C8051F020/1/2/3 MCU

1、概述
2、交叉开关配置
3、系统时钟源
4、系统复位
5、JTAG接口的在系统调试
6、单片机的初始化设置

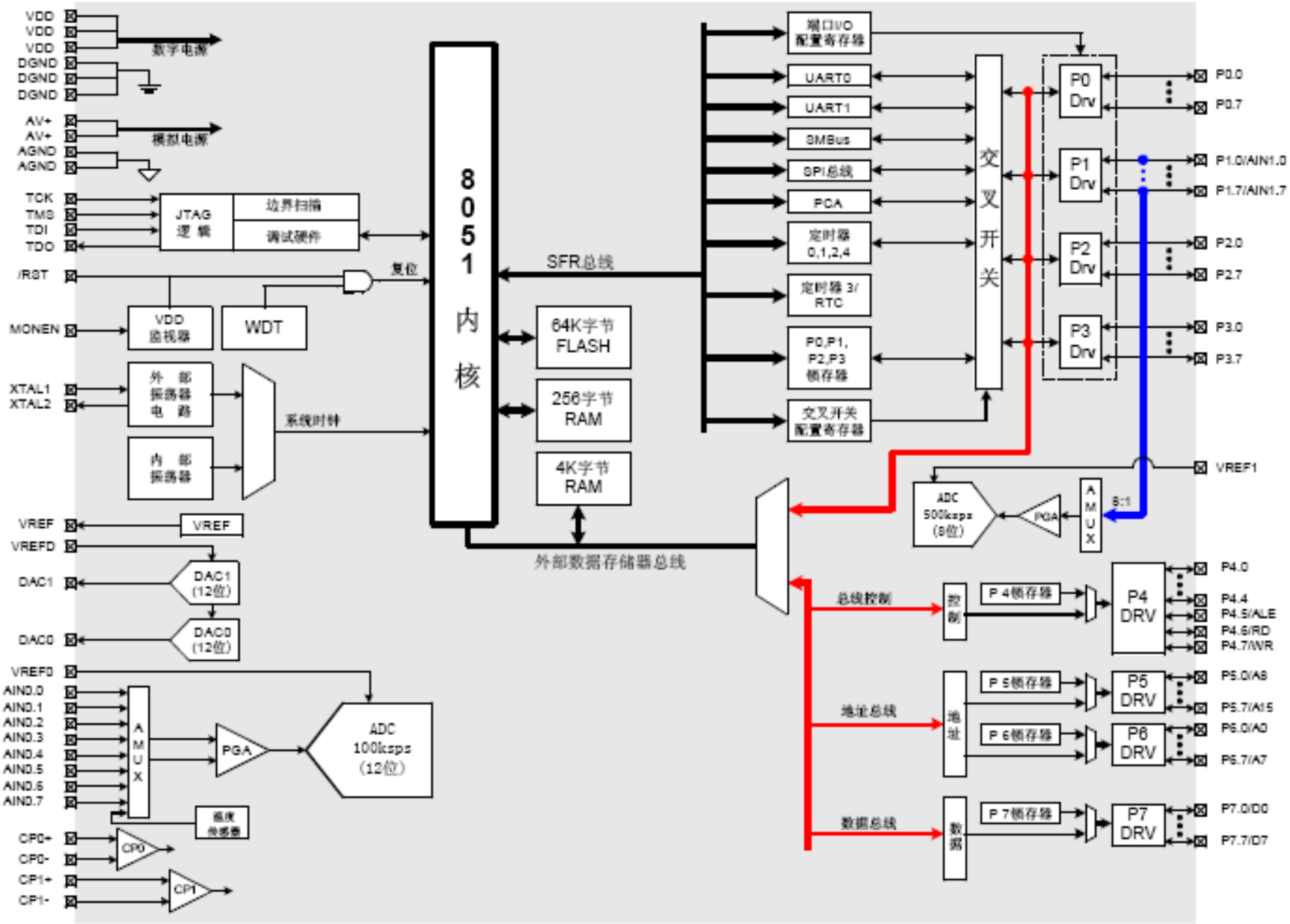
1、概述

C8051F020器件是完全集成的混合信号系统级MCU 芯片，具有64 个数字I/O 引脚（C8051F020/2）或32 个数字I/O 引脚（C8051F021/3）。下面列出了一些主要特性：

- 高速、流水线结构的8051 兼容的CIP-51 内核（可达25MIPS）
- 全速、非侵入式的在系统调试接口（片内）
- 真正12 位（C8051F020/1）或10 位（C8051F022/3）、 100 ksps 的8 通道ADC，带PGA和模拟多路开关
- 真正8 位500 ksps 的ADC，带PGA 和8 通道模拟多路开关
- 两个12 位DAC，具有可编程数据更新方式
- 64K 字节可在系统编程的FLASH 存储器
- 4352（4096+256）字节的片内RAM

- 可寻址64K 字节地址空间的外部数据存储器接口
- 硬件实现的SPI、SMBus/ I2C 和两个UART 串行接口
- 5 个通用的16 位定时器
- 具有5 个捕捉/比较模块的可编程计数器/定时器阵列
- 片内看门狗定时器、VDD 监视器和温度传感器

图 1.1 C8051F020 原理框图



1.1 CIP-51TM CPU

1.1.1 与8051 完全兼容

C8051F020 系列器件使用Silicon Labs 的专利CIP-51微控制器内核。CIP-51 与MCS-51TM指令集完全兼容，可以使用标准803x/805x 的汇编器和编译器进行软件开发。

CIP-51 内核具有标准8052 的所有外设部件，包括：

- 5 个16 位的计数器/定时器、
- 两个全双工UART、256 字节内部RAM、
- 128 字节特殊功能寄存器（SFR）地址空间、
- 8/4 个字节宽的I/O 端口。

1.1.2 速度提高

CIP-51 采用流水线结构，与标准的8051 结构相比指令执行速度有很大的提高。

在一个标准的8051 中，除MUL 和DIV 以外所有指令都需要12 或24 个系统时钟周期，最大系统时钟频率为12-24MHz。

而对于CIP-51 内核，70%的指令的执行时间为1 或2 个系统时钟周期，只有4 条指令的执行时间大于4 个系统时钟周期。

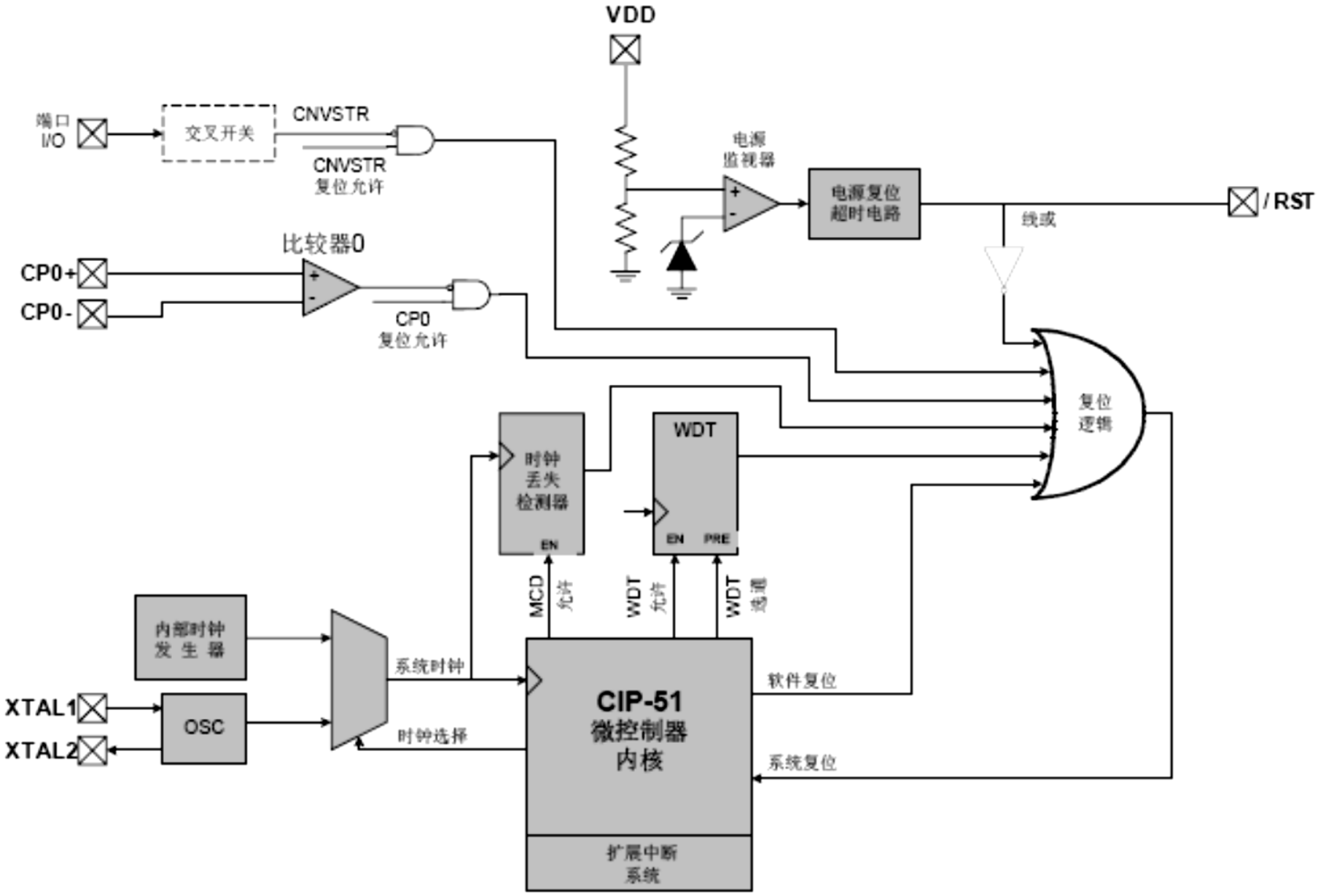
1.1.3 增加的功能

- 扩展的中断系统向CIP-51 提供22 个中断源（标准8051 只有7 个中断源），允许大量的模拟和数字外设中断微控制器。

- MCU 可有多达7 个复位源：一个片内VDD 监视器、一个看门狗定时器、一个时钟丢失检测器、一个由比较器0 提供的电压检测器、一个软件强制复位、CNVSTR 引脚及/RST 引脚。

- MCU 内部有一个独立运行的时钟发生器，在复位后被默认为系统时钟。如果需要，时钟源可以在运行时切换到外部振荡器，外部振荡器可以使用晶体、陶瓷谐振器、电容、RC 或外部时钟源产生系统时钟。

图 1.6 片内时钟和复位电路



1.2 片内存储器

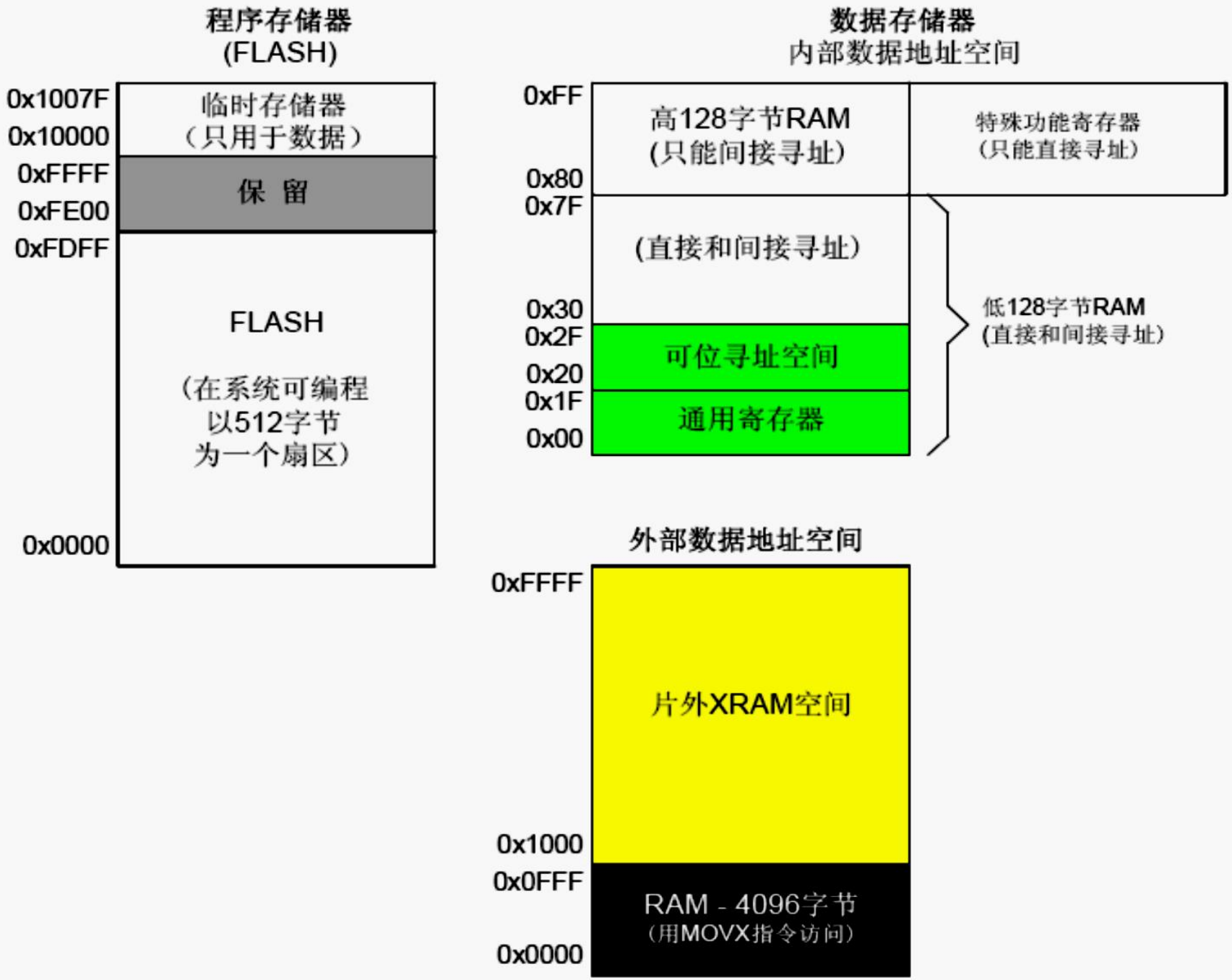
- **CIP-51 有标准的8051 程序和数据地址配置。**

- **C8051F020中的CIP-51 还另有位于外部数据存储器地址空间的4K 字节的RAM 块和一个可用于访问外部数据存储器的外部存储器接口（EMIF）。**

