



九齐科技股份有限公司  
Nyquest Technology Co., Ltd.

数  
据  
手  
册

# NY8A053D

---

## 12 I/O 8 位 EPROM-Based 单片机

**Version 1.4**

**Apr. 17, 2018**

本文内容是由英文规格书翻译, 目的是为了您的阅读更加方便。它无法跟随原稿的更新, 敬请参考英文规格书以获得更准确的信息。

NYQUEST TECHNOLOGY CO. reserves the right to change this document without prior notice. Information provided by NYQUEST is believed to be accurate and reliable. However, NYQUEST makes no warranty for any errors which may appear in this document. Contact NYQUEST to obtain the latest version of device specifications before placing your orders. No responsibility is assumed by NYQUEST for any infringement of patent or other rights of third parties which may result from its use. In addition, NYQUEST products are not authorized for use as critical components in life support devices/systems or aviation devices/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 NYQUEST.

## 改版记录

版本	日期	内容描述	修正页
1.0	2016/08/24	新发布。	-
1.1	2016/11/15	<ol style="list-style-type: none"> <li>1. 新增引脚叙述。</li> <li>2. 修改表 16 配置字节。</li> </ol>	<p style="text-align: center;">9</p> <p style="text-align: center;">64</p>
1.2	2017/05/23	修改配置字节表中的输入电压窗口水平。	64
1.3	2017/11/15	修改表 14 /TO 和/PD 值和相关事件概述。	47
1.4	2018/04/17	<ol style="list-style-type: none"> <li>1. 修改 PB3 的引脚叙述。</li> <li>2. 修改 T1CR1 的 Bit7、Bit6 读写属性。<span style="color: red;">(仅简中版修改)</span></li> <li>3. 修改直流电气特性表格。</li> </ol>	<p style="text-align: center;">9</p> <p style="text-align: center;">22</p> <p style="text-align: center;">65</p>

## 目 录

<b>1. 概述 .....</b>	<b>6</b>
1.1 功能 .....	6
1.2 系统框图 .....	8
1.3 引脚图 .....	8
1.4 引脚说明 .....	9
<b>2. 内存结构 .....</b>	<b>10</b>
2.1 程序存储器 .....	10
2.2 数据存储器 .....	11
<b>3. 功能概述 .....</b>	<b>14</b>
3.1 R-page特殊功能寄存器 .....	14
3.1.1 <i>INDF</i> (间接寻址寄存器) .....	14
3.1.2 <i>TMR0</i> (定时器 0 寄存器) .....	14
3.1.3 <i>PCL</i> (程序计数器低字节) .....	14
3.1.4 <i>STATUS</i> (状态寄存器) .....	15
3.1.5 <i>FSR</i> (数据指针寄存器) .....	15
3.1.6 <i>PortA</i> (PortA数据寄存器) .....	16
3.1.7 <i>PortB</i> (PortB数据寄存器) .....	16
3.1.8 <i>PCON</i> (Power寄存器) .....	16
3.1.9 <i>BWUCON</i> (PortB唤醒控制寄存器) .....	17
3.1.10 <i>PCHBUF</i> (程序计数器高字节) .....	17
3.1.11 <i>ABPLCON</i> (PortA/PortB下拉电阻控制寄存器) .....	17
3.1.12 <i>BPHCON</i> (PortB上拉电阻控制寄存器) .....	18
3.1.13 <i>INTE</i> (中断使能寄存器) .....	18
3.1.14 <i>INTF</i> (中断标志寄存器) .....	19
3.2 <i>T0MD</i> 定时器 0 控制寄存器 .....	19
3.3 F-page特殊功能寄存器 .....	20
3.3.1 <i>IOSTA</i> (PortA I/O控制寄存器) .....	20
3.3.2 <i>IOSTB</i> (PortB I/O控制寄存器) .....	21
3.3.3 <i>PS0CV</i> (预分频器 0 寄存器) .....	21
3.3.4 <i>BODCON</i> (PortB开漏控制寄存器) .....	21
3.3.5 <i>PCON1</i> (Power控制寄存器 1) .....	21

3.4 S-page特殊功能寄存器 .....	22
3.4.1 TMR1 (定时器 1 寄存器) .....	22
3.4.2 T1CR1 (定时器 1 控制寄存器 1) .....	22
3.4.3 T1CR2 (定时器 1 控制寄存器 2) .....	23
3.4.4 PWM1DUTY (PWM1 占空比寄存器) .....	24
3.4.5 PS1CV (预分频器 1 寄存器) .....	24
3.4.6 BZ1CR (蜂鸣器 1 控制寄存器) .....	24
3.4.7 IRCR (IR控制寄存器) .....	25
3.4.8 TBHP (表格指针高字节寄存器) .....	26
3.4.9 TBHD (表格数据高字节寄存器) .....	26
3.4.10 OSCCR (振荡器控制寄存器) .....	27
3.5 I/O Port .....	27
3.5.1 IO引脚结构框图 .....	29
3.6 定时器 0 .....	36
3.7 定时器 1/PWM1/Buzzer1 .....	37
3.8 红外线载波 (IR) .....	39
3.9 看门狗定时器 (WDT) .....	40
3.10 中断 .....	40
3.10.1 Timer0 上溢中断 .....	41
3.10.2 Timer1 下溢中断 .....	41
3.10.3 看门狗超时中断 .....	41
3.10.4 PB输入状态改变中断 .....	41
3.10.5 外部中断输入 .....	41
3.11 振荡器配置 .....	41
3.12 工作模式 .....	43
3.12.1 正常模式 .....	44
3.12.2 慢速模式 .....	45
3.12.3 待机模式 .....	45
3.12.4 睡眠模式 .....	45
3.12.5 唤醒稳定时间 .....	46
3.12.6 工作模式概述 .....	46
3.13 复位 .....	46
<b>4. 指令设置 .....</b>	<b>48</b>
<b>5. 配置字节表 .....</b>	<b>64</b>

<b>6. 电气特性</b> .....	<b>65</b>
6.1 最大绝对值 .....	65
6.2 直流电气特性 .....	65
6.3 特性图 .....	67
6.3.1 高速RC振荡频率与电源电压曲线图 .....	67
6.3.2 高速RC振荡频率与温度曲线图 .....	67
6.3.3 低速RC振荡频率与电源电压曲线图 .....	68
6.3.4 低速RC振荡频率与温度曲线图 .....	68
6.4 建议工作电压 .....	68
6.5 LVR电压与温度曲线图 .....	69
<b>7. 芯片脚位坐标图</b> .....	<b>69</b>
<b>8. 封装尺寸</b> .....	<b>70</b>
8.1 8 引脚SOP（150 毫寸） .....	70
8.2 14 引脚SOP（150 毫寸） .....	70
8.3 14 引脚DIP（300 毫寸） .....	71
<b>9. 订购信息</b> .....	<b>71</b>

## 1. 概述

NY8A053D是以EPROM作为存储器的8位单片机，专为多组I/O口产品的应用而设计。例如遥控器、风扇/灯光控制或是游乐器周边等等。采用CMOS制程并同时提供客户低成本、高性能、及高性价比等。NY8A053D核心建立在RISC精简指令集架构可以很容易地做编程和控制，共有55条指令。除了少数指令需要两个指令时钟，大多数指令都是一个指令时钟能完成，可以让用户轻松地以程序控制完成不同的应用。因此非常适合各种中低记忆容量但又复杂的应用。

在I/O的资源方面，NY8A053D有12根弹性的双向I/O脚，每个I/O脚都有单独的寄存器控制为输入或输出脚。而且每一个I/O脚位都能通过控制相关的寄存器达成如上拉或下拉电阻或开漏（Open-Drain）输出。此外针对红外线遥控的产品方面，NY8A053D内置了可选择频率的红外载波发射口。

NY8A053D有两组定时器，可用系统时钟当作一般的计时应用或者从外部讯号触发来计数。另外NY8A053D提供1组8位的PWM输出或蜂鸣器输出，用来驱动马达、LED、或蜂鸣器等等。

NY8A053D采用双时钟机制，高速振荡时钟或者低速振荡时钟都由内部RC振荡或外部晶振输入。在双时钟机制下，NY8A053D可选择多种工作模式如正常模式（Normal）、慢速模式（Slow mode）、待机模式（Standby mode）与睡眠模式（Halt mode），可节省电力消耗，延长电池寿命。并且单片机在使用内部RC高速振荡时，低速振荡可以同时使用外部精准的晶振计时。可以维持高速处理同时又能精准计算真实时间。

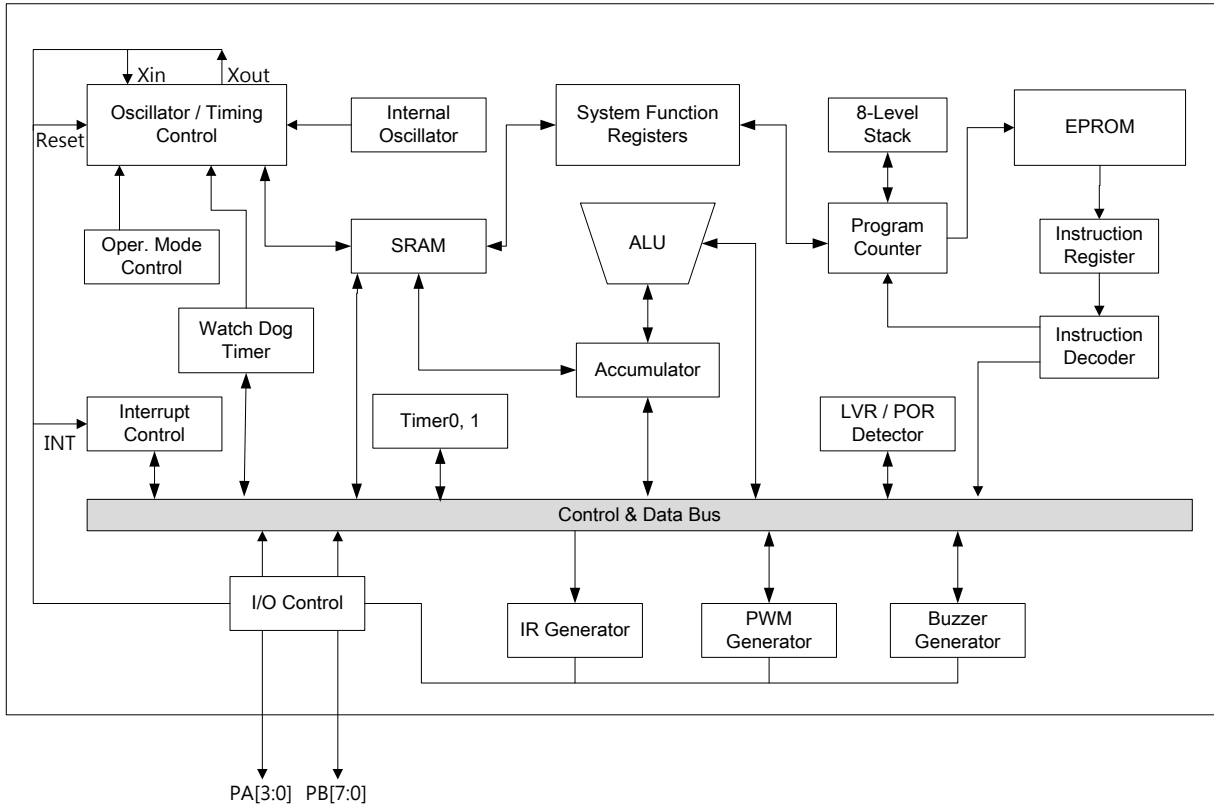
在省电的模式下，如待机模式（Standby mode）与睡眠模式（Halt mode）中，有多个中断源可以触发来唤醒NY8A053D进入正常操作模式（Normal mode）或慢速模式（Slow mode）来处理突发事件。

### 1.1 功能

- 宽广的工作电压：（指令时钟为4个CPU时钟，亦即4T模式）
  - 2.0V ~ 5.5V @系统时钟 ≤8MHz。
  - 2.2V ~ 5.5V @系统时钟 >8MHz。
- 宽广的工作温度：-40°C ~ 85°C。
- 1Kx14位的程序存储器空间。
- 64字节的通用数据存储器空间。
- 12根可分别单独控制输入输出方向的I/O脚（GPIO）、PA[3:0]、PB[7:0]。
- PA[3:0]及PB[3:0]可选择输入时使用内部下拉电阻。
- PB[7:4]及PB[2:0]可选择输入时使用内部上拉电阻。
- PB[5:0]可选择内部上拉电阻或开漏输出（Open-Drain）。
- PB[3]可选择当作输入或开漏输出（Open-Drain）。
- 8级深度硬件堆栈（Stack）。
- 访问数据有直接或间接寻址模式。
- 一组8位上数定时器（Timer0）包含可编程的预分频器。
- 一组8位下数定时器（Timer1）可选自动重载或连续下数计时。
- 一个8位的脉冲宽度调变输出（PWM1）。

- 一个蜂鸣器输出（BZ1）。
- 38/57KHz红外线载波（IR）频率可供选择，同时载波之极性也可以通过寄存器选择。
- 红外载波发射口。
- 内置上电复位电路（POR）。
- 内置低压复位功能（LVR）。
- 内置看门狗计时（WDT），可由配置字节（Configuration Word）控制开关。
- 双时钟机制，系统时钟可以随时切换高速振荡或者低速振荡。
  - 高速振荡时钟：
    - E\_HXT（超过 6MHz外部高速晶振）
    - E\_XT（455K~6MHz外部晶振）
    - I\_HRC（1~20MHz内部高速RC振荡）
  - 低速振荡时钟：
    - E\_LXT（32KHz外部低速晶振）
    - I\_LRC（内部 32KHz低速RC振荡）
- 四种工作模式可随系统需求调整电流消耗：正常模式（Normal mode）、慢速模式（Slow mode）、待机模式（Standby mode）与睡眠模式（Halt mode）。
- 五种硬件中断：
  - Timer0 上溢中断。
  - Timer1 下溢中断。
  - WDT中断。
  - PB输入状态改变中断。
  - 外部中断输入。
- NY8A053D在待机模式（Standby mode）下的五种唤醒中断：
  - Timer0 上溢中断。
  - Timer1 下溢中断。
  - WDT中断。
  - PB输入状态改变中断。
  - 外部中断输入。
- NY8A053D在睡眠模式（Halt mode）下的三种唤醒中断：
  - WDT中断。
  - PB输入状态改变中断。
  - 外部中断输入。

1.2 系统框图



1.3 引脚图

NY8A053D提供三种封装类型：SOP14, DIP14 及SOP8。

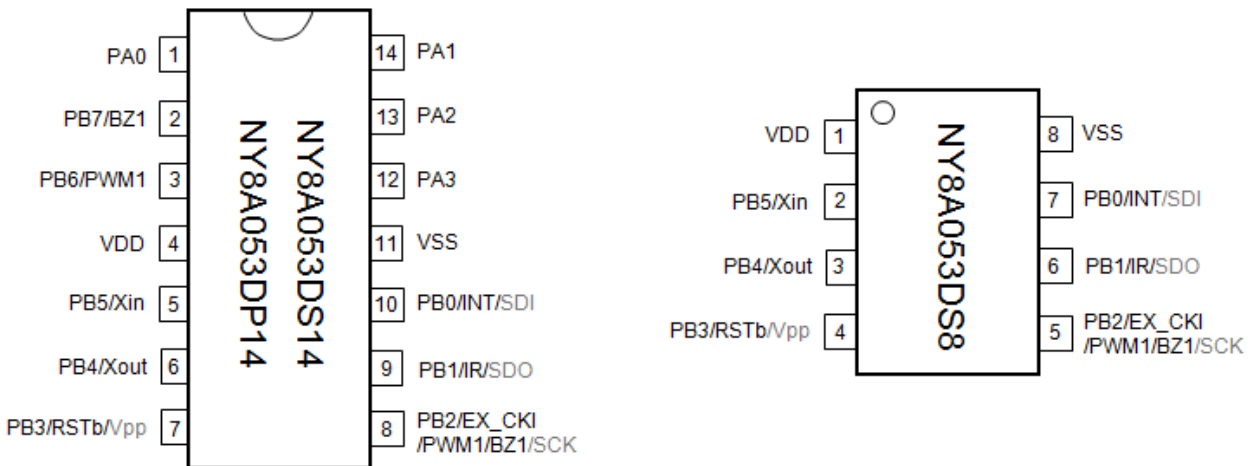


图 1 封装引脚图



**1.4 引脚说明**

引脚名	I/O	描述
PA0 ~ PA3	I/O	双向I/O引脚。
PB0/ INT/ SDI	I/O	PB0 是一个双向I/O引脚。 PB0 是外部中断输入引脚当EIS=1&INTIE=1。 PB0 也是编程数据输入SDI。
PB1/ IR/ SDO	I/O	PB1 是一个双向I/O引脚。 如果启用红外模式，该引脚为红外载波输出。 PB1 也是编程数据输出SDO
PB2/ EX_CK1/ PWM1/ BZ1/ SCK	I/O	PB2 是一个双向I/O引脚。 PB2 可当作定时器 0 / 1 外部时钟来源EX_CK1。 PB2 也是编程时钟输入SCK。 PB2 依配置字节设定可输出PWM1。 PB2 依配置字节设定可输出BZ1。 PWM1 比BZ1 有更高的优先权。
PB3/ RSTb/ Vpp	I/O	PB3 可选择当作输入脚或开漏输出脚。 PB3 可当作复位引脚RSTb。当RSTb引脚为低电压水平，将会重置NY8A053D。 当VPP超过 <b>7.75V</b> ，将会让NY8A053D进入EPROM烧写模式。
PB4/ Xout	I/O	PB4 是一个双向I/O引脚，当用于I_HRC与I_LRC。 PB4 可当作晶振输出引脚Xout，当用于E_HXT, E_XT或E_LXT。 PB4 也可以当成指令时钟输出。
PB5/ Xin	I/O	PB5 是一个双向I/O引脚，当用于I_HRC与I_LRC。 PB5 可当作晶振输入引脚Xin，当用于E_HXT, E_XT或E_LXT。
PB6/ PWM1	I/O	PB6 是一个双向I/O引脚。 PB6 依配置字节设定可输出PWM1。
PB7/ BZ1	I/O	PB7 是一个双向I/O引脚。 PB7 依配置字节设定可输出BZ1。
VDD	-	电源正端。
VSS	-	电源负端。

## 2. 内存结构

NY8A053D存储器分为两类：分别是程序存储器和数据存储器。

### 2.1 程序存储器

NY8A053D程序存储器空间是 1Kx14 位。因此，10 位宽的计数器（PC）可以访问程序存储器的任何地址。

复位地址位于 0x000，软件中断地址位于 0x001，内部和外部硬件中断地址位于 0x008。NY8A053D提供CALL、GOTOA和CALLA等指令去访问程序空间的 256 个地址。还提供GOTO指令去访问程序空间 512 个地址，LCALL和LGOTO指令访问程序空间的任何地址。

当发生子程序调用或中断情况时，下一个ROM地址写入堆栈的顶部。而当执行RET、RETIA或RETIE指令，堆栈顶部的数据会被读取并加载到程序计数器。

NY8A053D程序存储器地址 0x00E~0x00F和 0x3FE~0x3FF是保留地址。如果用户在这些地址写入程序可能会发生无法预期的程序执行错误。

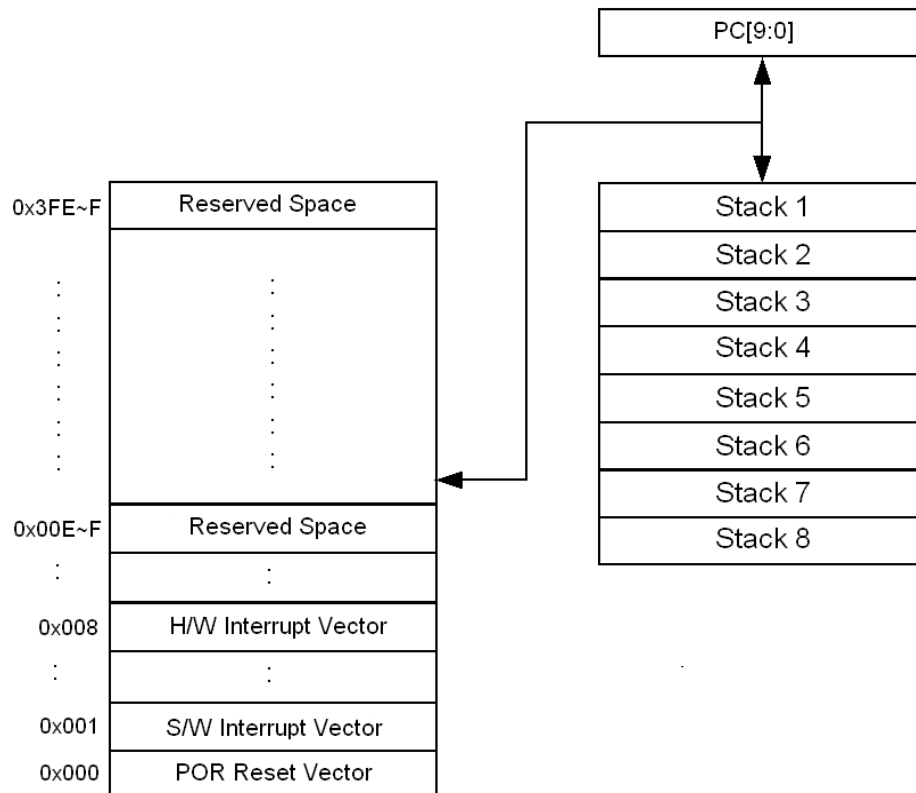


图 2 程序存储器对映地址

## 2.2 数据存储

根据用于访问数据存储器的指令，数据存储器可分为三类：**R-page**特殊功能寄存器（SFR）和通用寄存器（GPR）、**F-page**特殊功能寄存器、**S-page**特殊功能寄存器。GPR是由SRAM组成，用户可以使用它们来存储变量或计算结果。

R-page特殊功能寄存器和数据存储器分为四组Bank，可透过数据指针寄存器（FSR）来切换Bank。寄存器BK[1:0]为FSR[7:6]，可从四个Bank中选择其中一个。

