8 位 µC 内含 8k*16 OTP ROM 遥控器控制电路

概述

ET4007MTC 是一款高性能，低成本的 8 位单片机，它是内置 4K/8K*16bit OTP ROM、内置 5MHz 高精度晶振、13 个通用 IO 或键扫描口以及接收发射电路的遥控器控制芯片。

ET4007MTC 适用场合：普通遥控器或学习型遥控器的控制主片。

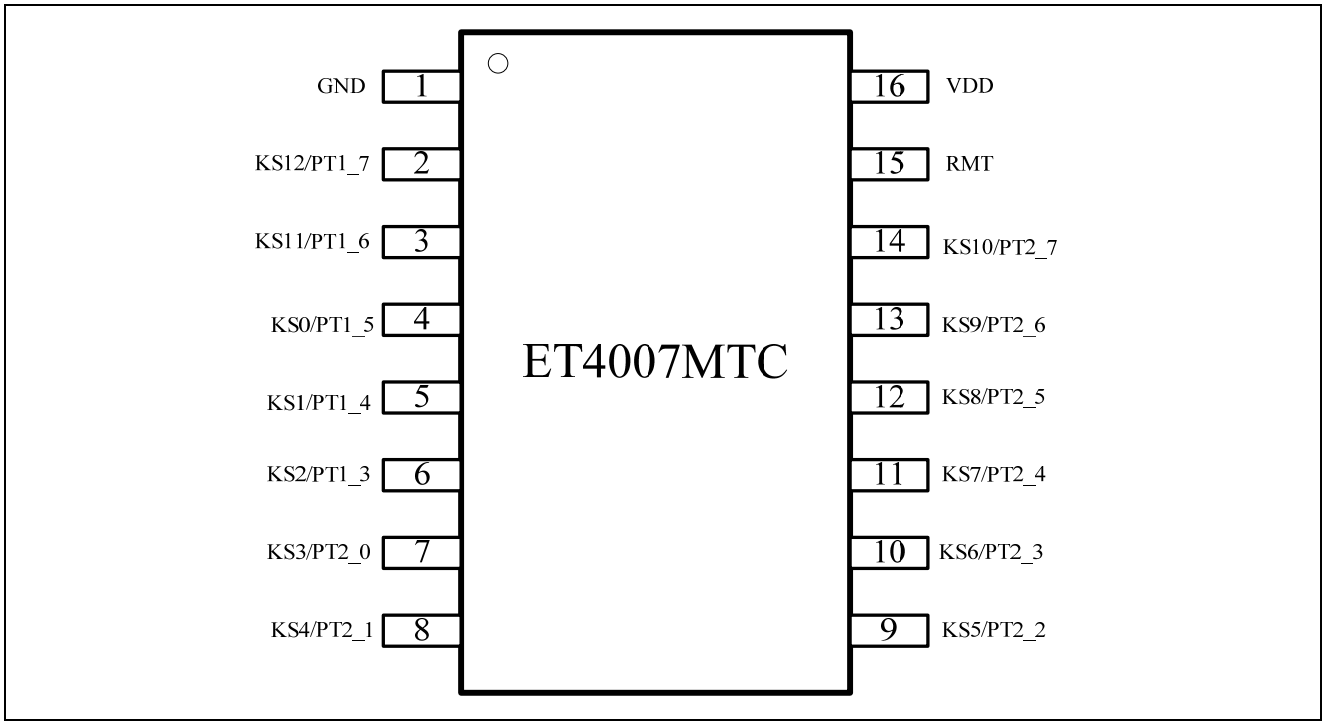
功能特点

- 8-bit 微控制器，56 条字指令
- 内置 4K/8K*16bit OTP 程序存储器，512byte 数据存储器
- 内核工作电压范围为 2.0v 到 3.6v
- 内置 5MHz 振荡器，可以微调；误差小于+/- 1%
- 静态电流约为 1.0µA（三角键盘扫描开启时）
- 8 级硬件堆栈
- 3 个中断源（内部：2 个定时中断，外部：1 个外部中断（13 个 IO 共享））
- 13 个 IO 端口可以任意配置 1~13bit 通用 I/O 或者 1~13 个键盘扫描端口
- 程序存储器可以设置成 4K*16bit MTP，可烧写两次
- T-type 型键盘按键键值硬件自动采集
- 睡眠模式下 T-type 或者 M-type 按键动作自动唤醒功能
- 内置两个定时器 TimeA 和 TimeB
- 内置 IR 接收模块，接收灵敏度可选（2uA，5uA，8uA，11uA）
- 内置 IR LED 发射驱动管，发射驱动能力八级可选
  I_{OLmin} = 120mA at V_{DD} =3V and V_O =0.3V and pwm_curt[2:0]=000
  I_{OLmax} = 260mA at V_{DD} =3V and V_O =0.3V and pwm_curt[2:0]=111
- 看门狗定时器
- 上电复位
- 低电压复位：1.65V
- 封装 SOP16

应用

- 普通遥控器
- 学习型遥控器
- 普通 IO 控制器

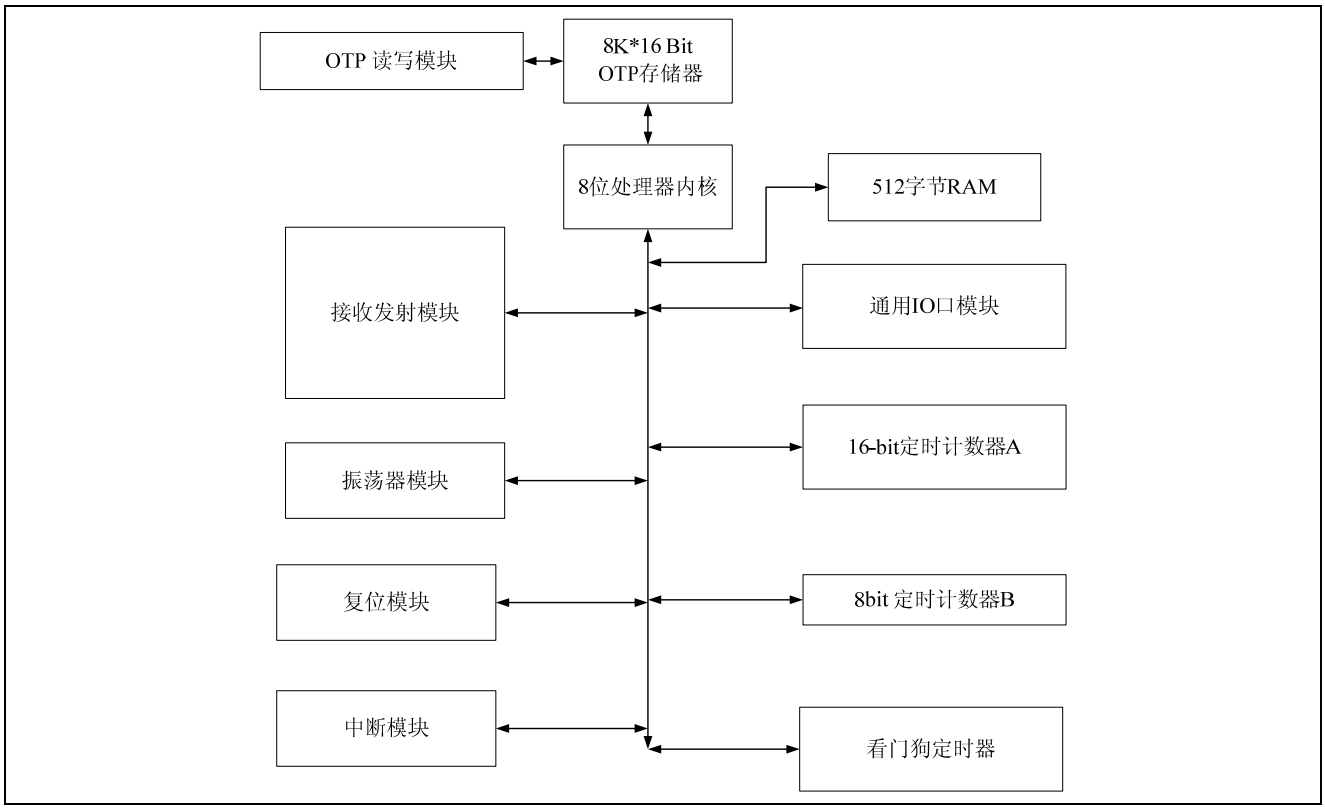
管脚排列图



管脚说明

名称	方向	管脚	说明
GND	I/O	1	地
KS12/PT1_7	I/O	2	键扫描端口 KS12 或者普通 IO 口 PT1.7
KS11/PT1_6	I/O	3	键扫描端口 KS11 或者普通 IO 口 PT1.6
KS0/PT1_5	I/O	4	键扫描端口 KS0 或者普通 IO 口 PT1.5
KS1/PT1_4	I/O	5	键扫描端口 KS1 或者普通 IO 口 PT1.4
KS2/PT1_3/SDA	I/O	6	键扫描端口 KS2 或者普通 IO 口 PT1.3 或者 硬件 I2C 的 SDA 端口
KS3/PT2_0/PWM/SCK	I/O	7	键扫描端口 KS3 或者普通 IO 口 PT2.0 或者 PWM 输出 或者硬件 I2C 的 SCK 端口
KS4/PT2_1/PWM	I/O	8	键扫描端口 KS4 或者普通 IO 口 PT2.1 或者 PWM 输出
KS5/PT2_2	I/O	9	键扫描端口 KS5 或者普通 IO 口 PT2.2
KS6/PT2_3	I/O	10	键扫描端口 KS6 或者普通 IO 口 PT2.3
KS7/PT2_4	I/O	11	键扫描端口 KS7 或者普通 IO 口 PT2.4
KS8/PT2_5	I/O	12	键扫描端口 KS8 或者普通 IO 口 PT2.5
KS9/PT2_6	I/O	13	键扫描端口 KS9 或者普通 IO 口 PT2.6
KS10/PT2_7/PWM	I/O	14	键扫描端口 KS10 或者普通 IO 口 PT2.7 或者 PWM 输出
RMT	I/O	15	接收或者发射端口
VDD	I/O	16	电源

