LPC2468 器件

用户手册

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4

网址：http://www.zlgmcu.com

技术支持

如果您对文档有所疑问，您可以在办公时间（星期一至星期五上午 8:30~11:50；下午 1:30~5:30；星期六上午 8:30~11:50）拨打技术支持电话或 E-mail 联系。

网　　址：www .zlgmcu.com

联系电话：　+86 (020)　22644358　22644359　2 2644360　22644361

E-mail：　zlgmcu.support@zlgmcu.com

销售与服务网络

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4 邮编：510630

电话：(020)38730972 38730976 38730916 38730917 38730977

传真：(020)38730925

网址：http://www.zlgmcu.com

广州专卖店

地址：广州市天河区新赛格电子城 203-204 室

电话：(020)87578634 87569917

传真：(020)87578842

南京周立功

地址：南京市珠江路 280 号珠江大厦 2006 室

电话：(025)83613221 83613271 83603500

传真：(025)83613271

北京周立功

地址：北京市海淀区知春路 113 号银网中心 A 座 1207-1208 室（中发电子市场斜对面）

电话：(010)62536178 62536179 82628073

传真：(010)82614433

重庆周立功

地址：重庆市石桥铺科园一路二号大西洋国际大厦（赛格电子市场）1611 室

电话：(023)68796438 68796439

传真：(023)68796439

杭州周立功

地址：杭州市天目山路 217 号江南电子大厦 502 室

电话：(0571)28139611 28139612 28139613 28139615 28139616 28139618

传真：(0571)28139621

成都周立功

地址：成都市一环路南二段 1 号数码同人港 401 室（磨子桥立交西北角）

电话：(028) 85439836 85437446

传真：(028) 85437896

深圳周立功

地址：深圳市深南中路 2070 号电子科技大厦 C 座 4 楼 D 室

电话：(0755)83781788（5 线）

传真：(0755)83793285

武汉周立功

地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室（华中电脑数码市场）

电话：(027)87168497 87168297 87168397

传真：(027)87163755

上海周立功

地址：上海市北京东路 668 号科技京城东座 7E 室

电话：(021)53083452 53083453 53083496

传真：(021)53083491

西安办事处

地址：西安市长安北路 54 号太平洋大厦 1201 室

电话：(029)87881296 83063000 87881295

传真：(029)87880865

目录

第16章 通用异步收发器（UART）0/2/3

16.1 特性

- 16 字节接收和发送 FIFO；
- 寄存器位置符合’550 工业标准；
- 接收器 FIFO 触发深度可为 1, 4, 8 和 14 字节；
- 内置波特率发生器；
- 用于控制波特率的小数分频器，并拥有赖以实现软件流控制的自动波特率（autobaud）检测能力和机制；
- 另外，UART3 还包含一种支持红外通信的 IrDA 模式。

16.2 管脚描述

表 16.1 UART0 管脚描述

管脚名称	类型	描述
RXD0，RXD2，RXD3	输入	**串行输入**。串行接收数据。
TXD0，TXD2，TXD3	输出	**串行输出**。串行发送数据。

16.3 寄存器描述

每个 UART 均包含下表列出的寄存器。除数锁存访问位（DLAB）位于 UnLCR7 寄存器，用于使能对除数锁存的访问。

表 16.2 UART 寄存器映射

通用名称	描述	位功能和地址								访问	复位值[1]
		MSB							LSB		
		位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0		
RBR (DLAB=0)	接收器缓冲	8 位读数据								RO	NA
THR (DLAB=0)	发送器保持	8 位写数据								WO	NA
DLL (DLAB=1)	除数锁存 LSB	8 位数据								R/W	0x01
DLM (DLAB=1)	除数锁存 MSB	8 位数据								R/W	0x00
IER (DLAB=0)	中断使能	保留						使能自动波特率超时中断	使能自动波特率中断结束	R/W	0x00
		0					使能RX线状态中断	使能 THRE 中断	使能 RX 数据可用中断		

续表 16.2

通用名称	描述	位功能和地址								访问	复位值[1]
		MSB							LSB		
		位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0		
IIR	中断 ID	保留						ABTO 中断	ABEO 中断	RO	0x01
		FIFO 使能		0		IIR3	IIR2	IIR1	IIR0		
FCR	FIFO 控制	Rx 触发		保留			Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0x00
LCR	线控制	DLAB	设置间隔	奇偶固定	偶选择	奇偶使能	停止位个数	字长度选择		R/W	0x00
LSR	线状态	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
SCR	高速缓存	8 位数据								R/W	0x00
ACR	自动波特率控制	保留[31:10]						ABTO IntCIr	ABEO IntCIr	R/W	0x00
		保留[7:3]					自动复位	模式	启动		
ICR	IrDA 控制	保留			PulseDiv		FixPulse En	IrDAInv	IrDAEn	R/W	0
FDR	小数分频器	MuIVal				DivAddVaI				R/W	0x10
TER	发送使能	TXEN	保留							R/W	0x80

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

16.3.1 UARTn接收器缓存寄存器（U0RBR - 0xE000 C000，U2RBR - 0xE007 8000，U3RBR - 0xE007 C000; DLAB＝0，只读）

UnRBR 是 UARTn Rx FIFO 的最高字节。它包含了最早接收到的字符，我们可以通过总线接口读取它。LSB(bit0)表示最早接收到的数据位。如果接收到的字符小于 8 位，未使用的 MSB 用 0 填充。

如果要访问 UnRBR，LCR 中的除数锁存访问位（DLAB）必须为 0。UnRBR 为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶部的字节（即下次读取 RBR 时将读取的字节）相关，因此，要正确地成对读出有效的接收字节及其状态位，应该先读取 U0LSR 寄存器的内容，然后再读取 UnRBR 中的字节。

表 16.3 UARTn 接收器缓存寄存器（U0RBR - 0xE000 C000，U2RBR - 0xE007 8000，U3RBR - 0xE007 C000;DLAB＝0，只读）位描述

位	符号	描述	复位值
7:0	RBR	UARTn 接收器缓存寄存器包含 UARTn Rx FIFO 中最早接收到的字节。	未定义

16.3.2 UARTn发送保持寄存器（U0THR - 0xE000 C000，U2THR - 0xE007 8000，U3THR - 0xE007 C000；DLAB=0，只写）

UnTHR 是 UARTn TX FIFO 的最高字节。它包含了 TX FIFO 中最新的字符，我们可以通过总线接口对它进行写操作。LSB 代表最先发送的位。

如果要访问 UnTHR，UnLCR 的除数锁存访问位（DLAB）必须为 0。UnTHR 为只写寄存器。

表 16.4 UART0 发送保持寄存器（U0THR – 地址 0xE000 C000，U2THR - 0xE007 8000，U3THR - 0xE007 C000；DLAB=0，只写）位描述

位	符号	描述	复位值
7:0	THR	写 UARTn 发送保持寄存器会使数据保存到 UARTn 发送 FIFO 中。当字节到达 FIFO 的最底部并且发送器就绪时，该字节将被发送。	N/A

16.3.3 UARTn除数锁存LSB寄存器（U0DLL - 0xE000 C000，U2DLL - 0xE007 8000，U3DLL - 0xE007 C000；DLAB=1）和UARTn除数锁存MSB寄存器（U0DLM - 0xE000 C004，U2DLM - 0xE007 8004，U3DLM - 0xE007 C004；DLAB=1）

UARTn 除数锁存是波特率发生器的一部分，它保存了用于产生波特率时钟的 APB 时钟（PCLK）的分频值，波特率时钟必须是所需波特率的 16 倍。UnDLL 和 UnDLM 寄存器一起构成一个 16 位除数。其中，UnDLL 包含的是除数的低 8 位，UnDLM 包含的是除数的高 8 位。0x0000 被看作是 0x0001，因为除数是不允许为 0 的。在访问 UARTn 除数锁存寄存器时，除数锁存访问位（DLAB）必须为 1。

表 16.5 UARTn 除数锁存 LSB 寄存器（U0DLL – 地址 0xE000 C000，U2DLL - 0xE007 8000，U3DLL - 0xE007 C000；DLAB=1）位描述

位	符号	描述	复位值
7:0	DLLSB	UARTn 除数锁存 LSB 寄存器与 UnDLM 寄存器共同决定 UARTn 的波特率。	0x01

表 16.6 UARTn 除数锁存 MSB 寄存器（U0DLM – 地址 0xE000 C004，U2DLM - 0xE007 8004，U3DLM - 0xE007 C004；DLAB=1）位描述

位	符号	描述	复位值
7:0	DLMSB	UARTn 除数锁存 MSB 寄存器与 U0DLL 寄存器共同决定 UARTn 的波特率。	0x00

16.3.4 UARTn中断使能寄存器（U0IER - 0xE000 C004，U2IER - 0xE007 8004，U3IER - 0xE007 C004；DLAB=0）

UnIER 用于使能 3 个 UARTn 中断源。

表 16.7 UARTn 中断使能寄存器（U0IER – 地址 0xE000 C004，U2IER - 0xE007 8004，U3IER - 0xE007 C004；DLAB=0）位描述

位	符号	值	描述	复位值
0	RBR 中断使能	UnIER[0]使能 UARTn 的接收数据可用中断。它还控制字符接收超时中断。		0
		0	禁止 RDA 中断。	
		1	使能 RDA 中断。	
1	THRE 中断使能	0	UnIER[1]使能 UARTn 的 THRE 中断。THRE 的状态可从 UnLSR[5]读出。 禁止 THRE 中断。	0
		1	使能 THRE 中断。	
2	RX 线状态中断使能	0	UnIER[2]使能 UARTn RX 线状态中断。该中断的状态可从 UnLSR[4:1]读出。 禁止 RX 线状态中断。	0
		1	使能 RX 线状态中断。	
7:3	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA
8	ABTOIntEn	U1IER[8]使能自动波特率超时中断。		0
		0	禁止自动波特率超时中断。	
		1	使能自动波特率超时中断。	
9	ABEOIntEn	U1IER[9]使能结束自动波特率中断。		0
		0	禁止自动波特率中断结束。	
		1	使能自动波特率中断结束。	
31:10	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA

16.3.5 UARTn中断标识寄存器（U0IIR - 0xE000 C008，U2IIR - 0xE007 8008，U3IIR - 0xE007 C008；只读）

UnIIR 提供状态代码用于指示挂起中断的中断源和优先级。在访问 UnIIR 过程中，中断被冻结。若访问 UnIIR 时产生了中断，则该中断将被记录，下次访问 UnIIR 时便可将其读出。

表 16.8 UARTn 中断标识寄存器（U0IIR – 地址 0xE000 C008，U2IIR - 0xE007 8008，U3IIR - 0xE007 C008；只读）位描述

位	符号	值	描述	复位值
0	IntStatus	中断状态。注：UnIIR[0]低电平有效。挂起的中断可通过 UnIIR[3:1]确定。		1
		0	至少有 1 个中断被挂起	
		1	没有挂起的中断	
3:1	IntId	中断标识。UnIER[3:1]指示与 UARTn Rx FIFO 对应的中断。上文未列出的 UnIER[3:1]的其它组合都为保留值（000, 100, 101, 111）。		0
		011	1 - 接收线状态（RLS）。	
		010	2a - 接收数据可用（RDA）。	
		110	2b - 字符超时指示（CTI）。	
		001	3 – THRE 中断。	
5:4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA

续表 16.8

位	符号	值	描述	复位值
7:6	FIFO 使能		这些位等效于 UnFCR[0]。	0
8	ABEOInt		结束自动波特率中断。如果成功完成自动波特率检测且中断使能，则 ABEOInt 为 1。	0
9	ABTOInt		自动波特率超时中断。如果自动波特率发生了超时且中断使能，则 ABTOInt 为 1。	0
31:10	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

自动波特率函数设定了 UnIIR[9:8]的状态，而且还会通知超时或自动波特率结束等情况。自动波特率中断条件可以通过置位自动波特率控制寄存器中相应的位来清除。

当IntStatus位为 1 时，表示没有挂起中断，且IntId位将为 0。当IntStatus为 0 时，表示挂起的是非自动波特率中断。在这种情况下，IntId位会确定中断的类型并按照表 16.9中所述的方式进行处理。只要知道UnIIR[3:0]的状态，中断处理程序就可以确定中断的原因以及如何清除激活的中断。用户如果要在退出中断服务程序前将中断清除，就必须读取UnIIR。

UARTn RLS 中断（UnIIR[3:1]=011）是优先级最高的中断。只要 UARTn Rx 输入产生某个错误条件（溢出错误（OE）、奇偶错误（PE）、帧错误（FE）或间隔中断（BI）），则该中断标志就会置位。产生该中断的 UARTn Rx 错误条件可通过查看 U0LSR[4:1]得到。在读取 UnLSR 时中断被清除。

UARTn RDA 中断（UnIIR[3:1]=010）与 CTI 中断（UnIIR[3:1]=110）共用第二优先级。当 UARTn Rx FIFO 到达 UnFCR[7:6]所定义的触发点时，RDA 就会被激活。当 UARTn Rx FIFO 的深度低于触发点时，RDA 复位。当 RDA 中断激活时，CPU 可读出由触发点所定义的数据块。

CTI 中断（UnIIR[3:1]=110）为第二优先级中断。当 UARTn Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符这段时间内没有发生 UARTn Rx FIFO 动作时，将产生该中断。UARTn Rx FIFO 的任何动作（读或写 UARTn RSR）都可以将该中断清除。如果接收到的信息不是触发深度的倍数，那么 CTI 中断将会清空 UARTn RBR。例如，如果某个外设要发送一个 105 个字符的信息，而触发深度为 10 个字符，那么在发送前 100 个字符时，CPU 会接收 10 个 RDA 中断，但在发送剩下的 5 个字符时，CPU 将会接收 1 到 5 个 CTI 中断（视服务程序而定）。

表 16.9 UARTn 中断处理

UnIIR[3:0] [1]	优先级	中断类型	中断源	中断复位
0001	—	无	无	—
0110	最高	Rx 线状态/错误	OE[2], PE[2], FE[2]或BI[2]	UnLSR读操作[2]
0100	第二	Rx 数据可用	Rx 数据可用或到达 FIFO（UnFCR0=1）触发深度	UnRBR 读[3] 或 UARTn FIFO低于触发深度

续表 16.9

UnIIR[3:0] [1]	优先级	中断类型	中断源	中断复位
1100	第二	字符超时指示	Rx FIFO 至少包含 1 个字符，在某段时间无字符输入或移出，这段时间的长短取决于 FIFO 中的字符数以及在（3.5 到 4.5 字符的时间内）设置的触发深度。准确的时间为：[(字长度)×7-2]×8+[(触发深度－字符数)×8+1]个 RCLK	UnRBR读[3]
0010	第三	THRE	THRE[2]	UnIIR读（如果是中断源）或THR写操作[4]

[1] “0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。

[2] 详见16.3.8小节 “UARTn线状态寄存器（U0LSR - 0xE000 C014，U2LSR - 0xE007 8014，U3LSR - 0xE007 C014；只读）”。

[3] 详见16.3.1小节 “UARTn接收器缓存寄存器（U0RBR - 0xE000 C000，U2RBR - 0xE007 8000，U3RBR - 0xE007 C000; DLAB＝0，只读）”。

[4] 详见16.3.5小节 “UARTn中断标识寄存器（U0IIR - 0xE000 C008，U2IIR - 0xE007 8008，U3IIR - 0xE007 C008；只读）”和16.3.2 节“UARTn发送保持寄存器（U0THR - 0xE000 C000，U2THR - 0xE007 8000，U3THR - 0xE007 C000；DLAB=0，只写）”。

UARTn THRE 中断(UnIIR[3:1]=001)为第三优先级中断。如果满足某一个初始化条件，那么当 UARTn THR FIFO 为空时，该中断将激活。这些初始化条件将使 UARTn THR FIFO 被数据填充，以免在系统启动时产生某些 THRE 中断。初始化条件在 THRE＝1，以及自上次 THRE＝1 事件发生后 UnTHR 中包含的字符少于 2 个时，它会延时一段时间（一个字符延时减去停止位）。正是有了这段延时，CPU 才有时间将数据写到 UnTHR 中，而无需对 THRE 中断进行译码和服务。如果 UARTn THR FIFO 在某个时间或当前包含的字符超过两个，那么将立即产生 THRE 中断。当发生 UnTHR 写操作或 UnIIR 读操作并且 THRE 中断优先级最高（UnIIR[3:1]=001）时，THRE 中断复位。

16.3.6 UARTn FIFO控制寄存器（U0FCR - 0xE000 C008，U2FCR - 0xE007 8008，U3FCR - 0xE007 C008；只写）

UnFCR 控制 UARTn Rx 和 TX FIFO 的操作。

表 16.10 UARTn FIFO 控制寄存器（U0FCR – 地址 0xE000 C008，U2FCR - 0xE007 8008，U3FCR - 0xE007 C008；只写）位描述

位	符号	值	描述	复位值
0	FIFO 使能	0	禁能 UARTn FIFO。不能在应用中使用。	0
		1	高电平使能对 UARTn Rx 和 Tx FIFO 以及 UnFCR[7:1]的访问。该位必须置位以实现正确的 UARTn 操作。该位的任何变化都将使 UARTn FIFO 清空。	
1	Rx FIFO 复位	0	对任意的 UARTn FIFO 无影响。	0
		1	写 1 到 UnFCR[1]将清零 UARTn Rx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	

续表 16.10

位	符号	值	描述	复位值
2	Tx FIFO 复位	0	对任意的 UARTn FIFO 无影响。	0
		1	写 1 到 UnFCR[2]将清零 UARTn Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	
5:3	-	0	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7:6	Rx 触发选择	这两个位决定在激活中断之前，接收 UARTn FIFO 必须写入多少个字符。		0
		00	触发点 0（1 个字符或 0x01）	
		01	触发点 1（4 个字符或 0x04）	
		10	触发点 2（8 个字符或 0x08）	
		11	触发点 3（14 个字符或 0x0E）	

16.3.7 UARTn线控制寄存器（U0LCR - 0xE000 C00C，U2LCR - 0xE007 800C，U3LCR - 0xE007 C00C）

UnLCR 决定了发送或接收的数据字符的格式。

表 16.11 UARTn 线控制寄存器（U0LCR – 地址 0xE000 C00C，U2LCR - 0xE007 800C，U3LCR - 0xE007 C00C）位描述

位	符号	值	描述	复位值
1:0	字长度选择	00	5 位字符长度。	0
		01	6 位字符长度。	
		10	7 位字符长度。	
		11	8 位字符长度。	
2	停止位选择	0	1 个停止位。	0
		1	2 个停止位（如果 UnTHR[1:0]=00 则为 1.5）。	
3	奇偶使能	0	禁止奇偶产生和校验。	0
		1	使能奇偶产生和校验。	
5:4	奇偶选择	00	奇数。在已发送的字符中 1 的数目和附加的奇偶校验位将为奇数。	0
		01	偶数。在已发送的字符中 1 的数目和附加的奇偶校验位将为偶数。	
		10	强制为 1。	
		11	强制为 0。	
6	间隔控制	0	禁止间隔发送。	0
		1	使能间隔发送；当 UnTHR[6]=1 时，输出管脚 UART0 TxD 强制为逻辑 0。	
7	除数锁存访问位（DLAB）	0	禁止访问除数锁存。	0
		1	使能访问除数锁存。	

16.3.8 UARTn线状态寄存器（U0LSR - 0xE000 C014，U2LSR - 0xE007 8014，U3LSR - 0xE007 C014；只读）

UnLSR 为只读寄存器，它提供 UARTn Tx 和 Rx 模块的状态信息。

表 16.12 UARTn 线状态寄存器（U0LSR – 地址 0xE000 C014，U2LSR - 0xE007 8014，U3LSR - 0xE007 C014；只读）位描述

位	符号	值	描述	复位值
0	接收数据就绪（RDR）	当 UnRBR 包含未读取的字符时，UnLSR0 置位；当 UARTn RBR FIFO 为空时，UnLSR0 清零。		0
		0	UnRBR 为空。	
		1	UnRBR 包含有效数据。	
1	溢出错误（OE）	OE 在错误发生后会立即置位。UnLSR 读操作会清零 UnLSR1。当 UARTn RSR 包含新字符而 UARTn RBR FIFO 已满时，UnLSR1 会置位。此时 UARTn RBR FIFO 不会被覆盖，UARTn RSR 中的字符将丢失。		0
		0	溢出错误状态未激活。	
		1	溢出错误状态激活。	
2	奇偶错误（PE）	在接收字符的奇偶位指示为错误状态时将产生一个奇偶错误。UnLSR 读操作会清零 UnLSR[2]位。奇偶错误检测时间取决于 UnFCR[0]的值。 **注：**奇偶错误与 UARTn RBR FIFO 中的顶部字符相关。		0
		0	奇偶错误状态未激活。	
		1	奇偶错误状态激活。	
3	帧错误（FE）	在接收字符的停止位为 0 时将产生帧错误。UnLSR 读操作会清零 UnLSR[3]位。帧错误检测时间取决于 UnFCR[0]的值。一旦检测到帧错误，Rx 就会尝试与数据重新同步并假设错误的停止位确实是早先的一个起始位。但即使没有出现帧错误，也不能假定下一个接收到的字节是正确的。 **注：**帧错误与 UARTn RBR FIFO 中的顶部字符相关。		0
		0	帧错误状态未激活。	
		1	帧错误状态激活。	
4	间隔中断（BI）	在发送整个字符（起始位、数据位、奇偶位和停止位）过程中 RXDn 如果都保持逻辑 0，则产生间隔中断。一旦检测到中断条件，接收器就会立即进入空闲状态直至 RXDn 变为全 1 状态。UnLSR 读操作会清零该状态位。间隔检测的时间取决于 UnFCR[0]的值。 **注：**间隔中断与 UARTn RBR FIFO 中的顶部字符相关。		0
		0	间隔中断状态未激活。	
		1	间隔中断状态激活。	
5	发送保持寄存器空（THRE）	一检测到 UARTn THR 为空，THRE 就置位。UnTHR 写操作会清零该位。		1
		0	UnTHR 包含有效数据。	
		1	UnTHR 为空。	
6	发送器空（TEMT）	当 UnTHR 和 UnTSR 均为空时，TEMT 置位。当 UnTSR 或 UnTHR 包含有效数据时，TEMT 清零。		1
		0	UnTHR 和/或 UnTSR 包含有效数据。	
		1	UnTHR 和 UnTSR 为空。	
7	RX FIFO 错误（RXFE）	当一个带有 Rx 错误（例如帧错误、奇偶错误或间隔中断）的字符装入 UnRBR 时，该位置位。当读取 UnLSR 寄存器并且 UARTn FIFO 中不再有错误发生时，该位清零。		0
		0	UnRBR 中没有 UARTn RX 错误，或 UnFCR[0]=0	
		1	UARTn RBR 包含至少一个 UARTn RX 错误	

16.3.9 UARTn高速缓存寄存器（U0SCR – 0xE000 C01C，U2SCR – 0xE007 801C，U3SCR – 0xE007 C01C）

在 UARTn 操作时 UnSCR 无效。用户可自由地读写该寄存器。中断接口无需向主机指示 UnSCR 是否发生了读写操作。

表 16.13 UARTn 高速缓存寄存器（U0SCR – 地址 0xE000 C01C，U2SCR – 0xE007 801C，U3SCR – 0xE007 C01C）位描述

位	符号	描述	复位值
7:0	Pad	一个可读可写的字节。	0x00

16.3.10 UARTn自动波特率控制寄存器（U0ACR – 0xE000 C020，U2ACR – 0xE007 8020，U3ACR – 0xE007 C020）

在用户测量生成波特率的输入时钟/数据速率期间，整个测量过程就是由 UARTn 自动波特率控制寄存器(UnACR)进行控制的。用户可自由地读写该寄存器。

表 16.14 UARTn 自动波特率控制寄存器（U0ACR – 0xE000 C020，U2ACR – 0xE007 8020，U3ACR – 0xE007 C020）位描述

位	符号	值	描述	复位值
0	Start	自动波特率结束后该位自动清零。		0
		0	自动波特率停止（自动波特率不运行）。	
		1	自动波特率启动（自动波特率正在运行）。自动波特率运行位。该位在自动波特率结束后自动清零。	
1	Mode	自动波特率模式选择位。		0
		0	模式 0	
		1	模式 1	
2	AutoRestart	0	不重新启动	0
		1	如果超时则重新启动（计数器在下一个 UART0 Rx 下降沿时重新启动）。	
7:3	—	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0
8	ABEOIntClr		自动波特率中断结束清零位（仅可写访问）。写 1 会将 UnIIR 中相应的中断清除。写 0 无影响。	0
9	ABTOIntClr		自动波特率超时中断清零位（仅可写访问）。写 1 会将 UnIIR 中相应的中断清除。写 0 无影响。	0
31:10	—	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0

16.3.10.1 自动波特率（Auto-baud）

UARTn 自动波特率功能可用于测量基于“AT”协议(Hayes 命令)的输入波特率。如果使能，那么自动波特率功能部件将测量接收数据流的位所消耗的时间，并因此设置除数锁存寄存器 UnDLM 和 UnDLL。

自动波特率通过置位 UnACR 起始位来启动，并通过清零 UnACR 起始位来停止。自动波特率一旦结束，起始位就将清零，并且读取该起始位将会返回自动波特率的状态（挂起/完成）。

UARTn 包含两种自动波特率测量模式，用户可以通过 UnACR 模式位进行选择。在模式 0 中，波特率在 UARTn Rx 管脚的两个连续下降沿（起始位的下降沿和最低位的下降沿）处测量。在模式 1 中，波特率在 UARTn Rx 管脚的下降沿和紧跟其后的上升沿之间（起始位的长度）测量。

如果出现超时（速率测量计数器溢出），那么我们可以通过 UnACR AutoRestart 位来自动重新启动波特率测量。若该位置位，则速率测量将在 UARTn Rx 管脚的下一个下降沿重新启动。

自动波特率功能可产生两个中断。

- 如果中断使能，那么将产生 UnIIR ABTOInt 中断（UnIER ABToIntEn 置位且自动波特率测量计数器溢出）。
- 如果中断使能，那么将产生 UnIIR ABEOInt 中断（UnIER ABEoIntEn 置位且自动波特率成功完成）。

我们可以通过置位相应的 UnACR ABTOIntClr 和 ABEOIntEn 位来清零自动波特率中断。

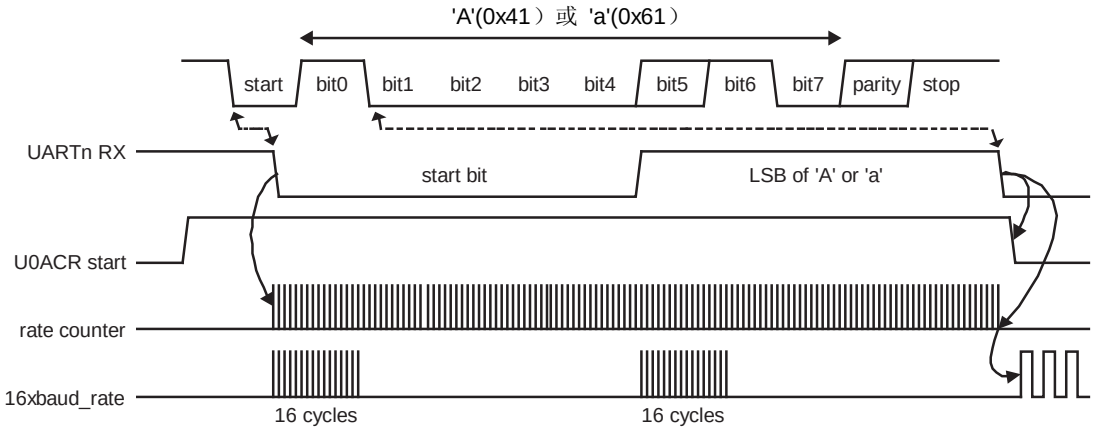
在自动波特率期间，小数波特率发生器通常被禁能(DIVADDVAL=0)。但是，如果小数波特率发生器被使能(DIVADDVAL＞0)，那么它将影响 UARTn Rx 管脚波特率的测量，但 UnFDR 寄存器的值在速率测量后不会被修改。此外，在使用波特率时，对 UnDLM 和 UnDLL 寄存器执行写操作应在写 UnACR 寄存器前完成。UARTn 支持的最小和最大波特率受 PCLK、数据的位数、停止位和奇偶校验位的影响。

$$\text{波特率最小值} = \frac{2\times PCLK}{16\times 2^{15}} \leqslant \text{UARTn}_{\text{波特率}} \leqslant \frac{\text{PCLK}}{16\times(2+\text{数据位}+\text{奇偶位}+\text{停止位})} = \text{最大波特率（7）}$$

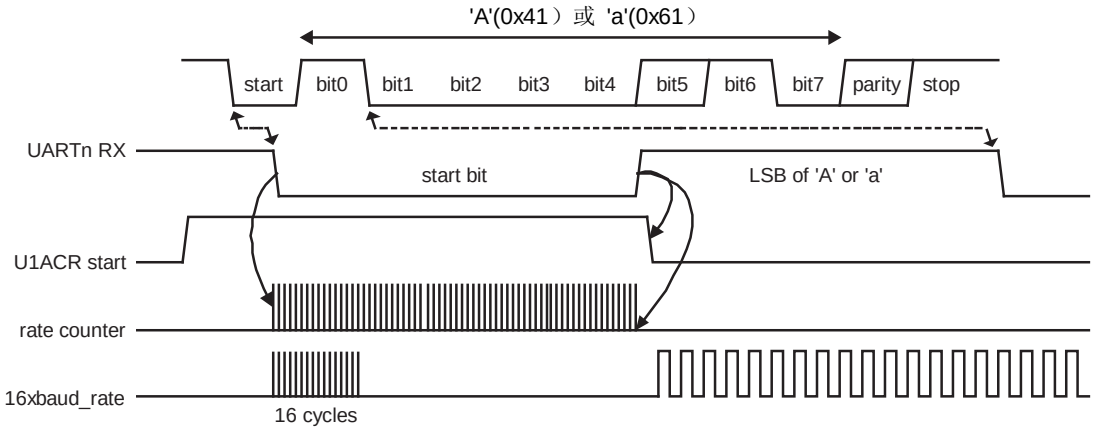
16.3.10.2 自动波特率模式

当软件期望执行“AT”命令时，它采用期望的字符格式对UARTn进行配置并置位UnACR起始位。用户不必关心除数锁存器UnDLL和UnDLM的初始值。由于“A”或“a”ASCII编码（“A”=0x41, “a”=0x61）的关系，UARTn Rx管脚检测起始位且目标字符的LSB的两个边界为两个下降沿。当UnACR起始位置位时，自动波特率协议将执行以下阶段：

1. UnACR 起始位一置位，波特率测量计数器就复位，同时 UARTn UnRSR 复位。UnRSR 波特率切换为最高的速率。
2. UARTn Rx 管脚下降沿触发起始位的开始。速率测量计数器将开始对 PCLK（可选择被小数波特率发生器预分频）进行计数。
3. 在接收起始位的过程中，RSR 波特率输入端产生 16 个脉冲，脉冲频率和（被小数波特率发生器预分频）的 UARTn 输入时钟相同，这样，保证了起始位存放在 UnRSR 中。
4. 在接收起始位（以及模式 0 下字符 LSB）的过程中，速率计数器将随着被预分频的 UARTn 输入时钟（PCLK）递增。
5. 如果是模式 0，那么速率计数器将在 UARTn Rx 管脚的下个下降沿停止。如果是模式 1，那么速率计数器将在 UARTn Rx 管脚的下个上升沿停止。
6. 速率计数器的值被装入 UnDLM/UnDLL，并且波特率将自动切换为正常操作模式。在设置完 UnDLM/UnDLL 后，如果自动波特率结束中断被使能，UnIIR ABEOInt 将置位。接着 UnRSR 继续接收“A/a”字符剩下的其它位。



a. 模式0（起始位和LSB均用于自动波特率）



b. 模式1（仅起始位用于自动波特率）

图 16.1 自动波特率波形图（a 图为模式 0，b 图为模式 1）

16.3.11 IrDA控制寄存器（U3ICR – 0xE007 C024）仅 存在于UART3）

IrDA 控制寄存器仅将 UART3 使能和配置成 IrDA 模式。在发送和接收数据时不应改变 U3ICR 的值，否则可能导致数据丢失或被破坏。

表 16.15 IrDA 控制寄存器（U3ICR – 地址 0xE007 C024）（仅存在于 UART3）

位	符号	值	描述	复位值
0	IrDAEn	0 1	禁能 UART3 的 IrDA 模式，UART3 充当标准的 UART。 使能 UART3 的 IrDA 模式。	0
1	IrDAInv	1	该位为 1 时，串行输入反相，该位对串行输出无影响。为 0 时，串行输入不会反相。	0
2	FixPulseEn		该位为 1 时，使能 IrDA 固定的脉冲宽度模式。	0
5:3	PulseDiv		当 FixPulseEn＝1 时对脉冲进行配置。详见下表。	0
31:6	-	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0

U3ICR 中的 PulseDiv 位用于选择 IrDA 模式（IrDAEn＝1 且 FixPulseEn＝1）使用的脉冲宽度。这些位的值都应该设定，使得最终的脉冲宽度至少为 1.63μs。下表列出了所有可能的脉冲宽度。

表 16.16 IrDA 脉冲宽度

FixPulseEn	PulseDiv	IrDA 发送脉冲宽度（μs）
0	X	3/（16×波特率）
1	0	$2\times T_{PCLK}$
1	1	$4\times T_{PCLK}$
1	2	$8\times T_{PCLK}$
1	3	$16\times T_{PCLK}$
1	4	$32\times T_{PCLK}$
1	5	$64\times T_{PCLK}$
1	6	$128\times T_{PCLK}$
1	7	$256\times T_{PCLK}$

16.3.12 UARTn 小数分频器寄存器（U0FDR – 0xE000 C028，U2FDR – 0xE007 8028，U3FDR – 0xE007 C028）

UARTn 小数分频器寄存器(UnFDR)控制产生波特率的时钟预分频器，并且用户可以通过自己的判断对它进行读写操作。

表 16.17 UARTn 小数分频器寄存器（U0FDR – 地址 0xE000 C028，U2FDR – 0xE007 8028，U3FDR – 0xE007 C028）位描述

位	功能	值	描述	复位值
3:0	DIVADDVAL	0	产生波特率的预分频除数值。如果该字段为 0，小数波特率发生器将不会影响 UARTn 的波特率。	0
7:4	MULVAL	1	波特率预分频乘数值。不管是否使用小数波特率发生器，为了让 UARTn 正常操作，该字段都必须大于或等于 1。	1
31:8	-	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0

该寄存器控制产生波特率的时钟预分频器。时钟进行预分频时的系数为：

$$\frac{MULVAL}{(MULVAL+DIVADDVAL)} \qquad (8)$$

UARTn 波特率可以通过下面的等式计算得到：

$$\text{UARTn 波特率} = \frac{PCLK}{16\times UnDL\times\left(1+\frac{DIVADDVAL}{MULVAL}\right)} \qquad (9)$$

其中，PCLK 指外设时钟，UnDL 的值由 UnDLM 和 UnDLL 寄存器确定（UnDL＝256×UnDLM + UnDLL），DIVADDVAL 和 MULVAL 为 UARTn 小数波特率发生器特定的参数。

MULVAL 和 DIVADDVAL 的值应遵循以下条件：

1. 0＜MULVAL≤15
2. 0≤DIVADDVAL≤15

如果 UnFDR 寄存器值不符合这两个要求，那么小数分频器的输出将不能确定。如果 DIVADDVAL 为 0，那么将禁能小数分频器并且时钟不会被分频。

备注：如果 DIVADDVAL＞0，那么 UnDL≥0x0002 或者 UART 不会在期望的波特率下工作。

在发送/接收数据时不能修改 UnFDR 的值，否则数据可能丢失或被破坏。

使用时应该注意：在实际使用中，UARTn 波特率公式可采用下列式子表示，在这个式子中，确定了无小数波特率发生器时产生的那部分 UART 波特率，以及增加的修正因子：

$$\text{UARTn 波特率} = \frac{PCLK}{16 \times UnDL} \times \frac{MULVAL}{(MULVAL+DIVADDVAL)} \qquad (10)$$

根据这个表达式，小数波特率发生器部分也可以描述成进行 MULVAL/(MULVAL+DIVADDVAL)系数的预分频。

16.3.13 UARTn波特率计算

例 1：使用上面的“UARTn 波特率”公式，我们可以确定：当 PCLK＝20MHz ，UnDL=130（UnDLM=0x00 和 UnDLL=0x82），DIVADDVAL＝0 且 MULVAL＝1 时，系统将可以得到波特率为 9615 波特的 UARTn。

例 2：使用上面的“UARTn 波特率”公式，我们可以确定：当 PCLK＝20MHz，UnDL=93（UnDLM=0x00 和 UnDLL=0x5D），DIVADDVAL＝2 且 MULVAL＝5 时，系统将可以得到波特率为 9600 波特的 UARTn。

波特率值的其它范例：下表列出了在 PCLK＝20MHz 时波特率值的其它范例。

表 16.18 外设时钟为 20MHZ 时的波特率(PCLK=20MHz)

目标波特率	MULVAL=0,DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		%误差[3]
	UnDLM:UnDLL		%误差[3]	UnDLM:UnDLL	小数的预分频值	
	hex[2]	dec[1]		dec[1]	MULDIV/(MULDIV+DIVADDVAL)	
50	61A8	25000	0.0000	25000	1/(1+0)	0.0000
75	411B	16667	0.0020	12500	3/(3+1)	0.0000
110	2C64	11364	0.0032	6250	11/(11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/(3+4)	0.0001
150	208D	8333	0.0040	6250	3/(3+1)	0.0000
300	1047	4167	0.0080	3125	3/(3+1)	0.0000
600	0823	2083	0.0160	1250	3/(3+2)	0.0000
1200	0412	1042	0.0320	625	3/(3+2)	0.0000
1800	02B6	694	0.0640	625	9/(9+1)	0.0000
2000	0271	625	0.0000	625	1/(1+0)	0.0000
3600	015B	347	0.0640	248	5/(5+2)	0.0064
4800	0104	260	0.1600	125	12/(12+13)	0.0000
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600

续表 16.18

目标波特率	MULVAL=0,DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		%误差[3]
	UnDLM:UnDLL		%误差[3]	UnDLM:UnDLL dec[1]	小数的预分频值	
	hex[2]	dec[1]			MULDIV/(MULDIV+DIVADDVAL)	
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

[1] 该行的值是 16 位（DLM：DLL）的等效十进制值。

[2] 该行的值是 16 位（DLM：DLL）的等效十六进制值。

[3] 目标波特率和实际波特率的百分比误差。

16.3.14 UARTn发送使能寄存器（U0TER – 0xE000 C030，U2TER – 0xE007 8030，U3TER – 0xE007 C030)

LPC2400 系列的 UnTER 可以实现软件流控制。当 TXEn＝1 时，只要数据可用，UARTn 发送器就会一直发送数据。TXEn 一变为 0，UARTn 就会停止发送。

表 16.19描述了如何利用TXEn位来实现软件流控制。

表 16.19 UARTn 发送使能寄存器（U0TER – 地址 0xE000 C030，U2TER – 0xE007 8030，U3TER – 0xE007 C030)位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7	TXEN	该位为 1 时（复位后），一旦先前的数据都被发送后，写入 THR 的数据就在 TxD 管脚上输出。如果在发送某字符时该位清零，则结束该字符的发送，但是不再发送更多的字符直至该位重新置位。换言之，该位为 0 时会阻止字符从 THR 或 TX FIFO 传输到发送移位寄存器。在接收到 XOFF 字符（DC3）时，软件通过执行软件握手可以将该位清零。在接收到 XON 字符（DC1）时，软件又能将该位重新置位。	1

16.4 结构

UART0，UART2 和UART3 的结构如图 16.2所示。

APB 接口提供 CPU 或主机与 UART 之间的通信连接。

UARTn 接收器模块 UnRx 监视串行输入线 RXDn 的有效输入。UARTn RX 移位寄存器（UnRSR）通过 RXDn 接收有效的字符。当 UnRSR 接收到一个有效字符时，它将该字符传送到 UARTn RX 缓冲寄存器 FIFO 中，等待 CPU 或主机通过主机接口进行访问。

UARTn 发送器模块 UnTx 接收 CPU 或主机写入的数据并缓冲存放在 UARTn TX 保持寄存器 FIFO（UnTHR）中的数据。UARTn TX 移位寄存器（UnTSR）读取 UnTHR 中的数据并将这些数据通过串行输出管脚 TXDn 发送。

UARTn 波特率发生器模块 UnBRG 产生供 UARTn Tx 模块使用的定时使能信号。UnBRG 时钟输入源为 APB 时钟（PCLK）。主时钟与 UnDLL 和 UnDLM 寄存器所定义的

除数相除得到 UARTn TX 模块使用的时钟。该时钟为 16 倍过采样时钟 NBAUDOUT。

中断接口包含寄存器 UnIER 和 UnIIR。中断接口接收几个来自 UnTX 和 UnRX 模块的单时钟宽度的使能信号。

UnTX 和 UnRX 的状态信息保存在 UnLSR 中。UnTX 和 UnRX 的控制信息保存在 UnLCR 中。

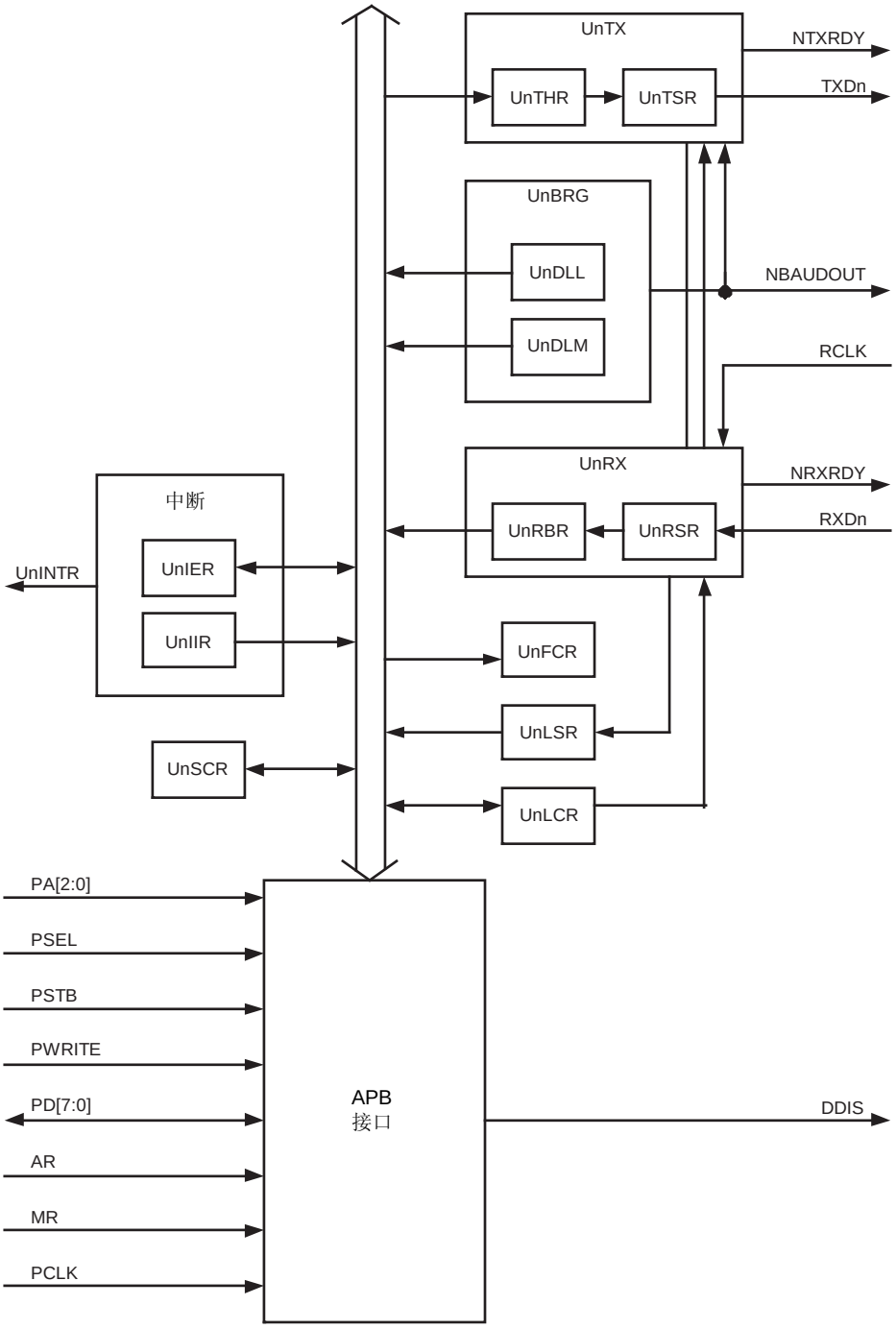


图 16.2 UART0，UART2 和 UART3 的方框图

第17章 通用异步收发器（UART）1

17.1 特性

- UART1 与 UART0/2/3 相同，只是增加了一个调制解调器（Modem）接口；
- 16 字节接收和发送 FIFO；
- 寄存器位置符合’550 工业标准；
- 接收器 FIFO 触发点可为 1，4，8 和 14 字节；
- 内置波特率发生器；
- 包含标准调制解调器接口信号（CTS，DCD，DTS，DTR，RI，RTS）；
- LPC2468 UART1 包含一种使能实现软件流和硬件流控制的机制。