这个片内的4K 字节RAM 块可以在整个64K 外部数据存储器地址空间中被寻址（以4K 为边界重叠）。外部数据存储器地址空间可以只映射到片内存储器、只映射到片外存储器、或两者的组合（4K 以下的地址指向片内，4K 以上的地址指向EMIF）。EMIF 可以被配置为地址/数据线复用方式或非复用方式。

- **MCU 的程序存储器包含64K 字节的FLASH。该存储器以512 字节为一个扇区，可以在系统编程，且不需特别的外部编程电压。**

图 1.7 片内存储器组织


程序存储器
(FLASH)
0x1007F
0x10000
0xFFFF
0xFE00
0xFDFF
0x0000
临时存储器
（只用于数据）
保 留
FLASH
（在系统可编程
以512字节
为一个扇区）
数据存储器
内部数据地址空间
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
高128字节RAM
(只能间接寻址)
特殊功能寄存器
（只能直接寻址）
(直接和间接寻址)
可位寻址空间
通用寄存器
低128字节RAM
(直接和间接寻址)
外部数据地址空间
0xFFFF
0x1000
0x0FFF
0x0000
片外XRAM空间
RAM - 4096字节
（用MOVX指令访问）

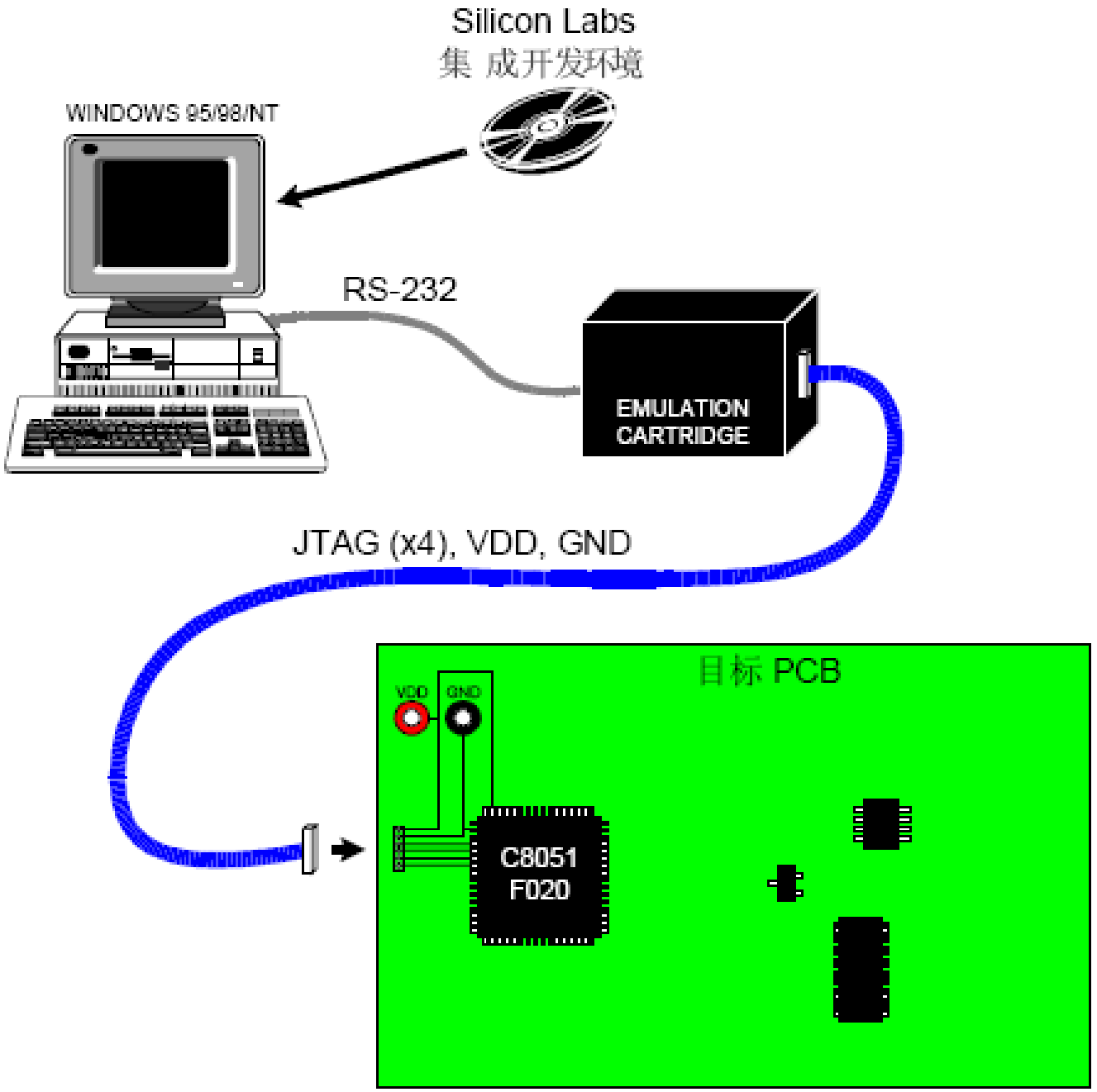
1.3 JTAG 调试和边界扫描

C8051F020系列具有片内JTAG边界扫描和调试电路，通过4脚JTAG接口并使用安装在最终应用系统中的产品器件就可以进行非侵入式、全速的在系统调试。

该JTAG接口完全符合IEEE1149.1规范，为生产和测试提供完全的边界扫描功能。

Silicon Labs的调试系统支持观察和修改存储器和寄存器，支持断点、观察点、堆栈指示器和单步执行。

图1.8 调试环境示意图


Silicon Labs
集 成开发环境
WINDOWS 95/98/NT
RS-232
EMULATION
CARTRIDGE
JTAG (x4), VDD, GND
目标 PCB
VDD
GND
C8051
F020