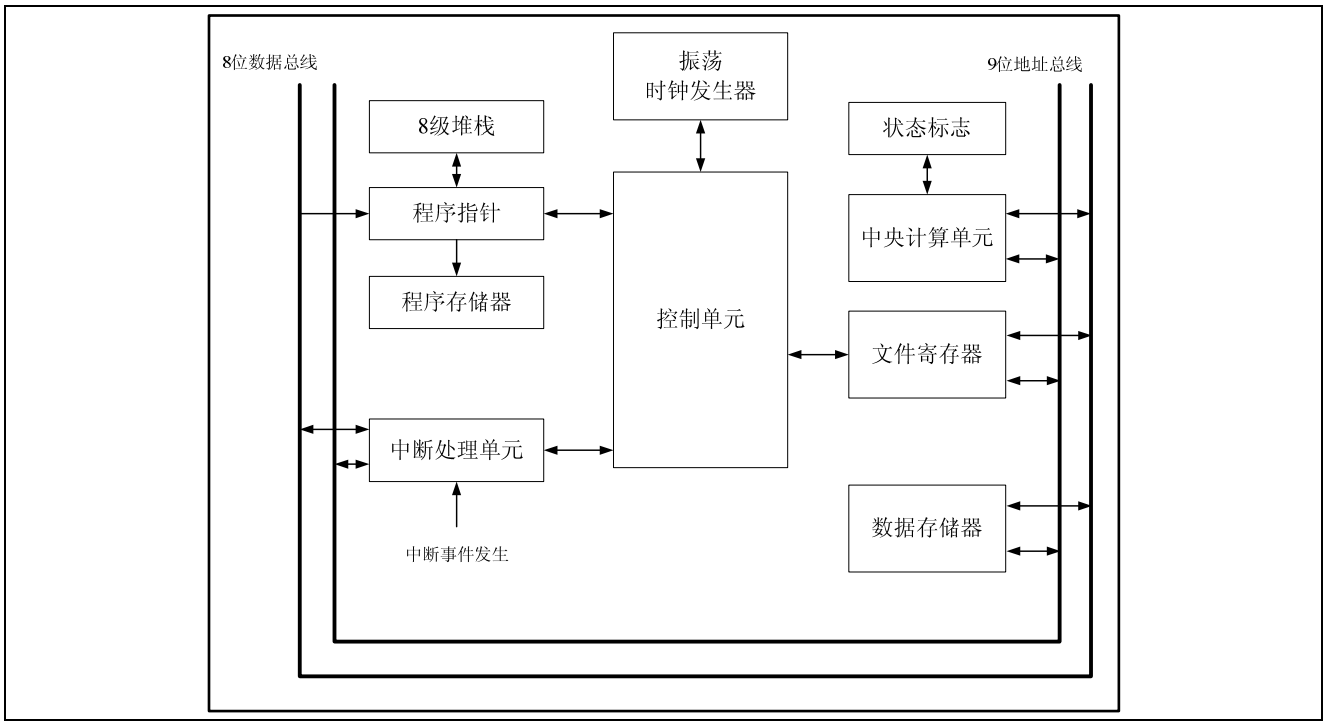
功能框图



功能说明

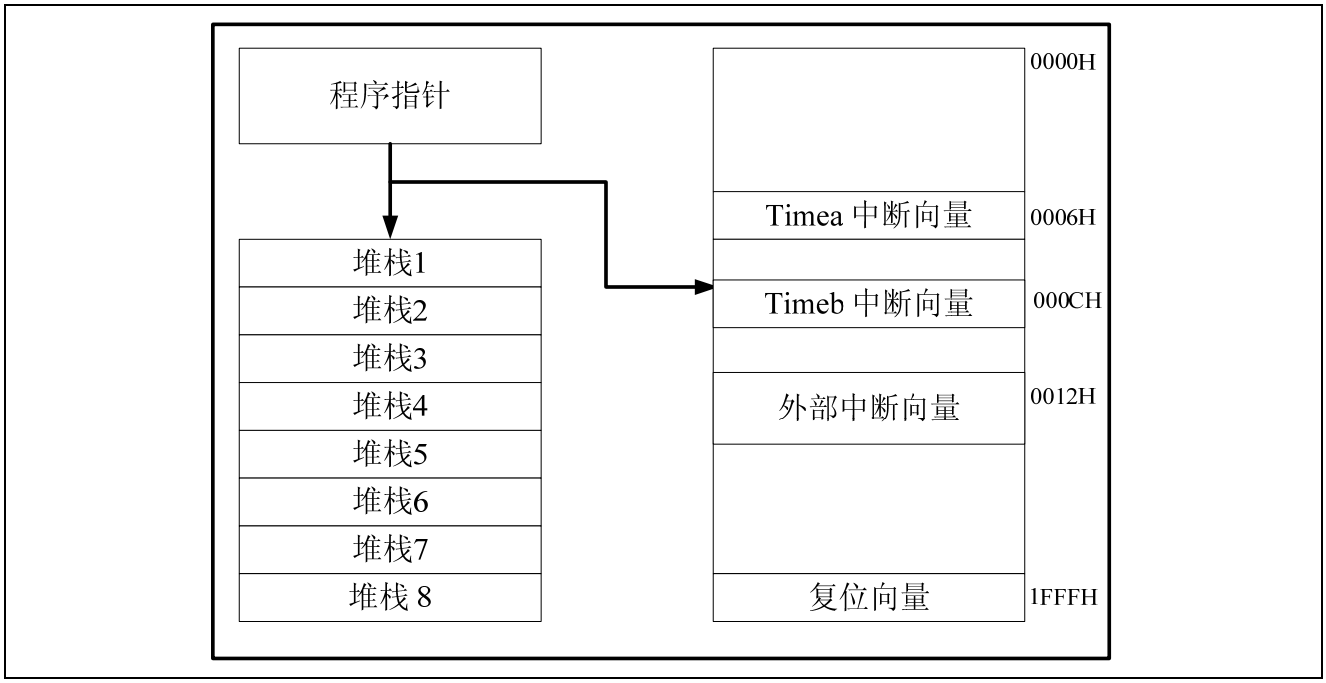
1 CPU core

1.1 CPU core 框图



1.2 程序存储器

图中程序计数器的位数为 13bit，单条指令长度为 16bit，也即是该存储器大小为 8K*16bit。复位时程序计数器初值为 1FFFH，中断时程序计数器值为 0006H、 000CH 或者 0012H。



1.3 数据存储器

数据存储器分为三个部分， 000H~007H 是系统特殊寄存器，例如中断标志寄存器，中断使能寄存器，工作寄存器等，008H~01FH 为外设特殊寄存器，例如定时器，接收发射模块电流设置等，020H~1FFH 为一般数据寄存器，主要用于存储用户数据。

地址	名称	内容							
000H	IND0	使用 FRS0 的内容作为地址访问 RAM							
001H	IND1	使用 FRS1 的内容作为地址访问 RAM							
002H	FRS0	RAM 的间接地址指针							
003H	FRS1	RAM 的间接地址指针							
004H	STATUS			lvd_f	pd	to	dc	c	z
005H	WORK	工作寄存器							
006H	INTF				tmaif	tmbif	capif	i2cif	e0if
007H	INTE	gie			tmaie	tmaie	capie	i2cie	e0ie
008H~01FH		外设特殊寄存器区							
020H~1FFH		用户 RAM 区							

- IND0，IND1：间接寻址模式地址
- FRS0，FRS1：间接寻址模式指针
- lvd_f 为低压检测标志，2.0 或者 2.2V 电压检测标志，由烧写 code option 决定
- pd：掉电标志。写 0 清除或者上电复位清除。由休眠指令设置
- to：看门狗暂停标志。写 0 清除或者上电清除。由看门狗暂停设置
- dc：位进位标志，ADDWF(c)和 SUBWF（c），这个位由执行第四位的结果设置
- c：进位标志（与借位相反）

- z: 零标志位
- gie: 全局中断使能位，高电平有效
- tmaie，tmaif: 16 位定时器中断使能和中断标志，高电平有效
- tmbie，tmbif: 8 位定时器中断使能和中断标志，高电平有效
- capie，capif:: RMT 端口信号周期捕获中断使能和中断标志，高电平有效
- e0ie，e0if: 外部中断使能和中断标志，高电平有效
- i2cie，i2cif: 硬件 I2C 端口数据接收和发射中断使能和中断标志，高电平有效

1.4 系统特殊寄存器

1.4.1 间接寻址寄存器

IND0，IND1 并不存在实际的物理地址与其对应，它们与 FRS0，FRS1 以及 DP[1:0]配合起来使用实现了间接寻址的功能，其中 DP[1:0]是由 SETDP 指令来设置。

例如：设置 IND0，FRS0 以及 DP 指针，可以实现数据存取范围内的寻址。

```
SETDP   00H ；dp point
MOVLW 80H ；w=80H
MOVWF FRS0，f ； FRS0=80H
MOVLW 15H ；w=15H
MOVWF IND0，f ； (FSR0})=(80H) = 15H

SETDP   01H ；dp point
MOVLW 80H ；w=80H
MOVWF FRS0，f ； FRS0=80H
MOVLW 15H ；w=15H
MOVWF IND0，f ； (FSR0})=(180H) = 15H
```

1.4.2 状态寄存器

c（bit1）：R/W，Carry Flag or (~Borrow)

z（bit0）：R/W，z will be set if ALU operation is zero. Reset otherwise.

```
Example:
M(80H)=3FH   M(81H)=F0H   WORK=99H
ADDWF 80H, f ;3FH + 99H = D8H dc=1, c=0, z=0
ADDWF 81H, f ;F0H + 99H = 89H dc=0, c=1, z=0
SUBWF 80H, f ;3FH - 99H = A6H dc=1, c=0, z=0
SUBWF 80H, f ;F0H - 99H = 57H dc=0 c=1, z=0  减法 c 等于实际借位的取反
```

1.4.3 中断标志和中断使能寄存器

设置全局中断使能位以及某个中断使能，当中断标志出现时，程序会响应中断，程序指针会跳到 0006H、000CH 或 0012H 执行，在执行该中断程序时，不会再响应其它中断。中断标志位需要软件进行清除。另外，在 sleep 和 halt 模式下，通过外部中断可以唤醒 cpu 进行工作。

1.5 外设特殊寄存器

地址	名称	内容						
008H	MCK			timeclksel	mclksel[1:0]		mcuclksel	
009H	TSETAL	tseta[7:0]						
00AH	TSETAH	tseta[15:8]						
00BH	PWMCON	pwm_sel	duty_sel[1:0]		tcouta_dir	pwm_oen	pwm_port_sel[2:0]	
00CH	TCOUTAL	tcouta[7:0](R)						
00DH	TCOUTAH	tcouta[15:8](R)						
00EH	TCCONA	tcrsta	cap_sel	cap_deben	cap_phaen	tcena	cclka_sel[2:0]	
00FH	TSETB	tsetb[7:0]						
010H	TCOUTB	tcoutb[7:0](R)						
011H	TCCONB	tcrstb				tcenb	cclkb_sel[2:0]	
012H	KEY_VAULE	key_value[7:0](R)						
013H	PT1	pt1[7:3]						
014H	PT1EN	pt1en[7:3]						
015H	PT1PU	pt1pu[7:3]						
016H	PT2	pt2[7:0]						
017H	PT2EN	pt2en[7:0]						
018H	PT2PU	pt2pu[7:0]						
019H	PT1SEL	pt1sel[7:3]						
01AH	PT2SEL	pt2sel[7:0]						
01BH	PT1MR	pt1mr[7:3]						
01CH	PT2MR	pt2mr[7:0]						
01DH	WDTCTR			osclen	scanclken	wdten	wts[2:0]	
01EH	RMTCTR	sens_curt[1:0]		pwm_curt[2]	rx_en	tx_en	pwm_curt[1:0]	rmt_in(R)
01FH	I2CCON	i2cen	i2c_ready	i2c_sto(R)	i2c_rw(R)	i2c_adr_sel	pt1od[5:3]	

注：R→只读

1.6 指令集

指令	解释	指令周期	二进制码	影响状态位
寄存器 f 字节操作(34)				
NOP	No operation	1	0000_0000_0000_0000	none
CLRF f	f=0	1	0000_001f_ffff_ffff	z
ADDWF(d=0) f	w=f+w	1	0000_010f_ffff_ffff	c,dc,z
ADDWF(d=1) f	f=f+w	1	0000_011f_ffff_ffff	c,dc,z
INCF(d=0) f	w=f+1	1	0000_100f_ffff_ffff	z
INCF(d=1) f	f=f+1	1	0000_101f_ffff_ffff	z
INCFSZ(d=0) f	w=f+1(skip if w=0)	1,2	0000_110f_ffff_ffff	none
INCFSZ(d=1) f	f=f+1(skip if f=0)	1,2	0000_111f_ffff_ffff	none
DECF(d=0) f	w=f-1	1	0001_000f_ffff_ffff	z
DECF(d=1) f	f=f-1	1	0001_001f_ffff_ffff	z

DECFSZ(d=0) f	w=f-1(skip if w=0)	1,2	0001_010f_ffff_ffff	none
DECFSZ(d=1) f	f=f-1(skip if f=0)	1,2	0001_011f_ffff_ffff	none
SUBWF(d=0) f	w=f-w	1	0001_100f_ffff_ffff	c,dc,z
SUBWF(d=1) f	f=f-w	1	0001_101f_ffff_ffff	c,dc,z
COMF(d=0) f	w=~f	1	0001_110f_ffff_ffff	z
COMF(d=1) f	f=~f	1	0001_111f_ffff_ffff	z
MOVFW(F-W) f	w=f	1	0010_000f_ffff_ffff	none
MOVWF(W-F) f	f=w	1	0010_001f_ffff_ffff	none
ADDWFC(d=0) f	w=f+w+c	1	0010_010f_ffff_ffff	c,dc,z
ADDWFC(d=1) f	f=f+w+c	1	0010_011f_ffff_ffff	c,dc,z
ANDWF(d=0) f	w=f&w	1	0010_100f_ffff_ffff	z
ANDWF(d=1) f	f=f&w	1	0010_101f_ffff_ffff	z
IORWF(d=0) f	w=f|w	1	0010_110f_ffff_ffff	z
IORWF(d=1) f	f=f|w	1	0010_111f_ffff_ffff	z
XORWF(d=0) f	w=f^w	1	0011_000f_ffff_ffff	z
XORWF(d=1) f	f=f^w	1	0011_001f_ffff_ffff	z
RLF(d=0) f	w[7:0]={f[6:0],c}	1	0011_010f_ffff_ffff	c,z
RLF(d=1) f	f[7:0]={f[6:0],c}	1	0011_011f_ffff_ffff	c,z
SUBWFC(d=0) f	w=f+(~w)+c	1	0011_100f_ffff_ffff	c,dc,z
SUBWFC(d=1) f	f=f+(~w)+c	1	0011_101f_ffff_ffff	c,dc,z
RRF(d=0) f	w[7:0]={c,f[7:1]}	1	0011_110f_ffff_ffff	c,z
RRF(d=1) f	f[7:0]={c,f[7:1]}	1	0011_111f_ffff_ffff	c,z
SWAPF(d=0) f	w[7:0]={f[3:0],f[7:4]}	1	0100_110f_ffff_ffff	none
SWAPF(d=1) f	f[7:0]={f[3:0],f[7:4]}	1	0100_111f_ffff_ffff	none
立即数操作(9)				
ADDLW k	w=k+w	1	0100_0100_kkkk_kkkk	c,dc,z
SUBLW k	w=k-w	1	0101_1000_kkkk_kkkk	c,dc,z
ANDLW k	w=k&w	1	0110_1000_kkkk_kkkk	z
IORLW k	w=k|w	1	0110_1100_kkkk_kkkk	z
XORLW k	w=k^w	1	0111_0000_kkkk_kkkk	z
MOVLW k	w=k	1	0100_0000_kkkk_kkkk	none
SETDP k	dp[1:0]=k	1	0100_0001_kkkk_kkkk	none
MOVDPW	w=dp	1	0100_0010_dpdp_dpdp	none
MOVI2CW	w=i2c_dbuf[7:0]	1	0100_0011_i2cd_ i2cd	none
控制操作(9)				
RETLW k	RETURN and w=k	2	0100_1000_kkkk_kkkk	none
CALL k	Push PC+1 and goto k	2	100k_kkkk_kkkk_kkkk	none
GOTO k	PC=k	2	101k_kkkk_kkkk_kkkk	none
RETFIE	Pop PC and gie=1	2	0000_0000_0000_0010	none
RETURN	Pop PC	2	0000_0000_0000_0011	none
HALT	Stop uP clock	1	0000_0000_0000_0101	none
SLEEP	Stop OSC	1	0000_0000_0000_0100	PD
ADDPCW	PC=PC+1+{6{w(7)},w[6:0]}	1	0000_0000_0000_1000	none