17.2 管脚描述

表 17.1 UART1 管脚描述

管脚名称	类型	描述
RXD1	输入	**串行输入**。串行接收数据。
TXD1	输出	**串行输出**。串行发送数据。
CTS1	输入	**清零以发送**。低电平有效信号指示外部 Modem 是否准备接收从 UART1 通过 TXD1 发送过来的数据。在 Modem 接口的正常操作中(U1MCR[4]=0)，该信号的补码存放在 U1MSR[4]中。状态改变信息存放在 U1MSR[0]中，如果第 4 优先级中断使能(U1IER[3]=1)，则该信息将作为其中断源。 在自动 CTS 模式中，我们也可以使用 CTS1 来控制 UART1 发送器。 清零发送。CTS1 信号是一个异步低电平有效的 Modem 状态信号。用户可以通过读取 Modem 状态寄存器的位 4（CTS）来检测其状态。Modem 状态寄存器的位 0（DCTS）表示自上次读取 MSR 时 CTS1 的状态已发生了改变。如果 Modem 状态中断在 CTS1 电平发生变化且没有使能自动 CTS 模式时被使能，那么将产生中断。CTS1 还可以使用在自动 CTS 模式中，用来控制 UART1 发送器。（IP_3106）
DCD1	输入	**数据载波检测**。低电平有效信号，指示外部 Modem 是否已经与 UART1 建立了通信连接，可以进行数据交换。在 Modem 的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[7]中。状态改变信息保存在 U1MSR[3]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为其中断源。
DSR1	输入	**数据设置就绪**。低电平有效，指示外部 Modem 是否准备与 UART1 建立通信连接。在 Modem 的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[5]中。状态改变信息保存在 U1MSR[1]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为其中断源。
DTR1	输出	**数据终端就绪**。低电平有效，指示 UART1 准备与外部 Modem 建立连接。该信号的补码保存在 U1MCR[0]中。

续表 17.1

管脚名称	类型	描述
RI1	输入	**铃响指示**。低电平有效，指示 Modem 检测到电话的响铃信号。在 Modem 的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[6]中。状态改变信息保存在 U1MSR[2]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为其中断源。
RTS1	输出	**请求发送**。低电平有效，指示 UART1 打算向外部 Modem 发送数据。该信号的补码保存在 U1MCR[1]中。 在自动 RTS 模式中，我们也可以使用 RTS1 来控制发送器 FIFO 的阈值逻辑。 请求发送。RTS1 信号是一个低电平有效信号，它会告知 Modem 或数据集：UART 准备接收数据。通过设置 RTS modem 控制寄存器中的位可以将 RTS1 设为有效（低电平），而由于系统复位或者在回送模式操作过程中或通过清零 MCR 的位 1（RTS）也可将其设为无效（高电平）。在自动 RTS 模式中，RTS1 由发送器 FIFO 阈值逻辑控制。（IP_3106）

17.3 寄存器描述

UART1 包含如表 17.2所示的寄存器。除数锁存访问位（DLAB）位于U1LCR[7]，用于使能对除数锁存的访问。

表 17.2 UART1 寄存器映射

名称	描述	位功能和地址								访问	复位值[1]
		MSB							LSB		
		位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0		
U1RBR	接收器缓冲	8 位读数据								RO	NA
U1THR	发送器保持	8 位写数据								WO	NA
U1DLL	除数锁存 LSB	8 位数据								R/W	0x01
U1DLM	初始锁存 MSB	8 位数据								R/W	0x00
U1IER	中断使能	保留						使能自动波特率超时中断	使能自动波特率结束中断	R/W	0
		使能 CTS 中断		0		使能 Modem 状态中断	使能 RX 线状态中断	使能 THRE 中断	使能 RX 数据可用中断		
U1IIR	中断 ID	保留						ABTO 中断	ABEO 中断	RO	0x01
		FIFO 使能		0		IIR3	IIR2	IIR1	IIR0		
U1FCR	FIFO 控制	Rx 触发		保留			Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0x00

续表 17.2

名称	描述	位功能和地址								访问	复位值[1]
		MSB							LSB		
		位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0		
U1LCR	线控制	DLAB	设置间隔	奇偶固定	偶选择	奇偶使能	停止位个数	字长度选择		R/W	0x00
U1MCR	Modem 控制	CTSen	RTSen	0	回送	0		RTS	DTR	R/W	0x00
U1LSR	线状态	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
U1MSR	Modem 状态	DCD	RI	DSR	CTS	Delta DCD	后沿 RI	Delta DSR	Delta CTS	RO	0x00
U1SCR	高速缓存	8 位数据								R/W	0x00
U1ACR	自动波特率控制	保留[31:10]						ABTO IntClr	ABEO IntClr	R/W	0x00
		保留[7:3]					自动复位	模式	启动		
U1FDR	小数分频器	保留[31:8]								R/W	0x10
		MuIVal				DivAddVaI					
U1TER	发送使能	TXEN	保留							R/W	0x80

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

17.3.1 UART1 接收器缓存寄存器（U1RBR – 0xE001 0000;DLAB=0，只读）

U1RBR 是 UART1 RX FIFO 的最高字节。它包含了最早接收到的字符，我们可以通过总线接口读取它。LSB(bit0)表示最早接收到的数据位。如果接收到的字符小于 8 位，则未使用的 MSB 用 0 填充。

如果要访问 U1RBR，U1LCR 的除数锁存访问位（DLAB）必须为 0。U1RBR 为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶部的字节（即下次读取 RBR 时将读取的字节）相关，因此，要正确地成对读出有效的接收字节及其状态位，应该先读取 U1LSR 寄存器的内容，然后再读取 U1RBR 中的字节。

表 17.3 UART1 接收器缓存寄存器（U1RBR – 地址 0xE001 0000;DLAB=0，只读）位描述

位	符号	描述	复位值
7:0	RBR	UART1 接收器缓存寄存器包含 UART1 RX FIFO 当中最早接收到的字节。	未定义

17.3.2 UART1 发送器保持寄存器（U1THR – 0xE001 0000;DLAB=0，只写）

U1THR 是 UART1 TX FIFO 的最高字节。它包含了 TX FIFO 中最新的字符，我们可以通

过总线接口对它进行写操作。LSB 代表最先发送的位。

如果要访问 U1THR，U1LCR 的除数锁存访问位（DLAB）必须为 0。U1THR 为只写寄存器。

表 17.4 UART1 发送器保持寄存器（U1THR – 地址 0xE001 0000;DLAB=0，只写）位描述

位	符号	描述	复位值
7:0	THR	写 UART1 发送保持寄存器会使数据保存到 UART1 发送 FIFO 当中。当字节到达 FIFO 的最底部并且发送器就绪时，该字节将被发送。	N/A

17.3.3 UART1 除数锁存LSB和MSB寄存器（U1DLL – 0xE001 0000 和U1DLM – 0xE001 0004;DLAB=1）

UART1 除数锁存是波特率发生器的一部分，它保存了用于产生波特率时钟的 AVPB 时钟（PCLK）分频值，波特率时钟必须是目标波特率的 16 倍。U1DLL 和 U1DLM 寄存器一起构成一个 16 位除数，U1DLL 包含的是除数的低 8 位，U1DLM 包含的是除数的高 8 位。0x0000 被看作是 0x0001，因为除数是不允许为 0 的。在访问 UART1 除数锁存寄存器时，U1LCR 的除数锁存访问位（DLAB）必须为 1。要详细了解如何为 U1DLL 和 U1DLM 选择正确值，可以参考本章后面的部分。

$$\text{UART1 波特率} = \frac{PCLK}{16 \times (256 \times U1DLM + U1DLL)}$$

表 17.5 UART1 除数锁存 LSB 寄存器（U1DLL – 地址 0xE001 0000;DLAB=1）位描述

位	符号	描述	复位值
7:0	DLLSB	UART1 除数锁存 LSB 寄存器与 U1DLM 寄存器共同决定 UART1 的波特率。	0x01

表 17.6 UART1 除数锁存 MSB 寄存器（U1DLM – 0xE001 0004;DLAB=1）位描述

位	符号	描述	复位值
7:0	DLMSB	UART1 除数锁存 MSB 寄存器与 U1DLL 寄存器共同决定 UART1 的波特率。	0x00

17.3.4 UART1 中断使能寄存器（U1IER – 0xE001 0004;DLAB=0）

U1IER 用于使能 4 个 UART1 中断源。

表 17.7 UART1 中断使能寄存器（U1IER – 地址 0xE001 0004;DLAB=0）位描述

位	符号	值	描述	复位值
0	RBR 中断使能	U1IER[0]使能 UART1 接收数据可用中断。它还控制字符接收超时中断。		0
		0	禁止 RDA 中断。	
		1	使能 RDA 中断。	
1	THRE 中断使能		U1IER[1]使能 UART1 的 THRE 中断。THRE 的状态可从 U1LSR[5]读出。	0
		0	禁止 THRE 中断。	
		1	使能 THRE 中断。	

续表 17.7

位	符号	值	描述	复位值
2	RX 线状态中断使能		U1IER[2]使能 UART1 RX 线状态中断。该中断的状态可从 U1LSR[4:1]读出。	0
		0	禁止 RX 线状态中断。	
		1	使能 RX 线状态中断。	
3	Modem 状态中断使能	U1IER[3]使能 Modem 中断。该中断的状态可以从 U1MSR[3:0]读出。		0
		0	禁止 Modem 中断。	
		1	使能 Modem 中断。	
6:4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA
7	CTS 中断使能	如果自动 CTS 模式被使能，那么该位使能/禁能 CTS1 信号跳变产生 Modem 状态中断。如果自动 CTS 模式被禁止，若 Modem 状态中断使能(U1IER[3])置位，那么 CTS1 跳变将产生一个中断。 在正常操作下，CTS1 信号跳变将产生 Modem 状态中断，除非中断已被禁止(通过清零 U1IER 寄存器中的位 U1IER[3])。在自动 CTS 模式中，只要 U1IER[3]和 U1IER[7]位置位，CTS1 位上的跳变就会触发一个中断。		0
		0	禁止 CTS 中断。	
		1	使能 CTS 中断。	
8	ABTOIntEn	U1IER8 使能自动波特率超时中断。		0
		0	禁止自动波特率超时中断。	
		1	使能自动波特率超时中断。	
9	ABEOIntEn	U1IER9 使能自动波特率结束中断。		0
		0	禁止自动波特率中断结束。	
		1	使能自动波特率中断结束。	
31:10	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA

17.3.5 UART1 中断标识寄存器（U1IIR – 0xE001 0008，只读）

U1IIR 提供状态代码用于指示挂起中断的中断源和优先级。在访问 U1IIR 过程中，中断被冻结。若在访问 U1IIR 时产生了中断，则该中断将被记录，下次访问 U1IIR 时便可将其读出。

表 17.8 UART1 中断标识寄存器（U1IIR – 地址 0xE001 0008，只读）位描述

位	符号	值	描述	复位值
0	IntStatus	中断状态。注：U1IIR[0]低电平有效。挂起的中断可通过 U1IIR[3:1]确定。		1
		0	至少有 1 个中断被挂起。	
		1	没有挂起的中断。	

续表 17.8

位	符号	值	描述	复位值
3:1	IntId	中断标识。U1IER[3:1]指示与 UART1 Rx FIFO 对应的中断。上文未列出的 U1IER[3:1]的其它组合都为保留值（100, 101, 111）。		0
		011	1 – 接收线状态（RLS）。	
		010	2a – 接收数据可用（RDA）。	
		110	2b – 字符超时指示（CTI）。	
		001	3 – THRE 中断。	
		000	4 – Modem 中断。	
5:4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA
7:6	FIFO 使能	这些位等效于 U1FCR[0]。		0
8	ABEOInt	结束自动波特率中断。如果成功完成自动波特率检测且中断使能，则 ABEOInt 为 1。		0
9	ABTOInt	自动波特率超时中断。如果自动波特率发生了超时且中断使能，则 ABTOInt 为 1。		0
31:10	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA

位 U1IIR[9:8]通过自动波特率函数设定且通知超时或自动波特率结束等情况。自动波特率中断条件可以通过置位自动波特率控制寄存器中相应的位来清除。

当IntStatus位为 1 时，表示没有挂起中断，且IntId位将为 0。当IntStatus为 0 时，表示挂起的是非自动波特率中断。在这种情况下，IntId位会确定中断的类型并按照表 17.9中所述的方式进行处理。只要知道U1IIR[3:0]的状态，中断处理程序就可以确定中断的原因以及如何清除激活的中断。用户如果要在退出中断服务程序前将中断清除，就必须读取U1IIR。

UART1 RLS 中断（U1IIR[3:1]=011）是优先级最高的中断。只要 UART1 Rx 输入产生某个错误条件（溢出错误（OE）、奇偶错误（PE）、帧错误（FE）或间隔中断（BI）），该中断标志就会置位。产生该中断的 UART1 Rx 错误条件可通过查看 U1LSR[4:1]得到。一读取完 U1LSR[4:1]，中断就会被清除。

UART1 RDA 中断（U1IIR[3:1]=010）与 CTI 中断（U1IIR[3:1]=110）共用第二优先级。当UART1 Rx FIFO 到达 U1FCR[7:6]所定义的触发点时，RDA 就会被激活。当 UART1 Rx FIFO 的深度低于触发点时，RDA 复位。当 RDA 中断激活时，CPU 可读出由触发点所定义的数据块。

CTI 中断（U1IIR[3:1]=110）为第二优先级中断。当 UART1 Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符这段时间内没有发生 UART1 Rx FIFO 动作时，将产生该中断。UART1 Rx FIFO 的任何动作（读或写 UART1 RSR）都可以将该中断清除。如果接收到的信息不是触发深度的倍数，那么 CTI 中断将会清空 UART1 RBR。例如，如果某个外设要发送一个 105 个字符的信息，而触发深度为 10 个字符，那么在发送前 100 个字符时，CPU 会接收 10 个 RDA 中断，但在发送剩下的 5 个字符时，CPU 将会接收 1 到 5 个 CTI 中断（视服务程序而定）。

表 17.9 UART1 中断处理

U1IIR[3:0][1]	优先级	中断类型	中断源	中断复位
0001	—	无	无	—
0110	最高	Rx 线状态/错误	OE[2], PE[2], FE[2], 或BI[2]	U1LSR读操作[2]
0100	第二	Rx 数据可用	Rx 数据可用或到达 FIFO（U1FCR0=1）触发深度	U1RBR 读[3]或UART1 FIFO低于触发深度
1100	第二	字符超时指示	Rx FIFO 至少包含 1 个字符，在某段时间无字符输入或移出，这段时间的长短取决于 FIFO 中的字符数以及在（3.5 到 4.5 字符的时间内）设置的触发深度。 准确的时间为： [(字长度)×7-2]×8+[(触发深度－字符数)×8+1]个 RCLK	U1RBR读[3]
0010	第三	THRE	THRE[2]	U1IIR读[4]（如果是中断源）或THR写操作
0000	第四	Modem 状态	CTS 或 DSR 或 RI 或 DCD	MSR 读

[1] “0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。

[2] 详见17.3.10小节“UART1 线状态寄存器（U1LSR - 0xE001 0014，只读）”。

[3] 详见17.3.1小节“UART1 接收器缓存寄存器（U1RBR – 0xE001 0000;DLAB=0，只读）”。

[4] 详见17.3.5小节“UART1 中断标识寄存器（U1IIR – 0xE001 0008，只读 ）”和17.3.2小节“UART1 发送器保持寄存器（U1THR – 0xE001 0000;DLAB=0，只写）”。

UART1 THRE 中断(U1IIR[3:1]=001)为第三优先级中断。如果满足某一个初始化条件，那么当 UART1 THR FIFO 为空时，该中断将激活。这些初始化条件将使 UART1 THR FIFO 被数据填充，以免在系统启动时产生某些 THRE 中断。初始化条件在 THRE＝1，以及自上次 THRE＝1 事件发生后 U1THR 中包含的字符少于 2 个时，它会延时一段时间（一个字符减去停止位）。正是有了这段延时，CPU 才有时间将数据写到 U1THR 中，而无需对 THRE 中断进行译码和服务。如果 UART1 THR FIFO 在某个时间或当前包含的字符超过两个，则将立即产生 THRE 中断。当发生 U1THR 写操作或 U1IIR 读操作并且 THRE 中断优先级最高（U1IIR[3:1]=001）时，THRE 中断复位。

Modem 中断是最低优先级中断，只要在 Modem 输入管脚 DCD,DSR 或 CTS 上有任何状态变化，Modem 中断就被激活。此外，Modem 输入 RI 上如果出现低到高的跳变，将会引起 Modem 中断。Modem 中断源可通过检测 U1MSR[3:0]来确定。U1MSR 读操作会将 Modem 中断清除。

17.3.6 UART1 FIFO控制寄存器（U1FCR – 0xE001 0008，只写）

U1FCR 控制 UART1 RX FIFO 和 TX FIFO 的操作。

表 17.10 UART1 FIFO 控制寄存器（U1FCR – 地址 0xE001 0008，只写）位描述

位	功能	值	描述	复位值
0	FIFO 使能	0	禁能 UART1 FIFO。不能在应用中使用。	0
		1	高电平使能对 UART1 Rx 和 Tx FIFO 以及 U1FCR[7:1]的访问。该位必须置位以实现正确的 UART1 操作。该位的任何变化都将使 UART1 FIFO 清空。	
1	Rx FIFO 复位	0	对任一 UART1 FIFO 无影响。	0
		1	写逻辑 1 到 U1FCR[1]会清除 UART1 Rx FIFO 中的所有字节，并复位指针逻辑。该位自动清零。	
2	Tx FIFO 复位	0	对任一 UART1 FIFO 无影响	0
		1	写逻辑 1 到 U1FCR[2]会清除 UART1 TX FIFO 中的所有字节，并复位指针逻辑。该位自动清零。	
5:3	-	0	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7:6	RX 触发选择	这两个位决定在激活中断之前，接收 UART1 FIFO 必须写入多少个字符。		0
		00	触发点 0（1 个字符或 0x01）	
		01	触发点 1（4 个字符或 0x04）	
		10	触发点 2（8 个字符或 0x08）	
		11	触发点 3（14 个字符或 0x0E）	

17.3.7 UART1 线控制寄存器（U1LCR - 0xE001 000C）

U1LCR 决定发送和接收数据字符的格式。

表 17.11 UART1 线控制寄存器（U1LCR – 地址 0xE001 000C）位描述

位	功能	值	描述	复位值
1:0	字长度选择	00	5 位字符长度。	0
		01	6 位字符长度。	
		10	7 位字符长度。	
		11	8 位字符长度。	
2	停止位选择	0	1 个停止位。	0
		1	2 个停止位（如果 U1LCR[1:0]=00 则为 1.5）。	
3	奇偶使能	0	禁止奇偶产生和校验。	0
		1	使能奇偶产生和校验。	
5:4	奇偶选择	00	奇数。在已发送的字符中 1 的数目和附加的奇偶校验位将为奇数。	0
		01	偶数。在已发送的字符中 1 的数目和附加的奇偶校验位将为偶数。	
		10	强制为 1。	
		11	强制为 0。	
6	间隔控制	0	禁止间隔发送。	0
		1	使能间隔发送；当 U1LCR[6]为高电平有效时，输出管脚 UART1 TXD 强制为逻辑 0。	
7	除数锁存访问位（DLAB）	0	禁止访问除数锁存。	0
		1	使能访问除数锁存。	

17.3.8 UART1 Modem控制寄存器（U1MCR – 0xE001 0010）

U1MCR 用于使能 Modem 回送模式并控制 Modem 的输出信号。

表 17.12 UART1 Modem 控制寄存器（U1MCR – 地址 0xE001 0010）位描述

位	符号	值	描述	复位值
0	DTR 控制		选择 Modem 输出管脚 DTR。该位在 modem 回写模式激活时读出为 0。	0
1	RTS 控制		选择 Modem 输出管脚 RTS。该位在 modem 回写模式激活时读出为 0。	0
3:2	-	NA	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	0
4	回写模式选择		Modem 回写模式提供了一种执行回写测试的诊断机制。发送器输出的串行数据在内部与接收器的串行输入端相连。输入脚 RXD1 对回写模式无影响，输出脚 TXD1 一直保持为“1”。4 个 Modem 输入（CTS, DSR, RI 和 DCD）与外部断开。从外部看，Modem 输出端（RTS, DTR）是无效的。从内部看，4 个 Modem 输出与 4 个 Modem 输入连接。结果变成 U1MSR 的高 4 位由 U1MCR 的低 4 位驱动，而不象在正常模式下是由 4 个 Modem 输入驱动。如此一来，在回写模式下，写 U1MCR 的低 4 位就会产生 Modem 状态中断。	0
		0	禁止 Modem 回写模式。	
		1	使能 modem 回写模式。	
5	-	NA	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
6	RTSen	0	禁止自动 RTS 流控制。	0
		1	使能自动 RTS 流控制。	
7	CTSen	0	禁止自动 CTS 流控制。	0
		1	使能自动 CTS 流控制。	

17.3.9 自动流控制

如果自动 RTS 模式被使能，那么将由 UART1 的接收器 FIFO 硬件对 UART1 的 RTS1 输出进行控制。如果自动 CTS 模式被使能，若 CTS1 输入信号有效，那么 UART1 的 U1TSR 硬件将仅启动发送。

17.3.9.1 自动-RTS

用户通过置位 CTSen 位可以使能自动 RTS 功能。自动 RTS 数据流控制在 U1RBR 模块中产生并链接到已编程好的接收 FIFO 触发深度。如果自动 RTS 被使能，当接收 FIFO 深度到达设定的触发深度时，RTS1 失效（高电平）。因为 UART 可能直到开始发送额外字节后，它才知道 RTS1 失效，所以发送 UART 在到达触发深度后会额外再发送一个字节（假设发送 UART 要发送另外一个字节）。一旦接收 FIFO 到达先前的触发深度，RTS1 就会自动重新生效（低电平）。RTS1 重新生效会指示发送 UART 继续发送数据。

如果自动 RTS 模式被禁止，那么由 RTSen 位控制 UART1 的 RTS1 输出。如果自动 RTS 模式被使能，那么将由硬件控制 RTS1 输出，并且 RTS1 的实际值将被复制到 UART1 的 RTSen 位。如果自动 RTS 使能，软件将只能对 RTSen 位执行读操作。

实例：假设某个UART1 在type550 下操作，它在U1FCR中的触发深度设为 0x2。这样，如果自动RTS被使能，那么接收FIFO只要一包含 8 个字节（见表 17.10），UART1 就让RTS1 输出变无效。此外，只要接收FIFO到达原来的触发深度——4 字节，RTS1 输出就将重新变有效。

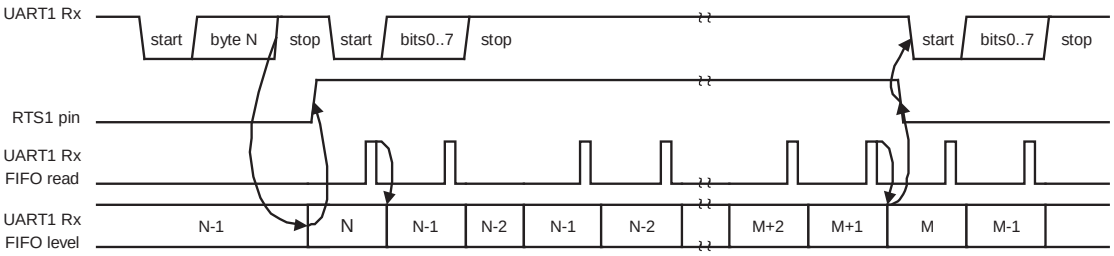


图 17.1 自动 RTS 功能时序

17.3.9.2 自动-CTS

用户可以通过置位CTSen位来使能自动CTS功能。如果自动CTS被使能，那么U1TSR模块中的发送器电路将在发送下一个数据字节之前检查CTS1 输入。当CTS1 有效（低电平）时，发送器发送下一个字节。为了让发送器停止发送后面的字节，CTS1 必须在发送最后一个停止位的中间以前被释放。在自动CTS模式下，CTS1 信号的变化不会触发Modem状态中断，除非CTS中断使能位置位，但U1MSR中的Delta CTS位可能会被置位。表 17.13列出了产生Modem状态中断的各种条件。

表 17.13 Modem 状态中断的产生

使能 Modem 状态中断 (U1IER[3])	CTSen (U1MCR[7])	CTS 中断使能 (U1IER[7])	DeltaCTS (U1MSR[0])	Delta DCD、后沿 RI、Delta DSR(U1MSR[3]、U1MSR[2]或(U1MSR[1]))	Modem 状态中断
0	x	x	x	x	否
1	0	x	0	0	否
1	0	x	1	x	是
1	0	x	x	1	是
1	1	0	x	0	否
1	1	0	x	1	是
1	1	1	0	0	否
1	1	1	1	x	是
1	1	1	x	1	是

自动 CTS 功能减少到主机系统的中断。当流控制使能时，CTS1 状态的变化不会触发主机中断，因为器件会自动控制自己的发送器。如果没有自动 CTS，发送器会发送存放在发送 FIFO 中的任意数据，从而可能导致接收器溢出错误。下图描述了自动 CTS 功能的时序。

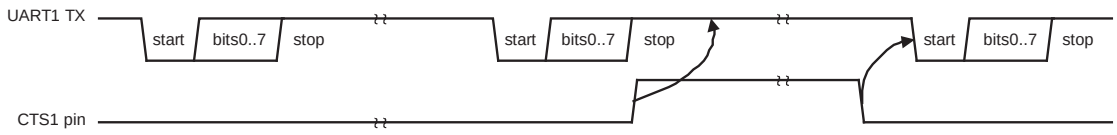


图 17.2 自动 CTS 功能时序图

当发送首个字符时，CTS1 信号有效。待处理的传输一结束，传输就停止。只要 CTS1 变无效（高电平），UART 就会继续发送一个 1 位。CTS1 一旦变无效，就恢复传输并发送起始位，紧跟着发送下个字符的数据位。

17.3.10 UART1 线状态寄存器（U1LSR - 0xE001 0014，只读）

U1LSR 为只读寄存器，它包含了 UART1 TX 和 RX 模块的状态信息。

表 17.14 UART1 线状态寄存器（U1LSR - 0xE001 0014，只读）位描述

位	功能	值	描述	复位值
0	接收数据就绪（RDR）	当 U1RBR 包含未读取的字符时，U1LSR[0]置位；当 UART1 RBR FIFO 为空时，U1LSR0 清零。		0
		0	U1RBR 为空。	
		1	U1RBR 包含有效数据。	
1	溢出错误（OE）	OE 在错误发生后会立即置位。U1LSR 读操作会清零 U1LSR[1]。当 UART1 RSR 包含新字符而 UART1 RBR FIFO 已满时，U1LSR[1]置位。此时 UART1 RBR FIFO 不会被覆盖，UART1 RSR 中的字符将丢失。		0
		0	溢出错误状态未激活。	
		1	溢出错误状态激活。	
2	奇偶错误（PE）	在接收字符的奇偶位指示为错误状态时将产生一个奇偶错误。U1LSR 读操作会清零 U1LSR[2]位。奇偶错误检测时间取决于 U1FCR[0]的值。 **注：**奇偶错误与 UART1 RBR FIFO 中的顶部字符相关。		0
		0	奇偶错误状态未激活。	
		1	奇偶错误状态激活。	
3	帧错误（FE）	在接收字符的停止位为 0 时将产生帧错误。U1LSR 读操作会清零 U1LSR[3]位。帧错误检测时间取决于 U1FCR[0]的值。一旦检测到帧错误，RX 就会尝试与数据重新同步并假设错误的停止位确实是先前的一个起始位。但即使没有出现帧错误，也不能假定下一个接收到的字节是正确的。 **注：**帧错误与 UART1 RBR FIFO 中的顶部字符相关。		0
		0	帧错误状态未激活。	
		1	帧错误状态激活。	
4	间隔中断（BI）	在发送整个字符（起始位、数据位、奇偶位和停止位）过程中 RXD1 如果都保持逻辑 0，则产生间隔中断。一旦检测到中断条件，接收器就会立即进入空闲状态直至 RXD1 变为全 1 状态。U1LSR 读操作会清零该状态位。间隔检测的时间取决于 U1FCR[0]的值。 **注：**间隔中断与 UART1 RBR FIFO 中读出的字符相关。		0
		0	间隔中断状态未激活。	
		1	间隔中断状态激活。	
5	发送保持寄存器空（THRE）	一检测到 UART1 THR 为空，THRE 就置位。U1THR 写操作会清零该位。		1
		0	U1THR 包含有效数据。	
		1	U1THR 为空。	
6	发送器空（TEMT）	当 U1THR 和 U1TSR 均为空时，TEMT 置位。当 U1TSR 或 U1THR 包含有效数据时，TEMT 清零。		1
		0	U1THR 和/或 U1TSR 包含有效数据。	
		1	U1THR 和 U1TSR 为空。	

续表 17.14

位	功能	值	描述	复位值
7	RX FIFO 错误（RXFE）	当一个带有 RX 错误（例如帧错误、奇偶错误或间隔中断）的字符装入 U1RBR 时，该位置位。当读取 U1LSR 寄存器并且 UART1 FIFO 中不再有错误发生时，该位清零。		0
		0	U1RBR 中没有 UART1 RX 错误，或 U1FCR[0]=0。	
		1	UART1 RBR 包含至少一个 UART1 RX 错误。	

17.3.11 UART1 Modem状态寄存器（U1MSR – 0xE001 0018）

U1MSR 是只读寄存器，它包含 Modem 输入信号的状态信息。U1MSR[3:0]在读取 U1MSR 时清零。需要注意的是，Modem 信号对 UART1 的操作无直接影响，但有助于通过软件执行 Modem 信号操作。

表 17.15 UART1 Modem 状态寄存器（U1MSR – 地址 0xE001 0018）位描述

位	符号	值	描述	复位值
0	Delta CTS	当输入 CTS 状态发生变化时，该位置位。读取 U1MSR 时清零。		0
		0	没有检测到 modem 输入 CTS 上的状态变化。	
		1	检测到 modem 输入 CTS 上的状态变化。	
1	Delta DSR	当输入 DSR 状态发生变化时，该位置位。读取 U1MSR 时清零。		0
		0	没有检测到 modem 输入 DSR 上的状态变化。	
		1	检测到 modem 输入 DSR 上的状态变化。	
2	后沿 RI	当输入 RI 发生低到高的跳变时，该位置位。读取 U1MSR 时清零。		0
		0	没有检测到 Modem 输入 RI 上的状态变化。	
		1	检测到 Modem 输入 RI 上低到高的跳变。	
3	Delta DCD	当输入 DCD 状态发生变化时，该位置位。读取 U1MSR 时清零。		0
		0	没有检测到 Modem 输入 DCD 上的状态变化。	
		1	检测到 Modem 输入 DCD 上的状态变化。	
4	CTS	清零发送状态。输入信号 CTS 的补码。在回写模式下，该位连接到 U1MCR[1]。		0
5	DSR	数据设备就绪状态。输入信号 DSR 的补码。在回写模式下，该位连接到 U1MCR[0]。		0
6	RI	响铃指示状态。输入信号 RI 的补码。在回写模式下，该位连接到 U1MCR[2]。		0
7	DCD	数据载波检测状态。输入信号 DCD 的补码。在回写模式下，该位连接到 U1MCR[3]。		0

17.3.12 UART1 高速缓存寄存器（U1SCR – 0xE001 001C）

在 UART1 操作时 U1SCR 无效。用户可自由地读写该寄存器。中断接口无需向主机指示 U1SCR 是否发生了读写操作。

表 17.16 UART1 高速缓存寄存器（U1SCR – 0xE001 001C）位描述

位	符号	描述	复位值
7:0	Pad	一个可读可写的字节。	0x00

17.3.13 UART1 自动波特率控制寄存器（U1ACR – 0xE001 0020）

在用户测量生成波特率的输入时钟/数据速率期间，整个测量过程就是由 UART1 自动波特率控制寄存器(U1ACR)进行控制的。用户可自由地读写该寄存器。

表 17.17 UART1 自动波特率控制寄存器（U1ACR – 地址 0xE001 0020）位描述

位	符号	值	描述	复位值
0	Start	自动波特率结束后该位自动清零。		0
		0	自动波特率停止（自动波特率不运行）。	
		1	自动波特率启动（自动波特率正在运行）。自动波特率运行位。该位在自动波特率结束后自动清零。	
1	Mode	自动波特率模式选择位。		0
		0	模式 0。	
		1	模式 1。	
2	AutoRestart	0	不重新启动。	0
		1	如果超时则重新启动(计数器在下一个 UART1 Rx 下降沿时重新启动)。	
7:3	—	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0
8	ABEOIntClr		自动波特率中断结束清零位（仅可写访问）。	0
		0	写 0 无影响。	
		1	写 1 会将 U1IIR 中相应的中断清除。	
9	ABTOIntClr		自动波特率超时中断清零位（仅可写访问）。	0
		0	写 0 无影响。	
		1	写 1 会将 U1IIR 中相应的中断清除。	
31:10	—	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	0

17.3.14 自动波特率

UART1 自动波特率功能可用于测量基于“AT”协议(Hayes 命令)的输入波特率。如果使能，那么自动波特率功能部件将测量接收数据流的位所消耗的时间，并因此设置除数锁存寄存器 U1DLM 和 U1DLL。

自动波特率通过置位 U1ACR 起始位来启动，并通过清零 U1ACR 起始位来停止。自动波特率一旦结束，起始位就将清零，并且读取该起始位将会返回自动波特率的状态（挂起/完成）。

UART1 中有两种自动波特率测量模式，用户可通过 U1ACR 模式位进行选择。在模式 0 中，波特率在 UART1 Rx 管脚的两个连续下降沿（起始位的下降沿和最低位的下降沿）测量。在模式 1 中，波特率在 UART1 Rx 管脚的下降沿和紧跟其后的上升沿之间（起始位的长度）测量。

若出现超时（速率测量计数器溢出），我们可以通过 U1ACR AutoRestart 位来自动重新启动波特率测量。若该位置位，速率测量将在 UART1 Rx 管脚的下一个下降沿重新启动。

自动波特率功能可产生两个中断。

- 如果中断使能，那么将产生 U1IIR ABTOInt 中断（U1IER ABToIntEn 置位且自动波特率测量计数器溢出）。
- 如果中断使能，那么将产生 U1IIR ABEOInt 中断（U1IER ABEOIntEn 置位且自动波特率成功完成）。

我们可以通过置位相应的 U1ACR ABTOIntClr 和 ABEOIntEn 位来清零自动波特率中断。

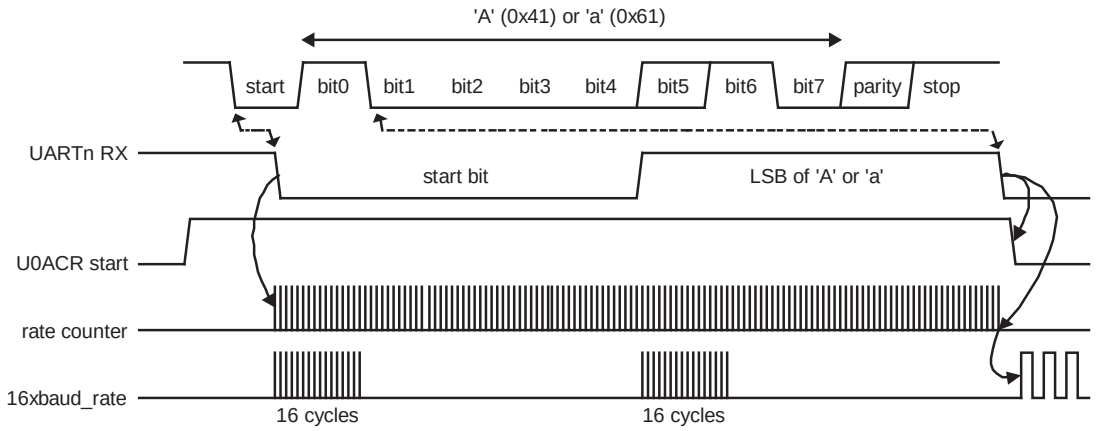
在自动波特率期间，小数波特率发生器通常被禁能(DIVADDVAL=0)。但是，如果小数波特率发生器被使能(DIVADDVAL＞0)，那么它将影响 UART1 Rx 管脚波特率的测量，但 U1FDR 寄存器的值在速率测量后不会被修改。此外，在使用波特率时，对 U1DLM 和 U1DLL 寄存器执行写操作应在写 U1ACR 寄存器前完成。UART1 支持的最小和最大波特率受 PCLK、数据位的数量、停止位和奇偶校验位的影响。

$$\text{波特率最小值}=\frac{2\times PCLK}{16\times 2^{15}}\leqslant \text{UART1}_{\text{波特率}}\leqslant\frac{\text{PCLK}}{16\times(2+\text{数据位}+\text{奇偶位}+\text{停止位})}=\text{最大波特率}\quad(12)$$

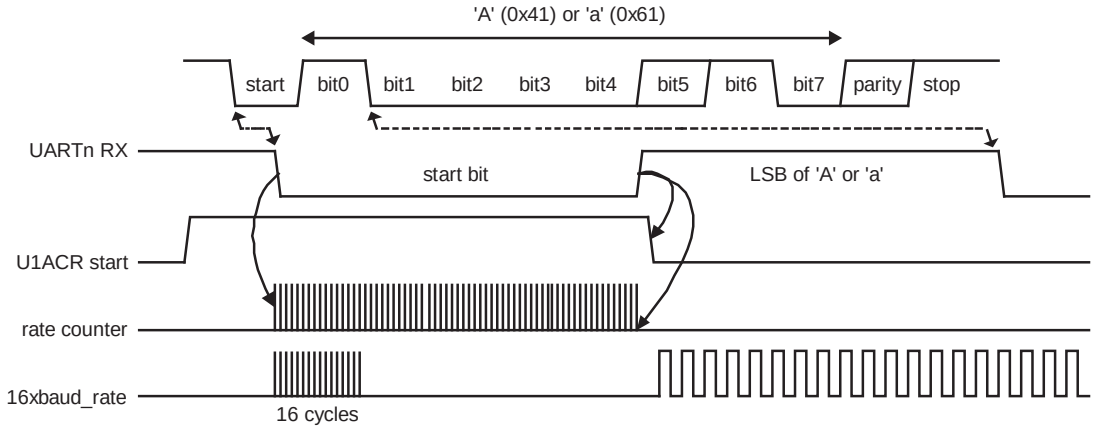
17.3.15 自动波特率模式

当软件期望执行“AT”命令时，它采用期望的字符格式对UART1进行配置并置位U1ACR起始位。用户不必关心除数锁存器U1DLL和U1DLM的初始值。由于“A”或“a”ASCII编码（“A”=0x41, “a”=0x61）的关系，UART1 Rx管脚检测起始位且目标字符的LSB的两个边界为两个下降沿。当U1ACR起始位置位时，自动波特率协议将执行以下阶段：

1. U1ACR 起始位一置位，波特率测量计数器就将复位，同时 UART1 U1RSR 复位。U1RSR 波特率切换为最高的速率。
2. UART1 Rx 管脚下降沿触发起始位的开始。速率测量计数器将开始对 PCLK（可选择被小数波特率发生器预分频）进行计数。
3. 在接收起始位的过程中，RSR 波特率输入端产生 16 个脉冲，脉冲频率和（被小数波特率发生器预分频的）UART1 输入时钟相同，这样，保证了起始位存放在 U1RSR 中。
4. 在接收起始位（以及模式 0 下字符 LSB）的过程中，速率计数器将随着被预分频的 UART1 输入时钟（PCLK）递增。
5. 如果是模式 0，那么速率计数器将在 UART1 Rx 管脚的下个下降沿停止。如果是模式 1，那么速率计数器将在 UART1 Rx 管脚的下个上升沿停止。
6. 速率计数器的值被装入 U1DLM/U1DLL，并且波特率将自动切换为正常操作模式。在设置完 U1DLM/U1DLL 后，如果自动波特率结束中断被使能，U1IIR ABEOInt 将置位。接着 U1RSR 继续接收“A/a”字符剩下的其它位。



a. 模式0（起始位和LSB位用于自动波特率）



b. 模式1（仅起始位用于自动波特率）

图 17.3 自动波特率波形图（a 图为模式 0，b 图为模式 1）

17.3.16 UART1 小数分频器寄存器（U1FDR – 0xE001 0028）

UART1 小数分频器寄存器(U1FDR)控制产生波特率的时钟预分频器，并且用户可以通过自己的判断对它进行读写操作。

表 17.18 UART1 小数分频器寄存器（U1FDR – 地址 0xE001 0028）位描述

位	功能	值	描述	复位值
3:0	DIVADDVAL	0	产生波特率的预分频除数值。如果该字段为 0，小数波特率发生器将不会影响 UART1 的波特率。	0
7:4	MULVAL	1	波特率预分频乘数值。不管是否使用小数波特率发生器，为了让 UART1 正常操作，该字段都必须大于或等于 1。	1
31:8	-	NA	保留，用户软件不要向保留位写 1。从保留位读出的值未被定义。	NA

该寄存器控制产生波特率的时钟预分频器。

UART1 波特率可以通过下面的等式计算得到：

$$\text{UART1 波特率} = \frac{PCLK}{16 \times U1DL \times \left(1 + \frac{DIVADDVAL}{MULVAL}\right)} \qquad (13)$$

其中，PCLK 指外设时钟，U1DL 的值由 U1DLM 和 U1DLL 寄存器确定，DIVADDVAL 和 MULVAL 为 UART1 小数波特率发生器特定的参数。

MULVAL 和 DIVADDVAL 的值应遵循以下条件：

1. 0＜MULVAL≤15
2. 0≤DIVADDVAL≤15

如果 U1FDR 寄存器值不符合这两个要求，那么小数分频器的输出将不能确定。如果 DIVADDVAL 为 0，那么将禁能小数分频器并且时钟不会被分频。

备注：如果 DIVADDVAL＞0，那么 UnDL≥0x0002 或者 UART 不会在期望的波特率下工作。

在发送/接收数据时不能修改 U1FDR 的值，否则数据可能丢失或被破坏。

使用时应该注意：在实际使用中，UART1 波特率公式可采用下列式子表示，在这个式子中，确定了无小数波特率发生器时产生的那部分 UART 波特率，以及增加的修正因子：

$$\text{UART1 波特率} = \frac{PCLK}{16 \times U1DL} \times \frac{MULVAL}{(MULVAL+DIVADDVAL)}$$