1.4 可编程数字I/O 和交叉开关

该系列MCU具有标准8051的端口（0、1、2和3）。在F020/2中有4个附加的端口（4、5、6和7），因此共有64个通用端口I/O。

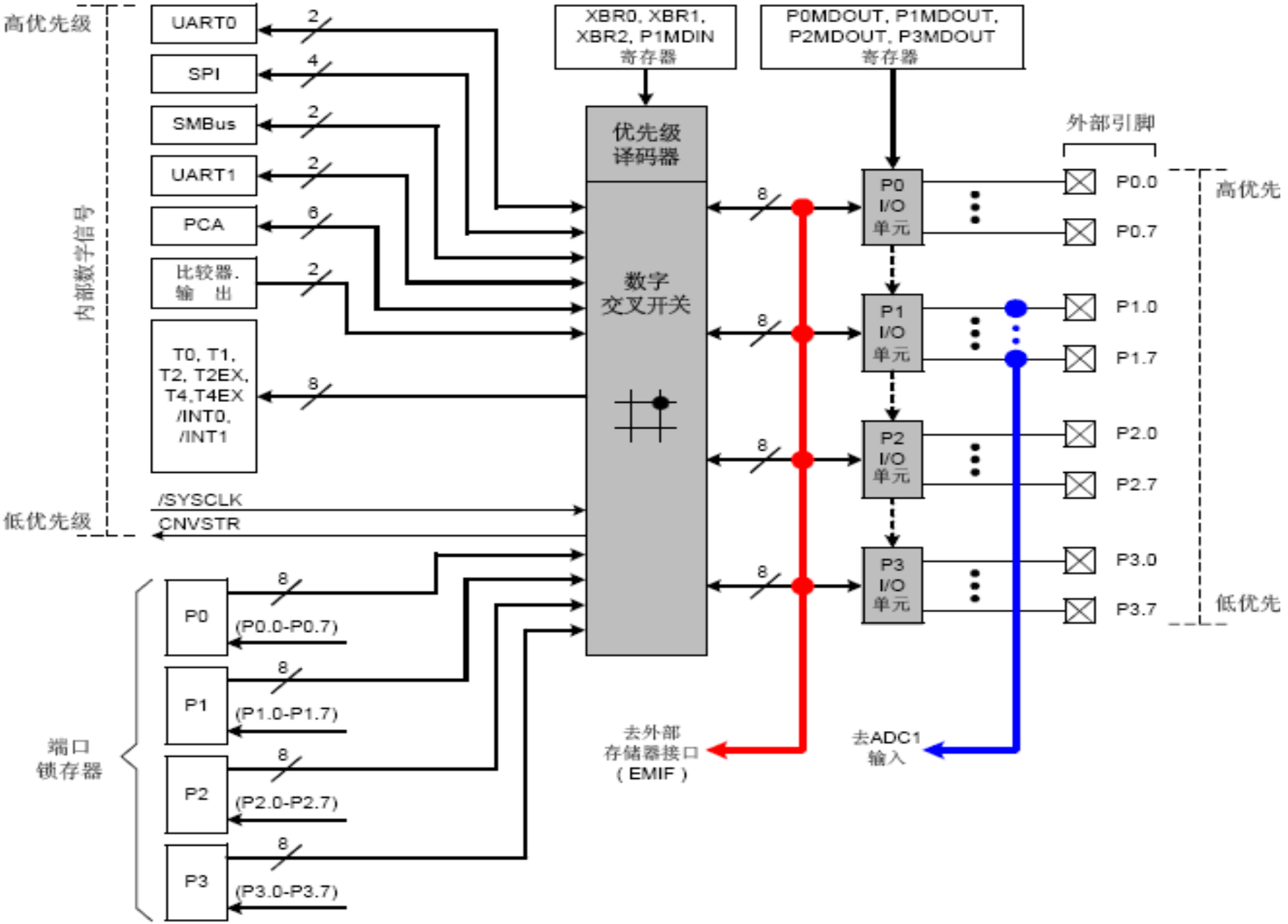
- 每个端口I/O引脚都可以被配置为推挽或漏极开路输出。
- 数字交叉开关。

这是一个大的数字开关网络，允许将内部数字系统资源映射到P0、P1、P2和P3的端口I/O引脚。

可通过设置交叉开关控制寄存器将片内的计数器/定时器、串行总线、硬件中断、ADC转换启动输入、比较器输出以及微控制器内部的其它数字信号配置为出现在端口I/O引脚。

这一特性允许用户根据自己的特定应用选择通用端口I/O和所需数字资源的组合。

图1.9 数字交叉开关原理框图



1.5 可编程计数器阵列

除了5个16位的通用计数器/定时器之外，C8051F020 MCU系列还有一个片内可编程计数器/定时器阵列（PCA）。

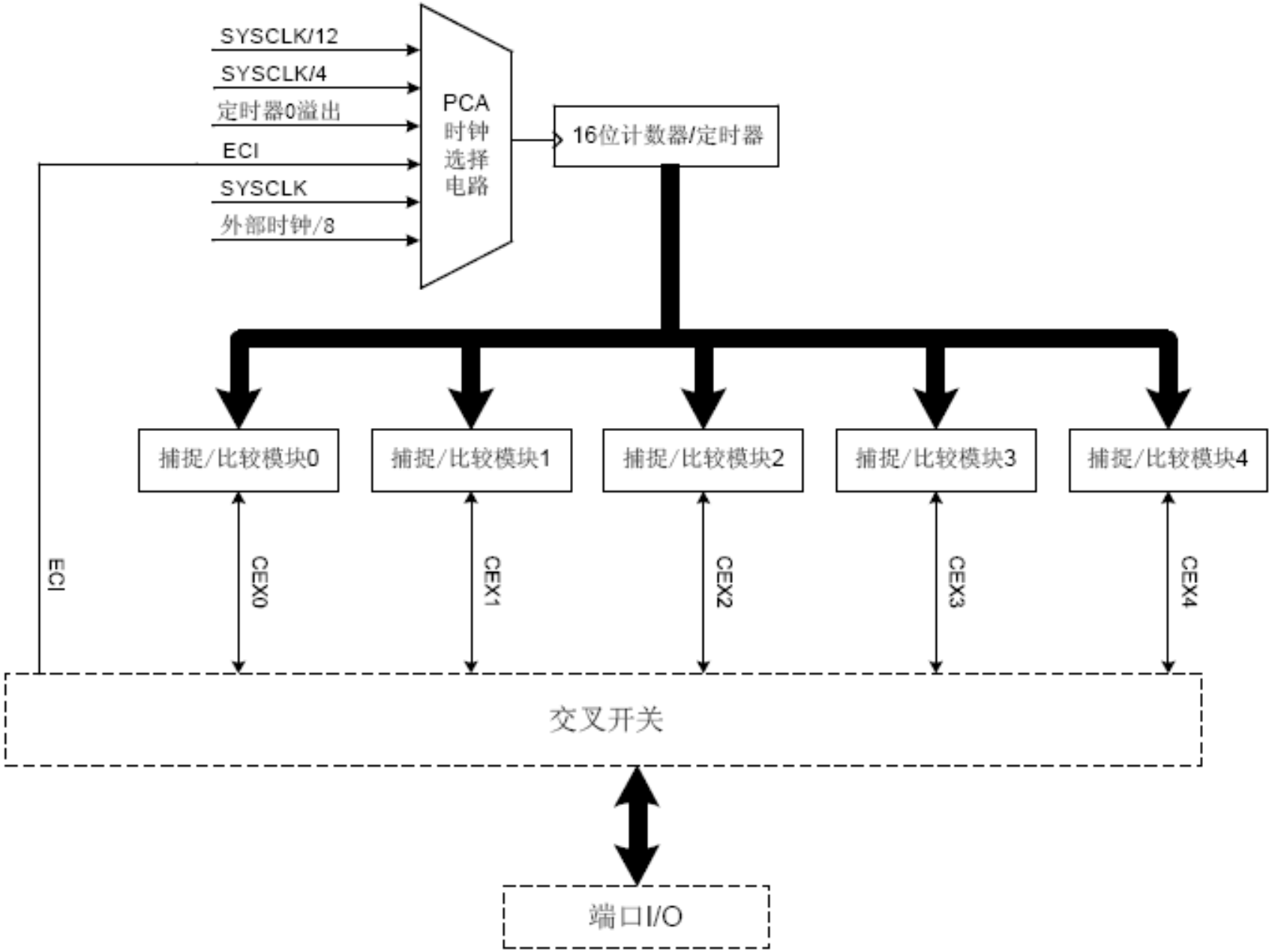
PCA包括一个专用的16位计数器/定时器时间基准和5个可编程的捕捉/比较模块。

时间基准的时钟可以是下面的六个时钟源之一：系统时钟/12、系统时钟/4、定时器0溢出、外部时钟输入（ECI）、系统时钟和外部振荡源频率/8。

每个捕捉/比较模块都有六种工作方式：边沿触发捕捉、软件定时器、高速输出、频率输出、8位脉冲宽度调制器和16位脉冲宽度调制器。

PCA捕捉/比较模块的I/O和外部时钟输入可以通过数字交叉开关连到MCU的端口I/O引脚。

图1.10　PCA原理框图



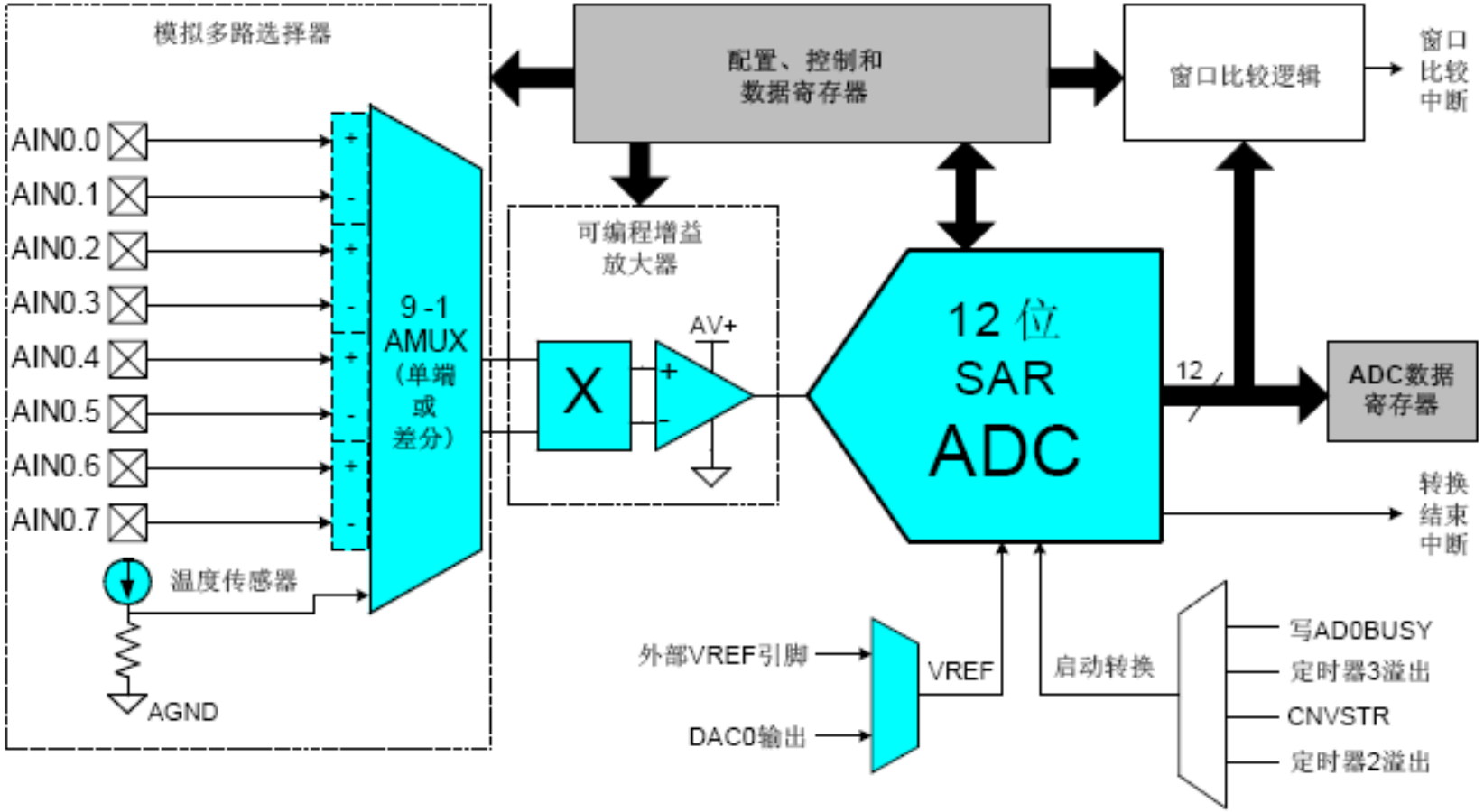
1.6 串行端口

- C8051F020系列MCU内部有两个增强型全双工UART、SPI总线和SMBus/I2C。
- 每种串行总线都完全用硬件实现，都能向CIP-51产生中断，因此需要很少的CPU干预。
- 这些串行总线不“共享”定时器、中断或端口I/O等资源，所以可以使用任何一个或全部同时使用。

1.7 12 位模/数转换器

C8051F020/1有一个片内**12**位**SAR ADC**（**ADC0**），一个**9**通道输入多路选择开关和可编程增益放大器。该**ADC**工作在**100ksps**的最大采样速率时可提供真正的**12**位精度，**INL**为±**1LSB**。

图1.11　12位ADC原理框图



- ADC完全由CIP-51通过特殊功能寄存器控制。有一个输入通道被连到内部温度传感器，其它8个通道接外部输入。8个外部输入通道的每一对都可被配置为两个单端输入或一个差分输入。

- 可编程增益放大器接在模拟多路选择器之后，增益可以用软件设置，从0.5到16以2的整数次幂递增。

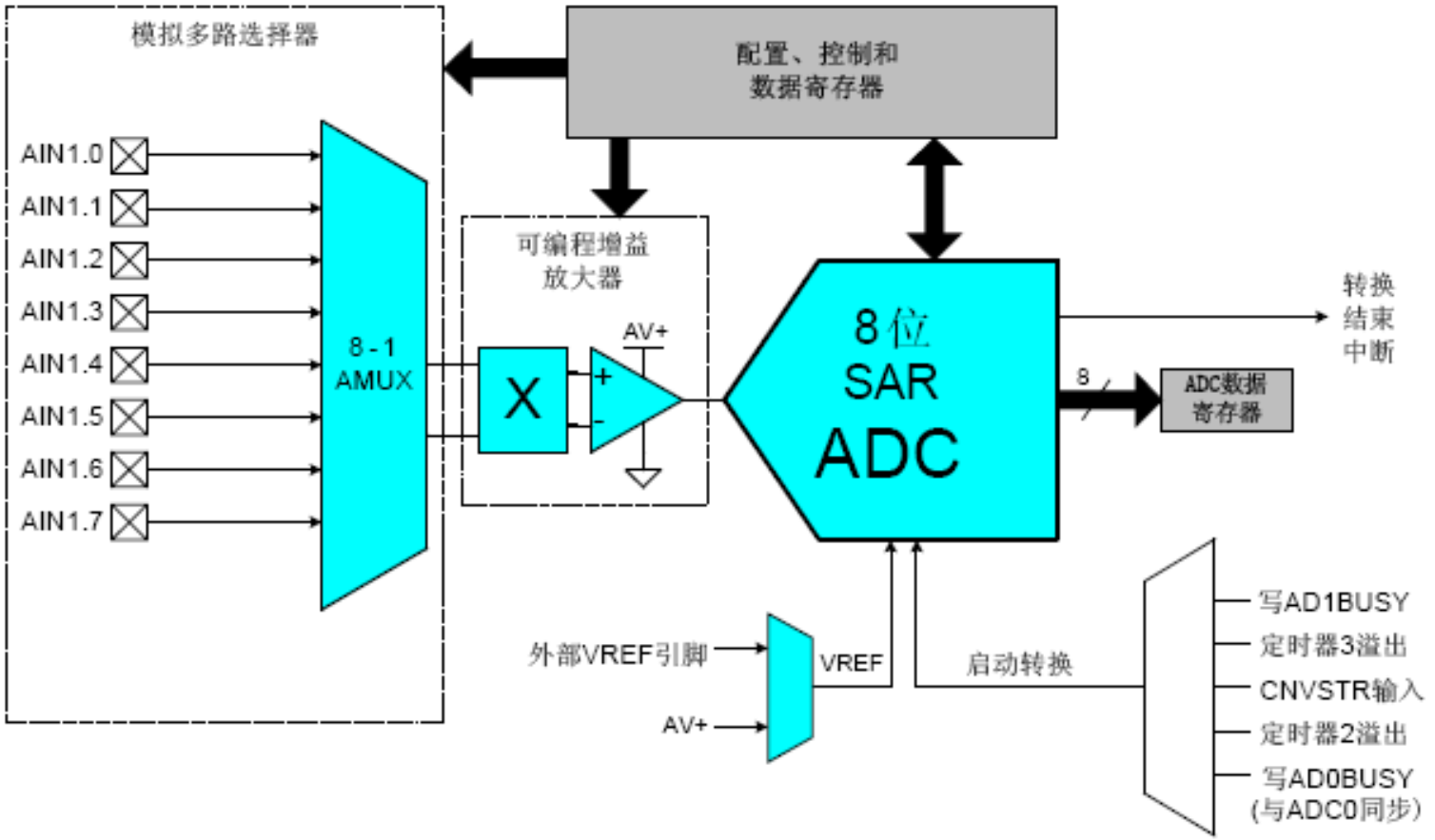
- A/D转换有4种启动方式：软件命令、定时器2溢出、定时器3溢出和外部信号输入。这种灵活性允许用软件事件、外部硬件信号或周期性的定时器溢出信号触发转换。

- 转换结束由一个状态位指示，或者产生中断（如果中断被使能）。在转换完成后，10或12位转换结果数据字被锁存到两个特殊功能寄存器中。这些数据字可以用软件控制为左对齐或右对齐。

1.8 8 位模/数转换器

C8051F020有一个片内8位SAR ADC（ADC1），带有一个8通道输入多路选择器和可编程增益放大器。该ADC工作在500ksps的最大采样速率时可提供真正的8位精度，INL为±1LSB。有8个用于测量的输入端。ADC1完全由CIP-51通过特殊功能寄存器控制。ADC0的电压基准可以在模拟电源电压（AV+）和一个外部VREF引脚之间选择。