CLRWDT	Clear watch dog timer	1	0000_0000_0000_0110	none
寄存器 f 位操作(4)				
BCF f,b	f[b]=0	1	1100_bbbf_ffff_ffff	none
BSF f,b	f[b]=1	1	1101_bbbf_ffff_ffff	none
BTFSC f,b	Skip if f[b]=0	1,2	1110_bbbf_ffff_ffff	none
BTFSS f,b	Skip if f[b]=1	1,2	1111_bbbf_ffff_ffff	none

- f: 内存地址（000h-1FFh）
- w: 工作寄存器
- k: 文字字段，常数或者标签
- d: 目的寄存器选择：d=0 将结果存入 w，d=1 将结果存取 f
- b: 位选择（0-7）
- pc: 程序指针

1.7 指令集描述

NOP	**NOP**
指令	NOP
操作	No Operation
影响状态	None
描述	空操作，用于实现一个指令周期的延时

CLRF	**Clear f**
指令	CLRF f
操作	0→M(f)
影响状态	1→Z
描述	M(f)清零

ADDWF	**ADD W to f**
指令	ADDWF f, d
操作	W+M(f)→d(destination)
影响状态	DC,C,Z
描述	将 W 寄存器和 M(f)相加，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

INCF	**Increment f**
指令	INCF f, d
操作	M(f)+1→d(destination)
影响状态	Z
描述	将 M(f)加 1，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

INCFSZ	**Increment f, skip if zero**
指令	INCFSZ f, d
操作	M(f)+1→d(destination), skip if result is zero

影响状态	None
描述	将 M(f)加 1，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器；若结果为 0，则下条指令将被自动被 NOP 指令取代，使得该指令执行两个指令周期，反之，执行下条指令，则该指令只运行一个指令周期

DECF	**Decrement f**
指令	DECF f, d
操作	M(f)-1→d(destination)
影响状态	Z
描述	将 M(f)减 1，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

DECFSZ	**Decrement f, skip if zero**
指令	DECFSZ f, d
操作	M(f)-1→d(destination), skip if zero
影响状态	None
描述	将 M(f)减 1，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器；若结果为 0，则下条指令将被自动被 NOP 指令取代，使得该指令执行两个指令周期，反之，执行下条指令，则该指令只运行一个指令周期

SUBWF	**Subtract W from f**
指令	SUBWF f, d
操作	M(f)+NOT(W)+1→d(destination)
影响状态	DC,C,Z
描述	将 M(f)和 W 寄存器相减，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

COMF	**Complement f**
指令	COMF f, d
操作	NOT(M(f))→d(destination)
影响状态	Z
描述	M(f) 取反，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

MOVWF	**Move W to f**
指令	MOVWF f
操作	W→M(f)
影响状态	None
描述	将 W 寄存器的内容写入 M(f)寄存器

MOVFW	Move f to W
指令	MOVFW f
操作	M(f) → W
影响状态	None
描述	将 M(f)的内容写入 W 寄存器

ADDWFC	Add W, f and Carry
指令	ADDWFC f, d
操作	W + M(f) + C→d(destination)
影响状态	DC,C,Z
描述	将 W 寄存器和 M(f)以及进位位 C 相加，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

ANDWF	And W, f
指令	ANDWF f, d
操作	W AND M(f) →d(destination)
影响状态	Z
描述	将 W 寄存器和 M(f)相与，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

IORWF	Inclusive OR W and f
指令	IORWF f, d
操作	W OR M(f) →d(destination)
影响状态	Z
描述	将 W 寄存器和 M(f)相或，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

XORWF	Exclusive OR W and f
指令	XORWF f, d
操作	W XOR M(f) →d(destination)
影响状态	Z
描述	将 W 寄存器和 M(f)异或，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

RLF	Rotate left M(f) through Carry
指令	RLF f, d
操作	M(f) [6:0],C→d(destination)，M(f) [7] → C
影响状态	C,Z
描述	将 M(f) 左移一位，最低位填入进位 C 位，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

SUBWFC	Subtract W and Carry from f
指令	SUBWFC f, d
操作	W + NOT(M(f))+C→d(destination)
影响状态	DC,C,Z
描述	将 W 寄存器和 M(f)以及进位 C 相减，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

RRF	Rotate right M(f) through Carry
指令	RRF f, d
操作	C,M(f) [7:1] →d(destination), M(f) [0] → C
影响状态	C,Z
描述	将 M(f) 右移一位，最高位填入进位 C 位，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

SWAPF	Exchange M(f) [7:4] and M(f)[3:0]
指令	RRF f, d
操作	{M(f) [3:0],M(f) [7:4]} →d(destination)
影响状态	none
描述	将 M(f) 高低 4 位交换，若 d=0，结果存入 W 寄存器，若 d=1，结果存入 M(f)寄存器

ADDLW	ADD literal to W
指令	ADDLW k
操作	W + k→W
影响状态	DC,C,Z
描述	将 8bit 立即数 k 和 W 寄存器的内容相加，结果写入 W 寄存器

SUBLW	Subtract literal from W
指令	SUBLW k
操作	k + NOT(W) + 1→W
影响状态	DC,C,Z
描述	将 8bit 立即数 k 和 W 寄存器的内容相减，结果写入 W 寄存器

ANDLW	AND literal from W
指令	ANDLW k
操作	W AND k→W
影响状态	Z
描述	将 8bit 立即数 k 和 W 寄存器的内容相与，结果写入 W 寄存器

IORLW	Inclusive OR literal from W
指令	IORLW k
操作	W OR k→W
影响状态	Z
描述	将 8bit 立即数 k 和 W 寄存器的内容相或，结果写入 W 寄存器

XORLW	Exclusive OR literal from W
指令	XORLW k
操作	W XOR k→W
影响状态	Z
描述	将 8bit 立即数 k 和 W 寄存器的内容异或，结果写入 W 寄存器

MOVLW	Move literal to W
指令	MOVLW k
操作	k→W
影响状态	None
描述	将 8bit 立即数 k 写入 W 寄存器

SETDP	Move literal k to dp
指令	SETDP k
操作	k→dp
影响状态	None
描述	将立即数 k 写入 dp 寄存器。Dp 是两比特数据指针，dp[0]与 IND0 配合、dp[1]与 IND1 配合使用进行间接寻址，若间接寻址的范围超过 FFH 时，对应的 dp 指针位必须设置为 1 来作为 RAM 的最高位地址，从而实现 9bit RAM 地址的间接寻址，反之则将 dp 设为 0

MOVDPW	Move dp to W
指令	MOVDPW
操作	dp→W
影响状态	None
描述	将数据指针 dp 写入 W 寄存器

MOVI2CW	Move I2C_DBUF to W
指令	MOVI2CW
操作	I2c_dbuf[7:0]→W
影响状态	None
描述	将硬件 I2C 缓存数据写入 W 寄存器

RETLW	Return and move literal to W
指令	RETLW k
操作	k→W　[Top Stack]→PC　Pop Stack
影响状态	None
描述	将立即数 k 写入 W 寄存器，且用堆栈顶部数据装载程序指针，然后出栈

CALL	Subroutine CALL
指令	CALL k
操作	Push Stack　PC + 1→[Top Stack]　k→PC
影响状态	None
描述	子程序调用，首先，将返回程序指针 PC+1 压入堆栈寄存器，然后将立即数 k 装载程序指针

GOTO	Unconditional Branch
指令	GOTO k
操作	k→PC
影响状态	None
描述	立即数 k 装载程序指针

RETFIE	Return from Interrupt
指令	RETFIE
操作	[Top Stack]→PC　Pop Stack　1→GIE
影响状态	None
描述	堆栈顶部数据装载程序指针，然后出栈，并且置位全局中断使能（GIE）

ADDPCW	Add W to Program Counter
指令	ADDPCW
操作	PC + 1 + W→PC
影响状态	None
描述	当工作寄存器 W 小于 128 时，程序指针 PC=PC+1+W

HALT	Stop CPU Core Clock
指令	HALT
操作	CPU 时钟停止
影响状态	None
描述	CPU 时钟停止，主振荡器工作，内部定时中断或者外部中断以及三角扫描按键可以恢复 CPU 工作

SLEEP	Oscillator Stop
指令	SLEEP
操作	内部振荡器停止工作
影响状态	None
描述	内部振荡器停止工作，CPU 处于休眠，外部中断或者三角扫描按键可以唤醒 CPU

CLRWDT	Clear Watch Dog Timer Counter
指令	CLRWDT
操作	看门狗计数器被复位
影响状态	None
描述	复位看门狗计数器

BSF	Bit Set f
指令	BSF f, b
操作	1→ M(f)[b]
影响状态	None
描述	M(f) 中‘b’ 位被置高

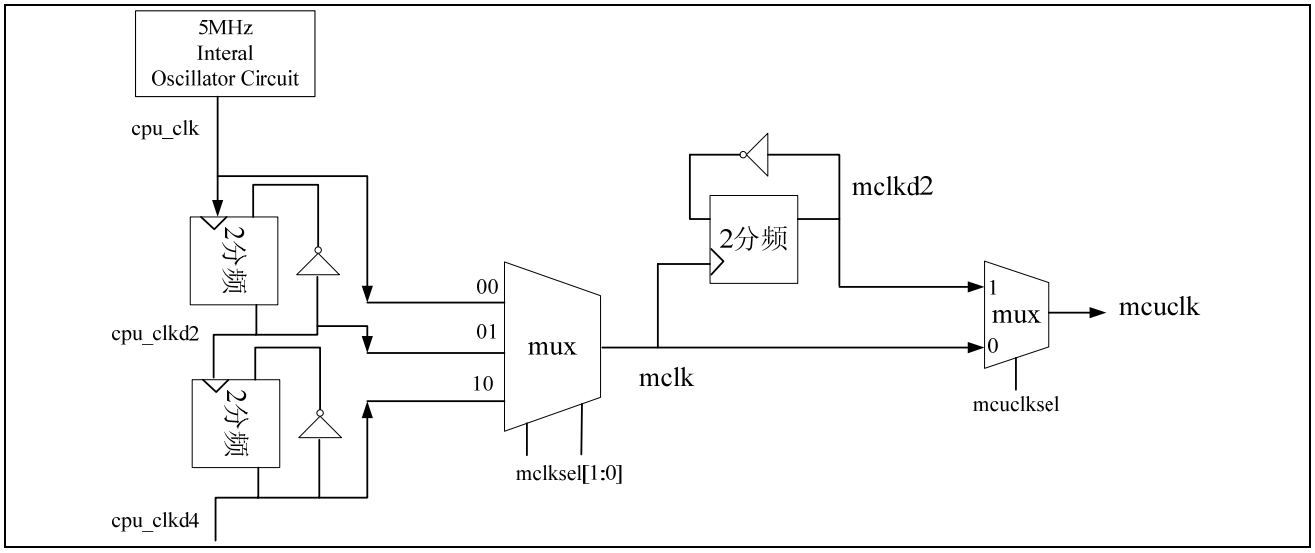
BCF	Bit Clear f
指令	BCF f, b
操作	0→ M(f)[b]
影响状态	None
描述	M(f) 中‘b’ 位被清零

BTFSC	Bit Test Skip if Clear
指令	BTFSC f, b
操作	Skip if M(f)[b]=0
影响状态	None
描述	若 M(f) 中‘b’ 位为 0，下条指令将被自动被 NOP 指令取代，使得该指令执行两个指令周期，反之，执行下条指令，则该指令只运行一个指令周期

BTFSS	Bit Test Skip if Set
指令	BTFSS f, b
操作	Skip if M(f)[b]=1
影响状态	None
描述	若 M(f) 中‘b’ 位为 1，下条指令将被自动被 NOP 指令取代，使得该指令执行两个指令周期，反之，执行下条指令，则该指令只运行一个指令周期

2 时钟系统

地址	名称	内容						
008H	MCK			timeclksel	mclksel[1:0]		mcuclksel	



通过设置 mclksel[1:0]，mcuclksel 来设置 mcuclk 频率以及指令周期，如下表所示

mclksel[1:0]	mcuclksel	mcuclk Source	mcuclk Frequency	指令周期
00	0	cpu_clk	5MHz	800ns
01	0	cpu_clkd2	2.5MHz	1600ns
10	0	cpu_clkd4	1.25MHz	3200ns
00	1	cpu_clk/2	2.5MHz	1600ns
01	1	cpu_clkd2/2	1.25MHz	3200ns
10	1	cpu_clkd4/2	0.625MHz	6400ns

2.1 晶振状态

输入	晶振状态
sleep	cpu_clk
1	Disable
0	Enable
0	Disable

2.2 定时器主时钟

timeclksel	timer_clk
1	mclk
0	cpu_clk

3 I/O 端口

3.1 PT1/2 口

3.1.1 端口功能选择寄存器

地址	名称	内容	
019H	PT1SEL	pt1sel[7:3]	
01AH	PT2SEL	pt2sel[7:0]	