根据这个表达式，小数波特率发生器部分也可以描述成进行 MULVAL/(MULVAL+DIVADDVAL)系数的预分频。

17.3.17 UART1 波特率的计算

例 1：使用上面的“UART1 波特率”公式，我们可以确定：当 PCLK＝20MHz ，UnDL=130（U1DLM=0x00 和 U1DLL=0x82），DIVADDVAL＝0 且 MULVAL＝1 时，系统将可以得到波特率为 9615 波特的 UART1。

例 2：使用上面的“UART1 波特率”公式，我们可以确定：当 PCLK＝20MHz，U1DL=93（U1DLM=0x00 和 U1DLL=0x5D），DIVADDVAL＝2 且 MULVAL＝5 时，系统将可以得到波特率为 9600 波特的 UART1。

表 17.19 外设时钟为 20MHz 时的波特率(PCLK=20MHz)

目标波特率	MULVAL=0,DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		
	U1DLM:U1DLL		%误差[3]	U1DLM:U1DLL	小数的预分频值	%误差[3]
	hex[2]	dec[1]		dec[1]	$\frac{MULDIV}{(MULDIV+DIVADDVAL)}$	
50	61A8	25000	0.0000	25000	1/ (1+0)	0.0000
75	411B	16667	0.0020	12500	3/ (3+1)	0.0000
110	2C64	11364	0.0032	6250	11/ (11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/ (3+4)	0.0001
150	208D	8333	0.0040	6250	3/ (3+1)	0.0000
300	1047	4167	0.0080	3125	3/ (3+1)	0.0000

续表 17.19

目标波特率	MULVAL=0,DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		%误差[3]
	U1DLM:U1DLL		%误差[3]	U1DLM:U1DLL	小数的预分频值	
	hex[2]	dec[1]		dec[1]	$\frac{MULDIV}{(MULDIV+DIVADDVAL)}$	
600	0823	2083	0.0160	1250	3/ (3+2)	0.0000
1200	0412	1042	0.0320	625	3/ (3+2)	0.0000
1800	02B6	694	0.0640	625	9/ (9+1)	0.0000
2000	0271	625	0.0000	625	1/ (1+0)	0.0000
2400	0209	521	0.0320	250	12/ (12+13)	0.0000
3600	015B	347	0.0640	248	5/ (5+2)	0.0064
4800	0104	260	0.1600	125	12/ (12+13)	0.0000
7200	00AE	174	0.2240	124	5/ (5+2)	0.0064
9600	0082	130	0.1600	93	5/ (5+2)	0.0064
19200	0041	65	0.1600	31	10/ (10+11)	0.0064
38400	0021	33	1.3760	12	7/ (7+12)	0.0594
56000	0021	22	1.4400	13	7/ (7+5)	0.0160
57600	0016	22	1.3760	19	7/ (7+1)	0.0594
112000	000B	11	1.4400	6	7/ (7+6)	0.1600
115200	000B	11	1.3760	4	7/ (7+12)	0.0594
224000	0006	6	7.5200	3	7/ (7+6)	0.1600
448000	0003	3	7.5200	2	5/ (5+2)	0.3520

[1] 该行的值是 16 位（DLM：DLL）的等效十进制值。

[2] 该行的值是 16 位（DLM：DLL）的等效十六进制值。

[3] 目标波特率和实际波特率的百分比误差。

17.3.18 UART1 发送使能寄存器（U1TER – 0xE001 0030）

LPC2400 系列的 U1TER 除了配有完整的硬件流控制（上述的自动 CTS 和自动 RTS 控制），它还可以实现软件流控制。当 TxEn＝1 时，只要数据可用，UART1 发送器就会一直发送数据。TxEn 一变为 0，UART1 就会停止发送。

尽管表 17.20描述了如何利用TxEn位来完成硬件流控制，但还是强烈建议采用通过 UART1 硬件实现的自动流控制功能，而将TxEn的作用域限制在软件流控制。

LPC2400 的 U1TER 可以实现软件和硬件流控制。当 TXEn＝1 时，只要数据可用，UART1 发送器就将一直发送数据。TXEn 一变为 0，UART1 就会停止发送。

表 17.20描述了如何利用TXEn位来实现软件流控制。

表 17.20 UART1 发送使能寄存器（U1TER – 地址 0xE001 0030）位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7	TXEN	该位为 1 时（复位后），一旦先前的数据都被发送后，写入 THR 的数据就在 TXD 管脚上输出。如果在发送某字符时该位清零，则结束该字符的发送，但是不再发送更多的字符直至该位重新置位。换言之，该位为 0 时会阻止字符从 THR 或 TX FIFO 传输到发送移位寄存器。当软件检测到硬件握手信号——TX-许可信号（CTS）为假时，它会将该位清零。或者当软件接收到 XOFF 字符（DC3）时，它通过执行软件握手信号也可以将该位清零。在接收到 XON 字符（DC1）时，软件又能将该位重新置位。当软件检测到 Tx-许可信号（CTS）为真时，或在接收到 XON 字符（DC1）时，它又能将该位重新置位。	1

17.4 结构

UART1 的结构如图 17.4所示。

APB 接口提供 CPU 或主机与 UART1 之间的通信连接。

UART1 接收器模块 U1RX 监视串行输入线 RXD1 的有效输入。UART1 RX 移位寄存器（U1RSR）通过 RXD1 接收有效的字符。当 U1RSR 接收到一个有效字符时，它将该字符传送到 UART1 RX 缓冲寄存器 FIFO 中，等待 CPU 或主机通过主机接口进行访问。

UART1 发送器模块 U1Tx 接收 CPU 或主机写入的数据并缓冲存放在 UART1 TX 保持寄存器 FIFO（U1THR）中的数据。UART1 TX 移位寄存器（U1TSR）读取 U1THR 中的数据并将这些数据通过串行输出管脚 TXD1 发送。

UART1 波特率发生器模块 U1BRG 产生供 UART1 TX 模块使用的定时使能信号。U1BRG 模块的时钟输入源为 APB 时钟（PCLK）。主时钟与 U1DLL 和 U1DLM 寄存器所定义的除数相除得到 UART1 TX 模块使用的时钟。该时钟为 16 倍过采样时钟 NBAUDOUT。

Modem 接口包含寄存器 U1MCR 和 U1MSR。该接口负责 Modem 外设和 UART1 之间的信号交换（handshaking）。

中断接口包含寄存器 U1IER 和 U1IIR。中断接口接收几个来自 U1TX 和 U1RX 模块的单时钟宽度的使能信号。

U1TX 和 U1RX 的状态信息保存在 U1LSR 中。U1TX 和 U1RX 的控制信息保存在 U1LCR 中。

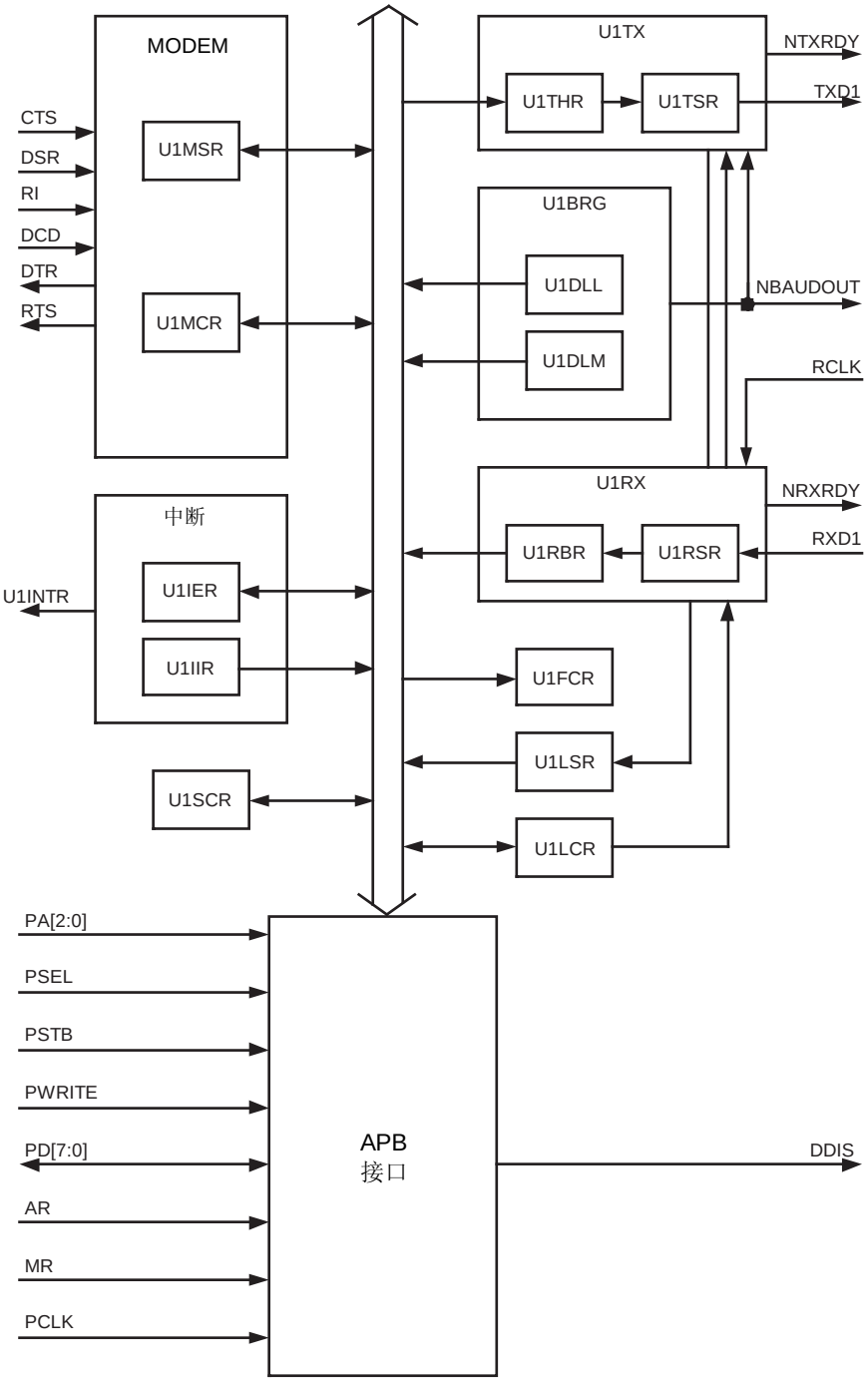


图 17.4 LPC2400 的 UART1 结构方框图

第18章 SPI接口

18.1 特性

- 遵循串行外设接口(SPI)规范；
- 同步、串行、全双工通信；
- SPI 主机或从机；
- 最大数据位速率为输入时钟速率的 1/8；
- 每次传输 8 到 16 位。

18.2 SPI概述

SPI 是一个全双工的串行接口。它可以处理在一个给定总线上多个互连的主机和从机。在给定的数据传输过程中，接口上只能有一个主机和一个从机能够通信。在一次数据传输中，主机总是向从机发送 8 到 16 位数据，而从机也总是向主机发送一个字节数据。

18.3 SPI数据传输

图 18.1所示为SPI的 4 种不同数据传输格式的时序。该时序图描述的是 8 位数据的传输。需要注意的是，该时序图分成了 3 个水平的部分。第一部分描述SCK和SSEL信号。第二部分描述了CPHA=0 时的MOSI和MISO信号。第三部分描述了CPHA＝1 时的MOSI和MISO信号。

在时序图的第一部分需要注意两点。第一，时序图包含了 CPOL 设置为 0 和 1 的情况。第二，SSEL 信号的激活和未激活。当 CPHA=0 时，SSEL 信号在数据传输之间时总是保持未激活状态。当 CPHA=1 时则不能保证这一点（信号有可能保持激活状态）。

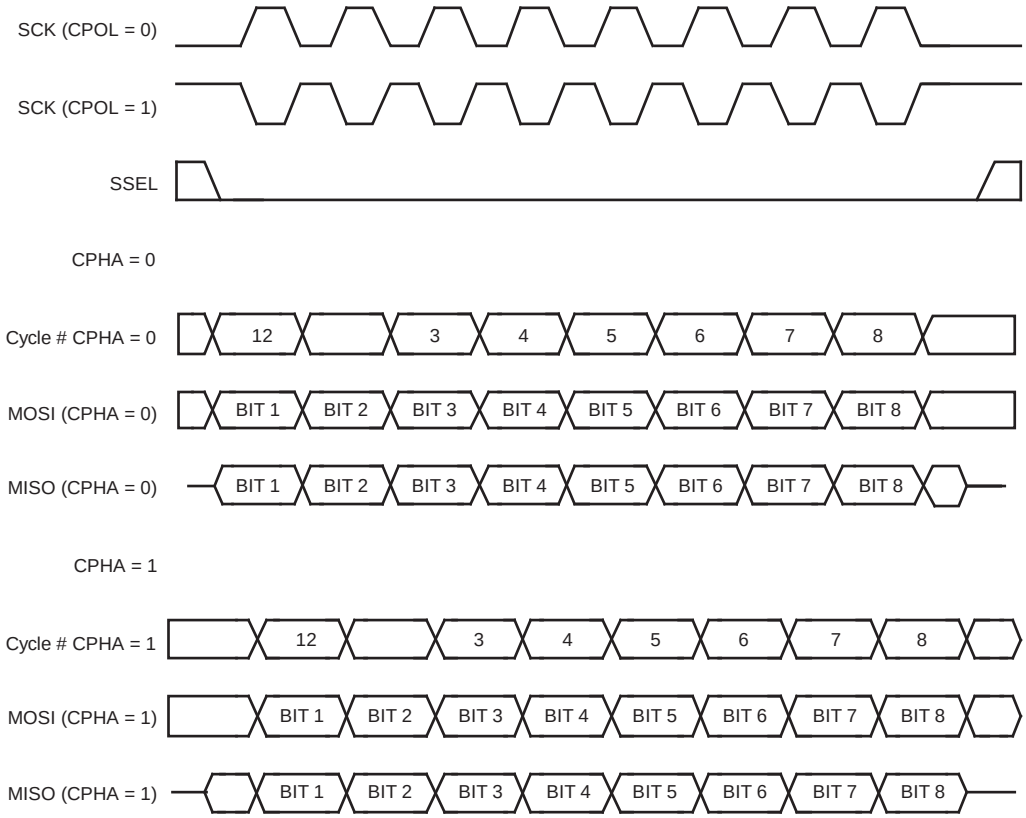


图 18.1 SPI 数据传输格式（CPHA=0 和 CPHA=1）

数据和时钟的相位关系在表 18.1中描述。该表汇集了在下列情况下CPOL和CPHA的每一种设定：

- 当驱动第一个数据位时
- 当驱动所有其它数据位时
- 当采样数据时

表 18.1 SPI 数据和时钟的相位关系

CPOL 和 CPHA 的设定	驱动的第一个数据	驱动的其它数据	采样的数据
CPOL=0, CPHA=0	在第一个 SCK 上升沿之前	SCK 下降沿	SCK 上升沿
CPOL=0, CPHA=1	第一个 SCK 上升沿	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=0	在第一个 SCK 下降沿之前	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=1	第一个 SCK 下降沿	SCK 下降沿	SCK 上升沿

8 位传输起始和停止时间的定义取决于器件为主机还是从机，以及 CPHA 变量的设定。

当器件为主机时，传输的起始由包含发送数据字节的主机来指示。此时，主机可激活时钟并开始传输。当传输的最后一个时钟周期结束时，传输结束。

当器件为从机并且 CPHA=0 时，传输在 SSEL 信号激活时开始，并在 SSEL 变为无效时结束。当器件为从机且 CPHA=1 时，如果该器件被选择，则传输从第一个时钟沿开始，并在数据采样的最后一个时钟沿结束。

18.4 SPI外设描述

18.4.1 概述

有 4 个寄存器控制SPI外设。它们将在18.6 节“寄存器描述”中详细讲述。

SPI 控制寄存器包含许多可编程位来控制 SPI 模块的功能。该寄存器必须在数据传输之前进行设定。

SPI 状态寄存器包含只读位，用于监视 SPI 接口的状态，包括一般性功能和异常状况。该寄存器的主要用途是检测数据传输的完成，这通过 SPIF 位来实现。寄存器中的其它位用于指示异常状况。异常情况将在后面描述。

SPI 数据寄存器用于提供发送和接收的数据字节。串行数据实际的发送和接收通过 SPI 模块逻辑中的内部移位寄存器来实现。在发送时向 SPI 数据寄存器写入数据。数据寄存器和内部移位寄存器之间没有缓冲区。写数据寄存器会使数据直接进入内部移位寄存器。因此数据只能在没有执行数据发送时写入该寄存器。读数据带有缓冲区。当传输结束时，接收到的数据转移到一个单字节的数据缓冲区，下次传输时将其读出。读 SPI 数据寄存器将返回读数据缓冲区的值。

当 SPI 模块处于主模式时，SPI 时钟计数器寄存器用于控制时钟速率。当 SPI 为主机时，该寄存器必须在数据传输之前设定。当 SPI 模块处于从模式时，该寄存器无效。

SPI 控制器具有一个使能位（SPEN），该位控制以下功能：

- 激活 SPI I/O 口
- 使能 SPI 内部状态机。如果传输正在进行，使能信号无效，则传输将结束，再禁能 SPI 控制器。
- 禁能其它可被禁止的逻辑转换为功率。
- 使能位不禁止寄存器访问 SPI 控制器。

SPI 所使用的 I/O 口为标准 CMOS I/O 口。设计上没有实现开漏 SPI 选项。当器件设置为从机时，它的 I/O 口只有在被有效的 SSEL 信号选择时才有效。

18.4.2 主机操作

下面的步骤描述了 SPI 设置为主机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。

1. 将 SPI 时钟计数寄存器设置为所需要的时钟率。
2. 将 SPI 控制寄存器设置为所需要的设定。
3. 将要发送的数据写入 SPI 数据寄存器。该写操作启动 SPI 数据传输。
4. 等待 SPI 状态寄存器中的 SPIF 位置位。SPIF 位将在 SPI 数据传输的最后一个周期之后置位。
5. 读取 SPI 状态寄存器。
6. 从 SPI 数据寄存器中读出接收到的数据（可选）
7. 如果有更多数据需要发送，则跳到第 3 步。

注：读或写 SPI 数据寄存器来清零 SPIF 状态位。因此，如果不执行可选的 SPI 数据寄存器读操作，则需要执行该寄存器的写操作以清零 SPIF 状态位。

18.4.3 从机操作

下面的步骤描述了 SPI 设置为从机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。要求驱动 SPI 逻辑的系统时钟速度至少 8 倍于 SPI。

1. 将 SPI 控制寄存器设置为所需要的设定。
2. 将要发送的数据写入 SPI 数据寄存器（可选）。注意这只能在从机 SPI 传输没有进行时执行。
3. 等待 SPI 状态寄存器中的 SPIF 位置位。SPIF 位将在 SPI 数据传输的最后一个采样时钟沿后置位。
4. 读取 SPI 状态寄存器。
5. 从 SPI 数据寄存器中读出接收到的数据（可选）。
6. 如果有更多数据需要发送，则跳到第 2 步。

注：读或写 SPI 数据寄存器来清零 SPIF 状态位。因此，至少需要执行一个该寄存器的读或写操作来清零 SPIF 状态位。

18.4.4 异常状况

读溢出

当 SPI 模块内部读缓冲区包含处理器没有读出的数据而新的传输已经完成时，就会发生读溢出。状态寄存器中的 SPIF 位置位表示读缓冲区包含了有效数据。当一次传输结束时，SPI 模块需要将接收到的数据移到读缓冲区。如果 SPIF 位有效(读缓冲区已满)，则新接收到的数据将会丢失，而状态寄存器的读溢出 (ROVR)位将被激活。

写冲突

我们在前面提到过，在 SPI 模块总线接口与内部移位寄存器之间没有写缓冲区。这样在 SPI 数据传输过程中不应向 SPI 数据寄存器写入数据。不能向 SPI 数据寄存器写入数据的时间从传输启动时开始，直到 SPIF 有效时读取状态寄存器为止。如果在这段时间内写 SPI 数据寄存器，写入的数据将会丢失，状态寄存器中的写冲突位（WCOL）将有效。

模式错误

当 SPI 模块为主机时，如果 SSEL 信号被激活，则表示有另外一个主机将该器件选择为从机。这种状态称为模式错误。当检测到一个模式错误时，状态寄存器的模式错误位（MODF）位有效，SPI 信号驱动器关闭，而 SPI 模式转换为从模式。

如果管脚 Px.y/SSEL/…在管脚功能选择寄存器 0 中指定 SSEL 功能，则 SSEL 信号在 SPI 控制器为主机时必须无效。

从机中止

如果 SSEL 信号在传输结束之前变为无效，则从机传输将被认为中止。此时，正在处理的发送或接收数据都将丢失，状态寄存器的从机中止（ABRT）位将有效。

18.5 管脚描述

表 18.2 SPI 管脚描述

管脚名称	类型	描述
SCK	输入/输出	**串行时钟**。SPI 是用于同步 SPI 接口间数据传输的时钟信号。SPI 总是由主机驱动并且从机接收。时钟可编程为高电平有效或低电平有效。它只在数据传输时才被激活，其它任何时候都处于非激活状态或三态。
SSEL	输入	**从机选择**。SPI 从机选择信号是一个低电平有效信号，用于指示被选择参与数据传输的从机。每个从机都有各自特定的从机选择输入信号。在数据处理之前，SSEL 必须为低电平并在整个处理过程中保持低电平。如果在数据传输中 SSEL 信号变为高电平，传输将被中止。这种情况下，从机返回到空闲状态并将任何接收到的数据丢弃。对于这样的异常没有其它的指示。该信号不直接由主机驱动。可通过软件使用一个通用 I/O 口来驱动。 在 LPC2400（不像其它较早的 Philips ARM 器件）上，当仅在主模式下使用 SPI 接口时，可将 SSEL 管脚用于不同的功能。例如，控制（hosting）SSEL 功能的管脚可被配置为输出数字 GPIO 管脚并用来选择其中一种 SPI 从机。
MISO	输入/输出	**主机输入从机输出**。MISO 信号是一个单向的信号，它将数据从从机传输到主机。当器件为从机时，串行数据从该端口输出。当器件为主机时，串行数据从该端口输入。当从机没有被选择时，将该信号驱动为高阻态。
MOSI	输入/输出	**主机输出从机输入**。MOSI 信号是一个单向的信号，它将数据从主机传输到从机。当器件为主机时，串行数据从该端口输出。当器件为从机时，串行数据从该端口输入。

18.6 寄存器描述

SPI包含 5 个寄存器，见表 18.3。所有寄存器都可以字节、半字和字的形式访问。

表 18.3 SPI 寄存器映射

名称	描述	访问	复位值[1]	地址
S0SPCR	SPI 控制寄存器。该寄存器控制 SPI 的操作。	R/W	0x00	0xE002 0000
S0SPSR	SPI 状态寄存器。该寄存器显示 SPI 的状态。	RO	0x00	0xE002 0004
S0SPDR	SPI 数据寄存器。该双向寄存器为 SPI 提供发送和接收的数据。发送数据通过写该寄存器提供。SPI0 接收的数据可从该寄存器中读出。	R/W	0x00	0xE002 0008
S0SPCCR	SPI 时钟计数寄存器。该寄存器控制主机 SCK0 的频率。	R/W	0x00	0xE002 000C
S0SPINT	SPI 中断标志寄存器。该寄存器包含 SPI 接口的中断标志。	R/W	0x00	0xE002 001C

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

18.6.1 SPI控制寄存器（S0SPCR - 0xE002 0000）

S0SPCR 寄存器根据每个配置位的设定来控制 SPI0 的操作。

表 18.4 SPI 控制寄存器（S0SPCR – 地址 0xE002 0000）位描述

位	符号	值	描述	复位值
1:0	-	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	BitEnable	0 1	SPI 控制器每次传输发送和接收 8 位数据。 SPI 控制器发送和接收位 11:8 选择的位数。	0
3	CPHA	0 1	时钟相位控制决定了 SPI 传输时数据和时钟的关系，并控制从机传输的起始和结束。 数据在 SCK 的第一个时钟沿采样。传输从 SSEL 信号激活时开始，并在 SSEL 信号无效时结束。 数据在 SCK 的第二个时钟沿采样。当 SSEL 信号激活时，传输从第一个时钟沿开始并在最后一个采样时钟沿结束。	0
4	CPOL	0 1	时钟极性控制。 SCK 为高电平有效。 SCK 为低电平有效。	0
5	MSTR	0 1	主模式选择。 SPI 处于从模式。 SPI 处于主模式。	0
6	LSBF	0 1	LSBF 用来控制传输的每个字节的移动方向。 SPI 数据传输 MSB (位 7)在先。 SPI 数据传输 LSB (位 0)在先。	0
7	SPIE	0 1	串行外设中断使能。 SPI 中断被禁止。 每次 SPIF 或 MODF 有效时都会产生硬件中断。	0
11:8	BITS	1000 1001 1010 1011 1100 1101 1110 1111 0000	当该寄存器的位 2 为 1 时，这个字段控制每次传输的位数： 每次传输 8 位 每次传输 9 位 每次传输 10 位 每次传输 11 位 每次传输 12 位 每次传输 13 位 每次传输 14 位 每次传输 15 位 每次传输 16 位	0000
15:12	-	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

18.6.2 SPI状态寄存器（S0SPSR - 0xE002 0004）

S0SPSR 寄存器根据配置位的设定来控制 SPI0 的操作。

表 18.5 SPI 状态寄存器（S0SPSR – 地址 0xE002 0004）位描述

位	功能	描述	复位值
2:0	保留	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
3	ABRT	从机中止。该位为 1 时表示发生了从机中止。当读取该寄存器时，该位清零。	0

续表 18.5

位	功能	描述	复位值
4	MODF	模式错误。为 1 时表示发生了模式错误。先通过读取该寄存器清零 MODF 位，再写 SPI0 控制寄存器。	0
5	ROVR	读溢出。为 1 时表示发生了读溢出。当读取该寄存器时，该位清零。	0
6	WCOL	写冲突。为 1 时表示发生了写冲突。先通过读取该寄存器清零 WCOL 位，再访问 SPI 数据寄存器。	0
7	SPIF	SPI 传输完成标志。为 1 时表示一次 SPI 数据传输完成。在主模式下，该位在传输的最后一个周期置位。在从机模式下，该位在 SCK 的最后一个数据采样边沿置位。当第一次读取该寄存器时，该位清零。然后才能访问 SPI 数据寄存器。 **注**：SPIF 不是 SPI 中断标志。中断标志位于 SPINT 寄存器中。	0

18.6.3 SPI数据寄存器（S0SPDR – 0xE002 0008）

双向数据寄存器为 SPI 提供数据的发送和接收。发送数据通过将数据写入该寄存器来实现。SPI 接收的数据可从该寄存器中读出。处于主模式时，写该寄存器将启动 SPI 数据传输。从数据传输开始到 SPIF 状态位置位并且还没有读取状态寄存器的这段时间内不能对该寄存器执行写操作。

表 18.6 SPI 数据寄存器（S0SPDR – 地址 0xE002 0008）位描述

位	符号	描述	复位值
7:0	DataLow	SPI 双向数据端口。	0x00
15:8	DataHigh	如果 SPCR 的位 2 为 1 且位 11:8 不是 1000，那么这些位的部分或全部含有其它的发送和接收位。当选择少于 16 位时，这些位中较高的位读为 0。	0x00

18.6.4 SPI时钟计数寄存器（S0SPCCR – 0xE002 000C）

该寄存器控制主机 SCK 的频率。寄存器指示构成一个 SPI 时钟的 PCLK 周期的数目。该寄存器的值必须为偶数。因此 bit0 必须为 0。该寄存器的值还必须大于等于 8。如果寄存器的值不符合上述条件，可能导致产生不可预测的动作。

表 18.7 SPI 时钟计数寄存器（S0SPCCR – 地址 0xE002 000C）位描述

位	功能	描述	复位值
7:0	计数值	SPI0 时钟计数值设定。	0x00

SPI0 速率可以这样进行计算：PCLK/SPCCR0 值。PCLK 速率为 CCLK/APB 分频器速率，由 PCLKSEL0 寄存器的内容决定。

18.6.5 SPI测试控制寄存器（SPTCR – 0xE002 0010）

注意：该寄存器中的位仅用于验证操作。在正常操作中不应使用该寄存器。

表 18.8 SPI 测试控制寄存器（SPTCR – 地址 0xE002 0010）位描述

位	功能	描述	复位值
0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7:1	Test	SPI 测试模式。该位为 0 时，SPI 正常操作。该位为 1 时，SCK 将总是打开，并且与主模式选择和数据的有效设置无关。	0

18.6.6 SPI测试状态寄存器（SPTSR – 0xE002 0014）

注意：该寄存器中的位仅用于验证操作。在正常操作中不应使用该寄存器。

该寄存器是 SPI 状态寄存器的复制。这两个寄存器的不同点是：读该寄存器将不会启动清零这些状态位所必须的事件序列。如果各个位的写数据为 1，则写该寄存器将设置中断。

表 18.9 SPI 测试状态寄存器（SPTSR – 地址 0xE002 0014）位描述

位	功能	描述	复位值
2:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
3	ABRT	从机中止。	0
4	MODF	模式错误。	0
5	ROVR	读溢出。	0
6	WCOL	写冲突。	0
7	SPIF	SPI 传输完成标志。	0

18.6.7 SPI中断寄存器（S0SPINT – 0xE002 001C）

该寄存器包含 SPI0 接口的中断标志。

表 18.10 SPI 中断寄存器（S0SPINT – 地址 0xE002 001C）位描述

位	功能	描述	复位值
0	SPI 中断标志	SPI 中断标志。由 SPI 接口置位以产生中断。向该位写入 1 清零。 **注**：当 SPIE=1 并且 SPIF 和 WCOL 位中至少有一位为 1 时该位置位。但是，只有当 SPI 中断位置位并且 SPI0 中断在 VIC 中被使能，SPI 中断才能由中断处理软件处理。	0
7:1	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

18.7 结构

图 18.2所示为SPI0 接口中的SPI方框图。

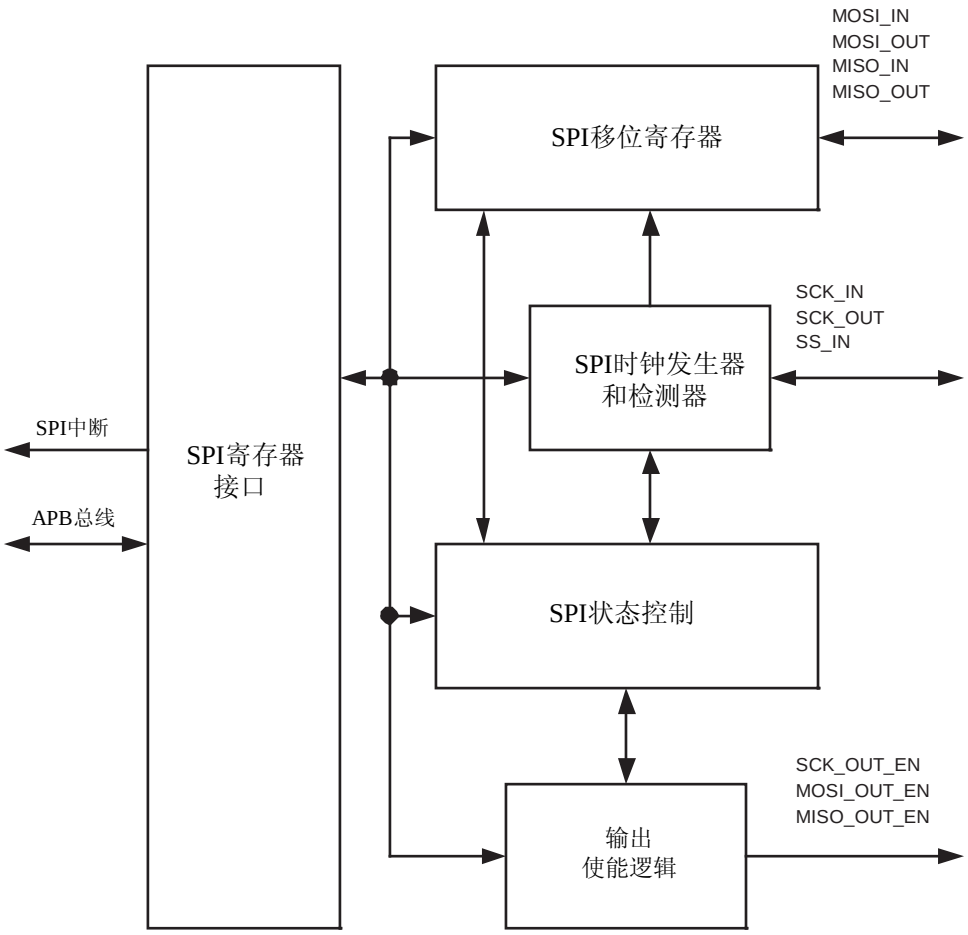

MOSI_IN
MOSI_OUT
MISO_IN
MISO_OUT
SPI移位寄存器
SCK_IN
SCK_OUT
SS_IN
SPI时钟发生器
和检测器
SPI中断
APB总线
SPI寄存器
接口
SPI状态控制
SCK_OUT_EN
MOSI_OUT_EN
MISO_OUT_EN
输出
使能逻辑

图 18.2 SPI 方框图

第19章 SSP0/1 接口

19.1 特性

- 兼容 Motorola SPI、4 线 TI SSI 和 National 半导体的 Microwire 总线；
- 同步串行通信；
- 主机或从机操作；
- 8 帧收发 FIFO；
- 每帧 4～16 位；
- GPDMA 支持的 DMA 传输。

19.2 描述

SSP 是一个同步串行端口（SSP）控制器，可在 SPI、4 线 SSI 或 Microwire 总线上操作。它与总线上多个主机和从机相互作用。在给定的数据传输过程中，总线上只能有一个主机和一个从机进行通信。数据传输原则上是全双工的，4～16 位帧的数据由主机发送到从机或由从机发送到主机。但实际上，大多数情况下只有一个方向上的数据流包含有意义的数据。

LPC2468 具有两个串行端口控制器－SSP0 和 SSP1。

19.3 管脚描述

表 19.1 SSP 管脚描述

管脚名称	类型	接口管脚名称/功能			管脚描述
		SPI	SSI	Microwire	
SCK0/1	I/O	SCK	CLK	SK	**串行时钟。**SCK/CLK/SK 是用来同步数据传输的时钟信号。它由主机驱动，从机接收。当使用 SPI 接口时，时钟可编程为高有效或低有效，否则，时钟总是高有效。SCK1 的状态只能在数据传输过程中改变，在任何其它时间里，SSPn 使其保持无效状态或不驱动它（使其处于高阻态）。
SSEL0/1	I/O	SSEL	FS	CS	**帧同步/从机选择。**当 SSPn 是总线主机时，它在串行数据启动前或串行数据传输终止后立刻驱动该信号来指示一次所选总线和模式的合适数据传输。当 SSPn 是总线从机时，该信号根据使用的通信协议来限制主机数据的去向。 当只有一个总线主机和一个总线从机时，主机的帧同步或从机选择信号直接与从机相应的输入相连。当总线上有多个从机时，必须要进一步限制这些从机的帧选择/从机选择输入，以防一次传输有多个从机响应。

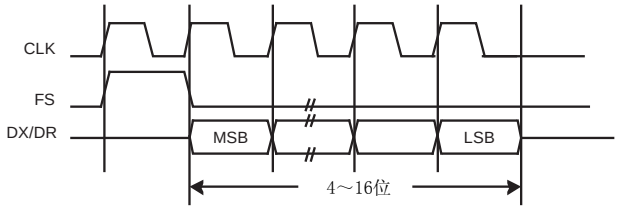
续表 19.1

管脚名称	类型	接口管脚名称/功能			管脚描述
		SPI	SSI	Microwire	
MISO0/1	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**主机输入从机输出**。MISO 信号使串行数据从从机传输到主机。当 SSPn 是从机时，串行数据从该信号输出。当 SSPn 是主机时，该信号输出串行数据的时钟。当 SSPn 是从机且未被 FS/SSEL 选择时，它将不驱动该信号（使其处于高阻态）。
MOSI0/1	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**主机输出从机输入。**MOSI 信号使串行数据从主机传输到从机。当 SSPn 是主机时，串行数据从该信号输出。当 SSPn 是从机时，该信号输出串行数据的时钟。

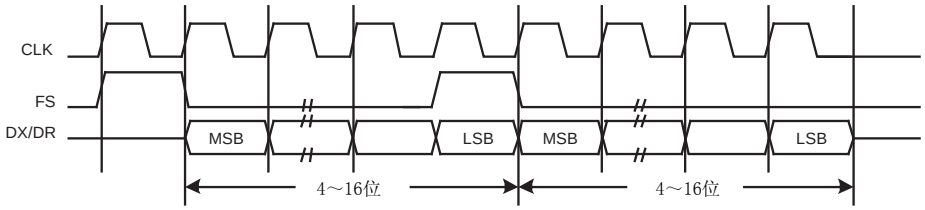
19.4 总线描述

19.4.1 Texas仪器同步串行(SSI)数据帧格式

图 19.1给出了SSP模块支持的 4 线Texas仪器同步串行数据帧的格式。



a. 单帧传输



b. 连续/连续两帧传输

图 19.1 Texas 仪器同步串行数据帧的格式：

a) 单帧传输 b) 连续/连续两帧的传输

假设器件在该模式下配置为主机，当 SSP 处于空闲模式时，CLK 和 FS 强制为低，发送数据线 DX 为三态。只要发送 FIFO 的底端入口包含数据，FS 将在一个 CLK 周期内保持高电平。被发送的数据从发送 FIFO 传输到发送逻辑的串行移位寄存器。在 CLK 的下个上升沿，4～16 位数据帧的 MSB 从 DX 管脚移出。同样地，接收到的 MSB 从片外串行从器件的 DR 管脚移入。

然后，SSP 和片外串行从器件在每个 CLK 下降沿的控制下将每一个数据位送入相应的串行移位器。LSB 被锁存后，接收到的数据在 CLK 的首个上升沿从串行移位器传递到接收 FIFO。

19.4.2 SPI帧格式

SPI 接口是一个 4 线接口，它的 SSEL 信号用作从机选择。SPI 格式的主要特性是 SCK 信号的无效状态和相位可通过 SSPCR0 控制寄存器的 CPOL 位和 CPHA 位来编程设定。

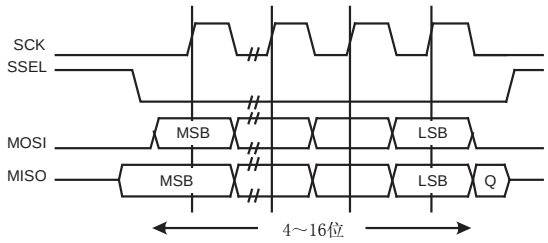
19.4.2.1 时钟极性（CPOL）和相位（CPHA）控制

当 CPOL 时钟极性控制位为 0 时，它将在 SCK 管脚上产生一个稳定的低电平。如果 CPOL 时钟极性控制位为 1，那么在不发生数据传输时 CLK 管脚上将出现一个稳定的高电平。

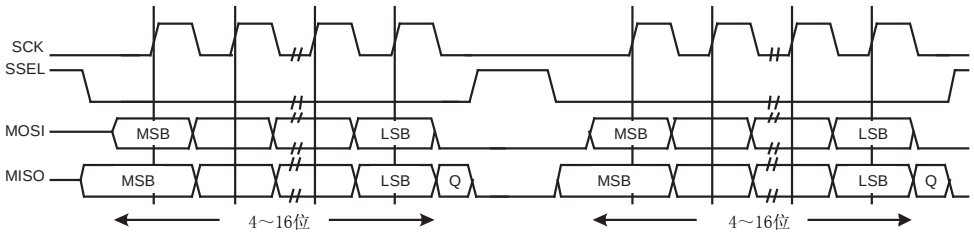
CPHA 控制位用来选择捕获数据的时钟沿，这个边沿的状态允许改变。它将对第一个数据捕获沿出现前由于允许或不允许时钟跳变而发送的第一个位产生最重大的影响。当 CPHA 相位控制位为 0 时，数据在第 1 个时钟沿跳变时被捕获。如果 CPHA 时钟相位控制位为 1，那么数据在第 2 个时钟沿跳变时被捕获。

19.4.2.2 CPOL=0, CPHA=0 时的SPI格式

CPOL=0，CPHA=0 时SPI格式的单帧传输和连续传输信号时序如图 19.2所示。



a. CPOL=0和CPHA=0时的单帧传输



b. CPOL=0和CPHA=0时的连续传输

图 19.2 CPOL＝0 和 CPHA＝0 时的 SPI 帧格式 （a）单帧 和 b）连续传输）

这种配置在空闲期间：

- CLK 信号强制为低
- SSEL 强制为高
- 发送 MOSI/MISO 处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。这就使得从机数据使能到主机的 MISO 输入线上。主机的 MOSI 被使能。

1/2 个 SCK 周期后，有效主机数据被传输到 MOSI 管脚。由于主机和从机数据都被设置，再过 1/2 个 SCK 周期后 SCK 主机时钟管脚将变高。

此时，数据在 SCK 信号的上升沿被捕获，保持到 SCK 的下降沿。

在发送单个字时，当数据字的所有位发送完后，最后一位被捕获后的一个 SCK 周期内，

SSEL 返回到空闲的高电平状态。

但是，在连续顺序的发送过程中，在每个数据字传输之间 SSEL 信号必须为高。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外围寄存器中的数据，不允许改变。因此，在每次数据传输之间主器件必须拉高从器件的 SSEL 管脚来使能串行外设数据的写操作。当连续传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲状态。

19.4.2.3 CPOL=0，CPHA=1 时的SPI格式

CPOL=0，CPHA=1 时SPI格式的传输信号时序如图 19.3所示，它包括单帧传输和连续传输两种方式。

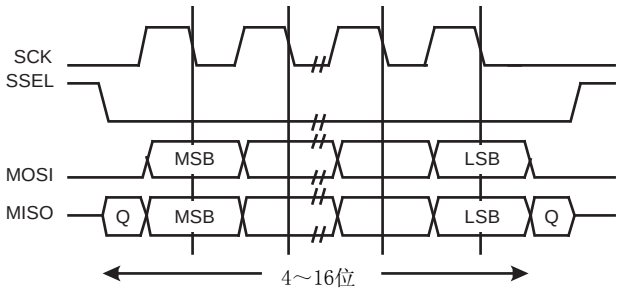


图 19.3 CPOL＝0 和 CPHA=1 时的 SPI 帧格式

这种配置在空闲期间：

- CLK 信号强制为低
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。主机的 MOSI 管脚使能。再过 1/2 个 SCK 周期，主机和从机的有效数据都被使能输出到各自的发送线上。同时，SCK 出现上升沿跳变。

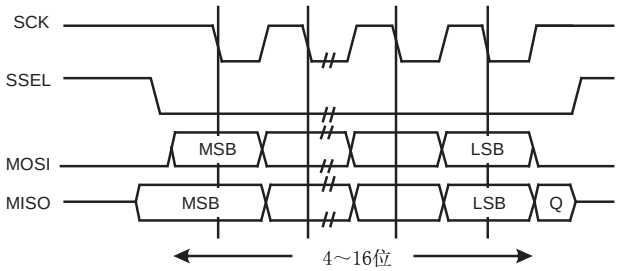
然后，数据在 SCK 信号的下降沿被捕获并保持到 SCK 信号的上升沿。

在单个字的传输过程中，当所有位传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 线返回到空闲的高电平状态。

