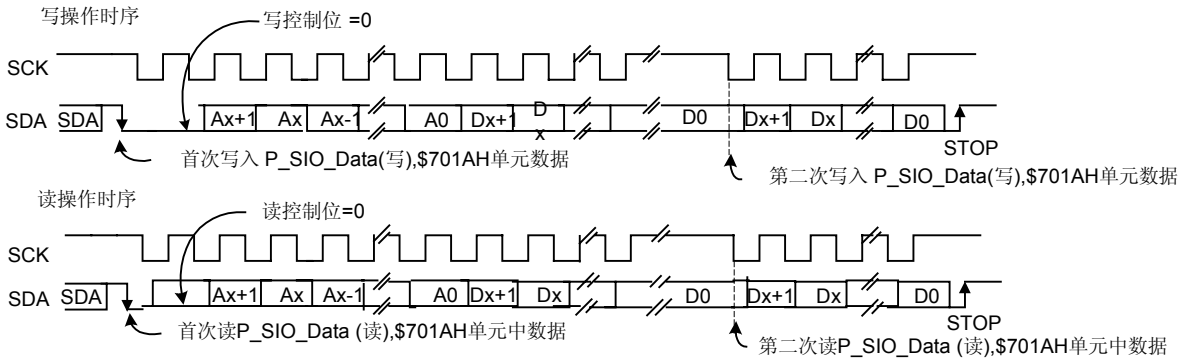
读出 P_SIO_Start($701FH)单元可获取 SIO 的数据传输状态，该单元的第 7 位 Busy 为占用标志位，Busy=‘1’表示正在传输数据，传输操作完成后，该位将被清为‘0’，可以开始传输新的数据字节。

b7	b6	b5	b4	b3	b2	b1	b0
Busy	-	-	-	-	-	-	-

14.7 P_SIO_Stop(写)($7020H)

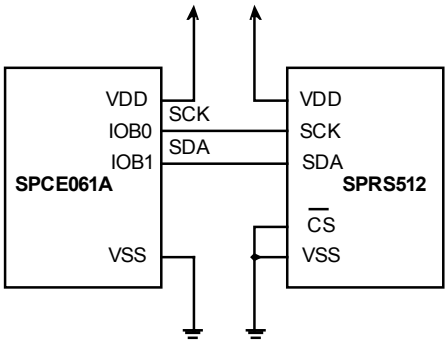
向 P_SIO_Stop(写)($7020H)单元写入任一数值，可以停止数据传输。通常，停止数据传输的终止指令应出现在激活数据传输的启动指令之前。但上电复位后的第一个启动命令之前不需要终止命令。

14.8 读/写时序



14.9 应用举例

通过串行输入/输出接口将 SPCE061A 与 SPRS512C 串行静态 RAM(SSRAM，Serial Static RAM)连接起来。在数据传输过程里，SPRS512 会自动在读/写操作的起始地址的基础上递增地址计数器。



[例 14.1]

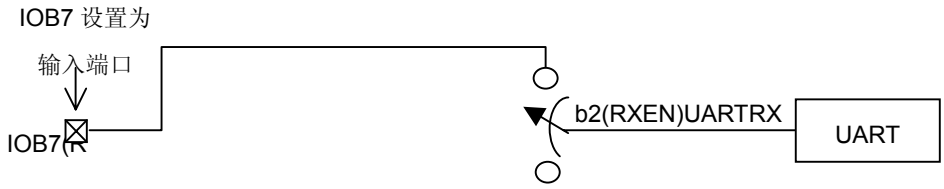
从串行输入/输出接口向 SSRAM 写入一个长度为 1 个字节的数据。SPRS512 有 24 位地址线，工作频率为 CPUCLK/4，可进行串行数据帧的写操作。注意，首先应将 SIO 设置为输出口。

```
//设置串行输入/输出控制端口(输出状态)
  R1=0x00CB；
  [P_SIO_Ctrl]=R1；
//写入串行输入输出启动端口
  R1=0x0000；
  [P_SIO_Start]=R1；
//写入起始地址到地址单元
   R1=0x0000；
   [P_SIO_Addr_High]=R1；          // Addr23~Addr16 =0000 0000
   [P_SIO_Addr_Mid]=R1；           //Addr15~Addr8 =0000 0000
   [P_SIO_Addr_Low]=R1；           //Addr7~Addr0 =0000 0000
 //向缓冲单元写入传送数据
    R1=0x0011；
    [P_SIO_Data]=R1；              //d7~d0 =0001 0001
 //读出占用标志 Busy
  L_busy:
  R1=0x0080；
  Test R1, [P_SIO_Start]；
  JNZ L_Busy；
  //向 P_SIO_Start(写)($701FH)单元写入任意一个数值，结束数据帧的写操作
  [P_SIO_Stop]=R1；
```