R-page特殊功能寄存器和数据存储器可用直接寻址方式和间接寻址方式来进行访问。

数据存储器使用间接寻址方式如下图所描述，这种间接寻址方式包含使用INDF寄存器。Bank选择是由FSR[7:6]决定，地址选择则是由FSR[5:0]而定。

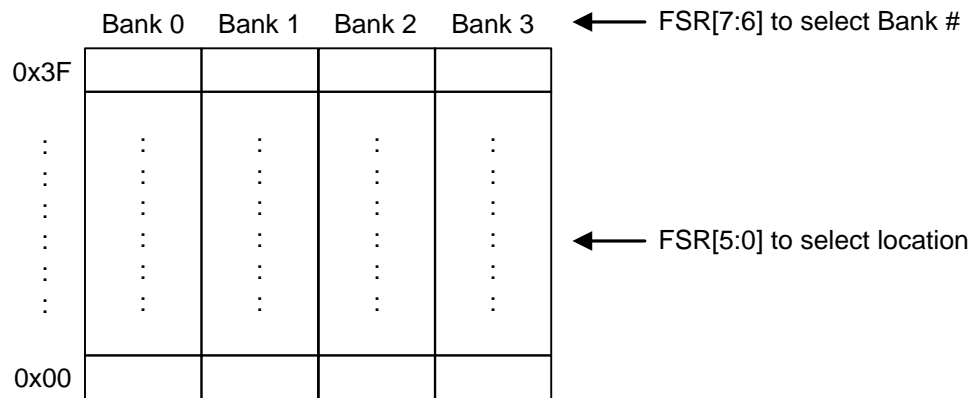


图 3 间接寻址方式访问数据存储器

下面描述了数据存储器使用的直接寻址方式。Bank选择是由寄存器FSR[7:6]决定，而地址选择则是由指令码OP-Code[5:0]直接决定。

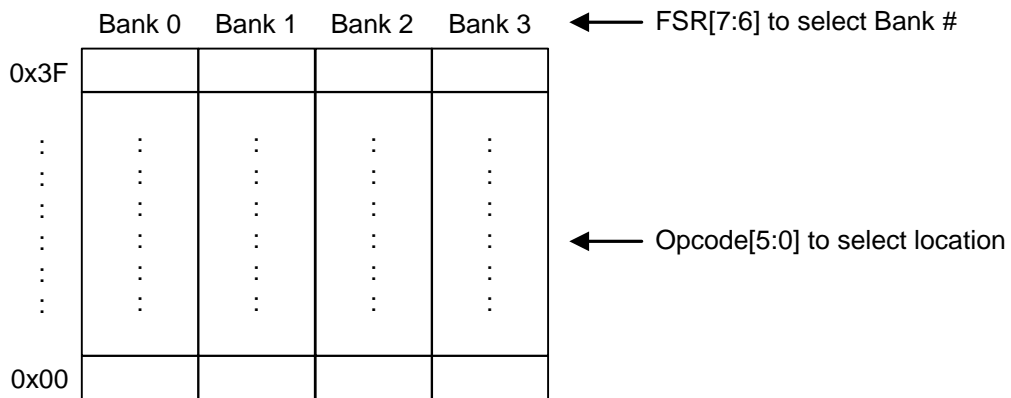


图 4 直接寻址方式访问数据存储器

R-page特殊功能寄存器可以通过一般的指令访问，如算术指令和数据搬移指令。R-page特殊功能寄存器占用了从Bank 0 的 0x0 到 0xF。然而，Bank 1、Bank 2 和Bank 3 的相同地址会映像到Bank 0。换句话说，R-page特殊功能寄存器只存在于Bank 0。GPR占用了Bank 0 数据存储器的 0x10 到 0x3F地址与Bank1 地址 0x10 到 0x1F，如表 1 所示。

NY8A053D寄存器名称和R-page特殊功能寄存器的映像地址说明如下表。

地址 \ FSR[7:6]	00 (Bank 0)	01 (Bank 1)	10 (Bank 2)	11 (Bank 3)
0x0	INDF	映射至 <i>Bank 0</i>		
0x1	TMR0			
0x2	PCL			
0x3	STATUS			
0x4	FSR			
0x5	PORTA			
0x6	PORTB			
0x7	-			
0x8	PCON			
0x9	BWUCON			
0xA	PCHBUF			
0xB	ABPLCON			
0xC	BPHCON			
0xD	-			
0xE	INTE			
0xF	INTF			
0x10 ~ 0x1F	通用寄存器	通用寄存器	映射至 <i>bank 0</i>	映射至 <i>Bank 1</i>
0x20 ~ 0x3F	通用寄存器	映射至 <i>bank 0</i>	映射至 <i>bank 0</i>	映射至 <i>bank 0</i>

表 1 R-page特殊功能寄存器地址映像表

F-page特殊功能寄存器只能被指令IOST和IOSTR访问，S-page特殊功能寄存器只能被指令SFUN和SFUNR访问。当F-page和S-page寄存器被访问时，STATUS[7:6]选择位会被忽略。寄存器名称和F-page、S-page的地址说明如下表。

地址	特殊功能寄存器种类	F-page SFR	S-page SFR
0x0		-	TMR1
0x1		-	T1CR1
0x2		-	T1CR2
0x3		-	PWM1DUTY
0x4		-	PS1CV
0x5		IOSTA	BZ1CR
0x6		IOSTB	IRCR
0x7		-	TBHP
0x8		-	TBHD
0x9		-	-
0xA		PS0CV	-
0xB		-	-
0xC		BODCON	-
0xD		-	-
0xE		-	-
0xF		PCON1	OSCCR

表 2 F-page特殊功能寄存器和S-page特殊功能寄存器地址表

### 3. 功能概述

本章节将详细描述NY8A053D的操作方式。

#### 3.1 R-page特殊功能寄存器

##### 3.1.1 INDF（间接寻址寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INDF	R	0x0	INDF[7:0]							
读写属性			读/写							
初始值			XXXXXXXX							

间接寻址寄存器并不是真的存在，而是以间接寻址模式来使用。任何指令访问间接寻址寄存器时，实际上是访问数据指针寄存器FSR所选择的寄存器。

##### 3.1.2 TMR0（定时器 0 寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0	R	0x1	TMR0[7:0]							
读写属性			读/写							
初始值			XXXXXXXX							

当读取TMR0 寄存器时，会得到定时器 0 目前计数数值。

当写入TMR0 寄存器时，会更新定时器 0 目前计数数值。

通过设置T0MD与配置字节（Configuration Word），定时器 0 时钟源可以从指令时钟F<sub>INST</sub>、外部时钟EX\_CK1或低频振荡器I\_LRC/E\_LXT中择一。

##### 3.1.3 PCL（程序计数器低字节）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCL	R	0x2	PCL[7:0]							
读写属性			读/写							
初始值			0x00							

程序计数器（PC）是一个 10 位寄存器，分高 2 位和低 8 位。当程序执行了一个指令，同时PC数值会增加，除了某些指令会直接更改PC数值。PCL寄存器可访问PC低字节（PC[7:0]），PC高字节（PC[9:8]）并不能直接访问，必须通过PCHBUF寄存器完成访问。

以GOTO指令来说，PC[8:0]是从指令码（OP-Code）取得，而PC[9]是从PCHBUF[1]加载。CALL指令的PC[7:0]是从指令码取得，而PC[9:8]是从PCHBUF[1:0]加载。下一个PC地址（PC+1），将会存到堆栈的顶部。LGOTO指令的PC[9:0]是从指令码取得。

LCALL指令的PC[9:0]是从指令码取得，下一个PC地址（PC+1），将被存到堆栈的顶部。

### 3.1.4 STATUS（状态寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	R	0x3	GP7	GP6	GP5	/TO	/PD	Z	DC	C
读写属性			读/写	读/写	读/写	读/写(*2)	读/写(*1)	读/写	读/写	读/写
初始值			0	0	0	1	1	X	X	X

状态寄存器包含算术/逻辑指令的结果和是否发生看门狗超时复位。

**C:** 进位/借位标志位。

C=1 时，加法运算有进位或减法运算无借位。

C=0 时，加法运算无进位或减法运算有借位。

**DC:** 半进位/半借位标志位。

DC=1 时，加法运算低四位有进位或减法运算时没有向高四位借位。

DC=0 时，加法运算低四位无进位或减法运算时有向高四位借位。

**Z:** 零位。

Z=1 时，算术或逻辑运算的结果是零。

Z=0 时，算术或逻辑运算的结果不为零。

**/PD:** 睡眠模式标志位。

/PD=1 时，上电或执行CLRWDT指令后。

/PD=0 时，执行SLEEP指令后。

**/TO:** 看门狗超时标志位。

/TO=1 时，上电或执行CLRWDT或SLEEP指令后。

/TO=0 时，发生WDT上溢。

**GP7, GP6, GP5:** 通用寄存器数据位。

(\*1) 可以被SLEEP指令清除。

(\*2) 可以由CLRWDT指令设定。

### 3.1.5 FSR（数据指针寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSR	R	0x4	BK[1:0]		FSR[5:0]					
读写属性			读/写							
初始值			0	0	X	X	X	X	X	X

**FSR[5:0]:** 从指定Bank数据存储器的 64 个寄存器中选择一个。

BK[1:0]: Bank 选择位，BK[1:0]=00b 选择Bank0，BK[1:0]=01b 选择Bank1，BK[1:0]=10b 选择Bank2。

BK[1:0]=11b，选择Bank3。

### 3.1.6 PortA (PortA 数据寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PortA	R	0x5	GP7	GP6	GP5	GP4	PA3	PA2	PA1	PA0
读写属性			读/写							
初始值			数据锁存器值是xxxx，读取值则是xxxx端口值 (PA3~PA0)							

读取PortA时，若特定脚位被配置为输入脚，将得到该脚位输入状态。然而，若该脚位被配置为输出脚，依据配置字节，得到该脚位的状态或相对应的输出数据锁存值。当写入PortA时，数据是被写入PortA的输出数据锁存器中。

**GP7~4:** 通用寄存器数据位。

### 3.1.7 PortB (PortB 数据寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PortB	R	0x6	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
读写属性			读/写							
初始值			数据锁存器值是xxxxxxxx，读取值则是xxxxxxxx端口值 (PB7~PB0)							

读取PortB时，若特定脚位被配置为输入脚，将得到该脚位输入状态。然而，若该脚位被配置为输出脚，依据配置字节，得到该脚位的状态或相对应的输出数据锁存值。当写入PortB时，数据是被写入PortB的输出数据锁存器中。

### 3.1.8 PCON (Power 寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCON	R	0x8	WDTEN	EIS	GP5	GP4	LVREN	GP2	GP1	GP0
读写属性			读/写							
初始值			1	0	0	0	1	0	0	0

**GP5, GP4, GP2~GP0:** 通用寄存器数据位。

**LVREN:** 开启/关闭 LVR。

LVREN=1 时，开启LVR。

LVREN=0 时，关闭LVR。

**EIS:** 外部中断引脚选择位。

EIS=1 时，PB0 选择为外部中断引脚。

EIS=0 时，PB0 选择为GPIO。

**WDTEN:** 开启/关闭 WDT。

WDTEN=1 时，开启WDT。

WDTEN=0 时，关闭WDT。



**3.1.9 BWUCON (PortB 唤醒控制寄存器)**

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BWUCON	R	0x9	WUPB7	WUPB6	WUPB5	WUPB4	WUPB3	WUPB2	WUPB1	WUPB0
读写属性			读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写
初始值			1	1	1	1	1	1	1	1

**WUPBx:** 开启/关闭PBx唤醒功能,  $0 \leq x \leq 7$ 。

WUPBx=1 时, 开启PBx唤醒功能。

WUPBx=0 时, 关闭PBx唤醒功能。

**3.1.10 PCHBUF (程序计数器高字节)**

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCHBUF	R	0xA	-	XSPD_STP		-		GP5	PCHBUF[2:0]	
读写属性			-	写		-		读/写		
初始值			X	0		X		000		

**PCHBUF[1:0]:** 程序计数器PC的第九个位到第八个位。

**XSPD\_STP:** 写 1 停止外部晶振 32.768KHz起振强化功能。

**GP5:** 通用寄存器数据位。

**3.1.11 ABPLCON (PortA/PortB 下拉电阻控制寄存器)**

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ABPLCON	R	0xB	/PLPB3	/PLPB2	/PLPB1	/PLPB0	/PLPA3	/PLPA2	/PLPA1	/PLPA0
读写属性			读/写							
初始值			1	1	1	1	1	1	1	1

**/PLPAx:** 关闭/开启PAx下拉电阻,  $0 \leq x \leq 3$ 。

/PLPAx=1, 关闭PAx下拉电阻。

/PLPAx=0, 开启PAx下拉电阻。

**/PLPBx:** 关闭/开启PBx下拉电阻,  $0 \leq x \leq 3$ 。

/PLPBx=1, 关闭PBx下拉电阻。

/PLPBx=0, 开启PBx下拉电阻。

### 3.1.12 BPHCON (PortB 上拉电阻控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BPHCON	R	0xC	/PHPB7	/PHPB6	/PHPB5 (*1)	/PHPB4 (*1)	GP3	/PHPB2	/PHPB1	/PHPB0
读写属性			读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写
初始值			1	1	1	1	1	1	1	1

**/PHPBx:** 关闭/开启PBx上拉电阻,  $0 \leq x \leq 7$ 。

/PHPBx=1 时, 关闭PBx上拉电阻。

/PHPBx=0 时, 开启PBx上拉电阻。

**GP3:** 通用寄存器数据位。

**注意:** PB4 与 PB5 作为晶振引脚时应关闭内置上拉电阻。

### 3.1.13 INTE (中断使能寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTE	R	0xE	-	WDTIE	-	-	T1IE	INTIE	PBIE	T0IE
读写属性			-	读/写	-	-	读/写	读/写	读/写	读/写
初始值			X	0	X	X	0	0	0	0

**T0IE:** 定时器 0 上溢 (overflow) 中断使能位。

T0IE=1 时, 开启定时器 0 上溢中断。

T0IE=0 时, 关闭定时器 0 上溢中断。

**PBIE:** PortB输入状态变化中断使能位。

PBIE=1 时, 开启PortB输入状态变化中断。

PBIE=0 时, 关闭PortB输入状态变化中断。

**INTIE:** 外部中断使能位。

INTIE=1 时, 开启外部中断。

INTIE=0 时, 关闭外部中断。

**T1IE:** 定时器 1 下溢 (underflow) 中断使能位。

T1IE=1 时, 开启定时器 1 下溢中断。

T1IE=0 时, 关闭定时器 1 下溢中断。

**WDTIE:** WDT上溢中断使能位。

WDTIE=1 时, 开启WDT上溢中断。

WDTIE=0 时, 关闭WDT上溢中断。

### 3.1.14 INTF（中断标志寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF	R	0xF	-	WDTIF	-	-	T1IF	INTIF	PBIF	T0IF
读写属性			-	读/写	-	-	读/写	读/写	读/写	读/写
初始值 (Note)			X	0	X	X	0	0	0	0

**TOIF:** 定时器 0 上溢中断标志位。

TOIF=1 时，发生定时器 0 上溢中断。

TOIF必须由程序清零。

**PBIF:** PortB输入状态变化中断标志位。

PBIF=1 时，发生PortB输入状态变化中断。

PBIF必须由程序清零。

**INTIF:** 外部中断标志位。

INTIF=1 时，发生外部中断。

INTIF必须由程序清零。

**T1IF:** 定时器 1 下溢中断标志位。

T1IF=1 时，发生定时器 1 下溢中断。

T1IF必须由程序清零。

**WDTIF:** WDT超时上溢标志位。

WDTIF=1 时，发生WDT上溢中断。

WDTIF必须由程序清零。

**注意:** 当对应的INTE寄存器控制位未使能，读取中断标志是 0。

### 3.2 T0MD定时器 0 控制寄存器

T0MD是可读写寄存器，但只能由指令T0MD / T0MDR访问。

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0MD	-	-	LCKTM0	INTEDG	T0CS	T0CE	PS0WDT	PS0SEL[2:0]		
读写属性			读/写							
初始值 (Note)			0	0	1	1	1	111		

**PS0SEL[2:0]:** 选择预分频器 0 的预分频比 (Dividing Rate)。预分频器 0 根据PS0WDT控制位决定分配给定时器 0 或WDT。当预分频器 0 被分配给WDT，预分频比取决于选择哪种计数机制 (WDT复位或WDT中断)。

PS0SEL[2:0]	预分频比选项		
	PS0WDT=0 (定时器 0)	PS0WDT=1 (WDT复位)	PS0WDT=1 (WDT中断)
000	1:2	1:1	1:2
001	1:4	1:2	1:4

PS0SEL[2:0]	预分频比选项		
	PS0WDT=0 (定时器 0)	PS0WDT=1 (WDT复位)	PS0WDT=1 (WDT中断)
010	1:8	1:4	1:8
011	1:16	1:8	1:16
100	1:32	1:16	1:32
101	1:64	1:32	1:64
110	1:128	1:64	1:128
111	1:256	1:128	1:256

表 3 预分频器 0 的预分频比选项

**PS0WDT:** 预分频器 0 分配选择。

PS0WDT=1 时，预分频器 0 被分配到WDT。

PS0WDT=0 时，预分频器 0 被分配到定时器 0。

**注意：**在使能看门狗或定时器中断前，要先设定PS0WDT和PS0SEL[2:0]，否则复位或中断可能导致错误触发。

**T0CE:** 定时器 0 外部时钟源触发沿选择。

T0CE=1 时，下降沿时定时器 0 加一。

T0CE=0 时，上升沿时定时器 0 加一。

**注意：**T0CE应用在外部 EX\_CKI脚作为定时器 0 时钟源。

**T0CS:** 定时器 0 时钟源选择。

T0CS=1 时，选择EX\_CKI脚或低频震荡I\_LRC/E\_LXT。

T0CS=0 时，选择指令时钟F<sub>INST</sub>。

**INTEDG:** INT沿触发选择位。

INTEDG=1 时，引脚PB0 输入上升沿将触发INTIF。

INTEDG=0 时，引脚PB0 输入下降沿将触发INTIF。

**LCKTM0:** T0CS=0 时，指令时钟F<sub>INST</sub>被选作定时器 0 时钟源。

T0CS=1 时，LCKTM0=0 时，外部EX\_CKI脚被选择当作定时器 0 时钟源。

T0CS=1 时，LCKTM0=1 时，低频震荡I\_LRC/E\_LXT为定时器 0 时钟源。

**注意：**有关定时器 0 时钟源选择的详细说明，请参考定时器 0 章节。

### 3.3 F-page特殊功能寄存器

#### 3.3.1 IOSTA (PortA I/O 控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOSTA	F	0x5	-	-	-	-	IOPA3	IOPA2	IOPA1	IOPA0
读写属性			-	-	-	-	读/写	读/写	读/写	读/写
初始值			X	X	X	X	1	1	1	1

**IOPAx:** PAx I/O模式选择,  $0 \leq x \leq 3$ 。

IOPAx=1 时, PAx设为输入口。

IOPAx=0 时, PAx设为输出口。

### 3.3.2 IOSTB (PortB I/O 控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOSTB	F	0x6	IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
读写属性			读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写
初始值			X	X	X	X	1	1	1	1

**IOPBx:** PBx I/O模式选择,  $0 \leq x \leq 7$ 。

IOPBx=1 时, PBx设为输入口。

IOPBx=0 时, PBx设为输出口。

### 3.3.3 PS0CV (预分频器 0 寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PS0CV	F	0xA	PS0CV[7:0]							
读写属性			读							
初始值			1	1	1	1	1	1	1	1

读取PS0CV时, 会得到预分频器 0 器的目前计数值。

### 3.3.4 BODCON (PortB 开漏控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BODCON	F	0xC	ODPB7	ODPB6	ODPB5	ODPB4	GP3	ODPB2	ODPB1	ODPB0
读写属性			读/写	读/写	读/写	读/写	读/写	读/写	读/写	读/写
初始值			0	0	0	0	0	0	0	0

**ODPBx:** 开启/关闭PBx的开漏,  $0 \leq x \leq 7$ 。

ODPBx=1 时, 开启PBx的开漏。

ODPBx=0 时, 关闭PBx的开漏。

**GP3:** 通用寄存器数据位。

### 3.3.5 PCON1 (Power 控制寄存器 1)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCON1	F	0xF	GIE	-	GP5	GP4	GP3	GP2	GP1	T0EN
读写属性			读/写(1*)	-	读/写	读/写	读/写	读/写	读/写	读/写
初始值			0	0	0	0	0	0	0	1

**T0EN:** 开启/关闭定时器 0。

T0EN=1 时，开启定时器 0。

T0EN=0 时，关闭定时器 0。

**GIE:** 开启/关闭总中断使能位。

GIE=1 时，开启总中断。

GIE=0 时，关闭总中断。

**GP5~GP1:** 通用寄存器数据位。

**(1\*):** 由指令 ENI 设置 1、指令 DISI 清除、指令 IOSTR 所读取。

### 3.4 S-page特殊功能寄存器

#### 3.4.1 TMR1（定时器 1 寄存器）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1	S	0x0	TMR1[7:0]							
读写属性			读/写							
初始值			XXXXXXXX							

当读取TMR1 寄存器时，会得到定时器 1 目前计数数值。

当写入TMR1 寄存器时，会更新定时器 1 目前计数数值。

#### 3.4.2 T1CR1（定时器 1 控制寄存器 1）

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CR1	S	0x1	PWM1OEN	PWM1OAL	-	-	-	T1OS	T1RL	T1EN
读写属性			只写	只写	-	-	-	读/写	读/写	读/写
初始值			0	0	X	X	X	0	0	0

此寄存器用于配置定时器 1 功能。

**T1EN:** 开启/关闭定时器 1。

T1EN=1 时，开启定时器 1。

T1EN=0 时，关闭定时器 1。

**T1RL:** 当连续模式被选择（T1OS=0），选择定时器 1 下数方式。

T1RL=1 时，当下溢发生，定时器 1 初始值从TMR1 寄存器被重新加载并继续下数。

T1RL=0 时，当下溢发生，定时器 1 继续从 0xFF下数。

**T1OS:** 当下溢发生，设置定时器 1 操作模式。

T1OS=1 时，单次计数模式（One-Shot mode）。定时器 1 会从初始值到 0x00 计数一次。

T1OS=0 时，连续计数模式（Non-Stop mode）。下溢后，定时器 1 会持续下数。

T1OS	T1RL	定时器 1 计数选项
0	0	定时器 1 从 0xFF 倒数到 0x00。 当下溢发生, 0xFF 被重载至定时器 1 并继续下数。
0	1	定时器 1 从重载的数值倒数到 0x00。 当下溢发生, 定时器 1 从 TMR1 重新载入数值并继续下数。
1	x	定时器 1 从初始值下数到 0x00。 当下溢发生, 定时器 1 停止下数。