图1.12　8位ADC原理框图



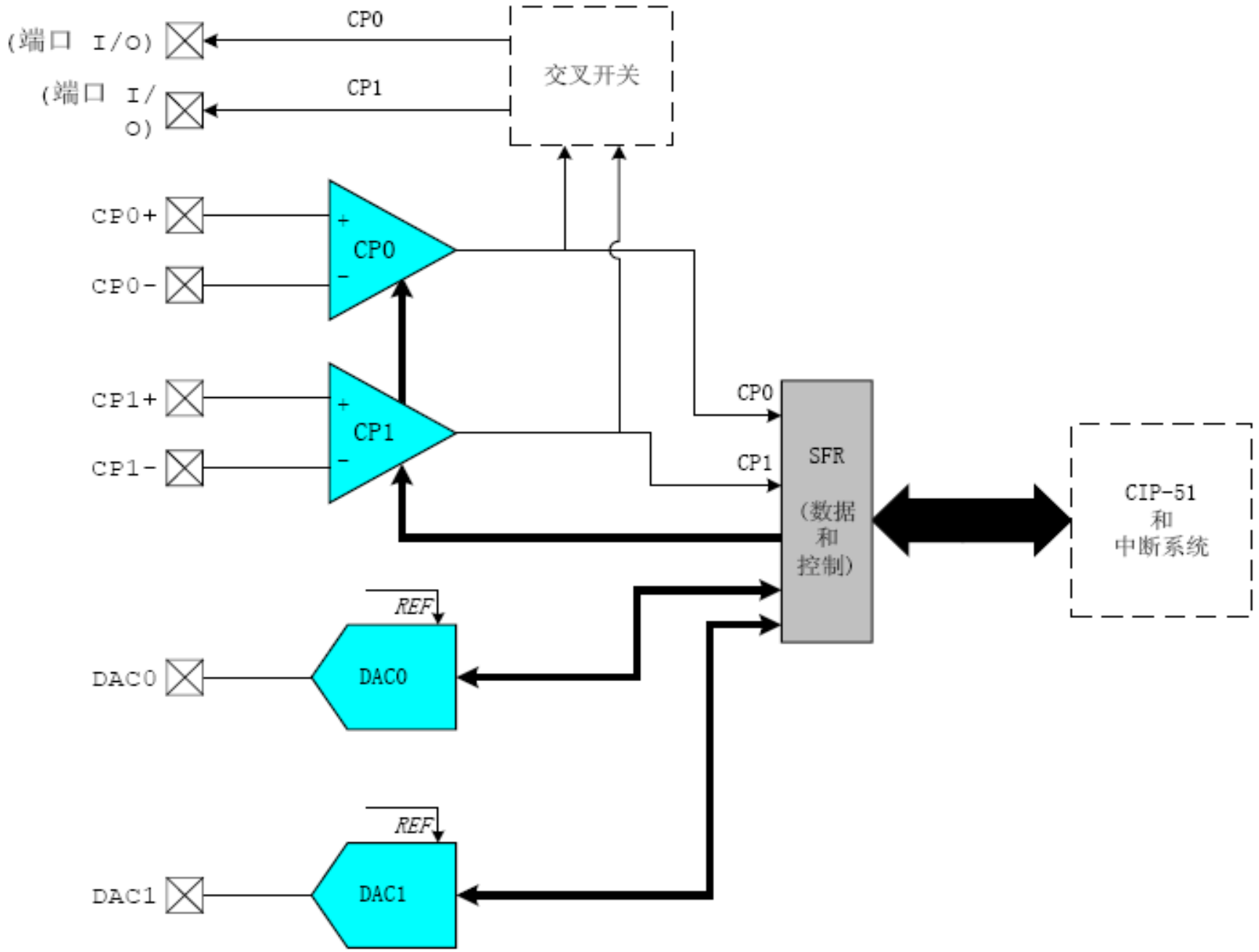
1.9 比较器和DAC

C8051F020/1/2/3系列MCU内部有两个12位DAC和两个比较器。MCU与每个比较器和DAC之间的数据和控制接口通过特殊功能寄存器实现。MCU可以将任何一个DAC或比较器置于低功耗关断方式。

比较器的回差电压可以用软件编程。每个比较器都能在上升沿、下降沿或在两个边沿都产生中断。这些中断能将MCU从休眠方式唤醒。比较器的输出状态可以用软件查询。可通过设置交叉开关将比较器的输出接到端口I/O引脚。

DAC为电压输出方式，有灵活的输出更新机制。这一机制允许用软件写和定时器2、定时器3及定时器4的溢出信号更新DAC输出。C8051F020/2的DAC之电压基准由专用的VREFD输入引脚提供，而C8051F021/3的DAC之电压基准由器件内部的电压基准提供。DAC在作为比较器的参考电压或为ADC差分输入提供偏移电压时非常有用。

图1.13 比较器和DAC原理框图



2、交叉开关配置

2.1 端口输入/输出

低端口（P0、P1、P2 和P3）既可以按位寻址也可以按字节寻址。高端口（P4、P5、P6 和P7）只能按字节寻址。

所有引脚都耐5V 电压，都可以被配置为漏极开路或推挽输出方式和弱上拉。

图 17.1　端口 I/O 单元功能框图

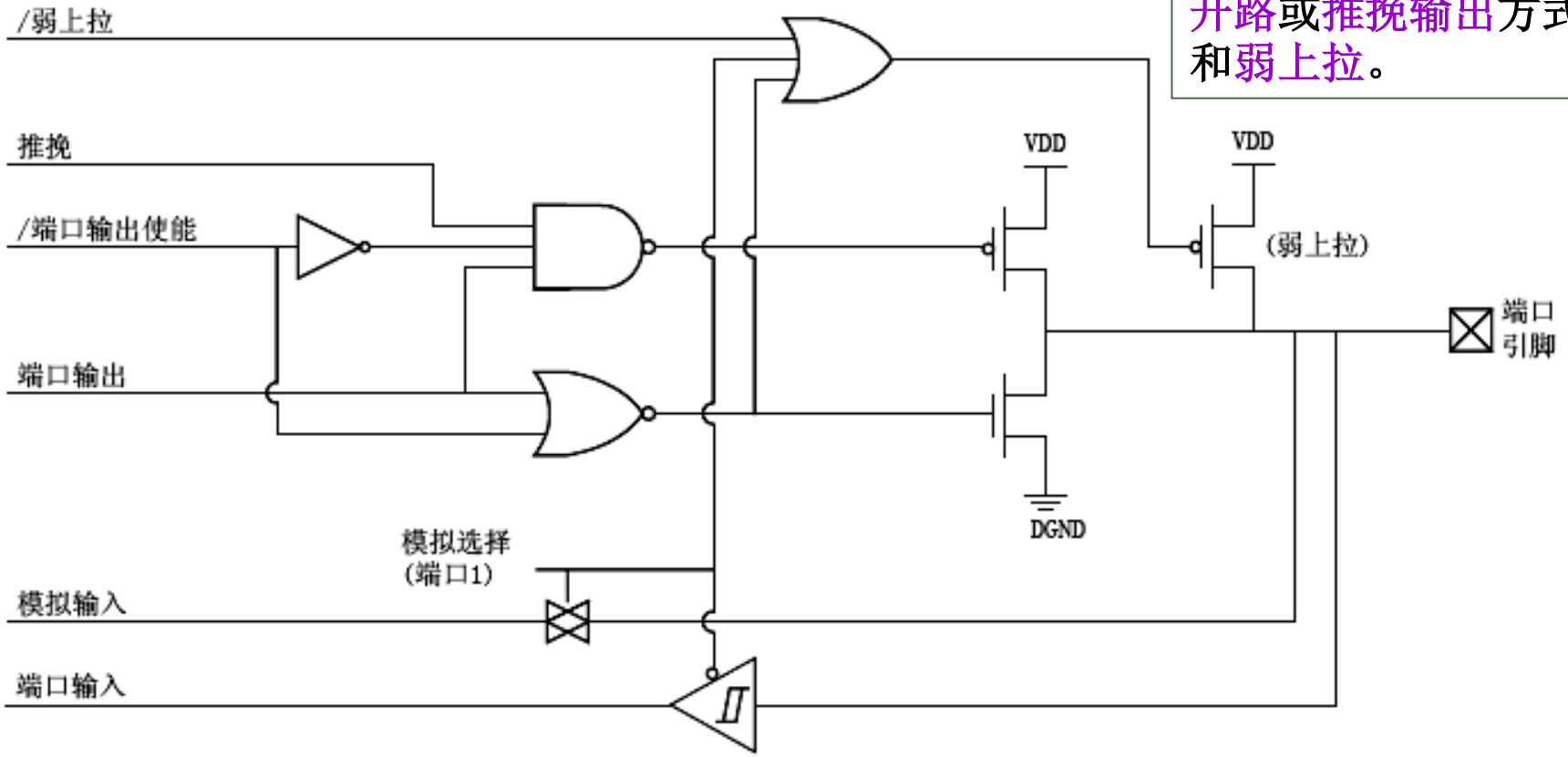


表 17.1　端口 I/O 直流电气特性

VDD = 2.7V – 3.6V, -40℃到+85℃（除非另有说明）。

参　数	条　件	最小值	典型值	最大值	单 位
输出高电压（V_{OH}）	I_{OH} = -10μA，端口 I/O 为推挽方式 I_{OH} = -3mA，端口 I/O 为推挽方式 I_{OH} = -10mA，端口 I/O 为推挽方式	VDD-0.1 VDD-0.7	 VDD-0.8		V
输出低电压（V_{OL}）	I_{OL} = 10μA I_{OL} = 8.5mA I_{OL} = 25mA		 1.0	0.1 0.6	V
输入高电压（V_{IH}）		0.7×VDD			V
输入低电压（V_{IL}）				0.3×VDD	V
输入漏电流	DGND < 端口引脚 < VDD，高阻态 弱上拉禁止 弱上拉使能		 10	±1	μA
输入电容			5		pF

C8051F020/1/2/3 器件有大量的数字资源需要通过4 个低端I/O 端口P0、P1、P2 和P3 才能使用。

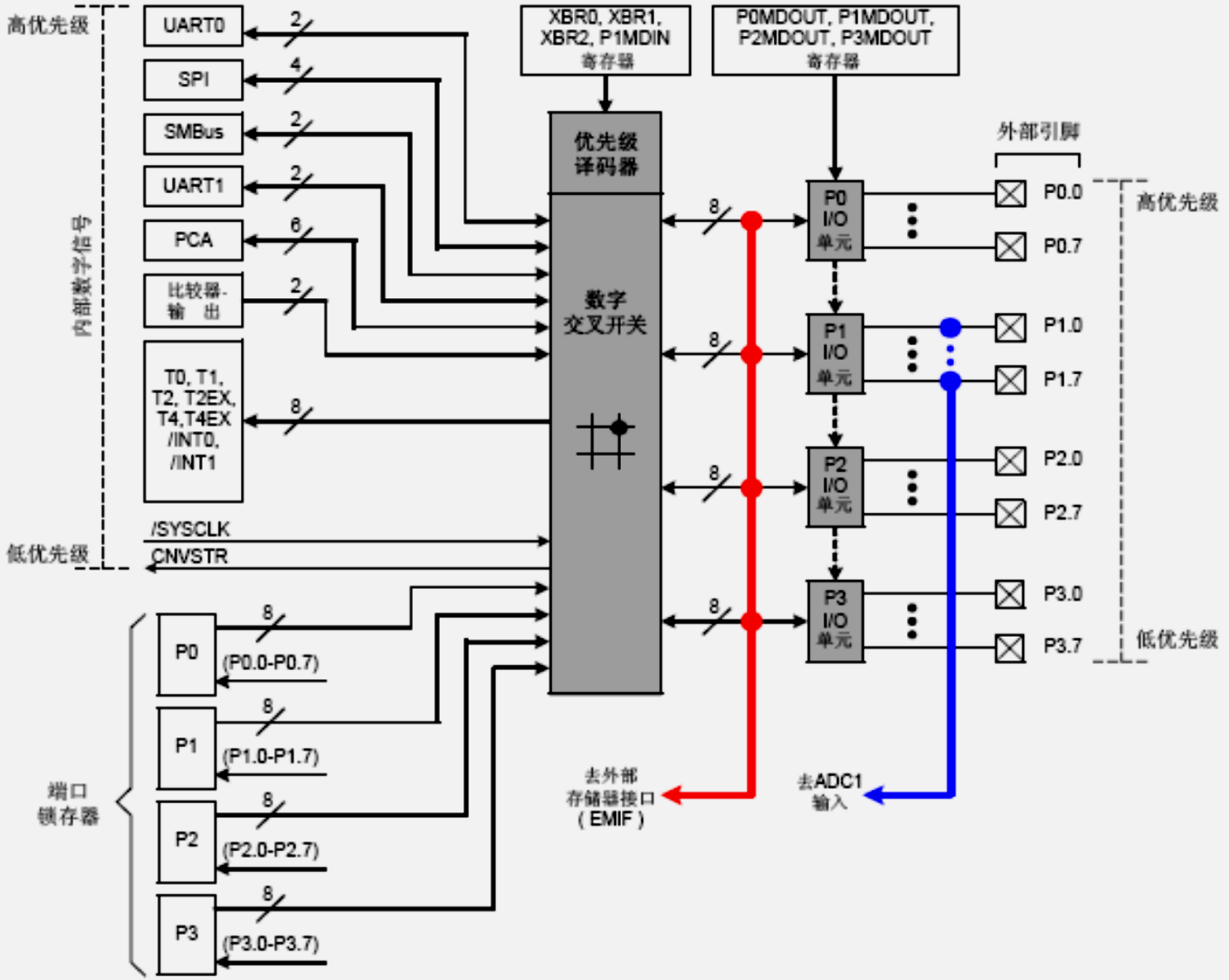
P0、P1、P2 和P3 中的每个引脚既可定义为通用的端口I/O（GPIO）引脚，又可以分配给一个数字外设或功能（例如：UART0 或/INT1），如图17.2 所示。