15 通用异步串行接口 UART

UART 模块提供了一个全双工标准接口，用于完成 SPCE061A 与外设之间的串行通讯。借助于 IOB 口的特殊功能和 UART IRQ 中断，可以同时完成 UART 接口的接收发送数据的过程。此外，UART 还可以缓冲地接收数据。也就是说，它可以在读取缓存器内当前数据之前接收新的数据。但是，如果新的数据被接收到缓存器之前一直未从中读取先前的数据，会发生数据丢失。P_UART_Data (读/写)($7023H)单元可以用于接收和发送数据的缓存。向该单元写入数据，可以将发送的数据送入缓存器；从该单元读数据，可以从缓存器读出单个的数据字节。UART 模块的接收管脚 Rx 和发送管脚 Tx 分别可与 IOB7 和 IOB10 共用。

通过两 UART 接收管脚 Rx(IOB7)来接收数据)， 须保证 P_IR_Ctrl ($7018H)单元的第 2 位 RXEN 被置为 ‘0’。



P_IR_Ctrl(读/写)($7018H)

b15	b14 - b3	b2	b1	b0
---	---	RXEN(写)	---	---

使用 UART 模块进行通讯时，必须事先分别将管脚 Rx(IOB7)、Tx(IOB10)设置为输入状态、输出状态。然后，通过设置 P_UART_BaudScalarLow ($7024H)、P_UART_BaudScalarHigh ($7025H)单元指定所需波特率。同时，设置 P_UART_Command1($7021H)和 P_UART_Command2 ($7022H) 单元以激活 UART 通讯功能。以上设置完成后，UART 将处于激活状态。设置 P_UART_Command1 单元的第 6、7 位可以激活 UART IRQ 中断，并决定中断是由 TxRDY 或 RxRDY 信号触发以及由二者共同触发。设置 P_UART_Command2 单元的第 6、7 位可以激活 UART Tx、Rx 管脚功能。当μ'nSP™接收或发送一个字节数据时，P_UART_Command2 ($7022H)单元的第 6、7 位被置为“1”且同时触发 UART IRQ。无论 UART IRQ 中断是否被激活，UART 接收/发送功能都可以由 P_UART_Command2 ($7022H)单元的第 6、7 位控制。在任意时刻读出 P_UART_Command2 ($7022H)单元将清除 UART IRQ 中断标志。

注意，UART IRQ 中断向量存储在 0FFFFH 单元，相对于其它 IRQ 中断来说，该中断的优先级别最低。

15.1 UART 数据帧的格式

15.2 P_UART_Command1(写)($7021H)

该单元为 UART 控制端口。设置该单元的第 2、3 位可以控制数据奇偶校验功能。第 6、7 位控制着 UART IRQ 中断，二者的区别在于：如果第 6 位 TxIntEn=1，中断由 TxRDY 信号触发，即数据发送完毕将产生 UART IRQ 中断；如果第 7 位 RxIntEn=1，中断由 RxRDY 信号触发，即数据接收完毕将产生 UART IRQ 中断。如果该单元的第 5 位 I_Reset=1，所有 UART 控制寄存器、状态寄存器将恢复为系统默认值。