表 4 定时器 1 功能

**PWM1OAL:** 定义 PWM1 输出有效状态。

PWM1OAL=1 时, PWM1 为低电平有效位输出。

PWM1OAL=0 时, PWM1 为电平高有效位输出。

**PWM1OEN:** 开启/关闭 PWM1 输出。

PWM1OEN=1, PB6/PB2 输出 PWM1。

PWM1OEN=0, PB6/PB2 为 GPIO。

### 3.4.3 T1CR2 (定时器 1 控制寄存器 2)

名称	SFR 类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CR2	S	0x2	-	-	T1CS	T1CE	/PS1EN	PS1SEL[2:0]		
读写属性			-	-	读/写	读/写	读/写	读/写	读/写	读/写
初始值			X	X	1	1	1	1	1	1

该寄存器用于配置定时器 1 功能。

**PS1SEL[2:0]:** 预分频器 1 预分频比选项。

PS1SEL[2:0]	预分频比选项
000	1:2
001	1:4
010	1:8
011	1:16
100	1:32
101	1:64
110	1:128
111	1:256

表 5 预分频器 1 预分频比选项

**注意:** 在 PS1EN=1 前须先设定 PS1SEL[2:0], 否则可能会误发生中断。

**/PS1EN:** 关闭/开启预除器 1。

/PS1EN=1 时, 关闭预分频器 1。

/PS1EN=0 时, 开启预分频器 1。

**T1CE:** 定时器 1 外部时钟触发沿选项。

T1CE=1 时, EX\_CKI脚下降沿时定时器 1 减一。

T1CE=0 时, EX\_CKI脚上升沿时定时器 1 减一。

**T1CS:** 定时器 1 时钟源选项。

T1CS=1 时, 选择EX\_CKI脚作为外部时钟输入。

T1CS=0 时, 选择指令时钟F<sub>INST</sub>。

### 3.4.4 PWM1DUTY (PWM1 占空比寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1DUTY	S	0x3	PWM1DUTY[7:0]							
读写属性			写							
初始值			XXXXXXXX							

此寄存器仅能写入。当定时器 1 被使能并开始倒计时后, PWM1 输出会保持在非有效位状态。当定时器 1 数值等于PWM1DUTY, PWM1 输出有效位状态直到发生下溢。

定时器 1 重新加载的数值储存在TMR1 寄存器, 以用来定义PWM1 帧率, PWM1DUTY寄存器用于定义PWM1 的占空比。

### 3.4.5 PS1CV (预分频器 1 寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PS1CV	S	0x4	PS1CV[7:0]							
读写属性			读							
初始值			1	1	1	1	1	1	1	1

读取PS1CV时, 将会得到预分频器 1 的目前数值。

### 3.4.6 BZ1CR (蜂鸣器 1 控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BZ1CR	S	0x5	BZ1EN	-	-	-	BZ1FSEL[3:0]			
读写属性			写	-	-	-	写			
初始值			0	X	X	X	1	1	1	1



**BZ1FSEL[3:0]:** BZ1 输出频率选项。

BZ1FSEL[3:0]	BZ1 频率选项	
	时钟源	预分频比
0000	预分频器 1 输出	1:2
0001		1:4
0010		1:8
0011		1:16
0100		1:32
0101		1:64
0110		1:128
0111		1:256
1000	定时器 1 输出	定时器 1 bit 0
1001		定时器 1 bit 1
1010		定时器 1 bit 2
1011		定时器 1 bit 3
1100		定时器 1 bit 4
1101		定时器 1 bit 5
1110		定时器 1 bit 6
1111		定时器 1 bit 7

表 6 蜂鸣器BZ1 (PB2/PB7) 输出频率选项

**BZ1EN:** 开启/关闭蜂鸣器 1 输出。

BZ1EN=1 时, 开启蜂鸣器 1。

BZ1EN=0 时, 关闭蜂鸣器 1。

### 3.4.7 IRCR (IR 控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IRCR	S	0x6	IROSC358M	-	-	-	-	IRCSEL	IRF57K	IREN
读写属性			写	-	-	-	-	写	写	写
初始值			0	X	X	X	X	0	0	0

**IREN:** 开启/关闭IR载波输出。

IREN=1 时, 开启IR载波输出。

IREN=0 时, 关闭IR载波输出。

**IRF57K:** IR载波频率选择。

IRF57K=1 时, IR载波频率是 57KHz。

IRF57K=0 时, IR载波频率是 38KHz。

**IRCSEL:** IR载波极性选择。

IRCSEL=0 且I/O脚数据是 1 时，IR载波会被产生。

IRCSEL=1 且I/O脚数据是 0 时，IR载波会被产生。

**IROSC358M:** 选择使用的外部晶振频率类型。

若选择 I\_HRC此位将被忽略。

IROSC358M=1，外部晶振频率请用 3.58MHz。

IROSC358M=0，外部晶振频率请用 455KHz。

**注意:**

1. 仅有高速震荡时钟  $F_{HOSC}$  (详见章节 3.11) 可以当作IR时钟源。

2. 不同振荡类型的分频比。

OSC. Type	57KHz	38KHz	条件
High IRC	64	96	HIRC 模式 (不论系统时钟频率是多少, IR 模块的输入时钟都设定为 4MHz)
Xtal 3.58MHz	64	96	E_XT 模式 & IROSC358M=1
Xtal 455KHz	8	12	E_XT 模式 & IROSC358M=0

表 7 不同振荡类型的分频比

### 3.4.8 TBHP (表格指针高字节寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TBHP	S	0x7	-	-	-	-	-	TBHP2	TBHP1	TBHP0
读写属性			-	-	-	-	-	读/写	读/写	读/写
初始值			X	X	X	X	X	X	X	X

当指令CALLA、GOTOA或TABLEA被执行时，程序计数器寄存器会指向欲寻址的 10 位ROM地址，此目标地址是由TBHP[1:0]与ACC组成。ACC是PC[9:0]的低字节，TBHP[1:0]是PC[9:0]的高字节。

### 3.4.9 TBHD (表格数据高字节寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TBHD	S	0x8	-	-	TBHD5	TBHD4	TBHD3	TBHD2	TBHD1	TBHD0
读写属性			-	-	读	读	读	读	读	读
初始值			X	X	X	X	X	X	X	X

当指令TABLEA被执行后，会得到ROM表格的 14 位数据内容，其中ROM表格的数据高字节内容被加载到TBHD[5:0]寄存器，ROM表格的数据低字节内容则被加载到ACC。

### 3.4.10 OSCCR (振荡器控制寄存器)

名称	SFR类型	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCR	S	0xF	-	-	-	-	OPMD[1:0]		STPHOSC	SELHOSC
读写属性			-	-	-	-	读/写		读/写	读/写
初始值			X	X	X	X	00		0	1

**SELHOSC:** 系统振荡器选择 ( $F_{OSC}$ )。

SELHOSC=1 时,  $F_{OSC}$ 是高频率振荡器 ( $F_{HOSC}$ )。

SELHOSC=0 时,  $F_{OSC}$ 是低频率振荡器 ( $F_{LOSC}$ )。

**STPHOSC:** 关闭/开启高频率震荡器 ( $F_{HOSC}$ )。

STPHOSC=1 时,  $F_{HOSC}$ 会停止振荡并被关闭。

STPHOSC=0 时,  $F_{HOSC}$ 保持振荡。

**OPMD[1:0]:** 选择操作模式。

OPMD[1:0]	操作模式
00	正常模式
01	睡眠模式
10	待机模式
11	保留

表 8 选择OPMD[1:0]的操作模式

**注意:** STPHOSC不能与SELHOSC或OPMD同时更改。在SELHOSC=1 时, STPHOSC不能与OPMD同时更改。

## 3.5 I/O Port

NY8A053D提供 12 个I/O口 (PA[3:0]和PB[7:0]), 用户可以由寄存器PORTA和PORTB读写这些脚位。每个I/O脚位都有一个对应的寄存器控制位以定义该脚位是输入或输出口, 寄存器IOSTA[3:0]定义PA[3:0]为输入或输出口, 寄存器IOSTB[7:0]定义PB[7:0]为输入或输出口。

当一个I/O脚位被配置为输入口, 它可以由寄存器开启或关闭内部上拉/下拉电阻。寄存器BPHCON[7:0]用于开启或关闭PB[7:0]的内部上拉电阻。寄存器ABPLCON[7:4]则是用于开启或关闭PB[3:0]的内部下拉电阻。ABPLCON[3:0]则是用于开启或关闭PA[3:0]的内部下拉电阻。

当一个I/O脚位配置为输出口, 可由寄存器开启或关闭开漏。寄存器BODCON[7:0]决定PB[7:0]是否为开漏输出脚。

I/O口功能摘要如下表:

功能		PA[3:0]	PB[2:0]	PB[3]	PB[7:4]
输入	上拉电阻	X	V	X	V
	下拉电阻	V	V	V	X
输出	开漏	X	V	总是	V

表 9 I/O端口功能摘要

在PA和PB的每个I/O脚都有输入状态改变产生中断功能。寄存BWUCON[7:0]会使能或禁止任一PB脚位的唤醒功能。只要BWUCON对应到的任一PB脚位被置为1时，且在此输入脚位有状态改变时，寄存器PBIF（INTF[1]）就会被设为1。如果寄存器PBIE（INTE[1]）与GIE（PCON1[7]）同时设定为1，将发生中断要求并执行中断服务程序。

NY8A053D仅提供一个外部中断，当寄存器EIS（PCON[6]）设定为1，PB0则被当作外部中断的输入脚。

**注意：当PB0同时设定成输入状态改变触发脚与外部中断脚，外部中断有较高的优先权，而PB0输入状态改变触发脚则会被关闭，但PB7~PB1输入状态改变触发脚不会被影响。**

NY8A053D提供红外线IR载波生成器。IR载波生成器是由寄存器IREN（IRCR[0]）开启，PB1会输出红外线载波。

由配置字节决定PB3可否当作外部复位输入RSTb。当PB3为低电平时将导致NY8A053D发生复位。

当配置字节设置外置晶振（E\_HXT，E\_XT or E\_LXT）用于高速振荡时钟或低速振荡时钟时，PB5晶振输入引脚（Xin），PB4晶振输出引脚（Xout）。

当配置字节设置I\_HRC 或 I\_LRC用于高速振荡时钟或低速振荡时钟时，用户可以在PB4输出指令时钟F<sub>INST</sub>。

如果寄存器T0CS（T0MD[5]）为1和LCK\_TM0（T0MD[7]）为0，EX\_CK1可以当作定时器0外部时钟源。如果寄存器T1CS（T1CR2[5]）为1，EX\_CK1可以当作定时器1外部时钟源。

如果寄存器PWM1OEN（T1CR1[7]）为1，PB2/PB6也可以当作脉冲宽度调制PWM1输出。若寄存器BZ1EN（BZ1CR[7]）为1，PB2/PB7也可以当作蜂鸣器1输出。PB2复用引脚输出优先级为PWM1输出 > 蜂鸣器1输出。

### 3.5.1 IO 引脚结构框图

IO\_SEL: 设定引脚为输入或输出。

WRITE\_EN: 将数据写入引脚。

READ\_EN: 读取引脚状态。

PULLDOWN\_EN: 开启内部下拉电阻。

RD\_TYPE: 选择读取脚位或数据锁存器。

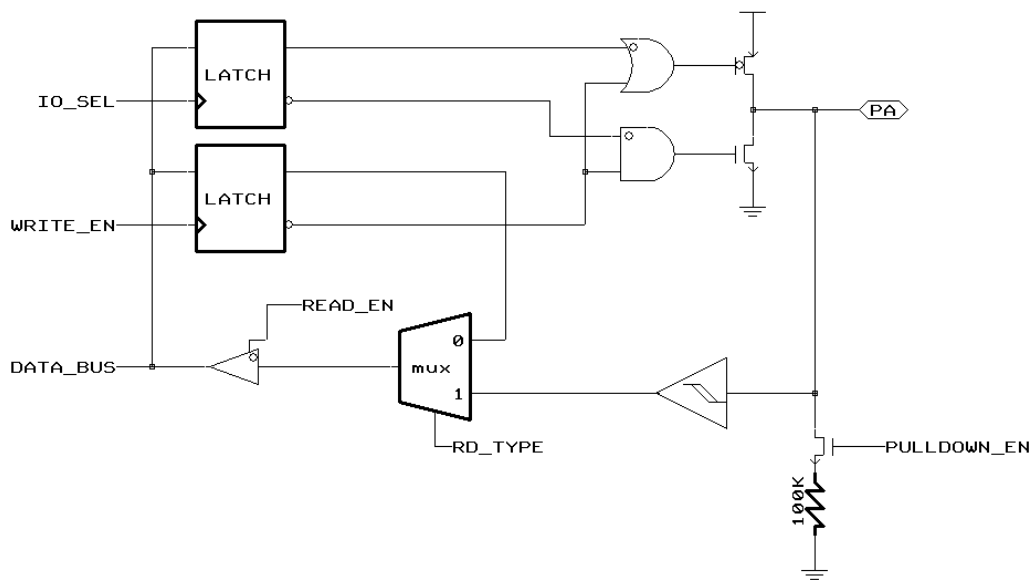


图 5 PA[3:0]引脚结构框图

- IO\_SEL: 设定引脚为输入或输出口。
- WRITE\_EN: 将数据写入引脚。
- READ\_EN: 读取引脚状态。
- OD\_EN: 开启开漏。
- PULLUP\_ENB: 开启内部上拉电阻。
- PULLDOWN\_EN: 开启内部下拉电阻。
- RD\_TYPE: 选择读取脚位或数据锁存器。
- EIS: 使能外部中断功能。
- INTEDG: 选择外部中断触发沿。
- WUB: 使能PB0 唤醒功能。
- SET\_PBIF: PB0 唤醒标志。

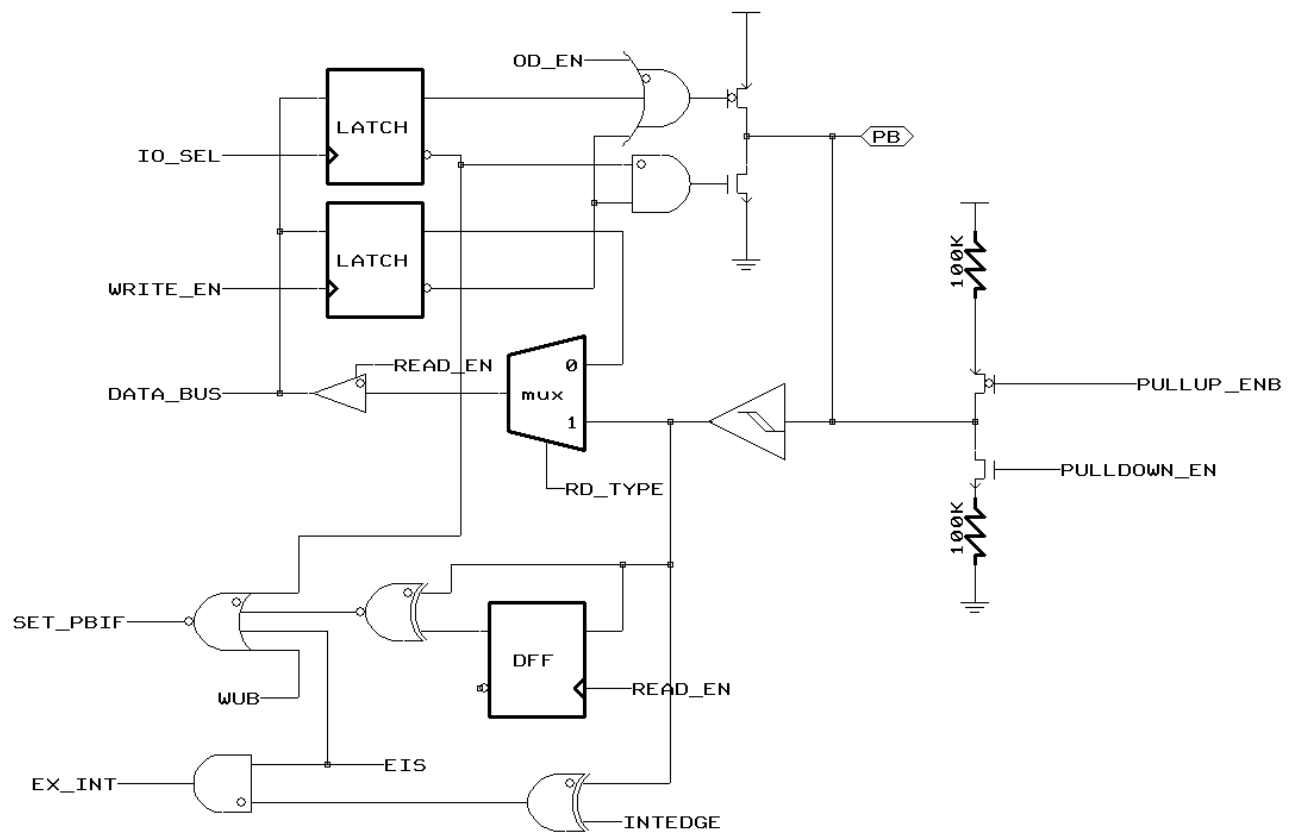


图 6 PB0 引脚结构框图

IO\_SEL: 设定引脚为输入或输出口。

WRITE\_EN: 将数据写入引脚。

READ\_EN: 读取引脚状态。

OD\_EN: 开启开漏。

PULLUP\_ENB: 开启内部上拉电阻。

PULLDOWN\_EN: 开启内部下拉电阻。

RD\_TYPE: 选择读取脚位或数据锁存器。

WUB: 使能PB1 唤醒功能。

SET\_PBIF: PB1 唤醒标志。

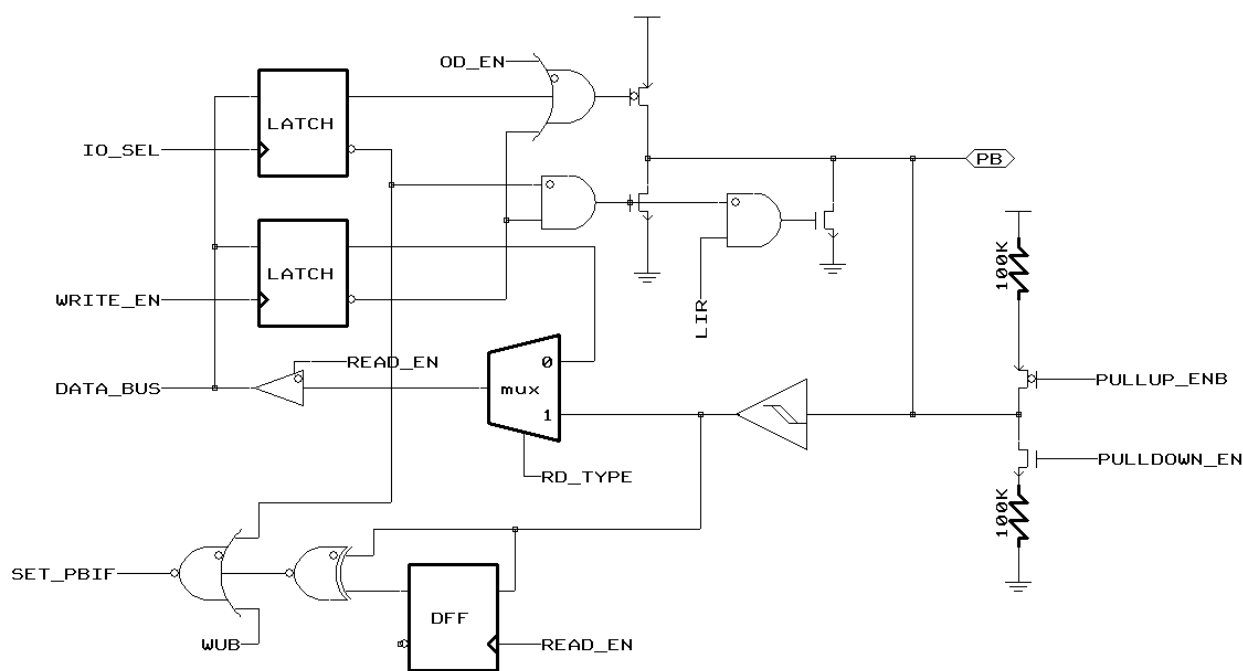


图 7 PB1 引脚结构框图

- IO\_SEL: 设定引脚为输入或输出。
- WRITE\_EN: 将数据写入引脚。
- READ\_EN: 读取引脚状态。
- OD\_EN: 开启开漏。
- PULLUP\_ENB: 开启内部上拉电阻。
- PULLDOWN\_EN: 开启内部下拉电阻。
- RD\_TYPE: 选择读取脚位或数据锁存器。
- WUB: 使能PB2 唤醒功能。
- SET\_PBIF: PB2 唤醒标志。
- EX\_CKI: Timer0,1 外部时钟输入。

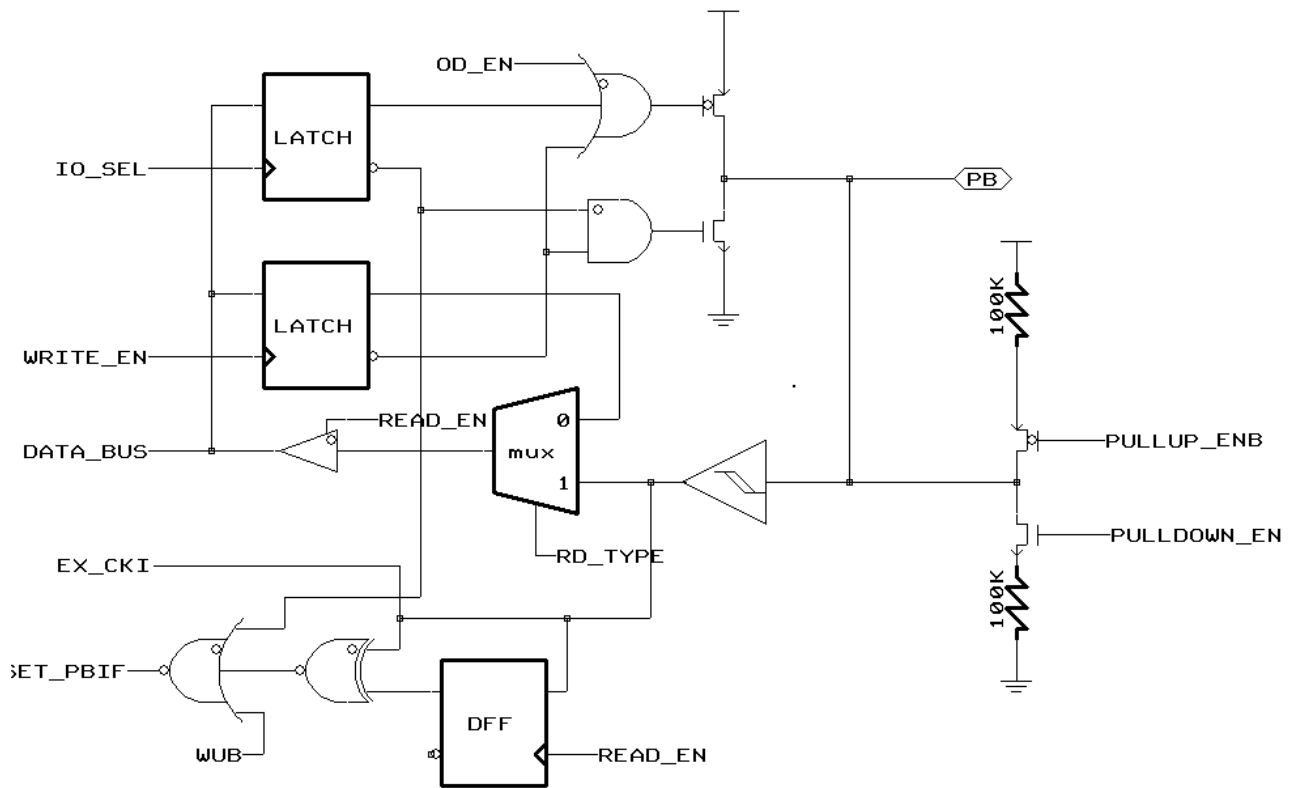


图 8 PB2 引脚结构框图



- IO\_SEL: 设定引脚为输入或输出口。
- WRITE\_EN: 将数据写入引脚。
- READ\_EN: 读取引脚状态。
- PULLDOWN\_EN: 开启内部下拉电阻。
- RD\_TYPE: 选择读取脚位或数据锁存器。
- WUB: 使能PB3 唤醒功能。
- SET\_PBIF: PB3 唤醒标志。
- RSTPAD\_EN: 开启外部复位引脚。
- RSTB\_IN: 复位引脚输入。

