CMT2187A

Sub-1G 发射微控制器

用户手册

➢ 本手册的目的和对象读者

1. 本手册主要介绍CMT2187A的功能、操作事项和使用方法，对象读者为使用本系列实际开发产品的工程师。
2. 本手册受限于文档篇幅，芯片各功能模块牵涉寄存器仅作列表，详细寄存器说明请查看《CMT2187A 寄存器详细手册》，用户结合此文档和本用户手册，能更高效理解芯片功能。

产品主要特性

MCU 特性

- CPU 内核
  - 高性能单指令周期 1T-8051 内核
  - 支持最高 26MHz（XOSC）或 24Mhz（HFOSC）运行频率，最高取址效率为 20MIPS
  - 运行功耗为 111uA/MHz
- 储存体
  - 4 KB MTP 程序储存体，支持 10K 次擦写
  - 512 Byte XRAM 和 256 Byte IRAM
  - 512 bits EEPROM，支持 10 万次擦写
- 电源
  - 上电复位和低电压检测
  - 内置独立 LDO 给 CPU 和数字电路供电
  - 内置超低功耗 ULPLDO，在 STOP 模式下实现 CPU/RAM/SFR/部分外设的 Retention 功能
- I/O
  - 11 / 9 个多功能 IO 管脚（SOP16 / SOP14）
  - 支持高度灵活的外设功能映射
  - 支持电平变化中断/唤醒
- 时钟源
  - 支持高速 26MHz XOSC（晶体振荡器）
  - 内置高速 24MHz HFOSC（±1% RC 振荡器）
  - 内置低功耗 32kHz LFOSC（±1% RC 振荡器）
- 片上调试
  - CPU 内置 1-Wire 调试器硬件电路
  - 支持使用 Keil C51 进行程序在线调试
  - 支持 3 个硬件断点，单步调试
- 外设
  - 1x UART
  - 1x SPI
  - 1x CDR（单线 RX 输入时钟恢复）
  - 1x WDT（独立硬件）
  - 1x 睡眠定时器（32KHz LFOSC）
  - 2x 16 位简易定时器
  - 2x 16 位多功能定时器（3 通道 PWM/CCP）
  - 2x 模拟比较器
- 代码安全性
  - 烧录串口和单线调试接口带锁死功能

Sub-1G 发射模块特性

- 工作频率：210 - 960MHz
- 调试模式：OOK、FSK、GFSK
- 数据率： 0.5– 40kbps（OOK）
- 输出功率：+13dBm（最大输出）
- 工作电流：24mA @+13dBm，433.92MHz CW
- 单端高效 Class E 高频发射 PA
- PA Ramping 斜率根据速率可变

工作条件

- -40 到 85℃温度范围
- 1.8 到 3.6V 工作电压范围

应用

- 车库门遥控
- 遥控门禁系统
- 消费类无线遥控
- 智能家居
- 家居安防
- 有源 RFID 标签
- 无线传感网络
- WM-Bus T1 模式

封装

- SOP14

SOP-14

8.65 x 6 x 1.75 mm

简介

CMT2187A 是内嵌增强型 1T-8051 内核的低功耗 SoC RF 发射器：

1. 该芯片支持 210～960 MHz，OOK / (G)FSK 调制的无线发射功能；
2. 发射模块提供高效的单端 PA，输出功率可调范围 0～+13dBm，+13dBm 发射时仅需 24mA；
3. 提供 4-KB MTP 程序存储体，512-Byte 的 XRAM，256-Byte 的 IRAM，以及 512-bit 的 EEPROM；
4. 内置超低功耗的 ULPLDO 支持芯片在 STOP 模式保存 CPU 状态，RAM 数据，以及配置寄存器的数据
5. 采用单线（1-WIRE）在线仿真功能，用户可通过专用 1-WIRE 调试器和 Keil C51 软件把目标调试代码直接下载到片内 MTP 中运行，调试非常便捷；
6. 支持外置 26MHz XO 或内置 24MHz HFOSC 作为系统主频，内置低功耗 32 kHz LFOSC 可用于低功耗定时唤醒；
7. 支持单线输入硬件时钟恢复模块，方便内核同步采集外来数据（如 RX 接收数据）。

CMT2187A 搭配 CMOSTEK 的 NextGenRF™ 系列接收机便能轻松胜任各种超低功耗无线网络应用需求。

产品型号信息

产品型号	封装	尺寸
CMT2187A-ESR	SOP-14	8.65 mm x 6.00 mm x 1.75mm

声明

- 深圳市华普微电子股份有限公司（以下简称华普微或华普）保有在不事先通知的情况下而修改这份文档的权利。华普微认为提供的信息是准确可信的。本文档信息于 2024 年 9 月开始使用。在实际进行生产设计时，请参阅各产品最新的数据手册等相关资料以获取本公司产品的最新规格。

- 华普微对本手册拥有包括版权等知识产权，受法律保护。未经本公司事先书面许可，任何单位及个人不得以任何方式或理由对本手册进行复制、修改、抄录、传播等。本文件所登载内容的错误，本公司概不负责。

- 华普微对于因使用本文件中列明的本公司产品而引起的，对第三方的专利，版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为华普微对其他公司或个人所拥有的专利，版权以及其它知识产权做出任何明示或默示的许可及授权。

- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用示例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，华普微概不负责。

- 另外，华普微的产品不建议应用于生命相关的设备和系统。在使用该器件中因为设备或系统运转失灵而导致的损失，华普微不承担任何责任。

- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应知晓并同意，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。

目录

1 系统架构介绍

CMT2187A 是一款内嵌 Sub-1GHz OOK / (G)FSK 发射器的高性能 8051 SoC，用户程序烧录于 4K Bytes 的 MTP 中，并可在最高 26MHz 的时钟频率下运行。该芯片集成了下面这些核心模块：

- 基于 MTP 的高性能 8051，带有 1-Wire 在线调试电路；
- 丰富的数字和模拟外设资源；
- Sub-1G OOK / (G)FSK 调制发射模块；

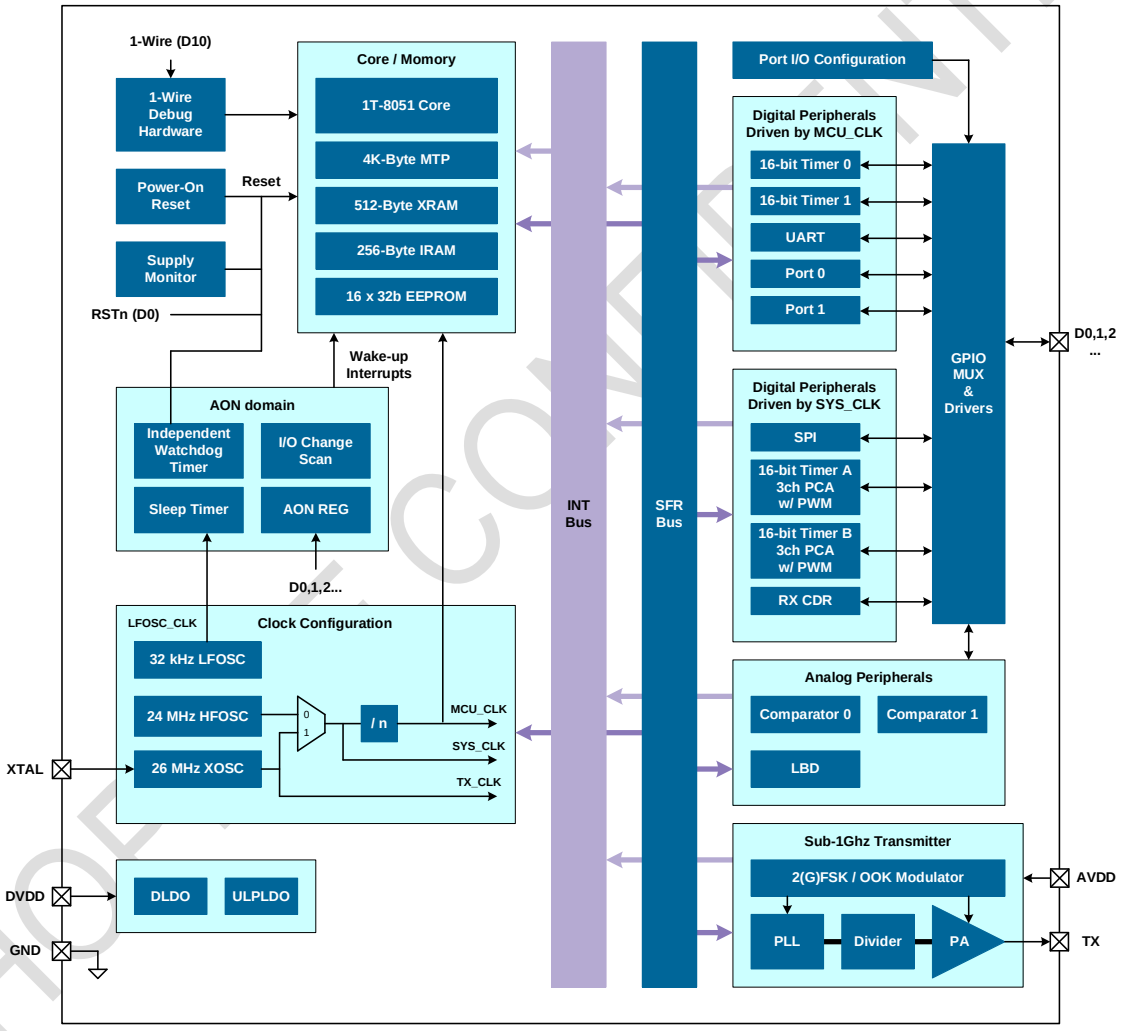

图 1-1. 系统框图

2 系统运行流程及工作模式

2.1 系统运行流程

CMT2187A 芯片系统运行流程如下图所示：

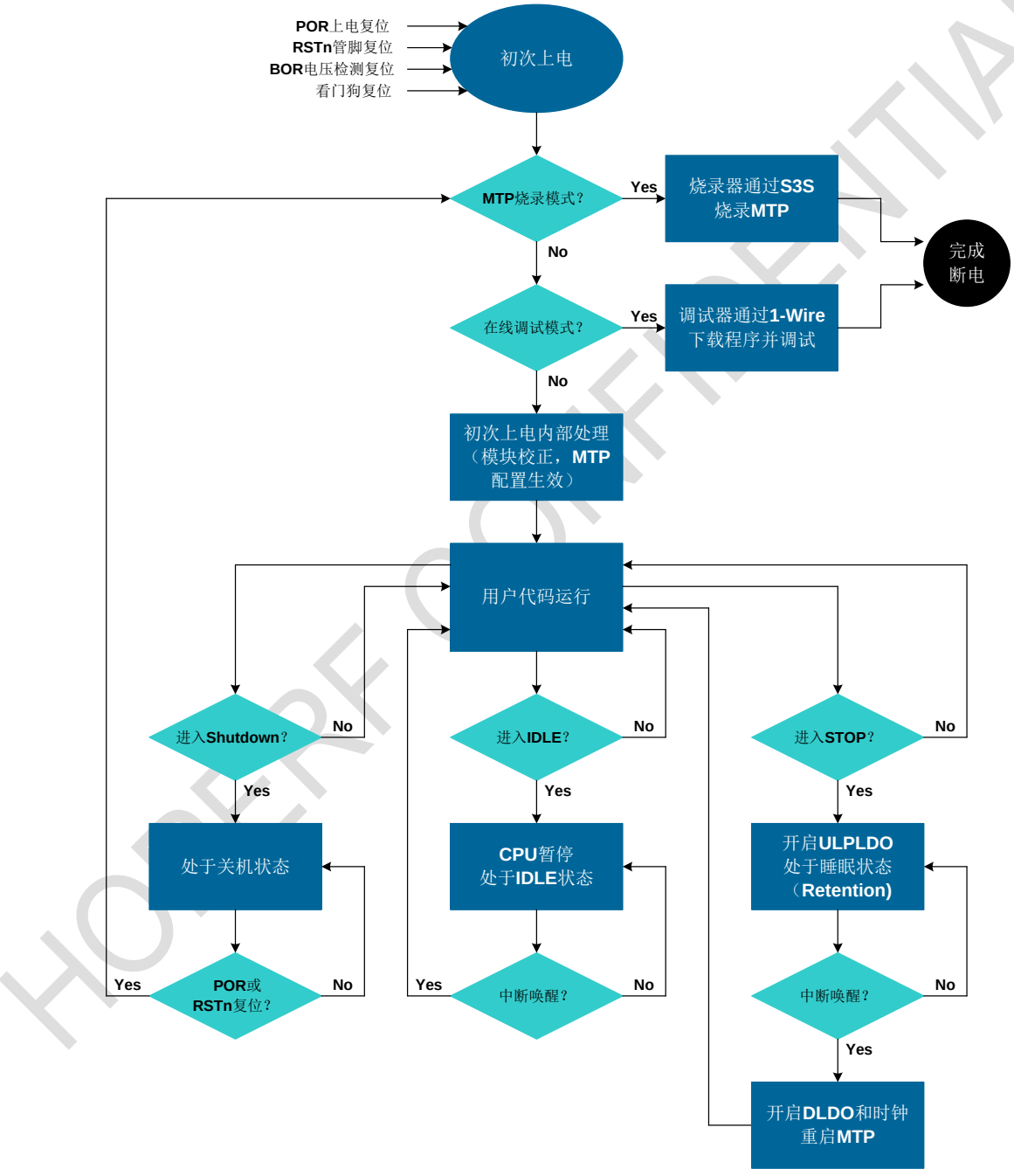

图 2-1. 系统运行流程图

上图中，芯片初次上电由 DVDD 管脚上电导致 POR 复位释放触发。当 RSTn 管脚复位，BOR 电压检测复位，看门狗复位生效时，芯片会进入同样的上电流程，这种情况下文均简称“上电”。在上电后，会打开一个约为 6 ms 的时间窗口，如果在窗口内检测 S3S 串口触发的烧录命令，则会进入烧录模式，允许烧录器烧录内部的 MTP；如果在窗口内检测到 1-Wire 接口触发的调试命令，则会进入调试模式，允许用户通过 Keil C51 软件和调试器进行用户代码调试。烧录或调试完成后，芯片需进行断电才能重新进行其它操作。

如果在上电后 6 ms 内没有触发烧录或调试模式，则芯片会继续进行初次上电的内部处理流程，包括电源和时钟校正，MTP 的 Config 区域内的配置生效等操作。接着用户代码就会开始从 0x0000 地址开始运行，运行过程中用户可以通过配置寄存器让芯片进入 IDLE 或 STOP 模式。在 IDLE 模式下可以通过 I/O 电平变化，或比较器输出翻转产生的中断唤醒。在 STOP 模式下，可以通过 I/O 电平变化，睡眠定时器超时，或比较器输出翻转产生的中断唤醒，唤醒后开启用于数字电路供电的 DLDO，时钟和 MTP，让用户代码可以继续从睡眠前的状态开始运行。

2.2 系统工作模式

芯片有如下 4 种工作模式：

表 2-1. CMT2187A 的 4 种工作模式

工作模式	详细描述	进入方式	唤醒源
Normal	正常工作状态	用户程序烧录后上电自动进入	无
IDLE	● DLDO 开启 ● 系统时钟（HFOSC 或 XOSC）开启 ● CPU 核暂停 ● 外设工作	设置 PCON 寄存器里面的 IDLE 位	I/O 电平变化 比较器输出
STOP (Retention)	● ULPLDO 开启 ● 系统时钟（HFOSC 或 XOSC）关闭 ● CPU 核，所有储存体，以及外设配置和状态保存 ● LFOSC 开启，Always-On 模块和比较器工作 ● GPIO 状态维持不变	1. 设置 PCON 寄存器里的 STOP 位 2. 设置 AON_SFR_03 寄存器里的 SLEEP 位	I/O 电平变化 比较器输出翻转 睡眠计时器超时

在 4 种模式中，从功耗的角度来说，Normal > IDLE > STOP。CMT2187A 有 2 个供电管脚，AVDD 负责给内部的射频电路供电，DVDD 负责给 Always-On 的数字模块和除射频外的模拟模块供电。绝大部分的数字模块工作在内置的 DLDO 下，在 STOP 时可切换到 ULPLDO 供电来实现低漏电保存（Retention）模式。

Retention 模式让芯片可以在 STOP 唤醒之后，立即从之前的状态恢复并继续工作，而不需要重头执行程序。在 Retention 模式下，所有 RAM 的数据都是保存的；MTP 和 EEPROM 的数据可断电保存。

表 2-2. CMT2187A 在 STOP 模式下储存体保存内容

储存体名称	保存数据	供电方式
MTP	√	断电保存
EEPROM	√	断电保存
IRAM	√	ULPLDO
XRAM	√	ULPLDO

在 Retention 模式下，上电复位（POR）和实时电压监测（Power Monitor）都维持工作状态。下面列出了所有功能模块是否保存 SFR 配置和工作状态，是否可工作，以及其对应的供电方式。

表 2-3. CMT2187A 在 STOP 下各功能模块保存内容

序号	模块名称	保存配置	保存工作状态	可否工作	供电方式
1	Watch Dog Timer	√	√	√	DVDD
2	Sleep Timer	√	√	√	DVDD
3	Key Scan	√	√	√	DVDD
4	Comparator 0	√	√	√	DVDD
5	Comparator 1	√	√	√	DVDD
6	UID & CFG 寄存器	√	√	×	DVDD
7	IO 配置和状态	√	√	×	DVDD
8	1T-8051 内核	√	√	×	ULPLDO
9	Timer 0	√	√	×	ULPLDO
10	Timer 1	√	√	×	ULPLDO
11	UART	√	√	×	ULPLDO
12	Port 0	√	√	×	ULPLDO
13	Port 1	√	√	×	ULPLDO
14	SPI	√	×	×	ULPLDO
15	Timer A	√	×	×	ULPLDO
16	Timer B	√	×	×	ULPLDO
17	CDR	√	×	×	ULPLDO
18	Sub-1G Transmitter	√	×	×	ULPLDO
19	LBD	√	×	×	断电
20	1-Wire Debug	×	×	×	断电

在上表中，序号为 1-7 的模块存在于 Always-On（一直上电）区域，在下面中简称 AON 区域，这个区域由 DVDD 直接供电，模块在不工作时漏电极小。其中看门狗，睡眠计时器，按键扫描，以及 2 个比较器在 STOP 模式下都可以根据用户配置决定是否工作；而 UID & CFG 寄存器，IO 配置和状态在 STOP 模式下则不会变化。

序号为 8-13 的模块是使用 MCU_CLK 驱动的 CPU 内核和外设，这部分电路的所有配置和当前状态在 STOP

模式下是保存的，但不能工作。

序号为 14-18 的模块是使用 SYS_CLK 驱动的外设，这部分电路的所有配置在 STOP 模式下是保存的，但当前的工作状态不能保存，在唤醒之后，用户无需重新配置这些模块，但模块会重新开始工作，其行为类似于模块被自动复位。

序号为 19-20 的模块是在 STOP 模式下完全断电的模块，不会保存任何内容。

2.3 保护机制

为了保证用户代码在烧录后不被窃取，芯片内部设计了安全机制。MTP 的 Config 区中有 READ_LOCK 保护位，当该保护位被烧录后，MTP 的用户代码和配置就无法通过 S3S 读取。如果用户希望解锁，可以通过烧录器重新烧录 MTP，过程中原来的用户代码和配置会被擦除，包括 READ_LOCK 位也会被擦除。

3 调试和烧录接口

3.1 1-WIRE 在线调试（烧录）器接口

CMT2187A 可通过 CMOSTEK 的 CMT2187A 仿真器器与 PC 连接，实现在线调试和 MTP 烧录的功能。下图工具连接图以及调试（烧录）器与 CMT2187A 芯片的接口连接图。需要注意的是，1-Wire 调试接口需要占用 D10 引脚，调试阶段建议用户留空此脚，勿做它用。MTP 烧录是通过三线的 S3S 接口实现的。

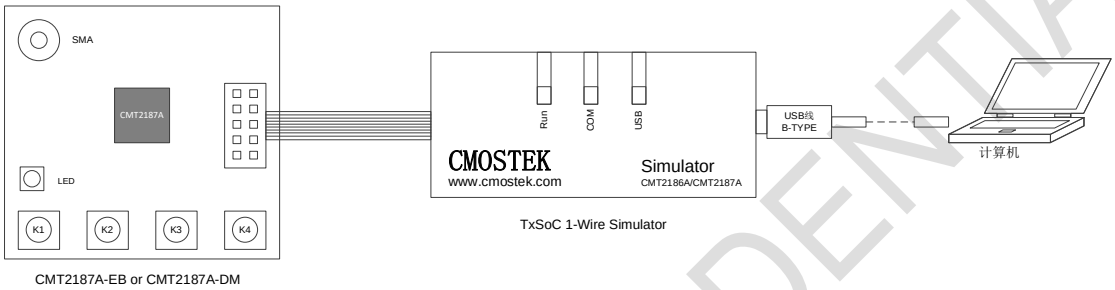

图 3-1. 1-wire 工具连接示意图

1-WIRE 在线调试接口，可在 Keil C51 平台下实现常用的功能：

- 全速运行、停止、单步执行、多步执行等调试方式；
- 支持设置软件断点（任意个数）；
- 支持 3 个硬件断点
- 读写 R0~R7、部分系统状态寄存器、memory 等内部存储体；
- *复位键（RST 图标）不可用，只能通过使用“退出→重新连接”的操作来代替。*

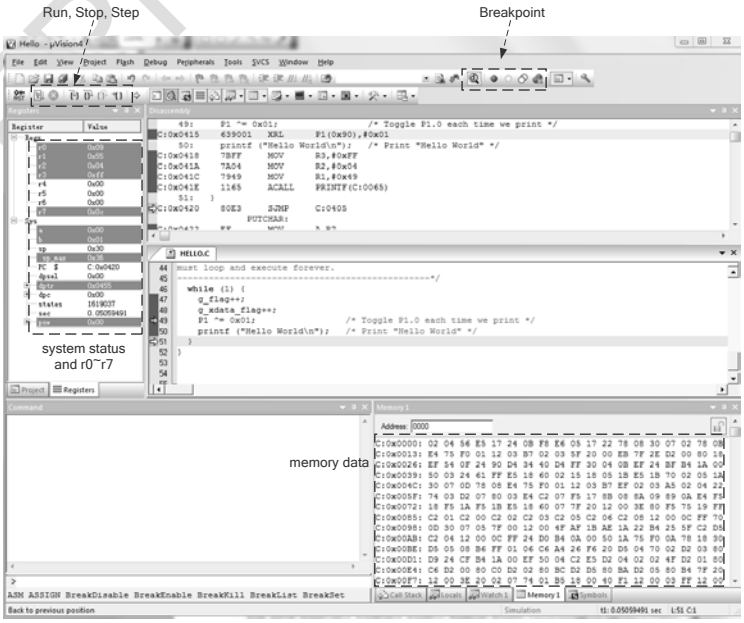

图 3-2. 1-wire 在 Keil C51 调试截图

3.2 S3S 总线烧录接口

S3S 总线用于烧录 MTP，仅限于烧录和生产工具使用，一般不向用户开放。如有特殊需求需要了解 S3S 总线具体时序和通讯协议，请联系华普微销售人员或其代理商。

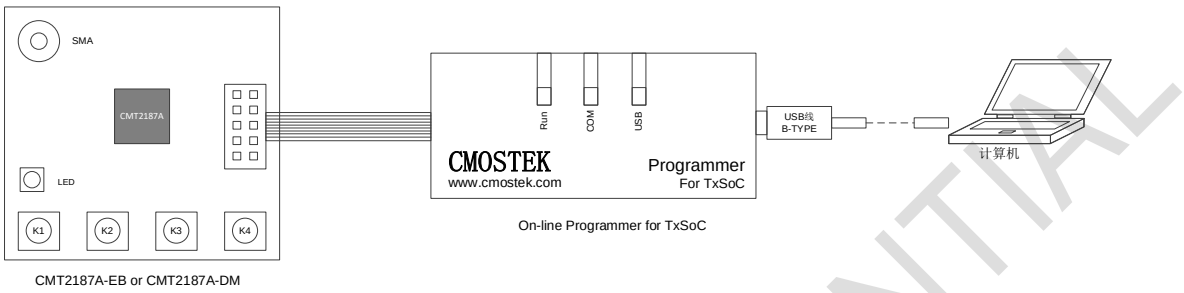

图 3-3. 在线烧录器工具连接示意图

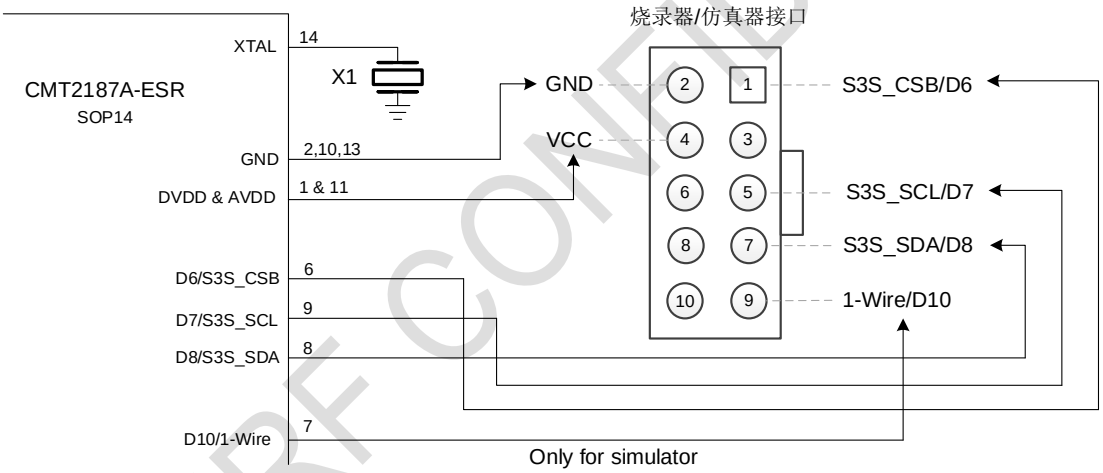

图 3-4. 烧录/仿真接口的接线示意图

4 T8051XC3 微控制器

4.1 处理器架构

CMT2187A 系列芯片采用 T8051XC3 作为系统的核心控制器，包含了增强型的 1T-8051 内核，单周期运行指令，完全兼容 MCS-51 指令集。其结构框图如下所示：