系统设计者控制数字功能的引脚分配，只受可用引脚数的限制。

这种资源分配的灵活性是通过使用优先权交叉开关译码器实现的。

注意，不管引脚被分配给一个数字外设或是作为通用I/O，总是可以通过读相应的数据寄存器得到端口I/O 引脚的状态。端口1 的引脚可以用做ADC1 的模拟输入。

图 17.2 低端口 I/O 功能框图



2.2 端口0 – 3 和优先权交叉开关译码器

1、优先权交叉开关译码器（“交叉开关”）

按优先权顺序将端口0–3的引脚分配给器件上的数字外设（UART、SMBus、PCA、定时器等）。端口引脚的分配顺序是从P0.0 开始，可以一直分配到P3.7。

2、优先权顺序

为数字外设分配端口引脚的优先权顺序列于图17.3，UART0 具有最高优先权，而CNVSTR 具有最低优先权。

图 17.3 优先权交叉开关译码表

（EMIFLE = 0；P1MDIN = 0xFF）

	P0								P1								P2								P3								交叉开关寄存器位
引脚I/O	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
TX0	●																																UART0EN:XBR0.2
RX0		●																															
SCK	●		●																														SPI0EN:XBR0.1
MISO		●		●																													
MOSI			●		●																												
NSS				●		●																											
SDA	●		●		●		●																										SMB0EN:XBR0.0
SCL		●		●		●		●																									
TX1	●		●		●		●		●																								UART1EN:XBR2.2
RX1		●		●		●		●	●	●																							
CEX0	●		●		●		●		●	●	●																						PCA0ME:XBR0.[5:3]
CEX1		●		●		●		●	●	●	●	●																					
CEX2			●		●		●		●	●	●	●	●																				
CEX3				●		●		●	●	●	●	●	●	●																			
CEX4					●		●		●	●	●	●	●	●	●																		
ECI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●																	ECI0E:XBR0.6
CP0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●																CP0E:XBR0.7
CP1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●															CP1E:XBR1.0
T0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●														T0E:XBR1.1
/INT0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●													INT0E:XBR1.2
T1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●												T1E:XBR1.3
/INT1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●											INT1E:XBR1.4
T2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●										T2E:XBR1.5
T2EX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									T2EXE:XBR1.6
T4	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								T4E:XBR2.3
T4EX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							T4EXE:XBR2.4
/SYSCLK	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						SYSCKE:XBR1.7
CNVSTR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					CNVSTE:XBR2.0
						ALE	/RD	/WR	AIN1.0/A8	AIN1.1/A9	AIN1.2/A10	AIN1.3/A11	AIN1.4/A12	AIN1.5/A13	AIN1.6/A14	AIN1.7/A15	A8m/A0	A9m/A1	A10m/A2	A11m/A3	A12m/A4	A13m/A5	A14m/A6	A15m/A7	AD0/D0	AD1/D1	AD2/D2	AD3/D3	AD4/D4	AD5/D5	AD6/D6	AD7/D7	
									AIN1 输入/非复用地址高								复用地址高/非复用地址低								复用数据/非复用数据								

2.1.1 交叉开关引脚分配

当交叉开关配置寄存器XBR0、XBR1 和XBR2 中外设的对应使能位被设置为逻辑‘1’时，交叉开关将端口引脚分配给外设。

图 17.7　XBR0：端口 I/O 交叉开关寄存器 0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位
CP0E	ECI0E	PCA0ME			UART0EN	SPI0EN	SMB0EN	00000
位7	位6	位5	位4	位3	位2	位1	位0	SFR地 0xE1

位 7：　CP0E：比较器 0 输出使能位
0：CP0 不连到端口引脚。
1：CP0 连到端口引脚。

位 6：　ECI0E：PCA0 外部计数器输入使能位
0：PCA0 外部计数器输入不连到端口引脚。
1：PCA0 外部计数器输入连到端口引脚。

位 5-3：　PCA0ME：PCA0 模块 I/O 使能位
000：所有的 PCA0 I/O 都不连到端口引脚。
001：CEX0 连到端口引脚。
010：CEX0、CEX1 连到 2 个端口引脚。
011：CEX0、CEX1、CEX2 连到 3 个端口引脚。

图 17.7　XBR0：端口 I/O 交叉开关寄存器 0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位
CP0E	ECI0E	PCA0ME			UART0EN	SPI0EN	SMB0EN	0000(
位7	位6	位5	位4	位3	位2	位1	位0	SFR地
								0xE1

位 2：　UART0EN：UART0 I/O 使能位。
　　0：UART0 I/O 不连到端口引脚。
　　1：UART0 的 TX 连到 P0.0，RX 连到 P0.1。
位 1：　SPI0EN：SPI 总线 I/O 使能位。
　　0：SPI0 I/O 不连到端口引脚。
　　1：SPI0 的 SCK、MISO、MOSI 和 NSS 连到 4 个端口引脚。
位 0：　SMB0EN：SMBus 总线 I/O 使能位
　　0：SMBus0 I/O 不连到端口引脚。
　　1：SMBus0 的 SDA 和 SCL 连到 2 个端口引脚。

- 端口0–3中所有未被交叉开关分配的引脚都可以作为通用I/O（GPI/O）引脚，通过读或写相应的端口数据寄存器访问。

- 被交叉开关分配的那些端口引脚的输出状态受使用这些引脚的数字外设的控制。

- 不管交叉开关是否将引脚分配给外设，读一个端口数据寄存器（或端口位）将总是返回引脚本身的逻辑状态。

- 交叉开关寄存器被正确配置后，通过将XBARE（XBR2.6）设置为逻辑‘1’来使能交叉开关。
- 在XBARE 被设置为逻辑‘1’之前，端口0-3 的输出驱动器被明确禁止，以防止对交叉开关寄存器和其它寄存器写入时在端口引脚上产生争用。

- 被交叉开关分配给输入信号（例如RX0）的引脚所对应的输出驱动器应被明确禁止；以保证端口数据寄存器和PnMDOUT 寄存器的值不影响这些引脚的状态。

2.1.2 配置端口引脚的输出方式

在XBARE（XBR2.6）被设置为逻辑‘1’之前，端口0-3 的输出驱动器保持禁止状态。

每个端口引脚的输出方式都可被配置为漏极开路或推挽方式，缺省状态为漏极开路。

1、推挽方式：

向端口数据寄存器中的相应位：

写逻辑‘0’将使端口引脚被驱动到GND，

写逻辑‘1’将使端口引脚被驱动到VDD。

2、漏极开路方式：

向端口数据寄存器中的相应位：

写逻辑‘0’将使端口引脚被驱动到GND，

写逻辑‘1’将使端口引脚处于高阻状态。

3、端口0-3 引脚的输出方式由PnMDOUT 寄存器中的对应位决定。

- 为逻辑‘1’时，配置为推挽方式；
- 为逻辑‘0’时， 配置为漏极开路方式。

◆ 不管交叉开关是否将端口引脚分配给某个数字外设，端口引脚的输出方式都受**PnMDOUT寄存器**控制。

2.1.3 配置端口引脚为数字输入

1、通过设置输出方式为“漏极开路”并向端口数据寄存器中的相应位写‘1’将端口引脚配置为数字输入。

例：

设置P3MDOUT.7 为逻辑‘0’并设置P3.7 为逻辑‘1’即可将P3.7 配置为数字输入。

2、如果一个端口引脚被交叉开关分配给某个数字外设，并且该引脚的功能为输入（例如UART0 的接收引脚RX0），则该引脚的输出驱动器被自动禁止。

2.1.4 外部中断（IE6 和IE7）

除了外部中断/INT0和/INT1（其引脚由交叉开关分配）之外，P3.6和P3.7可被配置为边沿触发的中断源，用IE6CF（P3IF.2）和IE7CF（P3IF.3）位可以将这两个中断源配置为下降沿或上升沿触发。

2.1.5 弱上拉

- 每个端口引脚都有一个内部弱上拉部件，在引脚与VDD 之间提供阻性连接（约100 kΩ），在缺省情况下该上拉器件被使能。
- 弱上拉部件可以被总体禁止，通过向弱上拉禁止位（WEAKPUD，XBR2.7）写‘1’实现。
- 当任何引脚被驱动为逻辑‘0’时，弱上拉自动取消；即输出引脚不能与其自身的上拉部件冲突。

2.1.6 配置端口1 的引脚为模拟输入（AIN.[7:0]）

端口1 的引脚可以用作ADC1 模拟多路开关的模拟输入。通过向P1MDIN 寄存器中的对应位写‘0’即可将端口引脚配置为模拟输入。

将一个端口引脚配置为模拟输入的过程如下：

1. 禁止引脚的数字输入路径。这可以防止在引脚上的电压接近VDD / 2 时消耗额外的电源电流。读端口数据位将返回逻辑‘0’，与加在引脚上的电压无关。
2. 禁止引脚的弱上拉部件。
3. 使交叉开关在为数字外设分配引脚时跳过该引脚。

注意：

1、被配置为模拟输入的引脚的输出驱动器并没有被明确地禁止。

因此 输出驱动器禁止的设置方式：

- 被配置为模拟输入的引脚所对应的P1MDOUT 位应被设置为逻辑‘0’（漏极开路方式）；
- 对应的端口数据位应被设置为逻辑‘1’（高阻态）。

2、将一个端口引脚用作ADC1 模拟多路开关的输入时并不要求将其配置为模拟输入，但强烈建议这样做。

2.1.8 交叉开关引脚分配示例

要求：

1、将配置交叉开关，为UART0、SMBus、UART1、/INT0 和/INT1 分配端口引脚（共8 个引脚）。

2、将外部存储器接口配置为复用方式并使用低端口。

3、将P1.2、P1.3 和P1.4 配置为模拟输入，以便用ADC1 测量加在这些引脚上的电压。

配置步骤如下：

1、按UART0EN = 1、UART1E = 1、SMB0EN = 1、INT0E = 1、INT1E = 1 和EMIFLE =1。设置XBR0、XBR1 和XBR2，则有：

XBR0 = 0x05，XBR1 = 0x14，XBR2 = 0x06。

2、将外部存储器接口配置为复用方式并使用低端口，有：

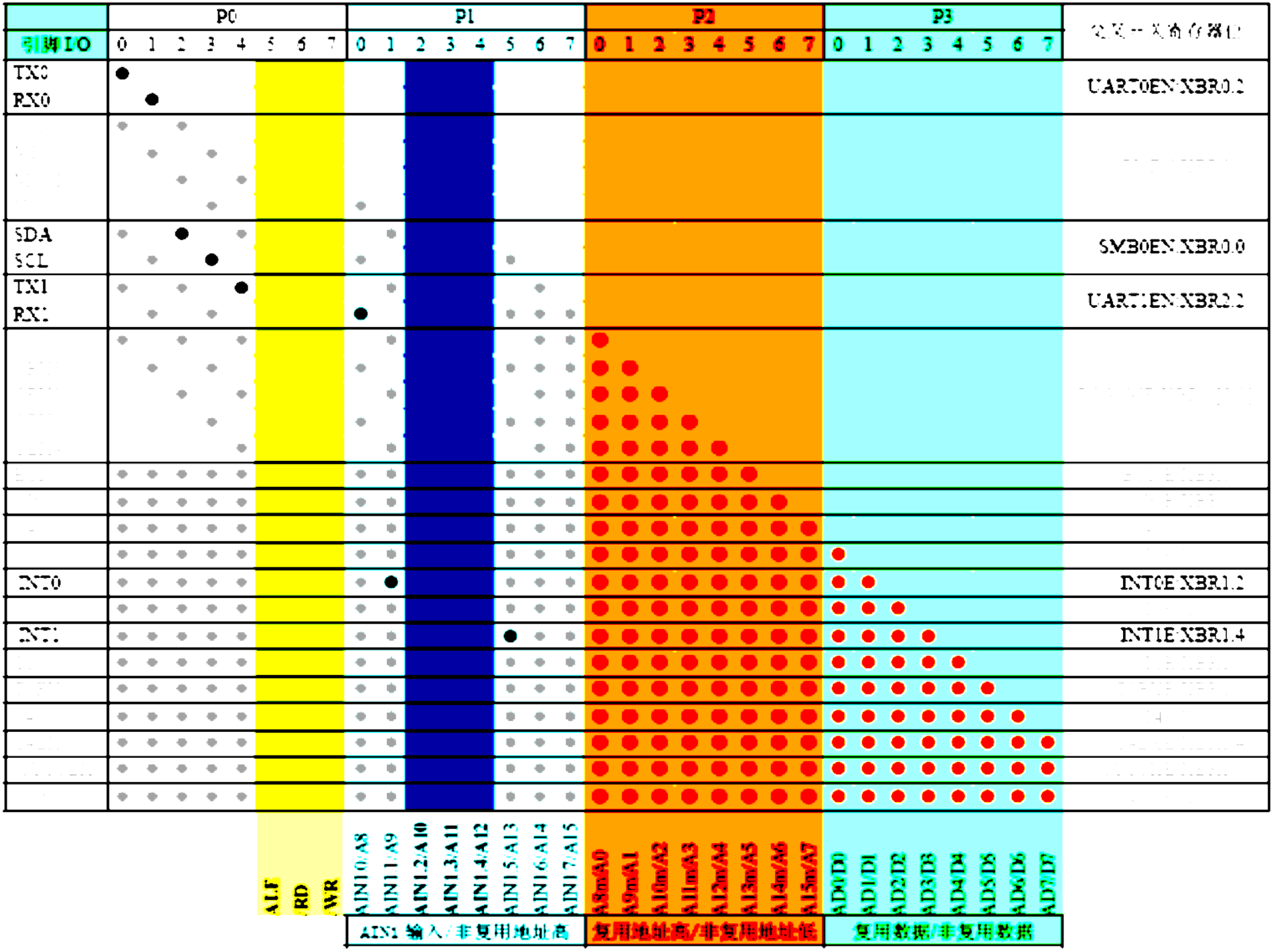
PRTSEL = 0，EMD2 = 0。

3、将作为模拟输入的端口1 引脚配置为模拟输入方式：设置P1MDIN 为0xE3（P1.4、P1.3 和P1.2 为模拟输入，所以它们的对应P1MDIN 被设置为逻辑‘0’）。