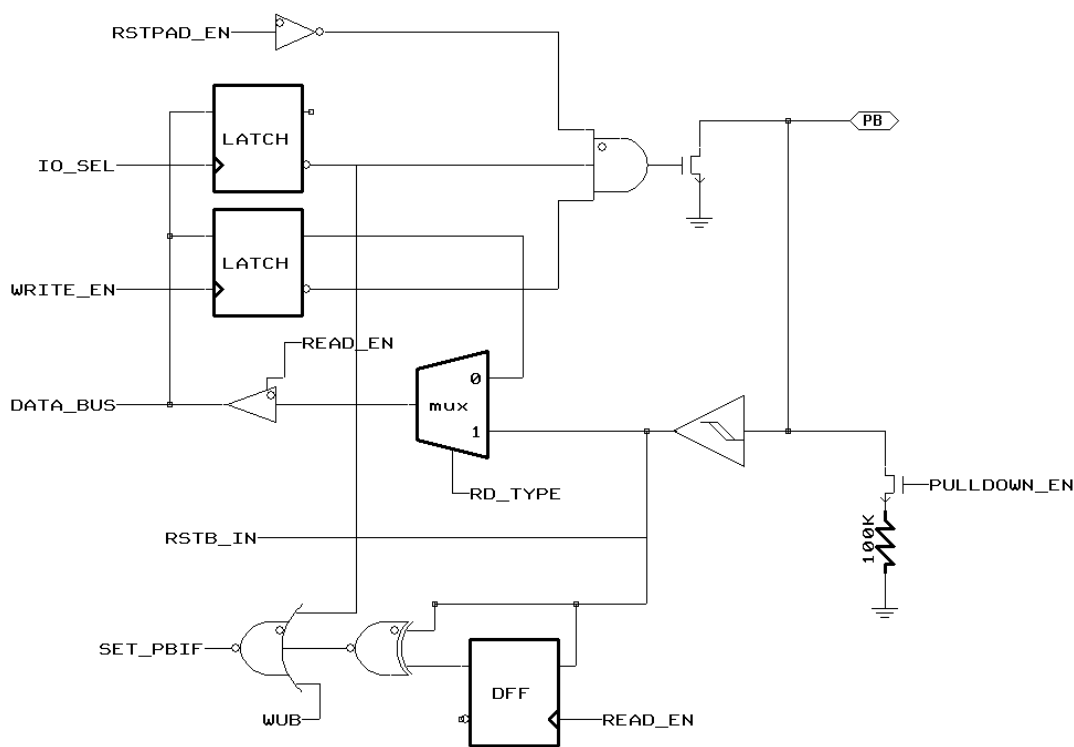


图 9 PB3 引脚结构框图

- IO\_SEL: 设定引脚为输入或输出。
- WRITE\_EN: 将数据写入引脚。
- READ\_EN: 读取引脚状态。
- OD\_EN: 开启开漏。
- PULLUP\_ENB: 开启内部上拉电阻。
- RD\_TYPE: 选择读取脚位或数据锁存器。
- WUB: 使能PB唤醒功能。
- SET\_PBIF: PB唤醒标志。
- XTL\_EN: 开启外置晶振。

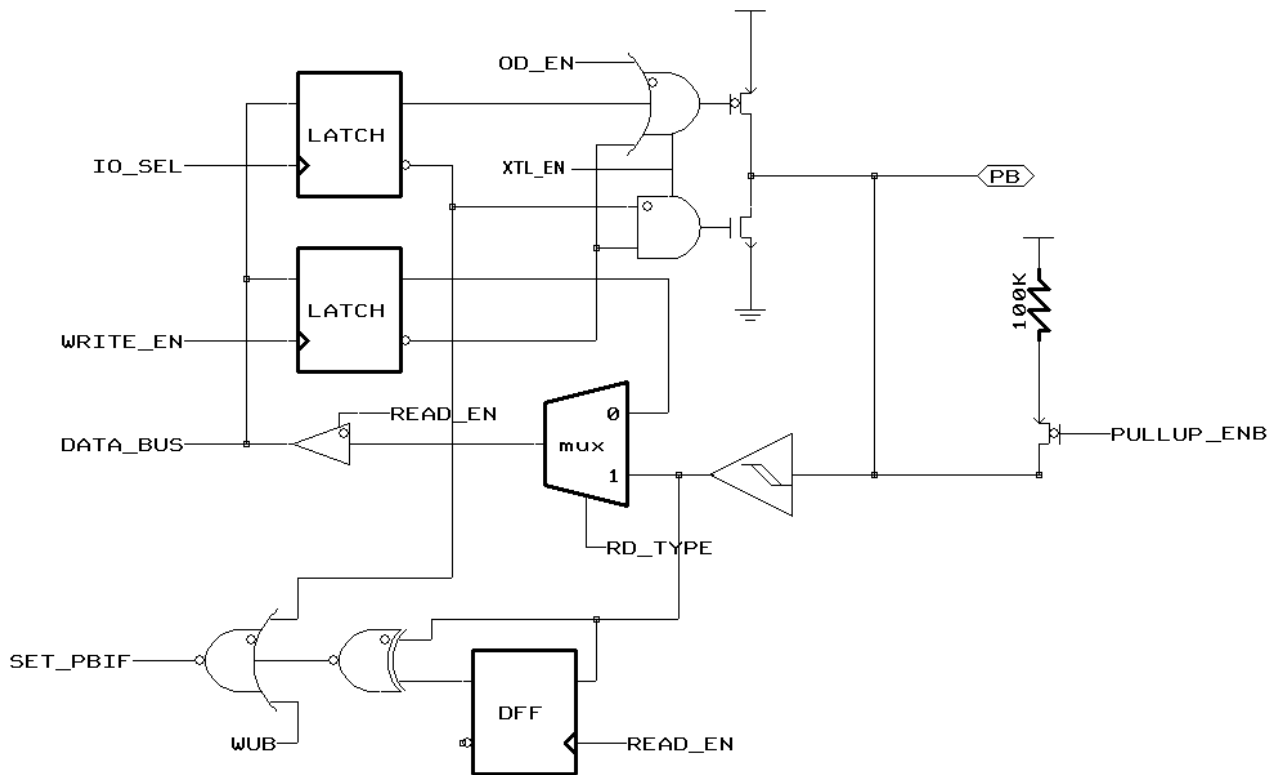


图 10 PB4, PB5 引脚结构框图

IO\_SEL: 设定引脚为输入或输出口。

WRITE\_EN: 将数据写入引脚。

READ\_EN: 读取引脚状态。

OD\_EN: 开启开漏。

PULLUP\_ENB: 开启内部上拉电阻。

RD\_TYPE: 选择读取脚位或数据锁存器。

WUB: 使能PB6/PB7 唤醒功能。

SET\_PBIF: PB6/PB7 唤醒标志。

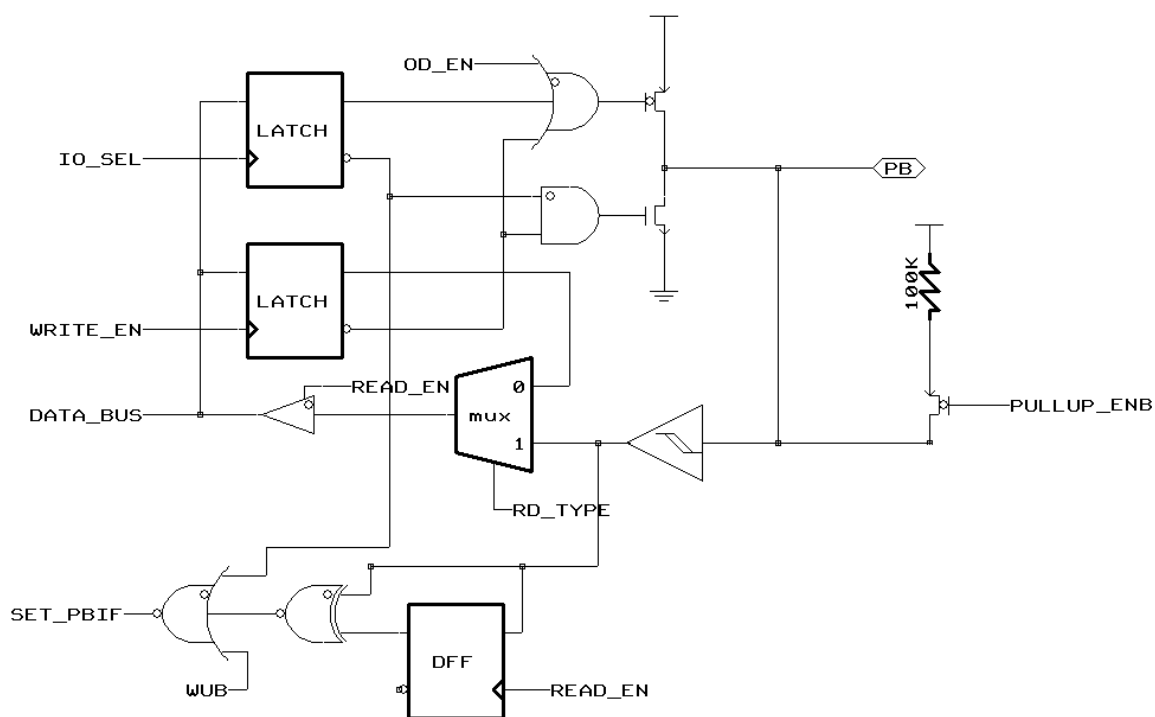


图 11 PB6, PB7 引脚结构框图

### 3.6 定时器 0

定时器 0 是 8 位上数定时器，由寄存器 T0EN (PCON1[0]) 开启/关闭。写入定时器 0 将会设定其初始值，读取定时器 0 时则会显示目前的计数数值。

定时器 0 的时钟源可由寄存器 T0CS (T0MD[5]) 与 LCK\_TM0 (T0MD[7]) 所决定，可以从指令时钟 F<sub>INST</sub>、外部时钟输入脚 EX\_CKI 或低频震荡 I\_LRC/E\_LXT 中择一。当 T0CS 为 0，指令时钟会被选择当作定时器 0 时钟源。当 T0CS 为 1 且 LCK\_TM0 为 0，EX\_CKI 会被当作定时器 0 时钟源。当 T0CS 是 1 且 LCK\_TM0 为 1，会选择低频震荡 I\_LRC/E\_LXT 当作定时器 0 时钟源。汇总成表格如下。（也请参考表 12）

定时器 0 时钟源	T0CS	LCKTM0	定时器 0 来源	低频震荡
Instruction clock	0	X	X	X
EX_CKI	1	0	X	X
		X	0	
E_LXT	1	1	1	1
I_LRC	1	1	1	0

表 10 定时器 0 时钟源摘要

寄存器 T0CE (T0MD[4]) 可决定 EX\_CKI 或 I\_LRC/E\_LXT 的时钟触发沿选择。当 T0CE 是 1，EX\_CKI 或 I\_LRC/E\_LXT 的下降沿将让定时器 0 计数加一。当 T0CE 是 0，EX\_CKI 或 I\_LRC/E\_LXT 的上升沿将让定时器 0 计数加一。

如果寄存器 PS0WDT (T0MD[3]) 为 0，定时器 0 时钟源可以由预分频器 0 所分频，预分频器 0 会被指定到定时器 0，且会在 PS0WDT 设为 0 时清除 Timer0 与寄存器 PS0CV。寄存器 PS0SEL[2:0] (T0MD[2:0]) 决定预分频器 0 的预分频比，其数值从 1:2 到 1:256。

定时器 0 时钟源默认为指令时钟。如果外部时钟脚 EX\_CKI 或低频震荡 I\_LRC/E\_LXT 被用来当作定时器 0 时钟源，用户必须注意分频后的频率不能超过指令时钟，否则会导致错误计数。当 I\_LRC/E\_LXT 同时被当作定时器 0 时钟源与指令时钟，用户必须指定预分频器 0 到定时器 0，且须注意预分频器 0 的预分频比不得小于 4。当配置字节设定为异步 (Async.)，定时器 0 时外部钟源 EX\_CKI 频率就可高于指令时钟。

当定时器 0 上溢，寄存器 T0IF (INTF[0]) 将会设定为 1，以标明定时器 0 发生上溢中断。如果寄存器 T0IE (INTE[0]) 与 GIE 都设定为 1，会发生中断的请求并执行中断服务程序。直到程序写入 0 到 T0IF，T0IF 才会被清除。

定时器 0 与 WDT 的结构框图如下图：

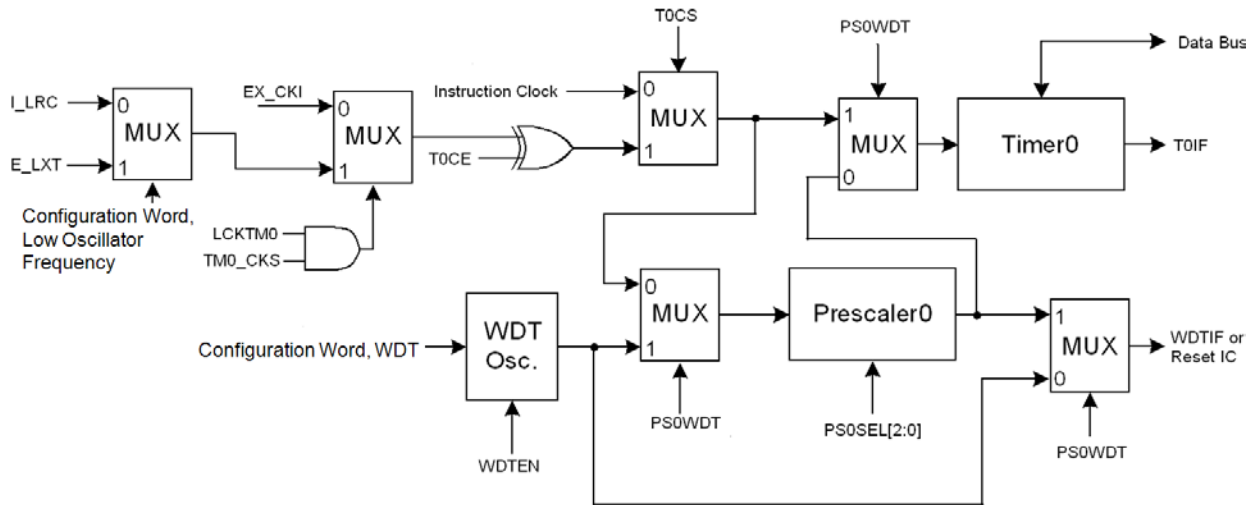


图 12 定时器 0 与 WDT 结构框图

### 3.7 定时器 1/PWM1/Buzzer1

定时器 1 是具有预分频器 1 的 8 位下数定时器，其预分频比是可编程的。定时器 1 的输出可以被用于产生 PWM1 输出与蜂鸣器 1 输出。写入 TMR1 时也会写入定时器 1 重载寄存器 (T1rd) 与定时器 T1 计数器。读取寄存器 TMR1 会显示定时器 1 目前计数数值的内容。

定时器 1 的结构框图如下图所示：

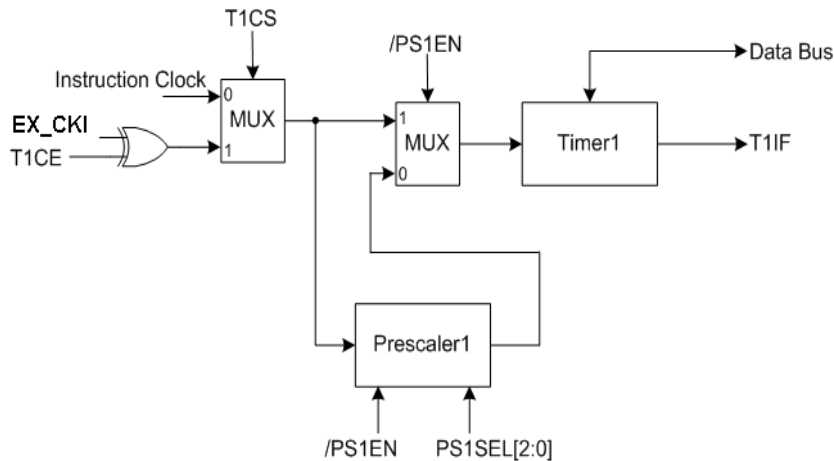


图 13 定时器 1 结构框图

定时器 1 的操作可以由寄存器 T1EN (T1CR1[0]) 开启或关闭。开启定时器 1 后，寄存器 T1CS (T1CR2[5]) 可决定时钟源是指令时钟  $F_{INST}$  或外部时钟 EX\_CKI。当 T1CS 为 0，指令时钟会被选择当做时钟源。当 T1CS 为 1，则是 EX\_CKI 当做时钟源。当 EX\_CKI 被选取，寄存器控制位 T1CE (T1CR2[4]) 可决定 EX\_CKI 的时钟触发沿。当 T1CE 是 1，EX\_CKI 的下降沿将让定时器 1 计数减一。当 T1CE 是 0，EX\_CKI 的上升沿将让定时器 1 计数减一。定时器 1 时钟源可以由预分频器 1 所分频。寄存器 /PS1EN (T1CR2[3]) 为 0，可开启预分频器 1。寄存器 PS1SEL[2:0] (T1CR2[2:0]) 可以决定其预分频比从 1:2 到 1:256。预分频器 1 的目前数值可以由读取寄存器 PS1CV 取得。

定时器 1 提供两种计数模式：单次计数与连续计数。当寄存器T1OS (T1CR1[2])为 1，即为单次计数模式。定时器 1 从储存在寄存器TMR1 的初始值下数到 0x00,当下溢发生时,定时器 1 停止计数。当寄存器T1OS (T1CR1[2])为 0，即为连续计数模式。当下溢发生，寄存器T1RL (T1CR1[1])会决定计数的初始值。当T1RL为 1，定时器 1 从寄存器TMR1 重新载入数值作为初始值并继续下数。当T1RL为 0，定时器 1 以 0xFF作为初始值并继续下数。

当定时器 1 下溢，寄存器T1IF (INTF[3])会被设定为 1，标明定时器 1 发生下溢中断。如果寄存器T1IE (INTE[3])与GIE同时设定为 1，会发生中断请求且执行中断服务程序。直到程序写入 0 到T1IF，T1IF才会被清除。

定时器 1 时序图如下图所示：

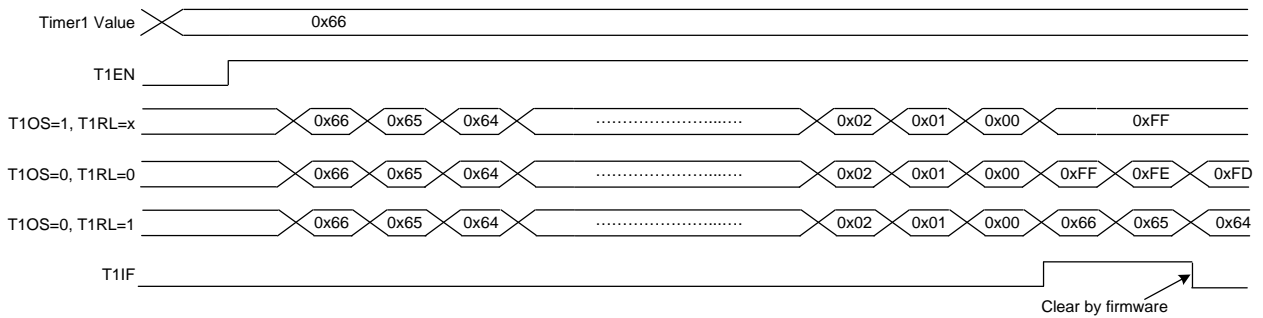


图 14 定时器 1 时序图

当寄存器PWM1OEN (T1CR1[7])设定为 1 且使能配置字节,PB2/PB6 为PWM1 输出。当PWM1OEN为 1,PB2/PB6 会自动成为输出脚。PWM1 输出的有效状态是由寄存器PWM1OAL (T1CR1[6])决定。当PWM1OAL为 1，PWM1 为低电平有效输出；PWM1OAL为 0，PWM1 为高电平有效输出。

PWM1 的占空比与帧率皆可编程的。占空比是由寄存器PWM1DUTY决定。当PWM1DUTY为 0，PWM1 无法输出占空比。当PWM1DUTY为 0xFF，PWM1 将输出 255/256 的占空比（当PWM1OAL为 0）。帧率是由TMR1 初始值所决定。因此，PWM1DUTY数值必须小于或等于TMR1。PWM1 的结构框图如下：