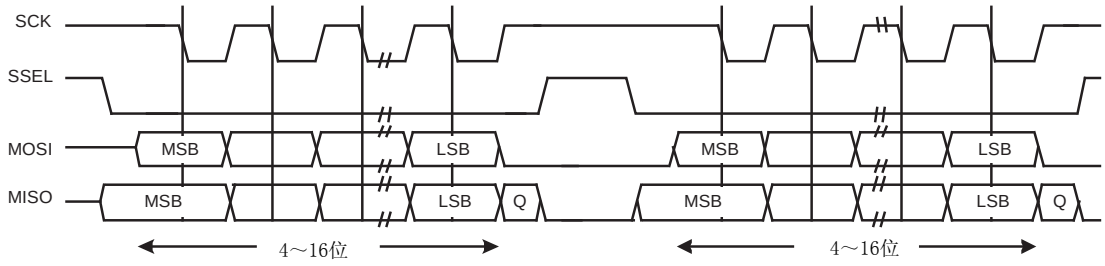
对于连续的顺序传输，SSEL 在两个连续的数据字传输之间保持低电平，终止传输的方法与单个字传输相同。

19.4.2.4 CPOL=1, CPHA=0 时的SPI格式

CPOL=1，CPHA=0 时SPI格式的单帧和连续传输信号时序如图 19.4所示。



a. CPOL=1和CPHA=0 时的单帧传输



b. CPOL=1和CPHA=0 时的连续传输

图 19.4 CPOL=1 和 CPHA=0 时的 SPI 帧格式（a）单帧和 b) 连续传输）

这种配置在空闲期间：

- CLK 信号强制为高
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据，使得从机数据立即传输到主机的 MISO 线上。主机的 MOSI 管脚被使能。

1/2 个 SCK 周期后，有效主机数据被传输到 MOSI 线。由于主机和从机数据都被设置，再过 1/2 个 SCK 周期后 SCK 主机时钟管脚将变低。这就意味着数据在 SCK 信号的下降沿被捕获，并保持到 SCK 的上升沿。

在发送单个字时，当数据字的所有位发送完后，最后一位被捕获后一个 SCK 周期内，SSEL 线返回到空闲的高电平状态。

但是，在连续顺序的发送过程中，在每个数据字传输之间 SSEL 信号必须为高。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外围寄存器中的数据，不允许改变。因此，在每次数据传输之间主器件必须拉高从器件的 SSEL 管脚来使能串行外设数据的写操作。当连续传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 管脚返回到空闲状态。

19.4.2.5 CPOL=1，CPHA=1 时的SPI格式

CPOL=1，CPHA=1 时SPI格式的传输信号时序如图 19.5所示，它包含单帧传输和连续传输两种方式。

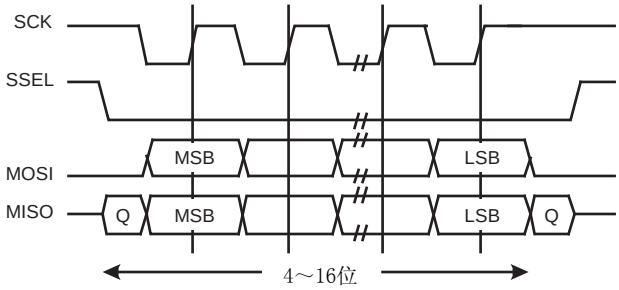


图 19.5 CPOL=1 和 CPHA=1 时的 SPI 帧格式

这种配置在空闲期间：

- CLK 信号强制为高
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。主机的 MOSI 管脚使能。再过 1/2 个 SCK 周期，主机和从机的有效数据都被使能输出到各自的发送线上。同时，SCK 出现下降沿跳变使能。然后，数据在 SCK 信号的上升沿被捕获并保持到 SCK 信号的下降沿。

在单个字的传输过程中，当所有位传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲的高电平状态。对于连续的顺序传输，SSEL 管脚仍然保持有效的低电平状态，直到最后一个字的最后一位捕获结束后，它再返回到上述的空闲状态。总的说来，SSEL 管脚在两个连续的数据字传输之间保持低电平，终止传输的方法与单个字传输相同。

19.4.3 半导体Microwire帧格式

图 19.6所示为Microwire帧格式的单帧传输。图 19.7所示为Microwire帧格式的连续帧传输。

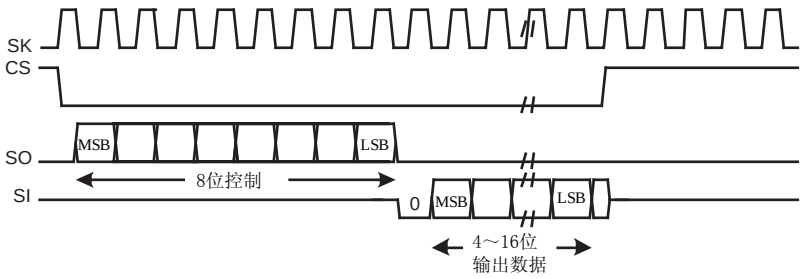


图 19.6 Microwire 帧格式（单帧传输）

Microwire 格式与 SPI 格式类似，但它的发送是半双工而非全双工模式，数据从主机传输到从机。每次串行发送以 SSP 传输到片外从器件的一个 8 位控制字开始。在发送控制字的过程中，SSP 不接收数据。控制字发送结束后，片外从器件对其进行译码，在 8 位控制信息的最后一位发送完一个串行时钟后，才响应到来的所需数据。返回的数据长度为 4～16 位，使得总的数据帧长度在 13～25 位之间。

这种配置在空闲期间：

- SK 信号强制为低
- CS 强制为高
- 发送数据线 SO 可强制为低

发送过程由写入到发送 FIFO 的一个控制字节来触发。CS 的下降沿使发送 FIFO 底端入口的数据传输到发送逻辑的串行移位寄存器，8 位控制帧的 MSB 移出到 SO 管脚。CS 在帧发送过程中保持低电平。SI 在帧发送过程中保持三态。

片外串行从器件在每个 SK 的上升沿将每个控制位锁存到串行移位器。当从器件完成最后一位的锁存后，一个时钟等待周期内对控制字节进行译码，然后从器件通过将数据发回给 SSP 来响应。每一位在 SK 的下降沿驱动到 SI 上。SSP 又在 SK 的上升沿将每一位锁存。在帧传输结束后，对于单帧传输，在最后一位被锁存到接收串行移位器后的一个时钟周期内 CS 信号被拉高，使数据传输到接收 FIFO。

注：在 LSB 被接收移位器锁存后或当 CS 变为高电平时，片外从器件的接收线在任何一个 SK 的下降沿都呈现三态。

连续传输过程的数据发送开始和结束的方法都与单帧传输相同。所不同的是，在连续传输过程中，CS 持续有效（保持低电平），数据连续发送。当前数据帧的 LSB 被接收后，下一帧的控制字节立刻直接发送。在一帧数据的 LSB 被锁存到 SSP 后，每个接收到的数据在 SK 的下降沿传送到接收移位器。

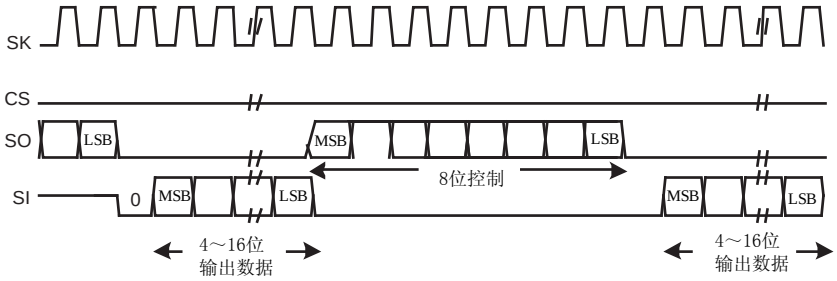


图 19.7 Microwire 帧格式（连续传输）

19.4.3.1 Microwire模式中CS相对SK的建立和保持时间

Microwire 模式中，当 CS 变低后，SSP 从机在 SK 的上升沿对接收数据的首位进行采样。因此，主机驱动 SK 自由运行时必须确保 CS 信号相对 SK 的上升沿有足够的建立和保持时间。

图 19.8给出了建立和保持时间的要求。相对SSP从机采样接收数据的首个位的SK上升沿，CS的建立时间至少为SSP的SK周期的 2 倍。相对于之前的SK上升沿，CS的保持时间至少为一个SK周期。

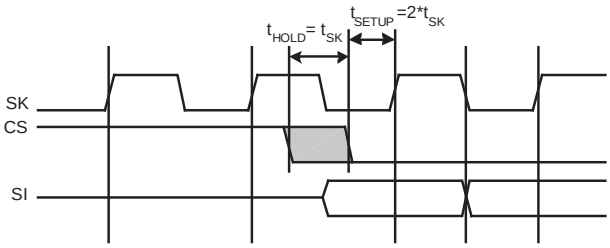


图 19.8 Microwire 帧格式的建立和保持时间

19.5 寄存器描述

SSP控制器基址的寄存器偏移量，请见表 19.2。

表 19.2 SSP 寄存器映射

名称	描述	访问	复位值[1]	SSPn 寄存器名称&地址
CR0	控制寄存器 0。选择串行时钟速率、总线类型和数据长度。	R/W	0	SSP0CR0 – 0xE006 8000 SSP1CR0 – 0xE003 0000
CR1	控制寄存器 1。选择主机/从机和其它模式。	R/W	0	SSP0CR1 – 0xE006 8004 SSP1CR1 – 0xE003 0004
DR	数据寄存器。写满发送 FIFO 和读空接收 FIFO。	R/W	0	SSP0DR – 0xE006 8008 SSP1DR – 0xE003 0008
SR	状态寄存器。	RO	0	SSP0SR – 0xE006 800C SSP1SR – 0xE003 000C
CPSR	时钟预分频寄存器。	R/W	0	SSP0CPSR – 0xE006 8010 SSP1CPSR – 0xE003 0010
IMSC	中断屏蔽设置和清零寄存器。	R/W	0	SSP0IMSC – 0xE006 8014 SSP1IMSC – 0xE003 0014
RIS	原始中断状态寄存器。	R/W	0	SSP0RIS – 0xE006 8018 SSP1RIS – 0xE003 0018
MIS	屏蔽后的中断状态寄存器。	R/W	0	SSP0MIS – 0xE006 801C SSP1MIS – 0xE003 001C
ICR	SSPICR 中断清零寄存器。	R/W	NA	SSP0ICR – 0xE006 8020 SSP1ICR – 0xE003 0020
DMACR	DMA 控制器寄存器。	R/W	0	SSP0DMACR – 0xE006 8024 SSP1DMACR – 0xE003 0024

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

19.5.1 SSPn控制寄存器 0 (SSP0CR0-0xE006 8000, SSP1CR0 -0xE003 0000)

该寄存器控制着 SSP 控制器的基本操作。

表 19.3 SSPn 控制寄存器 0（SSP0CR0– 地址 0xE006 8000, SSP1CR0– 地址 0xE003 0000）位描述

位	符号	值	描述	复位值
3:0	DSS		数据长度选择。该字段控制着每帧传输的位数目。不支持且不使用值 0000-0010。	0000
		0011	4 位传输	
		0100	5 位传输	
		0101	6 位传输	
		0110	7 位传输	
		0111	8 位传输	
		1000	9 位传输	
		1001	10 位传输	
		1010	11 位传输	
		1011	12 位传输	
		1100	13 位传输	
		1101	14 位传输	
		1110	15 位传输	
		1111	16 位传输	
5:4	FRF		帧格式。	00
		00	SPI	
		01	TI	
		10	Microwire	
		11	不支持且不应使用这个组合。	
6	SPO		时钟输出极性。该位只用在 SPI 模式。	0
		0	SSP 控制器在帧传输的第一个时钟跳变上捕获串行数据，即跳变**远离**时钟线在帧内部的状态。	
		1	SSP 控制器在帧传输的第二个时钟跳变沿捕获串行数据，即跳变**后退**到时钟线在帧内部的状态。	
7	SPH		时钟输出相位。该位只用在 SPI 模式。	0
		0	SSP 控制器使总线时钟在每帧传输之间保持低电平。	
		1	SSP 控制器使总线时钟在每帧传输之间保持高电平。	
15:8	SCR		串行时钟速率。其值为总线上每位的预分频器输出时钟数减 1。假设 CPSDVR 为预分频器分频值，APB 时钟 PCLK 为预分频器时钟，则位频率为 PCLK / (CPSDVSR×[SCR+1])。	0x00

19.5.2 SSPn控制寄存器 1（SSP0CR1 –0xE006 8004，SSP1CR1 –0xE003 0004）

该寄存器控制着 SSP 控制器的工作方式。

表 19.4 SSPn 控制寄存器 1（SSP0CR1 –地址 0xE006 8004，SSP1CR1 – 地址 0xE003 0004）位描述

位	符号	值	描述	复位值
0	LBM		回写模式。	0
		0	正常工作模式。	
		1	串行输入脚可用作串行输出脚（MOSI 或 MISO），而不是用作串行输入脚（分别是 MISO 或 MOSI）。	
1	SSE		SSP 使能。	0
		0	SSP 控制器禁能。	
		1	SSP 控制器可与串行总线上的其它器件相互通信。在置位该位前，软件应将正确的控制信息写入其它 SSP 寄存器和中断控制器寄存器。	
2	MS		主机/从机模式。该位只能在 SSE 位为 0 时写入。	0
		0	SSP 控制器用作总线主机，驱动 SCLK、MOSI 和 SSEL 线并接收 MISO 线。	
		1	SSP 控制器用作总线从机，驱动 MISO 线和接收 SCLK、MOSI 和 SSEL 线。	
3	SOD		从机输出禁能。该位只与从机模式有关（MS=1）。如果该位为 1，则禁止 SSP 控制器驱动发送数据线（MISO）。	0
7:4	-		保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.3 SSPn数据寄存器（SSP0DR – 0xE006 8008, SSP1DR – 0xE003 0008）

软件可将要发送的数据写入该寄存器，或从该寄存器中读出接收到的数据。

表 19.5 SSPn 数据寄存器（SSP0DR – 地址 0xE006 8008, SSP1DR – 0xE003 0008）位描述

位	符号	描述	复位值
15:0	DATA	**写：**当状态寄存器的 TNF 位为 1 指示 Tx FIFO 未满时，软件可将要发送的帧的数据写入该寄存器。如果 Tx FIFO 以前为空且 SSP 控制器空闲，则立刻开始发送数据。否则，写入该寄存器的数据要等到所有数据发送（或接收）完后才能发送。如果数据长度小于 16 位，软件必须对数据进行调整后再写入该寄存器。 **读：**当状态寄存器的 RNE 位为 1 指示 Rx FIFO 不为空时，软件可读取该寄存器。软件读取该寄存器时，SSP 控制器将返回从 Rx FIFO 中读走的最后一个数据。如果数据长度小于 16 位，则该字段的数据必须进行调整，低位对齐，高位补零。	0x0000

19.5.4 SSPn状态寄存器（SSP0SR – 0xE006 800C，SSP1SR – 0xE003 000C）

该只读寄存器反映了 SSP 控制器的当前状态。

表 19.6 SSPn 状态寄存器（SSP0SR – 地址 0xE006 800C，SSP1SR – 0xE003 000C）位描述

位	符号	描述	复位值
0	TFE	发送 FIFO 空。发送 FIFO 为空时该位为 1，反之为 0。	1
1	TNF	发送 FIFO 未满。Tx FIFO 满时该位为 0，反之为 1。	1
2	RNE	接收 FIFO 不为空。接收 FIFO 为空时该位为 0，反之为 1。	0
3	RFF	接收 FIFO 满。接收 FIFO 满时该位为 1，反之为 0。	0
4	BSY	忙。SSPn 控制器空闲时该位为 0，或当前正在发送/接收一帧数据和/或 Tx FIFO 不为空时该位为 1。	0
7:5	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.5 SSPn时钟预分频寄存器（SSP0CPSR – 0xE006 8010，SSP1CPSR – 0xE003 0010）

该寄存器控制着预分频器分频 APB 时钟 PCLK 得到预分频器时钟的因子，反过来，预分频时钟被 SSPnCR0 中的 SCR 分频来确定位时钟。

表 19.7 SSPn 时钟预分频寄存器（SSP0CPSR–地址 0xE006 8010，SSP1CPSR–0xE003 8010）位描述

位	符号	描述	复位值
7:0	CPSDVSR	这是一个 2～254 中的一个偶数。它是 PCLK 的分频因子，PCLK 通过分频后得到预分频器输出时钟。位 0 读出时总是为 0。	0

要点：必须适当地初始化 SSPnCPSR 值，否则 SSP 控制器将不能正确发送数据。如果 SSP 在主控器模式下操作则 $CPSDVSR_{min}$＝2，而如果 SSP 在从机模式下操作则 $CPSDVSR_{min}$＝12。

19.5.6 SSPn中断屏蔽设置/清除寄存器（SSP0IMSC – 0xE006 8014，SSP1IMSC – 0xE003 0014）

该寄存器控制着 SSP 控制器 4 个中断条件的使能。注意：ARM 使用“masked”一词时的理解与经典计算机术语相反，计算机术语中“masked”意思 是“禁 能”。而 ARM 中“masked”的意思是“使能“。为了防止混淆，ARM 文档中通常不用“masked”一词。

表 19.8 SSPn 中断屏蔽设置/清除寄存器(SSP0IMSC – 地址 0xE006 8014，SSP1IMSC – 0xE003 0014) 位描述

位	符号	描述	复位值
0	RORIM	当接收溢出时软件置位该位来使能中断，即当 Rx FIFO 满时又完成另一个帧的接收时该位置位。ARM 特别指出，接收溢出时新的数据帧会将前面的数据帧覆盖。	0
1	RTIM	当出现接收超时条件时，软件置位该位来使能中断。当 Rx FIFO 不为空且在“超时周期”中没有被读出时，产生接收超时。	0
2	RXIM	当 Rx FIFO 至少有一半为满时，软件置位该位来使能中断。	0
3	TXIM	当 Tx FIFO 至少有一半为空时，软件置位该位来使能中断。	0
7:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.7 SSPn原始中断状态寄存器（SSP0RIS – 0xE006 8018，SSP1RIS – 0xE003 0018）

当一个中断条件出现时，该只读寄存器中对应的位置位，与中断是否通过 SSPnIMSC 使能无关。

表 19.9 SSPn 原始中断状态寄存器（SSP0RIS –地址 0xE006 8018，SSP1RIS –0xE003 0018）位描述

位	符号	描述	复位值
0	RORRIS	当 Rx FIFO 满时又接收到另一帧数据时该位置位。ARM 特别指出，此时接收到的新数据帧会将前面的数据帧覆盖。	0
1	RTRIS	如果 Rx FIFO 不为空，且在“超时周期”中没有被读出时该位置位。	0
2	RXRIS	当 Rx FIFO 至少一半为满时该位置位。	0
3	TXRIS	当 Tx FIFO 至少一半为空时该位置位。	1
7:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.8 SSPn屏蔽后的中断状态寄存器（SSP0MIS – 0xE006 801C，SSP1MIS – 0xE003 001C）

当一个中断条件出现且相应的中断在 SSPnIMSC 中被使能时，对应在该只读寄存器的位置位。当产生 SSP 中断时，中断服务程序可通过读该寄存器来判断中断源。

表 19.10 SSPn 屏蔽后的中断状态寄存器（SSPnMIS – 地址 0xE006 801C, SSP1MIS – 0xE003 001C）位描述

位	符号	描述	复位值
0	RORMIS	当 Rx FIFO 满时又接收到另一帧数据，并且中断被使能时该位置位。	0
1	RTMIS	如果 Rx FIFO 不为空且在“超时周期”中未被读出，则该位置位，并且该中断被使能。	0
2	RXMIS	当 Rx FIFO 至少一半为满时该位置位，并且该中断被使能。	0
3	TXMIS	当 Tx FIFO 至少一半为空时该位置位，并且该中断被使能。	0
7:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.9 SSPn中断清除寄存器（SSP0ICR – 0xE006 8020，SSP1ICR – 0xE003 0020）

软件可通过向该只写寄存器写入 1 或多个 1 来清除 SSP 控制器的相关中断。注意：另外 2 个中断条件可通过写或读相应的 FIFO 来清除，或通过清除 SSPnIMSC 对应的位来禁能。

表 19.11 SSPn 中断清除寄存器（SSP0ICR – 地址 0xE006 8020，SSP1ICR – 0xE003 0020）位描述

位	名称	描述	复位值
0	RORIC	向该位写 1 来清除“Rx FIFO 满时还接收到帧”的中断。	NA
1	RTIC	向该位写 1 来清除“Rx FIFO 不为空且在超时周期中还没有被读出”的中断。	NA
7:2	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

19.5.10 SSPn DMA控制寄存器（SSP0DMACR – 0xE006 8024, SSP1DMACR – 0xE003 0024）

SSPnDMACR寄存器是DMA控制寄存器。它是读/写寄存器。表 19.12所示为SSPnDMACR寄存器的位分配。

表 19.12 SSPn DMA 控制寄存器（SSP0DMACR – 地址 0xE006 8024, SSP1DMACR – 0xE003 0024）位描述

位	名称	描述	复位值
0	接收 DMA 使能（RXDMAE）	当该位设为 1 时，接收 FIFO 的 DMA 被使能，否则接收 DMA 被禁能。	0
1	发送 DMA 使能（TXDMAE）	当该位设为 1 时，发送 FIFO 的 DMA 被使能，否则发送 DMA 被禁能。	0
15:2	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

第20章 SD/MMC卡接口

20.1 概述

加密的数字和多媒体卡接口（MCI）是先进的外设总线（APB）系统总线与多媒体和/或加密的数字存储卡之间的接口。它由两部分组成：

- MCI 适配器块，提供与加密数字/多媒体存储卡相关的所有功能，如时钟发生单元、电源管理控制、命令和数据传输。
- APB 接口，访问 MCI 适配器寄存器，产生中断和 DMA 请求信号。

20.2 MCI的特性

MCI 具有以下特性：

- 遵循多媒体卡规范 v2.11；
- 遵循加密的数字存储卡物理层规范 v0.96；
- 用作多媒体卡总线主机或加密的数字存储卡总线主机。它可以与多个多媒体卡相连，也可以和单个加密的数字存储卡相连；
- 通过通用 DMA 器件支持 DMA 传输。

20.3 SD/MMC卡接口管脚描述

表 20.1 SD/MMC 卡接口管脚描述

管脚名称	类型	描述
MCICLK	输出	时钟输出。
MCICMD	输入	命令输入/输出。
MCIDAT[3:0]	输出	数据线。只有 MCIDAT [0]用于多媒体卡。
MCIPWR	输出	外部 SD/MMC 电源的电源使能。

某些情况下该接口还需要一个额外的被称作电源控制线的 MCIPWR 信号，该功能可由任何一个 GPIO 信号产生。

20.4 功能概述

MCI可用作多媒体卡总线主机（见20.4.1 节“多媒体卡”）或加密数字存储卡总线主机（见20.4.2 节“加密数字存储卡”）。能够和多达 4 个左右的多媒体卡（受I/O管脚规范和电路板负载能力的限制）或单个加密的数字存储卡连接。

20.4.1 多媒体卡

图 20.1所示为多媒体卡系统。

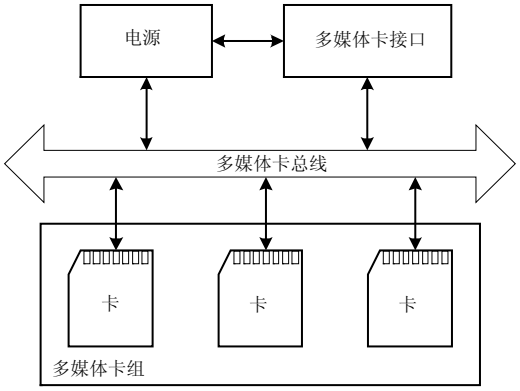


图 20.1 多媒体卡系统

根据功能将多媒体卡分成 3 类：

- 只读存储（ROM）卡，包含预编程的数据；
- 读/写（R/W）卡，用于大容量存储器；
- 输入/输出（I/O）卡，用于通信。

多媒体卡系统使用以下 3 条信号线来传输命令和数据：

- CLK：每经过一个时钟周期，每条命令和数据线就传输 1 位。多媒体卡的时钟频率为 0MHz～20MHz（用于多媒体卡），或为 0MHz～25MHz（用于加密数字存储卡）。
- CMD：初始化卡和传输命令的双向命令通道。CMD 有 2 种工作模式：
  - 开漏式，供初始化使用
  - 推挽式，供命令传输使用
- DAT：双向数据通道，在推挽模式下操作。

20.4.2 加密数字存储卡

图 20.2显示了加密的数字存储卡的连接。

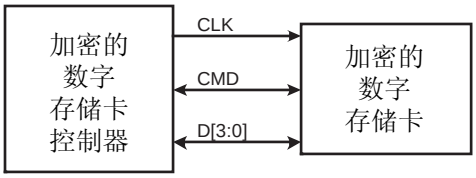


图 20.2 加密数字存储卡的连接

20.4.2.1 加密数字存储卡的总线信号

加密数字存储卡总线使用以下信号：

- CLK：主机到卡的时钟信号
- CMD：双向的命令/响应信号
- DAT[3:0]：双向的数据信号

20.4.3 MCI适配器

图 20.3给出了MCI适配器的简化方框图。

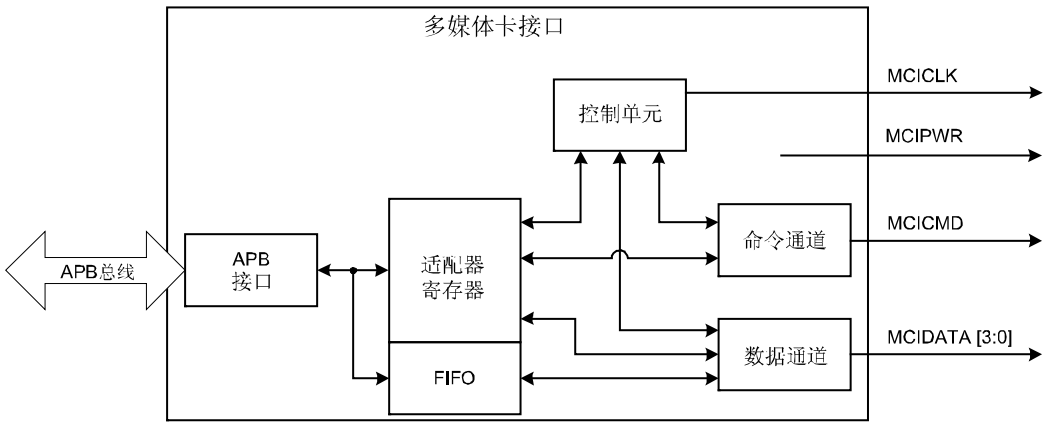


图 20.3 MCI 适配器

MCI 适配器是多媒体卡或加密数字存储卡的总线主机，它提供了一个到多媒体卡组（stack）或加密数字存储卡的接口。它包括 5 个子单元：

- 适配器寄存器块
- 控制单元
- 命令通道
- 数据通道
- 数据 FIFO

20.4.3.1 适配器寄存器块

适配器寄存器块包含所有系统寄存器。该寄存器块还产生用来清零多媒体卡的静态标志的信号。该清零信号在向 MCIClear 寄存器的相应位写 1 时产生。

20.4.3.2 控制单元

控制单元包括功率管理功能和存储卡时钟的时钟分频器。

含 3 种功率状态：

- 断电
- 上电
- 通电

功率管理逻辑控制一个外部电源单元，并在断电或上电阶段禁止卡总线的输出信号。上电是断电和通电之间的过渡阶段，并允许外部电源达到卡总线的工作电压。设备驱动器可以保证 PrimeCell MCI 保持上电状态，直至外部电源电压达到工作电压。

时钟管理逻辑产生 MCICLK 信号，并对它进行控制。MCICLK 输出可以采用时钟分频或时钟旁路模式。时钟输出在以下情况无效：

- 复位后
- 断电或上电过程

- 如果节电模式被使能，并且卡总线处于 IDLE（空闲）状态（命令和数据通道子单元进入 IDLE 状态之后的 8 个时钟周期）。

20.4.3.3 命令通道

命令通道子单元向卡发送命令，并从卡接收响应。

20.4.3.4 命令通道状态机

写命令寄存器且使能位置位时开始传输命令。当命令已经被发送时，如果不要求响应，命令通道状态机（CPSM）将置位状态标志并进入IDLE状态。如果要求响应，那么就等待响应（见图 20.4）。接收到响应时，接收到的CRC代码与内部产生的代码进行比较，并置位相应的状态标志。

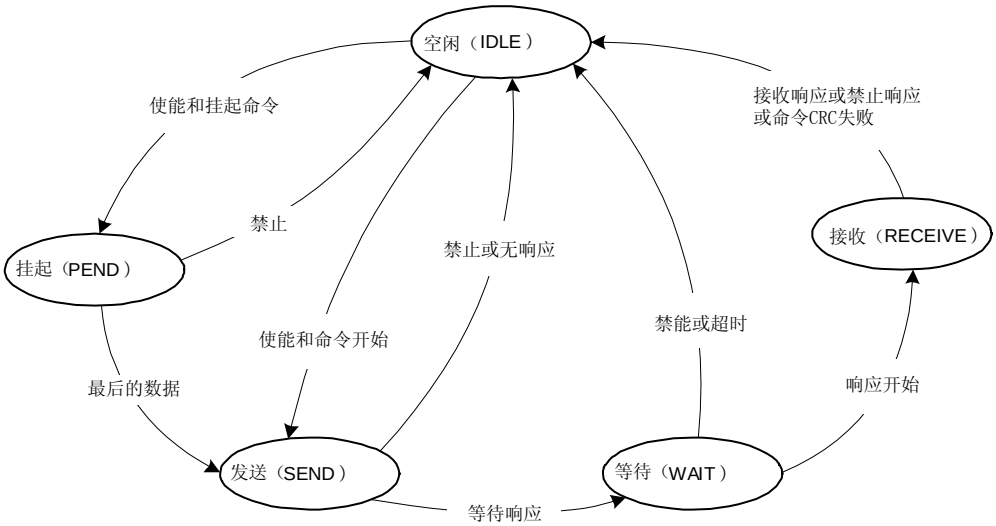


图 20.4 命令通道状态机

进入 WAIT 状态时命令定时器开始运行。如果在 CPSM 转至 RECEIVE（接收）状态前就达到了超时时间，那么置位超时标志，并进入 IDLE 状态。

注：超时周期固定为 64 个 MCICLK 时钟周期。

如果中断位在命令寄存器中被置位，那么定时器被禁能，CPSM 等待来自卡的中断请求。如果挂起位在命令寄存器中被置位，那么 CPSM 进入 PEND（挂起）状态，等待来自数据通道子单元的 CmdPend 信号。当检测到 CmdPend 信号时，CPSM 变为 SEND（发送）状态。这样，数据计数器就可以触发停止命令发送。

图 20.5显示了MCI命令传输的过程。

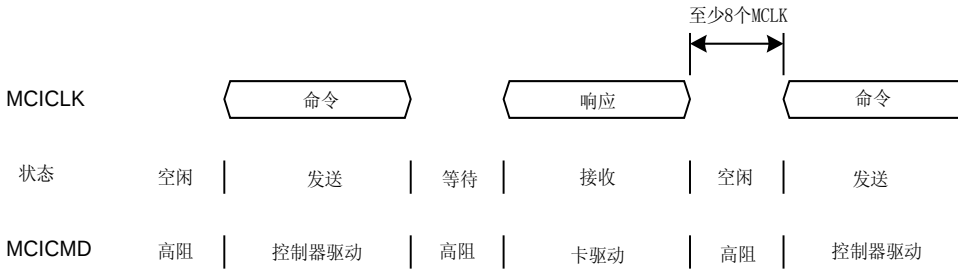


图 20.5 MCI 命令传输

20.4.3.5 命令格式

命令通道工作在半双工模式，所以命令和响应都可以被发送或被接收。如果CPSM不处于SEND（发送）状态，那么MCICMD输出为高阻（HI-Z）状态，如图 20.5所示。MCICMD的数据与MCICLK上升沿同步。所有命令的长度都固定为 48 位。表 20.2对命令格式进行了描述。

表 20.2 命令格式

位	宽度	值	描述
0	1	1	结束位
7:1	7	-	CRC7
39:8	32	-	参数
45:40	6	-	命令索引
46	1	1	发送位
47	1	0	起始位

MCI 适配器支持两种响应类型。这两种响应类型都使用 CRC 错误校验：

- 48 位短响应（见表 20.3）。
- 136 位长响应（见表 20.4）。

注：如果响应不含 CRC（CMD1 响应 ，那么设备驱动器必须忽略 CRC 失败的状态。

表 20.3 简单的响应格式

位	宽度	值	描述
0	1	1	结束位
7:1	7	-	CRC7（或 1111111）
39:8	32	-	参数
45:40	6	-	命令索引
46	1	0	发送位
47	1	0	起始位

表 20.4 长响应格式

位	宽度	值	描述
0	1	1	结束位
127:1	127	-	CID 或 CSD（包括内部 CRC7）
133:128	6	111111	保留
134	1	1	发送位
135	1	0	起始位

命令寄存器包含命令索引（发送给卡的 6 个位）和命令类型。两者决定了命令是否需要响应，以及响应是 48 位还是 136 位长（详见20.5.5 节“命令寄存器（MCICommand –0xE008 C00C）”）。命令通道处理表 20.5所示的状态标志（详见20.5.12 节“状态寄存器（MCIStatus – 0xE008 C034）”）。

表 20.5 命令通道状态标志

标志	描述
CmdRespEnd	响应 CRC 成功时置位。
CmdCrcFail	响应 CRC 失败时置位。
CmdSent	发送命令（不需要响应）时置位。
CmdTimeOut	响应超时。
CmdActive	命令传输正在进行。

CRC 发生器为 CRC 代码前面的所有位计算 CRC 检验和。这些位包括起始位、发送位、命令索引，以及命令参数（或卡状态）。CRC 发生器还为长响应格式下的 CID 或 CSD 的前 120 个位计算 CRC 检验和。注：计算 CRC 时不使用起始位、发送位和 6 个保留位。

CRC 检验和是一个 7 位的值：

CRC[6:0] = 余数 [(M(x) × x_7) / G(x)]

G(x) = x_7 + x_3 + 1

M(x) = (起始位) × x_{39} + ... + (CRC 前面的最后一位) × x_0 ,或者

M(x) = (起始位) × x_{119} + ... + (CRC 前面的最后一位) × x_0

20.4.3.6 数据通道

通过使用时钟控制寄存器用户可以对卡的数据总线宽度进行编程。如果使能宽总线模式，那么每个时钟周期在 4 个数据信号（MCIDAT[3:0]）上传输 4 位数据。如果没有使能宽总线模式，那么每个时钟周期只在 MCIDAT0 上传输 1 位数据。

根据具体的传输方向（发送还是接收），数据通道状态机（DPSM）在使能时可以相应地转至 WAIT_S 或 WAIT_R 状态。

- 发送：DPSM 转至 WAIT_S 状态。如果发送 FIFO 中存在数据，那么 DPSM 转至 SEND（发送）状态，且数据通道子单元开始向卡发送数据。
- 接收：DPSM 转至 WAIT_R 状态并等待起始位。当接收到起始位时，DPSM 转至 RECEIVE 状态，且数据通道子单元开始从卡接收数据。

20.4.3.7 数据通道状态机

DPSM在MCICLK频率下操作。卡总线信号上的数据与MCICLK的上升沿同步。DPSM有 6 种状态，如图 20.6所示。

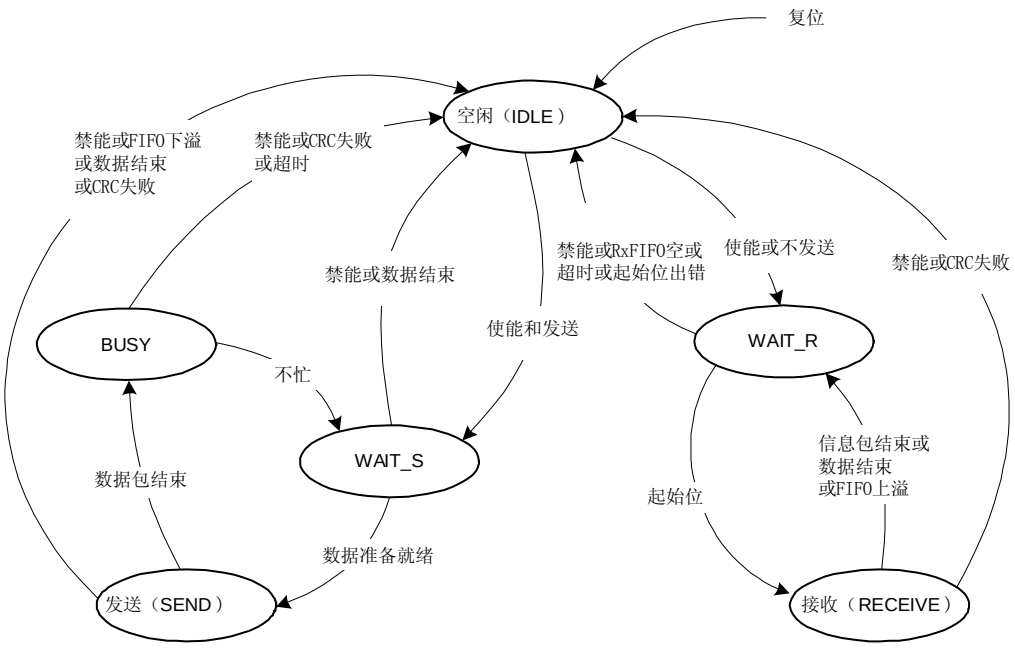


图 20.6 数据通道状态机

- IDLE（空闲）：数据通道无效，且 MCIDAT[3:0]输出都为高阻（HI-Z）状态。在写数据控制寄存器且使能位被置位时，DPSM 向数据计数器加载新的值，并根据具体的数据方向位转至 WAIT_S 或 WAIT_R 状态。

  WAIT_R:如果数据计数器等于 0，那么 DPSM 在接收 FIFO 为空时转至 IDLE（空闲）状态。如果数据计数器不等于 0，那么 DPSM 等待 MCIDAT 的起始位。

如果 DPSM 在超时前接收到起始位，那么它将转至 RECEIVE（接收）状态，并加载数据块计数器。如果 DPSM 在检测到起始位前就到达超时，或发生起始位错误，那么它将转至 IDLE 状态，并置位超时状态标志。

- RECEIVE：将接收到的卡的串行数据按字节打包，并写入数据 FIFO。根据数据控制寄存器的传输模式位的值，数据传输模式可以是块或流模式：

  －在块模式中，数据块计数器为 0 时 DPSM 等待，直至它接收到 CRC 代码。如果接收到的代码和内部产生的 CRC 代码相匹配，那么 DPSM 就转至 WAIT_R 状态。否则，CRC 失败状态标志置位且 DPSM 转至 IDLE 状态。

  －在流模式中，数据计数器不为 0 时 DPSM 接收数据。当计数器为 0 时，移位寄存器中剩余的数据被写入数据 FIFO，且 DPSM 转至 WAIT_R 状态。

如果发生 FIFO 溢出错误，那么 DPSM 会置位 FIFO 出错标志并转至 WAIT_R 状态。

- WAIT_S：如果数据计数器为 0，则 DPSM 将转至 IDLE 状态。否则，它会等待直至数据 FIFO 空标志无效，然后转至 SEND（发送）状态。

注：为了符合 Nwr 时序约束，DPSM 至少在两个时钟周期内保持在 WAIT_S 状态。

- SEND：DPSM 开始向卡发送数据。根据数据控制寄存器的传输模式位的具体值，数据传输模式可以是块或流模式：

  －在块模式中，数据块计数器为 0 时，DPSM 发送内部产生的 CRC 代码和结束位，并转至 BUSY 状态。

－在流模式中，当使能位为 1（HIGH）且数据计数器不为 0 时，DPSM 向卡发送数据。然后转至 IDLE 状态。

如果发生 FIFO 下溢错误，则 DPSM 将置位 FIFO 出错标志并转至 IDLE 状态。

- BUSY（忙）：DPSM 等待 CRC 状态标志：

  －在没有接收到正确的 CRC 状态时，它会转至 IDLE 状态并置位 CRC 失败状态标志。

  －在接收到正确的 CRC 状态时，如果 MCIDAT0 不为低（卡不忙），则它会转至 WAIT_S 状态。

如果在 DPSM 处于 忙(BUSY)状态时发生超时，那么它会置位数据超时标志并转至 IDLE 状态。

数据定时器在 DPSM 处于 WAIT_R 或 BUSY 状态时使能，并产生数据超时错误：

- 发送数据时，如果 DPSM 处于 BUSY 状态的时间超过编程设定的超时周期，那么会发生超时。
- 接收数据时，如果数据的结束不为真（TRUE）或者如果 DPSM 处于 WAIT_R 状态的时间超过编程设定的超时周期，那么会发生超时。

20.4.3.8 数据计数器

数据计数器有两个功能：

- 在数据计数器到达 0 时停止数据传输。这是数据条件的结束。
- 开始传输一个挂起命令（见图 20.7）。该功能用来为流数据传输发送停止命令。

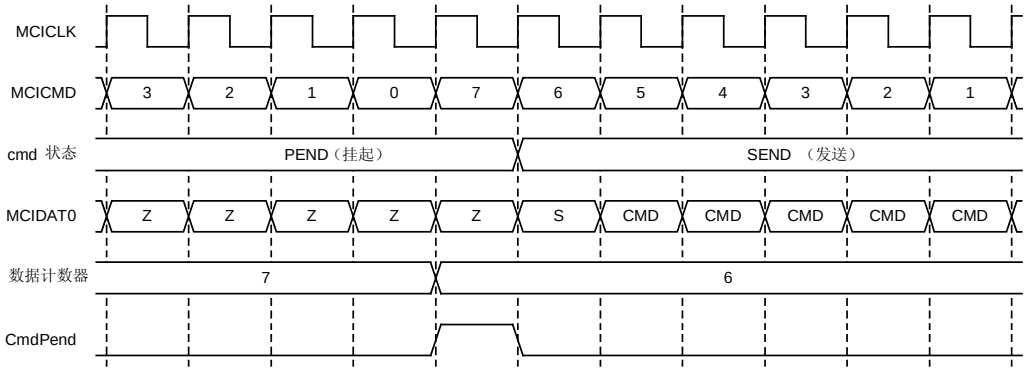


图 20.7 挂起命令开始

数据块计数器决定数据块的结束。如果计数器为 0，则数据结束条件就为真（TRUE）（详见20.5.10 节“数据控制寄存器（MCIDataCtrl – 0xE008 C02C）”）。

20.4.3.9 总线模式

在宽总线模式，所有 4 个数据信号（MCIDAT[3:0]）都用于传输数据，并且可以为每个数据信号单独计算 CRC 代码。在向卡发送数据块时，只有 MCIDAT0 用于 CRC 令牌和忙信号发送。起始位必须同时（在相同的时间周期内）在 4 个数据信号上被发送。接收数据时，如果在同一时钟沿没有检测到任何数据信号的起始位，那么 DPSM 将会置位起始位出错标志并转至 IDLE 状态。

数据通道也工作在半双工模式下，在该模式下向卡发送数据或者从卡接收数据。不传输

数据时 MCIDAT[3:0]处于高阻状态。

这些信号上的数据与时钟周期的上升沿同步。

如果选择了标准总线模式，MCIDAT[3:1]输出就总是高阻态，并且在发送数据时只有 MCIDAT0 输出被驱动为低。

设计备注：如果选择了宽总线模式，那么 nMCIDAT0EN 和 nMCIDATEN 输出就同时被驱动为低。否则，MCIDAT[3:1]输出就总是高阻态（nMCIDATEN 被驱动为高），并且在发送数据时只有 MCIDAT0 输出被驱动为低。

20.4.3.10 CRC令牌状态

CRC令牌状态紧跟在每个写数据块之后，并确定卡是否正确地接收到数据块。当接收到令牌时，卡通过将MCIDAT0 驱动为低电平来使忙信号有效。表 20.6对CRC令牌的状态值进行了描述。