b7	b6	b5	b4	b3	b2	b1	b0	功能
RxIntEn	TxIntEn	I_Reset	-	Parity	P_Check	-	-	
1	—	—	—	—	—	—	—	允许 UART IRQ 中断(由 RxRDY 信号触发)
0	—	—	—	—	—	—	—	禁止 UART IRQ 中断
—	1	—	—	—	—	—	—	允许 UART IRQ 中断(由 RxRDY 信号触发)
—	0	—	—	—	—	—	—	禁止 UART IRQ 中断
—	—	1	—	—	—	—	—	内部复位信号复位
—	—	0	—	—	—	—	—	内部复位信号置位
—	—	—	—	1	—	—	—	激活偶校检功能
—	—	—	—	0	—	—	—	激活奇校检功能
—	—	—	—	—	1	—	—	激活奇偶校验功能
—	—	—	—	—	0	—	—	屏蔽奇偶校检功能

P_UART_Command1(写)($7021H)单元的缺省值为$00。

15.3 P_UART_Command2(写)($7022H)

该单元写入时为 UART 数据发送/接收控制端口，第 6、7 位分别控制着数据发送和接收管脚的允通/禁通。P_UART_Command2(写)($7022H)单元的缺省值为$00。

b7	b6	b5	b4	b3	b2	b1	b0
RxPinEn	TxPinEn	-	-	-	-	-	-
1：允通接收管脚 0：禁通接收管脚	1：允通发送管脚 0：禁通发送管脚						

此时 IOB7 和 IOB10 必须分别被设置为输入和输出管脚作为 Rx 和 Tx 管脚。当发送管脚被允通时，

IOB10 Tx 输出管脚将自动被置为高电平。

15.4 P_UART_Command2(读)($7022H)

该单元读出为 UART 状态信息。第 7 位是 RxRDY 标志位，当接收到数据时该标志位被置为“1”，读 P_UART_Data 单元将清除该标志位；第 6 位是 TxRDY 标志位，当通过写入本单元第 6 位为'1'来允通发送管脚后，该标志位被置为“1”，表示发送器的数据缓存器为空，已准备好可以发送写入 P_UART_Data 单元的数据。

向 P_UART_Data 单元写入数据可以清除 TxRDY 标志位。读出 P_UART_Command2 ($7022H)单元的第 3~5 位是传输错误标志位，如果在传输过程中发生错误，相应位将被置为“1”；读 P_UART_Data($7023H)单元数据将清除错误标志位。

b7	b6	b5	b4	b3	b2	b1	b0
RxRDY	TxRDY	FE	OE	PE	-	-	-
1：数据已接收完毕 0：未接收到数据	1：数据发送已准备好 0：数据发送未准备好	1：存在帧错误 0：无帧错误	1：存在溢出错误 0：无溢出错误	1：存在奇偶校验错误 0：无奇偶校验错误			

以上错误信号代表传输过程可能出现的错误。下表给出了出错的原因及解决方法。

错误类型	原因	解决方法
FE (帧错误)	发送管脚 TX 和接收管脚 RX 的数据帧的格式或波特率不一致	1. 使用一致的数据格式 2. 设置一致的波特率
OE (溢出错误)	接收端 RX 接收数据的速度低于发送端 TX 发送数据的速度，从而导致 RX 端数据溢出	1. 提高接收数据的速度 2. 降低数据传输速度
PE (奇偶校验错误)	传输条件差，可能有噪声干扰	改善传输条件

15.5 P_UART_Data(读/写)($7023H)

b7	b6	b5	b4	b3	b2	b1	b0
数据							

15.6 P_UART_BaudScalarLow(读/写)($7024H)

15.7 P_UART_BaudScalarHigh (读/写)($7025)

P_UART_BaudScalarHigh ($7025)和 P_UART_BaudScalarLow($7024H)单元的组合控制着数据传输速率(波特率)。UART 波特率的计算公式如下：