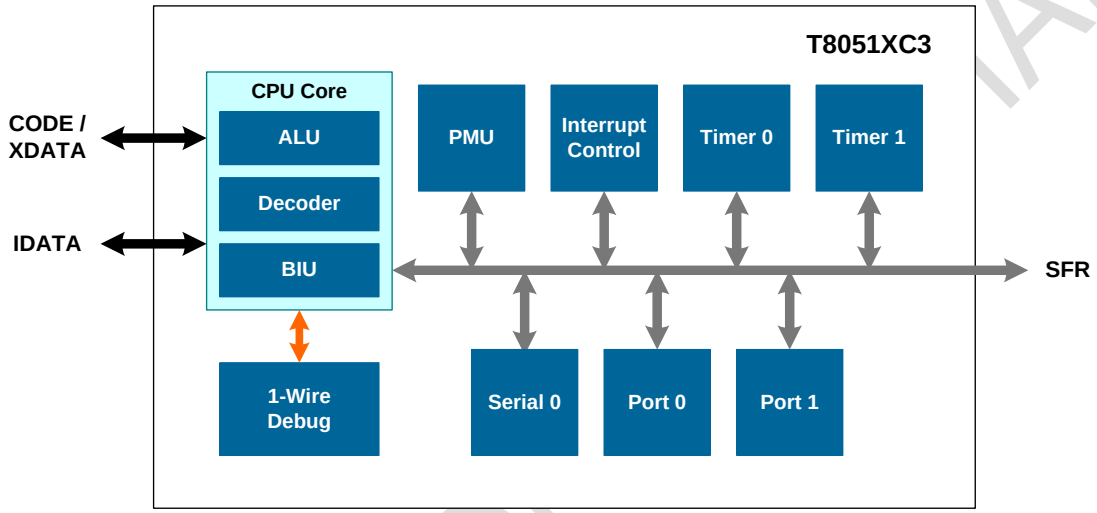

图 4-1. T8051XC3 系统框图

如图所示，T8051XC3 包含了如下部分：

- CPU 内核，由 BIU 总线接口单元，Decoder 指令译码单元，以及 ALU 算术逻辑单元组成
- 功耗管理单元，支持 IDLE 和 STOP 模式
- 中断控制单元，支持多达 8 个外部中断，带 2 级中断优先控制
- 两个定时器，即 Timer0 和 Timer1
- 一个串行口 Serial Port0，可实现 UART 模式
- 一个 11 位并行端口，即 Port0 和 Port1[1]，受限于芯片 I/O 数量，Port1 仅低 0-3 位可用
- 单线（1-WIRE）在线调试模块，支持 Keil C51 平台进行软件编程开发调试

注意[1]：Port0 和 Port1 是 T8051XC3 内核自带，与芯片的 GPIO 不直接等同。GPIO 相对内核而言属于外设，Port0 和 Port1 可映射到 GPIO 上。

T8051XC3 采用 8 位的 SFR 总线连接上面所说的外设。CMT2187A 支持更多的外设，也是通过 SFR 总线与内核连接。另外，内核采用独立的 IDATA 总线连接内部储存器 IRAM，以及共享的 CODE / XDATA 总线分别连接 MTP 和 XRAM。

4.2 指令集

8051 指令集有 111 条指令组成，每条指令有 1，2 或者 3 个字节构成，指令执行以单个时钟周期计算，所有指令及其执行周期请参阅附录 A。

4.3 8051 内核原生寄存器

内核 8051 原生关联寄存器组如下表所示，关于各个寄存器具体内容和含义，请查阅《CMT2187A 寄存器详细手册》。

表 4-1.内核 8051 原生的寄存器组列表

名称	SFR 页	地址	默认值	功能
P0	0	0x80	0x00	Port0 寄存器，支持位访问，对应 P0.0 – P0.7 这 8 个内核端口
SP	0	0x81	0x00	堆栈指针寄存器
DPL	0	0x82	0x00	数据指针（DPTR）寄存器，低 8 位
DPH	0	0x83	0x00	数据指针（DPTR）寄存器，高 8 位
PCON	0	0x87	0x00	功耗控制寄存器
TCON	0	0x88	0x00	Timer0 和 Timer1 控制寄存器
TMOD	0	0x89	0x00	Timer0 和 Timer1 工作模式寄存器
TL0	0	0x8A	0x00	Timer0 寄存器低 8 位
TL1	0	0x8B	0x00	Timer1 寄存器低 8 位
TH0	0	0x8C	0x00	Timer0 寄存器高 8 位
TH1	0	0x8D	0x00	Timer1 寄存器高 8 位
P1	0	0x90	0x00	Port1 寄存器，支持位访问，对应 P1.0 – P1.7 这 8 个内核端口，受限于 I/O 数量，仅 P1.0 – P1.3 可用。
SCON0	0	0x98	0x00	串口 0 控制寄存器
SBUF0	0	0x99	0x00	串口 0 数据缓存寄存器
IEN0	0	0xA8	0x00	中断使能寄存器 0
IPL0	0	0xB8	0x00	中断优先级寄存器 0
PSW	0	0xD0	0x00	程序状态/标志寄存器
ACC	0	0xE0	0x00	累加器寄存器
IEN1	0	0xE6	0x00	中断使能寄存器 1
B	0	0xF0	0x00	B 寄存器
IRCON1	0	0xF1	0x00	外设中断请求标志寄存器
IPL1	0	0xF6	0x00	中断优先级寄存器 1

5 存储体结构

5.1 基本介绍

CMT2187A 片内存储结构如图 5-1 所示。

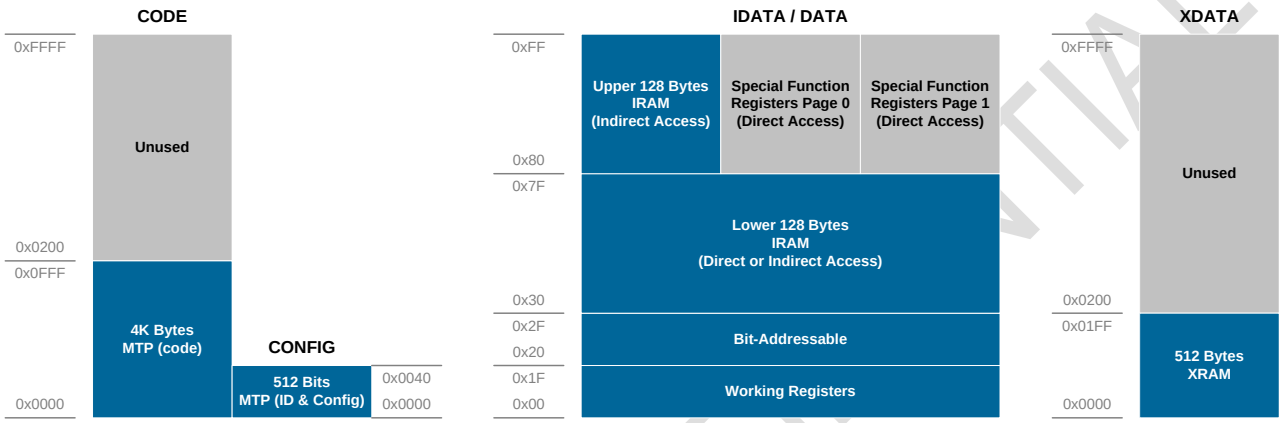

图 5-1. CMT2187A 存储体及逻辑地址

CMT2187A 存储空间可以分为如下三个主要部分：

- **程序代码空间**

  8051 内核代码存储和装载运行的空间，载体是可以多次擦写的 4K Bytes 的 MTP。MTP 同时支持 512 btis 大小的配置空间，用于存放用户 ID 和某些芯片特性配置。代码和配置空间的内容都是通过烧录器进行烧录。4KB 的代码空间全部提供给用户使用，代码从 0x0000 开始执行。

- **内部数据空间**

  8051 内核包含的 256 字节内部数据空间，以供 MCU 快速访问。内部数据空间按访问方式又可划分为 DATA、IDATA 及 SFR，和 Keil C51 编译器中关键字所对应，载体分别是 256 Bytes 的 IRAM 和 SFR 寄存器。SFR 分成两个页，用户可以通过 SFR_PAGE_SEL 比特进行选择。

- **外部数据空间**

  8051 外扩的数据存储，XDATA 存放于 512 bytes 的 XRAM 当中。用户还可以将某些需要掉电保存的数据存放在 512 bits 的 EEPROM，可通过 SFR 进行访问。另外，AON 区域中有一些 AON_REGS，这些寄存器主要用于配置和控制 AON 区域的模块和 I/O，同时用户可以通过 SFR 对它们进行间接访问。

表 5-1. 内存空间描述

储存空间	储存体	逻辑地址	容量	描述
程序代码空间	MTP	0x0000 - 0x0FFF	4K 字节	用户程序运行空间，Keil C51 中需要使用关键字 code 定义变量。
内部数据空间	IRAM	0x00 - 0x7F	128 字节	IDATA 低位段，可直接寻址或间接访问。此外，还提供 16 字节的可位寻址的地址空间，地址范围为 0x20 - 0x2F。Keil C51 中可使用关键字 data 或 idata 定义，用 sbit 定义按位访问的变量。
		0x80 - 0xFF	128 字节	IDATA 高位段，只支持间接访问。Keil C51 中必须用关键字 idata 定义。
	SFR	0x80 - 0xFF	145 字节	位于内部 RAM 地址区间的特殊功能寄存器，8051 直接访问。包含 Page 0 和 Page 1 两个页面，通过 SFR 中的 SFR_PAGE_SEL 比特进行选择。
外部数据空间	XRAM	0x0000 - 0x01FF	512 字节	Keil C51 需要用关键字 xdata 定义变量。
	EEPROM	0x00 - 0x1F	512 比特	大小为 16 Bit x 32 的可以多次编程内存。内核通过 SFR 进行间接访问，或通过开源的 API 程序对齐访问以增加其使用次数。
	AON REG	0x00 - 0x1F	32 字节	位于 AON 区域的寄存器，内核通过 SFR 进行间接访问。

注：

[1] MTP 烧录后，不管系统有电与否，或系统运行何种模式，所烧录数据（即用户程序）均不会丢失。

[2] EEPROM 操作擦写后（擦写过程需求确保供电稳定），不管系统有电与否，或系统运行何种模式，所擦写数据也不会丢失。

[3] AON REG 在 AON 区域内，只要 DVDD 持续有效供电，内容就不会丢失。

[4] IRAM，XRAM 和部分的 SFR 在 STOP 模式下可以保存内容。

5.2 特殊功能寄存器（SFR）

SFR 属于内部存储空间，8051 内核可以直接访问。CMT2187A 系列产品功能丰富，配置关联的 SFR 也较多，所以我们做了分页访问处理，包含了 Page 0 和 1。Page 0 包含了大部分的外设配置和控制，Page 1 包含了 EEPROM 访问的寄存器和 PA 的功率配置寄存器。所以直接访问对应的 SFR 时，务必确认 Page 指向正确，否则很容易导致配置错误。

SFR 在 STOP 模式下由 ULPLDO 进行供电，确保绝大部分外设的配置可以进行低漏电保存。

5.3 常开域寄存器（AON REG）

系统的常开（AON）域直接由 DVDD 供电，里面包含了看门狗，睡眠定时器，I/O 变化检测，以及 32 个字节的寄存器 AON REG。用户可通过 SFR 中的 AON_ADDR，AON_WDATA 和 AON_RDATA 寄存器间接访问 AON REG。这些寄存器控制和配置的对象包括：上述 AON 域里面的三个外设，两个模拟比较器，所有的 I/O。同时，AON REG 里面有 8 个字节，在芯片初次上电时，系统会自动将 MTP 中的 64 位的用户 ID 拷贝到这 8 个

字节的寄存器中以便用户使用。用户也可以随意将这 8 个字节的寄存器用作其它用途。

5.4 存储器运行中访问方式

下表总结各个存储体在运行中的访问方式：

表 5-2. 各存储体在运行中访问方式

存储类型	访问方式	示例
CODE	程序中常量定义，使用关键字“code”	uint8_t code array[3] = {0x12, 0x34, 0x56 };
XDATA	程序中变量定义，使用关键字“xdata”	uint8_t xdata tx_buf[64];
IDATA	程序中变量定义，使用关键字“idata”	uint8_t idata buf[3];
SFR	直接地址访问[2]	IEN0 = 0x00;
AON REG	通过 SFR 进行访问[1]	无
EEPROM	通过 SFR 进行访问，或 API 函数访问[2]	无
注： [1] 官方开源的历程会给出如何访问这些寄存器。 [2] 官方开源的历程会给出如何访问这些寄存器，以及 API 的源码，使用 API 可以增加 EEPROM 的反复擦写次数。		

6 复位结构

CMT2187A 有 4 个系统复位，包括：

- **上电复位（POR）**
  上电复位 POR 只会在 DVDD 上电时释放一次。

- **电压检测复位（BOR）**
  BOR 是 DVDD 发生非正常波动时产生，避免芯片工作错乱。

- **管脚复位（RSTn）**
  管脚复位 RSTn 复用了 D0 管脚，默认是使能的，如果用户不需要使用，可以在上电后屏蔽该功能。

- **看门狗复位（WDT_RSTn）**
  看门狗复位是防止程序跑飞或系统死机的复位，在用户程序正常运行时需要定时“喂狗”，避免看门狗定时器超时产生复位。

这 4 个复位都有相同的效果，即复位触发后，芯片都会重新进行初次上电。

7 时钟结构

7.1 时钟源

CMT2187A 有 3 个主时钟源，分别是 26 MHz 的高速晶体振荡器 XOSC，24 MHz 的内部高速 RC 振荡器 HFOSC，以及 32 kHz 的内部低速 RC 振荡器 LFOSC。芯片内部构建了精细化的时钟门控机制，以便用户能够尽可能节省功耗。

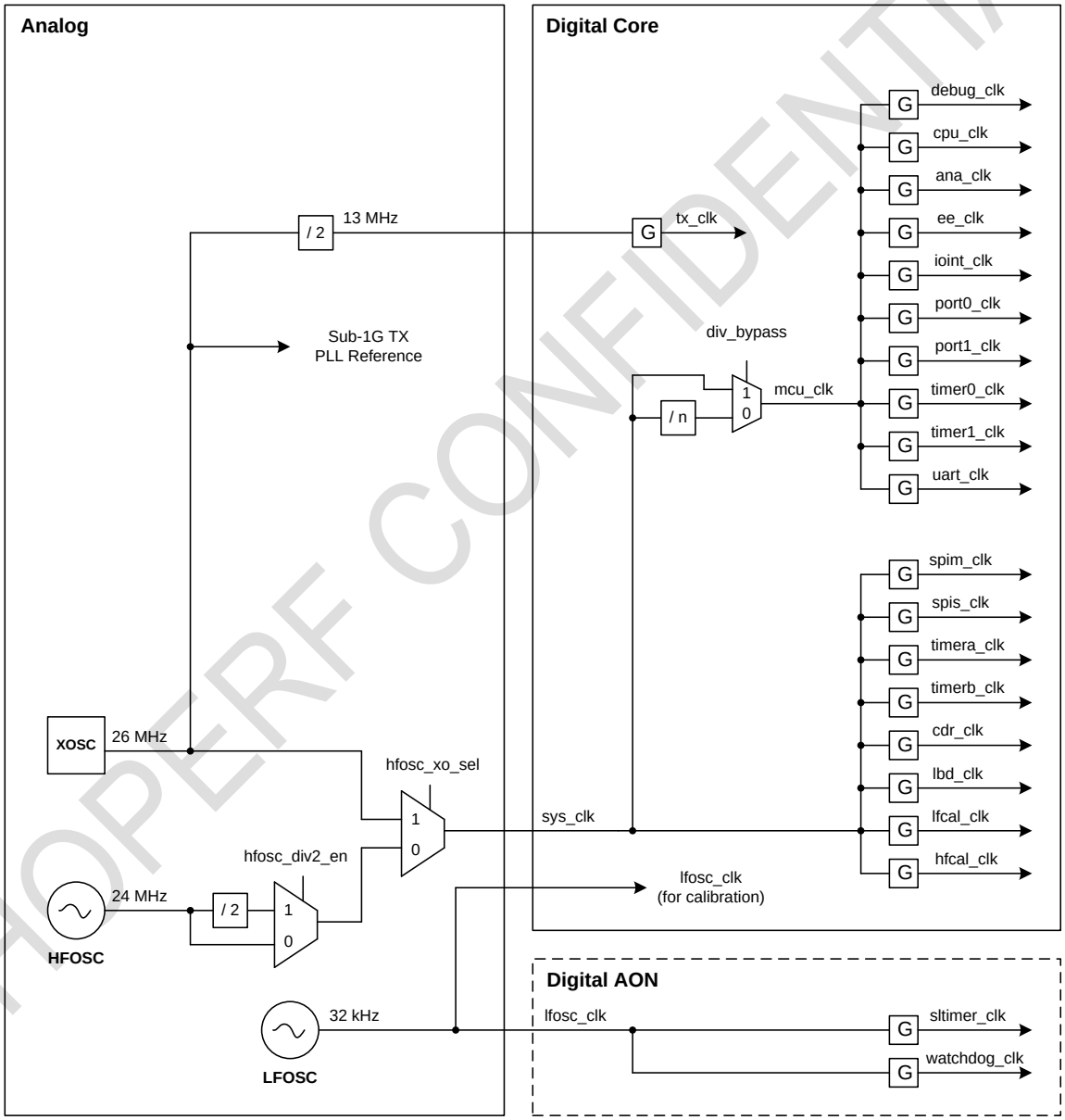

图 7-1.系统时钟框图

如上图所示，XOSC 作为 Sub-1G 无线发射机 PLL 的参考时钟，同时经过 2 分频后用于驱动数字发射控制

和调整模块。系统主时钟（SYS_CLK）是默认由 HFOSC 提供的，HFOSC 经过校正后可以达到±1%的精度。如果用户希望提升主时钟的精度，可以在烧录 MTP 时进行相关配置，让芯片在上电后自动将主时钟切换为 XOSC，在增加一些功耗的同时，可以将精度提升到晶体振荡器本身的精度，例如±10ppm。LFOSC 专门为睡眠定时器和看门狗提供时钟，经过校正后可以达到±1%的精度。

7.2 时钟校正

在芯片出厂时，HFOSC 和 LFOSC 会进行校正并将校正的结果烧录到 MTP 中。在用户使用的过程中，也可以通过操作 SFR 寄存器调用硬件校正模块来校正两个时钟。

HFOSC 的校正需要使用 XOSC 作为基准时钟。如果在某些应用中（例如非无线发射应用），芯片没有连接外部晶体振荡器，XOSC 就无法适应，这种情况下无法对 HFOSC 进行校正。在对 HFOSC 进行校正前，需要确保 HFOSC 作为 SYS_CLK 的时钟源，同时将 SFR 中 HFOSC_DIV2_EN 比特设为 1，将 HFOSC 时钟进行二分频后再作为 SYS_CLK 使用，这是为了避免在校正过程中 HFOSC 频率提升过高导致系统工作失常。

LFOSC 校正需要使用 SYS_CLK 作为基准时钟。如果用户通过 MTP 烧录选择了 HFOSC 作为 SYS_CLK 的时钟源，则建议先对 HFOSC 进行校正，再对 LFOSC 进行校正，因为此时 HFOSC 的精度决定了 LFOSC 的精度。

校正的具体操作可参考官方开源的例程代码。

7.3 时钟分频

用户可以操作一个分频器对 SYS_CLK 进行分频产生 MCU_CLK。该分频器的 8 位分频系数可配置范围是 1 到 255，不可配置成 0。因此 MCU_CLK 的最高工作频率是 24 MHz（HFOSC）或 26 MHz（XOSC），最低工作频率是 94 kHz（HFOSC）或 102 kHz（XOSC）。

SYS_CLK 除了上面介绍过的在校正时可以进行 2 分频（会同时影响 MCU_CLK），在正常工作时不进行任何分频。SYS_CLK 驱动的外设，除了 LBD 使用固定的时钟频率外，SPIM，SPIS，TIMERA，TIMERB，以及 CDR 都有配置自身工作频率的 SFR 寄存器，无需要对驱动时钟进行分频处理。

LFOSC_CLK 直接驱动了睡眠定时器和看门狗定时器，没有进行任何分频处理。

7.4 时钟门控

为了充分发挥芯片低功耗的特性，芯片内部对每一个模块都提供了独有的时钟门控，这些时钟门控既控制模块本身的时钟，同时也控制该模块对应的 SFR 寄存器的时钟。时钟门控默认都是打开的，建议用户在程序开始运行的初期，在配置好所有 SFR 后，将不需要立即工作的模块的门控时钟关闭，仅在需要配置，控制，和使用

该模块时才打开。

下面给出没一个时钟门控对应控制的模块，以及详细的 SFR 寄存器，以便用户使用：

表 7-1. 时钟门控对应的模块和寄存器

时钟门控	模块	SFR 页	SFR 地址	SFR 名称
TX_CLK_EN	OOK / (G)FSK 发射控制器和调制器	0	0xDD	ANA_CTL_0
		0	0xDE	ANA_CTL_1
		0	0xDF	ANA_CTL_2
		0	0xE1	ANA_CTL_3
		0	0xE8	PLLN
		0	0xE9	PLLK_H
		0	0xEA	PLLK_M
		0	0xEB	PLLK_L
		0	0xEC	TX_DR_0
		0	0xED	TX_DR_1
		0	0xEE	TX_DR_2
		0	0xEF	TX_SYM_BYTE
		0	0xF2	TX_SYM_CTL
		0	0xF3	TX_PKT_CTL
		0	0xF8	RAMP_STEP_H
		0	0xF9	RAMP_STEP_L
		0	0xFA	PA_IDAC_CODE
		0	0xFB	LBD_CTL_0[1]
		0	0xFC	LBD_CTL_1[1]
CPU_CLK_EN	CPU 内核	无	无	根据工作模式自动开关
DEBUG_CLK_EN	1-Wire 调试器	无	无	根据是否进入调试模式自动开关
EE_CLK_EN	EEPROM 控制器	1	0x2A	EE_CTL
		1	0x2B	EE_ADDR
		1	0x2C	EE_WDATA_H
		1	0x2D	EE_WDATA_L
		1	0x30	EE_RDATA_H
		1	0x31	EE_RDATA_L
		1	0x32	EE_STA
		1	0x33	EE_MANU
IOINT_CLK_EN	IO 和中断控制器	0	0x92	INTCTL_0
		0	0x93	INTCTL_1
		0	0x94	INTCTL_2
		0	0x95	INTCTL_3
		0	0xA1	GPIO_INA_SEL
		0	0xA2	GPIO_INB_SEL

时钟门控	模块	SFR 页	SFR 地址	SFR 名称
		0	0xA3	GPIO_INC_SEL
		0	0xA4	GPIO_IND_SEL
		0	0xA5	GPIO_INE_SEL
		0	0xA6	GPIO_INF_SEL
		0	0xA9	GPIO_ING_SEL
		0	0xAA	GPIO_OUTA_SEL
		0	0xAB	GPIO_OUTB_SEL
		0	0xAC	GPIO_OUTC_SEL
		0	0xAD	GPIO_OUTD_SEL
		0	0xB0	GPIO_OUTE_SEL
		0	0xB1	GPIO_OUTF_SEL
PORT0_CLK_EN	Port 0	0	0x80	P0（8051 原生寄存器）
PORT1_CLK_EN	Port 1	0	0x90	P1（8051 原生寄存器）
TIMER0_CLK_EN	Timer 0	0	0x8A	TL0（8051 原生寄存器）
		0	0x8C	TH0（8051 原生寄存器）
TIMER1_CLK_EN	Timer 1	0	0x8B	TL1（8051 原生寄存器）
		0	0x8D	TH1（8051 原生寄存器）
UART_CLK_EN	UART 0	0	0x98	SCON0（8051 原生寄存器）
		0	0x99	SBUF0（8051 原生寄存器）
ANA_CLK_EN	模拟电路控制器	0	0xE2	ANA_CTL_4
		0	0xE3	ANA_CTL_5
		0	0xE4	ANA_CTL_6
		0	0xE5	ANA_CTL_7
		0	0xE7	ANA_CTL_8
SPIM_CLK_EN	SPI 主机	0	0x96	SPI_CTL_0
		0	0x97	SPI_CTL_1
SPIS_CLK_EN	SPI 从机	0	0x96	SPI_CTL_0[2]
		0	0x97	SPI_CTL_1[2]
TIMERA_CLK_EN	Timer A	0	0xB7	TACLK_DIV_H
		0	0xB9	TACLK_DIV_L
		0	0xBA	TAC_H
		0	0xBB	TAC_L
		0	0xBC	TACCR0_H
		0	0xBD	TACCR0_L
		0	0xBE	TACCTL0_H
		0	0xBF	TACCTL0_L
		0	0xC0	TACCR1_H
		0	0xC1	TACCR1_L
		0	0xC2	TACCTL1_H
		0	0xC3	TACCTL1_L

时钟门控	模块	SFR 页	SFR 地址	SFR 名称
		0	0xC4	TACCR2_H
		0	0xC5	TACCR2_L
		0	0xC6	TACCTL2_H
		0	0xC7	TACCTL2_L
		0	0xC8	TACNT_H
		0	0xC9	TACNT_L
TIMERB_CLK_EN	Timer B	0	0xCA	TBCLK_DIV_H
		0	0xCB	TBCLK_DIV_L
		0	0xCC	TBC_H
		0	0xCD	TBC_L
		0	0xCE	TBCCR0_H
		0	0xCF	TBCCR0_L
		0	0xD1	TBCCTL0_H
		0	0xD2	TBCCTL0_L
		0	0xD3	TBCCR1_H
		0	0xD4	TBCCR1_L
		0	0xD5	TBCCTL1_H
		0	0xD6	TBCCTL1_L
		0	0xD7	TBCCR2_H
		0	0xD8	TBCCR2_L
		0	0xD9	TBCCTL2_H
		0	0xDA	TBCCTL2_L
		0	0xDB	TBCNT_H
		0	0xDC	TBCNT_L
CDR_CLK_EN	时钟恢复器	0	0x9D	CDR_DR_0
		0	0x9E	CDR_DR_1
		0	0x9F	CDR_DR_2
LBD_CLK_EN	低电压检测器	无	无	仅控制 LBD 模块本身[1]
LFOSC_CLK_EN	LFOSC 校正	无	无	仅控制 LFOSC 校正模块本身[3]
HFOSC_CLK_EN	HFOSC 校正	无	无	仅控制 HFOSC 校正模块本身[3]
SLTMR_CLK_EN	睡眠计时器	无	无	通过 AON_REG 开关模块时自动开关[4]
WDG_CLK_EN	看门狗计时器	无	无	通过 AON_REG 开关模块时自动开关[4]

注：

[1] LBD 寄存器的配置会影响部分 TX 电路的功能，因此由 TX_CLK_EN 门控时钟驱动，由 LBD_CLK_EN 门控的时钟仅驱动 LBD 模块本身。在实际使用中，在进行发射前都需要检测电池电压以便进行发射功率补偿，因此建议用户在发射前同时打开 TX_CLK_EN 和 LBD_CLK_EN。在不进行发射时，如需要单独使用 LBD 模块，需要打开 TX_CLK_EN 以便使用 LBD_CTL_0 和 LBD_CTL_1 寄存器，以及打开 LBD_CLK_EN 让 LBD 模块工作。

[2] 当 SPIM_CLK_EN = 1 或 SPIS_CLK_EN = 1 时，SPI_CTL_0 和 SPI_CTL_1 寄存器的时钟都会被打开。

[3] 当使用 LFOSC 或 HFOSC 校正模块时，需要通过 ANA_CTL_8 寄存器进行控制，因此需要设置 ANA_CLK_EN = 1。

[4] 所有的 AON_REG 都不需要时钟门控，这些寄存器的时钟仅会在 CPU 对它们访问时打开。

7.5 相关寄存器

表 7-2. 系统时钟相关寄存器列表

名称	SFR 页	地址	默认值	功能
CLK_GATE_0	0	0x84	0x7F	Port0 寄存器，支持位访问，对应 P0.0 – P0.7 这 8 个内核端口
CLK_GATE_1	0	0x85	0xFF	堆栈指针寄存器
CLK_GATE_2	0	0x86	0x7F	数据指针（DPTR）寄存器，低 8 位
MCU_CLK_DIV	0	0xFD	0x01	MCU_CLK 分频系数

8 中断及唤醒

8.1 基本介绍

CMT2187A 的中断控制主要起到两种作用：

- 第一种：中断当前运行流程，优先处理中断服务流程；
- 第二种：从低功耗模式中唤醒系统；

第一种作用，与传统的微控制器一样，都是在程序运行过程中响应中断处理的方式，所有的中断源都支持。第二种作用，是为了满足低功耗的应用需求，在系统进入各种低功耗的模式后，通过中断进行唤醒。在这种情况下，只有有限的中断源可以支持唤醒的功能。在这里我们把能支持唤醒的中断源，称为“唤醒源”，以便能更好理解其工作机制。唤醒源和低功耗模式是关联的，下面介绍 CMT2187A 的三种低功耗模式的详细信息：