4、设置XBARE = 1 以使能交叉开关：XBR2= 0x46。

5、配置结果。

- **UART0** 有最高优先权，所以**P0.0** 被分配给**TX0**，**P0.1** 被分配给**RX0**。
- **SMBus** 的优先权次之，所以**P0.2** 被分配给**SDA**，**P0.3** 被分配给**SCL**。
- 接下来是**UART1**，所以**P0.4** 被分配给**TX1**。
- 由于外部存储器接口选在低端口（**EMIFLE = 1**），所以交叉开关跳过**P0.6(/RD)**和**P0.7(/WR)**。又因为外部存储器接口被配置为复用方式，所以交叉开关也跳过**P0.5(ALE)**。
- 下一个未被跳过的引脚**P1.0** 被分配给**RX1**。
- **/INT0**，被分配到引脚**P1.1**。
- 将**P1MDIN** 设置为**0xE3**，使**P1.2**、**P1.3** 和**P1.4** 被配置为模拟输入，导致交叉开关跳过这些引脚。
- 优先权高的是**/INT1**，所以下一个未跳过的引脚**P1.5** 被分配给**/INT1**。


P0
P1
P2
P3
引脚 IO
0 1 2 3 4 5 6 7
TX0
RX0
SDA
SCL
TX1
RX1
INT0
INT1
UART0EN XBR0.2
SMB0EN XBR0.0
UART1EN XBR1.2
INT0E XBR1.2
INT1E XBR1.4
ALE
/RD
/WR
AIN10/A8
AIN11/A9
AIN12/A10
AIN13/A11
AIN14/A12
AIN15/A13
AIN16/A14
AIN17/A15
A8m/A0
A9m/A1
A10m/A2
A11m/A3
A12m/A4
A13m/A5
A14m/A6
A15m/A7
AD0/D0
AD1/D1
AD2/D2
AD3/D3
AD4/D4
AD5/D5
AD6/D6
AD7/D7
AIN1 输入/非复用地址高
复用地址高/非复用地址低
复用数据/非复用数据

6、 我们将UART0 的TX 引脚（TX0，P0.0）、UART1 的TX 引脚（TX1，P0.4）、ALE、/RD、/WR（P0.[7:3]）的输出设置为推挽方式，通过设置P0MDOUT = 0xF1 来实现。

7、 我们通过设置P2MDOUT = 0xFF 和P3MDOUT = 0xFF 将EMIF 端口（P2、P3）的输出方式配置为推挽方式。

8、 我们通过设置P1MDOUT = 0x00（配置输出为漏极开路）和P1 = 0xFF（逻辑‘1’选择高阻态）禁止3 个模拟输入引脚的输出驱动器。

2.3 端口4-7（仅C8051F020/2）

端口4-7 的所有端口引脚都可用作通用I/O（GPIO），通过读和写相应的端口数据寄存器访问每个端口，这些端口数据寄存器是一组按字节寻址的特殊功能寄存器。

读端口数据寄存器时，返回的是端口引脚本身的逻辑状态。

2.3.1 配置端口引脚的输出方式

每个端口引脚的输出方式都可被配置为:

漏极开路、推挽方式。

1、推挽方式:

向端口数据寄存器中的相应位:

写逻辑‘0’将使端口引脚被驱动到GND,

写逻辑‘1’将使端口引脚被驱动到VDD。

2、漏极开路方式:

向端口数据寄存器中的相应位:

写逻辑‘0’将使端口引脚被驱动到GND,

写逻辑‘1’将使端口引脚处于高阻状态。

3、端口4-7 引脚的输出方式由P74OUT 寄存器中的位决定。

P7-4OUT 中的每一位控制端口4-7 中一组引脚（每组4 位）的输出方式。

- P7-4OUT.7 为逻辑‘1’时将端口7 中高4 位（P7.[7:4]）的输出方式配置为推挽方式；
- P7-4OUT.7 为逻辑‘0’时将端口7 中高4 位（P7.[7:4]）的输出方式配置为漏极开路。

2.3.2 配置端口引脚为数字输入

通过设置输出方式为“漏极开路”并向端口数据寄存器中的相应位写 ‘1’ 将端口引脚配置为数字输入。

例如： 将P7.7 配置为数字输入

设置P4OUT.7 为逻辑‘0’

设置P7.7 为逻辑‘1’即可。

2.3.3 弱上拉

每个端口引脚都有一个内部弱上拉部件，在缺省情况下该上拉器件被使能，在引脚与VDD之间提供阻性连接（约100 kΩ）。

弱上拉部件可以被总体禁止，通过向弱上拉禁止位（WEAKPUD，XBR2.7）写‘1’实现。

当任何引脚被驱动为逻辑‘0’时，弱上拉自动取消；即输出引脚不能与其自身的上拉部件冲突。

3、系统时钟源

每个MCU 都有一个内部振荡器和一个外部振荡器驱动电路，每个驱动电路都能产生系统时钟。

MCU 在复位后从内部振荡器启动。

内部振荡器可以被使能/禁止，其振荡频率可以用内部振荡器控制寄存器（OSCICN）设置。

当/RST 引脚为低电平时，两个振荡器都被禁止。

MCU 可以从内部振荡器或外部振荡器运行，可使用OSCICN 寄存器中的CLKSL 位在两个振荡器之间随意切换。

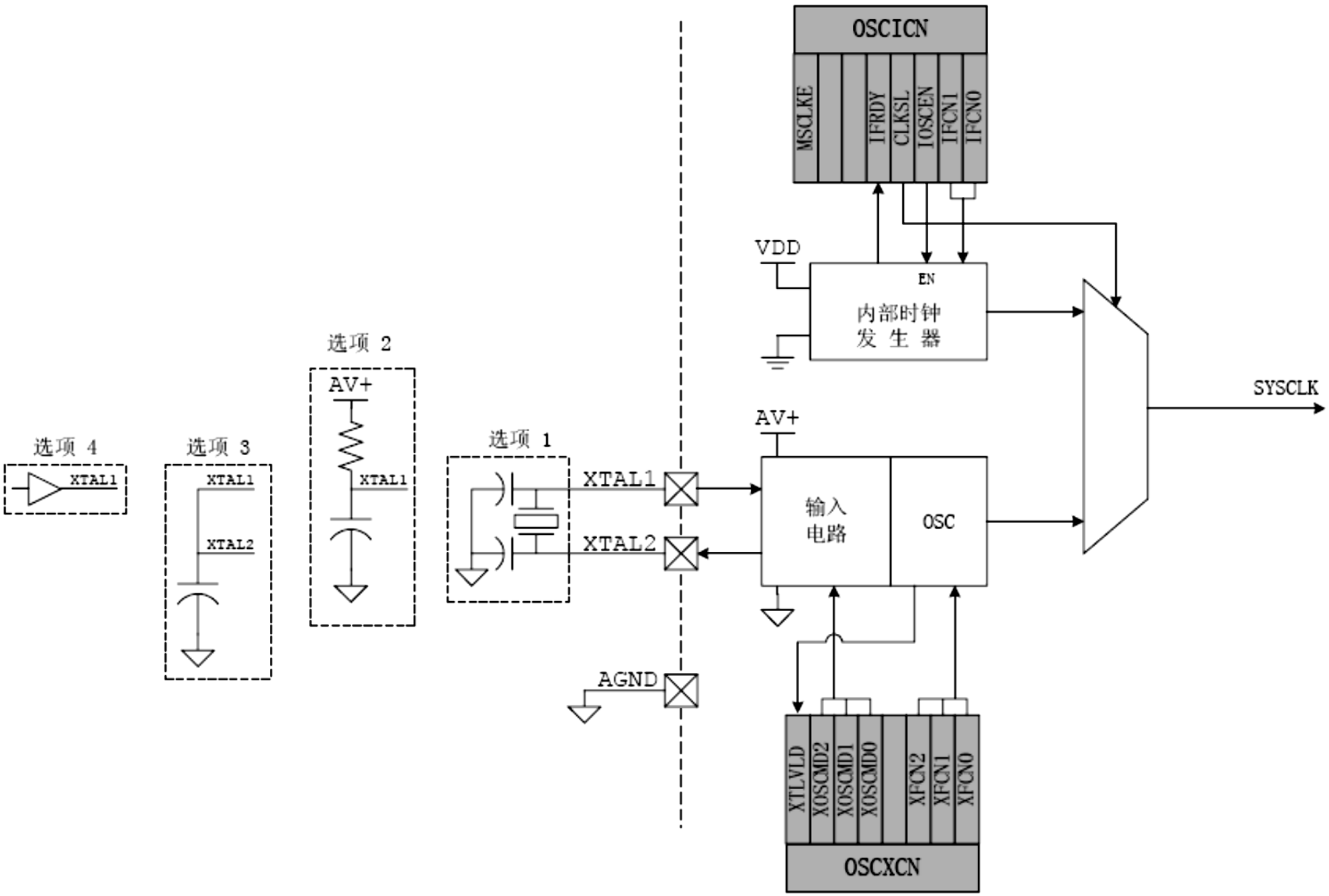

OSCICN
MSCLKE
IFRDY
CLKSL
IOSCEN
IFCN1
IFCN0
VDD
EN
内部时钟
发 生 器
SYSCLK
AV+
选项 1
选项 2
选项 3
选项 4
XTAL1
XTAL2
输入
电路
OSC
AGND
XTLVLD
XOSCMD2
XOSCMD1
XOSCMD0
XFCN2
XFCN1
XFCN0
OSCXCN

图 14.2　OSCICN：内部振荡器控制寄存器

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位值
MSCLKE	-	-	IFRDY	CLKSL	IOSCEN	IFCN1	IFCN0	00010100
位7	位6	位5	位4	位3	位2	位1	位0	SFR地址：0xB2

位 7：　MSCLKE：时钟丢失检测器使能位
0：禁止时钟丢失检测器。
1：使能时钟丢失检测器；检测到时钟丢失时间大于 100 微秒时将触发复位。

位 6-5：　未用。读=00b，写=忽略。

位 4：　IFRDY：内部振荡器频率准备好标志
0：内部振荡器频率不是按 IFCN 位指定的速度运行。
1：内部振荡器频率按照 IFCN 位指定的速度运行。

位 3：　CLKSL：系统时钟源选择位
0：选择内部振荡器作为系统时钟。
1：选择外部振荡器作为系统时钟。

位 2：　IOSCEN：内部振荡器使能位
0：内部振荡器禁止。
1：内部振荡器使能。

位 1-0：　IFCN1-0：内部振荡器频率控制位
00：内部振荡器典型频率为 2MHz。
01：内部振荡器典型频率为 4MHz。
10：内部振荡器典型频率为 8MHz。
11：内部振荡器典型频率为 16MHz。

图 14.3　OSCXCN：外部振荡器控制寄存器

R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位值
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00000000
位7	位6	位5	位4	位3	位2	位1	位0	SFR地址：0xB1

位 7：　XTLVLD：晶体振荡器有效标志
（只在 XOSCMD＝11x 时有效）
0：晶体振荡器未用或未稳定。
1：晶体振荡器正在运行并且工作稳定。

位 6-4：　XOSCMD2-0：外部振荡器方式位
00x：关闭。XTAL1 引脚内部接地。
010：系统时钟为来自 XTAL1 引脚的外部 CMOS 时钟。
011：系统时钟为来自 XTAL1 引脚的外部 CMOS 时钟的二分频。
10x：RC/C 振荡器方式二分频。
110：晶体振荡器方式。
111：晶体振荡器方式二分频。

位 3：　保留。读 = 无定义，写 = 忽略。

图 14.3　OSCXCN：外部振荡器控制寄存器

R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位值
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00000000
位7	位6	位5	位4	位3	位2	位1	位0	SFR地址：0xB1

位 2-0：　XFCN2-0：外部振荡器频率控制位。

000-111：见下表

XFCN	晶体（XOSCMD=11x）	RC(XOSCMD=10x)	C(XOSCMD=10x)
000	f ≤ 12kHz	f≤ 25kHz	K 因子= 0.44
001	12kHz < f ≤ 30kHz	25kHz < f ≤ 50kHz	K 因子= 1.4
010	30kHz < f ≤ 95kHz	50kHz < f ≤ 100kHz	K 因子= 4.4
011	95kHz < f ≤ 270kHz	100kHz < f ≤ 200kHz	K 因子= 13
100	270kHz < f ≤ 720kHz	200kHz < f ≤ 400kHz	K 因子= 38
101	720kHz < f ≤ 2.2MHz	400kHz < f ≤ 800kHz	K 因子= 100
110	2.2MHz < f≤ 6.7MHz	800kHz < f ≤ 1.6MHz	K 因子= 420
111	f > 6.7MHz	1.6MHz < f ≤ 3.2MHz	K 因子= 1400