表 20.6 CRC 令牌状态

令牌	描述
010	卡接收到正确的数据块。
101	卡检测到 CRC 错误。

20.4.3.11 状态标志

表 20.7列出了数据通道状态标志（详见20.5.12 节“状态寄存器（MCIStatus – 0xE008C034）”）。

表 20.7 数据通道状态标志

标志	描述
TxFifoFull	发送 FIFO 满
TxFifoEmpty	发送 FIFO 空
TxFifoHalfEmpty	发送 FIFO 半满
TxDataAvlbl	发送 FIFO 中的数据可用
TxUnderrun	发送 FIFO 下溢出错
RxFifoFull	接收 FIFO 满
RxFifoEmpty	接收 FIFO 空
RxFifoHalfFull	接收 FIFO 半满
RxDataAvlbl	接收 FIFO 中的数据可用
RxOverrun	接收 FIFO 上溢出错
DataBlockEnd	数据块发送/接收
StartBitErr	宽总线模式下，在所有数据信号上都没有检测到起始位
DataCrcFail	数据包 CRC 失败
DataEnd	数据结束（数据计数器为 0）
DataTimeOut	数据超时
TxActive	数据发送正在进行
RxActive	数据接收正在进行

20.4.3.12 CRC发生器

CRC 发生器只为单个块中的数据位计算 CRC 校验和，在数据流模式中将其旁路。校验和是一个 16 位的值：

CRC[15:0] = 余数 [(M(x) × x^{15}) / G(x)]

G(x) = x^{16} + x^{12} + x^5 + 1

M(x)－(第一个数据位) × x^n + ... + (最后一个数据位) × x^0

20.4.3.13 数据FIFO

数据 FIFO（先进先出）子单元是一个带有发送和接收逻辑的数据缓冲器。

FIFO 包含一个 32 位宽、16 字深的数据缓冲区以及发送和接收逻辑。由于数据 FIFO 工作在 APB 时钟域（PCLK），所以 MCI 时钟域（MCLK）子单元的所有信号都是重新同步的。

根据 TxActive 和 RxActive 的值，可以禁能 FIFO、使能发送或使能接收。TxActive 和 RxActive 都由数据通道子单元驱动，两者互斥。

- 发送FIFO是指TxActive有效时的发送逻辑和数据缓冲区（见20.4.3.14 节“发送FIFO”）。
- 接收FIFO是指RxActive有效时的接收逻辑和数据缓冲区（见20.4.3.15 节“接收FIFO”）。

20.4.3.14 发送FIFO

一旦 MCI 发送被使能，数据就通过 APB 接口写入发送 FIFO。

发送FIFO可通过 16 个连续的地址访问（见20.5.16 节“数据FIFO寄存器（MCIFIFO – 0xE008 C080 到 0xE008 C0BC）”）。发送FIFO含有一个数据输出寄存器。该寄存器保存着读指针指向的数据字。当数据通道子单元已经加载了移位寄存器时，它使读指针递增，并驱动输出新数据。

如果发送FIFO被禁能，那么所有状态标志都变成无效。数据通道子单元在发送数据时使TxActive有效。表 20.8列出了发送FIFO的状态标志。

表 20.8 发送 FIFO 状态标志

标志	描述
TxFifoFull	当 16 个发送 FIFO 字都包含有效数据时置位。
TxFifoEmpty	当发送 FIFO 不含有效数据时置位。
TxHalfEmpty	当 8 个或 8 个以上的发送 FIFO 字都为空时置位。该标志可以用作 DMA 请求。
TxDataAvlbl	当发送 FIFO 包含有效数据时置位。该标志是 TxFifoEmpty 标志的取反。
TxUnderrun	当发生下溢错误时置位。该标志通过写 MCIClear 寄存器来清零。

20.4.3.15 接收FIFO

当数据通道子单元接收到一个字的数据时，它会驱动写数据总线上的数据，并将写使能信号变为有效。写使能信号与 PCLK 域同步。写指针在每次写操作完成后都加 1，接收 FIFO 控制逻辑使 RxWrDone 有效，然后又将写使能信号变为无效。

在读取数据一方，读指针当前值指向的 FIFO 字的内容在读数据总线上被驱动。读指针在 APB 总线接口使 RxRdPrtInc 有效时加 1。

接收FIFO可通过 16 个连续的地址访问（见20.5.16 节“数据FIFO寄存器（MCIFIFO – 0xE008 C080 到 0xE008 C0BC）”）。

如果接收FIFO被禁能，那么所有状态标志都变成无效，并且读指针和写指针都复位。数据通道子单元在接收数据时将RxActive变成有效。表 20.9列出了接收FIFO的状态标志。

表 20.9 接收 FIFO 状态标志

标志	描述
RxFifoFull	在 16 个接收 FIFO 字都包含有效数据时置位。
RxFifoEmpty	当接收 FIFO 不包含有效数据时置位。
RxHalfFull	当 8 个或 8 个以上的接收 FIFO 字包含有效数据时置位。该标志可以用作 DMA 请求。
RxDataAvlbl	当接收 FIFO 不为空时置位。该标志是 RxFifoEmpty 标志的取反。
RxOverrun	当出现 FIFO 下溢错误时置位。该标志通过写 MCIClear 寄存器来清零。

20.4.3.16 APB接口

APB 接口产生中断和 DMA 请求，并访问 MCI 适配器寄存器和数据 FIFO。它由数据通道、寄存器译码器，以及中断/DMA 逻辑组成。DMA 由通用 DMA 控制器来控制，详见“通用 DMA 控制器”。

20.4.3.17 中断逻辑

中断逻辑产生 1 个中断请求信号，该信号仅当至少有一个选择的状态标志置位时有效。用户可以通过屏蔽寄存器来选择产生中断的条件。如果相应的屏蔽标志置位，那么状态标志就产生中断请求。

20.5 寄存器描述

本小节描述了 MCI 寄存器并对编程作了详细介绍。

20.5.1 MCI寄存器汇总

MCI寄存器如表 20.10所示。

表 20.10 MCI 寄存器映射

名称	描述	访问	宽度	复位值[1]	地址
MCIPower	电源控制寄存器	R/W	8	0x00	0xE008 C000
MCIClock	时钟控制寄存器	R/W	12	0x000	0xE008 C004
MCIArgument	参数（argument）寄存器	R/W	32	0x00000000	0xE008 C008
MMCCommand	命令寄存器	R/W	11	0x000	0xE008 C00C
MCIRespCmd	响应命令寄存器	RO	6	0x00	0xE008 C010
MCIResponse0	响应寄存器	RO	32	0x00000000	0xE008 C014
MCIResponse1	响应寄存器	RO	32	0x00000000	0xE008 C018
MCIResponse2	响应寄存器	RO	32	0x00000000	0xE008 C01C
MCIResponse3	响应寄存器	RO	31	0x00000000	0xE008 C020
MCIDataTimer	数据定时器	R/W	32	0x00000000	0xE008 C024
MCIDataLength	数据控制寄存器	R/W	16	0x0000	0xE008 C028

续表 20.10

名称	描述	访问	宽度	复位值[1]	地址
MCIDataCtrl	数据控制寄存器	R/W	8	0x00	0xE008 C02C
MCIDataCnt	数据计数器	RO	16	0x0000	0xE008 C030
MCIStatus	状态寄存器	RO	22	0x000000	0xE008 C034
MCIClear	清零寄存器	WO	11	-	0xE008 C038
MCIMask0	中断 0 屏蔽寄存器	R/W	22	0x000000	0xE008 C03C
MCIMask1	中断 1 屏蔽寄存器	R/W	22	0x000000	0xE008 C040
MCIFifoCnt	FIFO 计数器	RO	15	0x0000	0xE008 C048
MCIFIFO	数据 FIFO 寄存器	R/W	32	0x00000000	0xE008 C080 ~ 0xE008 C0BC

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

20.5.2 电源控制寄存器（MCIPower – 0xE008 C000）

MCIPower寄存器控制外部电源。电源可以打开也可以关闭，还可以调节输出电压。表 20.11所示为MCIPower寄存器的位分配。

MCIPWR（电源使能）管脚的有效电平可通过SCS寄存器的位 3（详见3.7.1 节“系统控制和状态寄存器（SCS – 0xE01F C1A0）”）来选择。

表 20.11 电源控制寄存器（MCIPower – 地址 0xE008 C000）位描述

位	符号	值	描述	复位值
1:0	Ctrl	00 01 10 11	电源关闭。 保留 上电 通电	00
5:2	-		保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA
6	OpenDrain		MCICMD 输出控制。	0
7	Rod		Rod 控制。	0
31:8	-		保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

当外部电源打开时，软件首先进入上电阶段，并且在转移到通电阶段前等待直至电源输出稳定。在上电过程中，MCIPWR 管脚被设为高电平（HIGH）。卡总线的输出在上电和通电过程中都被禁能。

注：数据写操作后，在 3 个 MCLK 时钟周期加 2 个 PCLK 时钟周期内都不能将数据写入该寄存器。

20.5.3 时钟控制寄存器（MCIClock – 0xE008 C004）

MCIClock寄存器控制MCICLK输出。表 20.12所示为时钟控制寄存器的位分配。

表 20.12 时钟控制寄存器（MCIClock – 地址 0xE008 C004）位描述

位	符号	值	描述	复位值
7:0	ClkDiv		MCI 总线时钟周期： MCICLK 频率＝MCLK / [2×（ClkDiv+1）]。	0x00
8	Enable	 0 1	使能 MCI 总线时钟。 时钟禁能。 时钟使能。	0
9	PwrSave	 0 1	当总线空闲时禁能 MCI 时钟输出： 总是使能。 当总线有效时时钟使能。	0
10	Bypass	 0 1	使能时钟分频逻辑的旁路： 禁能旁路。 使能旁路。MCLK 驱动到卡总线输出（MCICLK）。	0
11	WideBus	 0 1	使能宽总线模式： 标准总线模式（只有 MCIDAT0 使用）。 宽总线模式（MCIDAT3:0 使用）。	0
31:12	-		保留，用户软件不应该向该保留位写 1。从保留位读取的值未定义。	NA

当 MCI 处于识别模式时，MCICLK 频率必须小于 400kHz。在将相应的卡地址分配给所有卡时，时钟频率可以变成最大的卡总线频率。

注：写入数据后，在 3 个 MCLK 时钟周期加 2 个 PCLK 时钟周期内都不能将数据写入该寄存器。

20.5.4 参数寄存器（MCIArgument – 0xE008 C008）

MCIArgument寄存器包含一个 32 位的命令参数，这个参数作为命令消息的一部分发送给卡。表 20.13对MCIArgument寄存器进行了描述。

表 20.13 参数寄存器（MCIArgument –地址 0xE008 C008）位描述

位	符号	描述	复位值
31:0	CmdArg	命令参数	0x0000 0000

如果一个命令包含一个参数，那么在写命令到命令寄存器前，必须将该参数加载到参数寄存器。

20.5.5 命令寄存器（MCICommand –0xE008 C00C）

MCICommand 寄存器包含命令索引和命令类型位：

- 命令索引作为命令消息的一部分发送给卡。
- 命令类型位对命令通道状态机（CPSM）进行控制。向使能位写 1 来启动命令发送操作，而清零该位会禁能 CPSM。

表 20.14对MCICommand寄存器进行了描述。

表 20.14 命令寄存器（MCICommand –地址 0xE008 C00C）位描述

位	符号	描述	复位值
5:0	CmdIndex	命令索引。	0
6	Response	如果置位，CPSM 等待一个响应。	0
7	LongRsp	如果置位，CPSM 接收一个 136 位的长响应。	0
8	Interrupt	如果置位，CPSM 禁止命令定时器并等待中断请求。	0
9	Pending	如果置位，CPSM 在开始发送命令前等待 CmdPend。	0
10	Enable	如果置位，CPSM 被使能。	0
31:11	-	保留，用户软件不应该向保留位写 1，从保留位读取的值未定义。	NA

注：写入数据后，在 3 个 MCLK 时钟周期加 2 个 PCLK 时钟周期内都不能将数据写入该寄存器。

表 20.15对响应类型进行了描述。

表 20.15 命令响应类型

响应	长响应	描述
0	0	无响应，等待 CmdSent 标志。
0	1	无响应，等待 CmdSent 标志。
1	0	短响应，等待 CmdRespEnd 或 CmdCrcFail 标志。
1	1	长响应，等待 CmdRespEnd 或 CmdCrcFail 标志。

20.5.6 命令响应寄存器（MCIRespCommand – 0xE008 C010）

MCIRespCommand寄存器含有接收到的最后一个命令响应的命令索引字段。表 20.16所示为MCIRespCommand寄存器的位分配。

表 20.16 命令响应寄存器（MCIRespCommand –0xE008 C010）位描述

位	符号	描述	复位值
5:0	RespCmd	响应命令索引。	0x00
31:6	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

如果命令响应传输不包含命令索引字段（长响应），那么尽管 RespCmd 必须包含 111111（响应的保留字段的值），但其值还是不可知的。

20.5.7 响应寄存器（MCIResponse0-3 – 0xE008 C014，0xE008 C018，0xE008 C01C，0xE008 C020）

MCIResponse0-3 寄存器包含卡状态，它是已接收响应的一部分。表 20.17所示为 MCIResponse0-3 寄存器的位分配。

表 20.17 响应寄存器（MCIResponse0-3 - 地址 0xE008 C014，0xE008 C018，0xE008 C01C，0xE008 C020）位描述

位	符号	描述	复位值
31:0	Status	卡状态	0x0000 0000

卡状态可以是 32 位或 127 位，取决于响应的类型（见表 21.18）。

表 20.18 响应寄存器类型

描述	短响应	长响应
MCIResponse0	卡状态[31:0]	卡状态[127:96]
MCIResponse1	未用	卡状态[95:64]
MCIResponse2	未用	卡状态[63:32]
MCIResponse3	未用	卡状态[31:1]

首先接收卡状态的最高位。MCIResponse3 寄存器的最低位总是为 0。

20.5.8 数据定时器寄存器（MCIDataTimer – 0xE008 C024）

在卡总线时钟周期，MCIDataTimer寄存器包含数据超时周期。表 20.19对MCIDataTimer寄存器的位分配进行了描述。

表 20.19 数据定时器寄存器（MCIDataTimer –0xE008 C024）位描述

位	符号	描述	复位值
31:0	DataTime	数据超时周期	0x0000 0000

计数器从数据定时器寄存器加载值，并在数据通道状态机（DPSM）进入 WAIT_R 或 BUSY 状态时开始减 1。如果定时器在 DPSM 处于其中一种状态时计数值到达 0，那么超时状态标志置位。

在传输数据时，必须先写数据定时器寄存器和数据长度寄存器，再写数据控制寄存器。

20.5.9 数据长度寄存器（MCIDataLength – 0xE008 C028）

MCIDataLength寄存器包含传输的数据字节数。当数据传输开始时该值被载入数据计数器。表 20.20对MCIDataLength寄存器的位分配进行了描述。

表 20.20 数据长度寄存器（MCIDataLength – 地址 0xE008 C028）位描述

位	符号	描述	复位值
15:0	DataLength	数据长度值。	0x0000
31:6	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

对于块数据传输，数据长度寄存器的值必须是块大小的倍数（见20.5.10 节“数据控制寄存器（MCIDataCtrl – 0xE008 C02C）”）。

为了启动数据传输，在写数据控制器寄存器之前必须先写数据定时器寄存器和数据长度寄存器。

20.5.10 数据控制寄存器（MCIDataCtrl – 0xE008 C02C）

MCIDataCtrl寄存器控制DPSM。表 20.21对MCIDataCtrl寄存器的位分配进行了描述。

表 20.21 数据控制寄存器（MCIDataCtrl –地址 0xE008 C02C）位描述

位	符号	值	描述	复位值
0	Enable		数据传输使能。	0
1	Direction		数据传输方向：	0
		0	从控制器传输到卡。	
		1	从卡传输到控制器。	
2	Mode		数据传输模式：	0
		0	块数据传输。	
		1	流数据传输。	
3	DMAEnable		使能 DMA：	0
		0	DMA 禁能。	
		1	DMA 使能。	
7:4	BlockSize		数据块长度。	0
31:8	-	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

注：写入数据后，在 3 个 MCLK 时钟周期加 2 个 PCLK 时钟周期内都不能将数据写入该寄存器。

当向Enable位写 1 时启动数据传输。根据具体的方向位，DPSM相应地转至WAIT_S或WAIT_R状态。数据传输之后不必清零使能位。如果模式为 0，BlockSize将控制数据块长度，如表 20.22所示。

表 20.22 数据块长度

块大小	块长度
0	2^0=1 字节
1	2^1=2 字节
…	-
11	2^{11}=2048 字节
12:15	保留

20.5.11 数据计数器寄存器（MCIDataCnt – 0xE008 C030）

当DPSM从IDLE状态转至WAIT_R或WAIT_S状态时，将数据长度寄存器（见20.5.9 节“数据长度寄存器（MCIDataLength – 0xE008 C028）”）的值加载到MCIDataCnt寄存器。传输数据时，计数器值递减，直至计数器值达到 0。然后DPSM转至IDLE状态，且数据状态结束标志置位。表 20.23对MCIDataCnt寄存器的位分配进行了描述。

表 20.23 数据计数器寄存器（MCIDataCnt –地址 0xE008 C030）位描述

位	符号	描述	复位值
15:0	DataCount	剩余的数据。	0x0000
31:16	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

注：当数据传输结束时，该寄存器只能读取。

20.5.12 状态寄存器（MCIStatus – 0xE008 C034）

MCIStatus 寄存器是一个只读寄存器。它包含两种类型的标志：

- Static[10:0]：这些标志保持有效，直至通过写清零寄存器（见20.5.13 节“清零寄存器（MCIClear –0xE008 C038）”）将它们清零。
- Dynamic[21:11]：这些标志根据下面逻辑的状态来改变状态（例如，FIFO 满和空标志随着写入 FIFO 的数据变为有效或无效）。

表 20.24对MCIStatus寄存器进行了描述。

表 20.24 状态寄存器（MCIStatus – 地址 0xE008 C034）位描述

位	符号	描述	复位值
0	CmdCrcFail	接收到命令响应（CRC 检验失败）	0
1	DataCrcFail	发送/接收数据块（CRC 检验失败）	0
2	CmdTimeOut	命令响应超时	0
3	DataTimeOut	数据超时	0
4	TxUnderrun	发送 FIFO 下溢错误	0
5	RxOverrun	接收 FIFO 上溢错误	0
6	CmdRespEnd	接收到的命令响应（CRC 检验通过）	0
7	CmdSent	发送命令（不需要响应）	0
8	DataEnd	数据结束（数据计数器计数值为 0）	0
9	StartBitErr	宽总线模式下在所有的数据信号上都没有检测到起始位	0
10	DataBlockEnd	数据块发送/接收（CRC 检验通过）	0
11	CmdActive	命令传输正在进行	0
12	TxActive	数据发送正在进行	0
13	RxActive	数据接收正在进行	0
14	TxFifoHalfEmpty	发送 FIFO 半空	0
15	TxFifoHalfFull	接收 FIFO 半满	0
16	TxFifoFull	发送 FIFO 满	0
17	RxFifoFull	接收 FIFO 满	0
18	TxFifoEmpty	发送 FIFO 为空	0
19	RxFifoEmpty	接收 FIFO 为空	0
20	TxDataAvlbl	发送 FIFO 中的数据可用	0
21	RxDataAvlbl	接收 FIFO 中的数据可用	0
31:22	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

20.5.13 清零寄存器（MCIClear –0xE008 C038）

MCIClear寄存器是一个只写寄存器。通过向寄存器中相应的位写 1 来清除对应的静态状态标志。表 20.25对MCIClear寄存器的位分配进行了描述。

表 20.25 清零寄存器（MCIClear –地址 0xE008 C038）位描述

位	符号	描述	复位值
0	CmdCrcFailClr	清除 CmdCrcFail 标志。	-
1	DataCrcFailClr	清除 DataCrcFail 标志。	-
2	CmdTimeOutClr	清除 CmdTimeOut 标志。	-
3	DataTimeOutClr	清除 DataTimeOut 标志。	-
4	TxUnderrunClr	清除 TxUnderrun 标志。	-

续表 20.25

位	符号	描述	复位值
5	RxOverrunClr	清除 RxOverrun 标志。	-
6	CmdRespEndClr	清除 CmdRespEnd 标志。	-
7	CmdSentClr	清除 CmdSent 标志。	-
8	DataEndClr	清除 DataEnd 标志。	-
9	StartBitErrClr	清除 StartBitErr 标志。	-
10	DataBlockEndClr	清除 DataBlockEnd 标志。	-
31:11	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

20.5.14 中断屏蔽寄存器（MCIMask0 – 0xE008 C03C，MCIMask1 – 0xE008 C040）

中断屏蔽寄存器通过置位相应位为 1 来决定哪个状态标志产生中断请求。表 20.26对MCIMaskx寄存器的位分配进行了描述。

表 20.26 中断屏蔽寄存器（MCIMask0 –地址 0xE008 C03C，MCIMask1 –地址 0xE008 C040）位描述

位	符号	描述	复位值
0	Mask0	屏蔽 CmdCrcFail 标志	0
1	Mask1	屏蔽 DataCrcFail 标志	0
2	Mask2	屏蔽 CmdTimeOut 标志	0
3	Mask3	屏蔽 DataTimeOut 标志	0
4	Mask4	屏蔽 TxUnderrun 标志	0
5	Mask5	屏蔽 RxOverrun 标志	0
6	Mask6	屏蔽 CmdRespEnd 标志	0
7	Mask7	屏蔽 CmdSent 标志	0
8	Mask8	屏蔽 DataEnd 标志	0
9	Mask9	屏蔽 StartBitErr 标志	0
10	Mask10	屏蔽 DataBlockEnd 标志	0
11	Mask11	屏蔽 CmdActive 标志	0
12	Mask12	屏蔽 TxActive 标志	0
13	Mask13	屏蔽 RxActive 标志	0
14	Mask14	屏蔽 TxFifoHalfEmpty 标志	0
15	Mask15	屏蔽 RxFifoHalfFull 标志	0
16	Mask16	屏蔽 TxFifoFull 标志	0
17	Mask17	屏蔽 RxFifoFull 标志	0
18	Mask18	屏蔽 TxFifoEmpty 标志	0
19	Mask19	屏蔽 RxFifoEmpty 标志	0
20	Mask20	屏蔽 TxDataAvlbl 标志	0
21	Mask21	屏蔽 RxDataAvlbl 标志	0
31:22	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

20.5.15 FIFO计数器寄存器（MCIFifoCnt – 0xE008 C048）

MCIFifoCnt寄存器包含写入FIFO或从FIFO读取的剩余的字。当使能位在数据控制寄存器

中置位时，FIFO计数器从数据长度寄存器（见20.5.9 节“数据长度寄存器（MCIDataLength – 0xE008 C028）”）加载值。如果数据长度不是字对齐（4 的倍数），那么剩下的 1 到 3 个字节都当作字。表 20.27对MCIFifoCnt寄存器的位分配进行了描述。

表 20.27 FIFO 计数器寄存器（MCIFifoCnt –地址 0xE008 C048）位描述

位	符号	描述	复位值
14:0	DataCount	剩余的数据。	0x0000
31:15	-	保留，用户软件不应该向保留位写 1。从保留位读取的值未定义。	NA

20.5.16 数据FIFO寄存器（MCIFIFO – 0xE008 C080 到 0xE008 C0BC）

接收FIFO和发送FIFO都可以被当作 32 位宽的寄存器来读/写。FIFO的 16 个连续地址包含 16 个项。这样，微处理器可以使用它的装载和保存多个操作数指令来读/写FIFO。表 20.28对MCIFIFO寄存器进行了描述。

表 20.28 数据 FIFO 寄存器（MCIFIFO –地址 0xE008 C080 到 0xE008 C0BC）位描述

位	符号	描述	复位值
31:0	Data	FIFO 数据。	0x0000 0000

第21章 I²C接口I²C0,I²C1,I²C2

21.1 特性

- 标准 I²C 总线接口，可配置为主机、从机或者主机/从机模式；
- 在同时发送的主机之间进行仲裁，避免了串行总线数据的冲突；
- 可编程的时钟可实现 I²C 传输速率的调整；
- 主机从机之间双向数据传输；
- 串行时钟同步允许不同位速率的器件通过同一个串行总线通信；
- 串行时钟同步可作为一个握手机制来挂起和恢复串行传输；
- I²C 总线可用作测试和诊断。

21.2 应用

与外部 I²C 部件连接，如串行 RAM、LCD、音调发生器等等。

21.3 描述

I²C总线的典型配置如图 21.1所示。根据每帧中方向位（R/W）状态的不同，I²C总线上存在以下两种类型的数据传输：

- 主发送器向从接收器发送数据。主机发送的第一个字节是从机地址，接下来是数据字节数。从机每接收一个字节返回一个应答位。
- 从发送器向主接收器发送数据。主机发送的第一个字节（从机地址），然后从机返回一个应答位。接下来从机向主机发送数据字节。主机每接收一个字节都会返回一个应答位，最后一个字节除外。接收完最后一个字节后，主机返回一个“非应答位”。主机产生所有串行时钟脉冲、起始条件以及停止条件。每一帧都以一个停止条件或一个重复起始条件来结束。由于重新起始条件也是下一帧的开始，所以将不会释放 I²C 总线。

LPC2400 的每个 I²C 接口以字节方式为主，它有 4 种操作模式：主发送器模式、主接收器模式、从发送器模式和从接收器模式。

3 个 I²C 接口都相同，具有 I/O 特性的管脚除外。I²C0 符合整个 I²C 规范，支持断开到 LPC2400 的连接而不影响同一 I²C 总线上的其它器件（见“快速模式”标题下的“I²C 总线规范”描述，和数据手册中标题为“F/S 模式的 I²C 总线器件的 SDA 和 SCL I/O 特性”的表格注释）。此功能有时很有用，但它实际上限制了 I²C 接口不使用时相同管脚的交替使用。当一个微控制器中包含多个 I²C 接口时该功能几乎不用。因此，使用标准端口来实现 I²C1 和 I²C2，并且当 I²C 总线在其它器件之间操作时不支持关断 LPC2400 的能力。在系统设计过程中指定用作 I²C 接口时需要考虑这种不同的特性。

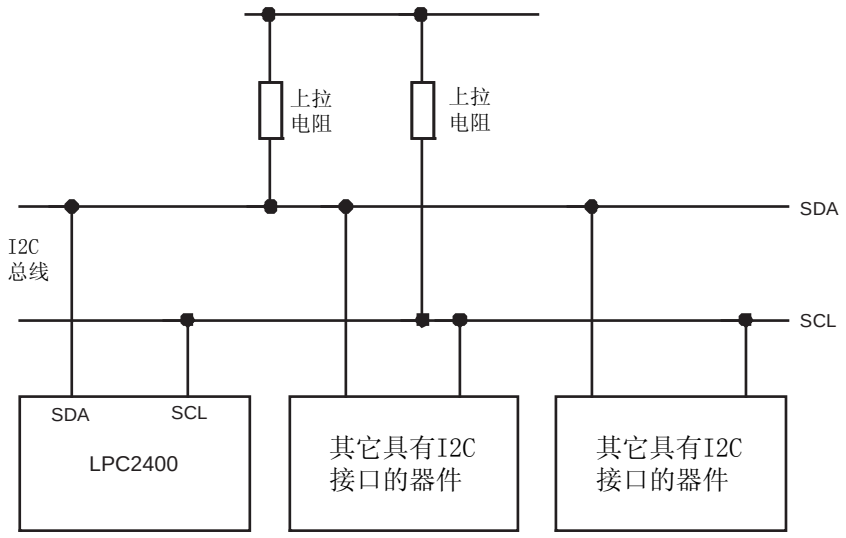


图 21.1 I²C 总线配置

21.4 管脚描述

表 21.1 I²C 管脚描述

管脚名称	类型	描述
SDA0, 1, 2	输入/输出	I²C 串行数据
SCL0, 1, 2	输入/输出	I²C 串行时钟

21.5 I²C操作模式

在一个给定的应用中，I²C 模块可用作主机、从机或同时用作主机和从机。在从机模式中，I²C 硬件查找其自身的从地址和通用调用地址。如果检测到其中的一个地址，则产生中断请求。如果处理器想成为总线主机，那么在进入主机模式前硬件必须等到总线空闲，使得从机操作不被中止。如果在主机模式中总线仲裁丢失，则 I²C 模块立刻切换到从机模式并在相同的串行传输中检测自身的从机地址。

21.5.1 主机发送器模式

在该模式中，数据从主机发送到从机。在进入主发送器模式之前，I2CONSET必须按照表21.2进行初始化。必须置位I2EN来使能I²C功能。如果AA位为 0，而另一个器件成为总线的主控器时，I²C接口将不会对任何地址产生应答，也就是说它无法进入从模式。STA，STO和SI位必须设置为 0。向I2CONCLR寄存器中的SIC位写入 1 可清零SI。

表 21.2 I2CnCONSET 用于配置主模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	0	-	-

第一个发送的字节包含接收器件的从地址（7 位）和数据方向位。在该模式下，数据方向位（R/W）应当为 0 表示执行写操作。因此第一个发送的字节为从地址和写方向位。数据

的发送每次为 8 位。每发送完一个字节，都接收到一个应答位。输出起始和停止条件来指示串行传输的起始和结束。

当软件置位 STA 位时，I²C 接口将进入主发送器模式。I²C 逻辑在总线空闲后立即发送一个起始条件。当发送完起始条件后，SI 置位。此时 I2STAT 中的状态代码应当为 0x08。该状态代码用于指向一个中断服务程序。该中断程序将从地址和写方向位装入 I2DAT（数据寄存器），然后清零 SI 位。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清零 SI。

当从地址和R/W位已发送且接收到应答位之后，SI位再次置位，并且对于主模式，可能的状态代码为 0x18，0x20 或 0x38，如果从模式使能（AA=1），可能的状态代码为 0x68，0x78 或 0xB0。每个状态代码对应的执行动作如表 21.17到表 21.20所示。

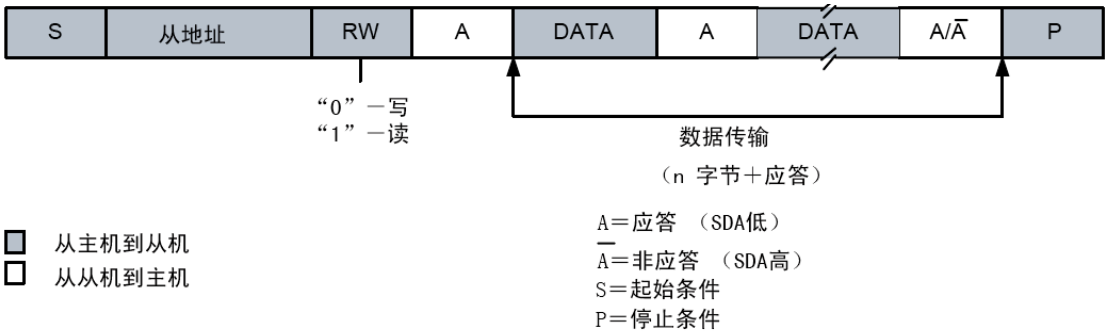


图 21.2 主发送器模式中的格式

21.5.2 主接收器模式

在主接收器模式中，接收的数据来自从发送器。传输的初始化与主发送器模式相同。当发送完起始条件后，中断服务程序必须将从地址和数据方向位装入 I²C 数据寄存器（I2DAT），然后清零 SI 位。在这种情况下，数据方向位(R/W)应为 1 来指示一个读操作。

当从地址和数据方向位已发送且接收到应答位之后，SI置位而状态寄存器将显示状态代码。对于主模式，可能的状态代码为 0x40，0x48 或 0x38。对于从模式，可能的状态代码为 0x68，0x78 或 0xB0。详细内容请参考表 21.18。

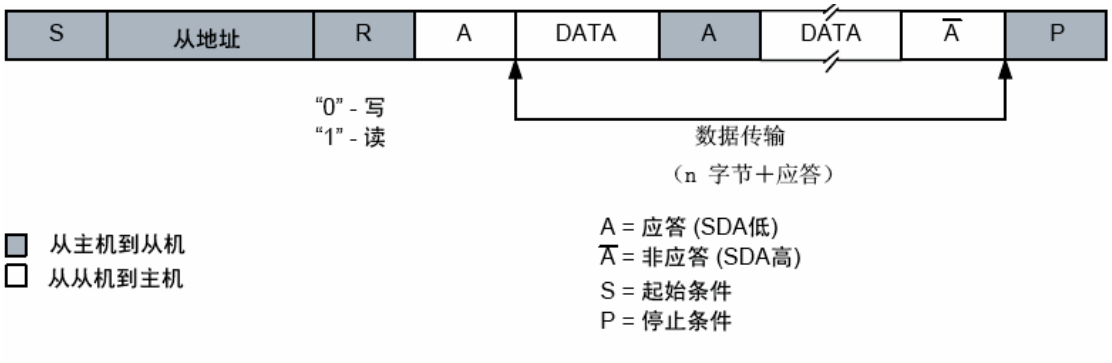


图 21.3 主接收器模式的格式

在一个重复的起始条件之后，I²C 可以切换到主发送器模式。

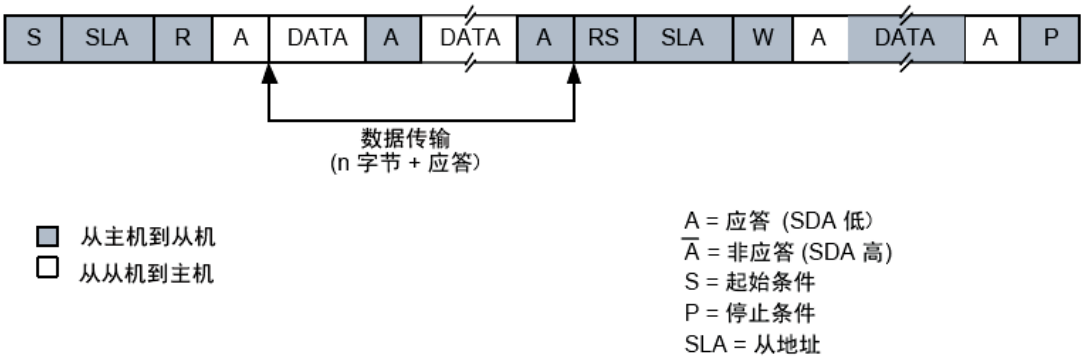


图 21.4 在发送重复起始条件后，主接收器切换为主发送器

21.5.3 从接收器模式

在从接收器模式中，从主发送器接收数据字节。要初始化从接收器模式，则用户应该写从地址寄存器（I2ADR）和写I²C控制置位寄存器（I2CONSET），如 表 21.3所示：

表 21.3 I2CnCONSET 用于配置从模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	1	-	-

I2EN 必须置位以使能 I²C 功能。AA 位必须置位以使 I²C 应答自身的从地址或通用调用地址。STA，STO 和 SI 设置为 0。

当I2ADR和I2CONSET完成初始化时，I²C接口一直等待直到它被自身的从地址或通用地址（两者后面都紧跟数据方向位）寻址为止。如果方向位为 0(W)，则I²C将进入从接收器模式。如果方向位为 1(R)，则I²C将进入从发送器模式。在接收到地址和方向位后，SI置位并可从状态寄存器（I2STAT）中读出有效的状态代码。状态代码及操作请参考表 21.19。

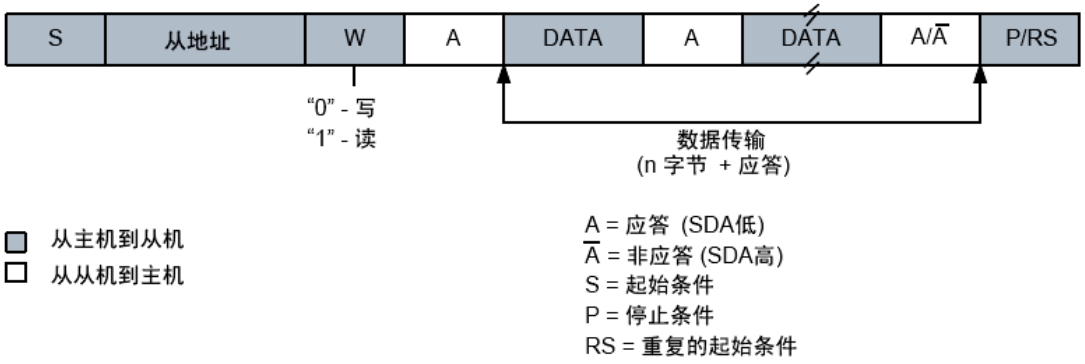


图 21.5 从接收器模式的格式

21.5.4 从发送器模式

第一个字节的接收和处理与从接收器模式相同。但在该模式中，方向位将为 1 指示一个读操作。串行数据通过 SDA 发送而串行时钟通过 SCL 输入。在串行传输开始和结束时对起始和停止条件进行识别。在一个给定的应用中，I²C 可以为主模式也可以为从模式。在从模式中，I²C 硬件寻找它自身的从地址和通用调用地址。如果检测到其中一个地址，将产生中

断请求。当微控制器希望成为总线主机时，硬件在进入主模式前一直等待，直到总线释放，这样就不会中断一个可能的从机动作。如果在主模式中总线仲裁丢失，I²C 接口将立即切换到从模式并可以在同一个串行传输中检测自身的从地址。

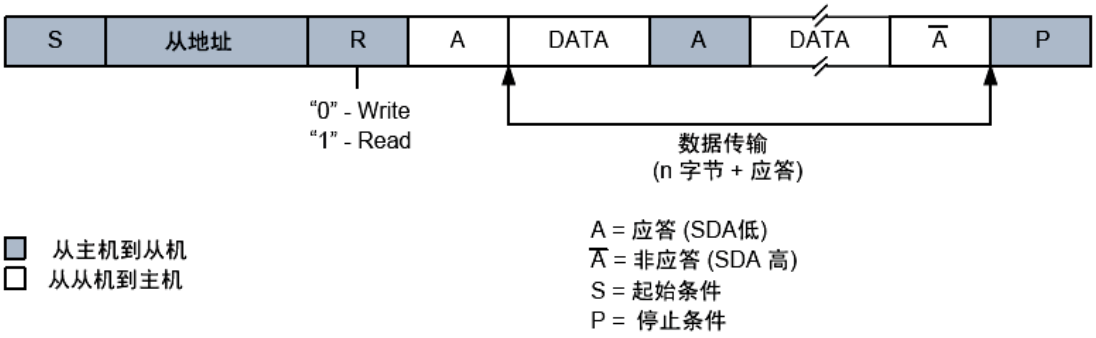


图 21.6 从发送器模式的格式

21.6 I²C的实现和操作

21.6.1 输入滤波器和输出部分

输入信号与内部时钟同步，短于 3 个时钟的脉冲干扰可以滤除。

I²C 输出是一个特殊的端口，它是为了遵循 I²C 规范而设计的。I2C1 和 I2C2 的输出为标准端口 I/O，支持整个 I²C 规范的子集。

图 21.7所示为如何实现片内I²C总线接口，下面的文档将对各个模块进行描述。

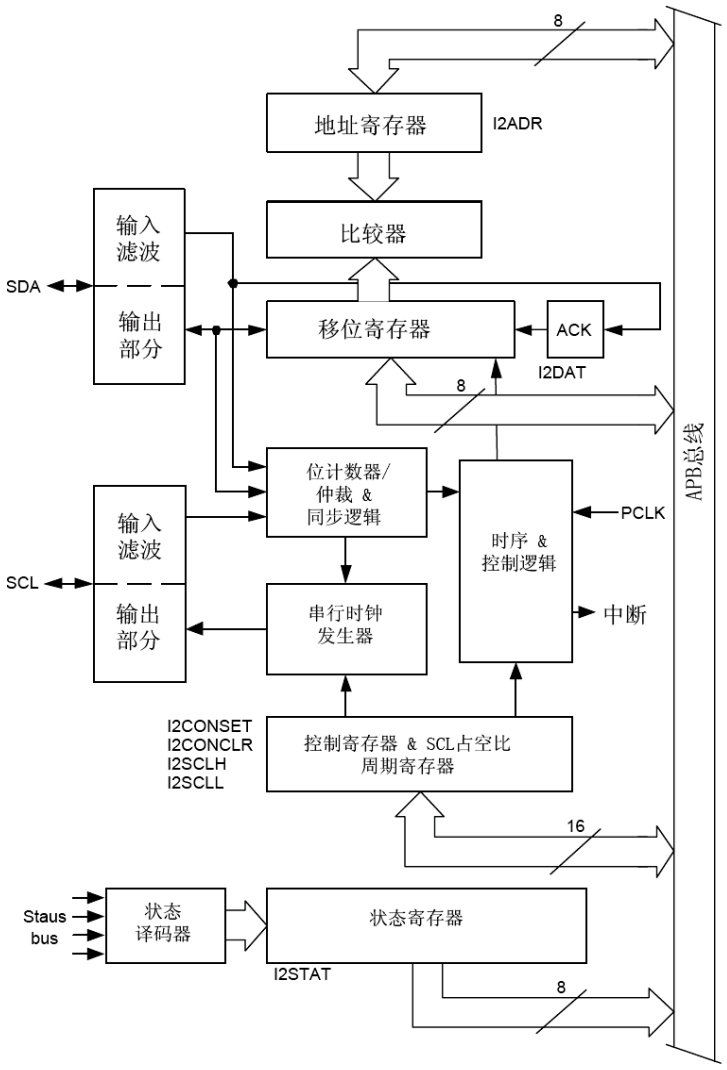


图 21.7 I²C 总线串行接口模块框图

21.6.2 地址寄存器I2ADDR

当器件编程用作从发送器或接收器时，该寄存器用来存放 I²C 模块响应的 7 位从地址（7 个最高位）。LSB（GC）用来使能通用调用地址（0x00）的识别。

21.6.3 比较器

比较器将接收到的 7 位从地址与其自身的从地址（I2ADR 的 7 个最高位）相比较。它还将首次接收到的 8 位字节与通用调用地址（0x00）相比较。如果任何一者相同，相应的状态位置位，产生中断请求。

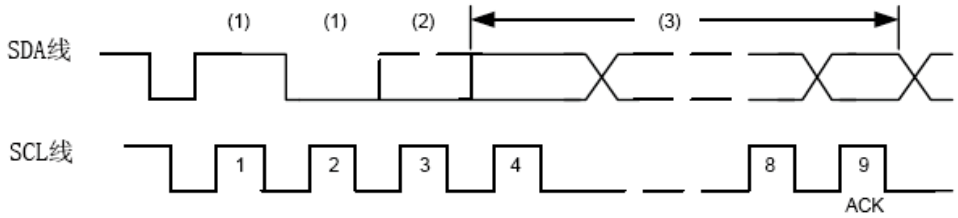
21.6.4 移位寄存器I2DAT

该 8 位寄存器用来存放要发送的一个字节的串行数据或刚接收到的一个字节。I2DAT 的数据总是从右向左移动；最先发送 MSB（位 7），接收完一个字节后，接收到的数据的第一位放置到 I2DAT 的 MSB。当数据正在移出时，总线上的数据同时移入；I2DAT 通常保存的是总线上的最后一个字节。因此，在仲裁丢失时，主发送器到从接收器的转变和 I2DAT 中数据的更新同时进行。

21.6.5 仲裁和同步逻辑

在主发送器模式中，仲裁逻辑检查每个发送的逻辑 1 作为一个逻辑 1 真正出现在 I²C 总线上。如果总线的另一个器件撤消了一个逻辑 1 并将 SDA 线拉低，仲裁丢失，且 I²C 模块立刻由主发送器变为从接收器。I²C 模块继续输出时钟脉冲（在 SCL 上），直至发送完当前的串行字节。