- **IDLE 模式**

  在 IDLE 模式下，8051 内核停止工作，此时 MCU_CLK 并没有停止，仅 CPU 内核的时钟 CPU_CLK 停止，因此原生外设和系统外设仍然正常工作，所以通过此两种外设中断可以唤醒 IDLE 模式。

- **STOP 模式**

  在 STOP 模式下，芯片进入睡眠状态，MCU 供电源从 DLDO 切换成 ULPLDO 以便低功耗保存当前工作状态，这时除了 LFOSC 外所有的时钟停止，因此只能依靠 AON 域的中断进行唤醒，包括 I/O 电平变化，睡眠定时器超时，以及模拟比较器输出翻转。如果用户不使用睡眠唤醒，在进入 STOP 模式前也可以不打开 LFOSC 和睡眠定时器，可以进一步节省功耗。

8.2 唤醒源

前面小节提及，只有位于常开域的系统外设才能支持 STOP 模式中唤醒系统，我们称其为唤醒源。CMT2187A 的唤醒源主要来自如下 3 个功能模块：

- **I/O 变化检测模块（I/O Change Scan）**

  CMT2187A 的 D0 - D11 均可支持该功能，用户需要在进入 STOP 模式前，配置需要检测唤醒的 GPIO 即可。

- **睡眠定时器（Sleep Timer）模块**

  低功耗睡眠定时器唤醒 STOP 模式。

- **模拟比较器模块**

  模拟比较器用于比较两个输入信号，当比较结果发生变化时，可以出发中断唤醒系统，用户需要在进入

STOP 模式前，配置好比较器的工作模式。

8.3 中断源和中断控制

前面小节已经介绍了 CMT2187A 的唤醒源，由于它们支持单纯唤醒系统，可以理解唤醒源相对于系统而言的。而本小节将要介绍的 CMT2187A 中断源，主要与 8051 运行所关联，即基于代码在运行过程中，对中断响应的具体处理。

CMT2187A 内部 8051 支持 11 个中断源，分别为：

- 1 个 Timer 0 中断；
- 1 个 Timer 1 中断；
- 1 个 Serial 0（即 UART）中断；
- 8 个外部中断（下文简称 INT）；

每个中断源可独立使能，且可配置 2 级中断优先级。表 8-1 给出 11 个中断源对应的中断向量及其对应的中断源关系。

表 8-1. CMT2187A 中断向量表

中断号	中断向量	中断源	中断请求标志	中断使能控制	中断优先级
0	0x0003	外部中断 0	IE0	EX0	IPL0[0]
1	0x000B	Timer 0 中断	TF0	ET0	IPL0[1]
2	0x0013	外部中断 1	IE1	EX1	IPL0[2]
3	0x001B	Timer 1 中断	TF1	ET1	IPL0[3]
4	0x0023	UART 中断	TI0/RI0	ES0	IPL0[4]
5	0x004B	外部中断 2	IE2	EX2	IPL1[2]
6	0x0053	外部中断 3	IE3	EX3	IPL1[3]
7	0x005B	外部中断 4	IE4	EX4	IPL1[4]
8	0x0063	外部中断 5	IE5	EX5	IPL1[5]
9	0x006B	外部中断 6	IE6	EX6	IPL1[6]
10	0x0073	外部中断 7	IE7	EX7	IPL1[7]
注： T8051XC3 内核控制器的中断响应时间最短为 3 个系统时钟，系统时钟是内部 24MHz RC 振荡器 HFOSC 或外部 26MHz 晶体振荡器 XOSC 提供。					

8.4 外部中断映射

上述提到 T8051XC3 支持的中断源中有 11 个中断，除了 Timer0，Timer1 和 UART 三个中断源是不可选择中断触发源外，剩余的 8 个外部中断则是可以进行灵活的中断源选择。CPU 内核通过 INT BUS（中断总线）连

接到各个外设的中断源。外部中断源一共有 27 个，如下所示：

- I/O 输入中断功能，D0 - D11，共 12 个；
- Timer A / B 模块，每组定时器有 4 个中断，共 8 个；
- Sub-1G 发射模块的 FIFO 空标志中断触发，1 个；
- SPI 模块收发数据中断触发，2 个；
- 比较器输出中断，2 个；
- CDR 输出中断，1 个。
- 睡眠定时器中断，1 个

外部中断 INT0 和 INT1 专门用于连接 3 个唤醒源， INT2 - INT7 用于连接各个外设和 I/O 的中断源，映射各有不同，下面以 INT0，INT1 和 INT2 为例，画出中断结构图，详细的中断映射关系请参考寄存器表。

如下图所示，INT0 连接到比较器中断和 IO 输入中断，任意一个比较器或者 I/O 边缘检测都会触发 INT0，在 STOP 模式下作为唤醒源使用；INT1 固定连接到睡眠定时器中断，也是在 STOP 模式下作为唤醒源使用。CPU 被唤醒后进入中断响应程序，可以通过 SFR 查询 FLAG 来确定具体是哪个中断唤醒了系统，并做相关处理。INT2 主要映射到外设中断，I/O 中断只映射了 D0 - D3。注意在这种情况下，D0 - D3 的变化既会影响 INT0 也会有可能映射到 INT2，用户应该合理配置中断使能和映射，在 STOP 时使用 INT0，在程序运行时使用 INT2。

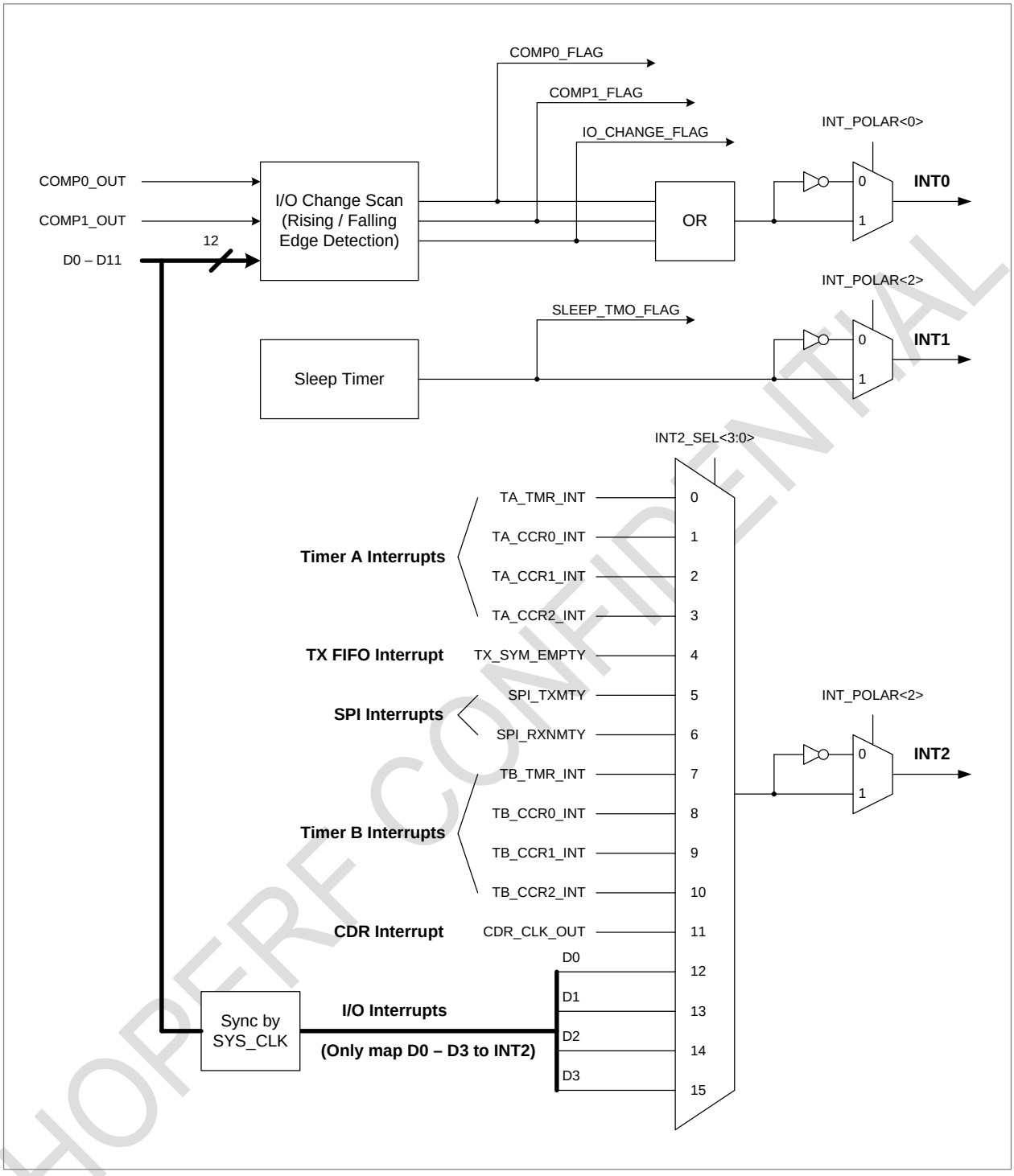

图 8-1. 外部中断 INT0，INT1 和 INT2 映射图

下面给出 INT2 - INT7 到各中断源的的详细映射关系：

表 8-2. 外设中断源与 8 个外部中断的映射关系表

中断	中断选择	中断源	说明	中断	中断选择	中断源	说明
INT2	INT2_SEL = 0000	TA_TMR_INT	Timer A 中断	INT3	INT3_SEL = 0000	TA_TMR_INT	Timer A 中断
	INT2_SEL = 0001	TA_CCR0_INT			INT3_SEL = 0001	TA_CCR0_INT	
	INT2_SEL = 0010	TA_CCR1_INT			INT3_SEL = 0010	TA_CCR1_INT	
	INT2_SEL = 0011	TA_CCR2_INT			INT3_SEL = 0011	TA_CCR2_INT	
	INT2_SEL = 0100	TX_SYM_EMPTY	发射机中断		INT3_SEL = 0100	TX_SYM_EMPTY	发射机中断
	INT2_SEL = 0101	SPI_TXMTY	SPI 中断		INT3_SEL = 0101	COMP0_OUT	比较器中断
	INT2_SEL = 0110	SPI_RXNMTY			INT3_SEL = 0110	COMP1_OUT	
	INT2_SEL = 0111	TB_TMR_INT	Timer B 中断		INT3_SEL = 0111	TB_TMR_INT	Timer B 中断
	INT2_SEL = 1000	TB_CCR0_INT			INT3_SEL = 1000	TB_CCR0_INT	
	INT2_SEL = 1001	TB_CCR1_INT			INT3_SEL = 1001	TB_CCR1_INT	
	INT2_SEL = 1010	TB_CCR2_INT			INT3_SEL = 1010	TB_CCR2_INT	
	INT2_SEL = 1011	CDR_CLK_OUT	CDR 中断		INT3_SEL = 1011	CDR_CLK_OUT	CDR 中断
	INT2_SEL = 1100	D0	I/O 输入中断		INT3_SEL = 1100	D4	I/O 输入中断
	INT2_SEL = 1101	D1			INT3_SEL = 1101	D5	
	INT2_SEL = 1110	D2			INT3_SEL = 1110	D6	
	INT2_SEL = 1111	D3			INT3_SEL = 1111	D7	
INT4	INT4_SEL = 0000	TA_TMR_INT	Timer A 中断	INT5	INT5_SEL = 0000	TA_TMR_INT	Timer A 中断
	INT4_SEL = 0001	TA_CCR0_INT			INT5_SEL = 0001	TA_CCR0_INT	
	INT4_SEL = 0010	TA_CCR1_INT			INT5_SEL = 0010	TA_CCR1_INT	
	INT4_SEL = 0011	TA_CCR2_INT			INT5_SEL = 0011	TA_CCR2_INT	
	INT4_SEL = 0100	TX_SYM_EMPTY	发射机中断		INT5_SEL = 0100	TX_SYM_EMPTY	发射机中断
	INT4_SEL = 0101	SPI_TXMTY	SPI 中断		INT5_SEL = 0101	COMP0_OUT	比较器中断
	INT4_SEL = 0110	SPI_RXNMTY			INT5_SEL = 0110	COMP1_OUT	
	INT4_SEL = 0111	TB_TMR_INT	Timer B 中断		INT5_SEL = 0111	TB_TMR_INT	Timer B 中断
	INT4_SEL = 1000	TB_CCR0_INT			INT5_SEL = 1000	TB_CCR0_INT	
	INT4_SEL = 1001	TB_CCR1_INT			INT5_SEL = 1001	TB_CCR1_INT	
	INT4_SEL = 1010	TB_CCR2_INT			INT5_SEL = 1010	TB_CCR2_INT	
	INT4_SEL = 1011	CDR_CLK_OUT	CDR 中断		INT5_SEL = 1011	CDR_CLK_OUT	CDR 中断
	INT4_SEL = 1100	D8	I/O 输入中断		INT5_SEL = 1100	D0	I/O 输入中断
	INT4_SEL = 1101	D9			INT5_SEL = 1101	D1	
	INT4_SEL = 1110	D10			INT5_SEL = 1110	D2	
	INT4_SEL = 1111	D11			INT5_SEL = 1111	D3	
INT6	INT6_SEL = 0000	TA_TMR_INT	Timer A 中断	INT7	INT7_SEL = 0000	TA_TMR_INT	Timer A 中断
	INT6_SEL = 0001	TA_CCR0_INT			INT7_SEL = 0001	TA_CCR0_INT	

中断	中断选择	中断源	说明	中断	中断选择	中断源	说明
	INT6_SEL = 0010	TA_CCR1_INT			INT7_SEL = 0010	TA_CCR1_INT	
	INT6_SEL = 0011	TA_CCR2_INT			INT7_SEL = 0011	TA_CCR2_INT	
	INT6_SEL = 0100	TX_SYM_EMPTY	发射机中断		INT7_SEL = 0100	TX_SYM_EMPTY	发射机中断
	INT6_SEL = 0101	SPI_TXMTY	SPI 中断		INT7_SEL = 0101	COMP0_OUT	比较器中断
	INT6_SEL = 0110	SPI_RXNMTY			INT7_SEL = 0110	COMP1_OUT	
	INT6_SEL = 0111	TB_TMR_INT	Timer B 中断		INT7_SEL = 0111	TB_TMR_INT	Timer B 中断
	INT6_SEL = 1000	TB_CCR0_INT			INT7_SEL = 1000	TB_CCR0_INT	
	INT6_SEL = 1001	TB_CCR1_INT			INT7_SEL = 1001	TB_CCR1_INT	
	INT6_SEL = 1010	TB_CCR2_INT			INT7_SEL = 1010	TB_CCR2_INT	
	INT6_SEL = 1011	CDR_CLK_OUT	CDR 中断		INT7_SEL = 1011	CDR_CLK_OUT	CDR 中断
	INT6_SEL = 1100	D4	I/O 输入中断		INT7_SEL = 1100	D8	I/O 输入中断
	INT6_SEL = 1101	D5			INT7_SEL = 1101	D9	
	INT6_SEL = 1110	D6			INT7_SEL = 1110	D10	
	INT6_SEL = 1111	D7			INT7_SEL = 1111	D11	

外部中断 INT0 和 INT1 支持电平和边缘两种中断触发方式，INT2 - INT7 则只支持边缘触发方式。中断触发的极性是用户可以通过配置相关 SFR 选择的，电平触发可以选择为高电平或低电平触发，边缘触发可以选择为上升沿或下降沿触发。

8.5 相关寄存器

表 8-3. 系统相关寄存器组列表

名称	SFR 页	地址	默认值	功能
TCON	0	0x88	0x00	Timer1 控制寄存器
IEN0	0	0xA8	0x00	中断使能寄存器 0
IPL0	0	0xB8	0x00	中断优先级寄存器 0
IEN1	0	0xE6	0x00	中断使能寄存器 1
IRCON1	0	0xF1	0x00	外设中断请求标志寄存器
IPL1	0	0xF6	0x00	中断优先级寄存器 1
INTCTL_0	0	0x92	0x00	INT0 - INT7 中断极性选择寄存器
INTCTL_1	0	0x93	0x00	INT2 和 INT3 映射配置寄存器
INTCTL_2	0	0x94	0x00	INT4 和 INT5 映射配置寄存器
INTCTL_3	0	0x95	0x00	INT6 和 INT7 映射配置寄存器

9 GPIO 模块

9.1 基本功能

CMT2187A 系列芯片，最多支持 6 个 GPIO，分别为 GPIO0, GPIO1, GPIO6, GPIO7, GPIO8, GPIO10。如下工作模式：

表 9-1. GPIO 的各种工作模式

属性 1	属性 2	工作模式
数字端口[2]	输入模式[3]	仅为输入模式（悬空输入）
		带上拉输入模式[4]
		带下拉输入模式
	输出模式	开漏输出
		推挽输出

注：

[1]. 模拟输入端口作为两个模拟比较器的输入；

[2]. 作为数字端口时用 D 表示，例如 D1，D2 等，标号与 GPIO 的序号意义对应；

[3]. 只有作为输入模式的 GPIO，可以选择开启 I/O 电平变化检测功能；

[4]. 片内提供可开关的上拉或下拉可供选择，该上下拉的典型为 50kΩ；同时上拉部分还可以提供一个极弱上拉电阻，典型为 500 kΩ。

9.2 GPIO 结构总体介绍

D0 至 D11 功能框图如下所示：

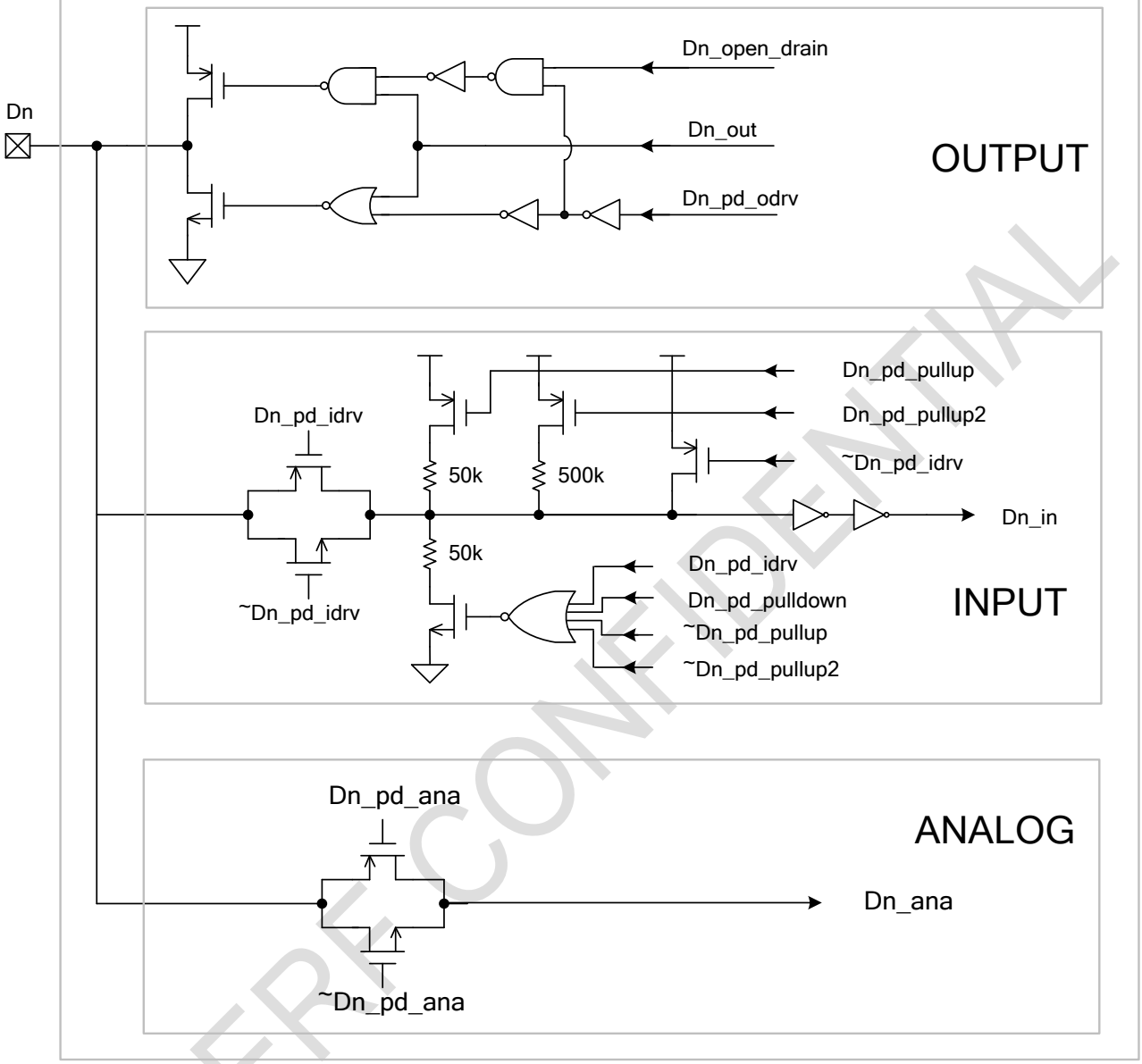

图 9-1. GPIO 功能框图

表 9-2. 图 9-1 图例说明

端口名称	信号类型	描述
Dn	通用 IO PAD	
Dn_open_drain	配置寄存器信号	0x19 寄存器 AON_REG_19 和 0x1A 寄存器 AON_REG_1A 里，配置相对应的 Dn open drain 值。 0: open_drain, 1: push_pull;
Dn_out	系统内部控制信号	当 Dn 作为数字输出模式时，Dn_out 为内部输出信号；
Dn_pd_odrv	系统内部控制信号	当 Dn 作为数字输出模式时，Dn_pd_odrv = 0，否则 Dn_pd_odrv = 1;
Dn_pd_idrv	系统内部控制信号	当 Dn 作为数字输入模式时，Dn_pd_idrv = 0，否则 Dn_pd_idrv = 1;

端口名称	信号类型	描述
Dn_pd_pullup	配置寄存器信号	当 Dn 作为数字输入模式时，通过 0x15 寄存器 AON_REG_15 和 0x16 寄存器 AON_REG_16 配置相应寄存器，独立控制各个 Dn 的 50Kohm 上拉电阻的使能与否。 Dn_pd_pullup， 0: 使能， 1: 不使能；
Dn_pd_pullup2	系统内部控制信号	当 Dn 作为数字输入模式时，通过 0x10 寄存器 AON_REG_10[5]的 pd_pullup_500K 可配置所有数字输入 Dn 的 500Kohm 弱上拉电阻使能与否。0: 使能， 1: 不使能；
Dn_pd_pulldown	系统内部控制信号	当 Dn 作为数字输入模式时，通过 0x17 寄存器 AON_REG_17 和 0x18 寄存器 AON_REG_18 配置相应寄存器，独立控制各个 Dn 的 50Kohm 下拉电阻的使能与否。 Dn_pd_pulldown， 0: 使能， 1: 不使能； 注意：下拉电阻与上拉电阻同时使能时，上拉电阻具有更高的优先级。
Dn_pd_ana	系统内部控制信号	当 Dn 作为模拟输入输出模式时，通过 0x10 寄存器 AON_REG_10[4:0]
Dn_ana	系统内部控制信号	连接到内部的模拟信号线
注：		

9.3 GPIO 数字输入

当 GPIO 被配置为数字输入时：

- 输出部分则不使能；
- 上拉/下拉电阻是否使能取决于 AON 寄存器中与 IO 上下拉相关的配置；
- IO 上的电压被采样至 GPIO_IN_0 和 GPIO_IN_1 的 SFR 寄存器，软件可读。

9.4 GPIO 数字输出

当 GPIO 被配置为输出时：

- 输出通道被使能；
- 开漏模式（open drain output mode）：
  - 若输出寄存器为 0，则输出 NMOS 被使能；
  - 若输出寄存器为 1，则输出 NMOS 和 PMOS 都不被使能，GPIO 为高阻态；
- 推挽输出模式（push-pull output mode）：
  - 若输出寄存器为 0，则输出 NMOS 被使能，输出 PMOS 不被使能；
  - 若输出寄存器为 1，则输出 NMOS 不被使能，输出 PMOS 被使能；
- 在输出模式下，输入模式不使能，输入信号在内部被上拉，因此 GPIOn_in 读取为 1。

9.5 GPIO 模拟输入输出

当 GPIO 被配置为模拟模式时：

- 数字输出功能禁用；
- 数字输入模式禁用，输入信号在内部被强制上拉，因此 GPIOn_in 读取为 1。

9.6 GPIO 数字输入映射

GPIO0 - GPIO11 作为数字输入模式时，简称为 D0 - D11，在经过系统时钟 SYS_CLK 同步前可用于 I/O 电平翻转检测产生中断，在同步后用作下面三种用途：

- 作为各个外设的外部输入
- 作为外部中断 INT2 - INT7 的输入源（在外部中断映射章节已介绍）
- 作为 GPIO_IN_SFR<11:0>，用户可通过 GPIO_IN_0<7:0>和 GPIO_IN_1<7:0>两个 SFR 读取

下图以外设 Timer 0 的外部输入选择配置 t0_gpio_sel<3:0>为例具说明：

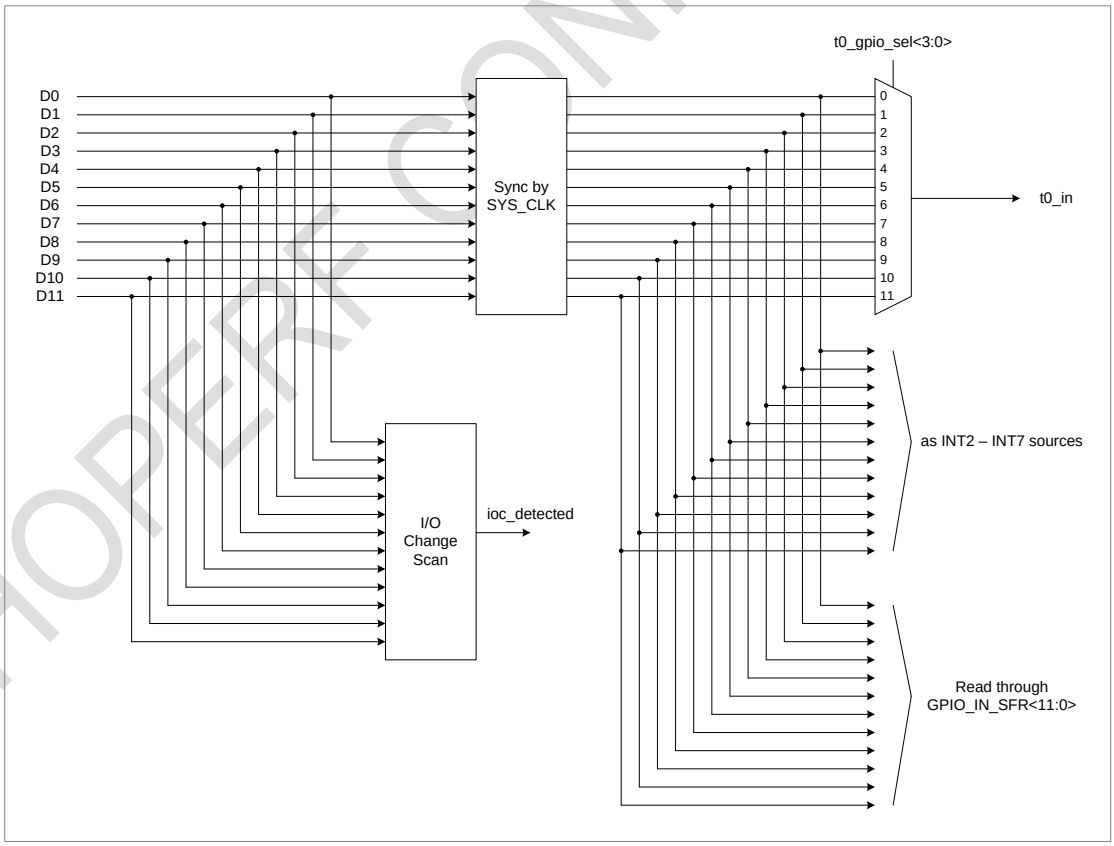

图 9-2. GPIO 作为数字输入的功能框图

可见用户可以通过配置 t0_gpio_sel<3:0>这个寄存器，从 D0 - D11 中任意选择一个经过 SYS_CLK 同步的数字输入信号输送到 Timer 0 的外部输入端 t0_in。像这样的 MUX 一共有 14 个，如下表所示：