3.1 外部晶体举例

如果使用晶体或陶瓷谐振器作为MCU 的外部振荡器源，则电路应为图14.1 中的选项1。

当外部晶体振荡器稳定运行时，晶体振荡器有效标志（OSCXCN 寄存器中的XTLVLD）被硬件置‘1’。

XTLVLD 检测电路要求在使能振荡器工作和检测XTLVLD 之间至少有1 ms的启动时间。

建议的过程为：

1. 使能外部振荡器
2. 等待至少1ms
3. 查询XTLVLD => ’1’
4. 将系统时钟切换到外部振荡器

4、系统复位

复位电路允许很容易地将控制器置于一个预定的缺省状态。在进入复位状态时，将发生以下过程：

- CIP-51 停止程序执行
- 特殊功能寄存器（SFR）被初始化为所定义的复位值
- 外部端口引脚被置于一个已知状态
- 中断和定时器被禁止。

- 所有的SFR 都被初始化为预定值，SFR 中各位的复位值在SFR 的详细说明中定义。

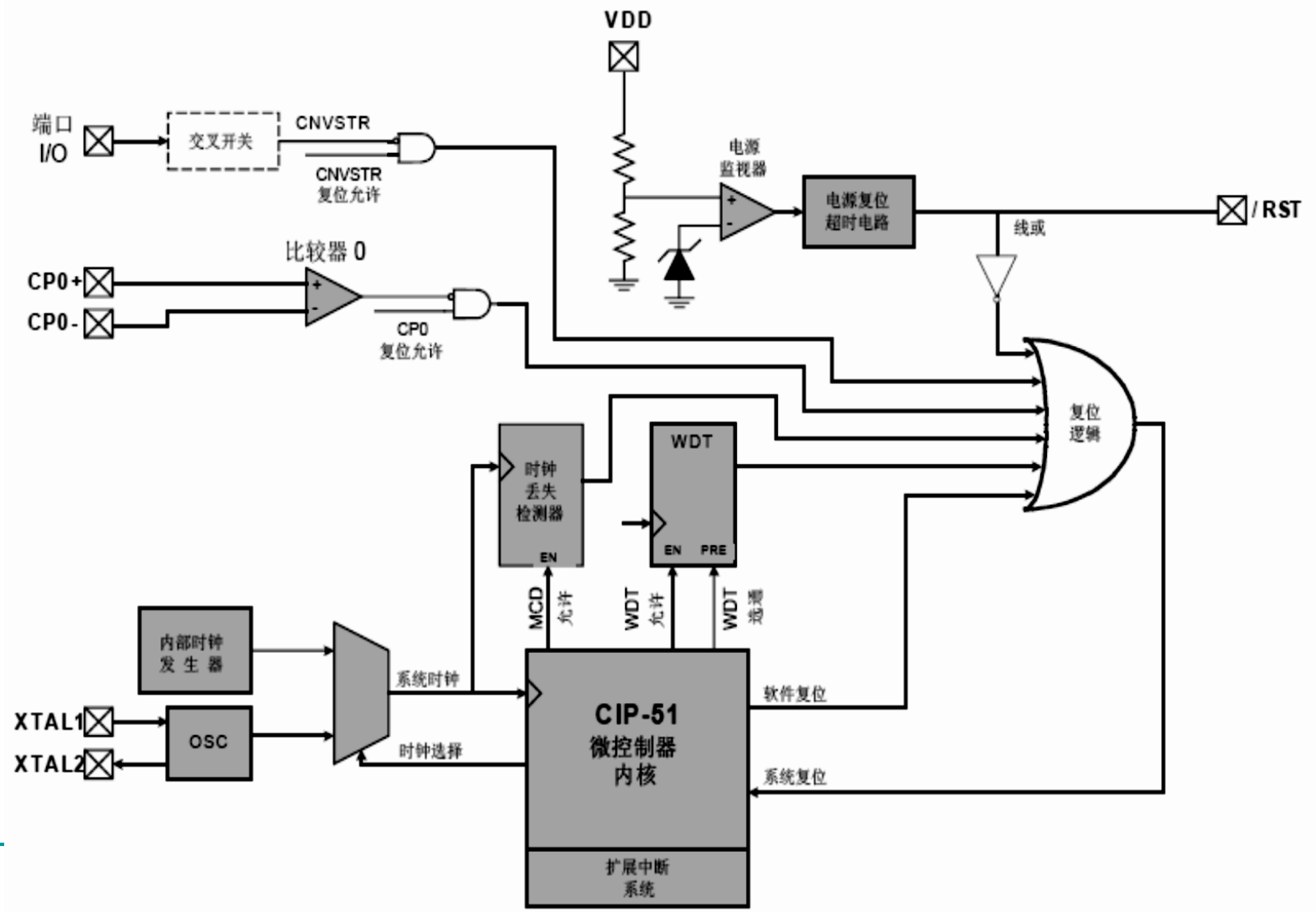
- I/O 端口锁存器的复位值为0xFF（全部为逻辑‘1’），内部弱上拉有效，使外部I/O 引脚处于高电平状态。

注1：外部I/O 引脚并不立即进入高电平状态，而是在进入复位状态后的四个系统时钟之内。
注2：在复位期间弱上拉是被禁止的，在器件退出复位状态时弱上拉被使能

- 在退出复位状态时，程序计数器（PC）被复位，MCU使用内部振荡器运行在2MHz 作为默认的系统时钟。

- 看门狗定时器被使能，使用其最长的超时时间。

有7 个能使MCU 进入复位状态的复位源：上电/掉电、外部/RST 引脚、外部CNVSTR 信号、软件命令、比较器0、时钟丢失检测器及看门狗定时器。

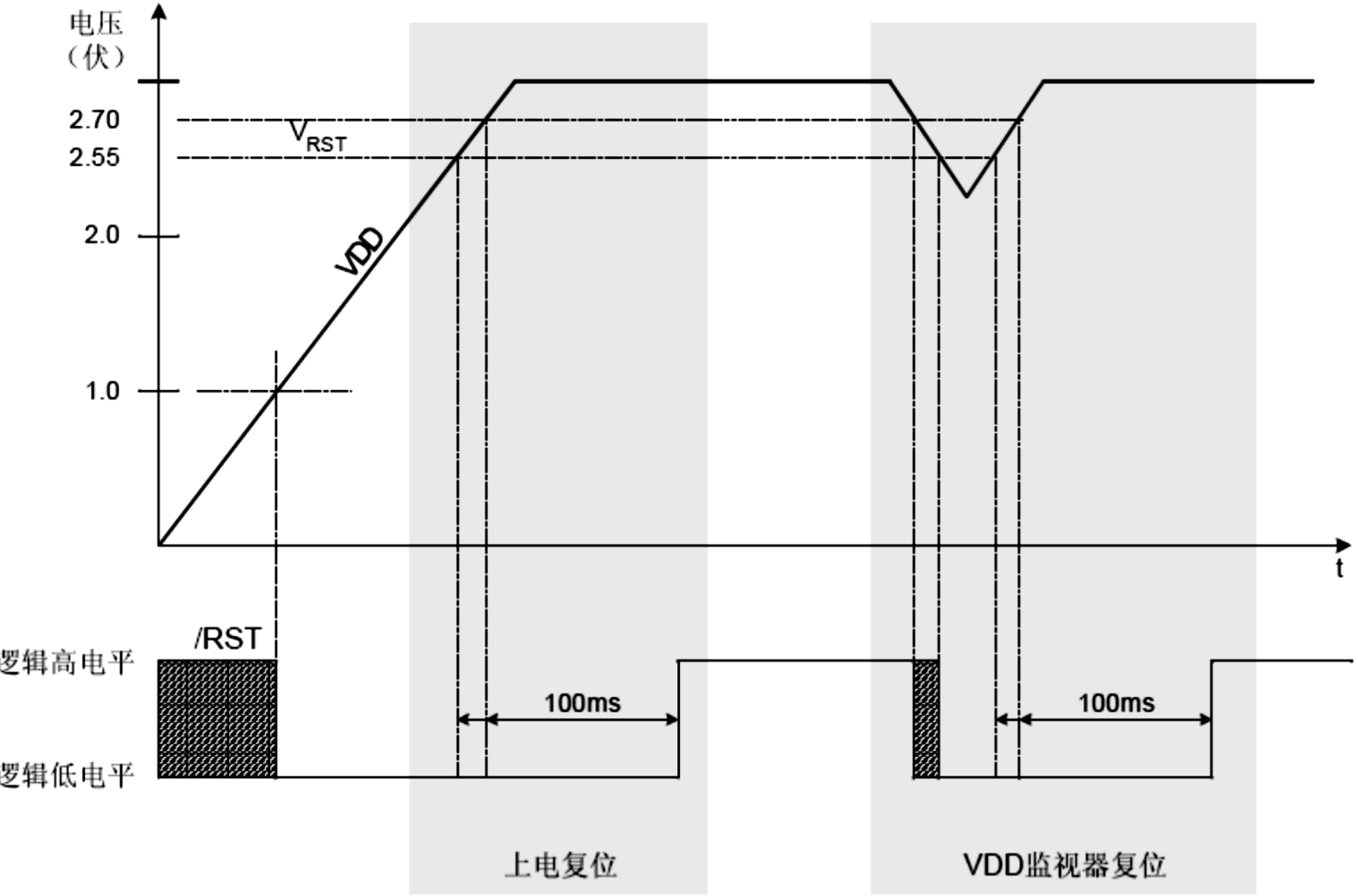


4.1 上电复位

C8051F020有一个电源监视器，在上电期间该监视器使MCU保持在复位状态，直到VDD 上升到超过VRST 电平。

/RST 引脚一直被置为低电平，直到100 毫秒的VDD 监视器超时时间结束，这100 毫秒的等待时间是为了使VDD 电源稳定。

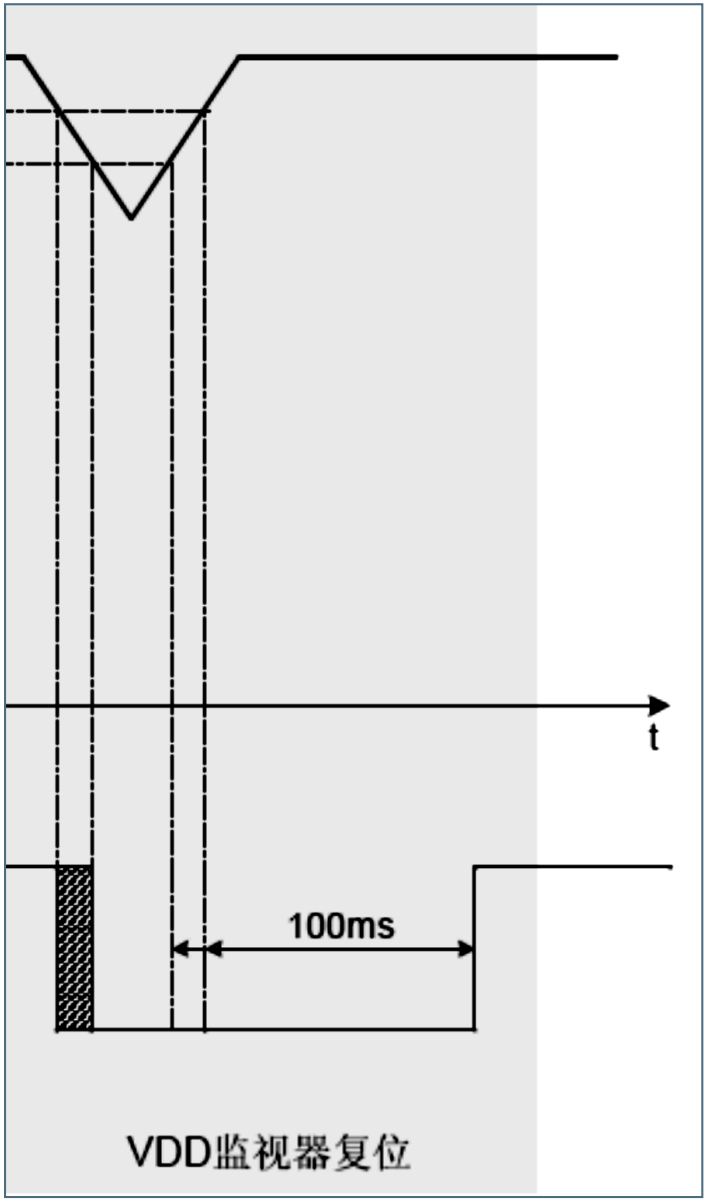
在退出上电复位状态时，PORSF 标志（RSTSRC.1）被硬件置为逻辑‘1’，RSTSRC 寄存器中的其它复位标志是不确定的。


电压
（伏）
2.70
2.55
2.0
1.0
V_{RST}
VDD
t
/RST
逻辑高电平
逻辑低电平
100ms
100ms
上电复位
VDD监视器复位