PWM1 的结构框图如下：

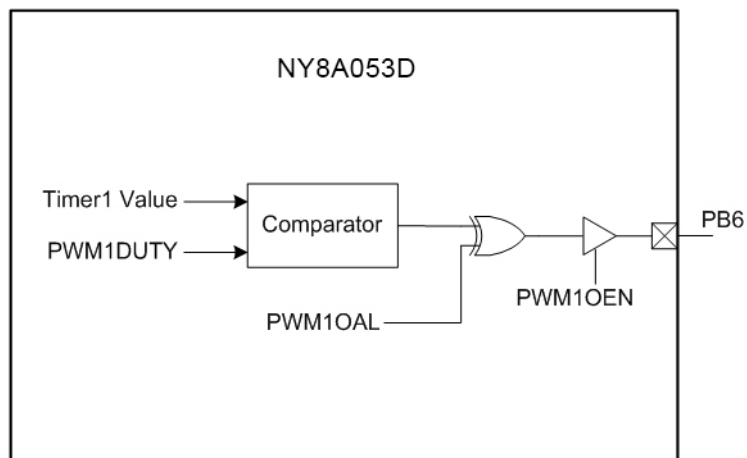


图 15 PWM1 结构框图

当寄存器BZ1EN (BZ1CR1[7])设定成 1 且使能配置字节, PB2/PB7 为蜂鸣器 1 输出。当BZ1EN设定为 1, PB2/PB7 会自动成为输出脚。BZ1 的频率是由寄存器BZ1FSEL[3:0] (BZ1CR[3:0])决定, 可以选择从定时器 1 输出或预分频器 1 输出。当BZ1FSEL[3]为 0, 预分频器 1 输出被选择来产生BZ1 输出。当BZ1FSEL[3]为 1, 定时器 1 输出被选来产生BZ1 输出。预分频比的范围是 1:2 到 1:256。蜂鸣器 1 结构框图如下所示:

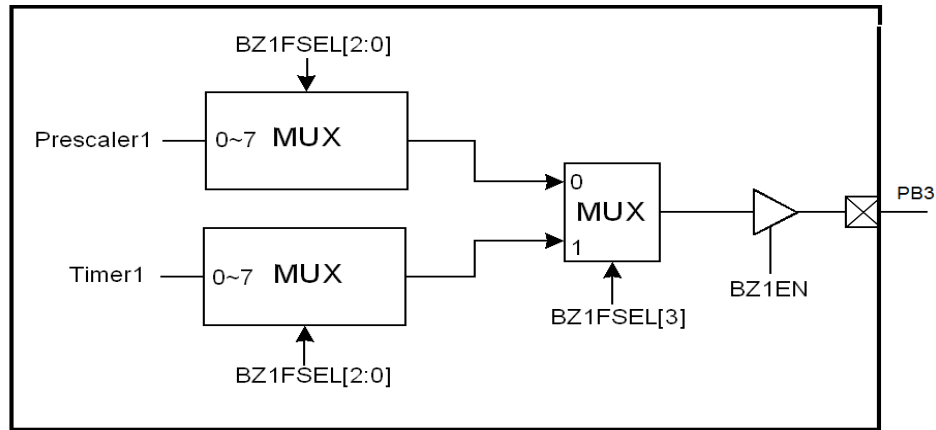


图 16 蜂鸣器 1 结构框图

**注意: PB2 复用引脚输出优先级为 PWM1 输出 > 蜂鸣器 1 输出。**

### 3.8 红外线载波 (IR)

寄存器IREN (IRCR[0])被设定为 1 后, PB1 为红外线载波输出, 而PB1 会自动成为输出脚。当IREN清零, PB1 将会成为一般I/O脚。

红外线载波频率是由寄存器IRF57K (IRCR[1])所选择。当IRF57K为 1, 红外线载波频率是 57KHz。当IRF57K为 0, 频率是 38KHz。红外线载波输出是以高速振荡时钟除频得来。如果使用外置晶振 3.58MHz当作高速振荡时钟来源, 寄存器IROSC358M (IRCR[7])请设为 1。如果使用外置晶振 455KHz当作高速振荡时钟来源, 寄存器IROSC358M (IRCR[7])请设为 0。若选择 I\_HRC当作高速振荡时钟来源, 此时IR模块时钟永远是 4MHz, IROSC358M (IRCR[7])将被忽略。

红外线载波的极性会根据PB1 输出数据所决定。当寄存器IRCSEL (IRCR[2])为 1 且PB1 输出数据为 0, 红外线载波将由PB1 输出。当寄存器IRCSEL (IRCR[2])为 0 且PB1 输出数据为 1, 红外线载波将由PB1 输出。红外线载波的极性如下图所示:

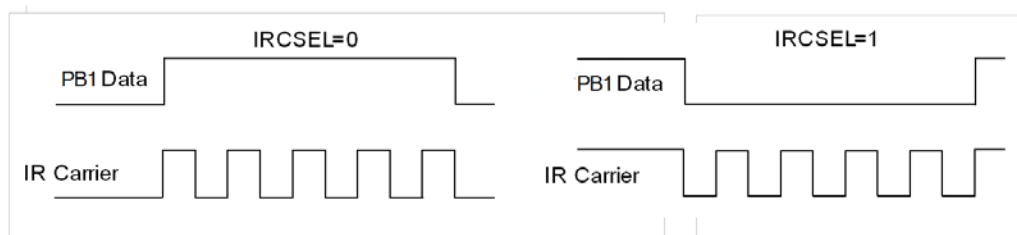


图 17 红外线载波的极性 vs. PB1 数据

### 3.9 看门狗定时器 (WDT)

NY8A053D中有独立振荡器被WDT所使用。由于该振荡器与其它振荡电路无关，故在待机模式和睡眠模式中WDT仍能继续工作。

WDT能被配置字节开启或关闭。当WDT被配置字节开启时，仍然可以通过寄存器WDTEN (PCON[7]) 位来开启/关闭。此外，WDT上溢后可由配置字节决定的复位NY8A053D或发出的中断请求。同时，在WDT上溢后，寄存器/TO (STATUS[4]) 位将被清除为 0。

WDT上溢的时基由配置字节决定，可以是 3.5 毫秒、15 毫秒、60 毫秒或 250 毫秒。如果将预分频器 0 分配给WDT，则可以延长上溢周期。通过将 1 写入寄存器PS0WDT位，预分频器 0 将分配给WDT。预分频器 0 对WDT的分频比由寄存器PS0SEL [2:0]位决定。如果WDT上溢将复位NY8A053D，分频速率从 1:1 到 1:128。如果选为WDT中断时，则分频速率从 1:2 到 1:256。

当预分频器 0 分配给WDT时，执行CLRWDW指令将清除WDT、预分频器 0。并设置/ TO标志位为 1。

如果用户选择WDT中断机制，在WDT上溢后，寄存器WDTIF (INTF[6]) 位将设置为 1。如果寄存器WDTIE (INTE[6]) 位和GIE位都设置为 1，则可能产生中断请求。直到程序将 0 写入WDTIF，WDTIF才会被清除为 0。

### 3.10 中断

NY8A053D提供二种中断：一种是软件中断，另一种是硬件中断。软件中断由执行指令INT来产生。硬件中断则有以下九种：

- Timer0 上溢中断。
- Timer1 下溢中断。
- WDT中断。
- PB输入状态改变中断。
- 外部中断输入。

GIE是总中断使能位，必须为 1 才能使能硬件中断功能。GIE可以通过ENI指令设置 1，通过DISI指令清除为 0。

执行完指令INT后，无论GIE是置 1 还是清除为零，下一条指令都将从地址 0x001 读取。同时，GIE将由NY8A053D自动清除为零，这将防止嵌套中断的发生。软件中断的中断服务程序最后一条指令必须是RETIE。执行此指令将设置GIE为 1 并返回中断前程序执行序列。

当发生硬件中断时，相应的中断标志位将被设置为 1。该位在程序将 0 写入该位之前不会清除为零。因此，用户可以通过轮呼相应的中断标志位得知哪个硬件引发中断。需注意只有当相应的中断使能位设置为 1 时，才能正确地读取相应的中断标志。如果相应的中断使能位设置为 1，GIE也为 1，将发生硬件中断，下一条指令将从 0x008 执行。同时，NY8A053D将自动清除寄存器GIE位为零。如果用户想要实现嵌套中断，可以使用ENI指令作为中断服务程序的第一条指令，将GIE设置为 1，并允许其他中断事件再次中断NY8A053D。指令RETIE必须是中断服务程序的最后一条指令，它将GIE设置为 1 并返回中断前程序执行序列。

用户应注意ENI指令不能放在RETIE指令之前，因为中断服务程序中的ENI指令将开启嵌套中断，但RETIE指令可能会误清除中断标志。



### 3.10.1 Timer0 上溢中断

Timer0 上溢（从 0x00 到 0xFF），如果 T0IE 和 GIE 设置为 1，寄存器 T0IF 位将被硬件设为 1 并处理此中断请求。

### 3.10.2 Timer1 下溢中断

Timer1 下溢（从 0xFF 到 0x00），如果 T1IE 和 GIE 设置为 1，寄存器 T1IF 位将被硬件设为 1 并处理此中断请求。

### 3.10.3 看门狗超时中断

当 WDT 上溢且配置字节选择 WDT 超时中断时，如果 WDTIE 和 GIE 设置为 1，寄存器 WDTIF 位将被硬件设为 1 并将处理此中断请求。

### 3.10.4 PB 输入状态改变中断

当 PB $x$  ( $0 \leq x \leq 7$ ) 设置为输入口且相应的寄存器 WUPB $x$  位设置为 1，且如果 PBIE 和 GIE 设置为 1，当这些选定输入口上的状态变化时，寄存器 PBIF 位将被硬件设为 1 并处理此中断请求。需注意当 PB0 同时设置为状态变化中断和外部中断时，设置 EIS=1 将禁止 PB0 状态变化中断。

### 3.10.5 外部中断输入

根据 EIS=1 和寄存器 INTEDG 的配置，如果 INTIE 和 GIE 设置为 1，PB0 引脚上的有效边沿触发会让寄存器 INTIF 位将被硬件设为 1 并处理此中断请求。

## 3.11 振荡器配置

因为 NY8A053D 是双时钟 IC，有高振荡时钟 ( $F_{HOSC}$ ) 和低振荡时钟 ( $F_{LOSC}$ ) 可选择作为系统振荡时钟 ( $F_{OSC}$ )。可用作  $F_{HOSC}$  的振荡器有内部高速 RC 振荡器 (I\_HRC)、外部高速晶体振荡器 (E\_HXT) 与外部晶体振荡器 (E\_XT)。可用作  $F_{LOSC}$  的振荡器是内部低速 RC 振荡器 (I\_LRC) 与外部低速晶体振荡器 (E\_LXT)。

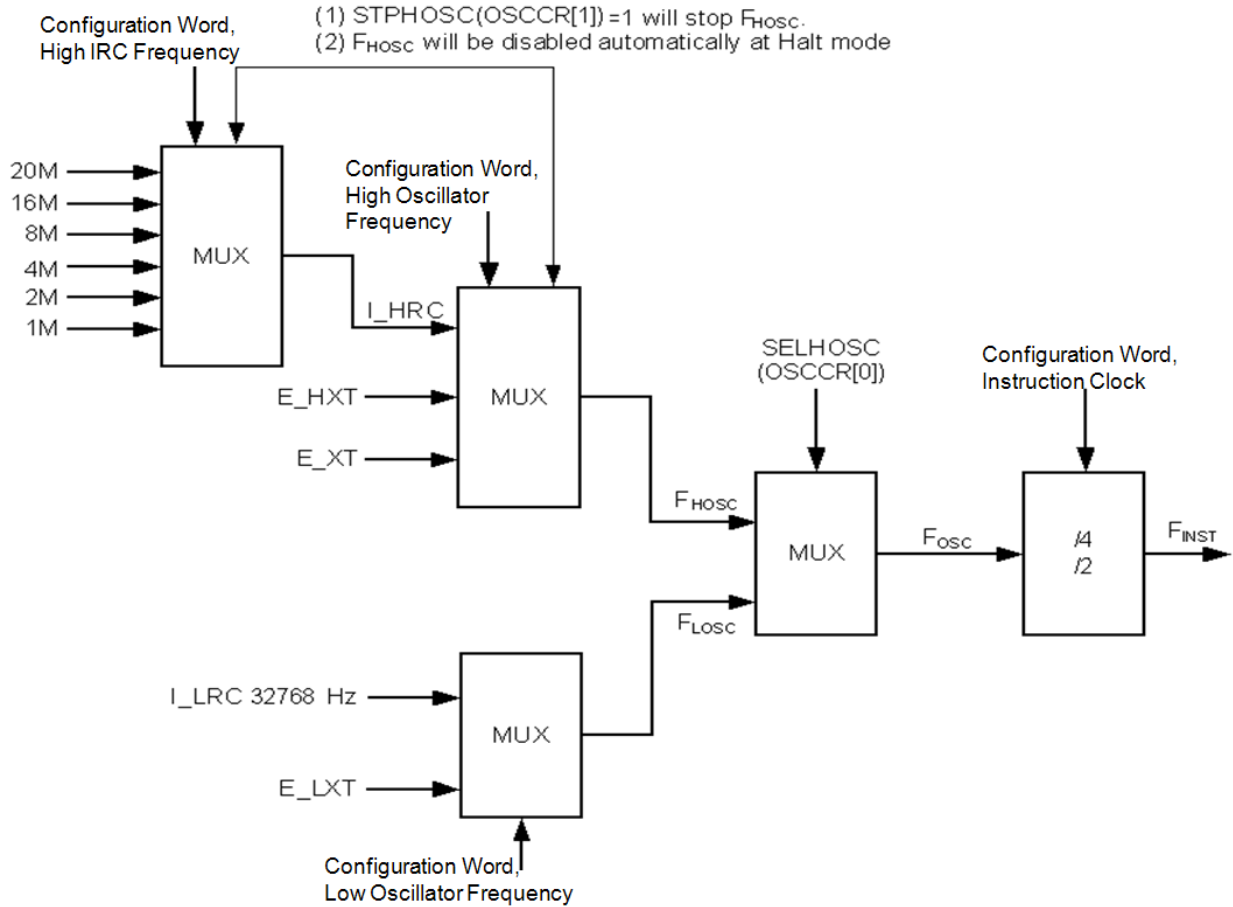


图 18 NY8A053D 振荡配置结构图

系统时钟频率可由配置字节设置为I\_HRC，可以选择的频率为 1M、2M、4M、8M、16M或 20MHz。此时PA7 引脚也可以输出指令时钟。外部晶体振荡器连接引脚为PB5 (Xin) 与PB4 (Xout)。如果外部晶体振荡器是 8MHz 到 20MHz，配置字节请选择E\_HXT。如果外部晶体振荡器是 455KHz到 6MHz，配置字节请选择E\_XT。当选择 E\_HXT或E\_XT时，PB4/PB5 不能当作一般I/O口。它们必须当作晶振的输出 (PB4) 引脚与输入 (PB5) 引脚。

当配置字节选择I\_LRC时，其频率约为 32768Hz。如果需要使用低速晶体振荡器，只能是 32768Hz晶体振荡器，配置字节请选择E\_LXT。当选择E\_LXT，PB4/PB5 不能当作一般I/O口。它们必须当作晶振的输出 (PB4) 引脚与输入 (PB5) 引脚。

No.	F <sub>HOSC</sub>	F <sub>LOSC</sub>
1	I_HRC	I_LRC
2	E_HXT 或 E_XT	I_LRC
3	I_HRC	E_LXT

表 11 双时钟组合

当使用外部晶体振荡器时，建议在Xin与Xout引脚各自接一颗电容器C1 和C2 到VSS，C1 和C2 数值请参考下表。

振荡模式	振荡频率 (Hz)	C1, C2 (pF)
E_HXT	16M	5 ~ 10
	10M	5 ~ 30
	8M	5 ~ 20
E_XT	4M	5 ~ 30
	1M	5 ~ 30
	455K	10 ~ 100
E_LXT	32768	5 ~ 30

表 12 不同外部晶体振荡器频率所推荐的C1 和C2 电容器数值

为得到精准且稳定的 32768Hz 频率，选择正确的 C1 和 C2 电容器数值是相当重要的。每家晶振厂商数据手册中都有记载低速晶体振荡器的负载电容值 (CL)，外接 C1 和 C2 电容器数值的计算如下公式：

$$C1=C2=2*CL-Cbt$$

其中 Cbt 是 NY8A053D 内部电容值，大约为 5pF。假设低速晶体振荡器的负载电容值 CL=12.5pF，依公式算出的 C1=C2=20pF。

I\_HRC 在 25°C 条件下的准确度为 ±1%。

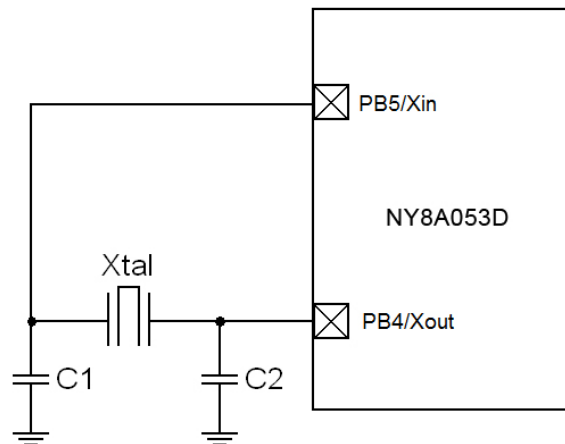


图 19 外部晶体振荡器的硬件连接图

根据寄存器 SELHOSC (OSCCR [0]) 位的值，可以选择 F<sub>HOSC</sub> 或 F<sub>LOSC</sub> 作为系统振荡时钟 F<sub>OSC</sub>。当 SELHOSC 为 1 时，选择 F<sub>HOSC</sub> 作为 F<sub>OSC</sub>。当 SELHOSC 为 0 时，选择 F<sub>LOSC</sub> 作为 F<sub>OSC</sub>。一旦确定 F<sub>OSC</sub>，根据配置字节设置，指令时钟可以选择为 F<sub>OSC</sub>/2 或 F<sub>OSC</sub>/4。

### 3.12 工作模式

NY8A053D 提供了四种操作方式来定制各种应用和节省电力消耗，分别是正常模式、慢速模式、待机模式和睡眠模式。正常模式被指定为高速运行模式，慢速模式被指定为低速模式，以节省功耗。在待机模式下，NY8A053D 将停止几乎所有的运作，可由定时器、看门狗与外部事件等来唤醒。在睡眠模式下，NY8A053D 将睡眠直到外部事件或看门狗定时器来唤醒。

四种工作模式如下图所示。

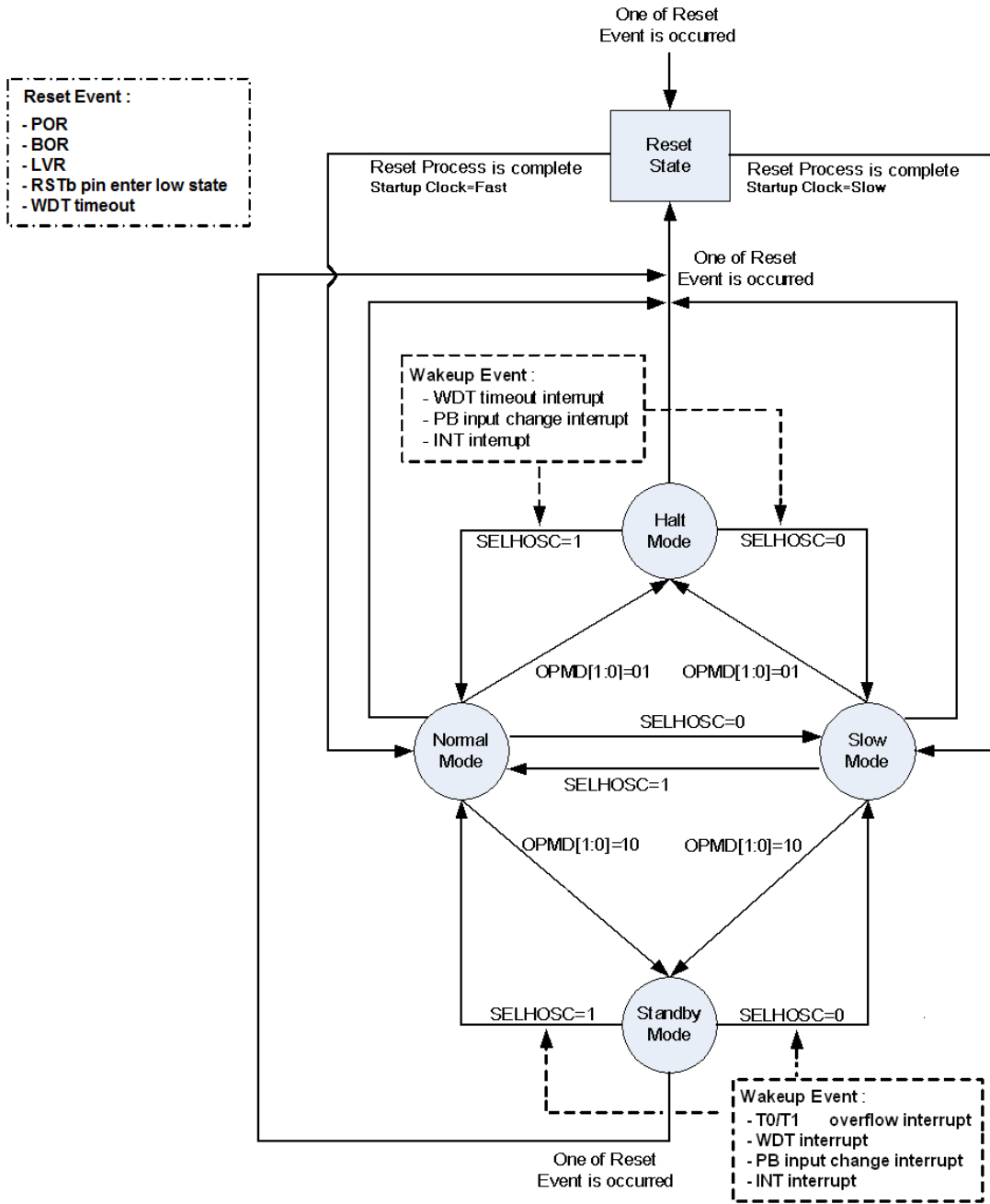


图 20 四种工作模式

### 3.12.1 正常模式

发生任何复位事件并且复位过程完成后，NY8A053D将在正常模式或慢速模式下开始执行程序。重置过程后选择的模式由启动时钟配置字节决定。如果启动时钟为 $F_{HOSC}$ ，NY8A053D将进入正常模式，如果启动时钟为 $F_{LOSC}$ ，NY8A053D将进入慢速模式。在正常模式下，为提供最高性能而以 $F_{HOSC}$ 作为系统振荡时钟，其功耗在四种操作模式中将是最大的。在上电或任何重置触发器被释放后，待复位程序完成NY8A053D将进入正常模式。