- PT1/2SEL 是端口功能选择位
- pt1/2sel[n]=0，则对应的端口为三角按键扫描口，pt1/2sel[n]=1，则对应端口为通用 IO 口

3.1.2 通用 IO 中断屏蔽寄存器

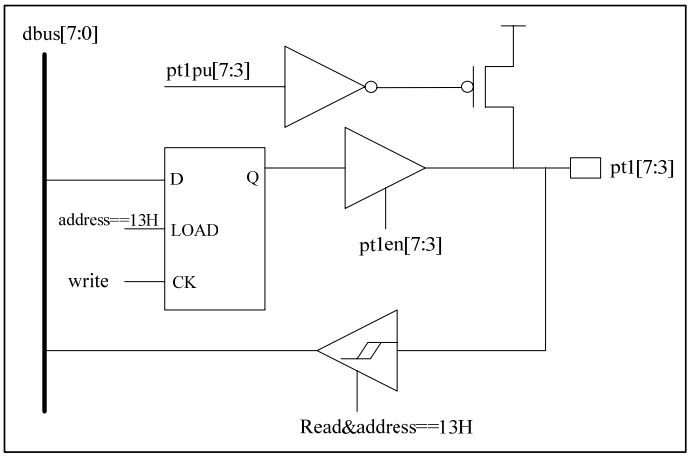
地址	名称	内容							
006H	1NTF				tmaif	tmbif			e0if
007H	1NTE	gie			tmaie	tmbie			e0ie
01BH	PT1MR	pt1mr[7:3]							
01CH	PT2MR	pt2mr[7:0]							

- 当端口为通用 IO 口时，每个端口可以作为外部中断口，下降沿产生中断信号
- 所有端口共享一个中断入口地址 0012H
- PT1/2MR 是外部中断屏蔽寄存器，pt1/2mr[n]=1，屏蔽外部中断信号，pt1/2mr[n]=0，不屏蔽外部中断信号
- 当 gie 和 e0ie 设置为 1，pt1/2mr[n]为 0，pt1/2sel=1，若 pt1/2[n]端口出现一个下降沿，则系统会响应这个中断信号，e0if 会变成 1。

3.1.3 PT1 口

地址	名称	内容	
013H	PT1	pt1[7:3]	
014H	PT1EN	pt1en[7:3]	
015H	PT1PU	pt1pu[7:3]	

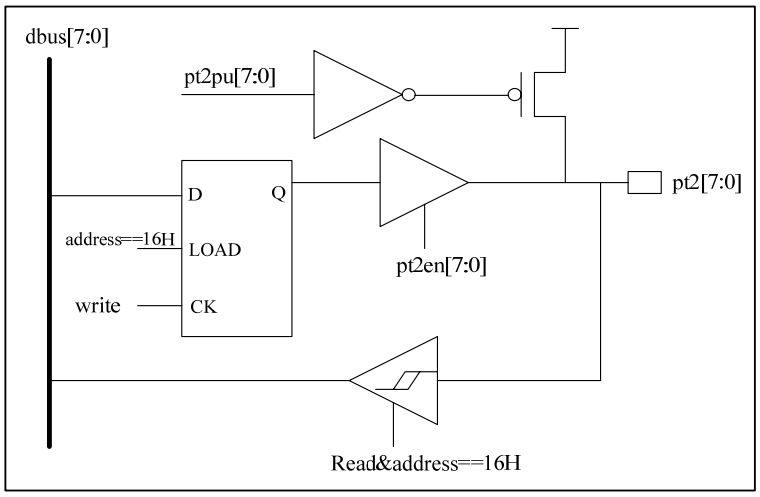
- PT1 是带上拉电阻使能控制的双向 I/O
- PT1 处理 pt1[5]端口为开漏输出，其他全部为 CMOS 输出
- pt1en[n]=0，pt1[n]为输入口，pt1en[n]=1，pt1[n]为输出口，系统复位值为 0
- pt1 口为施密特输出口



3.1.4 PT2 口

地址	名称	内容
016H	PT2	pt2[7:0]
017H	PT2EN	pt2en[7:0]
018H	PT2PU	pt2pu[7:0]

- PT2 是带上拉电阻使能控制的双向 I/O
- pt2en[n]=0，pt2[n]为输入口，pt2en[n]=1，pt2[n]为输出口，系统复位值为 0
- pt2 口为施密特输出口



4. 定时器

地址	名称	内容							
006H	INTF				tmaif	tmbif	capif		
007H	INTE	gie			tmaie	tmbie	capie		
009H	TSETAL	tseta[7:0]							
00AH	TSETAH	tseta[15:8]							
00BH	PWMCON	pwm_sel	duty_sel[1:0]		tcouta_dir	pwm_oen	pwm_port_sel[2:0]		
00CH	TCOUTAL	tcouta[7:0](R)							
00DH	TCOUTAH	tcouta[15:8](R)							
00EH	TCCONA	tcrsta	cap_sel	cap_deben	cap_phaen	tcena	cclka_sel[2:0]		
00FH	TSETB	tsetb[7:0]							
010H	TCOUTB	tcoutb[7:0](R)							
011H	TCCONB	tcrstb				tcenb	cclkb_sel[2:0]		
01EH	RMTCTR	sens_curt[1:0]	pwm_curt[2]		rx_en	tx_en	pwm_curt[1:0]		rmt_in(R)

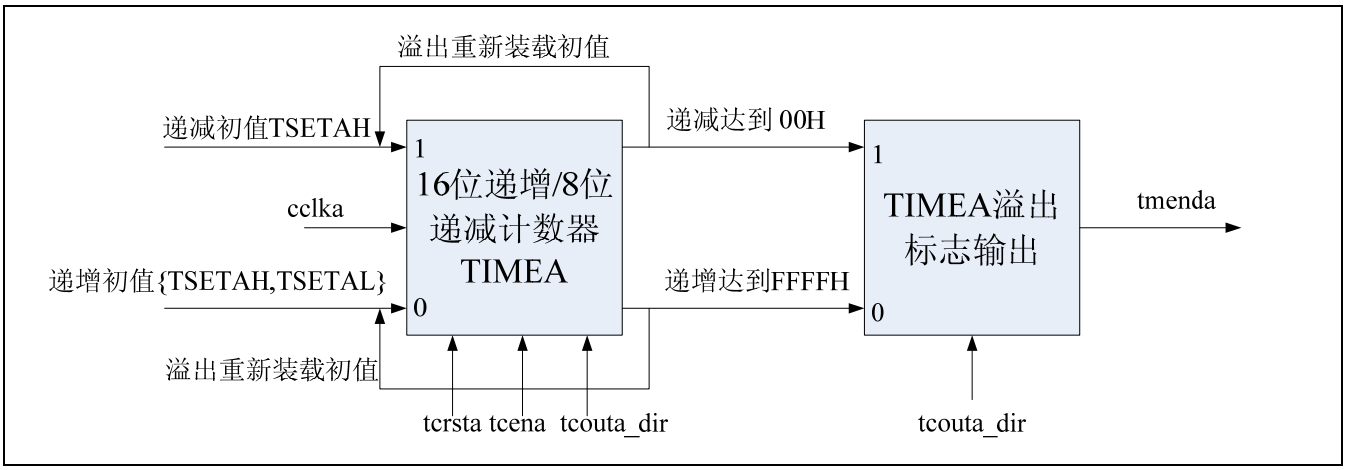
4.1 TIEMA 定时器

cclka_sel[2:0]	cclka
000	rmt_in
001	~rmt_in
010	timer_clk/4

011	timer_clk/8
100	timer_clk/16
101	timer_clk/32
110	timer_clk/2
111	timer_clk

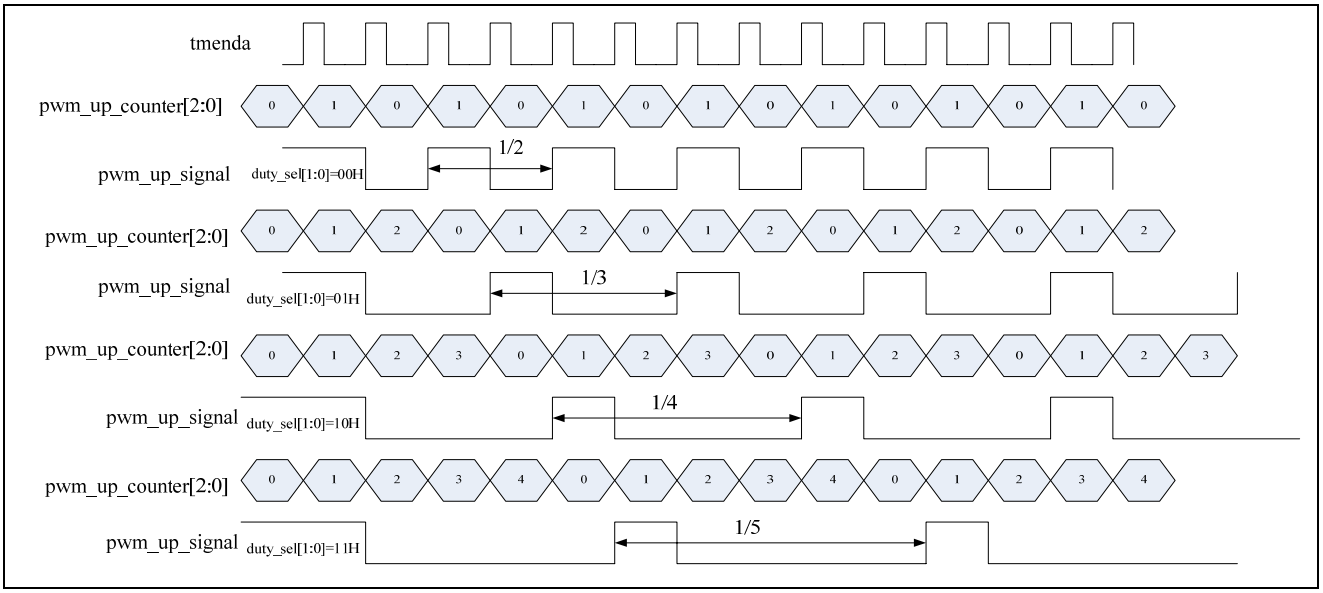
注：rmt_in 是 RMT 端口的输出信号，~rmt_in 是 rmt_in 的反相信号

TIMEA 定时器框图



- 向 TCCONA 寄存器的第七位写入“0”，cpu 就会产生一个低电平脉冲给 tcrsta 来复位 TIMEA，读 tcrsta 的值为“1”
- 当 tcena=1，TIMEA 计数器开始工作。tcena=0，计数器停止工作
- 当 INTE 寄存器 gie 和 tmaie 设置为 1，则 tmenda 产生的脉冲信号将会置位中断 INTF 寄存器标志位 tmaif，中断入口地址为 0006H
- touta_dir=1，TIMEA 为 8 位递减计数器，初值以及溢出都装载寄存器 TSETAH 值，touta_dir=0，TIMEA 为 16 位递增计数器，初值以及溢出都装载寄存器{TSETAH，TSETAL}值

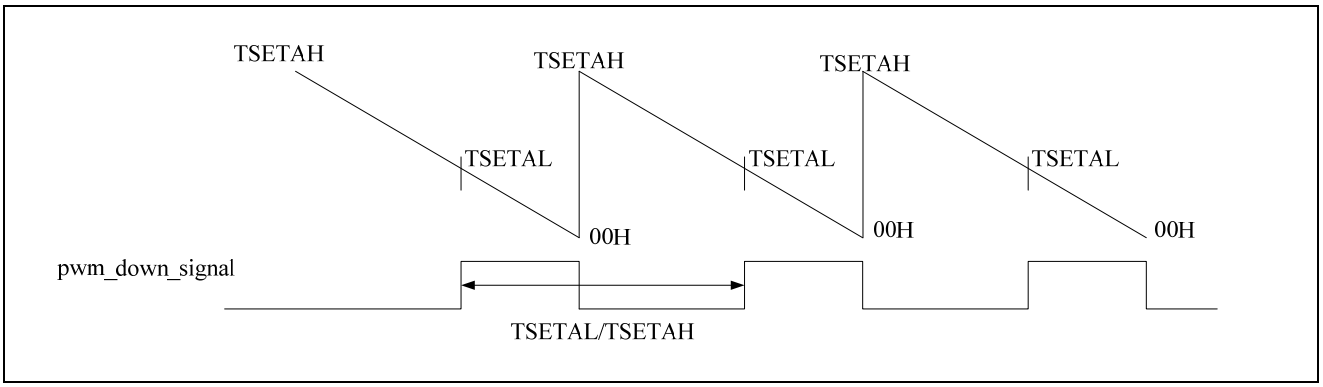
TIMEA 产生的 PWM 信号（16 位递增计数器产生，固定占空比）



- 此时计数器为 16 位递增计数器，touta_dir=0

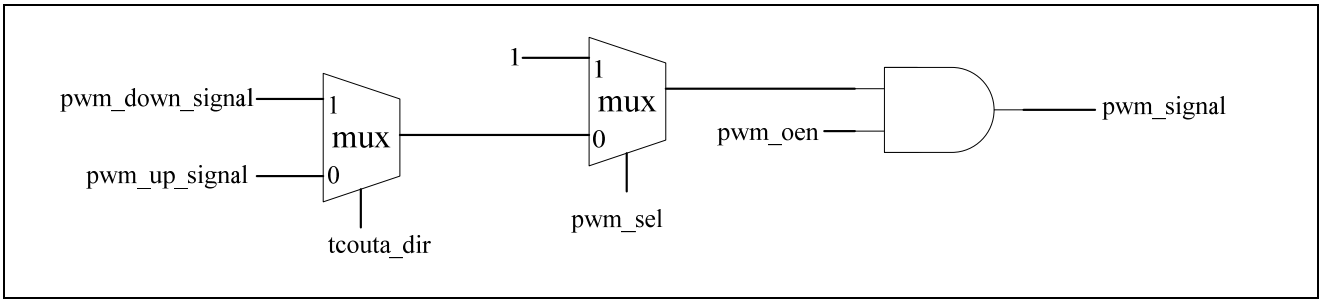
- 占空比（1/2，1/3，1/4，1/5）由 duty_sel 选择，其中高电平的宽度就是 tmenda 的周期，即两个溢出标志 tmenda 间隔时间