表 9-3. 各外设的 GPIO 输入映射寄存器

外设模块	SFR	MUX 选择信号	MUX 输出	MUX 输出用途
Timer 0	GPIO_INC_SEL	T0_GPIO_SEL<3:0>	t0_in	8051 内核原生外设 Timer 0 的外部信号输入
		T0_INTN_GPIO_SEL<3:0>	t0_int0_n	8051 内核原生外设 Timer 0 的计数门控输入
Timer 1	GPIO_IND_SEL	T1_GPIO_SEL<3:0>	t1_in	8051 内核原生外设 Timer 1 的外部信号输入
		T1_INTN_GPIO_SEL<3:0>	t1_int0_n	8051 内核原生外设 Timer 1 的计数门控输入
Timer A	GPIO_INF_SEL	TA_CCI0_GPIO_SEL<3:0>	ta_cci0_in	Timer A 外部捕获源之一
		TA_CCI1_GPIO_SEL<3:0>	ta_cci1_in	Timer A 外部捕获源之一
Timer B	GPIO_ING_SEL	TB_CCI0_GPIO_SEL<3:0>	tb_cci0_in	Timer B 外部捕获源之一
		TB_CCI1_GPIO_SEL<3:0>	tb_cci1_in	Timer B 外部捕获源之一
SPI	GPIO_INA_SEL	NSS_IN_GPIO_SEL<3:0>	nss_in	SPI 从模式的片选输入
		SCK_IN_GPIO_SEL<3:0>	sck_in	SPI 从模式的时钟输入
	GPIO_INB_SEL	MISO_IN_GPIO_SEL<3:0>	miso_in	SPI 从模式的数据输入
		MOSI_IN_GPIO_SEL<3:0>	mosi_int	SPI 主模式的数据输出
UART	GPIO_INE_SEL	RXD0_GPIO_SEL<3:0>	rxd0_in	8051 内核原生外设 UART 的外部信号输入
CDR		CDR_GPIO_SEL<3:0>	cdr_in	CDR 的外部信号输入

注：
[1] 同一个 GPIO 输入可以同时作为外设外部输入，中断源，以及映射到 SFR，用户需要通过适当的配置避免功能冲突。

9.7 GPIO 数字输出映射

当 GPIO0 - GPIO11 作为数字输出时，输出的信号源可以通过配置 SFR 从下表中选择：

表 9-4. 可选择从 GPIOn 输出的信号源

选择项	功能
gpio_out_sfr[n]	GPIO_OUT_0 和 GPIO_OUT_1 寄存器组
port0_out[n]	Port0[7:0]输出
Port1_out[n]	Port1[3:0]输出
ta_out0	TimerA 捕获/比较模块 0 的输出
ta_out1	TimerA 捕获/比较模块 1 的输出
ta_out2	TimerA 捕获/比较模块 2 的输出
tb_out0	TimerB 捕获/比较模块 0 的输出
tb_out1	TimerB 捕获/比较模块 1 的输出
tb_out2	TimerB 捕获/比较模块 2 的输出
sck_out	SPI 主模式的时钟输出
nss_out	SPI 主模式的片选输出

选择项	功能
mosi_out	SPI 主模式的数据输出
miso_out	SPI 从模式的数据输入
csb_out	CMT 专用 4 线 SPI 主模式的访问寄存器片选输出
fcsb_out	CMT 专用 4 线 SPI 主模式的访问 FIFO 片选输出
rxd0_out	UART 的输出使能信号
txd0_out	UART 的时钟或数据输出
T0_ov	Timer0 模块溢出的标志输出
t1_ov	Timer1 模块溢出的标志输出

下表是 GPIOn 与各输出信号源之间的映射关系：

表 9-5. GPIOn 与功能模块输出的映射列表

GPIOn	SFR	选择信号及码值	输出信号源
GPIO0	GPIO_OUTA_SEL	GPIO0_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<0>
		GPIO0_OUT_SEL<3:0> = 4'd1	port0_out<0>
		GPIO0_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO0_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO0_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO0_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO0_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO0_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO0_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO0_OUT_SEL<3:0> = 4'd9	fcsb_out
		GPIO0_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO0_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO0_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO0_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO0_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO0_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO1	GPIO_OUTA_SEL	GPIO1_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<1>
		GPIO1_OUT_SEL<3:0> = 4'd1	port0_out<1>
		GPIO1_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO1_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO1_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO1_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO1_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO1_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO1_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO1_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO1_OUT_SEL<3:0> = 4'd10	csb_out

GPIOn	SFR	选择信号及码值	输出信号源
		GPIO1_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO1_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO1_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO1_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO1_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO2	GPIO_OUTB_SEL	GPIO2_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<2>
		GPIO2_OUT_SEL<3:0> = 4'd1	port0_out<2>
		GPIO2_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO2_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO2_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO2_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO2_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO2_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO2_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO2_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO2_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO2_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO2_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO2_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO2_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO2_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO3	GPIO_OUTB_SEL	GPIO3_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<3>
		GPIO3_OUT_SEL<3:0> = 4'd1	port0_out<3>
		GPIO3_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO3_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO3_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO3_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO3_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO3_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO3_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO3_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO3_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO3_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO3_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO3_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO3_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO3_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO4	GPIO_OUTB_SEL	GPIO4_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<4>
		GPIO4_OUT_SEL<3:0> = 4'd1	port0_out<4>
		GPIO4_OUT_SEL<3:0> = 4'd2	tb_ccr0_out

GPIOn	SFR	选择信号及码值	输出信号源
		GPIO4_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO4_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO4_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO4_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO4_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO4_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO4_OUT_SEL<3:0> = 4'd9	fcsb_out
		GPIO4_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO4_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO4_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO4_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO4_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO4_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO5	GPIO_OUTC_SEL	GPIO5_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<5>
		GPIO5_OUT_SEL<3:0> = 4'd1	port0_out<5>
		GPIO5_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO5_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO5_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO5_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO5_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO5_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO5_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO5_OUT_SEL<3:0> = 4'd9	fcsb_out
		GPIO5_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO5_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO5_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO5_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO5_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO5_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO6	GPIO_OUTD_SEL	GPIO6_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<6>
		GPIO6_OUT_SEL<3:0> = 4'd1	port0_out<6>
		GPIO6_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO6_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO6_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO6_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO6_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO6_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO6_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO6_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO6_OUT_SEL<3:0> = 4'd10	txd0_out

GPIOn	SFR	选择信号及码值	输出信号源
		GPIO6_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO6_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO6_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO6_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO6_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO7	GPIO_OUTD_SEL	GPIO7_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<7>
		GPIO7_OUT_SEL<3:0> = 4'd1	port0_out<7>
		GPIO7_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO7_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO7_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO7_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO7_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO7_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO7_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO7_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO7_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO7_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO7_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO7_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO7_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO7_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO8	GPIO_OUTE_SEL	GPIO8_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<8>
		GPIO8_OUT_SEL<3:0> = 4'd1	port1_out<0>
		GPIO8_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO8_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO8_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO8_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO8_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO8_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO8_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO8_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO8_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO8_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO8_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO8_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO8_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO8_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO9	GPIO_OUTE_SEL	GPIO9_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<9>
		GPIO9_OUT_SEL<3:0> = 4'd1	port1_out<1>
		GPIO9_OUT_SEL<3:0> = 4'd2	tb_ccr0_out

GPIOn	SFR	选择信号及码值	输出信号源
		GPIO9_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO9_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO9_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO9_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO9_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO9_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO9_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO9_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO9_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO9_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO9_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO9_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO9_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO10	GPIO_OUTF_SEL	GPIO10_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<10>
		GPIO10_OUT_SEL<3:0> = 4'd1	port0_out<2>
		GPIO10_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO10_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO10_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO10_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO10_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO10_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO10_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO10_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO10_OUT_SEL<3:0> = 4'd10	txd0_out
		GPIO10_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO10_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO10_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO10_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO10_OUT_SEL<3:0> = 4'd15	t1_ov_out
GPIO11	GPIO_OUTF_SEL	GPIO11_OUT_SEL<3:0> = 4'd0	gpio_out_sfr<11>
		GPIO11_OUT_SEL<3:0> = 4'd1	port0_out<3>
		GPIO11_OUT_SEL<3:0> = 4'd2	tb_ccr0_out
		GPIO11_OUT_SEL<3:0> = 4'd3	tb_cc1_out
		GPIO11_OUT_SEL<3:0> = 4'd4	tb_ccr2_out
		GPIO11_OUT_SEL<3:0> = 4'd5	nss_out
		GPIO11_OUT_SEL<3:0> = 4'd6	sck_out
		GPIO11_OUT_SEL<3:0> = 4'd7	miso_out
		GPIO11_OUT_SEL<3:0> = 4'd8	mosi_out
		GPIO11_OUT_SEL<3:0> = 4'd9	rxd0_out
		GPIO11_OUT_SEL<3:0> = 4'd10	txd0_out

GPIOn	SFR	选择信号及码值	输出信号源
		GPIO11_OUT_SEL<3:0> = 4'd11	ta_ccr0_out
		GPIO11_OUT_SEL<3:0> = 4'd12	ta_cc1_out
		GPIO11_OUT_SEL<3:0> = 4'd13	ta_ccr2_out
		GPIO11_OUT_SEL<3:0> = 4'd14	t0_ov_out
		GPIO11_OUT_SEL<3:0> = 4'd15	t1_ov_out
注： [1] 默认映射是通过 GPIO_OUT_SFR<11:0>寄存器组进行控制，但该寄存器组不支持位访问模式，所以控制输出时需要遵循“读-改-写”的方式；			

9.8 GPIO 电平翻转检测

位于 AON 域的 GPIO 电平翻转检测（I/O Change Scan）模块，用于系统进入 STOP 模式后，检测任意 I/O 的电平翻转，产生唤醒系统的中断源并送到外部中断 INT0。当系统正常运行时，用户需要关闭该模块，以免与其它 IO 功能冲突。

如要使用该模块，用户程序的处理流程如下：

1. 该模块仅在 STOP 时才能打开，在程序运行时维持 SFR PAGE0 中的 WKINT_STA 寄存器的 IO_EVENT_RST_N 比特为 0，即整个模块处于复位状态；
2. 选择好需要检测的 GPIO，并开始使用 AON REG 对 GPIO 和检测模块进行配置，由于模块处于复位，因此配置过程不会造成误检测；
3. 配置该 GPIO 必须为数字输入模式，此时数字输出功能禁用；
4. 配置该 GPIO 的上拉/下拉电阻；
5. 配置该 GPIO 的翻转极性（上升沿或下降沿）；
6. 使能该 GPIO 的电平翻转检测；
7. 将 IO_EVENT_RST_N 设为 1 释放检测模块，并通过 SFR 配置好 INT0；
8. 系统进入 STOP 模式；
9. 当该 GPIO 检测到电平翻转并通过 INT0 唤醒系统后，在完成相关的 IO 查询和处理后，将 IO_EVEN_RST_N 比特设为 0。

该模块的常见应用场景是 GPIO 连接外部按键，通常由两种连接方法，下面用 D0 - D11 连接按键举例说明：

- 独立按键接法

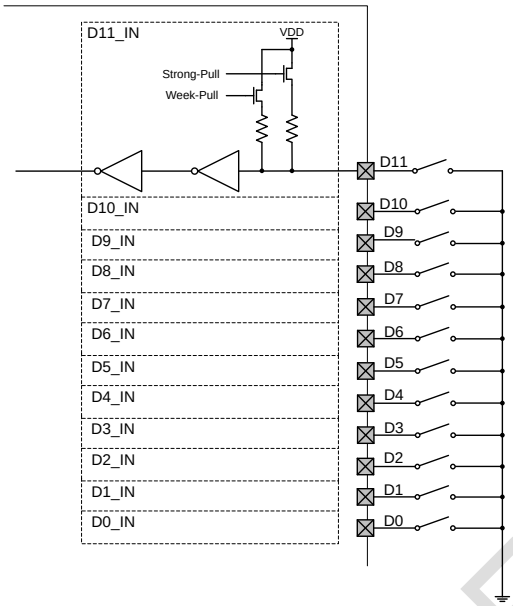

图 9-3. 独立按键接法图示

下图中 D0 - D11 均为连接按键到地，所以这些端口可以全部开启上拉的数字输入端口模式，并将 Dn_POLAR（n 泛指 0-11 任意一个数字，每个 IO 都有对应的极性选择比特）配置 1，即常态为 1，有按键按下时端口 0，检测到下降沿就触发中断。

- 矩阵按键接法

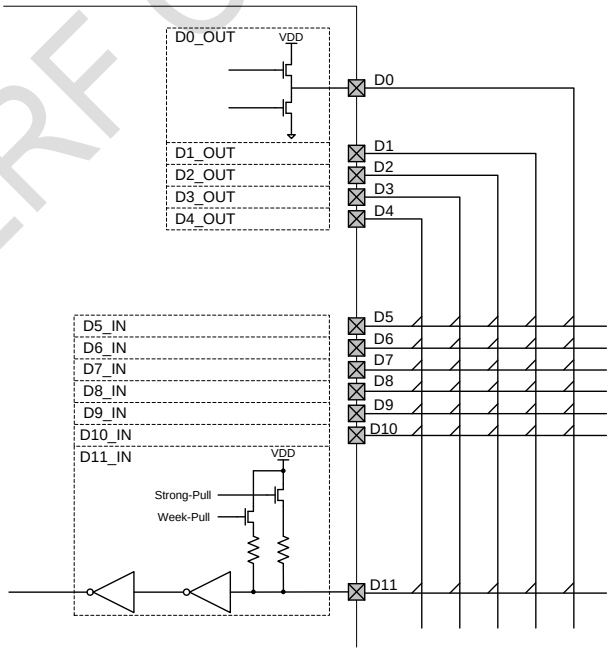

图 9-4. 矩阵按键接法图示

按上图接法，由 D5 - D11 作为一组输入检测，D0 - D4 作为一组输出控制的矩阵键盘。在进入 STOP 模式前，需要把 D0 - D4 配置为数字输出模式，输出 0 值。D5 - D11 这组数字输入，与独立按键一样配置使能检测功能即可。在进入 STOP 模式后，矩阵键盘任意按键都会通过 D5 - D11 唤醒系统。唤醒系统后，软件可以正常扫描键盘流程进行识别按键。

9.9 相关寄存器

表 9-6. GPIO 的 AON 寄存器组列表

名称	地址	默认值	功能
AON_REG_10	0x10	0xE0	使能 GPIO 模拟功能，配置上下拉电阻值
AON_REG_11	0x11	0x00	D0 - D7 的数字输出使能位
AON_REG_12	0x12	0x00	D8 - D11 的数字输出使能位
AON_REG_13	0x13	0x00	D0 - D7 的数字输入使能位
AON_REG_14	0x14	0x00	D8 - D11 的数字输入使能位
AON_REG_15	0x15	0xFF	D0 - D7 的上拉电阻开关
AON_REG_16	0x16	0x0F	D8 - D11 的上拉电阻开关
AON_REG_17	0x17	0xFF	D0 - D7 的下拉电阻开关
AON_REG_18	0x18	0x0F	D8 - D11 的下拉电阻开关
AON_REG_19	0x19	0xFF	D0 - D7 的开漏开关
AON_REG_1A	0x1A	0x0F	D8 - D11 的开漏开关
AON_REG_1B	0x1B	0x00	D0 - D7 的电平翻转检测使能位
AON_REG_1C	0x1C	0x00	D8 - D11 的电平翻转检测使能位
AON_REG_1D	0x1D	0x00	D0 - D7 的电平翻转检测极性选择
AON_REG_1E	0x1E	0x00	D8 - D11 的电平翻转检测极性选择

表 9-7. GPIO 的 SFR 寄存器组列表

名称	SFR 页	地址	默认值	功能
GPIO_INA_SEL	0	0xA1	0x00	GPIO 输入功能映射
GPIO_INB_SEL	0	0xA2	0x00	GPIO 输入功能映射
GPIO_INC_SEL	0	0xA3	0x00	GPIO 输入功能映射
GPIO_IND_SEL	0	0xA4	0x00	GPIO 输入功能映射
GPIO_INE_SEL	0	0xA5	0x00	GPIO 输入功能映射
GPIO_INF_SEL	0	0xA6	0x00	GPIO 输入功能映射
GPIO_ING_SEL	0	0xA9	0x00	GPIO 输入功能映射
GPIO_OUTA_SEL	0	0xAA	0x00	GPIO 输出功能映射
GPIO_OUTB_SEL	0	0xAB	0x00	GPIO 输出功能映射
GPIO_OUTC_SEL	0	0xAC	0x00	GPIO 输出功能映射
GPIO_OUTD_SEL	0	0xAD	0x00	GPIO 输出功能映射
GPIO_OUTE_SEL	0	0xB0	0x00	GPIO 输出功能映射
GPIO_OUTF_SEL	0	0xB1	0x00	GPIO 输出功能映射

名称	SFR 页	地址	默认值	功能
GPIO_OUT_0	0	0xB3	0x00	SFR 配置的 GPIO 输出数据
GPIO_OUT_1	0	0xB4	0x00	SFR 配置的 GPIO 输出数据
GPIO_IN_0	0	0xB5	0x00	从 SFR 可读取的 GPIO 输入数据
GPIO_IN_1	0	0xB6	0x00	从 SFR 可读取的 GPIO 输入数据

10 Timer0 模块

10.1 基本功能

Timer0 是一个 16-Bit 可编程的定时器/计数器，通过配置 TMOD 寄存器可以选择其工作方式、启动或停止计数，以及产生计数溢出中断等。Timer0 具体支持 3 种工作方式，如表 10-1 所示。

表 10-1. Timer0 各种工作模式列表

TMOD. M01	TMOD.M00	工作模式	功能
0	0	Mode 0	带 5 位预分频的 8-Bit 定时/计数器，即 13-Bit 定时/计数模式
0	1	Mode 1	16-Bit 定时/计数模式
1	0	Mode 2	带重载初值的 8-Bit 定时/计数模式
1	1	不工作	

10.2 Timer0 模式 0

Timer0 模式 0 工作框图如下所示。

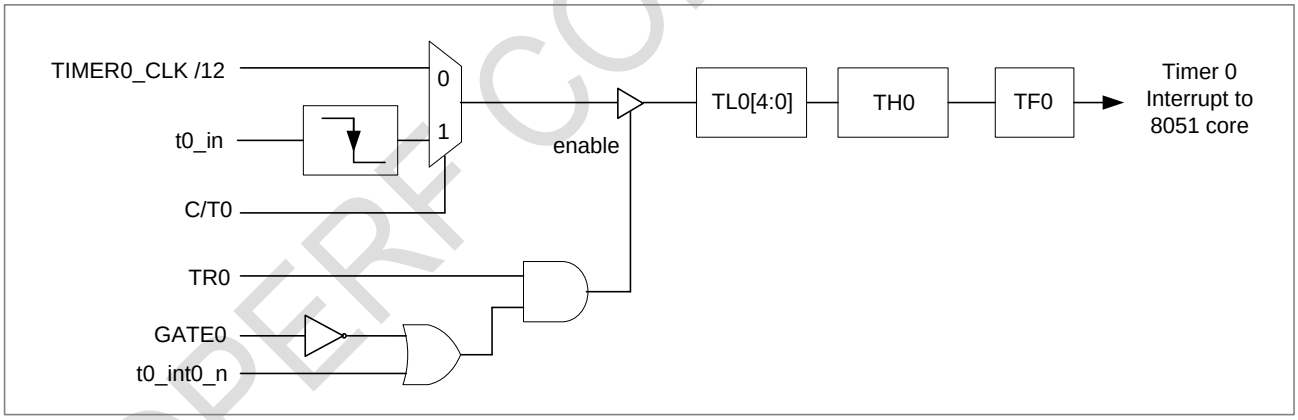

图 10-1. Timer0 Mode0 框图

Timer0 工作在 Mode0 时，由 TL0[4:0]提供 5 位预分频器，TH0 作为 8 位计数器，组成一个 13 位计数器：

- 当 TMOD.C/T0 配置为 0，则为选择定时模式，以 F_{PCLK} 的 12 分频作为计时时钟源；
- 当 TMOD.C/T0 配置为 1，则为选择计数模式，以外部输入管脚 t0 下降沿为计数信号；
- 当 TMOD.GATE0 配置为 1，则为带门控触发计数，需要外部中断 t0_int0_n 的高电平和 TCON.TR0 才能共同触发 Timer0 计数；
- 当 TMOD.GATE0 配置为 0，则只需要软件置 TCON.TR0 为 1，即可触发 Timer0 计数；
- Timer0 计数的总开关为 TCON.TR0，此位置 1 时启动计数功能；置 0 时关闭计数功能。

10.3 Timer0 模式 1

Timer0 模式 1 工作框图如下所示。

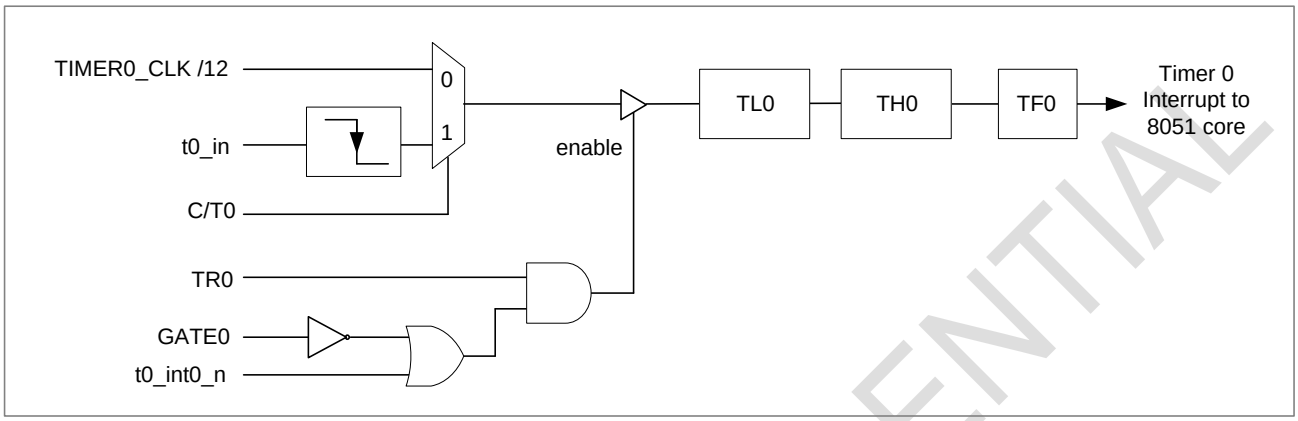

图 10-2. Timer0 Mode1 框图

当 TIMER0 工作在 Mode0 时，为 16 位计数器，由 TL0 和 TH0 构成，与 Mode0 的区别仅在于计数器位数不同。其它控制位功能与 Moode0 相同。

10.4 Timer0 模式 2

Timer0 模式 2 工作框图如下所示。

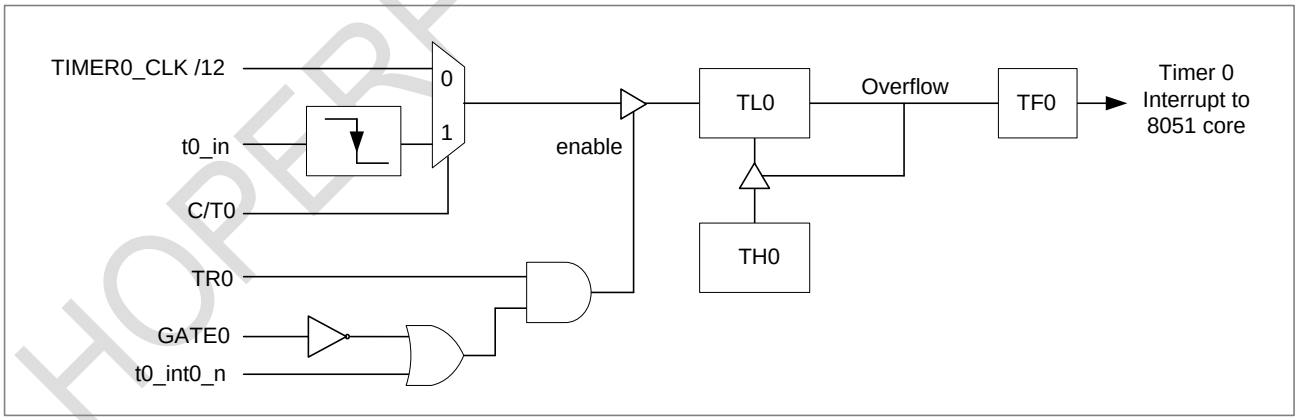

图 10-3. Timer0 Mode2 框图

当 Timer0 工作在 Mode2 时，为 8 位自动重载初值的计数器，当 TL0 计数溢出时，会自动装入 TH0（初值）保存的值，使 TL0 从初值开始重新计数。这是与 Mode0/1 的主要区别，Mode0/1 在计数溢出后，计数器被清零。其它控制位功能与 Moode0/1 相同。

10.5 相关寄存器

表 10-2. Timer0 模块寄存器组列表

名称	SFR 页	地址	默认值	功能
TCON	0	0x88	0x00	Timer0 和 Timer1 控制寄存器
TMOD	0	0x89	0x00	Timer0 和 Timer1 工作模式寄存器
TL0	0	0x8A	0x00	Timer0 寄存器低 8 位
TH0	0	0x8C	0x00	Timer0 寄存器高 8 位

11 Timer1 模块

11.1 基本功能

Timer1 是一个 16-Bit 可编程的定时器/计数器，通过配置 TMOD 寄存器可以选择其工作方式、启动或停止计数，以及产生计数溢出中断等。Timer1 具体支持 3 种工作方式，如表 11-1 所示。

表 11-1. Timer1 各种工作模式列表

TMOD. M11	TMOD.M10	工作模式	功能
0	0	Mode 0	带 5 位预分频的 8-Bit 定时/计数器，即 13-Bit 定时/计数模式
0	1	Mode 1	16-Bit 定时/计数模式
1	0	Mode 2	带重载初值的 8-Bit 定时/计数模式
1	1	不工作	