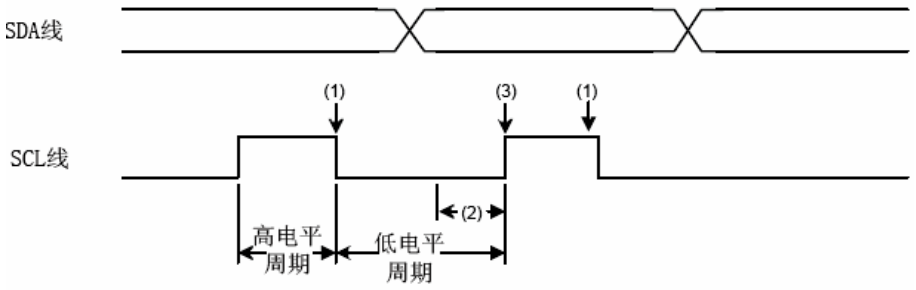
仲裁也可能在主接收器模式中丢失。这种情况只在I²C模块正在向总线返回一个“非应答：（逻辑 1）”时出现。当总线的另一个器件将信号拉低时仲裁丢失。由于它只在串行字节结束时出现，因此I²C模块不会再产生时钟脉冲。图 21.8所示为仲裁过程。



1. 器件发送串行数据。
2. 另一个器件通过拉低 SDA 线撤消了该 I²C 主机发送的一个逻辑 1（虚线）。仲裁丢失，I²C 进入从接收器模式。
3. 此时 I²C 处于从接收器模式，但仍产生时钟脉冲，直至发送完当前字节。I²C 将不为下个字节的传输产生时钟脉冲。一旦赢得仲裁，SDA 上的数据传输由新的主机来启动。

图 21.8 仲裁过程

同步逻辑使得串行时钟发生器与另一个器件SCL线上的时钟脉冲同步。如果 2 个或更多主器件产生时钟脉冲，则高电平周期取决于产生最短高电平时间的器件；低电平周期取决于产生最长低电平时间的器件。图 21.9所示为同步过程。



1. 另一个器件在 I²C 计时完一个完整的高电平时间之前拉低 SCL 线。其它器件决定了高电平（最短）的时间。
2. 另一个器件在 I²C 计时完一个完整的低电平时间并释放 SCL 之后继续拉低 SCL 线。I²C 时钟发生器被迫等待，直至 SCL 变高。其它器件决定了低电平（最长）的时间。
3. SCL 线被释放，时钟发生器开始计时高电平时间。

图 21.9 串行时钟同步

从机可以延长低电平时间来使总线主机减速。也可通过延长低电平时间来实现握手。延长低电平时间的操作在每位或一个完整字节传输之后执行。I²C 模块将在发送或接收完一个字节且传输完应答位后延长 SCL 低电平时间。设置串行中断标志（SI），继续延长低电平时间，直至串行中断标志清除。

21.6.6 串行时钟发生器

当 I²C 模块在主发送器或主接收器模式时，该可编程时钟脉冲发生器提供 SCL 时钟脉冲。当 I²C 模块处于从机模式时时钟发生器关闭。I²C 输出时钟频率和占空比可通过 I²C 时钟控制寄存器编程。详见 I2CSCLL 和 I2CSCLH 寄存器的描述。输出时钟脉冲使用编程设定的占空比，除非总线与上面描述的其它 SCL 时钟源同步。

21.6.7 时序和控制

时序和控制逻辑产生串行字节处理的时序和控制信号。该逻辑模块具有为 I2DAT 中的数据提供移位脉冲、使能比较器、产生和检测起始和停止条件、接收和发送应答位、控制主机和从机模式，含有中断请求逻辑以及监控 I²C 总线状态等功能。

21.6.8 控制寄存器，I2CONSET和I2CONCLR

I²C 控制寄存器包含用于控制以下 I²C 模块功能的位：串行传输的启动和重启、串行传输的终止、位速率、地址识别和应答。

I2CONSET 可作为 I²C 控制寄存器的内容被读出。写 I2CONSET 将置位 I²C 控制器寄存器中相应的被写为 1 的位；相反，写 I2CONCLR 将清除 I²C 控制寄存器中相应的被写为 1 的位。

21.6.9 状态译码器和状态寄存器

状态译码器取出所有内部状态位并将它们压缩成一个 5 位的代码。该代码与每个I²C总线状态位一一对应。5 位代码可用于产生向量地址，以便快速处理不同的服务程序。每个服务程序处理一个特定的总线状态。如果I²C模块的所有四种模式都被使用，则有 26 种可能的总线状态。当串行中断标志置位（通过硬件）并一直保持时，5 位状态代码锁存到状态寄存器的高 5 位，直至中断标志被软件清除。状态寄存器的低 3 位总是为 0。如果状态代码用作服务程序的向量，则程序转移到 8 位地址指向的空间。大多数的服务程序不会超过 8 字节（见本节的软件例程）。

21.7 寄存器描述

每个I²C接口包含 7 个寄存器，如表 21.4所示。

表 21.4 I²C 寄存器映射

名称	描述	访问	复位值[1]	I²Cn 寄存器名称&地址
I2CONSET	**I²C 控制置位寄存器。**当向该寄存器写入 1 时，I²C 控制寄存器中相应位置位。写 0 到 I²C 控制寄存器的相应位没有影响。	R/W	0x00	I2C0CONSET - 0xE001 C000 I2C1CONSET - 0xE005 C000 I2C2CONSET - 0xE008 0000
I2STAT	**I²C 状态寄存器。**在 I²C 操作中，该寄存器提供详细的状态码，使软件确定所需的下一步操作。	RO	0xF8	I2C0STAT - 0xE001 C004 I2C1STAT - 0xE005 C004 I2C2STAT - 0xE008 0004

续表 21.4

名称	描述	访问	复位值[1]	I²Cn 寄存器名称&地址
I2DAT	**I²C 数据寄存器。**在主/从机发送模式下，要发送的数据被写入该寄存器。在主/从机接收模式下，接收到的数据可从该寄存器中读取。	R/W	0x00	I2C0DAT - 0xE001 C008 I2C1DAT - 0xE005 C008 I2C2DAT - 0xE008 0008
I2ADR	**I²C 从地址寄存器。**包含从机模式下 I²C 接口操作的 7 位从地址，不在主机模式下使用。最低位决定从机是否响应通用调用地址。	R/W	0x00	I2C0ADR - 0xE001 C00C I2C1ADR - 0xE005 C00C I2C2ADR - 0xE008 000C
I2SCLH	**SCH 占空比寄存器高半字。**确定 I²C 时钟的高时间。	R/W	0x04	I2C0SCLH - 0xE001 C010 I2C1SCLH - 0xE005 C010 I2C2SCLH - 0xE008 0010
I2SCLL	**SCL 占空比寄存器低半字。**确定 I²C 时钟的低时间。12nSCLL 和 12nSCLH 一起确定 I²C 主机产生的时钟频率和从机模式下使用的特定时间。	R/W	0x04	I2C0SCLL - 0xE001 C014 I2C1SCLL - 0xE005 C014 I2C2SCLL - 0xE008 0014
I2CONCLR	**I²C 控制清零寄存器。**当向该寄存器中的位写入 1 时，I²C 控制寄存器中相应位被清零。写 0 到 I²C 控制寄存器的相应位没有影响。	WO	NA	I2C0CONCLR - 0xE001 C018 I2C1CONCLR - 0xE005 C018 I2C2CONCLR - 0xE008 0018

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

21.7.1 I²C控制置位寄存器（I2C[0/1/2]CONSET：0xE001 C000, 0xE005 C000, 0xE008 0000）

I2CONSET 寄存器控制 I2CON 寄存器中位的设置，该寄存器中位的设置控制 I²C 接口的操作。写 1 到该寄存器使 I²C 控制寄存器中相应位置位。写 0 没有影响。

表 21.5 I²C 控制置位寄存器（I2C[0/1/2]CONSET- 地址：0xE001 C000, 0xE005 C000, 0xE008 0000）位描述

位	符号	描述	复位值
1:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	AA	声明应答标志。	0
3	SI	I²C 中断标志。见下文。	0
4	STO	停止标志。见下文。	0
5	STA	起始标志。见下文。	0
6	I2EN	I²C 接口使能。见下文。	0
7	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

I2EN 为 I²C 接口使能。当该位置位时，使能 I²C 接口。向 I2CONCLR 寄存器中的 I2ENC 位写入 1 将使 I2EN 位清零。当 I2EN 位为 0 时，I²C 接口功能被禁止。

当 I2EN 为 0 时，SDA 和 SCL 输入信号被忽略，I²C 模块在“不可寻址的”从机状态下，且 STO 位被强制为“0”。

I2EN 不应用于暂时释放 I²C 总线，当 I2EN 复位时，I²C 总线状态丢失。应使用 AA 标志代替。

STA 为起始标志。当 STA=1 时，I²C 接口进入主模式并发送一个起始条件，如果已经处于主模式，则发送一个重复起始条件。

当 STA=1 并且 I²C 接口还没进入主模式时，I²C 接口将进入主模式，检测总线并在总线空闲时产生一个起始条件。如果总线忙，则等待一个停止条件（释放总线）并在延迟半个内部时钟发生器周期后发送一个起始条件。当 I²C 接口已经处于主模式中并发送或接收了数据时，I²C 接口会发送一个重复的起始条件。STA 可在任何时候置位，当 I²C 接口处于可寻址的从模式时，STA 也可以置位。

向 I2CONCLR 寄存器中的 STAC 位写入 1 使 STA 位清零。当 STA=0 时，不会产生起始或重复起始条件。

当 STA 和 STO 都置位时，如果 I²C 接口处于主模式，I²C 接口将向总线发送一个停止条件，然后发送一个起始条件。如果 I²C 接口处于从模式，则产生一个内部停止条件，但不发送到总线上。

STO 为停止标志。当 STO 为 1 时，在主模式中，向 I²C 总线发送一个停止条件或在从模式中使总线从错误状态中恢复。当主模式中 STO=1 时，向总线发送停止条件。当总线检测到停止条件时，STO 自动清零。

在从模式中，置位 STO 位可从错误状态中恢复。这种情况下不向总线发送停止条件。硬件的表现就好像是接收到一个停止条件并切换到不可寻址的从接收器模式。STO 标志由硬件自动清零。

SI 为 I²C 中断标志。当 I²C 状态改变时该位置位。但是，由于在那种情况下中断服务程序不起作用，所以进入状态 F8 不置位 SI。

当 SI 置位时，SCL 线上的串行时钟低周期扩展，且串行传输被中止。当 SCL 为高时，SI 标志的状态不受影响。SI 必须通过软件复位，通过向 I2CONCLR 寄存器的 SIC 位写入 1 来实现。

AA 为声明应答标志。当该位置位时，SCL 线的应答时钟脉冲内出现下面的任意条件之一将产生一个应答（SDA 上的低电平）：

1. 接收到从地址寄存器中的地址。
2. 当 I2ADR 中的通用调用位（GC）置位时，接收到通用调用地址。
3. 当 I²C 接口处于主接收器模式时，接收到一个数据字节。
4. 当 I²C 接口处于可寻址的从接收器模式时，接收到一个数据字节。

向 I2CONCLR 寄存器中的 AAC 位写入 1 会使 AA 位清零。当 AA 为零时，SCL 线的应答时钟脉冲内出现下列情况将返回一个非应答信号（SDA 上的高电平）：

1. 当 I²C 接口处于主接收器模式时，接收到一个数据字节。
2. 当 I²C 接口处于可寻址的从接收器模式时，接收到一个数据字节。

21.7.2 I²C控制清零寄存器（I2C[0/1/2]CONCLR：0xE001 C018, 0xE005 C018, 0xE008 0018）

I2CONCLR 寄存器控制 I2CON 寄存器中位的清零，I2CON 寄存器控制 I²C 接口的操作。写 1 到该寄存器使 I²C 控制寄存器中相应位清零。写 0 没有影响。

表 21.6 I²C 控制清零寄存器（I2C[0/1/2]CONCLR- 地址 0xE001 C018, 0xE005 C018, 0xE008 0018）位描述

位	符号	描述	复位值
1:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	AAC	声明应答标志清零位。	0
3	SIC	I²C 中断标志清零位。	0
4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
5	STAC	起始标志清零位。	0
6	I2ENC	I²C 接口禁止位。	0
7	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

AAC 声明应答标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 AA 位。写入 0 无效。

SIC 为 I²C 中断标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 SI 位。写入 0 无效。

STAC 为起始标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 STA 位。写入 0 无效。

I2ENC 为 I²C 接口禁止位。向该位写入 1 清零 I2CONSET 寄存器中的 I2EN 位。写入 0 无效。

21.7.3 I²C状态寄存器（I2C[0/1/2]STAT - 0xE001 C004, 0xE005 C004, 0xE008 0004）

每个 I²C 状态寄存器反映相应 I²C 接口的条件。I²C 状态寄存器为只读寄存器。

表 21.7 I²C 状态寄存器（I2C[0/1/2]STAT – 地址 0xE001 C004, 0xE005 C004, 0xE008 0004）位描述

位	符号	描述	复位值
2:0	-	这 3 个位不使用且总是为 0。	0
7:3	Status	这些位给出 I²C 接口的真实状态信息。	0x1F

最低 3 位总是为 0。状态寄存器中的一个字节表示一个状态代码。一共有 26 种可能存在的状态代码。当代码为 0xF8 时，无可用的相关信息，SI位不会置位。所有其它 25 种状态代码都对应一个已定义的I²C状态。当进入其中一种状态时，SI位将置位。所有状态代码的描述见表 21.17到表 21.20。

21.7.4 I²C数据寄存器（I2C[0/1/2]DAT- 0xE001 C008, 0xE005 C008, 0xE008 0008）

该寄存器包含要发送或刚接收的数据。当它没有处理字节的移位时，CPU 可对其进行读写。该寄存器只能在 SI 置位时访问。只要 SI 置位，I2DAT 中的数据就保持稳定。I2DAT 中的数据移位总是从右至左进行：第一个发送的位是 MSB（位 7），在接收字节时，第一个接收到的位存放在 I2DAT 的 MSB。

表 21.8 I²C 数据寄存器（I2C[0/1/2]DAT- 地址 0xE001 C008, 0xE005 C008, 0xE008 0008）位描述

位	符号	描述	复位值
7:0	Data	该寄存器保留已经接收，或准备要发送的数据值。	0

21.7.5 I²C从地址寄存器（I2C[0/1/2]ADR- 0xE001 C00C, 0xE005 C00C, 0xE008 000C）

这些寄存器可读可写，但只能在 I²C 接口设置为从模式时才能使用。在主模式中，该寄存器无效。I2ADR 的 LSB 为通用调用位。当该位置位时，通用调用地址（0x00）被识别。

表 21.9 I²C 从地址寄存器（I2C[0/1/2]ADR- 地址 0xE001 C00C, 0xE005 C00C, 0xE008 000C）位描述

位	符号	描述	复位值
0	GC	通用调用使能位	0
7:1	Address	从模式的 I²C 器件地址	0x00

21.7.6 I²C SCL高电平占空比寄存器（I2C[0/1/2]SCLH- 0xE001 C010, 0xE005 C010, 0xE008 0010）

表 21.10 I²C SCL 高电平占空比寄存器（I2C[0/1/2]SCLH- 地址 0xE001 C010, 0xE005 C010, 0xE008 0010）位描述

位	符号	描述	复位值
15:0	SCLH	SCL 高电平周期选择计数	0x0004

21.7.7 I²C SCL低电平占空比寄存器（I2C[0/1/2]SCLL- 0xE001 C014, 0xE005 C014, 0xE008 0014）

表 21.11 I²C SCL 低电平占空比寄存器（I2C[0/1/2]SCLL- 地址 0xE001 C014, 0xE005 C014, 0xE008 0014）位描述

位	符号	描述	复位值
15:0	SCLL	SCL 低电平周期选择计数	0x0004

21.7.8 选择合适的I²C数据率和占空比

软件必须通过对 I2SCLH 和 I2SCLL 寄存器进行设置来选择合适的数据率和占空比。I2SCLH 定义 SCL 高电平所保持的 PCLK 周期数，I2SCLL 定义 SCL 低电平的 PCLK 周期数。频率由下面的公式得出（f_{PCLK} 是 PCLK 的频率）：

$$I^2C_{bitfrequency} = \frac{f_{PCLK}}{I2CSCLH + I2CSCLL} \quad (15)$$

I2SCLL和I2SCLH的值不一定要相同。通过设定这两个寄存器可得到SCL的不同占空比。例如，I²C总线规范定义 400kHz I²C速率下不同值的SCL低时间和高时间。但寄存器的值必须确保I²C数据通信速率在 0 到 400kHz之间。每个寄存器的值都必须大于或等于 4。表 21.12给出基于PCLK频率和I2SCLL和I2SCLH值的I²C总线速率的实例。

表 21.12 I²C 时钟速率的实例

I2SCLL+ I2SCLH	PCLK (MHz)下 I²C 的位速率（kHz）						
	1	5	10	16	20	40	60
8	125	-	-	-	-	-	-
10	100	-	-	-	-	-	-
25	40	200	400	-	-	-	-
50	20	100	200	320	400	-	-
100	10	50	100	160	200	400	-
160	6.25	31.25	62.5	100	125	250	375
200	5	25	50	80	100	200	300
400	2.5	12.5	25	40	50	100	150
800	1.25	6.25	12.5	20	25	50	75

21.8 I²C操作模式的细节

4 种操作模式：

- 主发送器
- 主接收器
- 从接收器
- 从发送器

每种操作模式的数据传输如图 21.10到图 21.14所示。表 21.13列出描述I²C操作模式时这些图中使用的缩写：

表 21.13 描述 I²C 操作的缩写

缩写	说明
S	起始条件
SLA	7 位从机地址
R	读数据位（SDA 为高电平）
W	写数据位（SDA 为低电平）
A	应答位（SDA 为低电平）
$\overline{A}$	非应答位（SDA 为高电平）
Data	8 位数据字节
P	停止条件

在图 21.10到图 21.14中，圆圈用来指示串行中断标志何时被置位。圆圈中的数字表示的是I2STAT寄存器中的状态代码。每当出现这些状态代码时，必须执行服务程序来继续或结束串行传输。由于串行传输被挂起，这些服务程序并不重要，直至串行中断标志被软件清除。

当进入串行中断程序时，I2STAT的状态代码用来指向跳转到的相应的服务程序。对于每个状态代码需要的软件操作以及后面串行传输的详细情况见表 21.17到表 21.21。

21.8.1 主发送器模式

在主发送器模式下，数据字节发送到从接收器（见图 21.10）。在进入主发送器模式之前，I2CON必须进行如下初始化：

表 21.14 I2CONSET 用于初始化主发送器模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	x	-	-

I²C 的速率也必须在 I2SCLL 和 I2SCLH 寄存器中配置。I2EN 必须设置为逻辑 1 来使能 I²C 模块。如果 AA 位复位，当另一个器件变成总线主机时，I²C 模块将不会应答其自身的从机地址或通用调用地址。换句话说，如果 AA 位复位，I²C 接口就不能进入从机模式。STA, STO 和 SI 必须复位。

可通过置位 STA 位进入主发送器模式。一旦总线空闲，I²C 逻辑将测试 I²C 总线并产生一个起始条件。当 START 条件被发送时，串行中断标志（SI）置位，状态寄存器（I2STAT）中的状态代码变为 0x08。中断服务程序利用该状态代码进入相应的状态服务程序，将从机地址和数据方向位（SLA+W）装入 I2DAT。I2CON 的 SI 位必须在串行传输持续之前复位。

当发送完从机地址和方向位且接收到一个应答位时，串行中断标志(SI)被重新置位，I2STAT中可能是一系列不同的状态代码。主机模式下为 0x18, 0x20 或 0x38，从机模式(AA= 逻辑 1)为 0x68, 0x78 或 0xB0。每个状态代码的相应操作详见表 21.17。在重复起始条件（状态 0x10）后，I²C模块通过将SLA+R装入I2DAT可切换为主接收器模式。

21.8.2 主接收器模式

在主接收器模式下，接收来自从发送器的数据字节（见图 21.11）。传输在主发送器模式中初始化。当发送完起始条件后，中断服务程序必须向I2DAT装入 7 位从机地址和数据方向位(SLA+R)。必须先清除I2CON中的SI位，再继续执行串行传输。

当发送完从机地址和数据方向位且接收到一个应答位时，串行中断标志(SI)被重新置位，I2STAT中可能是一系列不同的状态代码。主机模式下为 0x40, 0x48 或 0x38，从机模式(AA= 1)为 0x68, 0x78 或 0xB0。每个状态代码的相应操作详见表 20.18。在重复起始条件（状态 0x10）后，I²C模块通过将SLA+W装入I2DAT可切换为主发送器模式。

21.8.3 从接收器模式

在从接收器模式下，接收来自主发送器的数据字节（见图 21.12）。要初始化从接收器模式，I2ADR和I2CON必须被装入以下内容：

表 21.15 在从接收器模式中使用 I2C0ADR 和 I2C1ADR

位	7	6	5	4	3	2	1	0
符号	自身 7 位从地址							GC

高 7 位是主机寻址时 I²C 模块响应的地址。如果 LSB(GC)被置位，I²C 模块将响应通用调用地址(0x00)；否则忽略通用调用地址。

表 21.16 I2C0CONSET 和 I2C1CONSET 用于初始化从接收器模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	1	-	-

I^2C 总线的速率设置不影响从机模式下的 I^2C 模块。必须置位 I2EN 来使能 I^2C 模块。AA 位必须置位以使能 I^2C 模块来应答其自身从机地址或通用调用地址。STA, STO 和 SI 必须被复位。

当I2ADR和I2CON完成初始化后，I^2C模块一直等待，直至被从机地址及其后面的数据方向位寻址，该数据方向位必须为“0”(W)，便于I^2C模块工作在从接收器模式下。接收完其自身的从机地址和W位后，串行中断标志(SI)置位，可从I2STAT中读出一个有效的状态代码。该状态代码用作状态服务程序的向量。每个状态代码的相应操作见表 21.19。如果I^2C模块工作在主机模式下时仲裁丢失，那么也可进入从接收器模式（见状态 0x68 和 0x78）。

如果 AA 位在传输过程中复位，则在接收完下一个数据字节后 I^2C 模块将向 SDA 返回一个非应答（逻辑 1）。当 AA 复位时，I^2C 模块不响应其自身的从机地址或通用调用地址。但是，I^2C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可用来暂时将 I^2C 模块从 I^2C 总线上分离出来。

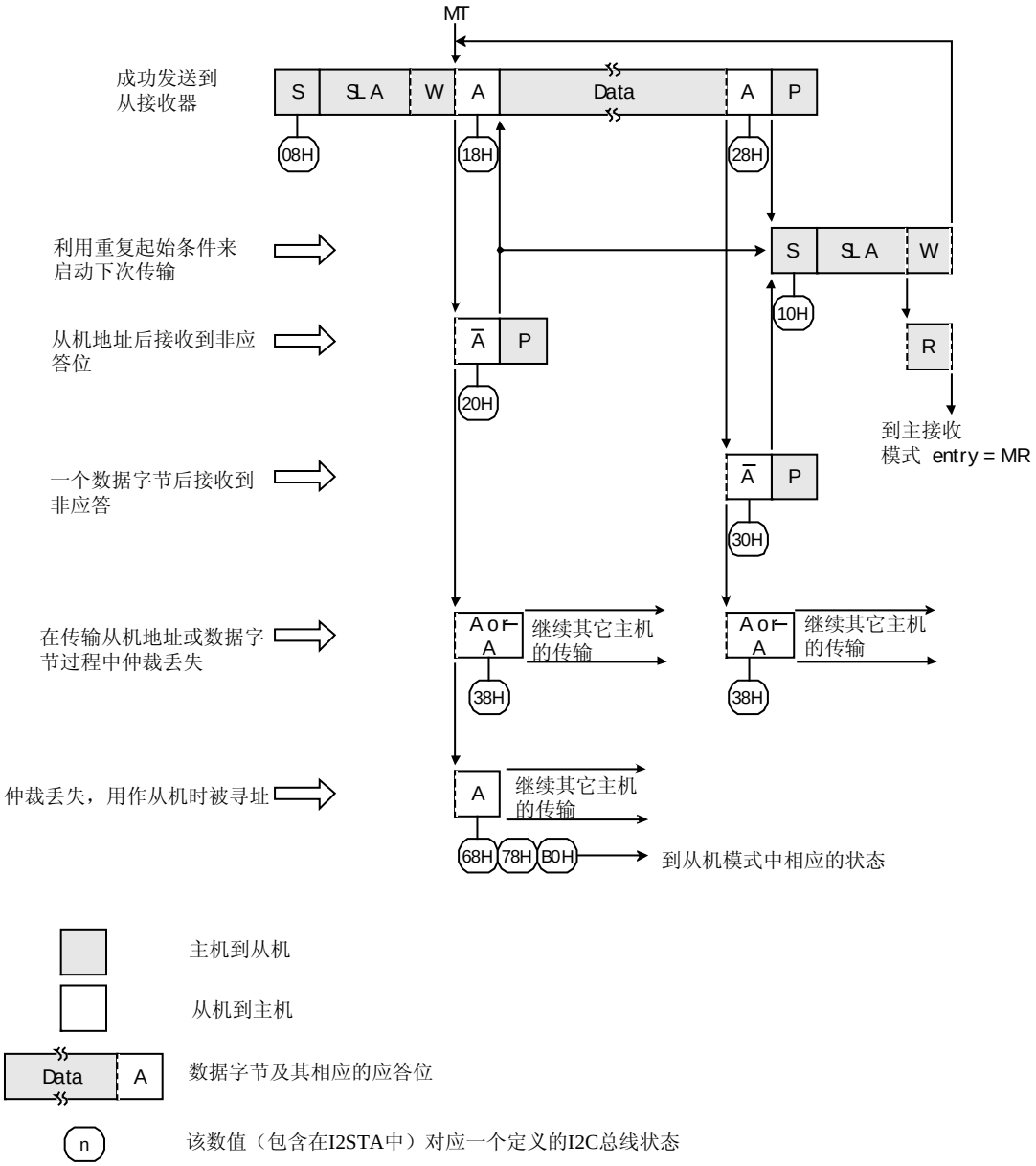


图 21.10 主发送器模式的格式和状态

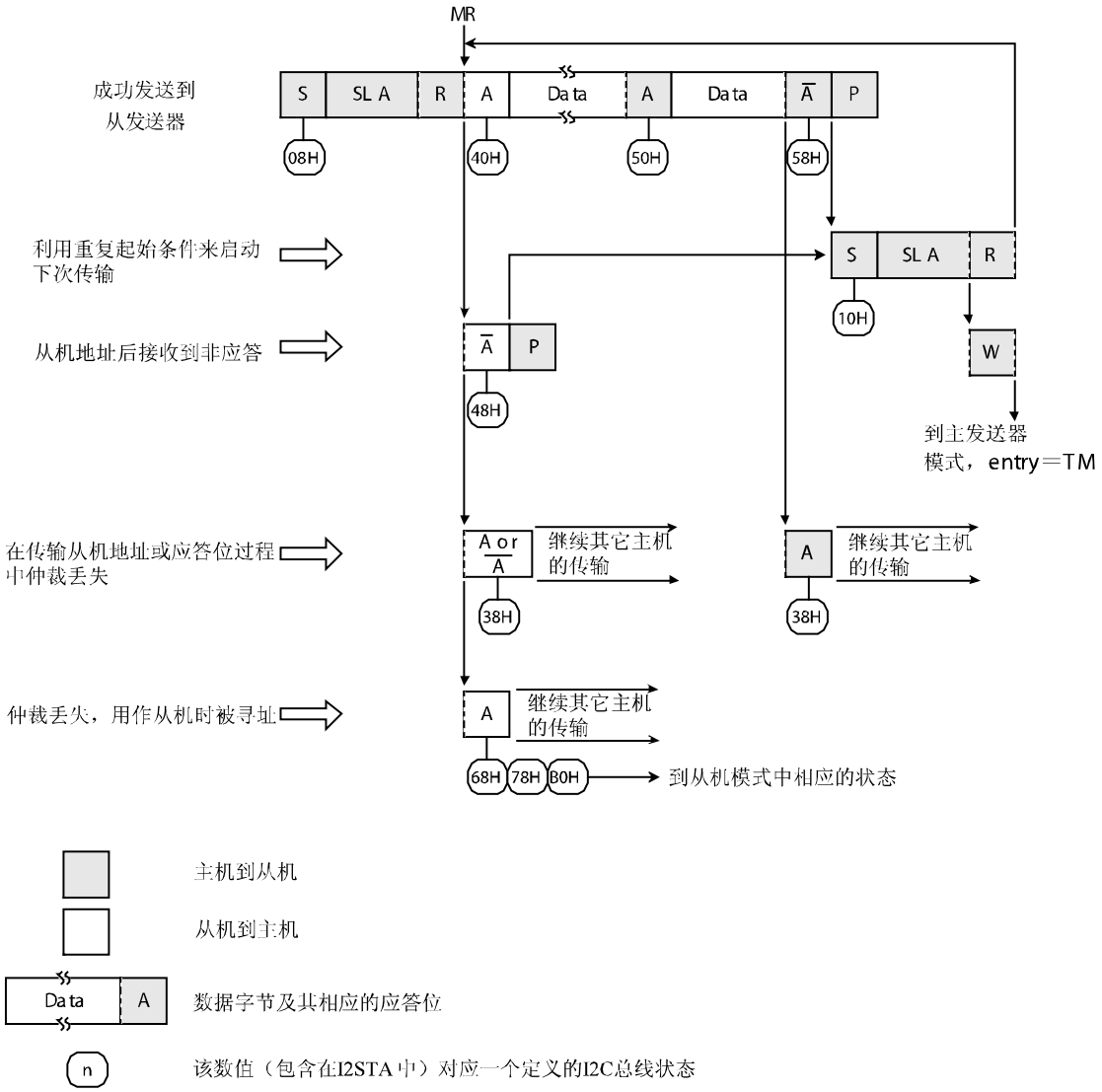


图 21.11 主接收器模式的格式和状态

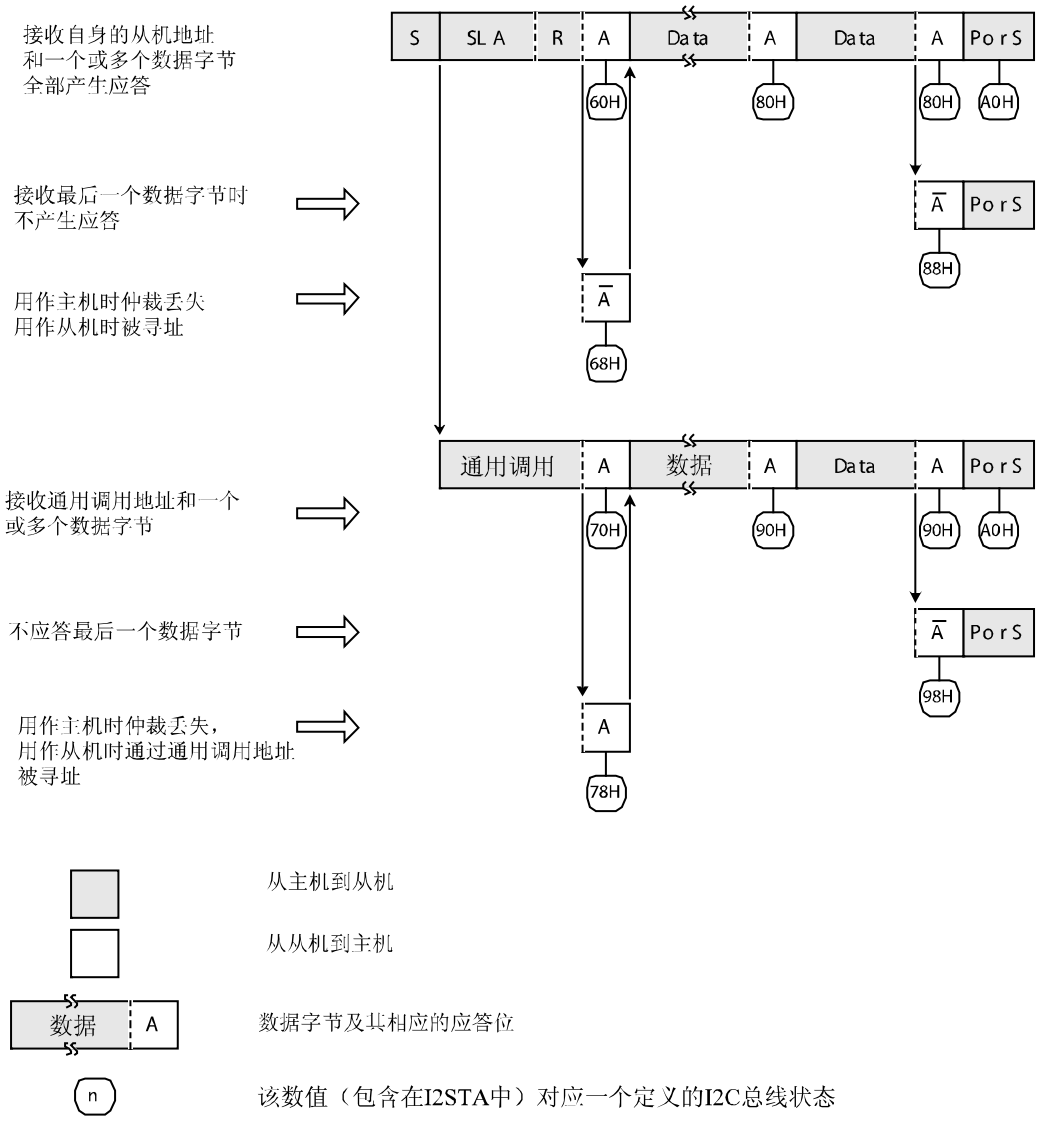


图 21.12 从接收器模式的格式和状态

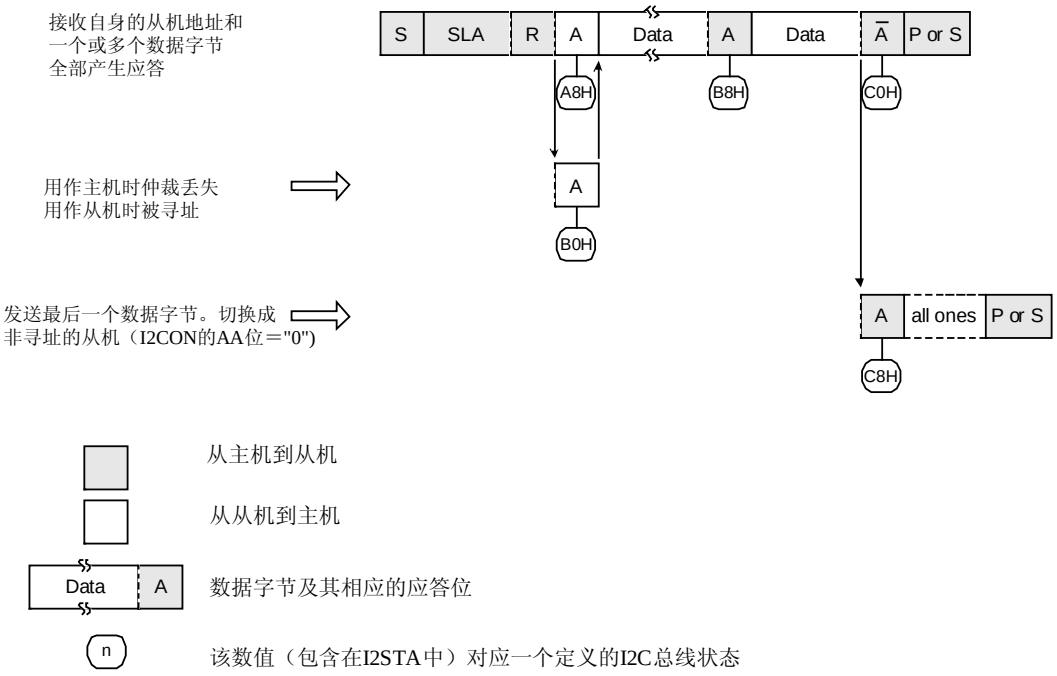


图 21.13 从发送器模式的格式和状态

21.8.4 从发送器模式

在从发送器模式下，向主接收器发送数据字节（见图 21.13）。数据传输在从接收器模式下初始化。当I2ADR和I2CON完成初始化后，I²C模块一直等待，直至被自身的从机地址及其后面的数据方向位寻址，该数据方向位必须为“1”(R)，以便I²C模块工作在从发送器模式下。接收完其自身的从机地址和R位后，串行中断标志(SI)置位，可从I2STAT中读出一个有效的状态代码。该状态代码用作状态服务程序的向量，每个状态代码的相应操作见表 21.20。如果I²C模块在主机模式下时仲裁丢失，则可进入从发送器模式（见状态 0xB0）。

如果 AA 位在传输过程中复位，则 I²C 模块将发送传输的最后一个字节并进入状态 0xC0 或 0xC8。I²C 模块切换为非寻址的从机模式，如果继续传输它将忽略主接收器。因此主接收器接收所有 1 作为串行数据。当 AA 复位时，I²C 模块不响应其自身的从机地址或通用调用地址。但是，I²C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可用来暂时将 I²C 模块从 I²C 总线上分离出来。

表 21.17 主发送器模式

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x08	已发送起始条件	装入 SLA+W	x	0	0	x	将发送 SLA+W，接收 ACK 位
0x10	已发送重复起始条件	装入 SLA+W 或	x	0	0	x	同上；
		装入 SLA+R	x	0	0	x	将发送 SLA+W，I²C 将切换为 MST/REC 模式
0x18	已发送 SLA+W；已接收 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	0	1	0	x	将发送停止条件；STO 标志将复位
		无 I2DAT 动作	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位

续表 21.17

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x20	已发送 SLA+W；已接收非 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	0	1	0	x	将发送停止条件；STO 标志将复位
		无 I2DAT 动作	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位
0x28	已发送 I2DAT 中的数据字节；已接收 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	0	1	0	x	将发送停止条件；STO 标志将复位
		无 I2DAT 动作	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位
0x30	已发送 I2DAT 中的数据字节；已接收非 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	0	1	0	x	将发送停止条件；STO 标志将复位
		无 I2DAT 动作	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位
0x38	在 SLA+R/W 或数据字节中丢失仲裁	无 I2DAT 动作	0	0	0	x	I²C 总线将被释放；进入不可寻址从模式
		无 I2DAT 动作	1	0	0	x	当总线变为空闲时发送起始条件

表 21.18 主接收器模式

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x08	已发送起始条件	装入 SLA+R	x	0	0	x	将发送 SLA+R，接收 ACK 位
0x10	已发送重复起始条件	装入 SLA+R	x	0	0	x	同上
		装入 SLA+W	x	0	0	x	将发送 SLA+W，I²C 将切换到 MST/TRX 模式
0x38	在非 ACK 位中丢失仲裁	无 I2DAT 动作	0	0	0	x	I²C 总线将被释放；I²C 模块将进入从模式
		无 I2DAT 动作	1	0	0	x	当总线恢复空闲后发送起始条件
0x40	已发送 SLA+R；已接收 ACK	无 I2DAT 动作	0	0	0	0	将接收数据字节；返回非 ACK 位
		无 I2DAT 动作	0	0	0	1	将接收数据字节；返回 ACK 位
0x48	已发送 SLA+R；已接收非 ACK	无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	0	1	0	x	将发送停止条件；STO 标志将复位
		无 I2DAT 动作	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位
0x50	已接收数据字节；已返回 ACK	读数据字节	0	0	0	0	将接收数据字节，返回非 ACK 位
		读数据字节	0	0	0	1	将接收数据字节；返回 ACK 位
0x58	已接收数据字节；已返回非 ACK	读数据字节	1	0	0	x	将发送重复起始条件
		读数据字节	0	1	0	x	将发送停止条件；STO 标志将复位
		读数据字节	1	1	0	x	将发送停止条件，然后发送起始条件；STO 标志将复位

表 21.19 从接收器模式

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x60	已接收自身 SLA+W; 已返回 ACK	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x68	主控器时在 SLA+R/W 中丢失仲裁;已接收自身 SLA+W,已返回 ACK	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x70	已接收通用调用地址 (0x00); 已返回 ACK	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x78	主控器时在 SLA+R/W 中丢失仲裁; 已接收通用调用地址;已返回 ACK	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x80	前一次寻址使用自身从地址; 已接收数据字节; 已返回 ACK	读数据字节	x	0	0	0	将接收数据字节并返回非 ACK 位
		读数据字节	x	0	0	1	将接收数据字节并返回 ACK 位
0x88	前一次寻址使用自身 SLA 地址; 已接收数据字节; 已返回非 ACK	读数据字节	0	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 或通用调用地址
		读数据字节	0	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=1, 将识别通用调用地址
		读数据字节	1	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 或通用调用地址; 当总线空闲后发送起始条件
		读数据字节	1	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=1, 将识别通用调用地址; 当总线空闲后发送起始条件
0x90	前一次寻址使用通用调用; 已接收数据字节; 已返回 ACK	读数据字节	x	0	0	0	将接收数据字节并返回非 ACK 位
		读数据字节	x	0	0	1	将接收数据字节并返回 ACK 位
0x98	前一次寻址使用通用调用; 已接收数据字节; 已返回非 ACK	读数据字节	0	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 或通用调用地址
		读数据字节或	0	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=1, 将识别通用调用地址
		读数据字节或	1	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 或通用调用地址; 当总线空闲后发送起始条件
		读数据字节	1	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=1, 将识别通用调用地址; 当总线空闲后发送起始条件

续表 21.19

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0xA0	当使用 SLV/REC 或 SLV/TRX 静态寻址时,接收到停止条件或重复的起始条件	无 STDAT 动作	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		无 STDAT 动作	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		无 STDAT 动作	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		无 STDAT 动作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址；当总线空闲后发送起始条件

表 21.20 从发送器模式

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0xA8	已接收自身 SLA+R；已返回 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xB0	主控器时在 SLA+R/W 中丢失仲裁；已接收自身 SLA+R，已返回 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xB8	已发送 I2DAT 中数据字节；已接收 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xC0	已发送 I2DAT 中数据字节；已接收非 ACK	无 I2DAT 动作或	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		无 I2DAT 动作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		无 I2DAT 动作或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		无 I2DAT 动作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址；当总线空闲后发送起始条件
0xC8	已发送 I2DAT 中最后的数据字节（AA=0）；已接收 ACK	无 I2DAT 动作或	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		无 I2DAT 动作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		无 I2DAT 动作或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		无 I2DAT 动作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址；当总线空闲后发送起始条件

21.8.5 各种不同的状态

有 2 个I2STAT代码与定义的I²C硬件状态不对应（见表 21.21），下面将对这两种状态进行讨论：

21.8.5.1 I2STAT= 0xF8

该状态码表示没有任何可用的相关信息，因为串行中断标志 SI 还没有被置位。这种情况在其它状态和 I²C 模块还未开始执行串行传输之间出现。

21.8.5.2 I2STAT= 0x00

该状态代码表示一个总线错误在 I²C 串行传输过程中出现。当起始或停止条件出现在格式帧的非法位置上时产生总线错误。这些非法位置是指在串行传输过程中的地址字节、数据字节或应答位。当外部干扰影响到内部 I²C 模块信号时也会产生总线错误。总线错误出现时 SI 置位。要从总线错误中恢复，STO 标志必须置位，SI 必须被清除。这使得 I²C 模块进入“非寻址的”从机模式（已定义的状态）并清除 STO 标志（I2CON 中的其它位不受影响）。SDA 和 SCL 线被释放（不发送停止条件）。

表 21.21 各种不同的状态

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0xF8	无可用的相关状态信息；SI=0	无 I2DAT 动作	无 I2CON 动作				等待或执行当前传输
0x00	由于非法的起始或停止条件，在 MST 或选择的从机模式中会出现总线错误。当外部干扰使 I²C 模块进入一个未定义的状态时也出现 0x00 状态。	无 I2DAT 动作	0	1	0	x	只有 MST 或寻址的 SLV 模式中的内部硬件受影响。所有情况下，总线被释放、I²C 模块切换到非寻址的 SLV 模式。STO 复位。