TIMEA 产生的 PWM 信号（8 位递减计数器产生，变占空比）



- 此时计数器为 8 位递减计数器，touta_dir=1
- 占空比有寄存器 TSETAH，TSETAL 的值决定，为 TSETL/TSETH

PWM 信号输出结构图



- pwm_signal 为控制发射管的控制信号
- 当 pwm_oen=1，pwm_signal 允许输出，为 0 则禁止输出
- 当 pwm_sel=1 且 pwm_oen=1，pwm_signal 输出高电平
- 当 pwm_sel=0 且 pwm_oen=1，pwm_signal 输出固定占空比或变占空比信号
- 当 pwm_sel=0 且 pwm_oen=1 且 tcouta_dir=0，pwm_signal 输出固定占空比信号
- 当 pwm_sel=0 且 pwm_oen=1 且 tcouta_dir=1，pwm_signal 输出变占空比信号

PWM 信号输出端口选择

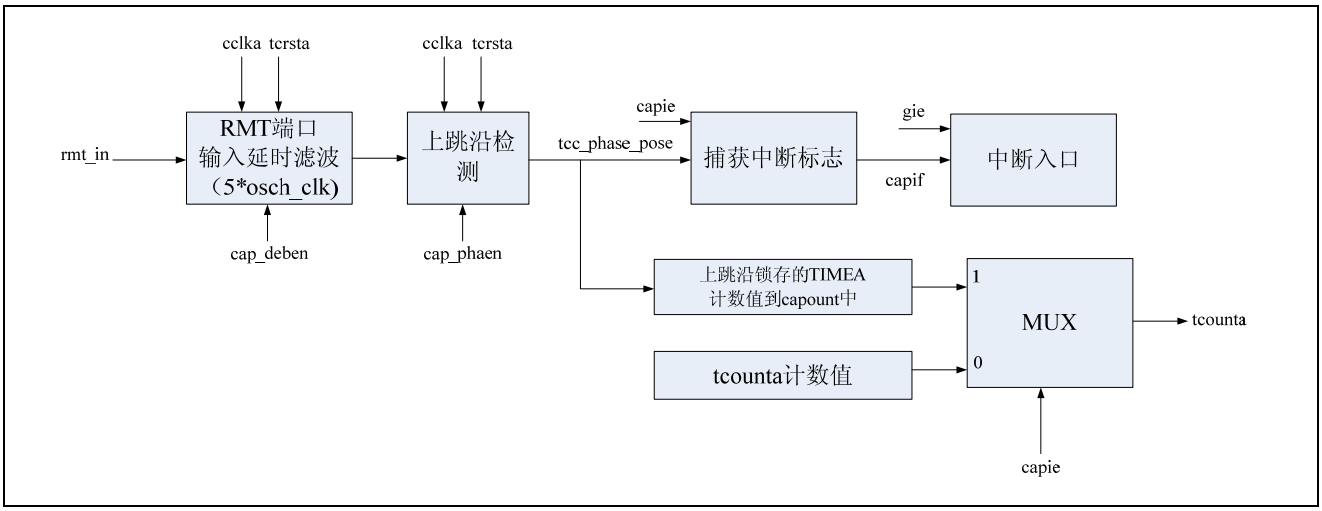
- pwm_signal 除了可以作为发射管的控制信号，还可以作为普通 PWM 输出
- 当 pwm_port_sel[0]=1，且 KS3/PT2_0 端口作为普通 IO 输出口时，pwm_signal 信号可以从此端口输出，反之不输出
- 当 pwm_port_sel[1]=1，且 KS4/PT2_1 端口作为普通 IO 输出口时，pwm_signal 信号可以从此端口输出，反之不输出
- 当 pwm_port_sel[2]=1，且 KS10/PT2_7 端口作为普通 IO 输出口时，pwm_signal 信号可以从此端口输出，反之不输出

RMT 端口周期捕获步骤:

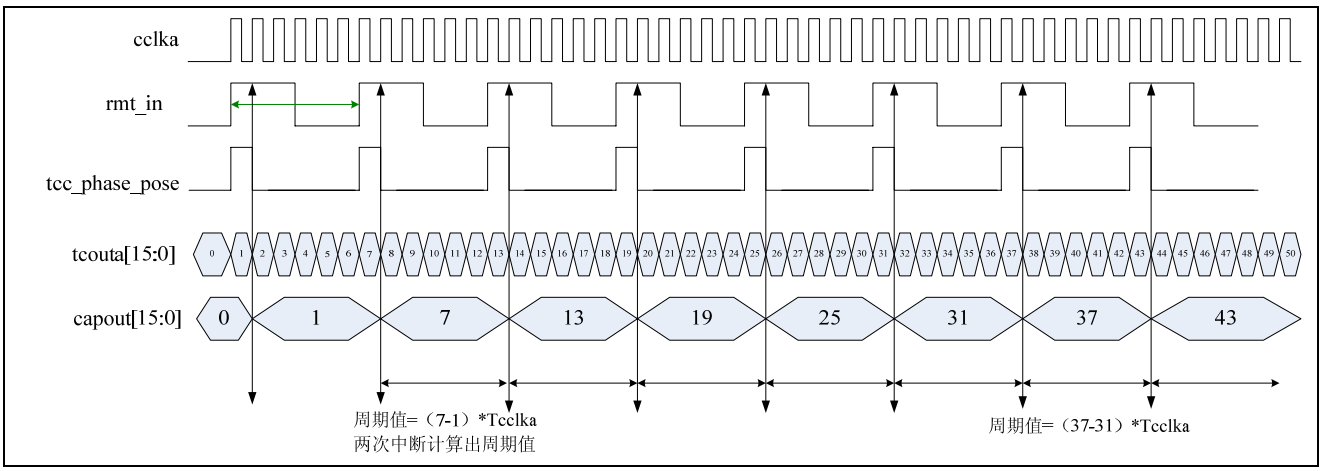
- 通过设置 TCCONA 寄存器中 cclka_sel[2:0] 来选择 TIMEA 定时的计数时钟;
- 通过设置 TCCONA 寄存器中的 tcena 位来使能 TIMEA 定时器

- 通过设置 TCCONA 寄存器中的 cap_sel 选择 RMT 端口作为捕获波形输出口，cap_sel=1，选择 PT1_4 端口作为捕获波形输入口，cap_sel=0，选择 RMT 端口作为捕获波形输入口
- 通过设置 TCCONA 寄存器中的 cap_deben 使能 RMT 端口波形滤波电路，cap_deben=1，启动滤波电路，小于 1us 的信号将被滤掉，cap_deben=0，不启动滤波电路
- 通过设置 INTE 中的 gie 和 capie 来打开捕获中断使能（捕获中断的和 TIMEA 定时溢出中断入口地址相同，为 0006H）
- 等待 RMT 端口波形输入
- 捕获中断发生，保存 TCOUTAH/L 寄存器的值到 RAM 中
- 清除捕获中断标志位并返回，等待下一次捕获中断发生
- 第二次中断发生后，保存当前的 TCOUTAH/L 寄存器的值到 RAM 中，将第二次的 RAM 值减去第一次 RAM 值即是 RAM 端口波形的周期值
- 为了提高周期捕获的准确性，程序可以多采样几组周期值来求平均值
- cap_phaen 是将 rmt_in 信号反相使能信号，1—反相，0—同相

RMT 端口周期捕获示意图



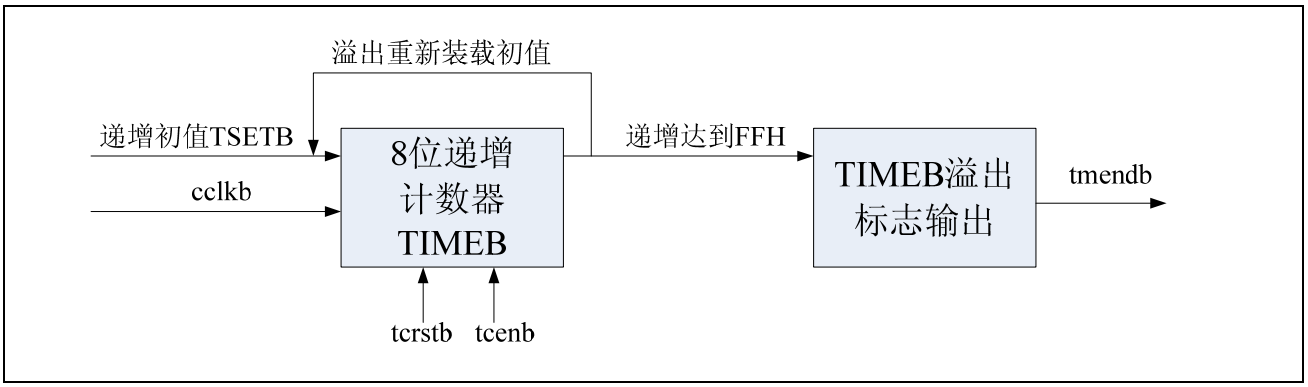
RMT 端口周期捕获时序图



4.2 TIEMB 定时器

cclkb_sel[2:0]	cclkb
000	timer_ clk/4
001	timer_ clk/8
010	timer_ clk/16
011	timer_ clk/32
100	timer_ clk/64
101	timer_ clk/128
110	timer_ clk/2
111	timer_ clk

TIMEB 定时器框图



- 向 TCCONB 寄存器的第七位写入“0”，cpu 就会产生一个低电平脉冲给 tcrstb 来复位 TIMEB，读 tcrstb 的值为“1”。
- 当 tcenb=1，TIMEB 计数器开始工作。tcenb=0，计数器停止工作
- 当 INTE 寄存器 gie 和 tmbie 设置为 1，则 tmendb 产生的脉冲信号将会置位中断 INTF 寄存器标志位 tmbif，中断入口地址为 000CH

5. 硬件 I2C 模块

地址	名称	内容					
005H	W	w[7:0]					
01FH	I2CCON	i2cen	i2c_ready	i2c_sto(R)	i2c_rw(R)	i2c_adr_sel	pt1od[5:3]

I2CCON 控制寄存器:

Bit7: i2cen→I2C 模块使能位，1 使能，0 禁止，选择 PT2_0，PT1_3 作为 I2C 的 SCL 和 SDA 通讯口。

Bit6: i2c_ready→在主机写模式下，数据完成一个字节接收标志位，1 表示接收完成，当执行 MOVI2CW 指令时自动清零该位。

在主机读数据模式下，1 表示从机数据准备好，当主机读完该字节，从机硬件清除该位

Bit5: i2c_sto→I2C 通讯结束信号标志位，1 表示结束，0 表示没有 I2C 结束信号发生，该位只读

Bit4: i2c_rw→I2C 数据读写标志，0 表示 I2C 写（Master to Slave），1 表示 I2C 读（Slave to Master）

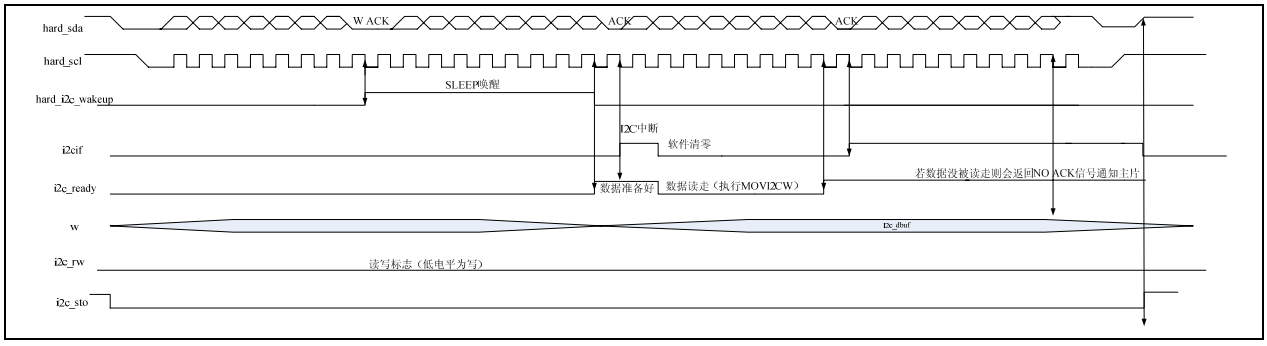
Bit3: i2c_adr_sel→I2C 片地址选择，1 表示写/读片地址为 E2H/E3H，0 表示写/读片地址为 E0H/E1H