- 指令的执行是基于 $F_{HOSC}$ 且所有硬件功能可以根据相应的硬件使能位来开启/关闭。
- $F_{LOSC}$ 仍运行。

- IC可由写 0 至寄存器SELHOSC (OSCCR[0]) 位切换为慢速模式。
- IC可通过寄存器OPMD[1:0] (OSCCR[3:2]) 位切换为待机或睡眠模式。
- 关于实时时钟的应用, NY8A053D在运行正常模式时可同时将低频振荡时钟设为Timer0 的时钟源, 这是通过设置LCKTM0 为 1 和配置字节中Timer0 时钟源来实现。

### 3.12.2 慢速模式

通过写 0 至寄存器SELHOSC位, NY8A053D将进入慢速模式。在低速模式下, 为节省功耗,  $F_{LOSC}$  被选为系统振荡时钟。然而,  $F_{HOSC}$  将不会自动被NY8A053D关闭。因此在慢速模式下, 用户可写 0 至寄存器STPHOSC (OSCCR[1]) 位来停止 $F_{HOSC}$ 进一步降低功耗。但需要注意的是, 禁止进入慢速模式同时停止 $F_{HOSC}$ , 必须先进入慢速模式, 然后关闭 $F_{HOSC}$ 。

- 指令执行是基于 $F_{LOSC}$ 且所有硬件功能可以根据相应的硬件使能位来开启/关闭。
- 通过写 1 至寄存器STPHOSC位,  $F_{HOSC}$ 可以被停止。
- IC可通过寄存器OPMD[1:0]位切换为待机模式或睡眠模式。
- IC可通过写 1 至寄存器SELHOSC切换到正常模式。

### 3.12.3 待机模式

通过写入 10b至寄存器OPMD[1:0], NY8A053D将进入待机模式。然而, 在待机模式下,  $F_{HOSC}$ 不会自动被NY8A053D关闭, 用户必须进入先低速模式後写入 1 至寄存器STPHOSC位, 以停止 $F_{HOSC}$ 。部分NY8A053D的硬件功能会被关闭, 如T0EN/T1EN位被设置为 1 则定时器仍可运作。因此Timer0 / Timer1 溢出后NY8A053D会被唤醒。

- 停止执行指令且一些硬件功能可以根据相应的硬件使能位来开启/关闭。
- 由写入 1 至寄存器STPHOSC位 $F_{HOSC}$ 可以被关闭。
- $F_{LOSC}$ 仍保持运作。
- 如遇以下任一状况IC便能从待机模式唤醒:
  - (a) Timer0 上溢中断 / Timer1 下溢中断 (b) 看门狗超时中断 (c) PB输入状态改变中断 (d) 外部中断。
- 在从待机模式唤醒后, 如SELHOSC=1, IC将回到正常模式, 如SELHOSC=0 则IC将回到慢速模式。  
不建议在同一时间进入待机模式並改变振荡模式 (正常到慢速/慢速到正常)。

### 3.12.4 睡眠模式

通过执行SLEEP指令或写入 01b至寄存器OPMD[1:0]位, NY8A053D将进入睡眠模式。在进入睡眠模式后, 寄存器/PD (STATUS[3]) 位将清除为 0, 寄存器/TO (STATUS[4]) 位将设置为 1 且清除WDT并保持运作。

在睡眠模式下, 所有硬件功能是被关闭的, 停止指令执行且NY8A053D只能通过一些特殊事件唤醒。因此, 睡眠模式是NY8A053D最省电的模式。

- 指令执行停止, 所有硬件功能关闭。

- $F_{HOSC}$ 和 $F_{LOSC}$ 两者都自动关闭。
- 如遇以下任一状况IC便能从睡眠模式中唤醒：
  - (a) 看门狗超时中断 (b) PB输入状态改变中断 (c) INT外部中断。
- 从睡眠模式唤醒后，如 $SELHOSC=1$ ，IC将回到正常模式，如 $SELHOSC=0$ 则IC将回到慢速模式。  
**注意：您可以在同一指令中更改STPHOSC并进入睡眠模式。**
- 不建议改变振荡模式（正常到慢速/慢速到正常），并在同一时间进入待机模式。

### 3.12.5 唤醒稳定时间

若外部晶体振荡器为系统振荡时钟来源，其睡眠模式的唤醒等待时间为  $512 \cdot F_{osc}$ ，若内部RC振荡器为系统振荡时钟来源，其睡眠模式的唤醒等待时间为  $16 \cdot F_{osc}$ ，由于待机模式下 $F_{HOSC}$ 或 $F_{LOSC}$ 仍在运行，因此无需为待机模式唤醒稳定时间。

在NY8A053D进入待机模式或睡眠模式之前，用户可以执行指令ENI。在唤醒后，NY8A053D将跳转到地址0x008，以便执行中断服务程序。如果在进入待机模式或睡眠模式之前执行DISI指令，则在唤醒后执行下一条指令。

### 3.12.6 工作模式概述

四种工作模式概述如下：

模式	正常模式	慢速模式	待机模式	睡眠模式
$F_{HOSC}$	使能	STPHOSC	STPHOSC	关闭
$F_{LOSC}$	使能	使能	使能	关闭
指令执行	执行	执行	停止	停止
计时器 0/1	TxEN	TxEN	TxEN	关闭
WDT	配置和WDTEN	配置和WDTEN	配置和WDTEN	配置和WDTEN
其它硬件	硬件使能位	硬件使能位	硬件使能位	全部关闭
唤醒源	-	-	- Timer0 上溢 - Timer1 下溢 - WDT超时 - PB输入状态改变 - 外部中断	- WDT超时 - PB输入状态改变 - 外部中断

表 13 工作模式概述

### 3.13 复位

当以下任一复位事件发生时，NY8A053D将会进入复位状态并开始复位动作：

- 当VDD检测到上升沿时为上电复位。
- 当VDD电压低于预设的LVR电压时，为LVR复位。
- RSTb引脚为低电平。
- WDT超时复位。

此外，所有寄存器如果初始值未知时，寄存器将会被初始化为初始值或保持不变。状态位/*TO*和/*PD*可以根据复位事件来初始化。*/TO*和/*PD*的值及其相关的事件概述如下。

事件	/ <i>TO</i>	/ <i>PD</i>
POR, LVR	1	1
非睡眠模式时发生RSTb复位	不变	不变
睡眠模式时发生RSTb复位	1	1
非睡眠模式时发生WDT复位	0	1
睡眠模式时发生WDT复位	0	0
执行SLEEP指令	1	0
执行CLRWDT指令	1	1

表 14 */TO*和/*PD*值和相关事件概述

复位事件发生后, NY8A053D将会开始复位进程。无论采用什么样的振荡器，它将等待一定的周期使振荡稳定。这个周期被称为上电复位时间，它由三位配置字节决定，这个时间可能是 140us，4.5ms，18ms，72ms或 288 ms。振荡器稳定后，NY8A053D将等待  $F_{osc}$  的 16 个时钟周期（OST，振荡器启动时间）后完成复位。若上电复位时间设为 140us时，将等待  $F_{osc}$  的 1 个时钟周期后完成复位。

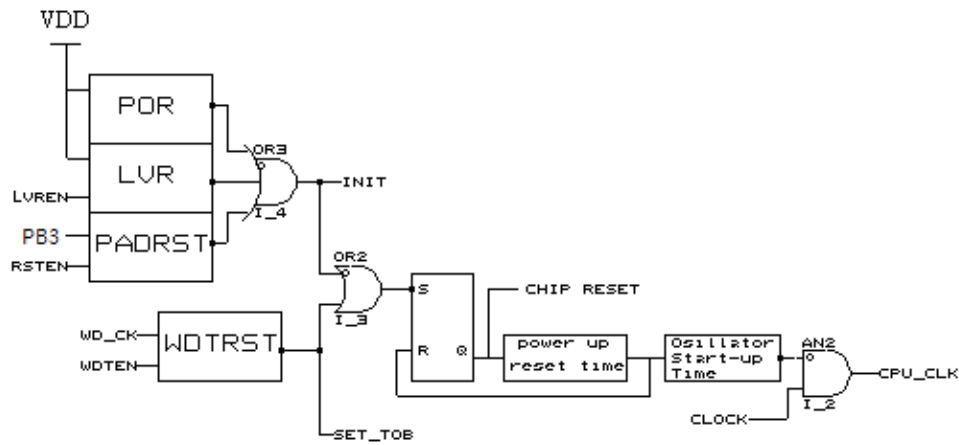


图 21 芯片复位电路框图

如果VDD缓慢上升，建议使用RSTb复位功能，如下图。

- 建议R阻值不大于 40KΩ。
- R1 值= 100Ω ~ 1KΩ时，将阻止过大电流，ESD或电气过载信号进入复位引脚。
- 二极管D使电容C能在VDD下电时快速放电。

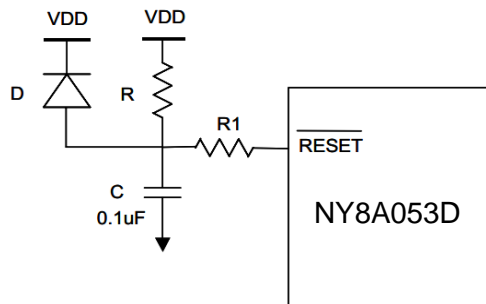


图 22 外部上电复位硬件连接图

#### 4. 指令设置

NY8A053D为各种应用程序提供了 55 个强大的指令。

指令	助记符		说明	周期数	影响标志
	1	2			
<b>算术指令</b>					
ANDAR	R	d	dest = ACC & R	1	Z
IORAR	R	d	dest = ACC   R	1	Z
XORAR	R	d	dest = ACC $\oplus$ R	1	Z
ANDIA	i		ACC = ACC & i	1	Z
IORIA	i		ACC = ACC   i	1	Z
XORIA	i		ACC = ACC $\oplus$ i	1	Z
RRR	R	d	Rotate right R	1	C
RLR	R	d	Rotate left R	1	C
BSR	R	bit	Set bit in R	1	-
BCR	R	bit	Clear bit in R	1	-
INCR	R	d	Increase R	1	Z
DECR	R	d	Decrease R	1	Z
COMR	R	d	dest = ~R	1	Z
<b>条件指令</b>					
BTRSC	R	bit	Test bit in R, skip if clear	1 or 2	-
BTRSS	R	bit	Test bit in R, skip if set	1 or 2	-
INCRSZ	R	d	Increase R, skip if 0	1 or 2	-
DECRSZ	R	d	Decrease R, skip if 0	1 or 2	-
<b>数据传送指令</b>					
MOVAR	R		Move ACC to R	1	-
MOV R	R	d	Move R	1	Z
MOVIA	i		Move immediate to ACC	1	-
SWAPR	R	d	Swap halves R	1	-
IOST	F		Load ACC to F-page SFR	1	-
IOSTR	F		Move F-page SFR to ACC	1	-
SFUN	S		Load ACC to S-page SFR	1	-
SFUNR	S		Move S-page SFR to ACC	1	-
T0MD			Load ACC to T0MD	1	-
T0MDR			Move T0MD to ACC	1	-
TABLEA			Read ROM	2	-

指令	助记符		说明	周期数	影响标志
	1	2			
<b>算术指令</b>					
ADDAR	R	d	dest = R + ACC	1	Z, DC,
SUBAR	R	d	dest = R + (~ACC)	1	Z, DC,
ADCAR	R	d	dest = R + ACC + C	1	Z, DC,
SBCAR	R	d	dest = R + (~ACC) + C	1	Z, DC,
ADDIA	i		ACC = I + ACC	1	Z, DC,
SUBIA	i		ACC = i + (~ACC)	1	Z, DC,
AD CIA	i		ACC = i + ACC + C	1	Z, DC,
SBCIA	i		ACC = i + (~ACC) + C	1	Z, DC,
DAA			Decimal adjust for ACC	1	C
CMPAR	R		Compare R with ACC	1	Z, C
CLRA			Clear ACC	1	Z
CLRR			Clear R	1	Z
<b>其它指令</b>					
NOP			No operation	1	-
SLEEP			Go into Halt mode	1	/TO, /PD
CLRWDT			Clear Watch-Dog Timer	1	/TO, /PD
ENI			Enable interrupt	1	-
DISI			Disable interrupt	1	-
INT			Software Interrupt	3	-
RET			Return from subroutine	2	-
RETIE			Return from interrupt and enable interrupt	2	-
RETIA	i		Return, place immediate in	2	-
CALLA			Call subroutine by ACC	2	-
GOTOA			unconditional branch by	2	-
CALL	adr		Call subroutine	2	-
GOTO	adr		unconditional branch	2	-
LCALL	adr		Call subroutine	2	-
LGOTO	adr		unconditional branch	2	-

表 15 指令设置



ACC: 累加器。

adr: 地址。

bit: R-page中 8 位寄存器的位地址。

C: 进位/借位。

C=1, 加法指令有进位, 减法指令无借位。

C=0, 加法指令无进位, 减法指令有借位。

d: 目标。

若d="0", 结果存入ACC。

若d="1", 结果存入R寄存器。

DC: 半字节进位/借位标记。

dest: 目标。

F: F 页面特殊功能寄存器, F 值为 0x5~0xF。

i: 8 位立即数。

PC: 程序计数器。

PCHBUF: 程序计数器的高字节。

/PD: 睡眠标志位。

/PD=1, 上电或CLRWDT指令执行后。

/PD=0, SLEEP指令执行后。

Prescaler: 预分频器。

R: R页面特殊功能寄存器, R值为 0x00~0x3F。

S: S页面特殊功能寄存器, S值为 0x0 ~ 0xF。

T0MD: T0MD寄存器。

TBHP: 表格指针高字节寄存器。

TBHD: 表格数据高字节寄存器。

/TO: 看门狗超时标志位。

/TO=1, 上电或执行 CLRWDT 或 SLEEP 指令后。

/TO=0, 看门狗超时。

WDT: 看门狗计时器。

Z: 清零标志。

<b>ADCAR</b>	<b>Add ACC and R with Carry</b>
语法	ADCAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$R + ACC + C \rightarrow dest$
状态影响	Z, DC, C
说明	ACC和R带进位加法：若d="0"，结果存入ACC；若d="1"，结果存入"R"。
周期	1
举例	ADCAR R, d 执行指令前： ACC=0x12, R=0x34, C=1, d=1。 执行指令后 R=0x47, ACC=0x12, C=0。

<b>ADDAR</b>	<b>Add ACC and R</b>
语法	ADDAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$ACC + R \rightarrow dest$
状态影响	Z, DC, C
说明	ACC和R加法：若d="0"，结果存入ACC；若d="1"，结果存入"R"。
周期	1
举例	ADDAR R, d 执行指令前： ACC=0x12, R=0x34, C=1, d=1 执行指令后： R=0x46, ACC=0x12, C=0。

<b>ADCIA</b>	<b>Add ACC and Immediate with Carry</b>
语法	ADCIA i
操作数	$0 \leq i < 255$
操作	$ACC + i + C \rightarrow ACC$
状态影响	Z, DC, C
说明	ACC和 8 位立即数带进位加法，结果存入ACC。
周期	1
举例	ADCIA i 执行指令前： ACC=0x12, i=0x34, C=1。 执行指令后： ACC=0x47, C=0。

<b>ADDIA</b>	<b>Add ACC and Immediate</b>
语法	ADDIA i
操作数	$0 \leq i < 255$
操作	$ACC + i \rightarrow ACC$
状态影响	Z, DC, C
说明	ACC和 8 位立即数加法，结果存入ACC。
周期	1
举例	ADDIA i 执行指令前： ACC=0x12, i=0x34, C=1。 执行指令后： ACC=0x46, C=0。

<b>ANDAR</b>	<b>AND ACC and R</b>
语法	ANDAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	ACC & R $\rightarrow$ dest
状态影响	Z
说明	ACC和R做“AND”运算；若d=“0”，结果存入ACC；若d=“1”，结果存入“R”。
周期	1
举例	ANDAR R, d 执行指令前： ACC=0x5A, R=0xAF, d=1。 执行指令后： R=0x0A, ACC=0x5A, Z=0。

<b>BCR</b>	<b>Clear Bit in R</b>
语法	BCR R, bit
操作数	$0 \leq R \leq 63$ $0 \leq \text{bit} \leq 7$
操作	$0 \rightarrow R[\text{bit}]$
状态影响	--
说明	将R寄存器的bit位（0~7）清0。
周期	1
举例	BCR R, B2 执行指令前： R=0x5A, B2=0x3。 执行指令后： R=0x52。

<b>ANDIA</b>	<b>AND Immediate with ACC</b>
语法	ANDIA i
操作数	$0 \leq i < 255$
操作	ACC & i $\rightarrow$ ACC
状态影响	Z
说明	ACC和8位立即数做“AND”运算。
周期	1
举例	ANDIA i 执行指令前： ACC=0x5A, i=0xAF。 执行指令后： ACC=0x0A, Z=0。

<b>BSR</b>	<b>Set Bit in R</b>
语法	BSR R, bit
操作数	$0 \leq R \leq 63$ $0 \leq \text{bit} \leq 7$
操作	$1 \rightarrow R[\text{bit}]$
状态影响	--
说明	设置R寄存器的bit位为1。
周期	1
举例	BSR R, B2 执行指令前： R=0x5A, B2=0x2。 执行指令后： R=0x5E。

<b>BTRSC</b>	<b>Test Bit in R and Skip if Clear</b>
语法	BTRSC R, bit
操作数	$0 \leq R \leq 63$ $0 \leq \text{bit} \leq 7$
操作	Skip next instruction, if R[bit] = 0
状态影响	--
说明	位判断指令, 为“0”则跳过下一条指令。
周期	1 or 2 (跳过)
举例	BTRSC R, B2 指令 1 指令 2 执行指令前: R=0x5A, B2=0x2。 执行指令后: 由于R[B2]=0, 则指令 1 不执行, 程序直接从指令 2 开始执行。

<b>CALL</b>	<b>Call Subroutine</b>
语法	CALL adr
操作数	$0 \leq \text{adr} < 255$
操作	PC + 1 → Top of Stack {PCHBUF, adr} → PC
状态影响	--
说明	子程序调用, 首先将返回地址PC+1压入栈顶。然后将 8 位立即地址载入 PC[7:0], 将 PCHBUF[1:0] 载入 PC[9:8]。
周期	2
举例	CALL SUB 执行指令前: PC=A0, Stack pointer=1。 执行指令后: PC=address of SUB, Stack[1] = A0+1, Stack pointer=2。

<b>BTRSS</b>	<b>Test Bit in R and Skip if Set</b>
语法	BTRSS R, bit
操作数	$0 \leq R \leq 63$ $0 \leq \text{bit} \leq 7$
操作	Skip next instruction, if R[bit] = 1
状态影响	--
说明	位判断指令, 为“1”则跳过下一条指令。
周期	1 or 2 (跳过)
举例	BTRSS R, B2 指令 2 指令 3 执行指令前: R=0x5A, B2=0x3。 执行指令后: 由于R[B2]=1, 则指令 2 不执行, 直接从指令 3 开始执行。

<b>CALLA</b>	<b>Call Subroutine</b>
语法	CALLA
操作数	--
操作	PC + 1 → Top of Stack {TBHP, ACC} → PC
状态影响	--
说明	子程序调用。首先将返回地址PC+1压入栈顶, 然后将TBHP[1:0]赋值给 PC[9:8], 将ACC赋值给PC[7:0]。
周期	2
举例	CALLA 执行指令前 TBHP =0x02, ACC =0x34, PC =A0。 Stack pointer=1。 执行指令后: PC=0x234, Stack[1]=A0+1, Stack pointer=2。

<b>CLRA</b>	<b>Clear ACC</b>
语法	CLRA
操作数	--
操作	00h → ACC 1 → Z
状态影响	Z
说明	ACC清零，Z标志位置“1”。
周期	1
举例	CLRA 执行指令前： ACC=0x55, Z=0。 执行指令后： ACC=0x00, Z=1。

<b>CLRWDT</b>	<b>Clear Watch-Dog Timer</b>
语法	CLRWDT
操作数	--
操作	00h → WDT 00h → WDT预分频器（若开启） 1 → /TO 1 → /PD
状态影响	/TO, /PD
说明	清WDT计数器和预分频器；/TO和/PD标志位置“1”。
周期	1
举例	CLRWDT 执行指令前： /TO=0 执行指令后： /TO=1

<b>CLRR</b>	<b>Clear R</b>
语法	CLRR R
操作数	$0 \leq R \leq 63$
操作	00h → R 1 → Z
状态影响	Z
说明	寄存器R清零，Z标志位置“1”。
周期	1
举例	CLRR R 执行指令前： R=0x55, Z=0。 执行指令后： R=0x00, Z=1。

<b>COMR</b>	<b>Complement R</b>
语法	COMR R, d
操作数	$0 \leq R \leq 63$ d = 0, 1
操作	$\sim R \rightarrow \text{dest}$
状态影响	Z
说明	R寄存器取补，结果存入d； d=“0”，结果存入ACC； d=“1”，结果存入R。
周期	1
举例	COMR, d 执行指令前： R=0xA6, d=1, Z=0。 执行指令后： R=0x59, Z=0。

<b>CMPAR</b>	<b>Compare ACC and R</b>
语法	CMPAR R
操作数	$0 \leq R \leq 63$
操作	R - ACC → (No restore)
状态影响	Z, C
说明	ACC和R比较：执行R - ACC，不改变ACC和R的值，只改变Z和C标志位。
周期	1
举例	CMPAR R 执行指令前： R=0x34, ACC=12, Z=0, C=0。 执行指令后： R=0x34, ACC=12, Z=0, C=1。

<b>DECR</b>	<b>Decrease R</b>
语法	DECR R, d
操作数	$0 \leq R \leq 63$ d = 0, 1
操作	R - 1 → dest
状态影响	Z
说明	R - 1, 若d="0", 结果存入ACC； 若d="1", 结果存入R。
周期	1
举例	DECR R, d 执行指令前： R=0x01, d=1, Z=0。 执行指令后： R=0x00, Z=1。

<b>DAA</b>	<b>Convert ACC Data Format from Hexadecimal to Decimal</b>
语法	DAA
操作数	--
操作	ACC(hex) → ACC(dec)
状态影响	C
说明	将累加器中的 16 进制数调整为十进制数，该指令必须紧跟在加法指令后。
周期	1
举例	ADDAR R, d DAA 执行指令前： ACC=0x28, R=0x25, d=0。 执行指令后： ACC=0x53, C=0。