11.2 Timer1 模式 0

Timer1 模式 0 工作框图如下所示。

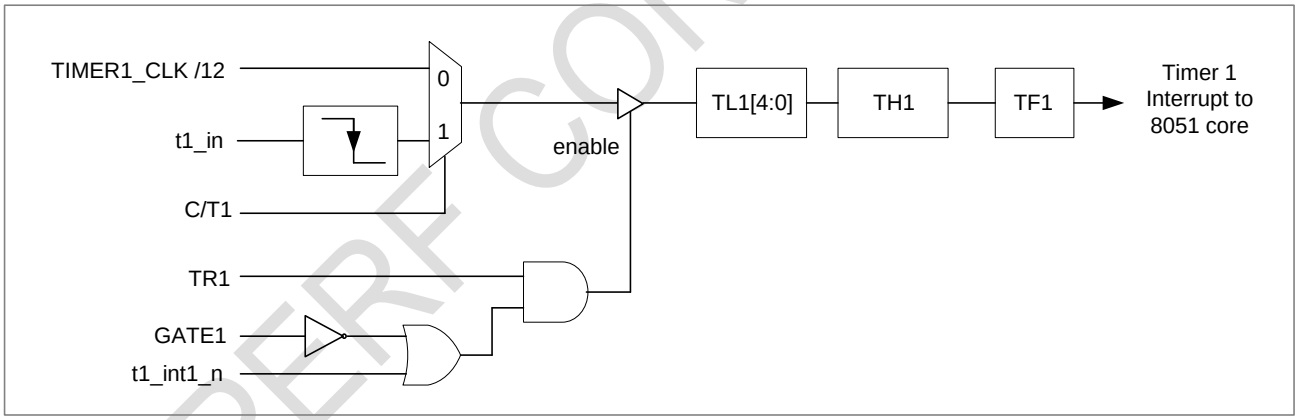

图 11-1. Timer1 Mode0 框图

Timer1 工作在 Mode0 时，由 TL1[4:0]提供 5 位预分频器，TH1 作为 8 位计数器，组成一个 13 位计数器：

- 当 TMOD.C/T1 配置为 0，则为选择定时模式，以 F_{PCLK} 的 12 分频作为计时时钟源；
- 当 TMOD.C/T1 配置为 1，则为选择计数模式，以外部输入管脚 t1 下降沿为计数信号；
- 当 TMOD.GATE1 配置为 1，则为带门控触发计数，需要外部中断 t1_int1_n 的高电平和 TCON.TR1 才能共同触发 Timer1 计数；
- 当 TMOD.GATE1 配置为 0，则只需要软件置 TCON.TR1 为 1，即可触发 Timer1 计数；
- Timer1 计数的总开关为 TCON.TR1，此位置 1 时启动计数功能；置 0 时关闭计数功能。

11.3 Timer1 模式 1

Timer1 模式 1 工作框图如下所示。

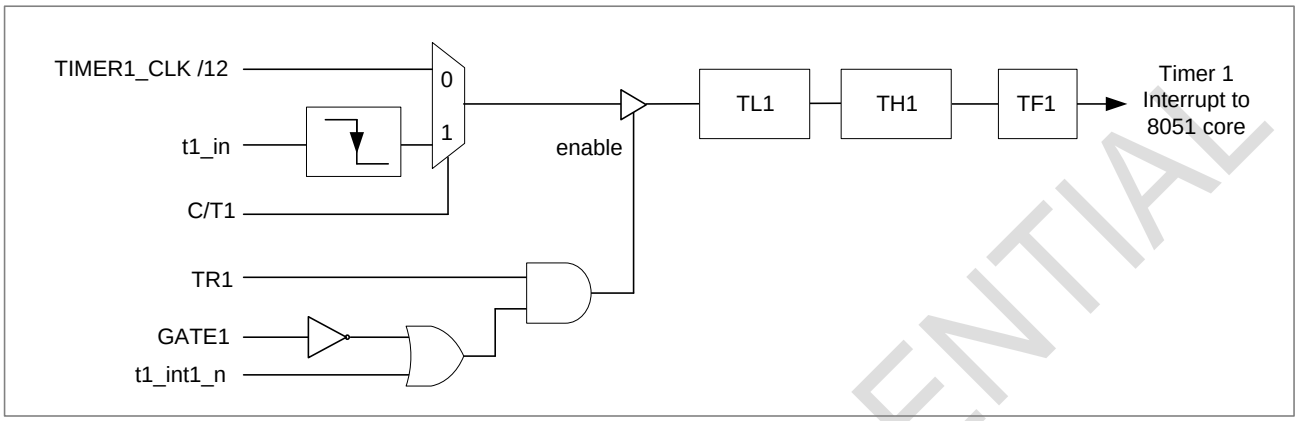

图 11-2. Timer1 Mode1 框图

当 TIMER1 工作在 Mode1 时，为 16 位计数器，由 TL1 和 TH1 构成，与 Mode0 的区别仅在于计数器位数不同。其它控制位功能与 Moode0 相同。

11.4 Timer1 模式 2

Timer1 模式 2 工作框图如下所示。

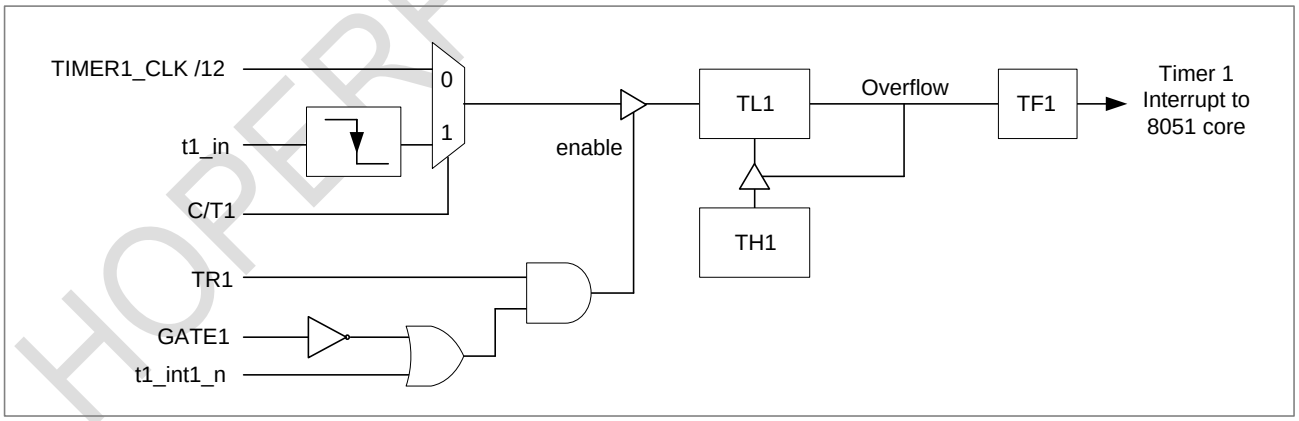

图 11-3. Timer1 Mode2 框图

当 Timer1 工作在 Mode2 时，为 8 位自动重载初值的计数器，当 TL1 计数溢出时，会自动装入 TH1（初值）保存的值，使 TL1 从初值开始重新计数。这是与 Mode0/1 的主要区别，Mode0/1 在计数溢出后，计数器被清零。其它控制位功能与 Moode0/1 相同。

11.5 相关寄存器

表 11-2. Timer1 模块寄存器组列表

名称	SFR 页	地址	默认值	功能
TCON	0	0x88	0x00	Timer0 和 Timer1 控制寄存器
TMOD	0	0x89	0x00	Timer0 和 Timer1 工作模式寄存器
TL1	0	0x8B	0x00	Timer1 寄存器低 8 位
TH1	0	0x8D	0x00	Timer1 寄存器高 8 位
USART_SEL	0	0x9F	0x01	Timer1 时钟源分频器选择

12 SPI 模块

12.1 基本功能

串行外设接口（Serial peripheral interface，简称 SPI）允许芯片与外设以半/全双工、同步、串行方式通信。从工作形态区分，有分主从设备模式，通信时钟由主设备提供给从设备。该接口还支持多主配置方式工作，或使用单线双向数据线的双线单工同步传输（即 3 线制模式）。

通常 SPI 的主从设备之间需要连接 4 个管脚：

- MISO：Master In Slave Out，主设备输入/从设备输出管脚。该管脚实现从设备往主设备上发数据。
- MOSI：Master Out Slave In，主设备输出/从设备输入管脚。该管脚实现主设备往从设备下发数据。
- SCK：串口数据同步时钟，由主设备输出给从设备。
- NSS：从设备片选使能，这是一个可选的管脚，由主设备来选择目标从设备。它的功能是让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。从设备的 NSS 管脚可以由主设备当作一个标准的 IO 来驱动。一旦被使能（SSOE 位）置位，NSS 管脚也可以作为输出管脚，并在 SPI 设置为主模式时拉低；此时，所有 NSS 管脚连接到主设备 NSS 管脚的 SPI 设备，会检测到低电平，如果它们被设置为 NSS 硬件模式，就会自动进入从设备状态。

图 12-1 和 12-2，给出主/从设备的 SPI 接口时序图。其中，SCK 的采样时钟相位 CPHA 由寄存器 SPI_CTL_1.SPI_EDGE_SEL 配置，若其为 1，SPI 将在时钟的第一个边沿发送数据，第二个边沿采样数据，置 0 则是在时钟的第一个边沿采样数据，第二个边沿发送。CPOL 决定 SCK 空闲时候的状态，CPOL 为 1 时 SCK 线会在空闲时保持高电平，CPOL 为 0 时 SCK 线则会在空闲时保持低电平。

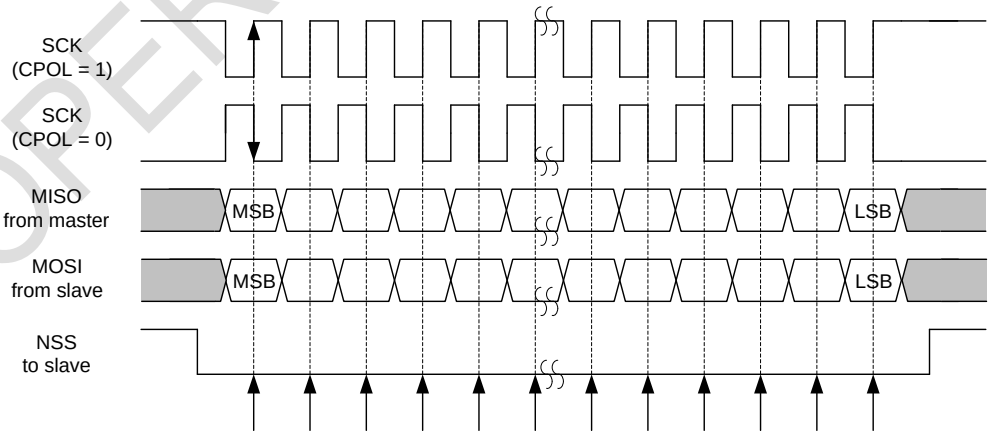

图 12-1. SPI 接口时序图（CPHA = 1）

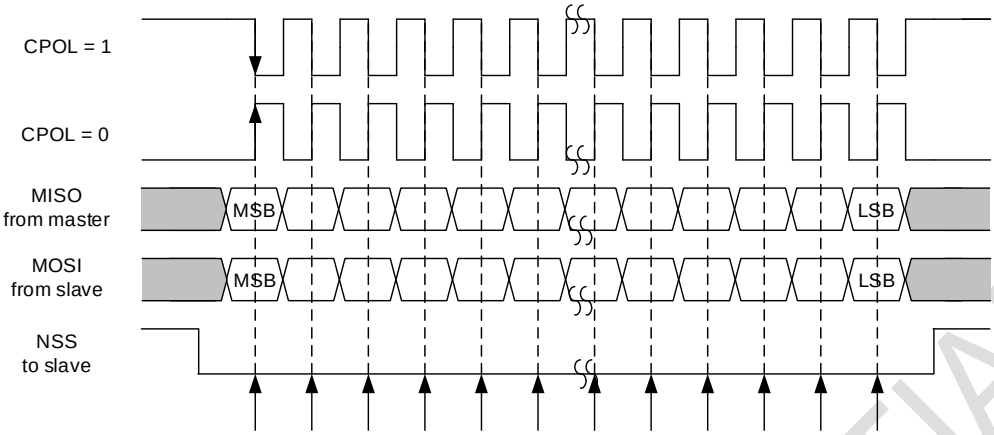

图 12-2. SPI 接口时序图（CPHA = 0）

12.2 配置选项

- **主机模式与从机模式的设置**

本 SPI 模块中包含了主机模式和从机模式的完整程序，用户可通过配置寄存器 SPI_CTL_1.SPI_MS_SEL 来将当前操控的 SPI 设置成主机或是从机，以便和其他设备的从机或主机进行通讯。若 SPI_MS_SEL 配置为 1'b1，当前 SPI 模块会被设置为主机模式（master mode），若 SPI_MS_SELl 配置为 1'b0，当前 SPI 模块则会被设置为从机模式（slave mode）。

- **时钟相位和时钟极性的配置**

要设置时钟相位，需要配置变量 SPI_CTL_1.SPI_EDGE_SEL，若该变量置 1，SPI 将在时钟的第二个边沿采样，置 0 则是在时钟的第一个边沿进行采样。

时钟极性可通过配置寄存器 SPI_CTL_1.SPI_CKPOL_SEL 来设置，若 SPI_CKPOL_SEL 为 1，那么 IDLE 状态下 SCK 将为高电平，若 SPI_CKPOL_SEL 为 0，IDLE 状态下 SCK 将为低电平。下图为配置不同的时钟极性与相位时数据采样与发送的边沿。

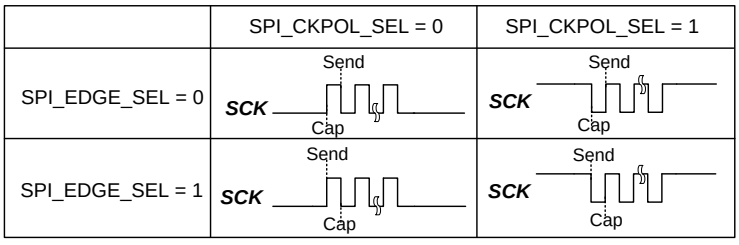

图 12-3. SPI 接口时序图（CPHA = 1）

- **发送数据位宽的设定**

寄存器 SPI_CTL_0.SPI_8B16B_SEL 设定为 1 时，SPI 将选择传输 16bit 的数据，主机端将会传输 16 个周期的时钟给从机，从机也会在相应的时钟沿依次进行 16 个 bit 数据的收集。而当 SPI_8B16B_SEL 设定为 0 时，则只传输 8bit 的数据，并且传输的是 SPI_DATA 中 16 位数据中的低 8 位。

- **波特率的设定**

表 12-1. SPI 模块速率

SPI_CTL_0.SPI_MBR 设定值	传输速率（sys_clk 为系统时钟，默认 24Mhz）
3'b000	sys_clk/4
3'b001	sys_clk/8
3'b010	sys_clk/16
3'b011	sys_clk/24
3'b100	sys_clk/32
3'b101	sys_clk/64
3'b110	sys_clk/128
3'b111	sys_clk/256

- **高低位优先传输的设置**

将输入信号 SPI_CTL_1.SPI_LSBF 设置为 1，将会优先传输低位数据，若设置为 0，则是优先传输高位数据。

12.3 工作模式

无论是主机还是从机。都有四种工作模式，并可分为全双工和半双工两种类别。将 SPI_CTL_0.SPI_BIDI_MODE；SPI_CTL_0.SPI_BIDI_OEN；SPI_CTL_0.SPI_RX_ONLY 三个变量进行如下表配置，可实现四种工作模式。

表 12-2. SPI 模块工作模式

SPI_BIDI_MODE	SPI_RX_ONLY	SPI_BIDI_OEN	模式选择优先级	工作模式
1'b0	1'b0	任意值	第一级	全双工正常模式
1'b0	1'b1	任意值	第二级	全双工只读模式
1'b1	任意值	1'b1	第三级	半双工只写模式
1'b1	任意值	1'b0	第四级	半双工只读模式

1.全双工正常模式：配置了该种模式主机或从机，收发数据是同时进行的。以主机为例，数据发送口 MOSI 向外发送数据，同时 MISO 口也接收从机发送的数据。

2.全双工只读模式：配置了该种模式主机或从机的接收端正常，但发送端将一直发送为 0 的数据。

3.半双工只写模式：该模式下的主机或从机对外只有三个接口，以主机为例，SCK 向外发送时钟，NSS 发送使能，最后一个 IO 端口分配给 MOSI 发送数据。

4.半双工只读模式：该模式下的主机或从机对外只有三个接口，以主机为例，SCK 向外发送时钟，NSS 发送使能，最后一个 IO 端口分配给 MISO 接收数据。

实际应用：

SPI 可以采用半双工的只读和只写来回切换的方式来与 S3S 接口通信。当 SPI 要向 S3S 接口输送地址时，可设置为半双工只写模式，并将 MOSI 映射到 PAD 上与 S3S 的 SDA 连接，实现地址的写入。当 SPI 要读出 S3S 发送的数据时，可将 SPI 设置为半双工只读模式，并将 MISO 映射到 PAD 上与 S3S 的 SDA 连接，以便读出 S3S 传输的数据。这样便可以在只有 3 根物理线连接的情况，实现 SPI 和 S3S 的数据通信。

12.4 状态标志

SPI 模块通过 3 个状态标志，提供总线的状态，以方便软件监控：

- 忙标志（SPI_CTL2.SPI_BUSY 标志位），当 SPI 正处于传输过程中时，该标志位会拉高。
- 发送缓冲器空闲标志（SPI_CTL2.SPI_TXMTY 标志位）；当 SPI 在进行新的传输数据的配置时，该标志位会拉低，当新的发送数据配置完成并成功开始发送后，该标志位会拉高。
- 接收缓冲器非空（SPI_CTL2.SPI_RXNMTY 标志位）；当 SPI 完成一个周期的数据发送之后，会将 SPI_RXNMTY 标志位拉高，表示成功发送并接收到了数据。当用户读取 SPI_DATA 中收到的数据后，该标志位会自动拉低。

12.5 相关寄存器

表 12-3. SPI 模块寄存器组列表

名称	SFR 页	地址	默认值	功能
SPI_CTL_0	Page0	0x96	0x00	SPI 模块控制寄存器 0
SPI_CTL_1	Page0	0x97	0x00	SPI 模块控制寄存器 1
SPI_CTL_2	Page0	0x9A	0x00	SPI 模块控制寄存器 2
SPI_DATA_H	Page0	0x9B	0x00	SPI 模块数据高字节
SPI_DATA_L	Page0	0x9C	0x00	SPI 模块数据低字节

13 UART 模块

13.1 基本功能

CMT2187A 片内 UART 是一个灵活的全双工异步收发器，完全兼容 8051 结构。波特率软件可配置，支持以下 4 种工作方式：

- Mode0：同步移位模式，固定 UART_CLK / 12 波特率；
- Mode1：8-Bit UART 模式，波特率可配置，由内部 TIMER 1 产生；
- Mode2：9-Bit UART 模式，波特率为 UART_CLK / 64 或 UART_CLK / 32；
- Mode3：9-Bit UART 模式，波特率可配置，由内部 TIMER 1 产生；

通过配置 SCON0 可以选择 UART 模块的工作方式、通讯使能、以及发送/接收数据等中断。

13.2 同步移位模式（Mode 0）

同步移位模式是 UART 模块的一种同步工作方式，可以用于类似 SPI 的从机模式，与其他 8051 之间进行串行通讯。该模式是半双功通讯，串口线 RxD 作为输入输出双向数据接口，TxD 则产生固定串行波特率时钟，时钟频率为 UART_CLK / 12。配置 SCON0.SM00 和 SCON0.SM10 为 00 时，UART 模块选择工作在 Mode 0，其发送结构图如下所示。

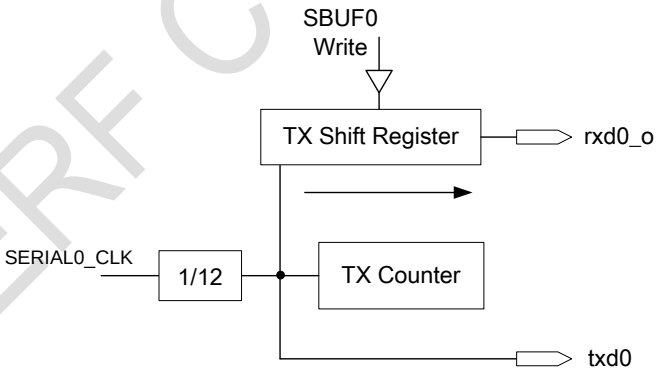

图 13-1. UART Mode 0 发送框图

发送过程中，当将数据写入发送缓冲器 SBUF 时，产生一个正脉冲，串行口开始发送数据，此时 SCON0.REN0 需要为 0。发送完成后，发送中断标志位 SCON0.TI0 置 1，时序图如下。

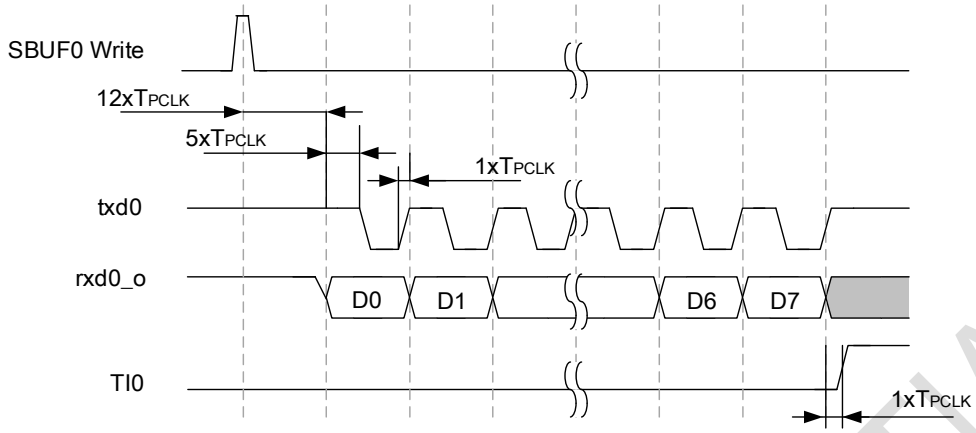

图 13-2. UART Mode 0 发送时序图

Mode0 接收结构图如下所示。

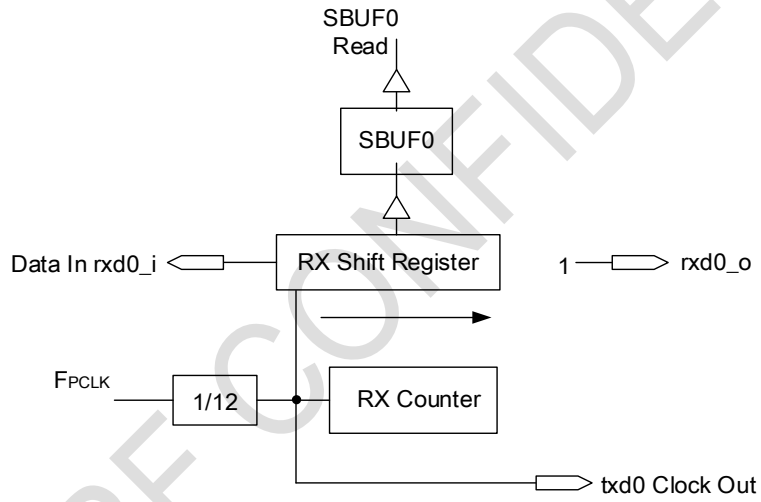

图 13-3. UART Mode 0 接收框图

Mode0 接收时，将接收使能位 SCON0.REN0 置 1，同时清零接收中断标志位 SCON0.RI0 时，产生一个正脉冲，串行口开始接收数据。TxD 输出移位时钟采样 RxD 线上的数据，当接收到 8 位数据后，接收中断标志位 SCON0.RI0 置 1，数据缓存在 SBUF0 中，时序图如下。

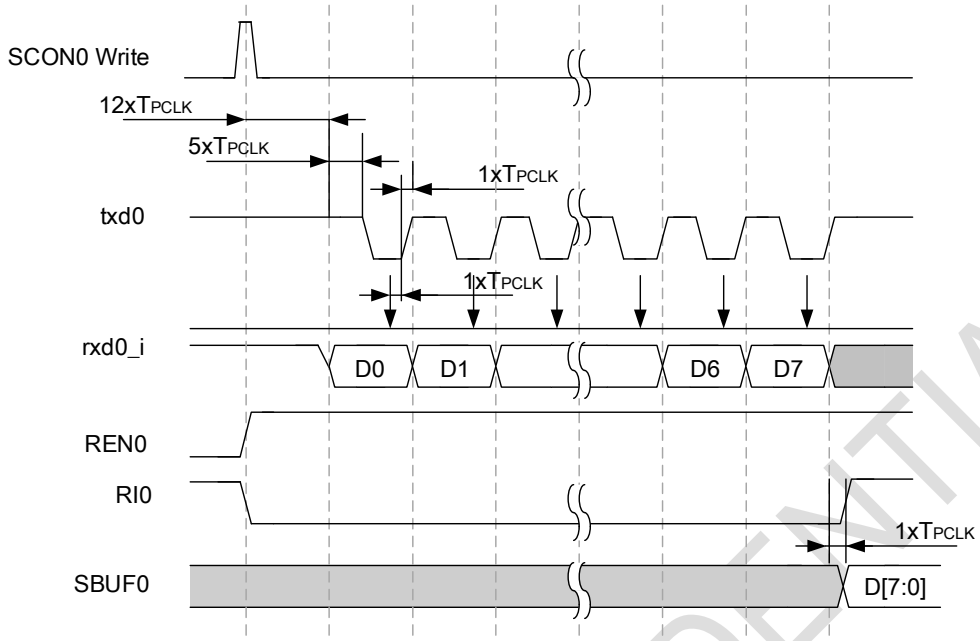

图 13-4. UART Mode 0 接收时序图

13.3 波特率可配置的异步全双工模式（Mode 1 和 Mode 3）

UART 模块的 Mode 1 和 Mode 3 都属于异步全双工收发、可变波特率，区别仅在于 Mode 1 是 8 位数据收发模式，Mode 3 是 9 位数据收发模式。

设置 Serial Port 0 为 Mode1：

- 将 SCON0.SM00 和 SCON0.SM10 配置为 01；
- 通过 Timer1 和 PCON.SMOD1 位产生波特率；
- 接收时，将接收使能位 SCON0.REN0 置 1；
- 发送时，将数据写入发送缓冲器 SBUF。

设置 Serial Port 0 为 Mode3：

- 将 SCON0.SM00 和 SCON0.SM10 配置为 11；
- 通过 Timer1 和 PCON.SMOD1 位产生波特率；
- 接收时，将接收使能位 SCON0.REN0 置 1；
- 发送时，将数据写入发送缓冲器 SBUF 及 SCON0.TB8 位。