Bit2-0: pt1od[5:3] → PT1_5，PT1_4，PT1_3 端口输出开漏设置，1 选择输出开漏，0 禁止开漏

I2C 数据连续写操作步骤:

- 设置 PT1SEL 和 PT2SEL 寄存器将 PT1_3 和 PT2_0 切换到普通 IO 口模式
- 设置 I2CCON 寄存器中的 i2cen 位使能 PT2_0，PT1_3 端口作为 I2C 通讯端口
- 程序进入 SLEEP 模式
- Master 发送一组 I2C 通讯数据，slave 在 I2C 片地址比对正确后才被唤醒
- 设置数据存储的起始 RAM 地址，设置 INTE 寄存器中的 i2cie 和 gie 位开启 i2c 数据传输中断
- 查询 I2CCON 寄存器中的 i2c_sto 位是否为 1，若不为 1 则等待，若为 1 则进入数据处理程序
- 当完成第一个数据字节传输时会产生中断,中断程序中首先查询 I2CCON 寄存器中的 i2c_rw 为是否为 1，若为 0 则进入 I2C 写子程序，若为 1 则进入读子程序。
- 写子程序中首先判读 I2CCON 寄存器中的 i2c_ready 是否为 1，若为 0 则清中断标志退出中断
- 第二个字节完成接收是也产生中断,此时重复判断片地址和 i2c_ready 为 1,若都为 1 则执行 MOVI2CW 指令将 I2C 数据存入到 W 中，随后将该数据暂存 RAM 中。
- 第三个到第 n 个字节都是采用上一步的流程直到 I2C 结束信号发生
- I2C 结束信号发生，则清除中断使能位，进行数据处理，处理完成以后，进入 SLEEP 状态

I2C 数据连续写操作时序图

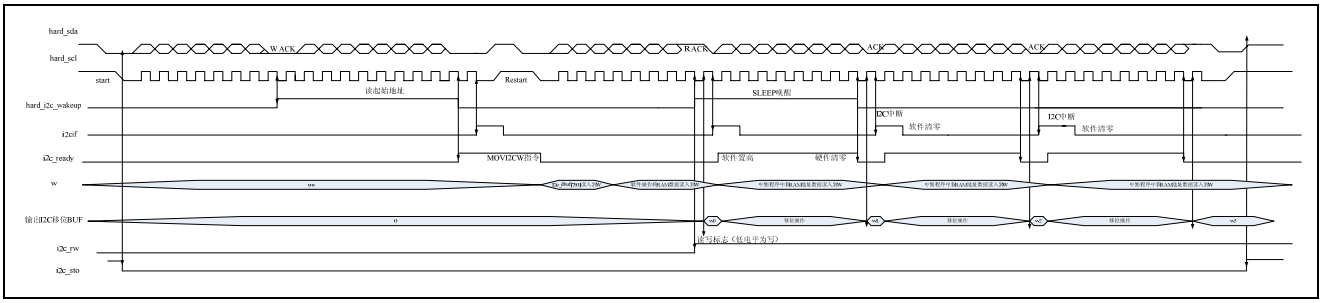


I2C 数据连续读操作步骤:

- 设置 PT1SEL 和 PT2SEL 寄存器将 PT1_3 和 PT2_0 切换到普通 IO 口模式
- 设置 I2CCON 寄存器中的 i2cen 位使能 PT2_0，PT1_3 作为 I2C 的 SCL 和 SDA 通讯口

- 将读出的起始 RAM 的数据放入 W 寄存器中，程序进入 SLEEP 模式
- Master 发送一组 I2C 通讯数据，slave 在 I2C 片地址比对正确后才被唤醒
- 设置 INTE 寄存器中的 i2cie 和 gie 位开启 i2c 数据传输中断
- 查询 I2CCON 寄存器中的 i2c_sto 位是否为 1，若不为 1 则等待，若为 1 则进入数据处理程序
- 当完成第一个数据字节传输时会产生中断，中断程序中首先判断寄存器中的 i2c_rw 为是否为 1，若为 0 则进入 I2C 写子程序，若为 1 则进入读子程序。
- 读子程序直接将第二个 RAM 地址的数据写入 W 中，然后清除中断标志位退出中断。
- 第二个字到第 n 个字节都是采用上一步的流程直到 I2C 结束信号发生
- I2C 结束信号发生，则清除中断使能位，进行数据处理，处理完成以后，进入 SLEEP 状态

I2C 数据连续读操作时序图



6. CPU 复位

CPU 有三种复位信号，它们分别为内部上电复位 RST，低电压复位（LVR）和看门狗定时复位。当复位作用时，CPU 程序指针（PC）被复位为 1FFFh。在复位释放后，CPU 开始工作。下面的表格就是在复位后 CPU 内部寄存器的状态值。

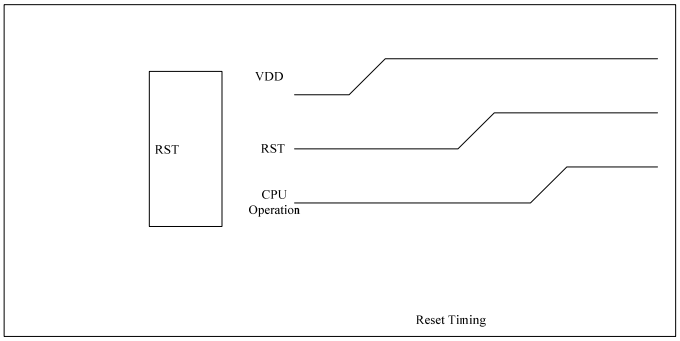
地址	名称	内部复位或者低压复位	看门狗复位
002H	FRS0	00000000	00000000
003H	FRS1	00000000	00000000
004H	STATUS	00000000	00001000
005H	WORK	00000000	00000000
006H	INTF	uuu00uuu	uuu00uuu
007H	INTE	0uu00000	0uu00000
008H	MCK	uu000u0u	uu000u0u
009H	TSETAL	00000000	00000000
00AH	TSETAH	00000000	00000000
00BH	PWMCON	00000u00	00000u00
00CH	TCOUTAL	00000000	00000000
00DH	TCOUTAH	00000000	00000000
00EH	TCCONA	1uuu0000	1uuu0000
00FH	TSETB	00000000	00000000
010H	TCOUTB	00000000	00000000
011H	TCCONB	1uuu0000	1uuu0000
012H	KEY_VALUE	11111111	11111111
013H	PT1	00000uuu	00000uuu

014H	PT1EN	00000uuu	00000uuu
015H	PT1PU	00000uuu	00000uuu
016H	PT2	00000000	00000000
017H	PT2EN	00000000	00000000
018H	PT2PU	00000000	00000000
019H	PT1SEL	00000uuu	00000uuu
01AH	PT2SEL	00000000	00000000
01BH	PT1MR	11111uuu	11111uuu
01CH	PT2MR	11111111	11111111
01DH	WDTCTR	uu010000	uu010000
01EH	RMTCTR	00000uu0	00000uu0
01FH	I2CCON	00000000	00000000

Note: u means unknown or unchanged.

6.1 内部上电复位

当内部复位 RST=0 时，CPU 就进去复位状态。外部 RC 电路如下所示。当 VDD 从低到变成高时，一个复位信号将产生使得 CPU 回到正常的工作模式下。

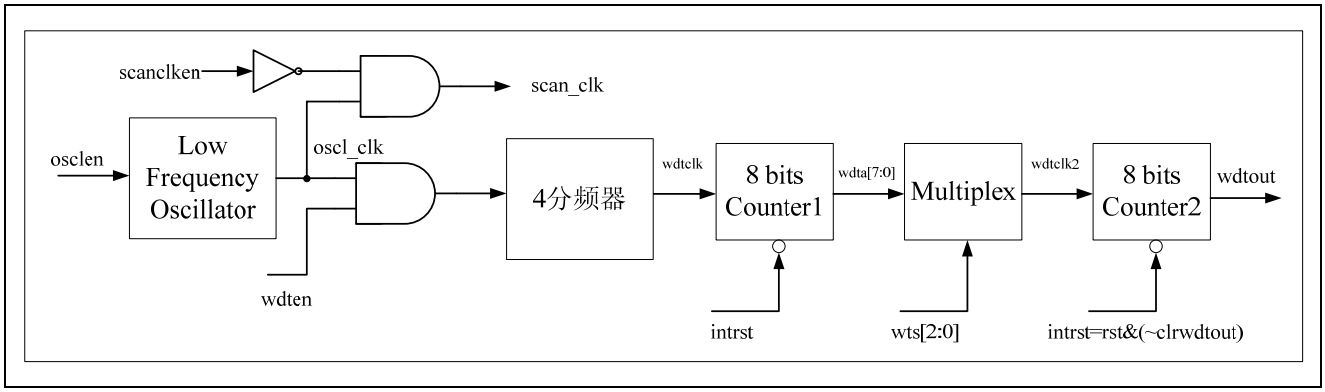


6.2 低电压复位

在异常的电压状态下，为了防止复位信号不能工作从而导致 CPU 工作异常，电路内部包含一个低压复位电路。当 VDD 的电压低于 LVR 时，CPU 就自动进入复位状态。

6.3 看门狗复位

地址	名称	内容						
04h	STATUS				to			
1DH	WDTCTR		osclen	scanclken	wdten	wts[2:0]		



当 osclen 为 1 时，低频振荡器停止工作，为 0 时工作（默认值为低），oscl_clk 频率为 4.5KHz。

当 scanclken 为 1 时，scan_clk=0，为 0 时工作，scan_clk 频率为 4.5KHz，此时钟仅仅作为键扫描时钟。

当启动看门狗定时器(共用低频振荡器，此时 osclen=0)时，wdten 位写 1，反之写 0。当使用 CLRWDT 指令时，WDT 计数器将被复位。看门狗定时器时钟来自内部独立 R/C 振荡器。看门狗输出由 wts 来决定，如下表所示。当看门狗输出发生时，CPU 将复位并设置 to 位为 1。

wdtclk 的频率典型值大约为 1.12kHz		
wdt[2:0]	**Frequency of wdtout**	**Typical Frequency of wdtout**
000	wdtclk/65536	18mHz
001	wdtclk/32768	34mHz
010	wdtclk/16384	68mHz
011	wdtclk/8192	136mHz
100	wdtclk/4096	273mHz
101	wdtclk/2048	550mHz
110	wdtclk/1024	1.09Hz
111	wdtclk/512	2.18Hz

7 暂停和休眠模式

7.1 暂停模式

在 CPU 执行一个 HALT 指令后，CPU 程序计数器（PC）停止计数直到中断命令发生。为了防止中断返回时程序出错，建议在 HALT 指令后加一条 NOP 指令。

推荐在 HALT 之前执行的程序如下所示:

```
NOP
HALT   ；暂停
NOP    ；多加一个 nop
```

7.2 休眠模式

在 CPU 执行 SLEEP 命令后，所有振捣器停止工作直到一个外部中断命令发生或者复位发生。为了防止中断返回时程序出错，建议在 SLEEP 指令后加一条 NOP 指令。

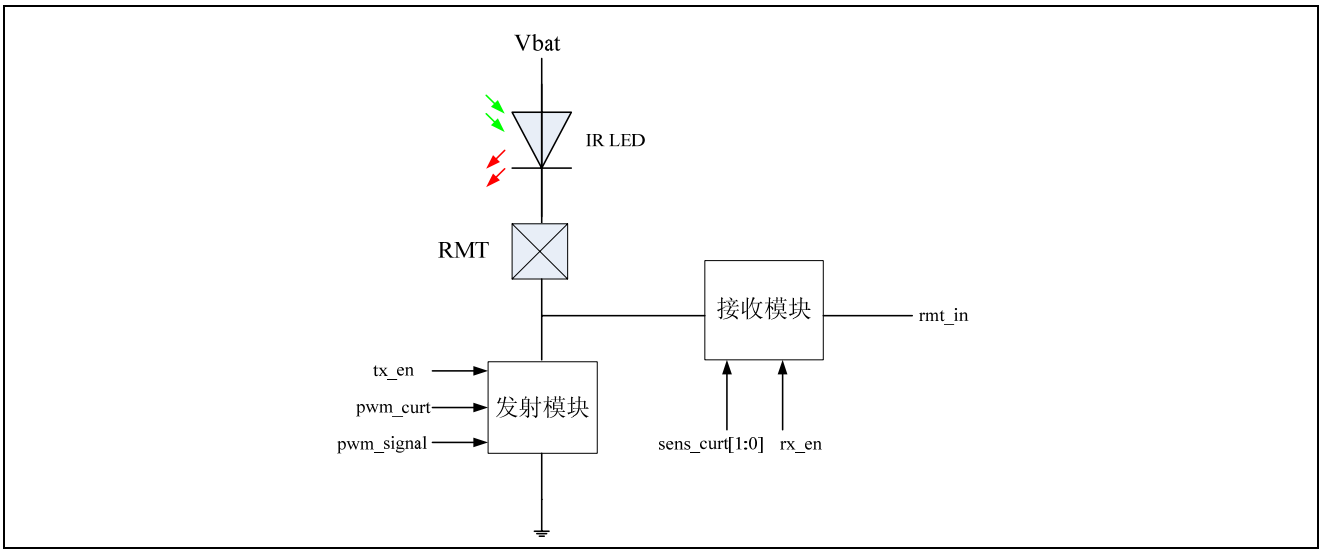
为了确保 CPU 在 SLEEP 模式下功耗最低，建议在执行 SLEEP 指令之前关闭所有的模拟电路，并确保所有的 I/O 都接在 VDD 或者 GND 电平上，若 PT1.7，PT1.6 作为通用 IO 与 EEPORM 进行 I2C 通讯，其他端口作为键扫描口，因通讯线上有上拉电阻，则对于 PT1.7，PT1.6 应该输出高电压。