21.8.6 一些特殊的情况

I²C 硬件可以处理出现在串行传输过程中的以下特殊情况。

21.8.7 两个主机同时启动重复起始条件

在主发送器或主接收器模式下可能产生重复起始条件。如果此时另一个主机同时产生重复起始条件将出现一种特殊情况（见图 21.14）。在出现这种情况之前，任何一个主机都不会丢失仲裁，因为它们都发送相同的数据。

如果 I²C 硬件在产生重复起始条件之前在 I²C 总线上检测到重复起始条件，它将释放总线，并且不产生中断请求。如果另一个主机通过产生停止条件来释放总线，则 I²C 模块将发送一个正常的起始条件（状态 0x08），并开始启动重新尝试完整的串行数据传输。

21.8.8 仲裁丢失后的数据传输

仲裁可能在主发送器和主接收器模式中丢失（见图 21.8）。仲裁丢失通过I2STAT中的下列状态代码来指示：0x38, 0x68, 0x78 和 0xB0（见图 21.10和图 21.11）。

如果 I2CON 中的 STA 标志被服务这些状态的程序置位，则当总线再次空闲时，发送起始条件（状态 0x08），不 受 CPU 的影响，并开始重新尝试完整的串行数据传输。

21.8.9 强制访问I²C总线

在某些应用中，非控制源有可能造成总线挂起。在这些情况下，干扰、总线的暂时中断或 SDA 和 SCL 之间的暂时短路都会引发问题的产生。

如果非控制源产生一个多余的起始条件或屏蔽一个停止条件，则 I²C 总线一直处于忙状态。如果 STA 标志被设置且在相应的时间内未访问总线，那么有可能强制访问 I²C 总线。这可通过在 STA 标志仍被设置时置位 STO 标志来实现。不发送停止条件。I²C 的硬件操作就好像是接收到停止条件一样，可以发送起始条件。STO 标志通过硬件清除。

21.8.10 SCL或SDA低电平妨碍I²C总线的操作

如果 SDA 或 SCL 被一个非控制源拉低，将出现 I²C 总线挂起。如果 SCL 线被总线上的器件妨碍（拉低），则不能进行更进一步的串行传输，并且 I²C 硬件也无法解决此类问题。这时，问题必须由将 SCL 线拉低的器件来解决。

如果SDA线被总线上的另一个器件妨碍（例如从机器件不能位同步），那么可通过向SCL线发送额外的时钟脉冲来解决（见图 21.16）。STA标志置位时I²C硬件发送额外的时钟脉冲，但不会产生起始条件，因为I²C总线空闲时SDA线被拉低。每当在SCL线上产生 2 个额外的时钟脉冲后，I²C硬件就尝试产生一个起始条件。当SDA线最终被释放时，发送正常的起始条件，进入 0x08 状态，并继续串行传输。

当 SDA 被干扰（拉低）时，若出现强制总线访问或重复发送起始条件，则 I²C 硬件如上所述执行相同的操作。每一种情况下，成功发送起始条件后进入 0x08 状态，继续进行正常的串行传输。注意 CPU 不参与此类总线挂起问题的解决。

21.8.11 总线错误

当起始或停止条件出现在格式帧的非法位置上时出现总线错误。非法位置是指串行传输过程中的地址字节、数据位或应答位。

当I²C硬件作为主机或被寻址的从机处理串行传输时，它仅对总线错误有反应。检测到总线错误时，I²C模块立即切换成非寻址的从机模式，释放SDA和SCL线，设置中断标志，并将 0x00 装入状态寄存器。该状态代码可用作状态服务程序的向量，尝试再次终止串行传输或仅从错误条件中恢复，如表 21.21所示。

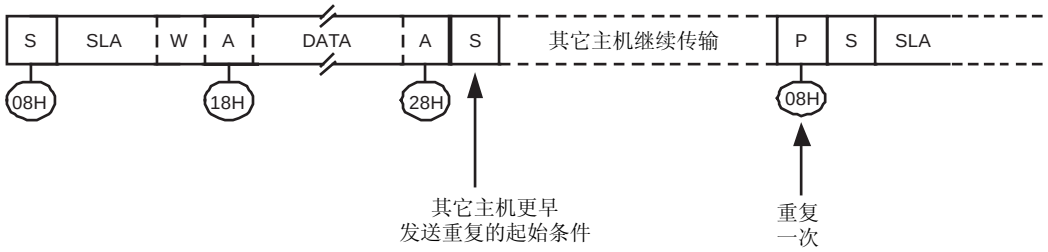


图 21.14 两个主机同时发送重复起始条件

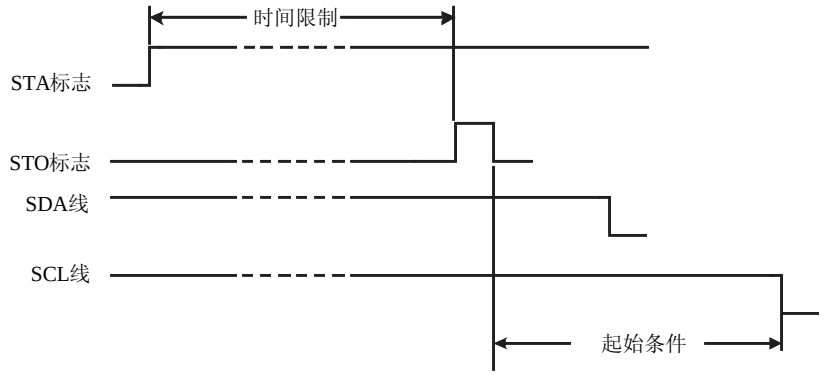
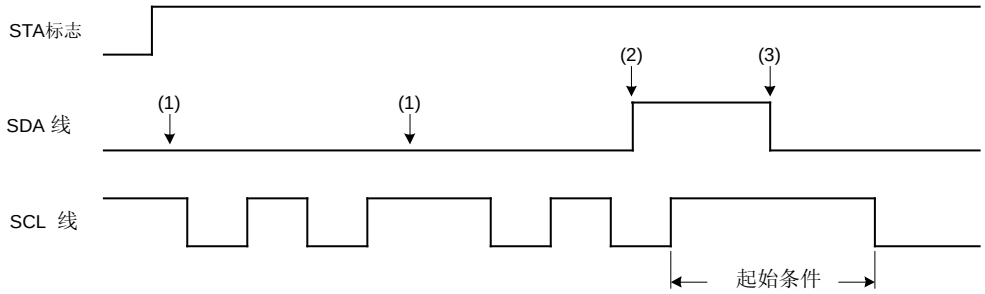


图 21.15 强制访问忙 I²C 总线



1. 发送起始条件失败。
2. SDA 线被释放。
3. 成功发送起始条件；进入 08H 状态。

图 21.16 从 SDA 低电平造成的总线操作干扰中恢复

21.8.12 I²C状态服务程序

本节提供了不同 I²C 状态服务程序必须执行的操作，它们包括：

- 复位后 I²C 模块的初始化
- I²C 中断服务
- 26 个状态服务程序支持 4 种 I²C 操作模式

21.8.12.1 初始化

在初始化示例中，I²C 模块可工作在主机和从机模式。每种模式下缓冲区都可用于发送和接收。初始化程序执行以下功能：

- I2ADR 装入器件自身的从机地址和通用调用位(GC)
- I²C 中断使能，设置中断优先级位
- 从机模式通过同时设置 I2CON 寄存器中的 I2EN 和 AA 位来使能，串行时钟频率（主机模式）由装入 I2CON 中 CR0 和 CR1 的值来定义。主机程序必须从主程序开始执行。

I²C 硬件开始在 I²C 总线上检查自身的从机地址和通用调用位。一旦检测到通用调用或自身从地址，请求中断，相应的状态信息装入 I2STAT。

21.8.12.2 I²C中断服务

当进入 I²C 中断时，I2STAT 包含一个状态代码，用来识别要执行的 26 个状态服务中的其中一个。

21.8.12.3 状态服务程序

每个状态程序都是 I²C 中断程序的组成部分，分别用来处理 26 种状态。

21.8.12.4 实际应用中的状态服务

状态服务程序显示了响应 26 个 I²C 状态代码必须执行的典型操作。如果 4 种 I²C 操作模式中的一个或多个未使用，则未使用的模式的相关的状态服务可被忽略，只要小心处理那些不会出现的状态即可。

在应用中，可能需要在 I²C 操作过程中执行一些超时处理，来限制不起作用的总线或丢失的服务程序。

21.9 软件示例

21.9.1 初始化程序

将 I²C 接口初始化用作从机和/或主机的例子。

1. 将自身的从机地址装入 I2ADR，使能通用调用识别（如果需要）。
2. 使能 I²C 中断。
3. 写 0x44 到 I2CONSET 来置位 I2EN 和 AA 位，使能从机功能。对于主机功能，写 0x40 到 I2CONSET。

21.9.2 启动主机发送功能

通过建立缓冲区、指针和数据计数器来开始主机发送操作，然后启动。

1. 初始化主机数据计数器。
2. 建立数据将被发送到的从机地址，并且添加写位。
3. 写 0x20 到 I2CONSET 来置位 STA 位。
4. 在主机发送缓冲区内建立要发送的数据。
5. 初始化主机数据计数器来匹配正在发送的信息长度。
6. 退出。

21.9.3 启动主机接收功能

通过建立缓冲区、指针和数据计数器来开始主机接收操作，然后启动。

1. 初始化主机数据计数器。
2. 建立数据将被发送到的从机地址，并且添加读位。
3. 写 0x20 到 I2CONSET 来置位 STA 位。
4. 建立主机接收缓冲区。
5. 初始化主机数据计数器来匹配接收到的信息长度。
6. 退出。

21.9.4 I²C中断程序

确定 I²C 状态和要使用的状态程序。

1. 从 I2STA 中读出 I²C 的状态。
2. 使用状态值跳转到 26 个可能状态程序中的一个。

21.9.5 非模式指定的状态

21.9.5.1 状态：0x00

总线错误。进入非寻址的从机模式并释放总线。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.6 主机状态

状态 08 和状态 10 适用于主发送和主接收模式。R/W 位决定下一个状态是否在主发送模式或主接收模式中。

21.9.6.1 状态：0x08

已发送起始条件。将发送从机地址＋R/W 位和接收 ACK 位。

1. 向 I2DAT 写入从机地址和 R/W 位。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立主发送模式数据缓冲区。
5. 建立主接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出。

21.9.6.2 状态：0x10

已发送重复起始条件。将发送从机地址＋R/W 位和接收 ACK 位。

1. 向 I2DAT 写入从机地址和 R/W 位。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立主发送模式数据缓冲区。
5. 建立主接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出。

21.9.7 主机发送器状态

21.9.7.1 状态：0x18

前面的状态是状态 8 或状态 10，已发送从机地址＋写，已接收 ACK。将发送第一个数据字节和接收 ACK 位。

1. 将主机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 主机发送缓冲区指针加 1。
5. 退出。

21.9.7.2 状态：0x20

已发送从机地址＋写，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.7.3 状态：0x28

已发送数据，已接收 ACK。如果发送的数据是最后一个数据字节则发送一个停止条件，否则发送下一个数据字节。

1. 主机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
2. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 退出。
5. 向 I2DAT 装入主机发送缓冲区的下一个数据字节。
6. 写 0x04 到 I2CONSET 来置位 AA 位。
7. 写 0x08 到 I2CONCLR 来清除 SI 标志。
8. 主机发送缓冲区指针加 1。
9. 退出。

21.9.7.4 状态：0x30

已发送数据，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.7.5 状态：0x38

仲裁在传输从机地址＋写或数据过程中已丢失。总线已被释放且进入非寻址的从机模式。当总线再次空闲时将发送一个新的起始条件。

1. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.8 主接收器状态

21.9.8.1 状态：0x40

前面的状态是状态 08 或状态 10，已发送从机地址＋读，已接收到 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.8.2 状态：0x48

已发送从机地址＋读，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.8.3 状态：0x50

已接收数据，已返回 ACK。将从 I2DAT 读取数据。将接收其它的数据。如果这是最后一个数据字节，则将返回非应答，否则将返回 ACK。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 主机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
3. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
4. 退出。
5. 写 0x04 到 I2CONSET 来置位 AA 位。
6. 写 0x08 到 I2CONCLR 来清除 SI 标志。
7. 主机接收缓冲区指针加 1。
8. 退出。

21.9.8.4 状态：0x58

已接收到数据，已返回非应答。将从 I2DAT 中读取数据和发送停止条件。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 退出。

21.9.9 从机接收器状态

21.9.9.1 状态：0x60

已接收到自身从机地址＋写和返回 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 建立从接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

21.9.9.2 状态：0x68

用作总线主机时仲裁已在传输从机地址和 R/W 位时丢失。已接收到自身从机地址＋写和返回 ACK。将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 建立从接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出.

21.9.9.3 状态：0x70

已接收到通用调用位和返回 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 建立从接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

21.9.9.4 状态：0x78

用作总线主机时仲裁已在传输从机地址＋R/W 位时丢失。已接收到通用调用和返回 ACK。将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 建立从接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

21.9.9.5 状态：0x80

以前用自身从机地址寻址。已接收到数据并返回 ACK。将读取其它的数据。

1. 读取 I2DAT 的数据字节，存放到从机接收缓冲区。
2. 从机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
3. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
4. 退出。
5. 写 0x04 到 I2CONSET 来置位 AA 位。
6. 写 0x08 到 I2CONCLR 来清除 SI 标志。
7. 从机接收缓冲区指针加 1。
8. 退出。

21.9.9.6 状态：0x88

以前用自身从机地址寻址。已接收到数据并返回了非应答。接收到的数据将不保存。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.9.7 状态：0x90

以前用通用调用寻址。已接收到数据和返回了 ACK。将保存接收到的数据。只有将要接收的第一个数据字节带有 ACK，其它数据字节都是非应答。

1. 读取 I2DAT 的数据字节，存放到从机接收缓冲区。
2. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
3. 退出。

21.9.9.8 状态：0x98

以前用通用调用寻址。已接收到数据和返回非应答。接收到的数据将不保存。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.9.9 状态：0xA0

接收到停止条件或重复起始条件，但仍作为从机寻址。将不保存数据。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.10 从机接收器状态

21.9.10.1 状态：0xA8

已接收到自身从机地址＋读和返回 ACK。将发送数据和接收 ACK 位。

1. 将从机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立从发送模式数据缓冲区。
5. 从机发送缓冲区指针加 1。
6. 退出。

21.9.10.2 状态：0xB0

用作总线主机时仲裁已在传输从机地址和 R/W 位时丢失。已接收到自身从机地址＋读和返回 ACK。将发送数据和接收 ACK 位。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 将从机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立从发送模式数据缓冲区。
5. 从机发送缓冲区指针加 1。
6. 退出。

21.9.10.3 状态：0xB8

已发送数据和接收到 ACK。将发送数据和接收 ACK 位。

1. 将从机发送缓冲区的数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 从机发送缓冲区指针加 1。
5. 退出。

21.9.10.4 状态：0xC0

已发送数据，已接收到非应答。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

21.9.10.5 状态：0xC8

已发送最后一个数据字节和接收到 ACK。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

第22章 I²S接口

22.1 特性

I²S 总线为数字音频应用提供了一个标准的通信接口。

I²S 总线规范定义了一条 3 线串行总线，它含有 1 个数据、1 个时钟和 1 个字选择信号。基本的 I²S 连接具有一个主机（其总是为主机）和一个从机。LPC2400 上的 I²S 接口提供了各自独立的发送和接收通道，每条通道都可作为主机或从机操作。

- I²S 输入可在主机和从机模式下操作；
- I²S 输出可在主机和从机模式下操作，与 I²S 输入无关；
- 能够处理大小为 8, 16 和 32 位的字；
- 支持单声道和立体声道音频数据；
- 采样的频率范围（实际上）为 16kHz -48kHz（16, 22.05, 32, 44.1, 48kHz）；
- 在主机模式下的字选择周期可配置（I²S 输入和 I²S 输出各自独立）；
- 提供两个 8 字 FIFO 数据缓冲区，一个用于发送，一个用于接收；
- 当缓冲区深度(level)超过可编程的边界时产生中断请求；
- 两个 DMA 请求由可编程的缓冲区深度(level)控制。这两个 DMA 请求被连接到通用 DMA 模块；
- 控制复位、停止和静音选项分别用于 I²S 输入和 I²S 输出。

22.2 描述

I²S 分别通过发送和接收通道来执行串行数据的输出和输入。这些操作支持 NXP Inter IC Audio 格式的 8、16 和 32 位音频数据，以用于立体声道和单声道模式。配置、数据访问和控制由 APB 寄存器集来执行。数据流通过 8 字节深度的 FIFO 进行缓冲。

I²S 接收和发送状态可以在从模式或主模式下独立操作。在 I²S 模块中，这些模式之间的差别在于决定数据发送时序的字选择（WS）信号。在 WS 改变后，数据字在发送时钟的下一个下降沿上开始。在立体声道模式下，WS 为低时发送左声道数据，WS 为高时发送右声道数据。在单声道模式下，相同的数据发送两次，WS 为低时发送一次，WS 为高时再发送一次。

- 在主机模式下（ws_sel=0），字选择通过 9 位计数器内部产生。该计数器的半周期计数值可在控制寄存器中设置。
- 在从机模式下（ws_sel=1），字选择从相关的总线管脚输入。
- 当 I²S 总线有效时，字选择、接收时钟和发送时钟信号通过总线主机连续发送，而数据则通过发送器连续发送。
- 可通过发送和接收的停止或静音控制位来禁能 I²S。
- 停止位将禁止通过发送通道或接收通道访问 FIFO，并把发送通道置于静音模式下。
- 静音控制位将发送通道置于静音模式下。在静音模式中，发送通道 FIFO 正常操作，但输出被废除或由 0 替换。该位不影响接收通道，并且数据接收可正常进行。

22.3 管脚描述

表 22.1 管脚描述

管脚名称	类型	描述
I2SRX_CLK	输入/输出	接收时钟。时钟信号用于使接收通道上的数据传输同步。它由主机驱动从机接收。相对于 I²S 总线规范中的信号 SCK 而言。
I2SRX_WS	输入/输出	接收字选择。选择正在接收数据的通道。它由主机驱动从机接收。相对于 I²S 总线规范中的信号 WS 而言。 WS=0 表示通道 1（左通道）正在接收数据。 WS=1 表示通道 2（右通道）正在接收数据。
I2SRX_SDA	输入/输出	接收数据。串行数据，先接收 MSB。它由发送器驱动、接受器读取。相对于 I²S 总线规范中的信号 SD 而言。
I2STX_CLK	输入/输出	发送时钟。时钟信号用于使发送通道上的数据传输同步。它由主机驱动从机接收。相对于 I²S 总线规范中的信号 SCK 而言。
I2STX_WS	输入/输出	发送字选择。选择正在发送数据的通道。它由主机驱动从机接收。相对于 I²S 总线规范中的信号 WS 而言。 WS=0 表示数据正被发送到通道 1（左通道）。 WS=1 表示数据正被发送到通道 2（右通道）。
I2STX_SDA	输入/输出	发送数据。串行数据，先发送 MSB。它由发送器驱动、接收器读取。相对于 I²S 总线规范中的信号 SD 而言。

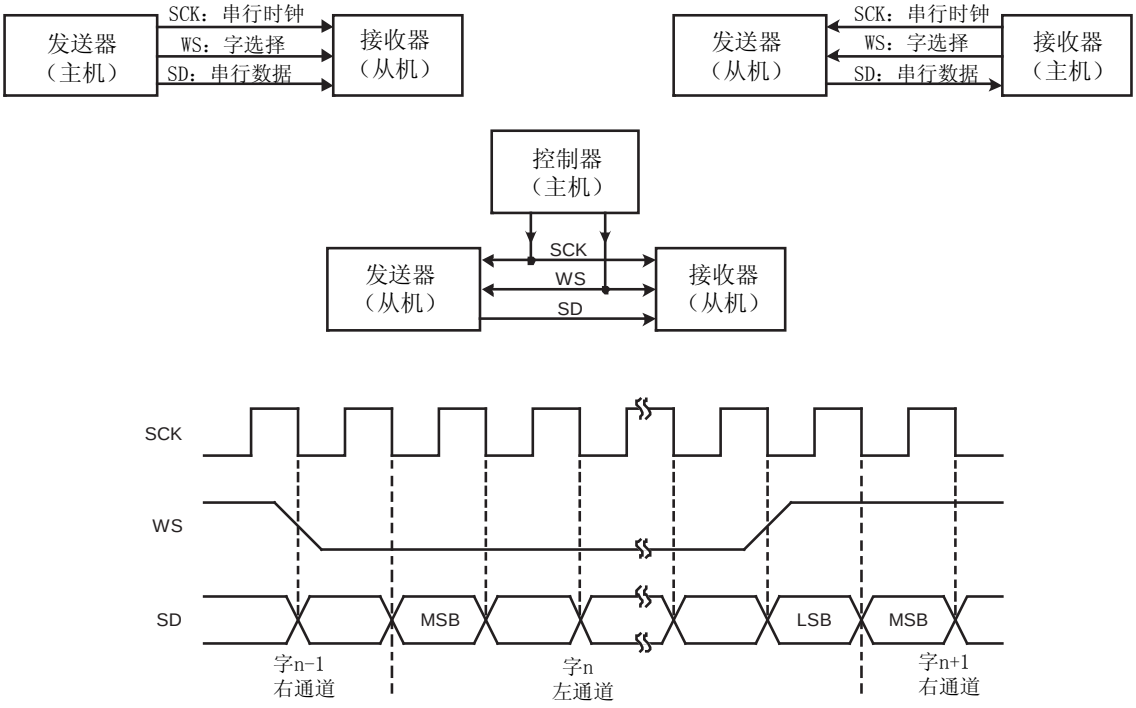


图 22.1 简单的 I²S 配置和总线时序

22.4 寄存器描述

表 22.2所示为I²S接口相关的寄存器及其功能汇总。表中给出了每个寄存器的详细内容。

表 22.2 I²S 寄存器映射

名称	描述	访问	复位值[1]	地址
I2SDAO	数字音频输出寄存器。含有 I²S 发送通道的控制位。	R/W		0xE008 8000
I2SDAI	数字音频输入寄存器。含有 I²S 接收通道的控制位。	R/W		0xE008 8004
I2STXFIFO	发送 FIFO。访问 8×32 位发送器 FIFO 的寄存器。	WO		0xE008 8008
I2SRXFIFO	接收 FIFO。访问 8×32 位接收器 FIFO 的寄存器。	RO		0xE008 800C
I2SSTATE	状态反馈寄存器。含有 I²S 接口的状态信息。	RO		0xE008 8010
I2SDMA1	DMA 配置寄存器 1。含有 DMA 请求 1 的控制信息。	R/W		0xE008 8014
I2SDMA2	DMA 配置寄存器 2。含有 DMA 请求 2 的控制信息。	R/W		0xE008 8018
I2SIRQ	中断请求控制寄存器。含有控制如何产生 I²S 中断请求的位。	R/W		0xE008 801C
I2STXRATE	发送位速率分频器。该寄存器确定 I²S 发送位速率，通过指定 PCLK 的分频值产生发送位时钟来完成。	R/W		0xE008 8020
I2SRXRATE	接收位速率分频器。该寄存器确定 I²S 接收位速率，通过指定 PCLK 的分频值产生接收位时钟来完成。	R/W		0xE008 8024

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

22.4.1 数字音频输出寄存器（I2SDAO – 0xE008 8000）

I2SDAO寄存器控制了I²S发送通道的操作。DAO寄存器中位的功能见表 22.3。

表 22.3 数字音频输出寄存器（I2SDAO – 地址 0xE008 8000）位描述

位	符号	值	描述	复位值
1:0	wordwidth		如下选择数据的字节数：	01
		00	8 位数据	
		01	16 位数据	
		10	保留，不使用该设置	
		11	32 位数据	
2	mono		该位为 1 时，数据格式为单声道；该位为 0 时，数据格式为立体声道。	0
3	stop		禁止访问 FIFO，将发送通道置于静音模式下。	0
4	reset		异步复位发送通道和 FIFO。	0
5	ws_sel		该位为 0 时在主模式下，该位为 1 时在从模式下。	1
14:6	ws_halfperiod		字选择半周期减 1，即，WS 64clk period->ws_halfperiod=31。	0x1F
15	mute		当正确（true）时，发送通道仅发送 0。	1

22.4.2 数字音频输入寄存器（I2SDAI – 0xE008 8004）

I2SDAI寄存器控制了I²S接收通道的操作。DAI寄存器中位的功能见表 22.4。

表 22.4 数字音频输入寄存器（I2SDAI – 地址 0xE008 8004）位描述

位	符号	值	描述	复位值
1:0	wordwidth	 00 01 10 11	如下选择数据的字节数： 8 位数据 16 位数据 保留，不使用该设置 32 位数据	01
2	mono		该位为 1 时，数据格式为单声道；该位为 0 时，数据格式为立体声道。	0
3	stop		禁止访问 FIFO，将发送通道置于静音模式下。	0
4	reset		异步复位发送通道和 FIFO。	0
5	ws_sel		该位为 0 时在主模式下，该位为 1 时在从模式下。	1
14:6	ws_halfperiod		字选择半周期减 1，即，WS 64clk period->ws_halfperiod=31。	0x1F
15	Unused		不使用。	1

22.4.3 发送FIFO寄存器（I2STXFIFO – 0xE008 8008）

I2STXFIFO寄存器可以访问发送FIFO，I2STXFIFO中位的功能见表 22.5。

表 22.5 发送 FIFO 寄存器（I2STXFIFO – 地址 0xE008 8008）位描述

位	符号	描述	复位值
31:0	I2STXFIFO	8×32 位发送 FIFO。	Level=0

22.4.4 接收FIFO寄存器（I2SRXFIFO – 0xE008 800C）

I2SRXFIFO寄存器可以访问接收FIFO，I2SRXFIFO中位的功能见表 22.6。

表 22.6 接收 FIFO 寄存器（I2SRXFIFO – 地址 0xE008 800C）位描述

位	符号	描述	复位值
31:0	I2SRXFIFO	8×32 位接收 FIFO。	Level=0

22.4.5 状态反馈寄存器（I2SSTATE – 0xE008 8010）

I2SSTATE寄存器提供有关I²S接口的状态信息。I2SSTATE寄存器的位描述见表 22.7。

表 22.7 状态反馈寄存器（I2SSTATE – 地址 0xE008 8010）位描述

位	符号	描述	复位值
0	irq	该位反映了出现接收中断或发送中断。	0
1	dmareq1	该位反映了出现接收或发送 DMA 请求 1。	0
2	dmareq2	该位反映了出现接收或发送 DMA 请求 2。	0
7:3	Unused	不使用。	0
15:8	rx_level	反映了当前接收 FIFO 的深度。	0
23:16	tx_level	反映了当前发送 FIFO 的深度。	0
31:24	-	保留，用户软件不应向保留位写入 1，从保留位读出的值未定义。	NA

22.4.6 DMA配置寄存器 1（I2SDMA1 – 0xE008 8014）

I2SDMA1 寄存器控制了DMA请求 1 的操作。I2SDMA1 寄存器中位的功能见表 22.8。有关DMA操作的详细内容请参考“通用DMA控制器”。

表 22.8 DMA 配置寄存器 1（I2SDMA1 – 地址 0xE008 8014）位描述

位	符号	描述	复位值
0	rx_dma1_enable	该位为 1 时，使能 DMA1 用于 I²S 接收。	0
1	tx_dma1_enable	该位为 1 时，使能 DMA1 用于 I²S 发送。	0
7:2	Unused	不使用。	0
15:8	rx__depth_dma1	设置 FIFO 深度，该深度在 DMA1 上触发接收 DMA 请求。	0
23:16	tx__depth_dma1	设置 FIFO 深度，该深度在 DMA1 上触发发送 DMA 请求。	0
31:24	-	保留，用户软件不应向保留位写入 1，从保留位读出的值未定义。	NA

22.4.7 DMA配置寄存器 2（I2SDMA2 – 0xE008 8018）

I2SDMA2 寄存器控制了DMA请求 2 的操作。I2SDMA2 寄存器中位的功能见表 22.9。

表 22.9 DMA 配置寄存器 2（I2SDMA2 – 地址 0xE008 8018）位描述

位	符号	描述	复位值
0	rx_dma2_enable	该位为 1 时，使能 DMA1 用于 I²S 接收。	0
1	tx_dma2_enable	该位为 1 时，使能 DMA1 用于 I²S 发送。	0
7:2	Unused	不使用。	0
15:8	rx__depth_dma2	设置 FIFO 深度，该深度在 DMA2 上触发接收 DMA 请求。	0
23:16	tx__depth_dma2	设置 FIFO 深度，该深度在 DMA2 上触发发送 DMA 请求。	0
31:24	-	保留，用户软件不应向保留位写入 1，从保留位读出的值未定义。	NA

22.4.8 中断请求控制寄存器（I2SIRQ – 0xE008 801C）

I2SIRQ寄存器控制了I²S中断请求的操作。I2SIRQ寄存器中位的功能见表 22.10。

表 22.10 中断请求控制寄存器（I2SIRQ – 地址 0xE008 801C）位描述

位	符号	描述	复位值
0	rx_Irq_enable	该位为 1 时，使能 I2S 接收中断。	0
1	tx_Irq_enable	该位为 1 时，使能 I2S 发送中断。	0
7:2	Unused	不使用。	0
15:8	rx__depth_Irq	设置产生中断请求的 FIFO 深度。	0
23:16	tx__depth_Irq	设置产生中断请求的 FIFO 深度。	0
31:24	-	保留，用户软件不应向保留位写入 1，从保留位读出的值未定义。	NA

22.4.9 发送时钟速率寄存器（I2STXRATE – 0xE008 8020）

I²S 发送器的位速率由 I2STXRATE 寄存器的值来决定。该值取决于所需的音频采样率，以及使用的数据大小和格式（立体/单声道）。例如 48kHz 采样率的 16 位数据立体声道需要的位速率为：48,000×16×2＝1.536MHz。

表 22.11 发送时钟速率寄存器（I2STXRATE – 地址 0xE008 8020）位描述

位	符号	描述	复位值
9:0	tx_rate	I2S 发送位速率。该值加 1 用于分频 PCLK 来产生发送位时钟。10 位分频支持 PCLK 速率范围内的 I²S 速率范围。	0
15:10	Unused	不使用。	0

22.4.10 接收时钟速率寄存器（I2SRXRATE – 0xE008 8024）

I²S 接收器的位速率由 I2SRXRATE 寄存器的值来决定。该值取决于音频采样率，以及使用的数据大小和格式。它的计算与 I2STXRATE 寄存器相同。

表 22.12 接收时钟速率寄存器（I2SRXRATE – 地址 0xE008 8024）位描述

位	符号	描述	复位值
9:0	rx_rate	I2S 接收位速率。该值加 1 用于分频 PCLK 来产生接收位时钟。10 位分频支持 PCLK 速率范围内的 I²S 速率范围。	0
15:10	Unused	不使用。	0

22.5 I²S发送和接收接口

I²S 接口可发送和接收 8、16 或 32 位立体声道或单声道音频信息。某些 I²S 实现的细节包括：

- 当 FIFO 为空时，发送通道将重复发送相同的数据直至新的数据被写入 FIFO。
- 当静音被选中（true）时，发送数据值 0。
- 当单声道为错误时，两个连续的数据字分别是左声道和右声道的数据。
- 数据字长度由配置寄存器中字宽度的值决定。接收通道和发送通道有各自字宽度的值。

  －0：字被看作为含有 4 个 8 位的数据字。

  －1：字被看作为含有 2 个 16 位的数据字。

  －3：字被看作为含有 1 个 32 位的数据字。
- 当发送 FIFO 含有不足够的数据时，发送通道将重复发送最后的数据直至新的数据可用。当微处理器或 DMA 在某些时候不能足够快地提供新数据时可能会出现这种情况。由于在新数据中存在这种延时，因此需要填充间隙，通过连续发送最后的采样来完成该操作。数据不能被屏蔽（muted），因为这将会在声音上产生明显而不合乎需要的效果。
- 发送通道和接收通道仅处理 32 位对齐的字，数据程序块（chunk）必须被省略一部分或将其扩展为 32 位的倍数。

在数据宽度或模式之间切换时，I2S 必须通过控制寄存器中的复位位进行复位来确保正确的同步操作。建议同时置位停止位直至有足够的数据被写入发送 FIFO。需要注意的是，在停止时数据输出被屏蔽（muted）。

所有访问FIFO的数据为 32 位。图 22.2所示为可能的数据序列。

FIFO 中的数据采样包括：

- 1×32 位，在 8 或 16 位立体声道模式下。
- 1×32 位，在单声道模式下。
- 2×32 位，第一个为左声道数据、第二个为右声道数据，在 32 位立体声道模式下。

数据在 WS 下降沿后从发送 FIFO 中读出，它将在 WS 上升沿后被传输到发送时钟域。在 WS 的下一个下降沿上，左声道数据将被载入移位寄存器并发送，在 WS 的下一个上升沿上，右声道数据被载入并发送。

接收通道将在 WS 改变后开始接收数据。字选择(WS)变低时接收数据被期望为左声道数据，而 WS 为高时接收数据被期望为右声道数据。当位计数器已到达字宽度设置的极限时停止接收。在 WS 的下一次改变时，接收的数据将存储在正确的保存寄存器中。当可以使用完整的数据时，它将被写入接收 FIFO 中。

22.6 FIFO控制器

发送和接收数据的处理通过 FIFO 控制器来执行，该 FIFO 控制器产生 2 个 DMA 请求和 1 个中断请求。控制器由一组比较器组成，这些比较器将 FIFO 触发深度与寄存器中设置的深度相比较。触发深度比较器（level comparator）的当前状态可在 APB 状态寄存器中看到。

表 22.13 FIFO 深度比较的条件

触发深度比较	条件
dmareq_tx_1	tx_depth_dma1>= tx_level
dmareq_rx_1	rx_depth_dma1<= rx_level
dmareq_tx_2	tx_depth_dma2>= tx_level
dmareq_rx_2	rx_depth_dma2<= rx_level
irq_tx	tx_depth_irq>=tx_level
irq_rx	rx_depth_irq<=rx_level

当触发深度检测为真实且使能时，系统发出信号。

表 22.14 DMA 和中断请求产生

系统发出信号	条件
irq	(irq_rx & rx_irq_enable) | (irq_tx & tx_irq_enable)
dmareq[0]	(dmareq_tx_1 & tx_dma1_enable) | (dmareq_rx_1 & rx_dma1_enable)
dmareq[1]	(dmareq_tx_2 & tx_dma2_enable) | (dmareq_rx_2 & rx_dma2_enable)

表 22.15 I2SSTATE 寄存器中的状态反馈

状态反馈	状态
irq	irq_rx | irq_tx
dmareq1	(dmareq_tx_1 | dmareq_rx_1)
dmareq2	(dmareq_rx_2 | dmareq_tx_2)

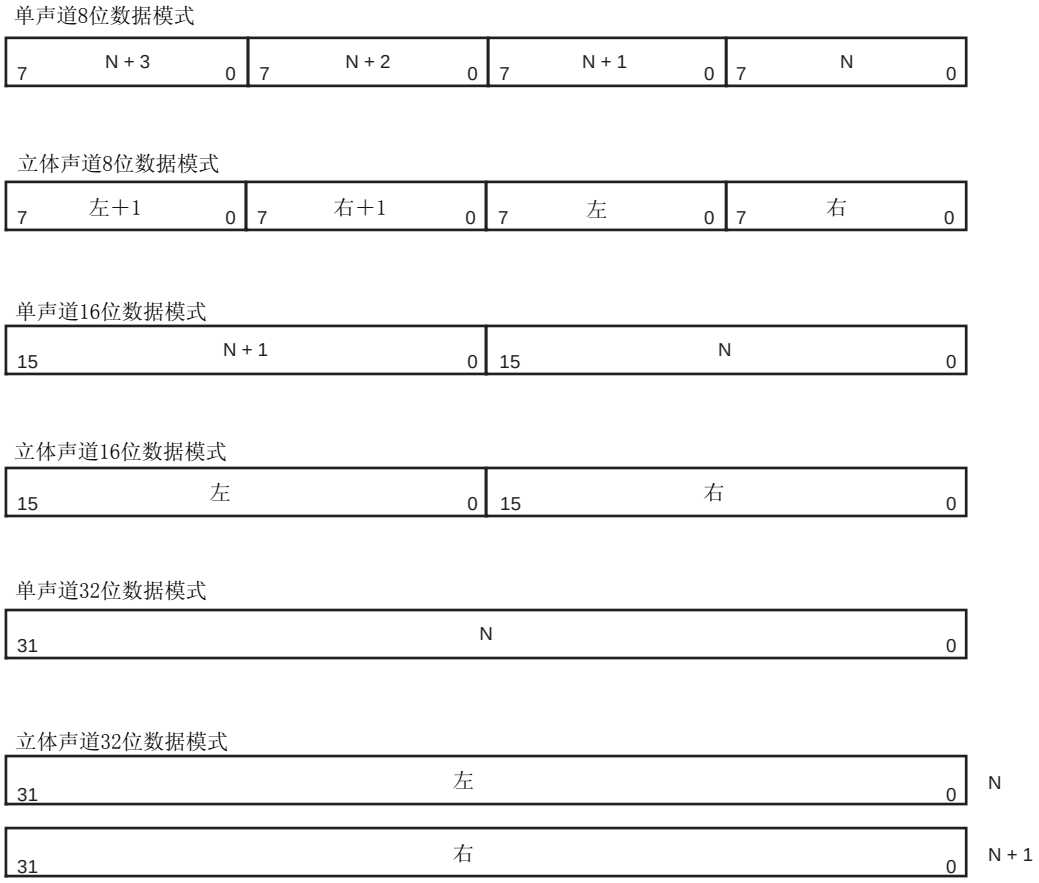


图 22.2 不同 I²S 模式的 FIFO 内容

第23章 定时器 0/1/2/3

23.1 特性

说明：除了外设基址之外，4 个定时器/计数器完全相同。4 个定时器最少有 2 个捕获输入和 2 个匹配输出，每个定时器可以选择含有几个管脚。定时器 1 引出了第 3 个匹配输出，而定时器 2 和 3 引出了全部 4 个匹配输出。

- 32 位的定时器/计数器，带有一个可编程的 32 位预分频器。
- 计数器或定时器操作。
- 每个定时器包含多达 4 个 32 位的捕获通道，可以在输入信号变化时捕捉定时器的瞬时值。捕获事件也可以选择产生中断。
- 4 个 32 位匹配寄存器，允许执行以下操作：
  - 连续工作，在匹配时可选择产生中断。
  - 在匹配时停止定时器运行，可选择产生中断。
  - 在匹配时复位定时器，可选择产生中断。
- 有多达 4 个外部输出与匹配寄存器相对应，这些输出具有以下功能：
  - 匹配时设为低电平。
  - 匹配时设为高电平。
  - 匹配时翻转。
  - 匹配时不执行任何操作。

23.2 应用

- 时间间隔定时器，用来计数内部事件
- 脉宽解调器（经由捕获输入）
- 自由运行的定时器

23.3 描述

定时器/计数器用来计数外设时钟（PCLK）或外部时钟的周期，可以选择在规定的时间处产生中断或执行其他操作，由 4 个匹配寄存器的值决定。它也包含 4 个捕获输入，用来在输入信号变化时捕捉定时器的瞬时值，也可以选择产生中断。

23.4 管脚描述

表 23.1对每个定时器/计数器的相关管脚进行了总结。

表 23.1 定时器/计数器管脚描述

管脚	类型	描述
CAP0[1:0] CAP1[1:0] CAP2[1:0] CAP3[1:0]	输入	捕获信号－捕获管脚的变化可以配置成用定时器计数器的值来装载其中一个捕获寄存器，也可以选择来产生一个中断。有很多管脚可以用作捕获功能。当有多个管脚可以被选择用作 TIMER0/1 通道的一个捕获输入时，端口编号最小的管脚被使用。 定时器/计数器可以选择一个捕获信号作为时钟源（而不是获得的PCLK时钟）。详情请见23.5.3 节“计数控制寄存器（T[0/1/2/3]CTCR - 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070）”。
MAT0[1:0] MAT1[2:0] MAT2[3:0] MAT3[3:0]	输出	外部匹配输出 0/1－当匹配寄存器 0/1（MR3:0）的值与定时器计数器（TC）相等时，相应的输出可以翻转、变低、变高或不执行任何操作。外部匹配寄存器（EMR）控制着输出的功能。并行的多个管脚可以被选择用作匹配输出功能。

23.4.1 多个CAP和MAT管脚

软件可以在管脚选择寄存器中选择多个管脚用作 CAP 或 MAT 功能，这些将在封装、管脚配置和多路复用等各章中描述。当有多个管脚被选择用作一个 MAT 输出时，所有管脚的驱动完全相同。当多个管脚被选择用作一个 CAP 输入时，使用最小端口编号的管脚。

23.5 管脚描述

每个定时器/计数器包含的寄存器如表 23.2所示（“复位值”只是指被使用位的数据；不包括保留位的内容）。详情请见下面的描述。

表 23.2 定时器/计数器 0-3 的寄存器映射

通用名称	描述	访问	复位值[1]	TIMERn 寄存器/名称&地址
IR	中断寄存器。可向 IR 写入相应的值来清除中断。也可读 IR 来确定哪个可用的中断源被挂起。	R/W	0	T0IR - 0xE000 4000 T1IR - 0xE000 8000 T2IR - 0xE007 0000 T3IR - 0xE007 4000
TCR	定时器控制寄存器。TCR 用来控制定时器计数器功能。定时器计数器可以通过 TCR 来禁能或复位。	R/W	0	T0TCR - 0xE000 4004 T1TCR - 0xE000 8004 T2TCR - 0xE007 0004 T3TCR - 0xE007 4004
TC	定时器计数器。32 位的 TC 每隔(PR+1)个 PCLK 周期递增一次。TC 通过 TCR 来控制。	R/W	0	T0TC - 0xE000 4008 T1TC - 0xE000 8008 T2TC - 0xE007 0008 T3TC - 0xE007 4008
PR	预分频寄存器。当预分频计数器的值与这个寄存器的值相等时，下个时钟将增加 TC 和清除 PC。	R/W	0	T0PR - 0xE000 400C T1PR - 0xE000 800C T2PR - 0xE007 000C T3PR - 0xE007 400C

续表 23.2

通用名称	描述	访问	复位值[1]	TIMERn 寄存器/名称&地址
PC	预分频计数器。32 位的 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当达到了 PR 的值时，TC 增加，PC 被清除。可以通过总线接口来观察和控制 PC。	R/W	0	T0PC - 0xE000 4010 T1PC - 0xE000 8010 T2PC - 0xE007 0010 T3PC - 0xE007 4010
MCR	匹配控制寄存器。MCR 用来控制在匹配出现时是否产生中断和 TC 是否复位。	R/W	0	T0MCR - 0xE000 4014 T1MCR - 0xE000 8014 T2MCR - 0xE007 0014 T3MCR - 0xE007 4014
MR0	匹配寄存器 0　MR0 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。	R/W	0	T0MR0 - 0xE000 4018 T1MR0 - 0xE000 8018 T2MR0 - 0xE007 0018 T3MR0 - 0xE007 4018
MR1	匹配寄存器 1。见 MR0 的描述。	R/W	0	T0MR1 - 0xE000 401C T1MR1 - 0xE000 801C T2MR1 - 0xE007 001C T3MR1 - 0xE007 401C
MR2	匹配寄存器 2。见 MR0 的描述。	R/W	0	T0MR2 - 0xE000 4020 T1MR2 - 0xE000 8020 T2MR2 - 0xE007 0020 T3MR2 - 0xE007 4020
MR3	匹配寄存器 3。见 MR0 的描述。	R/W	0	T0MR3 - 0xE000 4024 T1MR3 - 0xE000 8024 T2MR3 - 0xE007 0024 T3MR3 - 0xE007 4024
CCR	捕获控制寄存器。CCR 控制捕获输入的哪个边沿出现时装载捕获寄存器以及在捕获出现时是否产生中断。	R/W	0	T0CCR - 0xE000 4028 T1CCR - 0xE000 8028 T2CCR - 0xE007 0028 T3CCR - 0xE007 4028
CR0	捕获寄存器 0。当在 CAPn.0(CAP0.0 或 CAP1.0)输入上产生捕获事件时，CR0 装载 TC 的值。	R/O	0	T0CR0 - 0xE000 402C T1CR0 - 0xE000 802C T2CR0 - 0xE007 002C T3CR0 - 0xE007 402C
CR1	捕获寄存器 1。见 CR0 描述。	RO	0	T0CR1 - 0xE000 4030 T1CR1 - 0xE000 8030 T2CR1 - 0xE007 0030 T3CR1 - 0xE007 4030
CR2	捕获寄存器 2。见 CR0 描述。	RO	0	T0CR2 - 0xE000 4034 T1CR2 - 0xE000 8034 T2CR2 - 0xE007 0034 T3CR2 - 0xE007 4034