Mode 1 和 Mode 3 的帧格式分别如下图所示：

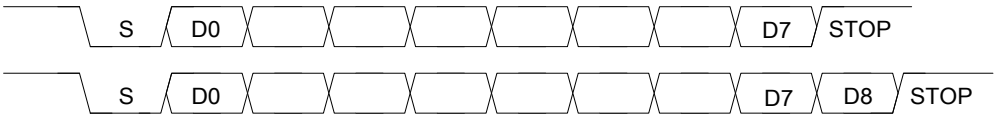

图 13-5. UART Mode 1 和 Mode 3 帧结构图

Mode1 和 Mode3 的波特率由 Timer1 的溢出率决定，其发送和接收框图分别如下：

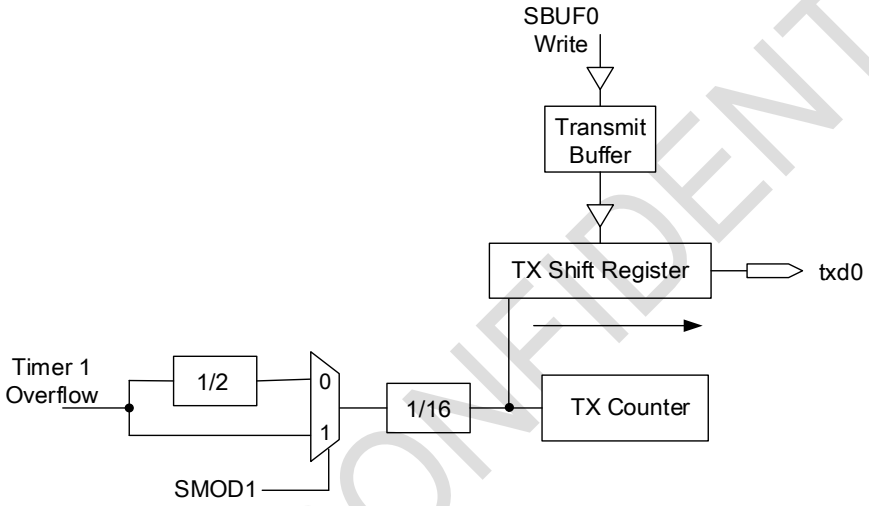

图 13-6. UART Mode1 和 Mode3 发送框图

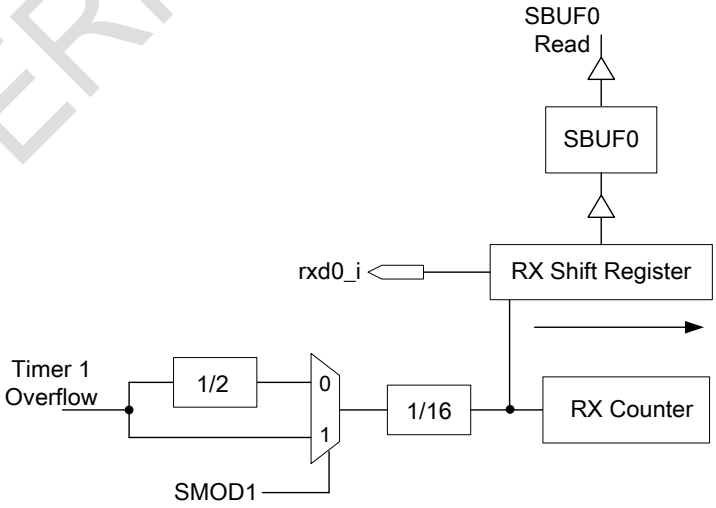

图 13-7. UART Mode1 和 Mode3 接收框图

配置 Timer1 产生波特率：

- TMOD.M1[1:0]配置为 10（Timer1 模式 2），TMOD.GATE1 配置为 0，TMOD.C/T1 配置为 0；
- TH1 写入 8 位计数初值；
- TCON.TR1 配置为 1，开始计时。

波特率由下面公式计算：

$$\text{Baud\_Rate} = 2^{SMOD1} \times \frac{F_{PCLK}}{[32 \times 12 \times (256 - TH1)]}$$

等价于：

$$\text{Baud}_{\text{Rate}} = 2^{SMOD1} \times \frac{Timer1_{Overflow\_Rate}}{32}$$

发送过程中，将 8bit 数据写入发送缓冲器 SBUF0 时（若为 Mode3，先将 SCON0.TB8 写入数据，再将 8bit 数据写入 SBUF0），产生一个正脉冲，串行口开始发送数据，先发送 1bit 起始位 0，再发送 SBUF 的 LSB 位，依次发送 8bit 数据，再发送 SCON0.TB8（Mode3），最后发送 1bit 停止位 1。数据发送结束后，发送中断标志位 SCON0.TI0 置 1。时序图如下：

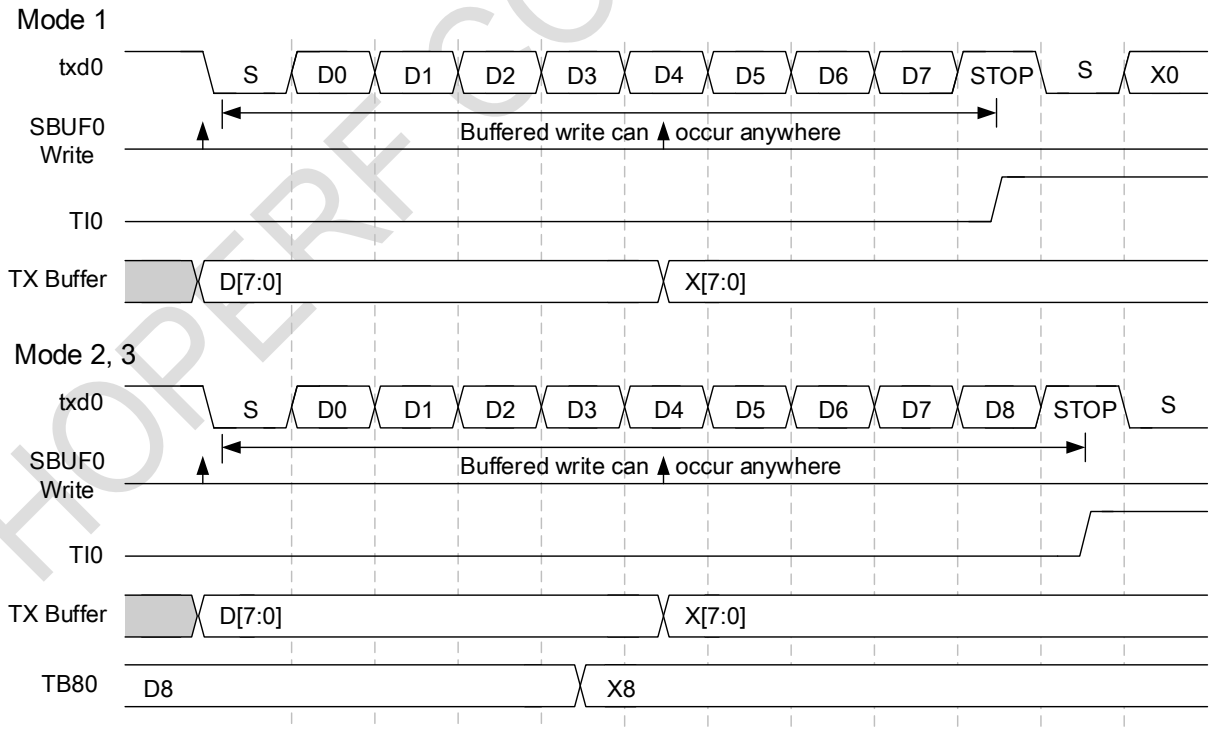

图 13-8. UART 异步发送时序图（Mode 1 和 Mode 3）

使能 UART Mode 1 或 Mode 3 接收模式时，需先将接收使能位 SCON0.REN0 置 1，然后以波特率的 16 倍速率采样 RxD 脚状态，等待起始位的下降沿。当采样到下降沿时，在起始位中间再次检测 RxD 是否为 0，确认是否是有效的起始位。若非 0，则继续采样起始位。当确认为有效起始位后，开始接收 8bit / 9bit 数据，每一位数据都在中间采样。在 8bit 接收中，采样停止位是否有效，并拷贝至 SCON0.RB80，接收中断标志位 SCON0.RI0 置 1。如果停止位错误，则 FE0（Framing Error）标志置 1。在 9bit 接收中，采样第 9 位后，在停止位中间位置中断标志位 SCON0.RI0 置 1，同时第 9 位写入 SCON0.RB80，停止位错误仅用来产生 FE0 标志。在 SCON0.RI0 置 1 时，SBUF 加载 8bit 接收到数据。时序图如下：

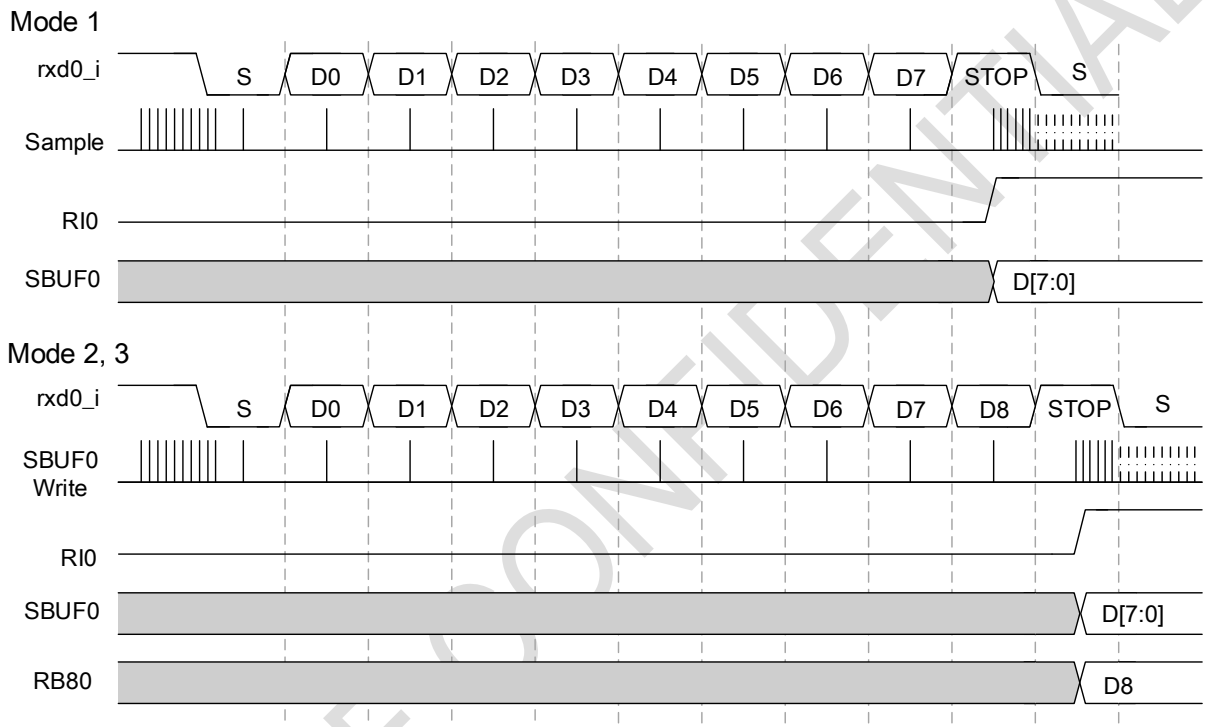

图 13-9. UART 异步接收时序图（Mode 1 和 Mode 3）

13.4 固定波特率的异步全双工模式（Mode 2）

Mode2 是固定波特率的 9 位异步全双工收发的工作模式。其发送和接收框图如下所示。

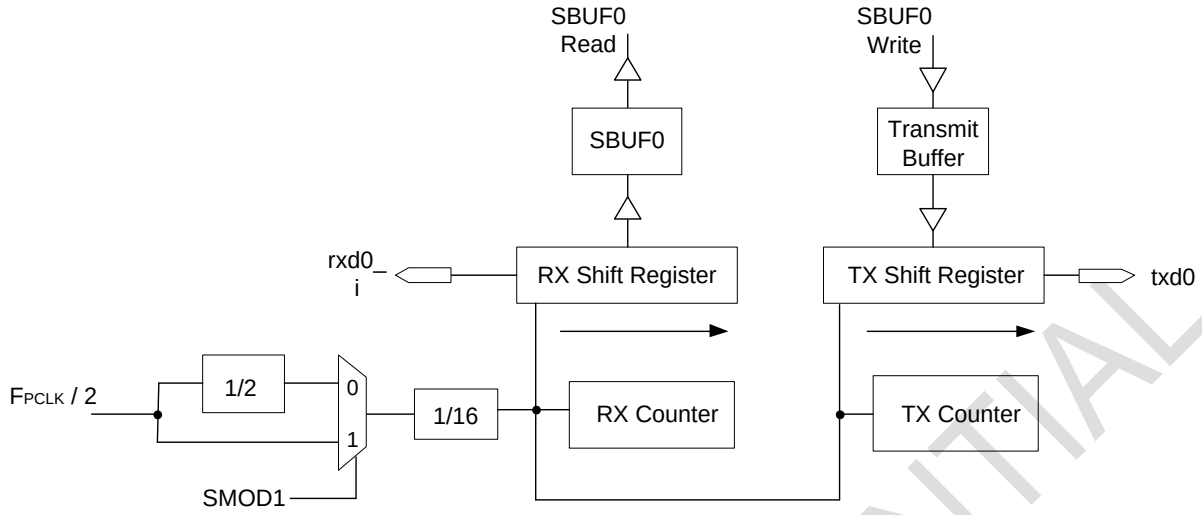

图 13-10. UART Mode 2 收发框图

设置 UART 为 Mode 2 的步骤：

- 将 SCON0.SM00 和 SCON0.SM10 配置为 10；
- 接收时，将接收使能位 SCON0.REN0 置 1；
- 发送时，将数据写入发送缓冲器 SBUF 及 SCON0.TB8 位。

Mode2 和 Mode3 的区别仅在于波特率不同，Mode2 的波特率计算如下：

$$\text{Baud\_Rate} = 2^{\text{SMOD1}} \times \frac{F_{\text{PCLK}}}{64}$$

UART 的 Mode2 和 Mode3 都支持多机通讯，设置 SCON0.SM20 为 1 即可。在该模式下，主机可以通过串口线向多个从机发送数据，只有当从机接收到第 9 位数据 RB8 为 1 时才能被主机识别，其余的 8bit 数据用于传输从机地址，只有地址相符才能接收全部数据。接下来的数据流仅发给已识别的从机设备，且该数据流的第 9 位需置 0，以便其他从机无法被识别。

13.5 USART 增强模式

前面章节介绍了 UART 的各种工作模式，其中只有模式 1 和模式 3 的波特率是可根据 Timer1 溢出周期实现不同波特率。其中，以 Timer1 工作在模式 2（8 位重载模式）最为简单实用。但是，CMT2187A 的 HFOSC 时钟源并不支持太多选择，内部只有 24MHz，又或是外部的 13MHz（26MHz/2 产生），在这种情况下导致波特率被限制，如下表所示。

表 13-1. 标准模式（Timer1 模式 2）UART 波特率误差表

目标波特率	SMOD=1			SMOD=0		
	TH1 设置值	实际波特率	误差值	TH1 设置值	实际波特率	误差值
300	--	--	--	48	300.48	0.16%
600	48	601	0.17%	152	601	0.17%
1200	152	1202	0.17%	204	1202	0.17%
2400	204	2404	0.17%	230	2404	0.17%
4800	230	4808	0.17%	243	4808	0.17%
9600	243	9615	0.16%	249	8929	-6.99%
14400	247	13889	-3.55%	252	15625	8.51%
19200	249	17857	-6.99%	253	20833	8.51%

注意：选择 HFOSC 时钟，频率为 24MHz（即 F_{PCLK}=24MHz）。

由表 13-1 可见只有 300、600、1200、2400、4800、9600 这六个常见的低速率误差较少，能满足使用，但更高波特率则基本完全不能使用。为了让 CMT2187A 能支持更多波特率可选择，在片内增设增强模式：可通过把 USART_SEL（位于 SFR 寄存器 USART_CTL）置 1，则可以取消 Timer1 时钟源前置的 12 分频器，直接由 F_{PCLK} 作为时钟源提供 Timer1，如下图所示。

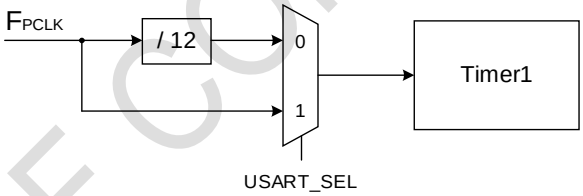

图 13-11. 增强型模式下 Timer1 时钟源示意图

- 当 USART_SEL=0 时，Timer1 时钟源按标准 51 架构，是 F_{PCLK}/12；
- 当 USART_SEL=1 时，Timer1 时钟源为增强性模式，直接由 F_{PCLK} 提供时钟；

按增强模式计算波特率误差表如下。

表 13-2. 增强模式（Timer1 模式 2）UART 波特率误差表

目标波特率	SMOD=1			SMOD=0		
	TH1 设置值	实际波特率	误差值	TH1 设置值	实际波特率	误差值
4800	--	--	--	100	4808	0.16%
9600	100	9615	0.16%	178	9615	0.16%
14400	152	14423	0.16%	204	14423	0.16%

目标波特率	SMOD=1			SMOD=0		
	TH1 设置值	实际波特率	误差值	TH1 设置值	实际波特率	误差值
19200	178	19230	0.16%	217	19231	0.16%
38400	217	38462	0.16%	236	37500	-2.34%
56000	229	55556	-0.79%	243	57692	3.02%
57600	230	57692	0.16%	243	57692	0.16%
115200	243	115385	0.16%	249	107143	-6.99%

从上表可见，采用增强模式后，可以支持常见的高于 9600 波特率，如：14400、19200、38400、56000、57600、115200，正好可以与标准模式互补。

13.6 相关寄存器

表 13-3. UART 寄存器组列表

名称	SFR 页	地址	默认值	功能
PCON	0	0x87	0x00	功耗控制寄存器
SCON0	0	0x98	0x00	串口 0 控制寄存器
SBUF0	0	0x99	0x00	串口 0 数据缓存寄存器
USART_SEL	0	0x9F	0x01	Timer1 时钟源分频器选择

14 Timer A/Timer B 模块

Timer A 和 Timer B 均为由 1 个 16bit 计时/计数器以及 3 个捕获/比较器组成，可以实现多个捕获/比较器、PWM 输出以及计数触发条件的时间间隔。Timer A 具有多种中断模式，触发来自于计时/计数器的溢出和捕获/比较器。Timer A / Timer B 特性包括：

- 包括 4 种工作模式的 16-bit 计时/计数器；
- 时钟源可配置为系统时钟的 1~65535 分频；
- 2~3 个可配置的捕获/比较器；
- 可配置的 PWM 输出；
- 异步输入采样；
- 快速捕捉中断来源。

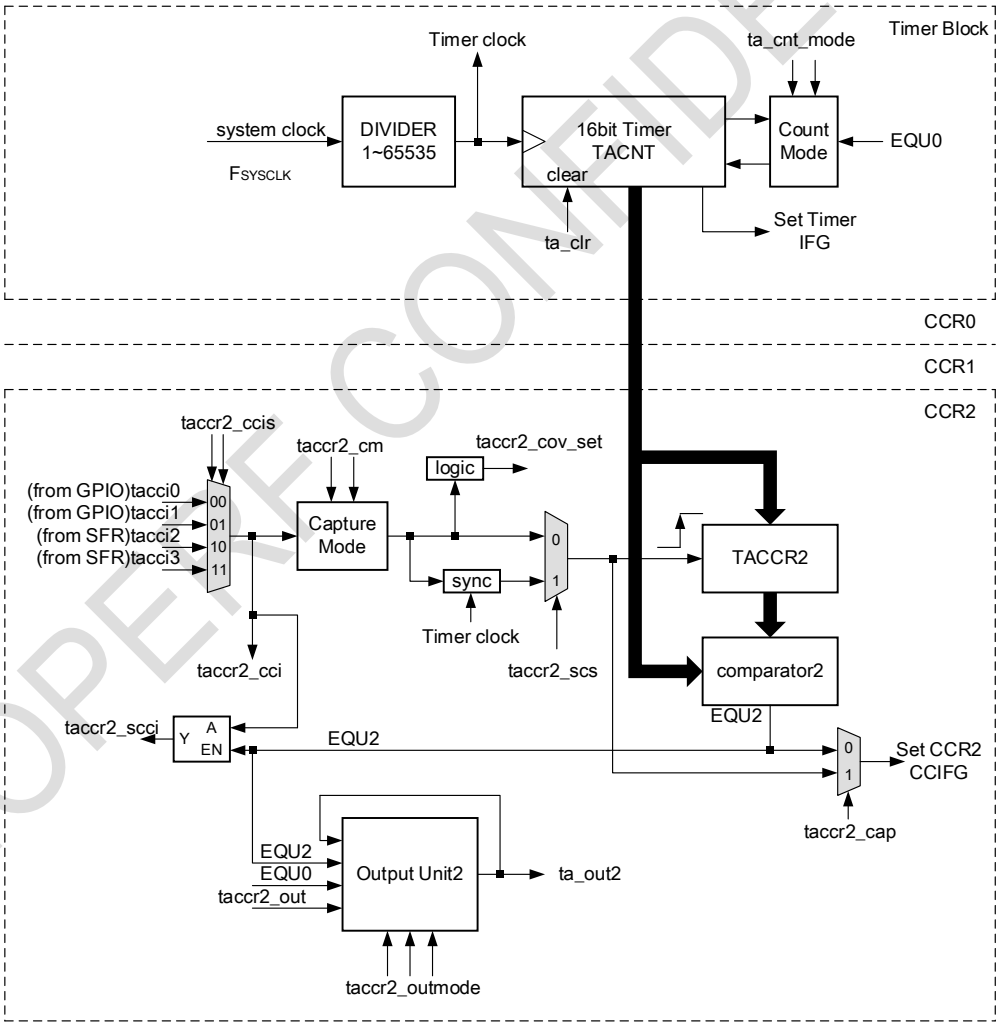

图 14-1. Timer A 结构框图

注意：Timer A 和 Timer B 结构完全相同，上图以 Timer A 结构作为说明。

14.1 操作方法

Timer A / Timer B 模块的操作由软件控制，该章节中的提到的 TACCR0TH / TBCCR0TH 为可配置的 16 位计数阈值，由寄存器 TACCR0TH_H / TBCCR0TH_H 和 TACCR0TH_L /TBCCR0TH_L 组合而成.

16-bit 计时/计数器（TACNT 和 TBCNT）根据不同的工作模式，会进行不同方式的计数，如表 14-1 所示。

表 14-1. Timer 各种计数工作模式

TA/TB_CNT_MODE	工作模式	描述
00	Stop	TACNT / TBCNT 不工作，停止计数模式
01	Up	TACNT / TBCNT 重复地从 0 递增到 TACCR0TH / TBCCR0TH 值，递增计数模式
10	Continuous	TACNT / TBCNT 重复地从 0 递增到 0xFFFF，重复计数模式
11	Up/ Down	TACNT / TBCNT 重复地从 0 递增到 TACCR0TH / TBCCR0TH，然后再递减到 0，先增后减计数模式

从上表可以了解，Timer A 或 Timer B 包括 4 种工作模式：停止（Stop）、递增计数（Up）、重复计数（Continuous）、先增后减（Up/Down），通过配置 TA_CNT_MODE 或 TB_CNT_MODE 决定工作在何种模式。

当需要临时修改 Timer A 或 Timer B 的工作模式（对中断使能、中断标志的修改除外）时，建议先停止 TACNT 或 TBCNT 的计数，以免产生不可预知的误操作。

在递增计数模式下， TACNT 或 TBCNT 一旦达到设定的阈值 TACCR0TH / TBCCR0TH 就会产生中断。

在重复计数模式下， TACNT 或 TBCNT 一旦达到 0xFFFF 就会产生中断。

在先增后减计数模式下， TACNT 或 TBCNT 递减到 0x0001 就会产生中断。

启动 Timer A 或 Timer B 之前，需要配置 TAC_L.TA_CNT_MODE 或 TBC_L.TB_CNT_MODE（非 Stop 模式）、计数阈值 TACCR0TH 或 TBCCR0TH 的值（≠0）、捕获/比较等参数（详见 4.14.5 节），然后先把 TAC_H.TA_START 或 TBC_H.TB_START 先置零，再置 1，触发生效。

计数过程中，用户可以将 TACL.TA_CLR 或 TBCL.TB_CLR 置 1 以清空计数器的绝大部分配置，以 Timer A 为例，TA_CLR 可清空的值包括：计数时钟源的分频值 TACLK_DIV，计数值 TACNT，计数器工作模式 TA_CNT_MODE，计数阈值 TACCR0TH 等等。

14.2 递增计数模式（Up Mode）

在 Up 模式下，用户可将计数阈值 TACCR0TH 或 TBCCR0TH 配置为任意值，TACNT（或 TBCNT）会重复地从 0 递增到阈值 TACCR0TH（或 TBCCR0TH），计数周期为 TACCR0TH（或 TBCCR0TH）+1。当 TACNT（或 TBCNT）计数到阈值时，立即返回到 0 重新计数。

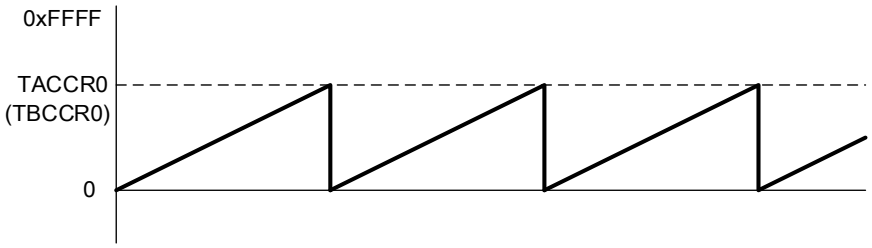

图 14-2. Timer A / Timer B 工作 Up 模式示意图

当 TACNT（或 TBCNT）计数到 TACCR0TH 或 TBCCR0TH 并溢出返回为 0 后，中断标志 TA_CCR0_INT 或 TB_CCR0_INT 会置位，而 Timer A（或 Timer B）的中断标志 TA_TMR_INT（或 TB_TMR_INT）会比 TA_CCR0_INT 晚一拍置位。下图为预分频值 TA_CLK_DIV 设置为 3 之后两个不同中断的产生示意图：

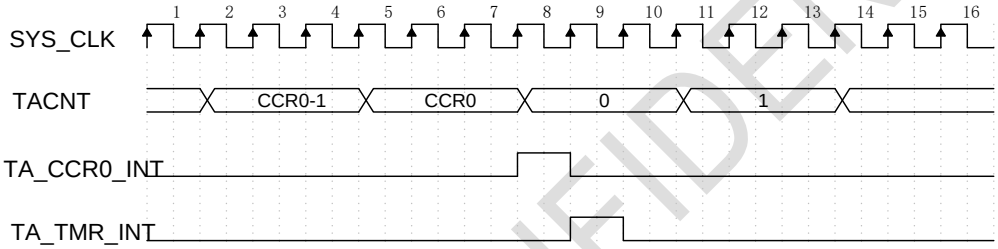

图 14-3. Timer A 工作 Up 模式中断示意图（TimerB 相同）

注意：在 TACNT（或 TBCNT）在计数过程中不建议修改 TACCR0TH（或 TBCCR0TH）的值，但如果用户强行修改就会根据条件不同，产生两种情况：

1. *如果新的 TACCR0TH（或 TBCCR0TH）修改值大于之前的值，或大于当前的 TACNT（或 TBCNT）计数值，则 TACNT（或 TBCNT）需要继续计数到新的阈值 TACCR0TH（或 TBCCR0TH），再返回到 0 重新计数；*
2. *如果新的 TACCR0TH（或 TBCCR0TH）修改值小于当前 TACNT（或 TBCNT）计数值，则 TACNT（或 TBCNT）会被马上置 0，并从 0 开始不断计数到新的 TACCR0TH（或 TBCCR0TH）。*

14.3 重复计数模式（Continuous Mode）

Continuous 模式下，TACNT（或 TBCNT）重复地从 0 递增到 0xFFFF，复位然后再从 0 开始计数。此模式下，三组捕获/比较器 ccr0 ~ ccr2 具有相同的功能且独立工作，这个与 Up 模式有区别。Up 模式情况下，TACCR0TH（或 TBCCR0TH）是 TACNT（或 TBCNT）计数的周期值。