推荐在 SLEEP 之前执行的程序如下所示:

```
MOVLW 0FFh
MOVWF PT1EN ；输出使能
MOVLW 0C0h
MOVWF PT1 ；PT1.7，PT1.6 输出为高
CLRF INTF ；清除中断标志
CLRF RMTCTR ；关断发射和接收模块
NOP
SLEEP   ；休眠
NOP     ；多加一个 nop
```

8. 接收发射模块

地址	名称	内容					
01EH	RMTCTR	sens_curt[1:0]	pwm_curt[2]	rx_en	tx_en	pwm_curt[1:0]	rmt_in(R)

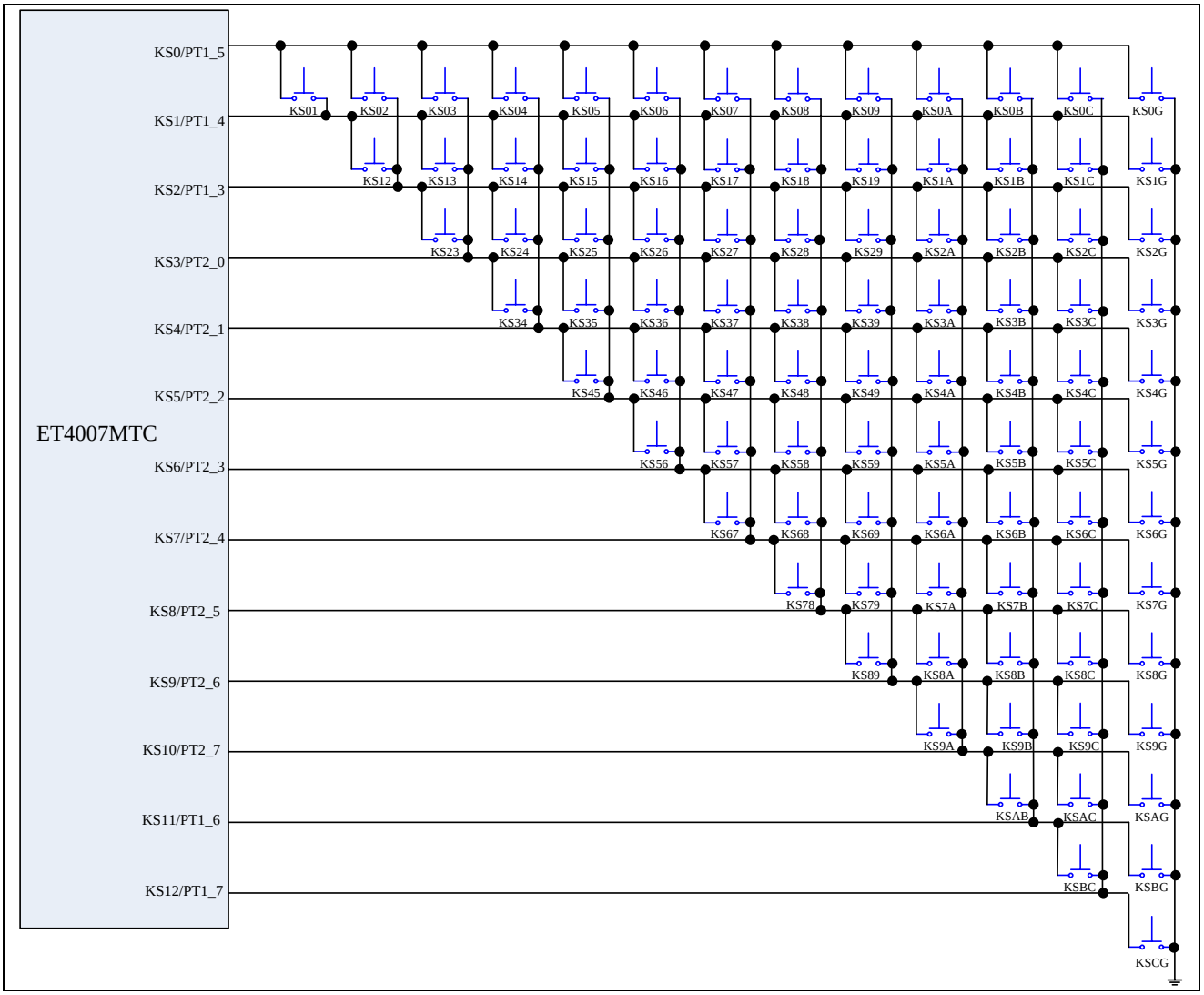


- tx_en 写 1，打开发射模块，0 关闭发射模块
- pwm_curt[2:0]为发射电流选择，
  000→150mA(VoL=0.5V)，001→200mA，010→250mA，011→300mA，
  100→350mA，101→400mA，110→450mA，111→500mA
- pwm_signal 为发射的 pwm 波形，由 TIMEA 模块产生，高电平打开驱动管，低电平关闭
- rx_en 写 1，接收模块使能，0 接收模块关闭
- sens_curt[1:0]为接收电流选择，00—2uA，01—5uA，10—8uA，11—11uA
- rmt_in 为接收波形输入

9. 按键处理模块

地址	名称	内容
012H	KEY_VAULE	key_value[7:0](R)

T-type 型键盘示意图

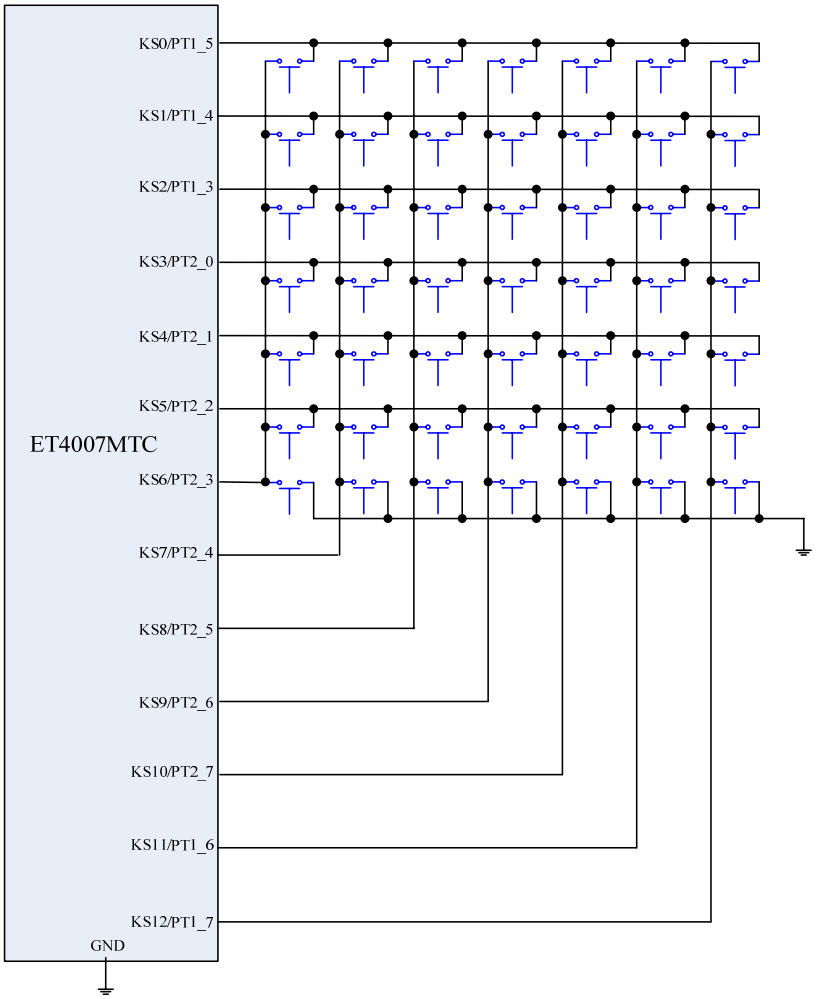


- T-type 型键盘键值由硬件自动采集完成
- 按下一个键，对应的键值自动存储在 012H 的寄存器 key_value 中
- 按下两个或两个以上的键值也会自动存储一个相同的 FFH 键值

T-type 型键盘键值对用表如下:

键名称	键值（hex）	键名称	键值（hex）	键名称	键值（hex）
KS0G	0FH	KS01	01H	KS12	12H
KS1G	1FH	KS02	02H	KS13	13H
KS2G	2FH	KS03	03H	KS14	14H
KS3G	3FH	KS04	04H	KS15	15H
KS4G	4FH	KS05	05H	KS16	16H
KS5G	5FH	KS06	06H	KS17	17H
KS6G	6FH	KS07	07H	KS18	18H
KS7G	7FH	KS08	08H	KS19	19H
KS8G	8FH	KS09	09H	KS1A	1AH
KS9G	9FH	KS0A	0AH	KS1B	1BH
KSAG	AFH	KS0B	0BH	KS1C	1CH
KSBG	BFH	KS0C	0CH		
KSCG	CFH				
键名称	**键值（hex）**	**键名称**	**键值（hex）**	**键名称**	**键值（hex）**
KS23	23H	KS34	34H	KS45	45H
KS24	24H	KS35	35H	KS46	46H
KS25	25H	KS36	36H	KS47	37H
KS26	26H	KS37	37H	KS48	48H
KS27	27H	KS38	38H	KS49	49H
KS28	28H	KS39	39H	KS4A	4AH
KS29	29H	KS3A	3AH	KS4B	4BH
KS2A	2AH	KS3B	3BH	KS4C	4CH
KS2B	2BH	KS3C	3CH		
KS2C	2CH				
键名称	**键值（hex）**	**键名称**	**键值（hex）**	**键名称**	**键值（hex）**
KS56	56H	KS67	67H	KS78	78H
KS57	57H	KS68	68H	KS79	79H
KS58	58H	KS69	69H	KS7A	7AH
KS59	59H	KS6A	6AH	KS7B	7BH
KS5A	5AH	KS6B	6BH	KS7C	7CH
KS5B	5BH	KS6C	6CH		
KS5C	5CH				
键名称	**键值（hex）**	**键名称**	**键值（hex）**	**键名称**	**键值（hex）**
KS89	89H	KS9A	9AH	KSAB	ABH
KS8A	8AH	KS9B	9BH	KSAC	ACH
KS8B	8BH	KS9C	9CH		
KS8C	8CH				
键名称	**键值（hex）**	**其他**		**键值（hex）**	
KSBC	BCH	0、2 及以上按键同时按下		FFH	