波特率波特率= (Fosc / 2) / Scale

其中：

Scale=(Fosc/2)/波特率(Scale 为 7024H 单元和 7025H 单元组成的十进制整数)

Fosc=24.576MHz(或 20.480MHz)，取决于 P_SystemClock 单元的第 5 位。

下表列示出当 Fosc = 24.576MHz 时常用的波特率值。

波特率(bps)	高字节($7025H)	低字节($7024H)	Scale (十进制)	实际波特率(bps)
1500 (最小值)	1FH	FFH	8192	1500
2400	14H	00H	5120	2400
4800	0AH	00H	2560	4800
9600	05H	00H	1280	9600
19200	02H	80H	640	19200
38400	01H	40H	320	38400
48000 (默认值)	01H*	00H*	256	48000
51200	00H	F0H	240	51200
57600	00H	D5H	213	57690
102400	00H	78H	120	102400
115200 (最大值)	00H	6BH	107	114841

[例 15 .1]:

用 UART 来接收 PC 机的 RS232 串行接口的数据。由于 UART 每次只接收 8 位数据，所以在接收了 2 个 8 位数据之后才将其存入 16 位的 SRAM 内。

```
//将 IOB10、IOB7 分别设为输出、输入状态
        R1=0x0480；
        [P_IOB_Attrib]=R1；
        R1=0x0400；
        [P_IOB_Dir]=R1；
//设置波特率= 12.288MHz/107 = 114.84KHz (与 115.2KHz 最接近，所以可以选择 115.2KHz)
        R1=0x006b；
        [P_UART_BaudScalarLow]=R1；
        R1=0x00；
        [P_UART_BaudScalarHigh]=R1；
//设置 UART_command1 和 UART_command2 单元
        R1=0xc0；
        [P_UART_Command1]=R1；
        [P_UART_Command2]=R1；
//主程序
 L_begin_loop：
        R4=0；                          // SRAM 的接收数据队列的起始地址
 L_loop：
        R2=0；
        R2=R2 LSL 4；                   //清除移位缓存器
```

```
            CALL F_UART_RECV；
            R1=R1 LSL 4；                    //左移 8 位
            R1=R1 LSL 4；
            R3=R1；
            R3=R3 and 0xFF00；               //R3 =高 8 位
            R2=0；
            R2=R2 LSL 4；                    //清除移位缓存器
            CALL F_UART_RECV；
            R1=R1 and 0x00FF；               //R1 =低 8 位
            R1=R1 or R3；                    //R1=从 UART 接收来的最终数据字
            [R4++]=R1；                      //将接收的数据字存入 SRAM 的数据队列
            CMP R4，0x800；                  //SRAM 中的数据队列接收到 2048 个数据字
            JNE L_loop；
            JMP L_begin_loop；
// UART 子程序
F_UART_RECV：
      PUSH R2，R3 to [SP]；
 L_RxRDY：
            R2= 0X0080；                     //检查 RxRDY 是否为 1
TEST R2，  [P_UART_Command2]；
            JZ    L_RxRDY
            R1 = [P_UART_Data]；
POP R2，R3 from [SP]；
            RETF；                           //返回到主程序
```