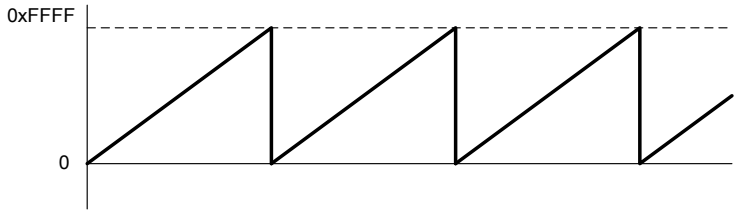

图 14-4. Timer A 工作 Continuous 模式示意图（Timer B 相同）

重复计数模式下，捕获/比较模块可以单独产生中断，如下图 14-5 所示：

- 当 TACNT（TBCNT）计数到 TACCR0TH+1（TBCCR0TH+1）值时，比较器 ta_ccr0 或 tb_ccr0 的中断标志 TA_CCR0_INT（TB_CCR0_INT）置位。
- 当 TACNT（TBCNT）计数到 0xFFFF 再返回到 0 重新计数时，Timer 中断标志 TA_TMR_INT（TB_TMR_INT）置位。

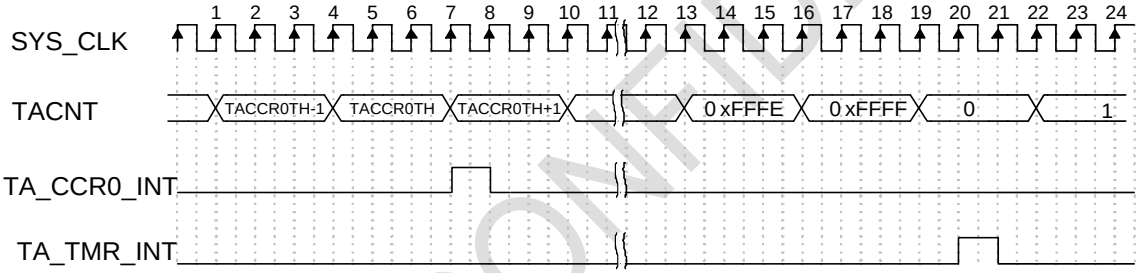

图 14-5. Timer A 工作 Continuous 模式中断示意图（Timer B 相同）

Continuous 模式可以利用上述方式，用于产生独立的时间间隔和输出频率。以比较器 ccr0 举例，TimerA 会在计数到 TACCR0TH 时产生中断，软件在寄存器中检测到该中断后，可以将比较阈值 TACCR0TH 配置为 TACCR0TH + n，其中 n 为设定周期值，且 n<0XFFFF。如此一直往复更新 TACCR0TH 的值，便可以产生周期为 n 的中断。

因此使用三组捕获/比较器 ta_ccr0~ ta_ccr2（tb_ccr0~ tb_ccr2）以产生三组独立的时间间隔和频率输出，具体如下图所示，其中 TACCR0THa，TACCR0THb，TACCR0THc，TACCR0THd 为软件根据 TACCR0TH =TACCR0TH+ n 算式而计算出的不同比较值，ta_ccr1 同理。

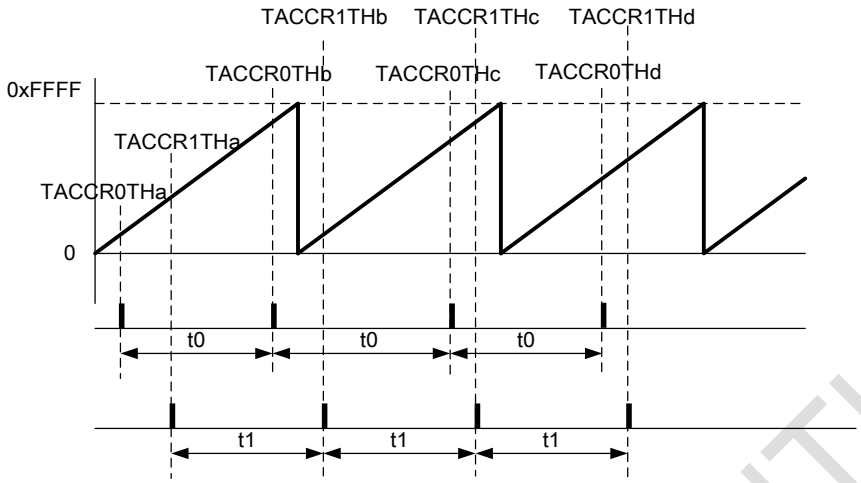

图 14-6. Timer A 各组捕获/比较器独立工作示例（Timer B 相同）

14.4 先增后减模式（Up / Down Mode）

在 Up / Down 模式下，TACNT（TBCNT）重复地从 0 递增到 TACCR0TH（TBCCR0TH），再递减到 0。一个周期为 2 倍 TACCR0TH（TBCCR0TH）值。

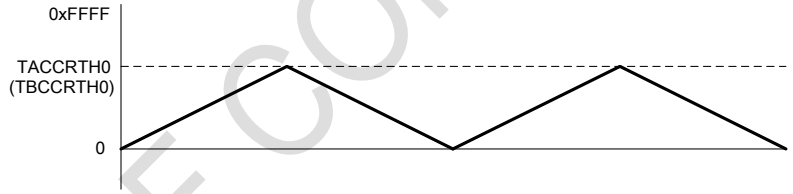

图 14-7. Timer A / B 工作 Up / Down 模式示意图

ta_ccr0 和 tb_ccr0 的中断 TA_CCR0_INT（TB_CCR0_INT）和 TACNT（TBCNT）的中断标志 TA_TMR_INT（TB_TMR_INT）在一个周期内，分布于一个周期的前半周期和后半周期。与 Up 模式类似，当 TACNT（TBCNT）计数到 TACCR0TH（TBCCR0TH）值时，ta_ccr0（tb_ccr0）的中断标志 TA_CCR0_INT（TB_CCR0_INT）置位，当 TACNT（TBCNT）计数到 ta_ccr0（tb_ccr0）的阈值再返回到 0 重新计数时，Timer A（Timer B）中断标志 TA_TMR_INT（TB_TMR_INT）置位。

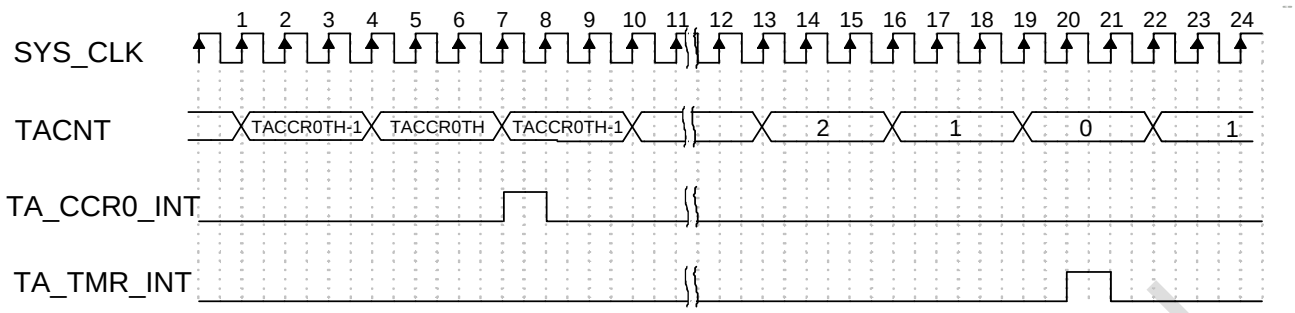

图 14-8. Timer A 工作 Up / Down 模式中断示意图

Up/ Down 模式支持在两个输出信号之间需要死区（Dead Time）的应用，比如，两个同时驱动一个 H 桥的输出，不能同时输出高电平，避免出现过载情况。其中，

$$T_{Dead}=T_{Timer}x（TACCR1TH–TACCR2TH）$$

其中：

1. T_{Dead} 是指死区的持续时间
2. T_{Timer} 是指 TACNT 或 TBCNT 的时钟周期
3. TACCR1TH 和 TACCR2TH 是指两组捕获/比较器的配置值。

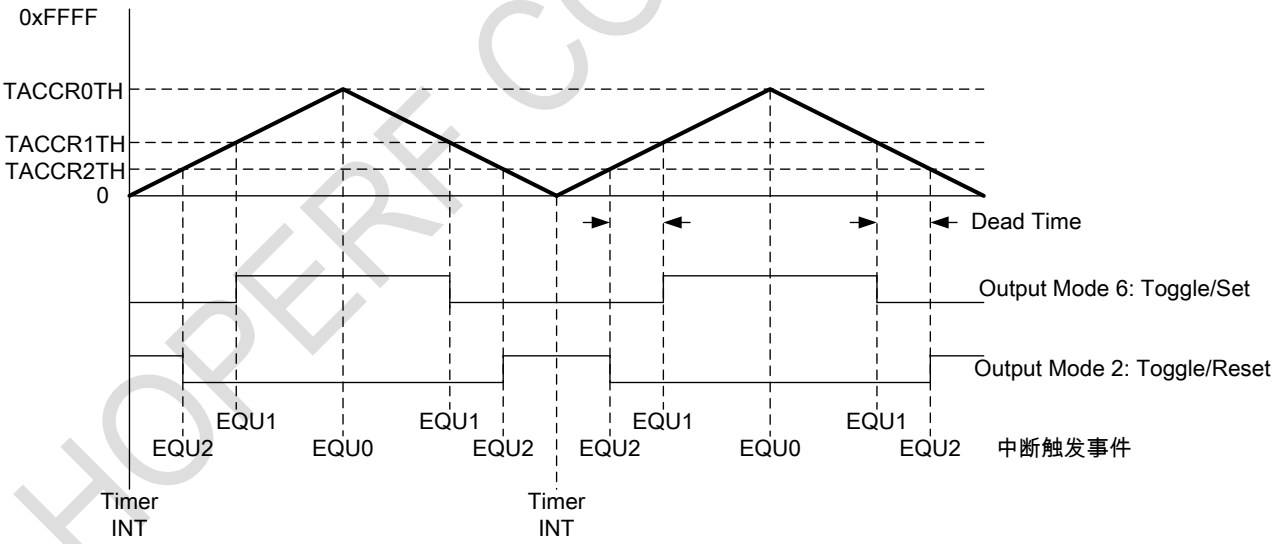

图 14-9. Up / Down 模式带死区控制示意图（以 TimerA 为例）

14.5 捕获/比较模块

Timer A / B 中包含 2~3 个独立的捕获/比较模块，用于捕获 TACNT（或 TBCNT）数据或产生时间间隔。其

中，在 Up 和 Up / Down 模式下，TACCR0TH（TBCCR0TH）作为周期寄存器，不能存储捕获值。在 Continuous 模式，ta_ccr0~ ta_ccr2（tb_ccr0~ tb_ccr2）都能存储捕获值。

- **捕获模式**

配置 TACCTL0_H.TA_CCR0_FUNC_MODE ~ TA_CCR2_FUNC_MODE 为 1，则对应的捕获/比较模块进入捕获模式。捕获模式用于记录时间事件，如速度估计或时间测量。捕获源有 4 个，其中 TACCI0 和 TACCI1 来自 GPIOn（可配置选择，具体可见 4.9 GPIO 模块一节），TAC_H.TA_CCI2_IN_SFR 和 TAC_H.TA_CCI3_IN_SFR 来自内部 SFR 寄存器，软件可访问，配置 TA_CCR0_SRC_SEL~TA_CCR1_SRC_SEL 可以给三个捕捉通道选择不同的捕获源（CCI0~CCI3）。Timer B 完全相同，在此不重复介绍，下文同。

配置 TA_CCR0_CAP_MODE~ TA_CCR2_CAP_MODE 可选择对应捕获/比较模块的捕获模式为上升沿、下降沿或双沿触发。捕获成功后，TACNT 的值会被存储到对应捕获/比较模块的 TACCRn 寄存器，同时对应的中断标志 TA_CCR0_INT~TA_CCR2_INT 置位。

配置 TA_CCR0_SYNC_EN ~ TA_CCR2_SYNC_EN 可选择是否对捕获源进行系统时钟同步。

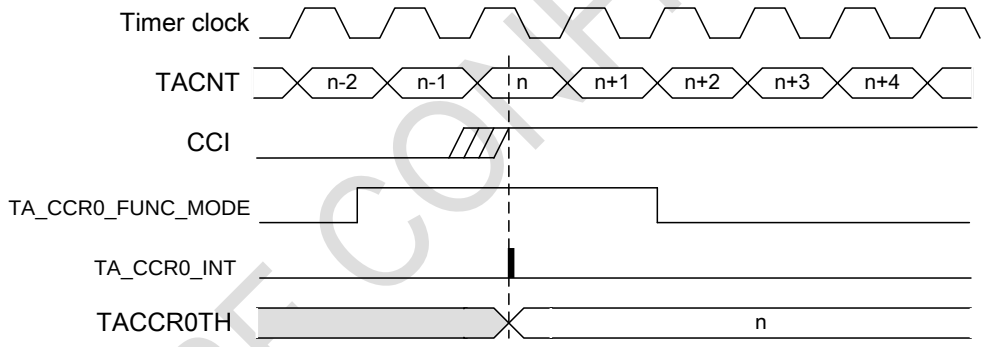

图 14-10. Timer A 工作捕获模式示意图

如果前一次的捕获结果还未被读取，而捕获源再次触发，将产生捕获溢出，对应捕获/比较模块的 COV 标志将被置 1，由软件清零后重新捕获。

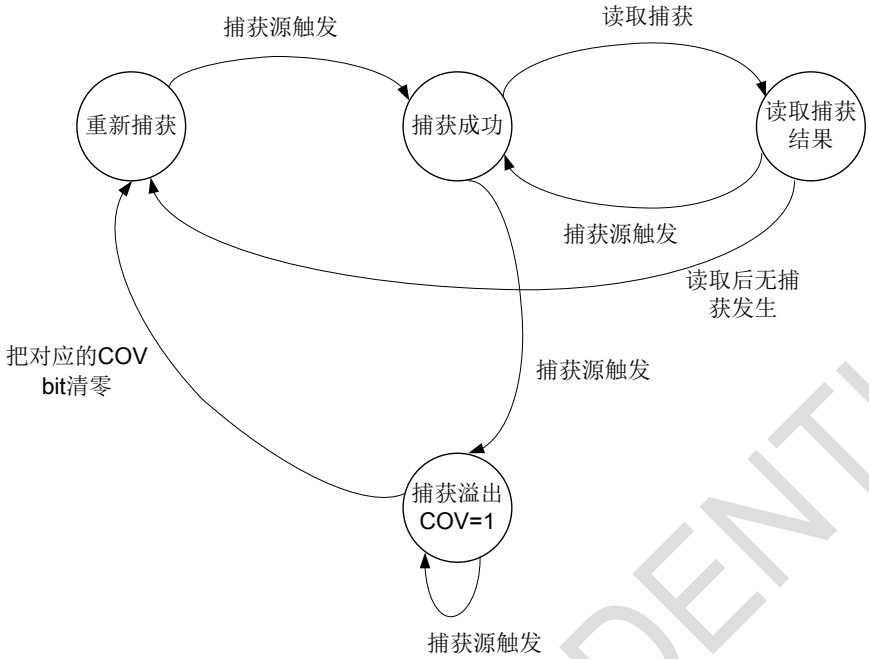

图 14-11. Timer A 工作捕获模式状态和中断示意图

- **比较模式**

配置 TA_CCR0_FUNC_MODE ~ TA_CCR2_FUNC_MODE 为 0，则对应的捕获/比较模块进入比较模式。比较模式用于产生 PWM 输出信号或在特定的时间间隔产生中断。当 TACNT 计数到 TACCR0TH ~ TACCR2TH 时：

1) 对应的中断标志 TA_CCR0 /1/2_ INT 置 1；
2) 对应的计数相等判定信号 EQU0 ~ EQU2 为 1；
3) EQU0~EQU2 根据不同的输出模式影响输出信号；
4) 每个比较器选择后的捕获源会存储到对应的 TA_CCR0_SRC ~ TA_CCR2_SRC 寄存器；

- **输出单元**

每个捕获/比较模块都包含一个输出单元，用来产生输出信号如 PWM 信号。每个输出单元基于 EQU0 和 EQU1/EQU2 信号，可组合成 8 种输出模式。

TA_CCR0_OUT_MODE ~ TA_CCR2_OUT_MODE 为对应捕获/比较模块的输出配置寄存器，其中输出模式 2、3、6 和 7 不适用于输出单元 0，因为 EQUx = EQU0。（EQUx 意为 EQU1 与 EQU2）

表 14-2. 输出单元各种模式

OUTMODE	模式	描述	备注
000	输出	直通模式，输出 TA_OUTx 由寄存器 CCRx_OUT 配置。	适用于 3 个捕获/比较模块
001	置位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 置位并保持，直到 Timer A 被复位，或更改到另一种输出模式并影响到输出。	适用于 3 个捕获/比较模块
010	翻转/复位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 翻转；当 TAR 计数到 TACCR0TH 时，输出 TA_OUTx 复位。	仅适用于捕获器 1 和 2
011	置位/复位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 置位；当 TAR 计数到 TACCR0TH 时，输出 TA_OUTx 复位。	仅适用于捕获器 1 和 2
100	翻转	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 翻转。	适用于 3 个捕获/比较模块
101	复位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 复位并保持，直到更改到另一种输出模式并影响到输出。	适用于 3 个捕获/比较模块
110	翻转/置位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 翻转；当 TAR 计数到 TACCR0TH 时，输出 TA_OUTx 置位。	仅适用于捕获器 1 和 2
111	复位/置位	当 TACNT 计数到 TACCRxTH 时，输出 TA_OUTx 复位；当 TAR 计数到 TACCR0TH 时，输出 TA_OUTx 置位。	仅适用于捕获器 1 和 2

注意：

1. *TACCRxTH 指 TACCR1TH 或 TACCR2TH；用 Timer B 时，指 TBCCR1TH 和 TBCCR2TH；*
2. *TA_OUTx 指 TA_OUT1 或 TA_OUT2；用 Timer B 时，指 TB_OUT1 和 TB_OUT2；*

14.6 各工作模式举例

- **Up 模式的输出举例**

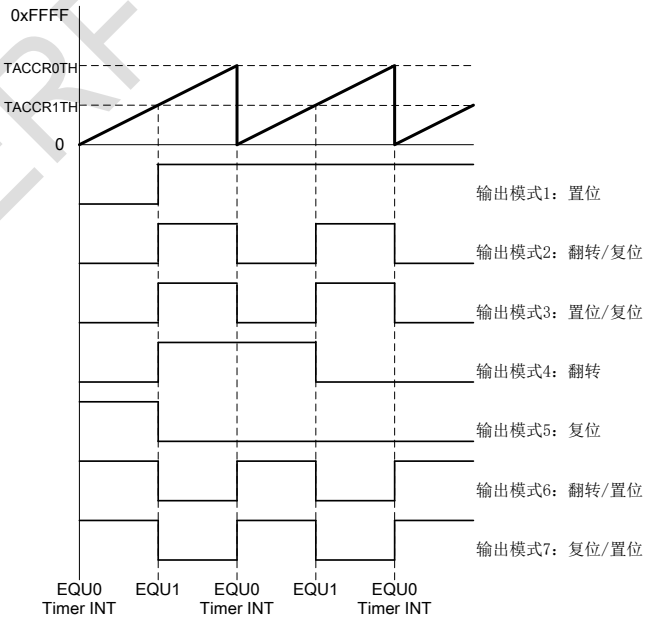

图 14-12. Timer A 工作 Up 模式输出效果示意图

- **Continuous 模式的输出举例**

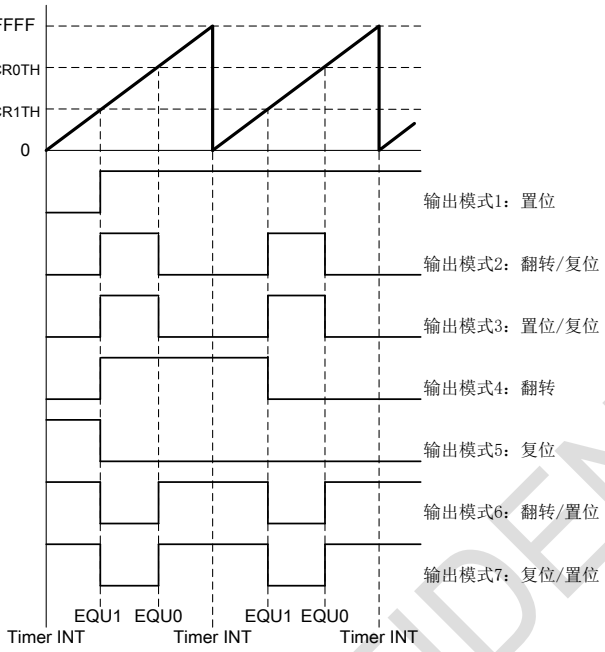

图 14-13. Timer A 工作 Continuous 模式输出效果示意图

- **Up / Down 模式的输出举例**

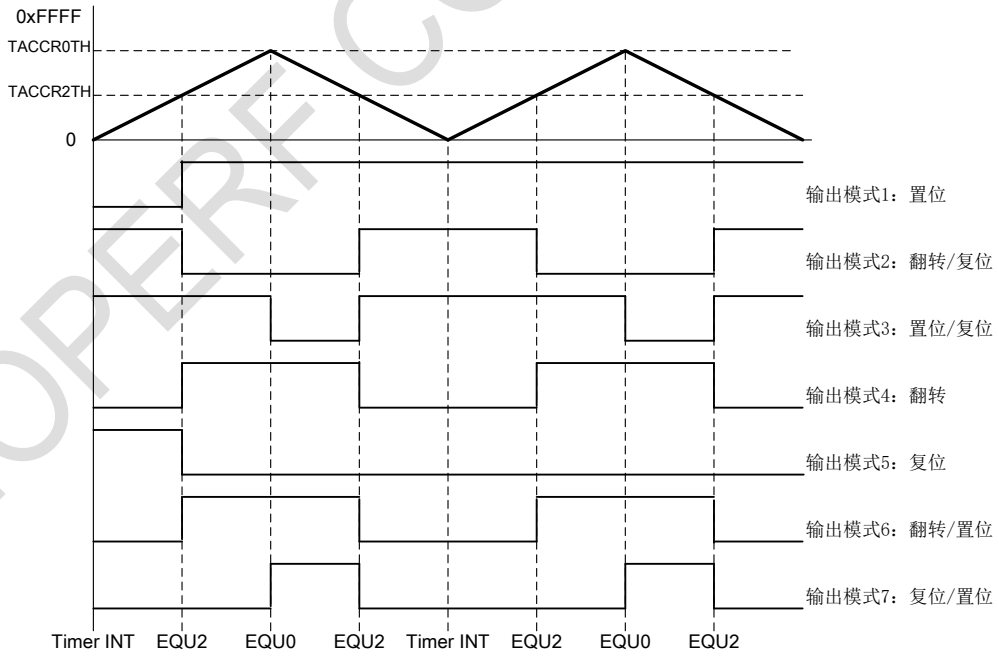

图 14-14. Timer A 工作 Up / Down 模式输出效果示意图

14.7 相关寄存器

表 14-3. 16-Bit TimerA 和 TimerB 存器组列表

名称	SFR 页	地址	默认值	功能
TACLK_DIV_H	0	0xB7	0x00	TimerA 时钟源分频寄存器
TACLK_DIV_L	0	0xB9	0x00	
TAC_H	0	0xBA	0x00	TimerA 控制寄存器
TAC_L	0	0xBB	0x00	
TACCR0TH_H	0	0xBC	0x00	TimerA 捕获/比较模块 0 寄存器
TACCR0TH_L	0	0xBD	0x00	
TACCTL0_H	0	0xBE	0x00	TimerA 捕获/比较模块 0 控制寄存器
TACCTL0_L	0	0xBF	0x00	
TACCR1TH_H	0	0xC0	0x00	TimerA 捕获/比较模块 1 寄存器
TACCR1TH_L	0	0xC1	0x00	
TACCTL1_H	0	0xC2	0x00	TimerA 捕获/比较模块 1 控制寄存器
TACCTL1_L	0	0xC3	0x00	
TACCR2TH_H	0	0xC4	0x00	TimerA 捕获/比较模块 2 寄存器
TACCR2TH_L	0	0xC5	0x00	
TACCTL2_H	0	0xC6	0x00	TimerA 捕获/比较模块 2 控制寄存器
TACCTL2_L	0	0xC7	0x00	
TACNT_H	0	0xC8	0x00	TimerA 16 位实时计数寄存器
TACNT_L	0	0xC9	0x00	
TBCLK_DIV_H	0	0xCA	0x00	TimerB 时钟源分频寄存器
TBCLK_DIV_L	0	0xCB	0x00	
TBC_H	0	0xCC	0x00	TimerB 控制寄存器
TBC_L	0	0xCD	0x00	
TBCCR0TH_H	0	0xCE	0x00	TimerB 捕获/比较模块 0 寄存器
TBCCR0TH_L	0	0xCF	0x00	
TBCCTL0_H	0	0xD1	0x00	TimerB 捕获/比较模块 0 控制寄存器
TBCCTL0_L	0	0xD2	0x00	
TBCCR1TH_H	0	0xD3	0x00	TimerB 捕获/比较模块 1 寄存器
TBCCR1TH_L	0	0xD4	0x00	
TBCCTL1_H	0	0xD5	0x00	TimerB 捕获/比较模块 1 控制寄存器
TBCCTL1_L	0	0xD6	0x00	
TBCCR2TH_H	0	0xD7	0x00	TimerB 捕获/比较模块 2 寄存器
TBCCR2TH_L	0	0xD8	0x00	
TBCCTL2_H	0	0xD9	0x00	TimerB 捕获/比较模块 2 控制寄存器
TBCCTL2_L	0	0xDA	0x00	
TBCNT_H	0	0xDB	0x00	TimerB 16 位实时计数寄存器
TBCNT_L	0	0xDC	0x00	

15 看门狗（WDT）模块

15.1 基本功能

系统内部集成了硬件看门狗（Watchdog Timer，简称 WDT）模块，用来监控系统的运行状态。在系统运行以后，用户可以启动看门狗定时器进行计数，如果在配置时间内没有去清零看门狗定时器，系统会产生全局复位，此复位等效于系统重新上电。

15.2 相关寄存器

表 15-1. 看门狗寄存器列表

名称	地址	默认值	功能
AON_REG_03	0x03	0x00	看门狗控制寄存器

16 睡眠定时器模块

16.1 基本功能

CMT2187A 内置一个睡眠定时器（Sleep Timer），用于低功耗休眠定时唤醒的应用场合，由系统低频时钟（LFOSC）提供时钟源。

Sleep Timer 只在 STOP 模式时开始计时，计时溢出时把处于 STOP 模式中的系统给唤醒。系统被唤醒后，处于（程序）运行状态时，Sleep Timer 停止计时，直到系统进入 STOP 模式。

16.2 LPOSC 的校准

32kHz LPOSC 频率会随着工艺，温度和电压的改变而漂移，芯片在出厂阶段会进行一次 LPOSC 的校准，并将校准之后的配置值储存在芯片内部。在芯片工作过程中，用户可适时调用校准模块对其进行校准，具体操作可参考例程。

16.3 相关寄存器

表 16-1. 睡眠定时器模块寄存器组列表

名称	地址	默认值	功能
AON_REG_01	0x01	0x00	睡眠定时器控制和配置
AON_REG_02	0x01	0x00	睡眠定时器控制和配置

表 16-2. LFOSC 校准和输出相关寄存器列表

名称	SFR 页	地址	默认值	功能
ANA_CTL_4	0	0xE2	0x04	LFOSC 校准控制
ANA_CTL_8	0	0xE7	0x80	LFOSC 校准控制