4.2 掉电复位

通过将MONEN 引脚直接连VDD 来使能VDD 监视器。
这是MONEN 引脚的推荐配置。

当发生掉电或因电源不稳定而导致VDD 下降到低于VRST 电平时，电源监视器将/RST 引脚置于低电平并使CIP-51 回到复位状态。



VDD监视器复位

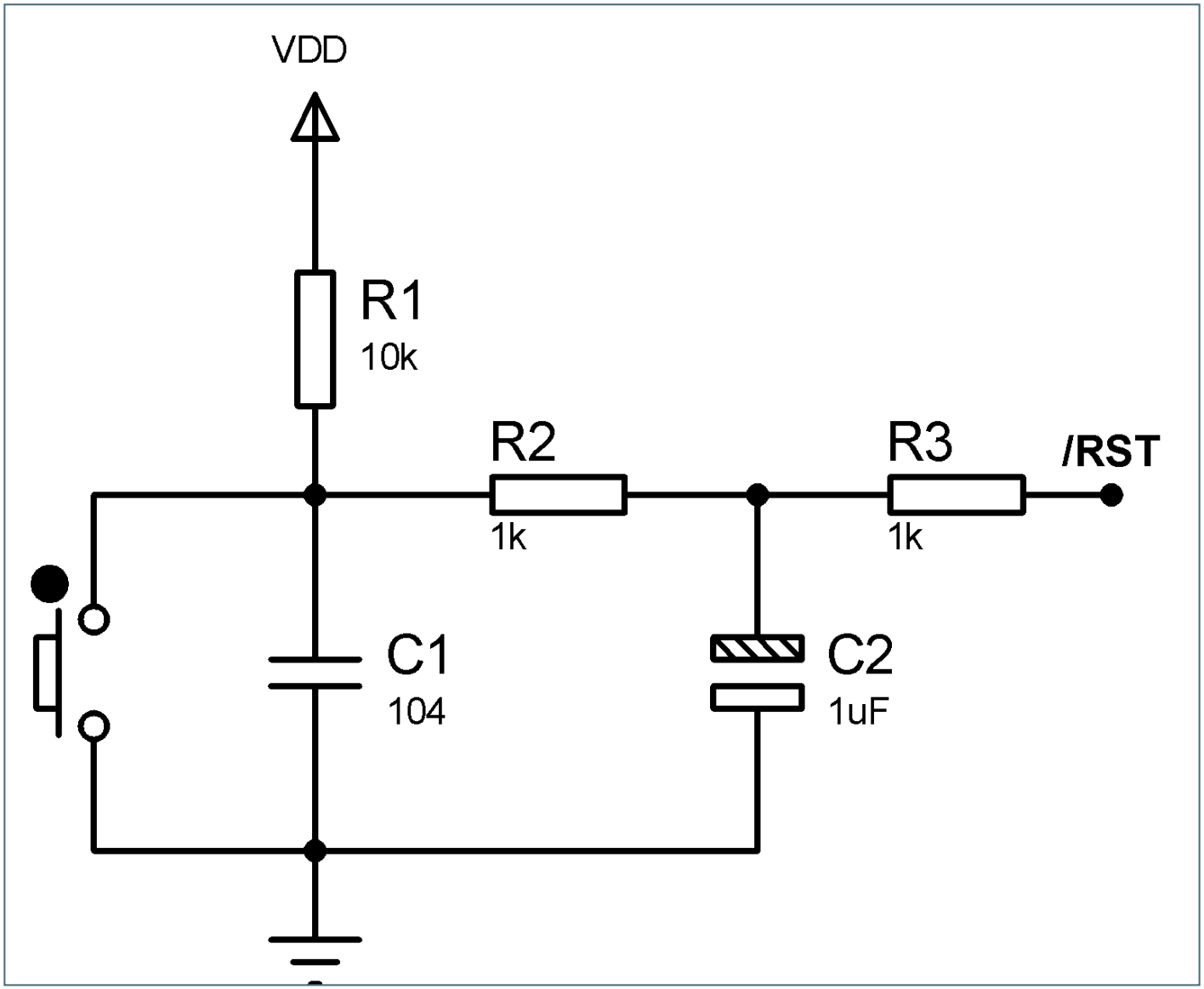
4.3 外部复位

- 外部/RST 引脚提供了使用外部电路强制MCU 进入复位状态的手段。在/RST 引脚上加一个低电平有效信号将导致MCU 进入复位状态。

- 最好能提供一个外部上拉和/或对/RST 引脚去耦以防止强噪声引起复位。

- 在低有效的/RST 信号撤出后，MCU 将保持在复位状态至少12 个时钟周期。

- 从外部复位状态退出后，PINRSF 标志（RSTSRC.0）被置位。


VDD
R1
10k
R2
1k
R3
1k
/RST
C1
104
C2
1uF

4.4 看门狗定时器复位

- **MCU 内部有一个使用系统时钟的可编程看门狗定时器（WDT）。当看门狗定时器溢出时，WDT 将强制CPU 进入复位状态。**

- **为了防止复位，必须在溢出发生前由应用软件重新触发WDT。**

- **如果系统出现了软件/硬件错误，使应用软件不能重新触发WDT，则WDT 将溢出并产生一个复位，这可以防止系统失控。**

什么是看门狗(watchdog)

看门狗,又叫 watchdog timer,是一个定时器电路, 一般有一个输入,叫喂狗,一个输出到MCU的RST端,MCU正常工作的时候,每隔一端时间输出一个信号到喂狗端,给 WDT 清零,如果超过规定的时间不喂狗,(一般在程序跑飞时),WDT 定时超过,就回给出一个复位信号到MCU,是MCU复位. 防止MCU死机. 看门狗的作用就是防止程序发生死循环，或者说程序跑飞。

工作原理：在系统运行以后也就启动了看门狗的计数器，看门狗就开始自动计数，如果到了一定的时间还不去清看门狗，那么看门狗计数器就会溢出从而引起看门狗中断，造成系统复位。所以在使用有看门狗的芯片时要注意清看门狗。

硬件看门狗是利用了一个定时器，来监控主程序的运行，也就是说在主程序的运行过程中，我们要在定时时间到之前对定时器进行复位。如果出现死循环，或者说PC指针不能回来，那么定时时间到后就会使单片机复位。常用的WDT芯片如MAX813 ,5045, IMP 813等

软件看门狗:编程方法。通过程序对定时器进行设定

在从任何一种复位退出时，**WDT** 被自动使能并使用缺省的最大时间间隔运行。

系统软件可以根据需要禁止**WDT** 或将其锁定为运行状态以防止意外产生的禁止操作。

看门狗的功能可以通过看门狗定时器控制寄存器（**WDTCN**）控制。

图 13.3 WDTCN: 看门狗定时器控制寄存器

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	复位值
								xxxxx111
位7	位6	位5	位4	位3	位2	位1	位0	SFR地址：0xFF

位 7-0： WDT 控制

写入 0xA5 将使能并重新装载 WDT。

写入 0xDE 后四个系统周期内写入 0xAD，将禁止 WDT。

写入 0xFF 将锁定禁止功能。

位 4： 看门狗状态位（读）

读 WDTCN.[4]得到看门狗定时器的状态。

0：WDT 处于不活动状态。

1：WDT 处于活动状态。

位 2-0： 看门狗超时间隔位

位 WDTCN.[2:0]设置看门狗的超时间隔。在写这些位时，WDTCN.7 必须被置为‘0’。

使能/复位WDT

向WDTCN 寄存器写入0xA5 将使能并复位看门狗定时器。

用户的应用软件应周期性地向WDTCN 写入0xA5，以防止看门狗定时器溢出。

每次系统复位都将使能并复位WDT。

禁止WDT

向WDTCN 寄存器写入0xDE 后再写入0xAD 将禁止WDT。下面的代码段说明禁止WDT的过程。

```
CLR EA ;                    禁止所有中断
MOV WDTCN, #0DEh ;          禁止软件看门狗定时器
MOV WDTCN, #0ADh
SETB EA ;                   重新允许中断
```

写0xDE 和写0xAD 必须发生在4 个时钟周期之内，否则禁止操作将被忽略。

禁止WDT 锁定

向WDTCN 写入0xFF 将使禁止功能无效。

一旦锁定，在下一次复位之前禁止操作将被忽略。
写0xFF 并不使能或复位看门狗定时器。

设置WDT 定时间隔

WDTCN.[2:0]控制看门狗超时间隔。超时间隔由下式给出：

$4^{3+WDTCN[2:0]} \times T_{SYSCLK}$；（其中 T_{SYSCLK} 为系统时钟周期）

5、JTAG接口的在系统调试

每个MCU 内部都有JTAG 和调试电路，可以通过JTAG 接口使用安装在最终应用系统上的产品MCU 进行非侵入式、全速、在系统调试。

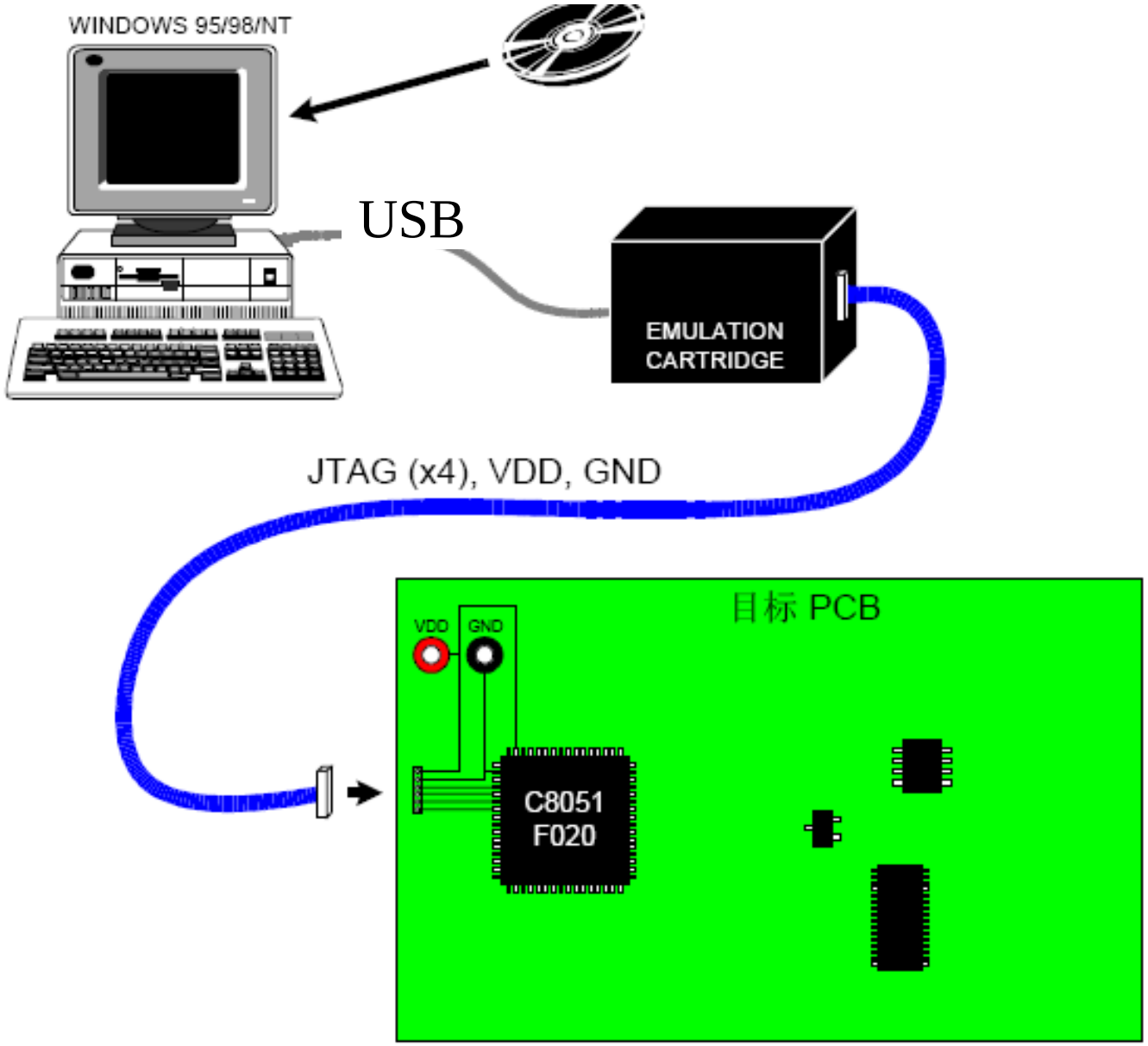
Silicon Labs 的调试系统支持观察和修改存储器和寄存器、断点和单步执行；不需要额外的目标RAM、程序存储器或通信通道。

在调试时，所有的模拟和数字外设都全功能正确运行（保持同步）。

当MCU 因单步执行或执行到断点而停机时，WDT 被禁止。

JTAG 接口使用MCU 上的四个专用引脚，它们是：

TCK、TMS、TDI 和TDO。


WINDOWS 95/98/NT
USB
EMULATION
CARTRIDGE
JTAG (x4), VDD, GND
目标 PCB
VDD
GND
C8051
F020

6、单片机的初始化设置

单片机初始化，包括以下几点：

- 看门狗初始化—开启还是禁止、如果开启则喂狗周期为多少；
- 时钟系统的初始化—确定系统的工作时钟源及频率；
- I/O引脚输入输出方式初始化—输入：模拟还是数字、输出：推挽还是开漏；
- 数字外设的配置和交叉开关设置；