<b>DECRSZ</b>	<b>Decrease R, Skip if 0</b>
语法	DECRSZ R, d
操作数	$0 \leq R \leq 63$ d = 0, 1
操作	R - 1 → dest, Skip if result = 0
状态影响	--
说明	R 先- 1, 若d="0", 结果存入ACC； 若d="1", 结果存入R, 若结果为"0" 则跳过下一条指令，改为执行NOP 指令，因此结果为"0"时要执行两个 周期。
周期	1 or 2 (跳过)
举例	DECRSZ R, d 指令 2 指令 3 执行指令前： R=0x1, d=1, Z=0。 执行指令后： R=0x0, Z=1, 操作结果为 0, 指 令 2 被跳过。

<b>DISI</b>	<b>Disable Interrupt Globally</b>
语法	DISI
操作数	--
操作	Disable Interrupt, 0 → GIE
状态影响	--
说明	GIE设置为 0, 关闭总中断。
周期	1
举例	DISI 执行指令前: GIE=1。 执行指令后: GIE=0。

<b>GOTO</b>	<b>Unconditional Branch</b>
语法	GOTO adr
操作数	$0 \leq \text{adr} < 511$
操作	{PCHBUF, adr} → PC
状态影响	--
说明	无条件短跳转指令, 9位地址写入 PC[8:0], PCHBUF[1]写入PC[9]。
周期	2
举例	GOTO Level 执行指令前: PC=A0。 执行指令后: PC=address of Level。

<b>ENI</b>	<b>Enable Interrupt Globally</b>
语法	ENI
操作数	--
操作	Enable Interrupt, 1 → GIE
状态影响	--
说明	GIE设置为 1, 开启总中断。
周期	1
举例	ENI 执行指令前: GIE=0。 执行指令后: GIE=1。

<b>GOTOA</b>	<b>Unconditional Branch</b>
语法	GOTOA
操作数	--
操作	{TBHP, ACC} → PC
状态影响	--
说明	无条件跳转指令, ACC值写入 PC[7:0]; TBHP[1:0]值写入PC[9:8]。
周期	2
举例	GOTOA 执行指令前: PC=A0, TBHP=0x02, ACC=0x34 执行指令后: PC=0x234。

<b>INCR</b>	<b>Increase R</b>
语法	INCR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$R + 1 \rightarrow \text{dest}$
状态影响	Z
说明	R+ 1, 若d="0", 结果存入ACC; 若d="1", 结果存入R。
周期	1
举例	INCR R, d 执行指令前: R=0xFF, d=1, Z=0。 执行指令后: R=0x00, Z=1。

<b>INT</b>	<b>Software Interrupt</b>
语法	INT
操作数	--
操作	$PC + 1 \rightarrow \text{Top of Stack}$ , $001h \rightarrow PC$
状态影响	--
说明	软中断指令。首先将返回地址 (PC+1) 压入栈顶, 然后将 001H 的地址装入PC[9:0]。
周期	3
举例	INT 执行指令前: PC=address of INT code。 执行指令后: PC=0x01。

<b>INCRSZ</b>	<b>Increase R, Skip if 0</b>
语法	INCRSZ R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$R + 1 \rightarrow \text{dest}$ , Skip if result = 0
状态影响	--
说明	R先+ 1, 若d="0", 结果存入ACC; 若 d="1", 结果存入R。若结果为"0"则跳 过下一条指令(执行NOP指令)。
周期	1 or 2(skip)
举例	INCRSZ R, d 指令 2 指令 3 执行指令前: R=0xFF, d=1, Z=0。 执行指令后: R=0x00, Z=1, 因结果为 0, 程序 跳过指令 2。

<b>IORAR</b>	<b>OR ACC with R</b>
语法	IORAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$ACC   R \rightarrow \text{dest}$
状态影响	Z
说明	ACC和R做"OR"运算, 若d="0", 结 果存入ACC; 若d="1", 结果存入R。
周期	1
举例	IORAR R, d 执行指令前: R=0x50, ACC=0xAA, d=1, Z=0。 执行指令后: R=0xFA, ACC=0xAA, Z=0。



<b>IORIA</b>	<b>OR Immediate with ACC</b>
语法	IORIA i
操作数	$0 \leq i < 255$
操作	ACC   i → ACC
状态影响	Z
说明	ACC和 8 位立即数做“OR”运算，结果存入ACC。
周期	1
举例	IORIA i 执行指令前： i=0x50, ACC=0xAA, Z=0。 执行指令后： ACC=0xFA, Z=0。

<b>IOSTR</b>	<b>Move F-page SFR to ACC</b>
语法	IOSTR F
操作数	$5 \leq F \leq 15$
操作	F-page SFR → ACC
状态影响	--
说明	将F-page特殊寄存器数值给ACC。
周期	1
举例	IOSTR F 执行指令前： F=0x55, ACC=0xAA。 执行指令后： F=0x55, ACC=0x55。

<b>IOST</b>	<b>Load F-page SFR from ACC</b>
语法	IOST F
操作数	$5 \leq F \leq 15$
操作	ACC → F-page SFR
状态影响	--
说明	将ACC的值赋给F-page特殊寄存器
周期	1
举例	IOST F 执行指令前： F=0x55, ACC=0xAA。 执行指令后： F=0xAA, ACC=0xAA。

<b>LCALL</b>	<b>Call Subroutine</b>
语法	LCALL adr
操作数	$0 \leq \text{adr} \leq 1023$
操作	PC + 1 → Top of Stack, adr → PC[9:0]
状态影响	--
说明	长调用子程序。首先将PC+1压入栈顶，然后将 10 位立即数载入PC[9:0]。
周期	2
举例	LCALL SUB 执行指令前： PC=A0, Stack level=1。 执行指令后： PC=address of SUB, Stack[1]=A0+1, Stack pointer =2。

<b>LGOTO</b>	<b>Unconditional Branch</b>
语法	LGOTO adr
操作数	$0 \leq \text{adr} \leq 1023$
操作	$\text{adr} \rightarrow \text{PC}[9:0]$
状态影响	--
说明	无条件长跳转，10位立即数写入PC[9:0]。
周期	2
举例	LGOTO Level 执行指令前： PC=A0。 执行指令后： PC=address of Level。

<b>MOVIA</b>	<b>Move Immediate to ACC</b>
语法	MOVIA i
操作数	$0 \leq i < 255$
操作	$i \rightarrow \text{ACC}$
状态影响	--
说明	8位立即数赋值给ACC。
周期	1
举例	MOVIA i 执行指令前： i=0x55, ACC=0xAA。 执行指令后： ACC=0x55。

<b>MOVAR</b>	<b>Move ACC to R</b>
语法	MOVAR R
操作数	$0 \leq R \leq 63$
操作	$\text{ACC} \rightarrow R$
状态影响	--
说明	ACC赋值给R-page寄存器。
周期	1
举例	MOVAR R 执行指令前： R=0x55, ACC=0xAA。 执行指令后： R=0xAA, ACC=0xAA。

<b>MOVR</b>	<b>Move to ACC or R</b>
语法	MOVR R, d
操作数	$0 \leq R \leq 63$ d = 0, 1
操作	$R \rightarrow \text{dest}$
状态影响	Z
说明	R-page寄存器赋值给d，若d="0"，结果存入ACC；若d="1"，结果存入寄存器R。指令执行后，通过状态标志Z检查R是否为0。
周期	1
举例	MOVR R, d 执行指令前： R=0x0, ACC=0xAA, Z=0, d=0。 执行指令后： R=0x0, ACC=0x00, Z=1。

<b>NOP</b>	<b>No Operation</b>
语法	NOP
操作数	--
操作	No operation.
状态影响	--
说明	空操作。
周期	1
举例	<b>NOP</b> 执行指令前: PC=A0 执行指令后: PC=A0+1

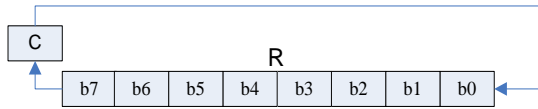
<b>RETIA</b>	<b>Return with Data in ACC</b>
语法	RETIA i
操作数	$0 \leq i < 255$
操作	$i \rightarrow \text{ACC}$ Top of Stack $\rightarrow$ PC
状态影响	--
说明	带参数返回：8 位立即数赋值给 ACC，栈顶地址载入 PC，GIE 标志为 0。
周期	2
举例	<b>RETIA i</b> 执行指令前: GIE=0, Stack pointer =2, i=0x55, ACC=0xAA。 执行指令后: GIE=0, PC=Stack[2], Stack pointer =1, ACC=0x55。

<b>RETIE</b>	<b>Return from Interrupt and Enable Interrupt Globally</b>
语法	RETIE
操作数	--
操作	Top of Stack $\rightarrow$ PC 1 $\rightarrow$ GIE
状态影响	--
说明	中断返回，栈顶地址载入 PC 同时使能中断。
周期	2
举例	<b>RETIE</b> 执行指令前: GIE=0, Stack level=2。 执行指令后: GIE=1, PC=Stack[2], Stack level =1。

<b>RET</b>	<b>Return from Subroutine</b>
语法	RET
操作数	--
操作	Top of Stack $\rightarrow$ PC
状态影响	--
说明	子程序返回，栈顶载入 PC。
周期	2
举例	<b>RET</b> 执行指令前: Stack level=2。 执行指令后: PC=Stack[2], Stack level=1。

**RLR Rotate Left R Through Carry**

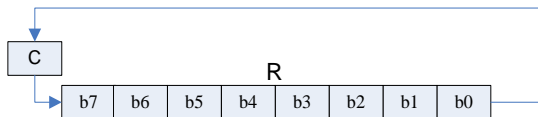
语法 RLR R, d  
 操作数  $0 \leq R \leq 63$   
 $d = 0, 1$   
 操作  $R[7] \rightarrow C, R[6:0] \rightarrow \text{dest}[7:1],$   
 $C \rightarrow \text{dest}[0]$ 。



状态影响 C  
 说明 带进位R循环左移：若d="0"，结果存入ACC；若d="1"，结果存入R。  
 周期 1  
 举例 RLR R, d  
 执行指令前：  
 $R=0xA5, d=1, C=0$ 。  
 执行指令后：  
 $R=0x4A, C=1$ 。

**RRR Rotate Right R Through Carry**

语法 RRR R, d  
 操作数  $0 \leq R \leq 63$   
 $d = 0, 1$   
 操作  $C \rightarrow \text{dest}[7], R[7:1] \rightarrow \text{dest}[6:0],$   
 $R[0] \rightarrow C$ 。



状态影响 C  
 说明 带进位R循环右移：若d="0"，结果存入ACC；若d="1"，结果存入R。  
 周期 1  
 举例 RRR R, d  
 执行指令前：  
 $R=0xA5, d=1, C=0$ 。  
 执行指令后：  
 $R=0x52, C=1$ 。

**SBCAR Subtract ACC and Carry from R**

语法 SBCAR R, d  
 操作数  $0 \leq R \leq 63$   
 $d = 0, 1$   
 操作  $R + (\sim\text{ACC}) + C \rightarrow \text{dest}$   
 状态影响 Z, DC, C

说明 R和ACC带借位减法，若d="0"，结果存入ACC；若d="1"，结果存入R。

周期 1  
 举例 SBCAR R, d  
 (a) 执行指令前：  
 $R=0x05, \text{ACC}=0x06, d=1, C=0$ 。  
 执行指令后：  
 $R=0xFE, C=0$ 。(-2)  
 (b) 执行指令前：  
 $R=0x05, \text{ACC}=0x06, d=1, C=1$ 。  
 执行指令后：  
 $R=0xFF, C=0$ 。(-1)  
 (c) 执行指令前：  
 $R=0x06, \text{ACC}=0x05, d=1, C=0$ 。  
 执行指令后：  
 $R=0x00, C=1(-0), Z=1$ 。  
 (d) 执行指令前：  
 $R=0x06, \text{ACC}=0x05, d=1, C=1$ 。  
 执行指令后：  
 $R=0x1, C=1$ 。( +1)

<b>SBCIA</b>	<b>Subtract ACC and Carry from Immediate</b>
语法	SBCIA i
操作数	$0 \leq i < 255$
操作	$i + (\sim\text{ACC}) + C \rightarrow \text{dest}$
状态影响	Z, DC, C
说明	常数和ACC带借位减法，结果存入ACC。
周期	1
举例	<b>SBCIA i</b> (a) 执行指令前： $i=0x05, \text{ACC}=0x06, C=0$ 。 执行指令后： $\text{ACC}=0xFE, C=0$ 。 (-2) (b) 执行指令前： $i=0x05, \text{ACC}=0x06, C=1$ 。 执行指令后： $\text{ACC}=0xFF, C=0$ 。 (-1) (c) 执行指令前： $i=0x06, \text{ACC}=0x05, C=0$ 。 执行指令后： $\text{ACC}=0x00, C=1$ 。 (-0), Z=1。 (d) 执行指令前： $i=0x06, \text{ACC}=0x05, C=1$ 。 执行指令后： $\text{ACC}=0x1, C=1$ 。 (+1)

<b>SFUNR</b>	<b>Move S-page SFR from ACC</b>
语法	SFUNR S
操作数	$0 \leq S \leq 15$
操作	S-page SFR $\rightarrow$ ACC
状态影响	--
说明	读S-page特殊寄存器到ACC
周期	1
举例	<b>SFUNR S</b> 执行指令前： $S=0x55, \text{ACC}=0xAA$ 。 执行指令后： $S=0x55, \text{ACC}=0x55$ 。

<b>SFUN</b>	<b>Load S-page SFR from ACC</b>
语法	SFUN S
操作数	$0 \leq S \leq 15$
操作	ACC $\rightarrow$ S-page SFR
状态影响	--
说明	ACC写到S-page特殊寄存器。
周期	1
举例	<b>SFUN S</b> 执行指令前： $S=0x55, \text{ACC}=0xAA$ 。 执行指令后： $S=0xAA, \text{ACC}=0xAA$ 。

<b>SLEEP</b>	<b>Enter Halt Mode</b>
语法	SLEEP
操作数	--
操作	00h $\rightarrow$ WDT 00h $\rightarrow$ WDT prescaler 1 $\rightarrow$ /TO 0 $\rightarrow$ /PD
状态影响	/TO, /PD
说明	WDT和分频器 0 清零。/TO标志为0, /PD清零, IC进入睡眠。
周期	1
举例	<b>SLEEP</b> 执行指令前： /PD=1, /TO=0。 执行指令后： /PD=0, /TO=1。

<b>SUBAR</b>	<b>Subtract ACC from R</b>
语法	SUBAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$R - ACC \rightarrow dest$
状态影响	Z, DC, C
说明	R 减去ACC, 若d="0", 结果存入ACC。 若d="1", 结果存入R。
周期	1
举例	SBCAR R, d (a) 执行指令前: R=0x05, ACC=0x06, d=1。 执行指令后: R=0xFF, C=0. (-1) (b) 执行指令前: R=0x06, ACC=0x05, d=1。 执行指令后: R=0x01, C=1. (+1)

<b>SWAPR</b>	<b>Swap High/Low Nibble in R</b>
语法	SWAPR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$R[3:0] \rightarrow dest[7:4]$ , $R[7:4] \rightarrow dest[3:0]$
状态影响	--
说明	寄存器半字节交换, 若d="0", 结果存入ACC; 若d="1", 结果存入R。
周期	1
举例	SWAPR R, d 执行指令前: R=0xA5, d=1。 执行指令后: R=0x5A。

<b>SUBIA</b>	<b>Subtract ACC from Immediate</b>
语法	SUBIA i
操作数	$0 \leq i < 255$
操作	$i - ACC \rightarrow ACC$
状态影响	Z, DC, C
说明	8 位立即数减ACC, 结果存入ACC。
周期	1
举例	SUBIA i (a) 执行指令前: i=0x05, ACC=0x06。 执行指令后: ACC=0xFF, C=0. (-1) (b) 执行指令前: i=0x06, ACC=0x05, d=1。 执行指令后: ACC=0x01, C=1. (+1)

<b>TABLEA</b>	<b>Read ROM data</b>
语法	TABLEA
操作数	--
操作	ROM data{ TBHP, ACC } [7:0] $\rightarrow ACC$ 。 ROM data{TBHP, ACC} [13:8] $\rightarrow TBHD$ 。
状态影响	--
说明	ROM查表指令, 高字节存入TBHD, 低字节存入ACC。
周期	2
举例	TABLEA 执行指令前: TBHP=0x02, ACC=0x34。 TBHD=0x01。 ROM data[0x234]= 0x35AA。 执行指令后: TBHD=0x35, ACC=0xAA。

<b>T0MD</b>	<b>Load ACC to T0MD</b>
语法	T0MD
操作数	--
操作	ACC → T0MD
状态影响	--
说明	ACC写入T0MD寄存器。
周期	1
举例	T0MD 执行指令前： T0MD=0x55, ACC=0xAA。 执行指令后： T0MD=0xAA。

<b>XORAR</b>	<b>Exclusive-OR ACC with R</b>
语法	XORAR R, d
操作数	$0 \leq R \leq 63$ $d = 0, 1$
操作	$ACC \oplus R \rightarrow dest$
状态影响	Z
说明	ACC和R做“XOR”运算，若d=“0”，结果存入ACC；若d=“1”，结果存入R。
周期	1
举例	XORAR R, d 执行指令前： R=0xA5, ACC=0xF0, d=1。 执行指令后： R=0x55。

<b>T0MDR</b>	<b>Move T0MD to ACC</b>
语法	T0MDR
操作数	--
操作	T0MD → ACC
状态影响	--
说明	读T0MD寄存器到ACC。
周期	1
举例	T0MDR 执行指令前： T0MD=0x55, ACC=0xAA。 执行指令后： ACC=0x55。

<b>XORIA</b>	<b>Exclusive-OR Immediate with ACC</b>
语法	XORIA i
操作数	$0 \leq i < 255$
操作	$ACC \oplus i \rightarrow ACC$
状态影响	Z
说明	ACC和8位立即数做“XOR”运算。
周期	1
举例	XORIA i 执行指令前： i=0xA5, ACC=0xF0。 执行指令后： ACC=0x55。

**5. 配置字节表**

项目	名称	选项				
1	高频振荡模式	1. I_HRC	2. E_HXT	3. E_XT		
2	低频振荡模式	1. I_LRC	2. E_LXT			
3	内部高速 RC 频率	1. 1MHz	2. 2MHz	3. 4MHz		
		4. 8MHz	5. 16MHz	6. 20MHz		
4	外部高频振荡器	1. $6\text{MHz} < F_{\text{HOSC}} \leq 8\text{MHz}$	2. $8\text{MHz} < F_{\text{HOSC}} \leq 10\text{MHz}$		3. $10\text{MHz} < F_{\text{HOSC}} \leq 12\text{MHz}$	
		4. $12\text{MHz} < F_{\text{HOSC}} \leq 16\text{MHz}$	5. $16\text{MHz} < F_{\text{HOSC}} \leq 20\text{MHz}$			
5	指令时钟	1. 2 个振荡周期		2. 4 个振荡周期		
6	看门狗定时器	1. 看门狗开启（程序控制）				
		2. 看门狗关闭（永远关闭）				
7	看门狗定时器事件	1. 看门狗复位	2. 看门狗中断			
8	定时器 0 时钟源	1. EX_CKI	2. 低频振荡器（I_LRC/E_LXT）			
9	PB.3	1. PB.3 为 I/O 口	2. PB.3 为复位脚			
10	PB.4	1. PB.4 为 I/O 口	2. PB.4 输出指令时钟			
11	上电复位时间	1. 140us	2. 4.5ms	3. 18ms	4. 72ms	5. 288ms
12	看门狗定时器时基	1. 3.5ms	2. 15ms	3. 60ms	4. 250ms	
13	LVR 开关设定	1. 寄存器控制		2. LVR 永远开启		
14	LVR 电压	1. 1.6V	2. 1.8V	3. 2.0V	4. 2.2V	5. 2.4V
		6. 2.7V	7. 3.0V	8. 3.3V	9. 3.6V	10. 4.2V
15	VDD 电压	1. 3.0V	2. 4.5V	3. 5.0V		
16	读取输出口数据	1. I/O 口		2. 寄存器		
17	E_LXT 强化起振开关设定	1. 自动关闭		2. 寄存器关闭		
18	EX_CKI to Inst. Clock	1. 同步		2. 不同步		
19	上电时钟源	1. 高速（I_HRC/E_HXT/E_XT）		2. 低速（I_LRC/E_LXT）		
20	PWM 输出引脚	1. PB.6		2. PB.2		
21	Buzzer 输出引脚	1. PB.7		2. PB.2		
22	输入高电压（VIH）	1. CMOS（0.7VDD）		2. TTL（0.5VDD）		
23	输入低电压（VIL）	1. CMOS（0.3VDD）		2. TTL（0.2VDD）		

表 16 配置字节



## 6. 电气特性

### 6.1 最大绝对值

符号	参数	额定值	单位
$V_{DD} - V_{SS}$	工作电压	-0.5 ~ +6.0	V
$V_{IN}$	输入电压	$V_{SS}-0.3V \sim V_{DD}+0.3$	V
$T_{OP}$	工作温度	-40 ~ +85	°C
$T_{ST}$	储存温度	-40 ~ +125	°C

### 6.2 直流电气特性

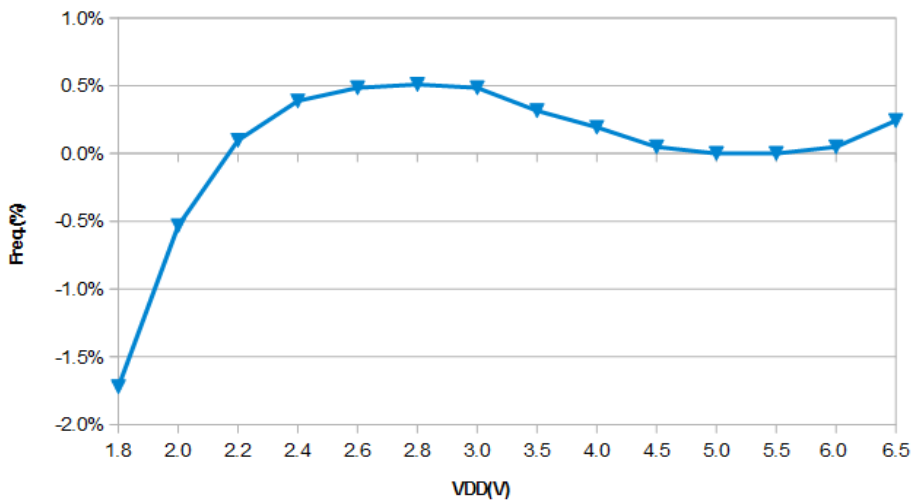
(  $F_{INST}=F_{HOSC}/4$ ,  $F_{HOSC}=16MHz@I_{HRC}$ , WDT开启, 环境温度 $T_A=25^{\circ}C$ )