15.8 端口总述

配置单元	读写属性	存储地址	配置功能说明
P_IOA_Data	读/写	$7000H	写入数据到数据寄存器里，读出 IOA 管脚上的电平状态
P_IOA_Buffer	读/写	$7001H	写入数据到数据寄存器里,读出数据寄存器里的数据
P_IOA_Dir	读/写	$7002H	I/O A 口的方向向量
P_IOA_Attrib	读/写	$7003H	I/O A 口的属性向量
P_IOA_Latch	读	$7004H	进入睡眠前锁存 I/O A 口的数据，为睡眠后触键引起唤醒准备
P_IOB_Data	读/写	$7005H	写入数据到数据寄存器里，读出 I/O B 口管脚上的电平状态
P_IOB_Buffer	读/写	$7006H	写入数据到数据寄存器里,读出数据寄存器里的数据
P_IOB_Dir	读/写	$7007H	I/O B 口的方向向量
P_IOB_Attrib	读/写	$7008H	I/O B 口的属性向量
P_FeedBack	写	$7009H	通过反馈电路将外部 RC 振荡源引入
P_TimerA_Data	读/写	$700AH	I/O A 口的数据口
P_TimerA_Ctrl	写	$700BH	I/O A 口的控制单元
P_TimerB_Data	读/写	$700CH	I/O B 口的数据口
P_TimerB_Ctrl	写	$700DH	I/O B 口的控制单元
P_Timebase_Setup	写	$700EH	时基发生器的选频设置单元
P_Timebase_Clear	写	$700FH	时基计数器复位单元
P_INT_Ctrl	读/写	$7010H	中断源的控制单元
P_INT_Clear	写	$7011H	中断源中断请求清除单元
P_SystemClock	写	$7013H	系统时钟选频单元(包括系统进入睡眠状态的时钟频率选择)
P_ADC	读/写	$7014H	ADC 的数据口
P_ADC_Ctrl	读/写	$7015H	ADC 的控制单元
P_DAC2	读/写	$7016H	DAC2 的数据口
P_DAC1	读/写	$7017H	DAC1 的数据口
P_IR_Ctrl	读/写	$7018H	红外通讯的控制单元
P_LVD_Ctrl	读/写	$7019H	低电压监测的控制单元
P_SIO_Data	读/写	$701AH	串行设备接口 SIO 的数据口
P_SIO_Addr_Low	读/写	$701BH	SIO 的低字节地址单元
P_SIO_Addr_Mid	读/写	$701CH	SIO 的中字节地址单元
P_SIO_Addr_High	读/写	$701DH	SIO 的高字节地址单元
P_SIO_Ctrl	读/写	$701EH	SIO 的控制单元
P_SIO_Start	读/写	$701FH	SIO 数据传输的启动单元
P_SIO_Stop	写	$7020H	SIO 数据传输的结束单元
P_UART_Command1	写	$7021H	通用异步串行 I/O 口 UART 的控制单元
P_UART_Command2	读/写	$7022H	UART 接收/发送功能的开通/关断控制单元
P_UART_Data	读/写	$7023H	UART 的数据口
P_UART_BaudScalarLow	读/写	$7024H	UART 波特率设定控制字的低字节
P_UART_BaudScalarHigh	读/写	$7025H	UART 波特率设定控制字的高字节
P_DAC_Ctrl	读/写	$702AH	音频输出方式及双 DAC 通道和单 PWM 驱动通道的控制单元
P_ADC_MUX_Ctrl	读/写	$702BH	ADC 多通道控制
P_ADC_MUX_Data	读	$702CH	读出多通道 10 位 ADC 转换的数字数据
P_INT_Ctrl_New	(读/写)	$702DH	激活和屏蔽中断

16 保密设定

如果希望将内部的闪存进行保密设定，可将 P_{FUSE} 接 5V, P_{VIN} 接 GND 并维持 2s 以上即可将内部保险丝熔化，此后就无法再完成 download, debug 等功能。

17 语音编码类型

SPCE061A 支持的语音编码类型如下表所示：

编码方式	编码名称	功能解释
脉冲编码调制方式	PCM	
对数脉冲编码调制方式	LOG PCM	
凌阳音频编码方式	SACM_A3200	
	SACM_A2000	以 24K，20K，16K 位/秒的速率编码/解码
	SACM_S720	以 7.2K 位/秒的速率编码/解码
	SACM-S480	以 4.8K 位/秒的速率编码/解码
	SACM_S240	以 2.4K 位/秒的速率编码/解码
	SACM_MS01	用于音调合成器的 FM 合成器和波表合成器