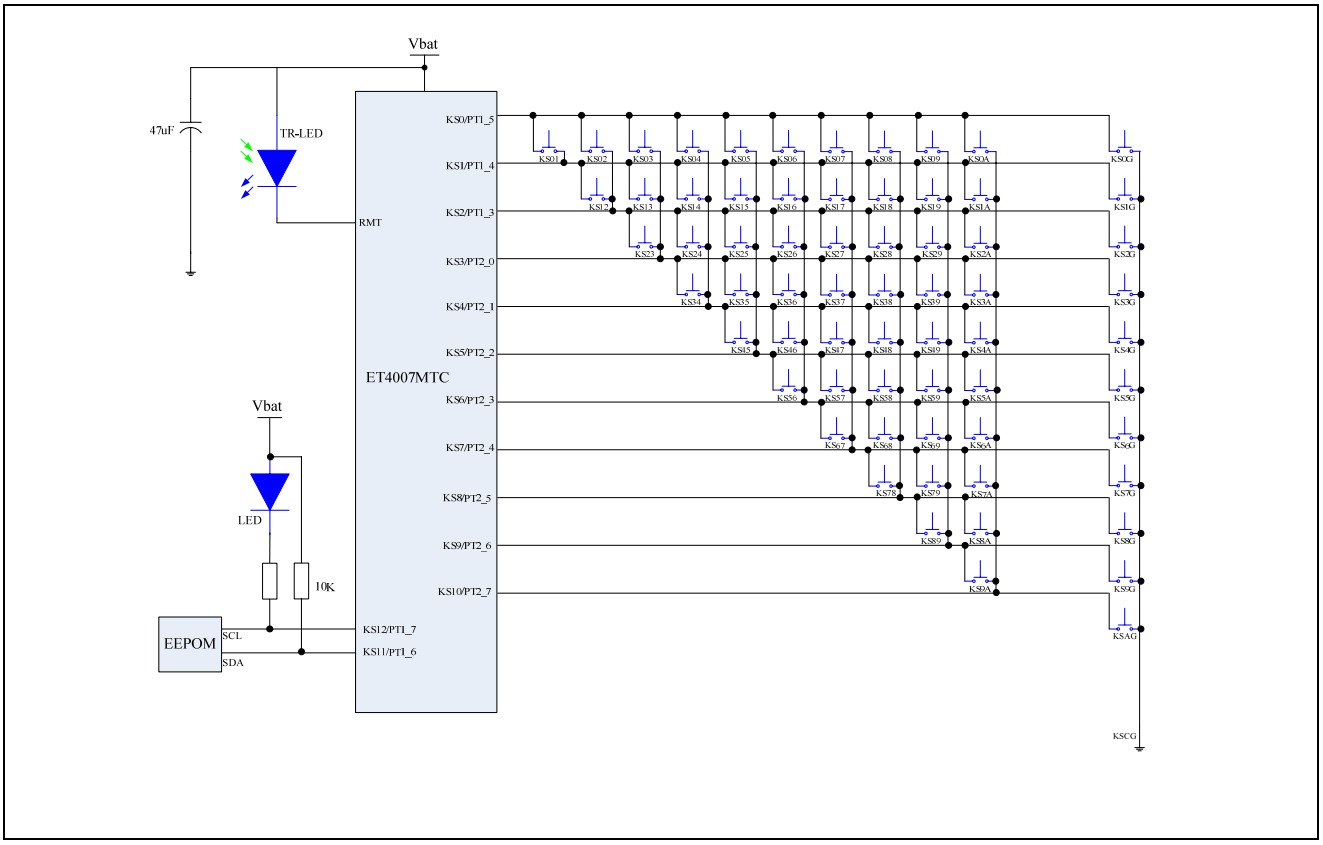
M-type 型键盘示意图



- M-type 型键盘键值由软件采集完成

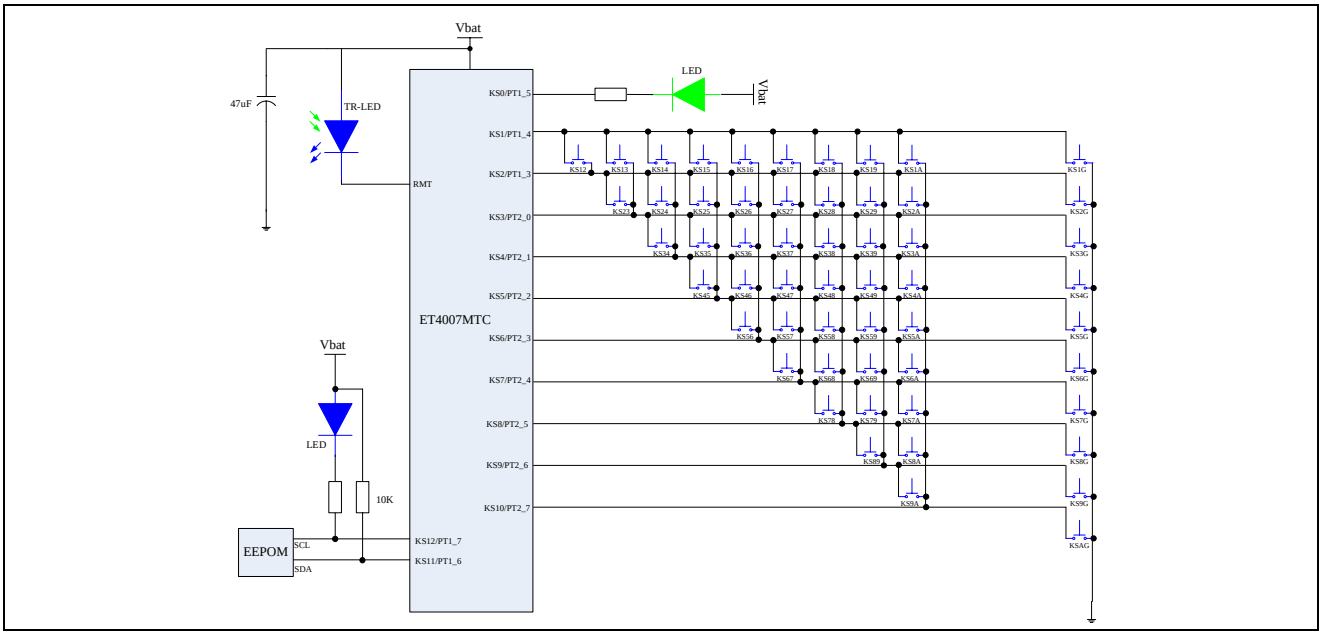
典型应用图

1）学习型遥控器（一个指示 LED，一个接收发射 TR-LED，共 66 个 T-type 型键盘按键）



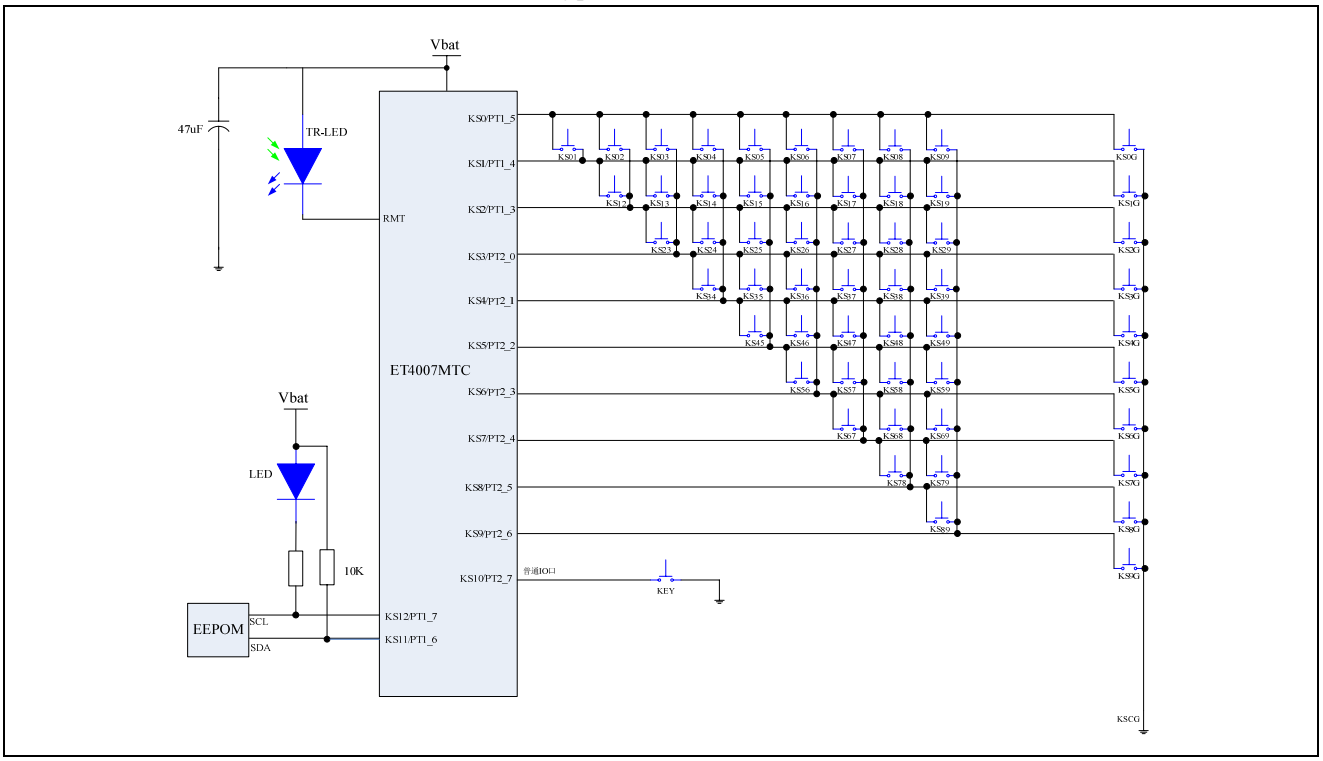
注：此图仅供参考

2）学习型遥控器（两个指示 LED，一个接收发射 TR-LED，共 55 个 T-type 型键盘按键）



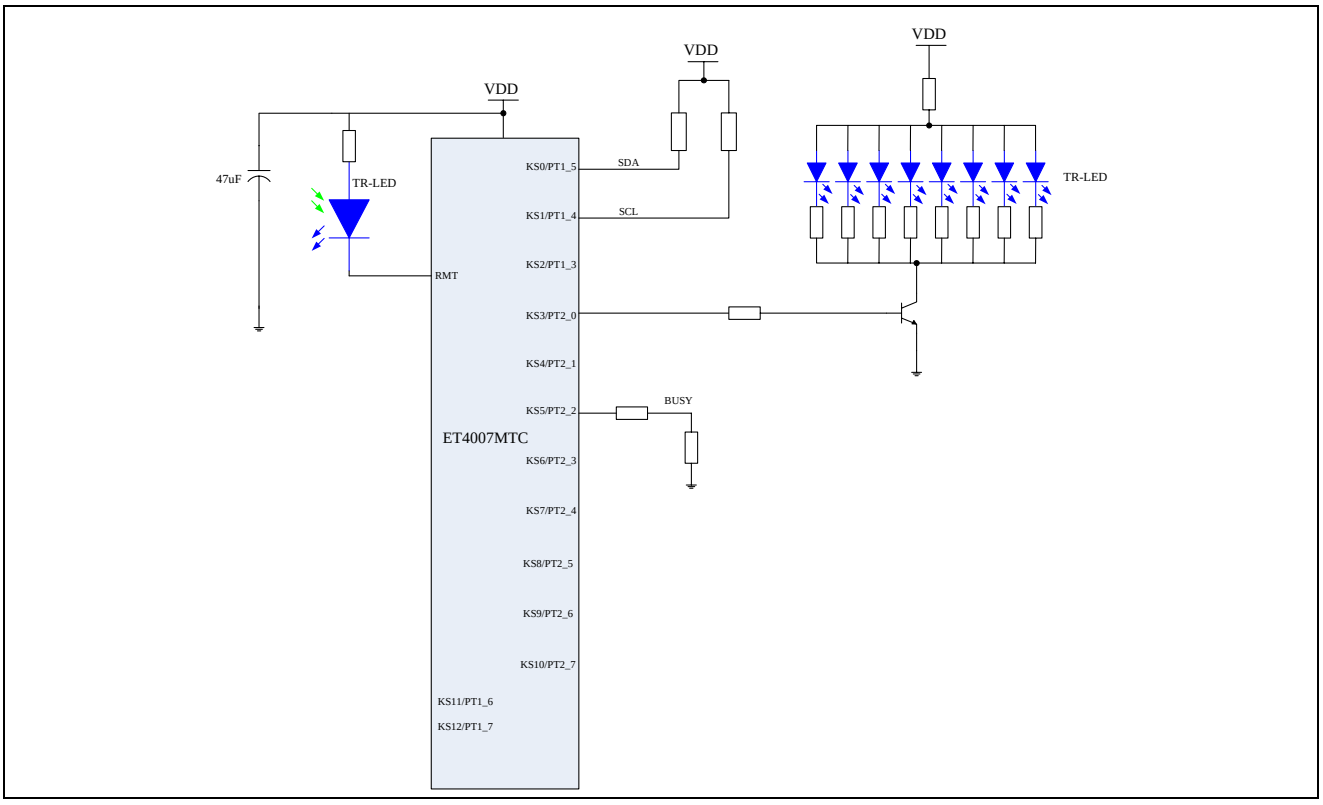
注：此图仅供参考

3）学习型遥控器（一个指示 LED，一个接收发射 TR-LED，共 55 个 T-type 型键盘按键，一个普通 IO 按键，支持普通一个 IO 按键和任意一个 T-type 型键盘按键组成的双键）



注：此图仅供参考

4）智能家居学习型遥控方案



注：此图仅供参考

PCB 布线注意事项

为了防止噪声影响 CPU 的正常工作，在 PCB 布线时需要注意以下几点:

1、 电源、地和滤波电容(47uF)之间以及 RMT 和 TR-LED 之间的连接线尽量短和粗，线径最好大于 0.5mm。

2、 EEPROM 的 SCL 和 SDA 管脚需要外接 4.7~10kΩ上拉电阻。

3、 若 TR-LED 和 RMT 管脚的连接线比较长，则该线需要一根 GND 线来屏蔽以便减小来自 T-type 键扫描线的开关干扰。

极限参数

参　数	符　号	范　围	单　位
提供电压	VDD	-0.3 ~ 4.0	V
贮藏温度	Tstg	-55 ~ 125	℃
输入电压	VIN	-0.3 ~ VDD+0.3	V
输出电压	VOUT	-0.3 ~ VDD+0.3	V
工作温度	Topr	-20～+70	℃

电参数

Ta=25℃, V_{DD}=3V

特　性	符 号	测 试 条 件	最小值	典型值	最大值	单 位
工作电压	V_{DD}		2.0		3.6	V
工作电流	I_{DD}	Fosc=5MHz	—	1.5	3.0	mA
待机电流	I_{STOP}	待机模式	—	1.0	1.5	μA
输入高电平	V_{IH}		$0.5V_{DD}$			V
输入低电平	V_{IL}				$0.3V_{DD}$	V
PT1，PT2 输出高电平	V_{OH}	I_{OH}=-8mA	$0.8V_{DD}$			V
PT1，PT2 输出低电平	V_{OL}	I_{OL}=8mA			$0.2V_{DD}$	V
RMT 输出驱动	I_{OH}	V_{OL}=0.3V，case1		120		mA
RMT 输出驱动	I_{OH}	V_{OL}=0.3V，case2		260		mA
系统时钟频率	Fosc	V_{DD}=3V Ta=25℃	4.95	5	5.05	MHz
		V_{DD}=3V Ta=-20℃~85℃	4.9	5	5.1	

注：case1：RMTCTR 寄存器 pwm_curt=000

case2：RMTCTR 寄存器 pwm_curt=111

封装

SOP16 Package