符号	参数	$V_{DD}$	最小值	典型值	最大值	单位	条件
$V_{DD}$	工作电压	--	3.3	--	5.5	V	$F_{INST}=20MHz @ I_{HRC}/2$
			2.2				$F_{INST}=20MHz @ I_{HRC}/4$
			3.0				$F_{INST}=16MHz @ E_{HXT}/2$
			2.0				$F_{INST}=16MHz @ E_{HXT}/4$
			2.0				$F_{INST}=8MHz @ I_{HRC}/4 \& I_{HRC}/2$
							$F_{INST}=8MHz @ E_{HXT}/4 \& E_{HXT}/2$
			1.8				$F_{INST}=4MHz @ I_{HRC}/4 \& I_{HRC}/2$
							$F_{INST}=4MHz @ E_{XT}/4 \& E_{XT}/2$
							$F_{INST}=32KHz @ I_{LRC}/4 \& I_{LRC}/2$
							$F_{INST}=32KHz @ E_{LXT}/4 \& E_{LXT}/2$
$V_{IH}$	输入高电平	5V	4.0	--	--	V	RSTb (0.8 $V_{DD}$ )
		3V	2.4	--	--	V	所有I/O引脚, EX_CKI, INT CMOS (0.7 $V_{DD}$ )
		5V	3.5	--	--		
		3V	2.1	--	--	V	所有I/O引脚, EX_CKI TTL (0.5 $V_{DD}$ )
		5V	2.5	--	--	V	RSTb (0.2 $V_{DD}$ )
		3V	1.5	--	--		
$V_{IL}$	输入低电平	5V	--	--	1.0	V	RSTb (0.2 $V_{DD}$ )
		3V	--	--	0.6	V	所有I/O引脚, EX_CKI, INT CMOS (0.3 $V_{DD}$ )
		5V	--	--	1.5		
		3V	--	--	0.9	V	所有I/O引脚, EX_CKI TTL (0.2 $V_{DD}$ )
		5V	--	--	1.0	V	RSTb (0.2 $V_{DD}$ )
		3V	--	--	0.6		
$I_{OH}$	输出驱动电流	5V	--	-20	--	mA	$V_{OH}=4.0V$
		3V	--	-10	--		$V_{OH}=2.0V$
$I_{OL}$	输出灌电流	5V	--	40	--	mA	$V_{OL}=1.0V$
		3V	--	26	--		
$I_{IR}$	红外输出灌电流	5V	--	40	--	mA	$V_{OL}=1.0V$
		3V	--	26	--		

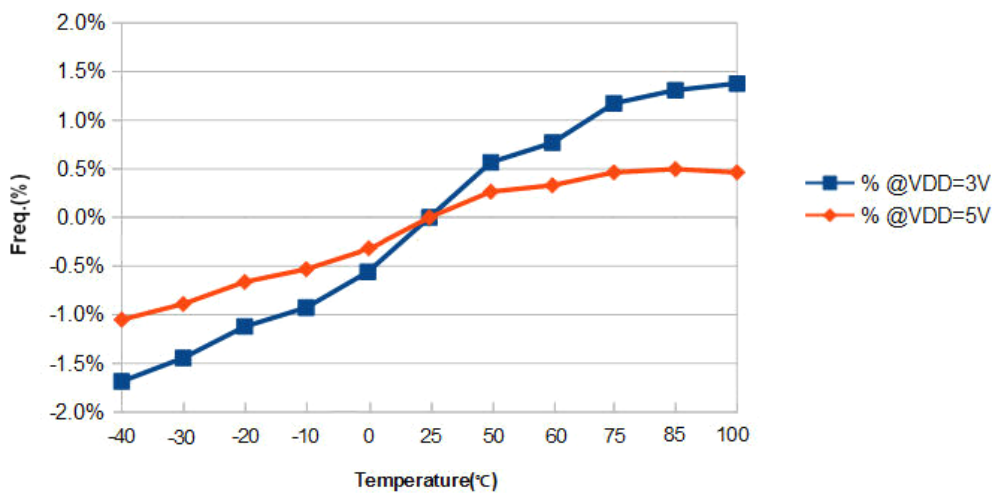
符号	参数	V <sub>DD</sub>	最小值	典型值	最大值	单位	条件
I <sub>OP</sub>	工作电流	正常模式					
		5V	--	2.7	--	mA	F <sub>HOSC</sub> =20MHz @ I <sub>HRC</sub> /2 & E <sub>HXT</sub> /2
		3V	--	1.3	--		
		5V	--	2.1	--	mA	F <sub>HOSC</sub> =20MHz @ I <sub>HRC</sub> /4 & E <sub>HXT</sub> /4
		3V	--	1.0	--		
		5V	--	2.4	--	mA	F <sub>HOSC</sub> =16MHz @ I <sub>HRC</sub> /2 & E <sub>HXT</sub> /2
		3V	--	1.1	--		
		5V	--	1.9	--	mA	F <sub>HOSC</sub> =16MHz @ I <sub>HRC</sub> /4 & E <sub>HXT</sub> /4
		3V	--	0.8	--		
		5V	--	1.5	--	mA	F <sub>HOSC</sub> =8MHz @ I <sub>HRC</sub> /2 & E <sub>HXT</sub> /2
		3V	--	0.7	--		
		5V	--	1.3	--	mA	F <sub>HOSC</sub> =8MHz @ I <sub>HRC</sub> /4 & E <sub>HXT</sub> /4
		3V	--	0.5	--		
		5V	--	1.1	--	mA	F <sub>HOSC</sub> =4MHz @ I <sub>HRC</sub> /2 & E <sub>XT</sub> /2
		3V	--	0.5	--		
		5V	--	0.9	--	mA	F <sub>HOSC</sub> =4MHz @ I <sub>HRC</sub> /4 & E <sub>XT</sub> /4
		3V	--	0.4	--		
		5V	--	0.9	--	mA	F <sub>HOSC</sub> =1MHz @ I <sub>HRC</sub> /2 & E <sub>XT</sub> /2
		3V	--	0.4	--		
		5V	--	0.8	--	mA	F <sub>HOSC</sub> =1MHz @ I <sub>HRC</sub> /4 & E <sub>XT</sub> /4
3V	--	0.4	--				
慢速模式							
I <sub>STB</sub>	待机电流	5V	--	6.0	--	uA	F <sub>HOSC</sub> 关闭, F <sub>LOSC</sub> =32KHz @ I <sub>LRC</sub> /2
		3V	--	2.7	--		
		5V	--	7.0	--	uA	F <sub>HOSC</sub> 关闭, F <sub>LOSC</sub> =32KHz @ E <sub>LXT</sub> /2
		3V	--	2.9	--		
		5V	--	4.3	--	uA	F <sub>HOSC</sub> 关闭, F <sub>LOSC</sub> =32KHz @ I <sub>LRC</sub> /4
		3V	--	1.8	--		
		5V	--	5.3	--	uA	F <sub>HOSC</sub> 关闭, F <sub>LOSC</sub> =32KHz @ E <sub>LXT</sub> /4
		3V	--	2.0	--		
I <sub>HALT</sub>	睡眠电流	5V	--	--	0.5	uA	睡眠模式, 关闭WDT
		3V	--	--	0.2		
		5V	--	--	5.0	uA	睡眠模式, 开启WDT
		3V	--	--	2.0		
R <sub>PH</sub>	上拉电阻	5V	--	50	--	KΩ	上拉电阻
		3V	--	100	--		
R <sub>PL</sub>	下拉电阻	5V	--	50	--	KΩ	下拉电阻
		3V	--	100	--		

### 6.3 特性图

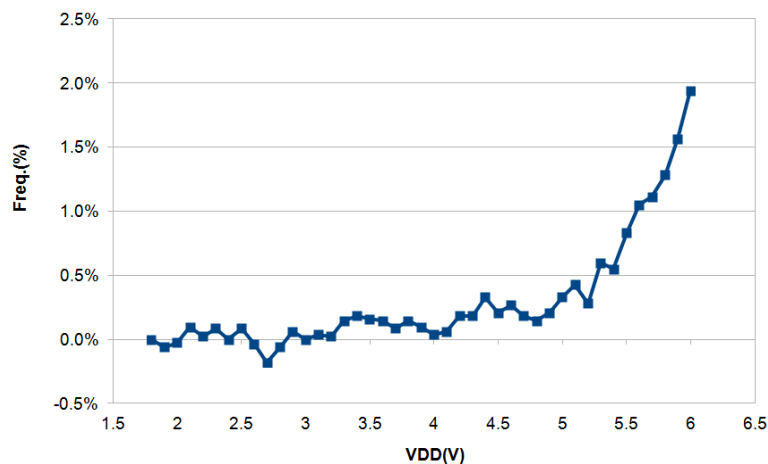
#### 6.3.1 高速 RC 振荡频率与电源电压曲线图



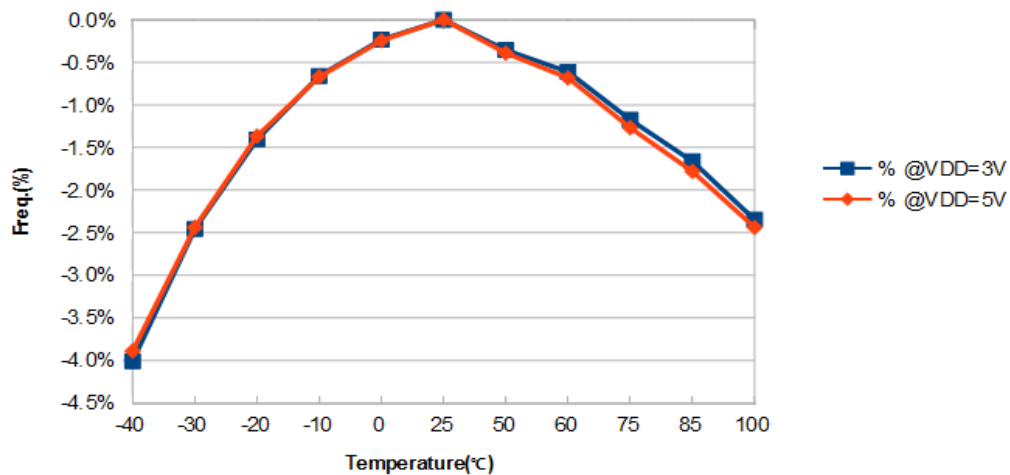
#### 6.3.2 高速 RC 振荡频率与温度曲线图



### 6.3.3 低速 RC 振荡频率与电源电压曲线图



### 6.3.4 低速 RC 振荡频率与温度曲线图

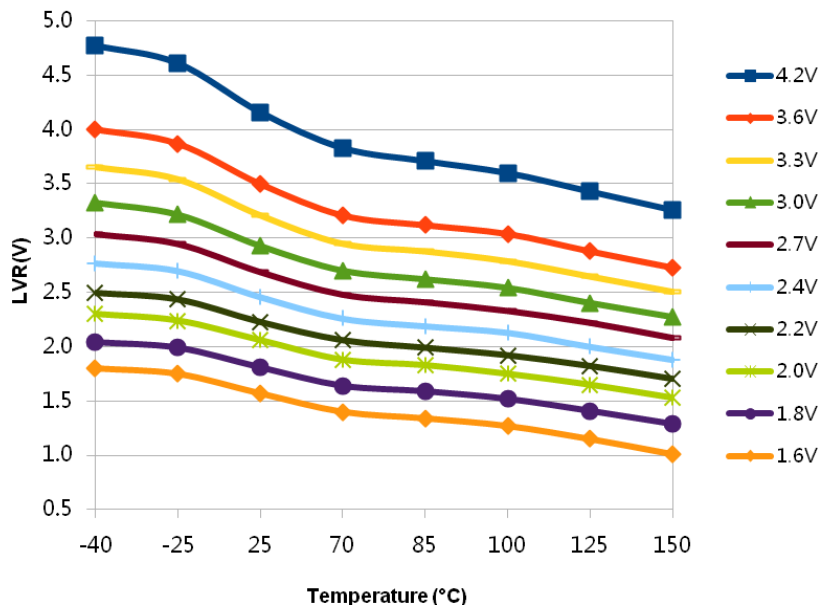


## 6.4 建议工作电压

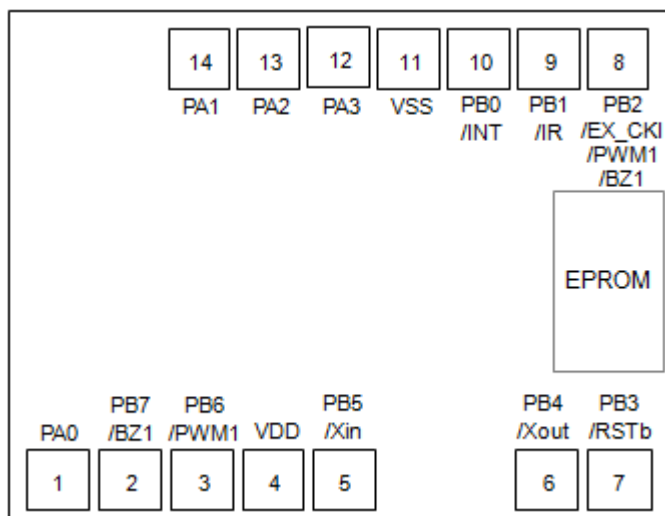
建议工作电压（温度范围：-40 °C ~ +85 °C）

频率	最小电压	最大电压	LVR: 默认值 (25 °C)	LVR: 建议值 (-40 °C ~ +85 °C)
20M/2T	3.3V	5.5V	3.6V	4.2V
16M/2T	3.0V	5.5V	3.3V	3.6V
20M/4T	2.2V	5.5V	2.4V	3.0V
16M/4T	2.0V	5.5V	2.2V	2.4V
8M (2T or 4T)	2.0V	5.5V	2.2V	2.4V
≤6M (2T or 4T)	1.8V	5.5V	2.0V	2.2V

### 6.5 LVR电压与温度曲线图

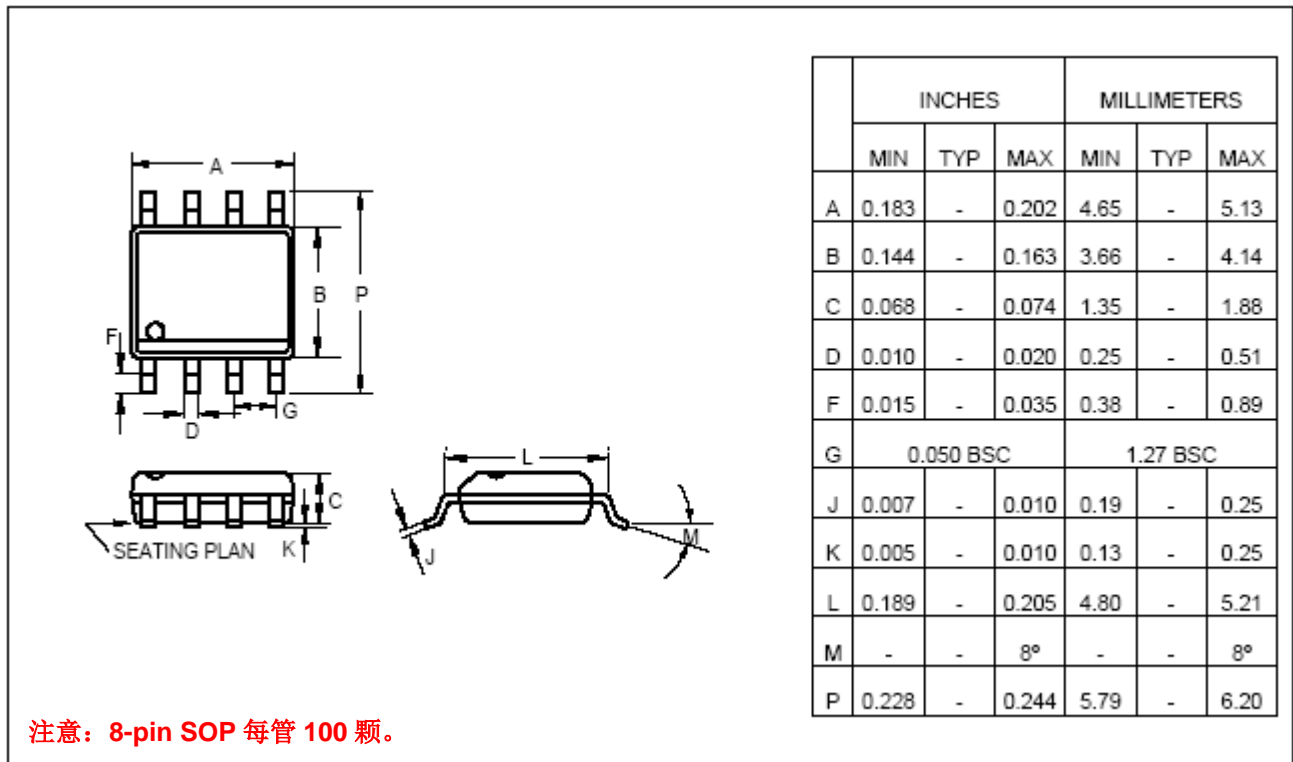


### 7. 芯片脚位坐标图

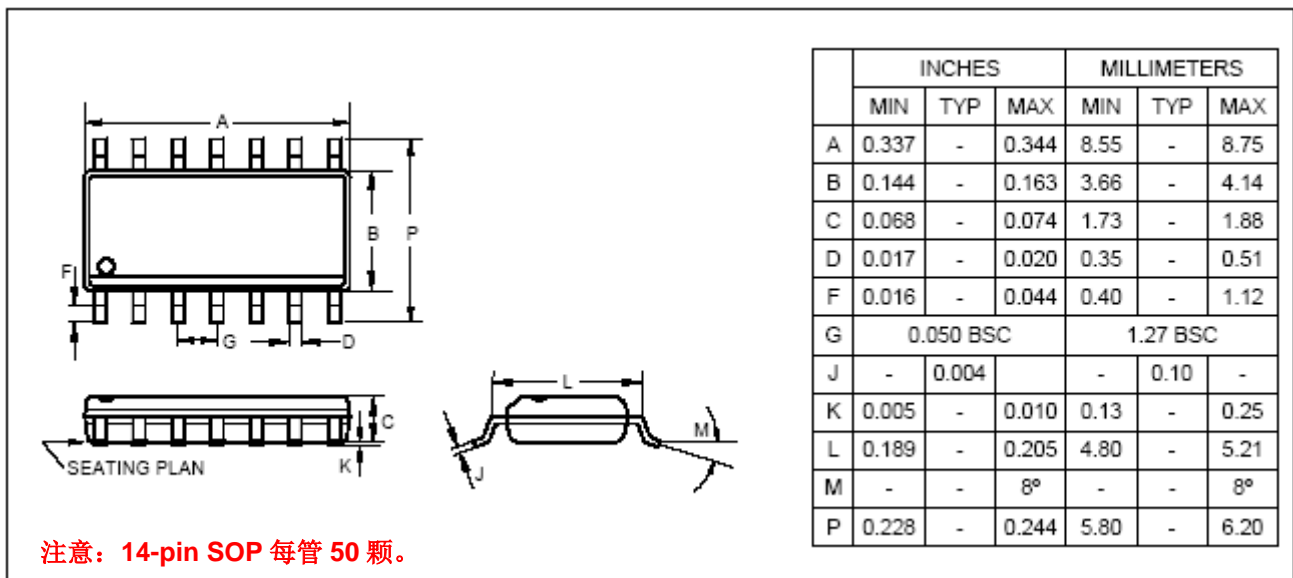


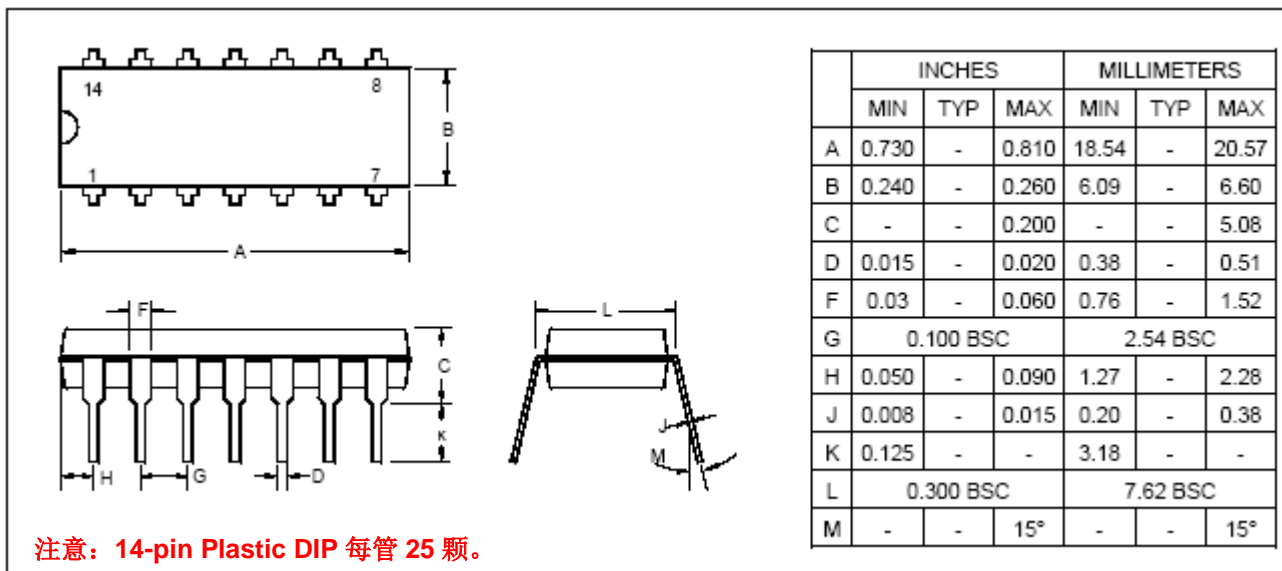
8. 封装尺寸

8.1 8 引脚SOP (150 毫寸)



8.2 14 引脚SOP (150 毫寸)



**8.3 14 引脚DIP (300 毫寸)**

**9. 订购信息**

产品名称	封装类型	引脚数	封装尺寸	配送方式
NY8A053D	Die	--	--	--
NY8A053DS8	SOP	8	150 mil	卷装：每卷 2.5K颗 管装：每管 100 颗
NY8A053DS14	SOP	14	150 mil	卷装：每卷 2.5K颗 管装：每管 50 颗
NY8A053DP14	PDIP	14	300 mil	管装：每管 25 颗