17 低电压复位（LVR）

低电压复位指当电源电压低于 VLVR 门限电压时会产生芯片的复位信号。CMT2187A 第一次上电时默认的复位释放电压为 1.8V，后续的复位释放电压和是否带迟滞检测功能。

表 18-1. 低电压复位配置

名称	地址	默认值	功能
AON_REG_06	0x06	0x00	LVR 配置寄存器

18 低电压监测（LBD）模块

18.1 基本功能

CMT2187A 内嵌低电压监测模块，主要作用在于在进行发射前进行一次电源电压检测，如发现电压不足，会对发射功率进行补偿。发射功率补偿是片内发射机根据当前的电压检测值自动进行计算的，用户无需参与。用户只需要在进行发射前触发一次 LBD 检测即可。

用户也可以随时使用 LBD 模块满足应用需求。用户可先设置好 LBD_TH<3:0>的值，然后将 LBD_START 设为 1 触发 LBD 模块工作，LBD 模块完成工作后会自动将 LBD_DONE 设为 1，并得出代表当前 DVDD 电压值的 LBD_CODE<3:0>的值，如果 LBD_CODE<3:0>≥LBD_TH<3:0>，那么 LBD_STATUS 会等于 1，否则等于 0。用户查询到 LBD_DONE 位 1 时就可以去读取 LBD_STATUS 和 sdLBD_CODE<3:0>，并根据如下表格得出对应的电压值。

表 19-1. LBD 模块的各个电压阈值

LBD_CODE<3:0>	电压值	LBD_CODE<3:0>	电压值
0	1.45 V	8	3.05 V
1	1.65 V	9	3.25 V
2	1.85 V	10	3.45 V
3	2.05 V	11	3.65 V
4	2.25 V	12	3.85 V
5	2.45 V	13	4.05 V
6	2.65 V	14	4.25 V
7	2.85 V	15	4.45 V

需要注意的是，此 LBD 模块在系统进入 STOP 后不会工作，与系统的实时电源电压监控（Supply Monitor）功能是无关的。前者用于检测具体的电压值，后者用于检测电压突变从而产生系统复位。

18.2 相关寄存器

表 19-2. LBD 模块相关寄存器

名称	SFR 页	地址	默认值	功能
LBD_CTL_0	0	0xFB	0x00	LBD 配置和控制寄存器
LBD_CTL_1	0	0xFC	0x00	LBD 配置和控制寄存器

19 Sub-1G 发射模块

19.1 基本介绍

CMT2187A 内嵌的 Sub-1G 发射模块采用小数型分频技术的锁相环（PLL），使用 26MHz 的 XOSC，通过软件设置不同的分频比，实现不同的发射频点。该模块支持(G)FSK 和 OOK 调制方式，是基于射频频率直接综合的调制方式。其载波频率是由一个低噪声小数分频频率综合器产生，调制数据由一个高效的功率放大器（PA）发射出去。其总体框图如下所示。

下图是发射机的结构图，基带信号由用户程序产生，先后经过高斯变换，频率控制，AFC / DSM 控制，RAMP 控制等模块的处理，根据发射相关的配置，将频率控制值输送到 PLL 模块，将功率控制值和电流控制值输送到 PA 模块，同时完成对 PA 的校正过程。

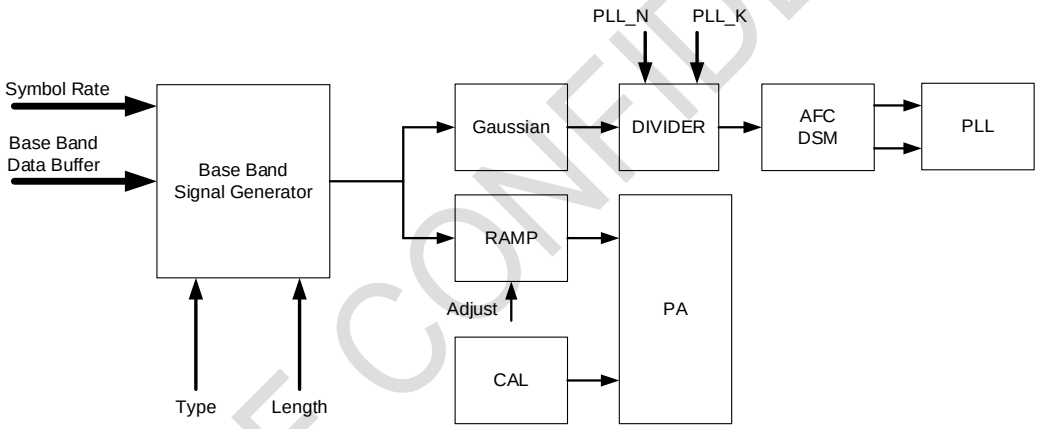

图 20-1. CMT2187A UHF 发射简单框图

PLL 的锁定时间大概在 150uS，OOK 通过直接调节 PA 的输出幅度实现。CMT2187A 可以配置频率范围是 210MHz – 960MHz，在这个范围内不支持 480MHz-630MHz。

19.2 PA 输出方式

CMT2187A 的单端 PA 是 CLASS-E 结构，最大输出功率为+13 dBm。PA 具有 RAMP 斜率可调的特点，如下图所示。用户通过使用 TxSoC RFPDK 工具软件，设置好目标工作参数后，会自动生成相应的 PA Ramping 参/数。使用 PA Ramping 能有效抑制发射过程中造成杂讯。

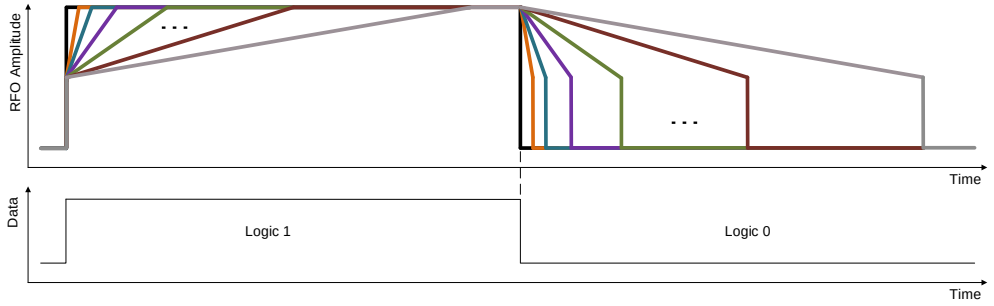

图 20-2. PA Ramping 时序

19.3 Buffer 模式发射流程

发射流程是发射模块根据基带信号，进行控制模拟 PLL 和 PA 的过程。根据基带数据来源区分，可以分为：

- **Buffer 发射模式**
- **Direct（直通）发射模式**

本小节主要介绍 Buffer 模式的发射流程。Buffer 模式顾名思义，用户需要在发射前配置（准备）好基带信号的内容、类型、长度和发射数据率等（即待发数据包格式及内容），然后往基带缓存填充的发射方式。详细可见下图。

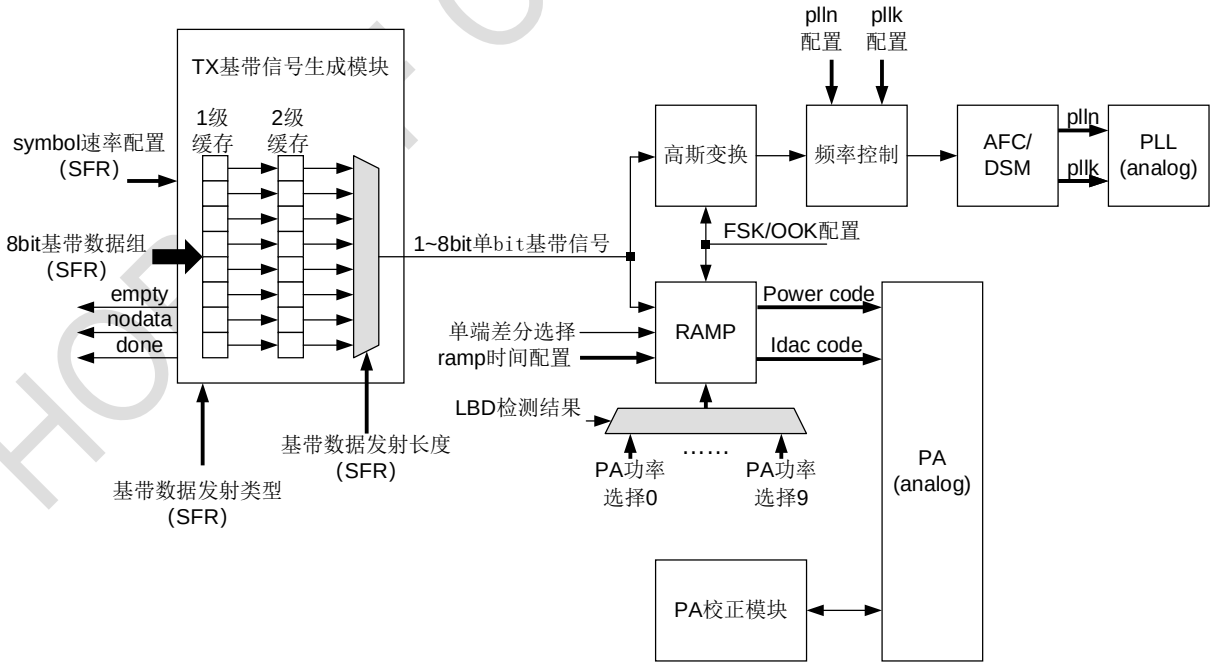

图 20-3. 发射模块结构框图

其中和用户使用关联的几个重要标志和寄存器：

- Symbol 速率配置（TX_DR 寄存器组），即每个符号发射时间参数（T_{SYM_TIME}），其与发射速率（BR）和晶体频率（F_{XTAL}，即 26MHz）关系如下：

$$T_{SYM\_TIME} = \frac{BR \times 2^{23}}{F_{XTAL}}$$

- 基带数据组寄存器（TX_SYM_BYTE 寄存器）：即待发数据缓存寄存器，是发射模块数据入口，支持 1～8bits 的待发数据填充，具体有效发射长度，由“基带数据发射长度”决定；但该寄存器数据发射完毕后，产生发射空标志（empty 标志），此时软件继续填充下一组待发射的基带数据，直到待发数据全部发射完毕。
- 基带数据发射长度（TX_SYM_CTL 寄存器）：可配置决定 TX_SYM_BYTE 的有效发射长度，范围是 1～8 bits。例如：当基带数据发射长度配置为 4bits 有效时，则每次只发射取 TX_SYM_BYTE 的低 4 bits 数据。
- 基带数据发射类型：发射完最后 1 位数据（后续不再填充）之后的工作模式，可配置为：停止发射、循环发射最后填充的数据组、发射常 0 或发射常 1。
- empty 标志：TX 基带信号生成模块包括 2 级缓存，当 1 级缓存的数据存储到 2 级存储，empty 标志生效，即可往 1 级缓存继续填充下一组需要发射的数据；
- nodata 标志：当发射完最后 1 位数据后，后续也不再继续填充数据组，nodata 标志生效；
- done 标志：只有第 1 种基带数据发射类型时才有效，当发射完最后一 bit 数据且后续不再填充后，done 标志生效。

注意：

1. *根据上面讲述，Buffer 模式似乎涉及众多参数，用户操作可参考 TX_PACKET_SAMPLE 等例程。*
2. *大部分配置参数，用户无需手动计算，可以通过 TxSoC RFPDK 工具软件生成。*

19.4 直通模式发射流程

直通模式相对 buffer 模式简单很多，通过配置 TX_DIRECT_EN（位于 TX_SYM_CTL 寄存器）控制使能有效。此模式下，图 20-3 中的“TX 基带信号生成模块”不起作用，直接通过控制 TX_SYM_BYTE 寄存器 Bit0，实现数据“1”和数据“0”的发射。同时，由于“TX 基带信号生成模块”被禁用，每个符号（数据）发射时间长短（即发射速率）完全由 8051 的软件进行控制。

虽然直通发射模式对比 buffer 发射模式简单，但两者之间各有优劣，具体对比如下表所示。

表 20-1. CMT2187A Direct 模式和 Buffer 模式比较

对比项	Buffer 模式	Direct 模式	说明
发射速率准确度	高	低	Buffer 模式是由 TX 基带信号生成模块生成速率时钟，这个时钟源来自 26MHz 晶体，所以速率精度为 ppm 级别；Direct 模式是由软件控制，时钟源来自于内部的 24MHz RC 振荡器。

19.5 相关寄存器

表 20-2. Sub-1G 发射模块存器组列表

名称	SFR 页	地址	默认值	功能
PLLN	0	0xE8	0x42	锁相环 N 值配置寄存器
PLLK_H	0	0xE9	0x00	锁相环 K 值配置寄存器，K 值高 8 位
PLLK_M	0	0xEA	0xC5	锁相环 K 值配置寄存器，K 值中 8 位
PLLK_L	0	0xEB	0xC1	锁相环 K 值配置寄存器，K 值低 8 位
TX_DR_0	0	0xEC	0x00	发射数据率配置寄存器高 8 位
TX_DR_1	0	0xED	0x00	发射数据率配置寄存器中 8 位
TX_DR_2	0	0xEE	0x01	发射数据率配置寄存器低 8 位
TX_SYM_BYTE	0	0xEF	0x00	发射编码寄存器
TX_SYM_CTL	0	0xF2	0x00	发射编码控制寄存器
TX_PKT_CTL	0	0xF3	0x00	发射模式配置寄存器
FREQ_DEV_H	0	0xF4	0x00	频偏配置寄存器高 8 位
FREQ_DEV_M	0	0xF5	0x00	频偏配置寄存器中 8 位
FREQ_DEV_L	0	0xF7	0x00	频偏配置寄存器低 8 位
RAMP_STEP_H	0	0xF8	0x00	RAMP 配置寄存器高 8 位
RAMP_STEP_L	0	0xF9	0x00	RAMP 配置寄存器低 8 位
PA_IDAC_CODE	0	0xFA	0x00	发射功率配置寄存器
PA_POWER_TH_9	1	0xC0	0x00	发射功率补偿寄存器 9，3.85V
PA_POWER_TH_8	1	0xC1	0x00	发射功率补偿寄存器 8，3.65V
PA_POWER_TH_7	1	0xC2	0x00	发射功率补偿寄存器 7，3.45V
PA_POWER_TH_6	1	0xC3	0x00	发射功率补偿寄存器 6，3.25V
PA_POWER_TH_5	1	0xC4	0x00	发射功率补偿寄存器 5，3.05V
PA_POWER_TH_4	1	0xC5	0x00	发射功率补偿寄存器 4，2.85V
PA_POWER_TH_3	1	0xC6	0x00	发射功率补偿寄存器 3，2.65V
PA_POWER_TH_2	1	0xC7	0x00	发射功率补偿寄存器 2，2.45V
PA_POWER_TH_1	1	0xC8	0x00	发射功率补偿寄存器 1，2.25V
PA_POWER_TH_0	1	0xC9	0x00	发射功率补偿寄存器 0，2.05V

20 封装外形

CMT2187A 的封装信息如下图及下表所示。

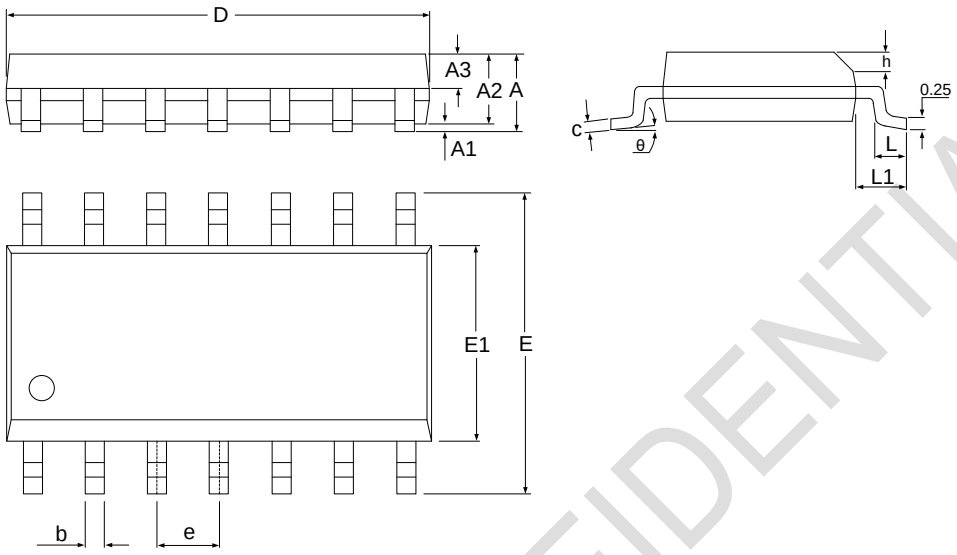

图 21-1. SOP14 封装

表 21-1. SOP14 封装尺寸

符号	尺寸 (毫米 mm)		
	最小值	典型值	最大值
A	-	-	1.75
A1	0.05	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.48
c	0.21	-	0.26
D	8.45	8.65	8.85
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27 BSC		
L	0.25	-	0.50
L1	1.05 BSC		
θ	0	-	8°

21 顶部丝印

图 22-1. CMT2187A 顶部丝印

表 22-1. CMT2187A 顶部丝印说明

丝印方式	激光
管脚 1 标记	圆圈直径=1 mm
字体尺寸	0.6 mm，右对齐
字体宽度	0.4 mm
第一行丝印	CMT2187A，代表型号 CMT2187A；
第二行丝印	YYWW 是封装厂制定的日期编号。YY 代表年份的最后 2 位数，WW 代表工作周 ①②③④⑤⑥是内部追踪号

22 其它文档

表 23-1. CMT2187A 相关其它文档

文档号	文档名称	描述
	CMT2187A 开发环境搭建	开发环境搭建，初步上手指南
	CMT2187A 寄存器详细手册	寄存器详细说明手册

23 文档变更记录

表 24-1. CMT2187A 用户手册变更记录

版本号	变更章节	变更记录	发布日期
0.1	所有	初始版本	2024/11/11
0.2	所有	文字勘误	2024/11/25

24 联系方式

深圳市华普微电子股份有限公司

中国广东省深圳市南山区西丽街道万科云城三期 8A 栋 30 层

邮编： 518052

电话： +86 - 755 - 82973805

销售： sales@hoperf.com

网址： www.hoperf.cn

附录 A

表 10-1.T8051XC3 指令集

No.	Mnemonic	Description	Opcode	Bytes	Cycles
Arithmetic Operations					
1	ADD A, Rn	Add Register to Accumulator	0x28.. ..0x2F	1	3
2	ADD A, direct	Add direct data to Accumulator	0x25	2	3
3	ADD A, @Ri	Add indirect data to Accumulator	0x26 0x27	1	4
4	ADD A, #data	Add immediate data to Accumulator	0x24	2	2
5	ADDC A, Rn	Add Register to Accumulator with Carry	0x38.. ..0x3F	1	3
6	ADDC A, direct	Add direct data to Accumulator with Carry	0x35	2	3
7	ADDC A, @Ri	Add indirect data to Accumulator with Carry	0x36 0x37	1	4
8	ADDC A, #data	Add immediate data to Accumulator with Carry	0x34	2	2
9	SUBB A, Rn	Subtract Register from Accumulator with Borrow	0x98.. ..0x9F	1	3
10	SUBB A, direct	Subtract direct data from Accumulator with Borrow	0x95	2	3
11	SUBB A, @Ri	Subtract indirect data from Accumulator with Borrow	0x96 0x97	1	4
12	SUBB A, #data	Subtract immediate data from Accumulator with Borrow	0x94	2	2
13	INC A	Increment Accumulator	0x04	1	1
14	INC Rn	Increment Register	0x08.. ..0x0F	1	3
15	INC direct	Increment direct data	0x05	2	3
16	INC @Ri	Increment indirect data	0x06 0x07	1	4
17	INC DPTR	Increment Data Pointer register	0xA3	1	2
18	DEC A	Decrement Accumulator	0x14	1	1
19	DEC Rn	Decrement Register	0x18.. ..0x1F	1	3
20	DEC direct	Decrement direct data	0x15	2	3
21	DEC @Ri	Decrement indirect data	0x16 0x17	1	4
22	MUL AB	Multiply A by B	0xA4	1	8
23	DIV AB	Divide A by B	0x84	1	8

No.	Mnemonic	Description	Opcode	Bytes	Cycles
24	DA A	Decimal Adjust	0xD4	1	1
Logical Operations					
25	ANL A, Rn	AND Accumulator with Register	0x58.. ..0x5F	1	3
26	ANL A, direct	AND Accumulator with direct data	0x55	2	3
27	ANL A, @Ri	AND Accumulator with indirect data	0x56 0x57	1	4
28	ANL A, #data	AND Accumulator with immediate data	0x54	2	2
29	ANL direct, A	AND direct data with Accumulator	0x52	2	3
30	ANL direct, #data	AND direct data with immediate data	0x53	3	3
31	CLR A	Clear Accumulator	0xE4	1	1
32	CPL A	Couple Accumulator	0xF4	1	1
33	ORL A, Rn	OR Accumulator with Register	0x48.. ..0x4F	1	3
34	ORL A, direct	OR Accumulator with direct data	0x45	2	3
35	ORL A, @Ri	OR Accumulator with indirect data	0x46 0x47	1	4
36	ORL A, #data	OR Accumulator with immediate data	0x44	2	2
37	ORL direct, A	OR direct data with Accumulator	0x42	2	3
38	ORL direct, #data	OR direct data with immediate data	0x43	3	3
39	RL A	Rotate Accumulator left	0x23	1	1
40	RLC A	Rotate Accumulator left through Carry	0x33	1	1
41	RR A	Rotate Accumulator right	0x03	1	1
42	RRC A	Rotate Accumulator right through Carry	0x13	1	1
43	SWAP A	Swap Accumulator	0xC4	1	1
44	XRL A, Rn	Exclusive-OR Accumulator with Register	0x68.. ..0x6F	1	3
45	XRL A, direct	Exclusive-OR Accumulator with direct data	0x65	2	3
46	XRL A, @Ri	Exclusive-OR Accumulator with indirect data	0x66.. ..0x67	1	4
47	XRL A, #data	Exclusive-OR Accumulator with immediate data	0x64	2	3
48	XRL direct, A	Exclusive-OR direct data with Accumulator	0x62	2	3
49	XRL direct, #data	Exclusive-OR direct data with immediate data	0x63	3	3
Data Transfer Operations					
50	MOV A, Rn r	Move Register to Accumulator	0xE8.. ..0xEF	1	2
51	MOV A, direct	Move direct data to Accumulator	0xE5	2	2

No.	Mnemonic	Description	Opcode	Bytes	Cycles
52	MOV A, @Ri	Move indirect data to Accumulator	0xE6 0xE7	1	3
53	MOV A, #data	Move immediate data to Accumulator	0x74	2	2
54	MOV Rn, A	Move Accumulator to Register	0xF8.. ..0xFF	1	1
55	MOV Rn, direct	Move direct data to Register	0xA8.. ..0xAF	2	2
56	MOV Rn, #data	Move immediate data to Register	0x78.. ..0x7F	2	2
57	MOV direct, A	Move Accumulator to direct	0xF5	2	2
58	MOV direct, Rn	Move Register to direct	0x88.. ..0x8F	2	3
59	MOV direct, direct	Move direct data to direct	0x85	3	3
60	MOV direct, @Ri	Move indirect data to direct	0x86 0x87	2	4
61	MOV direct, #data	Move immediate data to direct	0x75	3	3
62	MOV @Ri, A	Move Accumulator to indirect	0xF6 0xF7	1	2
63	MOV @Ri, direct	Move direct data to indirect	0xA6 0xA7	2	3
64	MOV @Ri, #data	Move immediate data to indirect	0x76 0x77	2	3
65	MOV DPTR, #data16	Move 16-bit immediate data to Data Pointer	0x90	3	3
66	MOVC A, @A+DPTR	Move Code relative to DPTR to Accumulator	0x93	1	4
67	MOVC A, @A+PC	Move Code relative to PC to Accumulator	0x83	1	4
68	MOVX A, @Ri	Move XDATA (by Register) to Accumulator	0xE2 0xE3	1	4
69	MOVX A, @DPTR	Move XDATA (by DPTR) to Accumulator	0xE0	1	3
70	MOVX @Ri, A	Move Accumulator to XDATA (by Register)	0xF2 0xF3	1	4
71	MOVX @DPTR, A	Move Accumulator to XDATA (by DPTR)	0xF0	1	3
72	PUSH direct	Push direct data onto stack	0xC0	2	2
73	POP direct	Pop direct from stack	0xD0	2	3
74	XCH A, Rn	Exchange Accumulator with Register	0xC8.. ..0xCF	1	2
75	XCH A, direct	Exchange Accumulator with direct	0xC5	2	2
76	XCH A, @Ri	Exchange Accumulator with indirect	0xC6 0xC7	1	3

No.	Mnemonic	Description	Opcode	Bytes	Cycles
77	XCHD A, @Ri	Exchange Accumulator nibble with indirect	0xD6 0xD7	1	3
Boolean (Bit-Wise) Operations					
78	ANL C, bit	AND Carry with direct bit	0x82	2	2
79	ANL C, /bit	AND Carry with direct bit inverted	0xB0	2	2
80	CLR C	Clear Carry	0xC3	1	1
81	CLR bit	Clear direct bit	0xC2	2	3
82	CPL C	Couple Carry	0xB3	1	1
83	CPL bit	Couple direct bit	0xB2	2	3
84	MOV C, bit	Move direct bit to Carry	0xA2	2	2
85	MOV bit, C	Move Carry to direct bit	0x92	2	3
86	ORL C, bit	OR Carry with direct bit	0x72	2	2
87	ORL C, /bit	OR Carry with direct bit inverted	0xA0	2	2
88	SETB C	Set Carry	0xD3	1	1
89	SETB bit	Set direct bit	0xD2	2	3
Program Branch Operation					
90	ACALL addr11	Absolute call	0baaa10001	2	3
91	AJMP addr11	Absolute jump	0baaa00001	2	3
92	CJNE A, direct, rel	Compare Accumulator to direct data and jump if not equal	0xB5	3	4
93	CJNE A, #data, rel	Compare Accumulator to immediate data and jump if not equal	0xB4	3	3
94	CJNE Rn, #data, rel	Compare Register to immediate data and jump if not equal	0xB8.. ..0xBF	3	4
95	CJNE @Ri, #data, rel	Compare indirect to immediate data and jump if not equal	0xB6 0xB7	3	5
96	DJNZ Rn, rel	Decrement Register and jump if not zero	0xD8.. ..0xDF	2	3
97	DJNZ, direct, rel	Decrement direct and jump if not zero	0xD5	3	4
98	JB bit, rel	Jump if direct bit is set	0x20	3	3
99	JBC bit, rel	Jump if direct bit is set and clear it	0x10	3	3
100	JC rel	Jump if Carry is set	0x40	2	3
101	JZ rel	Jump if Accumulator is zero	0x60	2	3
102	JMP @A+DPTR	Jump relatively to DPTR	0x73	1	2
103	JNC rel	Jump if Carry is cleared	0x50	2	3
104	JNB bit, rel	Jump if direct bit is cleared	0x30	3	3
105	JNZ rel	Jump if Accumulator is not zero	0x70	2	3
106	LCALL addr16	Long call	0x12	3	4
107	LJMP addr16	Long jump	0x02	3	3

No.	Mnemonic	Description	Opcode	Bytes	Cycles
108	RET	Return from subroutine	0x22	1	3
109	RETI	Return from interrupt	0x32	1	3
110	SJMP rel	Short jump	0x80	2	3
111	NOP	No operation	0x00	1	1