续表 23.2

通用名称	描述	访问	复位值[1]	TIMERn 寄存器/名称&地址
CR3	捕获寄存器 3。见 CR0 描述。	RO	0	T0CR3 - 0xE000 4038 T1CR3 - 0xE000 8038 T2CR3 - 0xE007 0038 T3CR3 - 0xE007 4038
EMR	外部匹配寄存器。EMR 控制外部匹配管脚 MATn.0-3（MAT0.0-3 和 MAT1.0-3）。	R/W	0	T0EMR - 0xE000 403C T1EMR - 0xE000 803C T2EMR - 0xE007 003C T3EMR - 0xE007 403C
CTCR	计数控制寄存器。CTCR 在定时器模式和计数器模式之间进行选择，在计数器模式中选择计数的信号和边沿。	R/W	0	T0CTCR - 0xE000 4070 T1CTCR - 0xE000 8070 T2CTCR - 0xE007 0070 T3CTCR - 0xE007 4070

[1] 保留位只反映了使用位的值，不包括保留位的内容。

23.5.1 中断寄存器（T[0/1/2/3]IR - 0xE000 4000, 0xE000 8000, 0xE007 0000, 0xE007 4000）

中断寄存器包含 4 个位用于匹配中断，4 个位用于捕获中断。如果有中断产生，IR 中的对应位会置位，否则为 0。向对应的 IR 位写入 1 会复位中断。写入 0 无效。

表 23.3 中断寄存器（T[0/1/2/3]IR – 地址：0xE000 4000, 0xE000 8000, 0xE007 0000, 0xE007 4000）位描述

IR	功能	描述	复位值
0	MR0 中断	匹配通道 0 的中断标志	0
1	MR1 中断	匹配通道 1 的中断标志	0
2	MR2 中断	匹配通道 2 的中断标志	0
3	MR3 中断	匹配通道 3 的中断标志	0
4	CR0 中断	捕获通道 0 事件的中断标志	0
5	CR1 中断	捕获通道 1 事件的中断标志	0
6	CR2 中断	捕获通道 2 事件的中断标志	0
7	CR3 中断	捕获通道 3 事件的中断标志	0

23.5.2 定时器控制寄存器（T[0/1/2/3]CR - 0xE000 4004, 0xE000 8004, 0xE007 0004, 0xE007 4004）

定时器控制寄存器（TCR）用来控制定时器/计数器的操作。

表 23.4 定时器控制寄存器（TCR, TIMERn: TnTCR – 地址：0xE000 4004, 0xE000 8004, 0xE007 0004, 0xE007 4004）位描述

位	符号	描述	复位值
0	计数器使能	为 1 时，定时器计数器和预分频计数器使能计数。为 0 时，计数器被禁止。	0
1	计数器复位	为 1 时，定时器计数器和预分频计数器在 pclk 的下一个上升沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
7:2	-	保留，用户软件不应该向保留位写 1。从保留位读出的值未定义。	NA

23.5.3 计数控制寄存器（T[0/1/2/3]CTCR - 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070）

计数控制寄存器（CTCR）用来在定时器模式和计数器模式之间进行选择，在计数器模式中选择计数的管脚和边沿。

当选择计数器模式作为工作模式时，在每个 PCLK 时钟的上升沿对 CAP 输入（由 CTCR 的位 3:2 来选择）进行采样。在对这个 CAP 输入的连续两次采样值进行比较之后，可以识别出下面其中一种事件：上升沿、下降沿、上升/下降沿或所选 CAP 输入的电平不变。如果识别出的事件与 CTCR 寄存器位 1:0 选择的一个事件相对应，那么定时器计数器寄存器的值将增加。

当计数器时钟由外部时钟提供时，处理的效率会受到一些限制。由于识别 CAP 所选输入的一个边沿需要使用 PCLK 时钟 2 个连续的上升沿，因此，CAP 输入的频率不能大于 PCLK 时钟的 1/4。所以，在这种情况下同一个 CAP 管脚的高/低电平时间不能短于 1/(2PCLK)。

表 23.5 计数控制寄存器（T[0/1/2/3]CTCR – 地址：0xE000 4070, 0xE000 8070,0xE007 0070, 0xE007 4070）位描述

位	符号	值	描述	复位值
1:0	计数器/定时器模式		这个字段选择哪个上升的 PCLK 边沿会增加定时器的预分频计数器（PC）、清除 PC 和增加定时器计数器（TC）。 定时器模式：当预分频计数器与预分频寄存器的值匹配时 TC 增加。	00
		00	定时器模式：每个上升的 PCLK 边沿。	
		01	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升沿出现时增加。	
		10	计数器模式：TC 在位 3:2 选择的 CAP 输入的下降沿出现时增加。	
		11	计数器模式：TC 在位 3:2 选择的 CAP 输入的两个边沿出现时增加。	
3:2	计数输入选择		当该寄存器的位 1:0 不为 00 时，这两位选择采样用于计时的 CAP 管脚：	00
		00	CAPn.0 用于 TIMERn	
		01	CAPn.1 用于 TIMERn	
		10	CAPn.2 用于 TIMERn	
		11	CAPn.3 用于 TIMERn **注意**：如果在 TnCTCR 中选择计数器模式用于某个特定的 CAPn 输入，则这个输入对应在捕获控制寄存器（TnCCR）中的 3 个位必须被编程为 000。但是，同一个定时器的其他 3 个 CAPn 输入可以被选择用作捕获和/或中断功能。	
7:4	-	-	保留，用户软件不应该向保留位写 1。从保留位读出的值未定义。	NA

23.5.4 匹配寄存器（MR0 – MR3）

匹配寄存器值连续与定时器计数值相比较。当两个值相等时自动触发相应动作。这些动作包括产生中断，复位定时器计数器或停止定时器。所执行的动作由 MCR 寄存器控制。

23.5.5 匹配控制寄存器（T[0/1/2/3]MCR - 0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014）

匹配控制寄存器用来控制在发生匹配时所执行的操作。每个位的功能见表 23.6。

表 23.6 匹配控制寄存器（T[0/1/2/3]MCR – 地址：0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014）位描述

位	符号	值	描述	复位值
0	MR0I	1	MR0 引发的中断：MR0 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	
1	MR0R	1	MR0 引发的复位：MR0 与 TC 值的匹配将使 TC 复位。	0
		0	该特性被禁止。	
2	MR0S	1	MR0 引发的停止：MR0 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	
3	MR1I	1	MR1 引发的中断：MR1 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	
4	MR1R	1	MR1 引发的复位：MR1 与 TC 值的匹配将使 TC 复位。	0
		0	该特性被禁止。	
5	MR1S	1	MR1 引发的停止：MR1 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	
6	MR2I	1	MR2 引发的中断：MR2 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	
7	MR2R	1	MR2 引发的复位：MR2 与 TC 值的匹配将使 TC 复位。	0
		0	该特性被禁止。	
8	MR2S	1	MR2 引发的停止：MR2 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	
9	MR3I	1	MR3 引发的中断：MR3 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	
10	MR3R	1	MR3 引发的复位：MR3 与 TC 值的匹配将使 TC 复位。	0
		0	该特性被禁止。	
11	MR3S	1	MR3 引发的停止：MR3 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	
15:12	-		保留，用户软件不应该向保留位写 1。从保留位读出的值未定义。	NA

23.5.6 捕获寄存器（CR0-CR3）

每个捕获寄存器都与一个器件管脚相关联。当管脚发生特定的事件时，可将定时器计数器的值装入该寄存器。捕获控制寄存器的设定决定捕获功能是否被使能以及捕获事件在管脚的上升沿、下降沿还是双边沿发生。

23.5.7 捕获控制寄存器（T[0/1/2/3]CCR – 0xE000 4028, 0xE000 8028, 0xE007 0028，0xE007 4028）

捕获控制寄存器用来控制在捕获事件发生时是否将定时器计数器的值装入其中一个捕获寄存器以及是否产生中断。同时设置上升沿和下降沿位也是有效的配置，这样会在两个边沿都触发捕获事件。在下面的描述中，“n”代表定时器的编号 0 或 1。

注意：如果在 CTCR 中选择计数器模式用于某个特定的 CAP 输入，则这个输入对应在这个寄存器中的 3 个位必须被编程为 000。但是，其他 3 个 CAP 输入可以被选择用作捕获和/或中断功能。

表 23.7 捕获控制寄存器（T[0/1/2/3]CCR – 地址 0xE000 4028, 0xE000 8020, 0xE007 0028, 0xE007 4028）位描述

位	符号	值	描述	复位值
0	CAP0RE	1	CAPn.0 上升沿时捕获：CAPn.0 上 0 到 1 的跳变将导致 TC 的内容装入 CR0。	0
		0	该特性被禁止。	
1	CAP0FE	1	CAPn.0 下降沿时捕获：CAPn.0 上 1 到 0 的跳变将导致 TC 的内容装入 CR0。	0
		0	该特性被禁止。	
2	CAP0I	1	CAPn.0 事件中断：CAPn.0 的捕获事件所导致的 CR0 装载将产生一个中断。	0
		0	该特性被禁止。	
3	CAP1RE	1	CAPn.1 上升沿时捕获：CAPn.1 上 0 到 1 的跳变将导致 TC 的内容装入 CR1。	0
		0	该特性被禁止。	
4	CAP1FE	1	CAPn.1 下降沿时捕获：CAPn.1 上 1 到 0 的跳变将导致 TC 的内容装入 CR1。	0
		0	该特性被禁止。	
5	CAP1I	1	CAPn.1 事件中断：CAPn.1 的捕获事件所导致的 CR1 装载将产生一个中断。	0
		0	该特性被禁止。	
6	CAP2RE	1	CAPn.2 上升沿时捕获：CAPn.2 上 0 到 1 的跳变将导致 TC 的内容装入 CR2。	0
		0	该特性被禁止。	
7	CAP2FE	1	CAPn.2 下降沿时捕获：CAPn.2 上 1 到 0 的跳变将导致 TC 的内容装入 CR2。	0
		0	该特性被禁止。	
8	CAP2I	1	CAPn.2 事件中断：CAPn.2 的捕获事件所导致的 CR2 装载将产生一个中断。	0
		0	该特性被禁止。	
9	CAP3RE	1	CAPn.3 上升沿时捕获：CAPn.3 上 0 到 1 的跳变将导致 TC 的内容装入 CR3。	0
		0	该特性被禁止。	

续表 23.7

位	符号	值	描述	复位值
10	CAP3FE	1	CAPn.3 下降沿时捕获：CAPn.3 上 1 到 0 的跳变将导致 TC 的内容装入 CR3。	0
		0	该特性被禁止。	
11	CAP3I	1	CAPn.3 事件中断：CAPn.3 的捕获事件所导致的 CR3 装载将产生一个中断。	0
		0	该特性被禁止。	
15:12	-		保留，用户软件不应该向保留位写 1。从保留位读出的值未定义。	NA

23.5.8 外部匹配寄存器（T[0/1/2/3]EMR – 0xE000 403C, 0xE000 803C, 0xE007 003C, 0xE007 403C）

外部匹配寄存器提供外部匹配管脚的控制和状态。在下面的描述中，“n”代表定时器编号，值为 0 或 1；“m”代表匹配编号，值为 0～3。

表 23.8 外部匹配寄存器（T[0/1/2/3]EMR – 地址：0xE000 403C, 0xE000 803C, 0xE007 003C, 0xE007 403C）位描述

位	功能	描述	复位值
0	EM0	外部匹配 0。当 TC 和 MR0 匹配时，该位可翻转，变为低电平，变为高电平或不执行任何动作，由本寄存器的 5:4 位来决定。该位的值会被驱动到 MATn.0 管脚上，采用正逻辑方式（0＝低电平，1＝高电平）。	0
1	EM1	外部匹配 1。当 TC 和 MR1 匹配时，该位可翻转，变为低电平，变为高电平或不执行任何动作，由本寄存器的 7:6 位来决定。该位的值会被驱动到 MATn.1 管脚上，采用正逻辑方式（0＝低电平，1＝高电平）。	0
2	EM2	外部匹配 2。当 TC 和 MR2 匹配时，该位可翻转，变为低电平，变为高电平或不执行任何动作，由本寄存器的 9:8 位来决定。该位的值会被驱动到 MATn.0 管脚上，采用正逻辑方式（0＝低电平，1＝高电平）。	0
3	EM3	外部匹配 3。当 TC 和 MR3 匹配时，该位可翻转，变为低电平，变为高电平或不执行任何动作，由本寄存器的 11:10 位来决定。该位的值会被驱动到 MATn.0 管脚上，采用正逻辑方式（0＝低电平，1＝高电平）。	0
5:4	EMC0	外部匹配控制 0。决定外部匹配 0 的功能。表 23.9所示为这两个位的编码。	00
7:6	EMC1	外部匹配控制 1。决定外部匹配 1 的功能。表 23.9所示为这两个位的编码。	00
9:8	EMC2	外部匹配控制 2。决定外部匹配 2 的功能。表 23.9所示为这两个位的编码。	00
11:10	EMC3	外部匹配控制 3。决定外部匹配 3 的功能。表 23.9所示为这两个位的编码。	00
15:12	-	保留，用户软件不应该向保留位写 1。从保留位读出的值未定义。	NA

表 23.9 外部匹配控制

EMR[11:10], EMR[9:8] EMR[7:6],或 EMR[5:4]	功能
00	不执行任何动作。
01	清零对应的外部匹配位/输出（如果连接到管脚，则 MATn.m 管脚为低电平）。
10	置位对应的外部匹配位/输出（如果连接到管脚，则 MATn.m 管脚为高电平）。
11	翻转对应的外部匹配位/输出。

23.6 定时器举例操作

图 23.1所示为定时器配置为在匹配时复位计数并产生中断。预分频器设置为 2，匹配寄存器设置为 6。在发生匹配的定时器周期结束时，定时器计数值复位。这样就使匹配值具有完整长度的周期。指示匹配发生的中断在定时器到达匹配值的下一个时钟产生。

图 23.2所示为定时器配置为在匹配时停止并产生中断。预分频器设置为 2，匹配寄存器设置为 6。在定时器到达匹配值后的下一个周期，TCR中的定时器使能位被清零并产生指示匹配发生的中断。

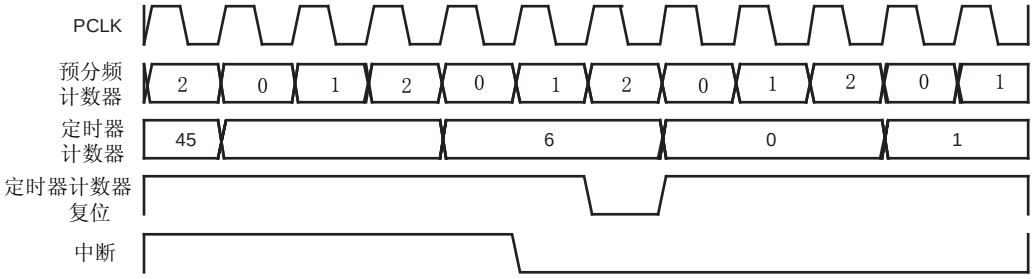


图 23.1 定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和复位

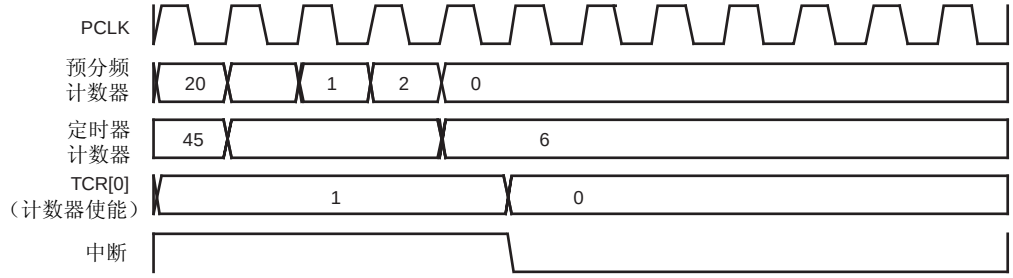


图 23.2 定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和停止

23.7 结构

定时器/计数器 0 和定时器/计数器 1 的方框图，见图 23.3。

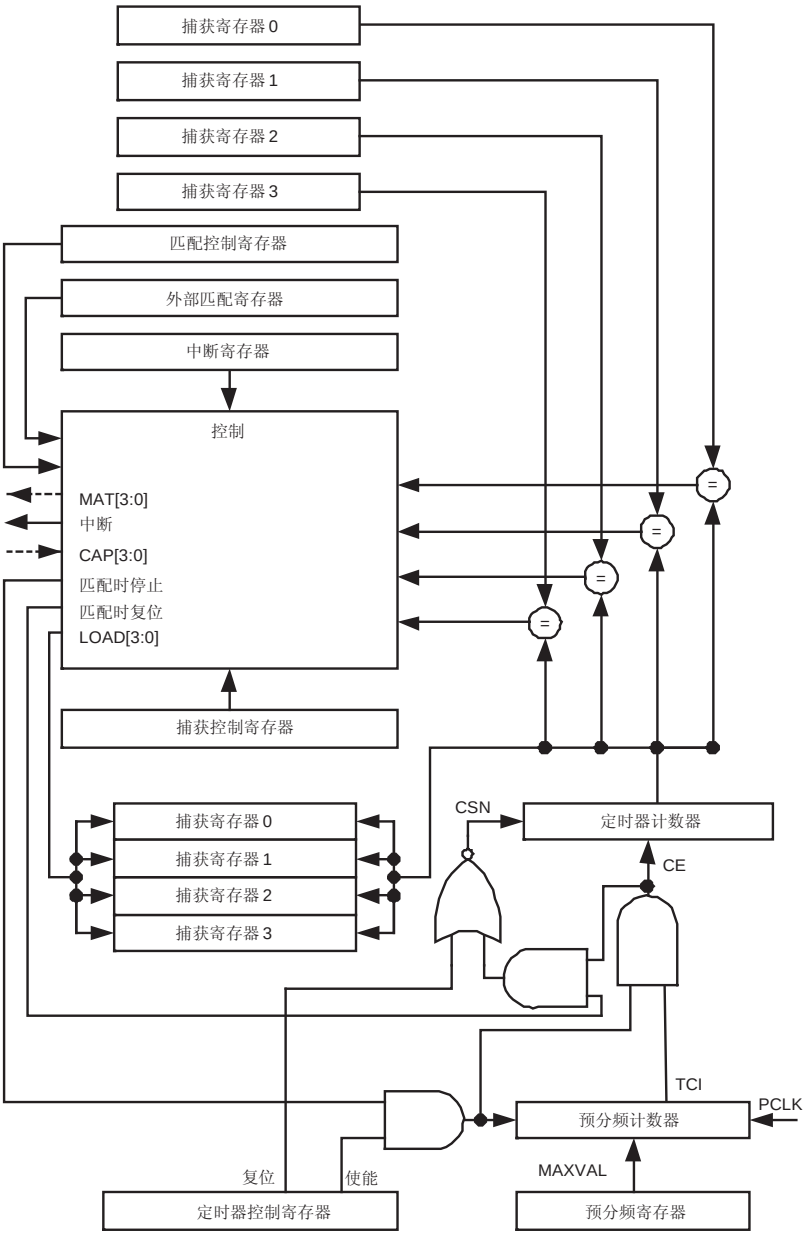

捕获寄存器 0
捕获寄存器 1
捕获寄存器 2
捕获寄存器 3
匹配控制寄存器
外部匹配寄存器
中断寄存器
控制
MAT[3:0]
中断
CAP[3:0]
匹配时停止
匹配时复位
LOAD[3:0]
捕获控制寄存器
捕获寄存器 0
捕获寄存器 1
捕获寄存器 2
捕获寄存器 3
CSN
定时器计数器
CE
TCI
PCLK
预分频计数器
复位
使能
MAXVAL
定时器控制寄存器
预分频寄存器

图 23.3 定时器框图

第24章 看门狗定时器（WDT）

24.1 特性

- 如果没有周期性重装，则产生片内复位；
- 调试模式；
- 由软件使能，但要求禁止硬件复位或看门狗复位/中断；
- 错误/不完整的喂狗时序会导致复位/中断（如果使能）；
- 指示看门狗复位的标志；
- 带内部预分频器的可编程 32 位定时器；
- 可选择 $T_{WDCLK}\times4$ 倍数的时间周期：从（$T_{WDCLK}\times256\times4$）到（$T_{WDCLK}\times2^{32}\times4$）；
- 看门狗时钟源（WDCLK）可以选择 RTC 时钟、内部 RC 振荡器（IRC）或 APB 外设时钟（PCLK）。这给不同功耗条件下的看门狗操作提供了宽范围的定时选择。它还可以使看门狗定时器在一个与外部晶体及其相关元件无关的内部时钟源下运行，这样使可靠性得到增强。

24.2 应用

看门狗的用途是使微控制器在进入错误状态后的一定时间内复位。当看门狗使能时，如果用户程序没有在预定的时间内喂狗（重装），看门狗会产生一个系统复位。

有关片内看门狗和外围功能的相互作用，尤其是复位和引导过程的关系，请参考本文档的3.4 节“复位”。

24.3 描述

看门狗包括一个 4 分频的预分频器和一个 32 位计数器。时钟通过预分频器输入定时器。定时器递减计数。定时器递减的最小值为 0xFF。如果设置一个小于 0xFF 的值，系统会将 0xFF 装入计数器。因此最小看门狗间隔为($T_{WDCLK}\times256\times4$)，最大间隔为($T_{WDCLK}\times2^{32}\times4$)，两者都是 $T_{WDCLK}\times4$ 的倍数。看门狗应当根据下面的方法来使用：

- 在 WDTC 寄存器中设置看门狗定时器的固定装载值
- 在 WDMOD 寄存器中设置模式
- 通过向 WDFEED 寄存器顺序写入 0xAA 和 0x55 使能看门狗
- 在看门狗向下溢出之前应当再次喂狗以防止复位/中断

当看门狗计数器向下溢出时，程序计数器将从 0x0000 0000 开始，和外部复位一样。可以检查看门狗超时标志（WDTOF）来确定看门狗是否产生了复位条件。WDTOF 标志必须由软件清零。

看门狗定时器模块使用 2 个时钟：PCLK 和 WDCLK。PCLK 供 APB 访问看门狗寄存器使用。WDCLK 供看门狗定时器计数使用。

在这两个时钟区（clock domain）有一些同步逻辑。当 WDMOD 和 WDTC 被 APB 操作

更新时，新的值将在 WDCLK 时钟区逻辑的 3 个 WDCLK 周期内生效。当看门狗定时器正在计数 WDCLK 时，同步逻辑将先锁定 WDCLK 的计数器值，然后再使其与 PCLK 同步，由 CPU 将计数器作为 WDTV 寄存器读出。

24.4 寄存器描述

看门狗包含 4 个寄存器，如下面的表 24.1所示。

表 24.1 看门狗寄存器映射

名称	描述	访问	复位值[1]	地址
WDMOD	看门狗模式寄存器。该寄存器包含看门狗定时器的基本模式和状态。	R/W	0	0xE000 0000
WDTC	看门狗定时器常数寄存器。该寄存器决定超时值。	R/W	0xFF	0xE000 0004
WDFEED	看门狗喂狗序列寄存器。向该寄存器顺序写入 0xAA 和 0x55 使看门狗定时器重新装入 WDTC 的值。	WO	NA	0xE000 0008
WDTV	看门狗定时器值寄存器。该寄存器读出看门狗定时器的当前值。	RO	0xFF	0xE000 000C
WDCLKSEL	看门狗时钟源选择寄存器。	R/W	0	0xE000 0010

[1] 复位值只反映了使用位的值，不包含保留位的内容。

24.4.1 看门狗模式寄存器（WDMOD – 0xE000 0000）

WDMOD 寄存器通过 WDEN 位和 RESET 位的组合来控制看门狗的操作。

表 24.2 看门狗工作模式选择

WDEN	WDRESET	工作模式
0	X（0 或 1)	看门狗关闭时的调试/操作。
1	0	看门狗中断模式：看门狗中断使能，但 WDRESET 不使能时的调试。 当这种模式被选择时，看门狗计数器向下溢出会置位 WDINT 标志，也会产生看门狗中断请求。
1	1	看门狗复位模式：看门狗中断和 WDRESET 都使能时的操作。 当这种模式被选择时，看门狗计数器向下溢出会复位微控制器。尽管在这种情况下看门狗中断也使能（WDEN＝1），但由于看门狗复位会清零 WDINT 标志，所以无法判断出看门狗中断。

一旦 **WDEN** 和/或 **WDRESET** 位被置位，它们就无法使用软件来清零。这两个标志由外部复位或看门狗定时器溢出来清除。

WDTOF 当看门狗发生超时，看门狗超时标志置位。该标志由软件清零。

WDINT 当看门狗发生超时，看门狗中断标志置位。产生的任何复位都会使这个标志被清零。一旦看门狗中断服务被响应，看门狗中断可能会在 VIC 中被禁能，或者，可能不会产生看门狗中断请求。

表 24.3 看门狗模式寄存器（WDMOD – 地址 0xE000 0000）位描述

位	符号	描述	复位值
0	WDEN	看门狗中断使能位（只能置位）。	0
1	WDRESET	看门狗复位使能位（只能置位）。	0
2	WDTOF	看门狗超时标志。	0（仅在外部复位后）
3	WDINT	看门狗中断标志（只读）。	0
7:4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

24.4.2 看门狗定时器常数寄存器（WDTC – 0xE000 0004）

WDTC 寄存器决定看门狗超时值。当喂狗时序产生时，WDTC 的内容重新装入看门狗定时器。它是一个 32 位寄存器，低 8 位在复位时设置为 1。写入一个小于 0xFF 的值会使 0x0000 00FF 装入 WDTC，因此超时的最小时间间隔为 T_{WDCLK}×256×4。

表 24.4 看门狗常数寄存器（WDTC – 地址 0xE000 0004）位描述

位	功能	描述	复位值
31:0	计数值	看门狗超时间隔。	0x0000 00FF

24.4.3 看门狗喂狗寄存器（WDFEED - 0xE000 0008）

向该寄存器写入 0xAA，然后写入 0x55 会使 WDTC 的值重新装入看门狗定时器。如果看门狗通过 WDMOD 寄存器使能，该操作还将启动看门狗运行。置位 WDMOD 中的 WDEN 位不足以使能看门狗。在置位 WDEN 之后必须完成一次有效的喂狗时序，然后看门狗才能够产生复位。到那时，看门狗将忽略喂狗错误。在看门狗使能的前提下，如果向 WDFEED 寄存器写入 0xAA 之后的下一个操作不是向 WDFEED 寄存器写入 0x55，而是访问 APB 寄存器，那么会立刻造成复位/中断。在喂狗时序中，一次对看门狗寄存器不正确访问之后的第二个 PCLK 周期将产生复位。

上面的描述意味着在喂狗过程中应该禁能中断。

表 24.5 看门狗喂狗寄存器（WDFEED – 地址 0xE000 0008）位描述

位	功能	描述	复位值
7:0	喂狗	喂狗值应当为 0xAA，然后是 0x55。	NA

24.4.4 看门狗定时器值寄存器（WDTV – 0xE000 000C）

WDTV 寄存器用来读取看门狗定时器的当前值。

当读取 32 位定时器的值时，锁定和同步过程要占用 6 个 WDCLK 周期和 6 个 PCLK 周期，因此，WDTV 的值比 CPU 正在读取的定时器的实际值要旧。

表 24.6 看门狗定时器值寄存器（WDTV – 地址 0xE000 000C）位描述

位	符号	描述	复位值
31:0	计数	计数器定时器值。	0x0000 00FF

24.4.5 看门狗定时器时钟源选择寄存器（WDCLKSEL – 0xE000 0010）

这个寄存器允许选择看门狗定时器的时钟源。看门狗的时钟源可以是：内部RC振荡器（IRC）、RTC振荡器和APB外设时钟（pclk）。WDCLKSEL的位功能如表 24.7所示。

表 24.7 看门狗定时器时钟源选择寄存器（WDCLKSEL – 地址 0xE000 0010）位描述

位	符号	值	描述	复位值
1:0	WDSEL		这两位按照下面的描述来选择看门狗定时器的时钟源。 **警告：**这两位值的错误设置会导致看门狗定时器操作错误，会对系统操作造成不利的影响。	0
		00	选择内部 RC 振荡器作为看门狗时钟源（默认）。	
		01	选择 APB 外设时钟（PCLK）作为看门狗时钟源。	
		10	选择 RTC 振荡器作为看门狗时钟源。	
		11	保留。	
31:2	-	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

24.5 方框图

看门狗的方框图如图 24.1所示。方框图中并未显示同步逻辑（PCLK – WDCLK）。

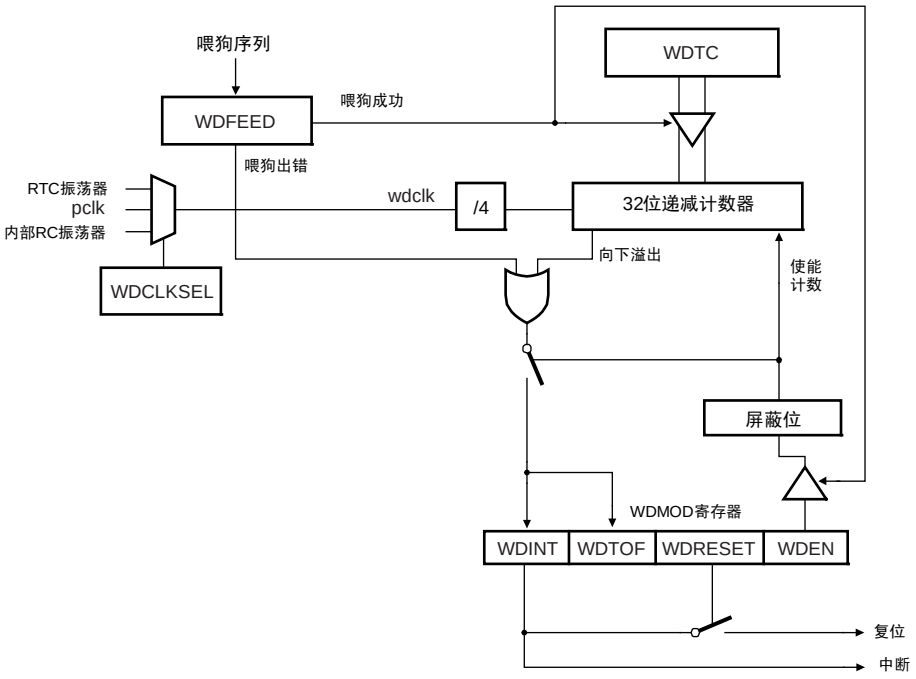


图 24.1 看门狗方框图

第25章 脉宽调制器PWM0 和PWM1

25.1 特性

- 具有两个相同操作特性的 PWM。通过将 PWM 设置在相同速率下运行，然后同时使能，就有可能使 PWM 在同步方式下操作。PWM0 用作主机，PWM1 用作从机；
- 计数器或定时器操作（可以使用外设时钟或其中一个捕获输入作为时钟源）；
- 7 个匹配寄存器，可实现 6 个单边沿控制或 3 个双边沿控制的 PWM 输出，或两种类型的混合输出。匹配寄存器还允许：

  －连续操作，可在匹配时选择产生中断

  －匹配时停止定时器，可选择产生中断

  －匹配时复位定时器，可选择产生中断
- 支持单边沿控制和/或双边沿控制的 PWM 输出。单边沿控制的 PWM 输出在每个周期开始时全部置高，除非输出保持恒定低电平。双边沿控制的 PWM 输出可在一个周期内的任何位置上产生任何边沿，因此该输出可同时产生正和负脉冲；
- 脉冲周期和脉冲宽度可以是定时器计数的任何值，这就允许在分辨率和重复率之间灵活地权衡。所有 PWM 输出都以相同的重复率出现；
- 双边沿控制的 PWM 输出可编程为正脉冲或负脉冲；
- 匹配寄存器的更新与脉冲的输出同步，以防止错误脉冲的产生。软件必须在新的匹配值生效之前将它们释放；
- 可在 PWM 模式没有使能时作为标准定时器使用；
- 带可编程 32 位预分频器的 32 位定时器/计数器；
- 当输入信号跳变时，3 个 32 位的捕获通道可取得定时器的瞬时值，捕获事件可选择产生中断。

25.2 描述

PWM 基于标准的定时器模块并继承了该模块的所有特性，不过 LPC2400 只将其 PWM 功能输出到管脚。定时器对外设时钟((PCLK)进行计数，可选择产生中断或在出现指定的定时器值（参考 7 个匹配寄存器）时执行其他操作。它还包括 4 个捕获输入，以便在输入信号跳变时保存定时器的值，并在出现这些事件时产生中断。PWM 功能是一个附加特性，建立在匹配寄存器事件的基础之上。

独立控制上升和下降沿位置的能力使 PWM 可以应用于更多的领域。例如，多相位电机控制通常需要 3 个非重叠的 PWM 输出，而且要求对 3 个输出的脉宽和位置进行独立的控制。

两个匹配寄存器可用于提供单边沿控制的 PWM 输出。一个匹配寄存器（MR0）通过匹配时重新设置计数值来控制 PWM 的速率。另一个匹配寄存器控制 PWM 边沿的位置。其它单边沿控制的 PWM 输出只需要一个匹配寄存器，因为所有 PWM 输出的重复率是相同的。当 MR0 匹配出现时，多个单边沿控制的 PWM 输出在每个 PWM 周期开始时有一个上升沿。

3 个匹配寄存器可用于提供一个双边沿控制的 PWM 输出。也就是说，MR0 匹配寄存器

控制着 PWM 的速率，另一个匹配寄存器控制两个 PWM 边沿的位置。其它双边沿控制的 PWM 输出都只需要两个匹配寄存器，因为所有 PWM 输出的重复率是相同的。

使用双边沿控制的 PWM 输出时，指定的匹配寄存器控制输出的上升和下降沿。这样就产生了正脉冲（当上升沿先于下降沿时）和负脉冲（当下降沿先于上升沿时）。

图 25.1为PWM框图。框图的右边和顶端是已添加到标准定时器模块的部分。在框图的左下方可以找到定时器控制寄存器的主机使能输出，如果需要，它允许主机PWM（PWM0）同时使能自身以及从机PWM（PWM1）。PWM0 的主机使能输出同时与主从PWM块的外部使能输入相连。

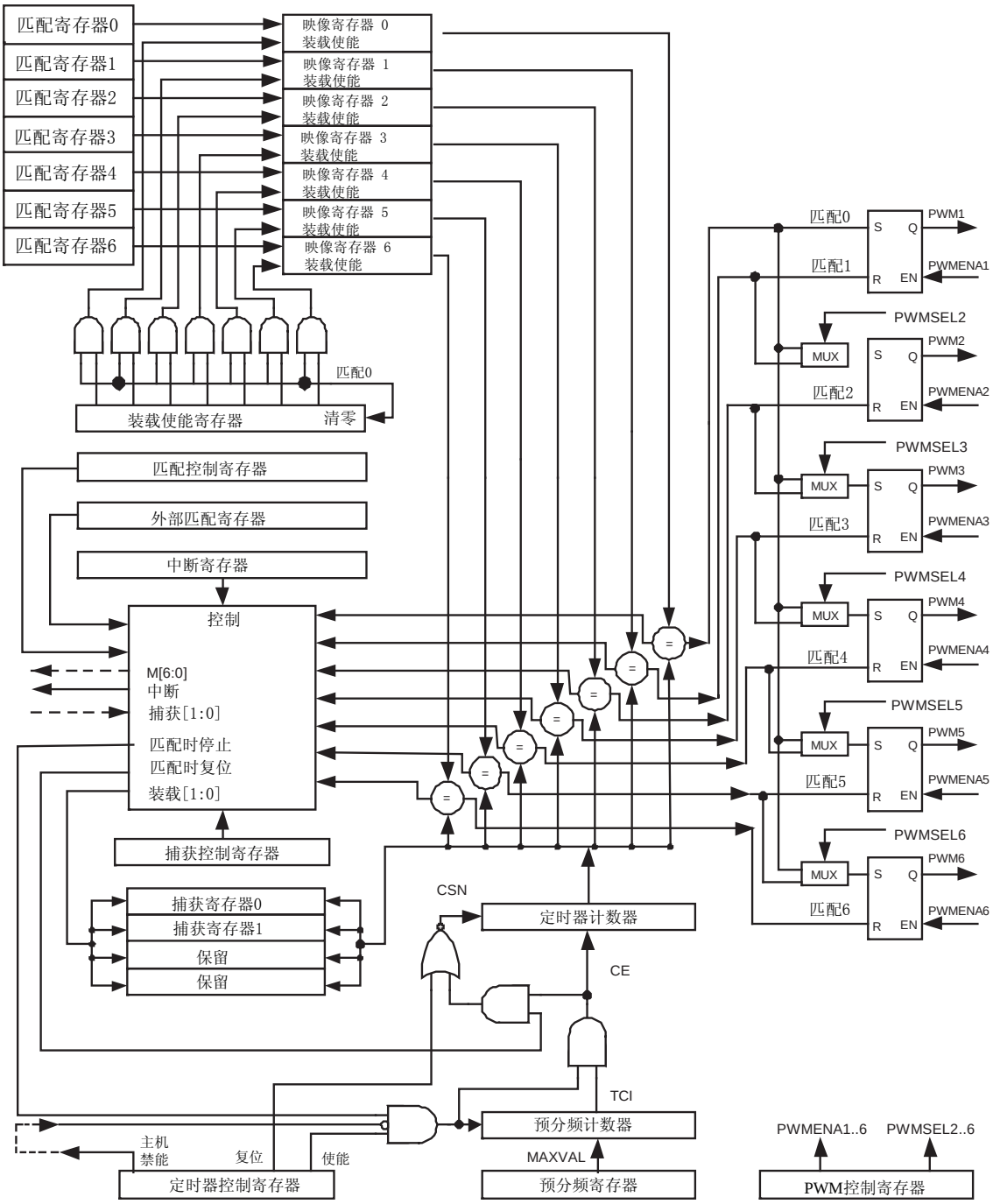


图 25.1 PWM 框图

25.2.1 单边沿控制的PWM输出规则

1. 所有单边沿控制的 PWM 输出在 PWM 周期开始时都为高电平，除非它们的匹配值等于 0。
2. 每个 PWM 输出都在到达其匹配值时变为低电平。如果没有发生匹配（例如匹配值大于 PWM 速率），则 PWM 将一直保持高电平。

25.2.2 双边沿控制的PWM输出规则

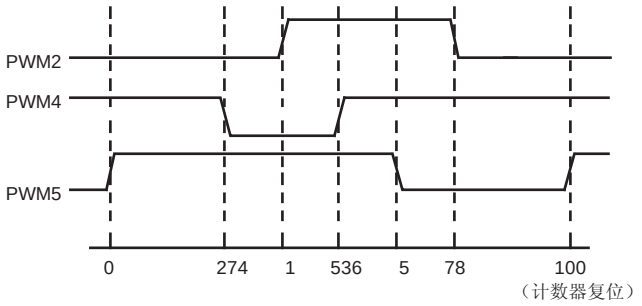
当一个新的周期将要开始时，使用以下 5 个规则来决定下一个 PMW 输出的值：

1. 在 PWM 周期结束时（与下一个 PWM 周期的开始重合的时间点）使用下一个 PMW 周期的匹配值，例外的情况见规则 3。
2. 匹配值等于 0 或等于当前 PWM 速率（与匹配通道 0 的值相同）具有相同的效果，例外的情况见规则 3。例如，在 PWM 周期开始时的下降沿请求与 PWM 周期结束时的下降沿请求等效。
3. 当匹配值正在改变时，如果有其中一个“旧”的匹配值等于 PWM 速率，只要新的匹配值不等于 0 也不等于 PWM 速率，并且旧的匹配值不等于 0，则旧的匹配值将再次被使用。
4. 如果同时请求 PWM 输出置位和清零，清零优先。这种状况可能在置位和清零匹配值相同时，或者置位或清零值等于 0 并且其它值等于 PWM 速率时发生。
5. 如果匹配值超出范围（如大于 PMW 速率值），将不会发生匹配事件，匹配通道对输出不起作用。也就是说 PWM 输出将一直保持一种状态，可以为低电平、高电平或是“无变化”输出。

25.2.3 标准定时器模块不同点的总结

1. 添加同步的寄存器（映像寄存器）到每个匹配寄存器以允许仅在软件请求且在 PWM 周期之间跳变时发生改变。
2. 添加新的装载使能寄存器（LER）以允许软件控制匹配寄存器更新。LER 含有一个位用于每个匹配寄存器。当 LER 中的位被写入 1 时，相应的匹配通道的映像寄存器内容在计数器复位（匹配 0 出现时）被载入真正的匹配寄存器中。LER 位在计数器复位时自动复位。
3. 在 TCR 寄存器中添加一个 PWM 模式位。PWM 模式在上述的软件/硬件控制下使能从映像寄存器中载入真正的匹配寄存器。当 PWM 模式不被使能时，匹配值映像寄存器被透明（transparent）或忽略。
4. 在 TCR 寄存器中添加主机使能位，它的值从 PWM 模块中得出。一个外部使能输入被添加到 PWM 模块，PWM 模块连接主机 PWM 模块之外的主机使能输出。
5. 匹配寄存器的最大数目增加到 7，因此允许支持多达 3 个双边沿 PWM 通道。这包括所需的匹配输出、控制位等，以用于每个匹配寄存器：
   －增加 3 个新的匹配寄存器，创建匹配通道 4 到 6。
   －增加 3 组额外的停止位（S）、复位（R）和中断位（I）到 MCR 寄存器（每个额外的匹配寄存器 3 个位）。
6. 增加 PWM 输出到定时器，该定时器将等效的 RS 触发器连接到 2 个匹配输出。每个 PWM 输出上的 2-to-1 多路复用器允许选择单边沿或双边沿的 PWM。增加一个新的寄存器（PCR）来保存多路复用器的控制位（PWMSEL 位）。
7. 增加 3 个中断位到 IR 寄存器。

图 25.2展示了PWM值与波形输出之间的关系。图 25.1中的PWM输出逻辑允许通过PWMSELn位控制的多路复用器选择单边沿或者双边沿控制的PWM输出。表 25.1给出不同PWM输出的匹配寄存器选项。这个实现方案支持N-1 个单边沿PWM输出或(N-1)/2 个双边沿PWM输出，其中N是实现的匹配寄存器个数。如果需要，也可使用混合的PWM类型。如果LPC2468 器件N=7，则它同时给出多达 6 个可用的单边沿PWM输出或多达 3 个可用的双边沿PWM输出。



下面所示的波形显示了单个 PWM 周期，并显示了在下列条件下的 PWM 输出：

- 定时器配置为 PWM 模式。
- 匹配寄存器 0 配置为在发生匹配事件时复位定时器/计数器。
- 控制位 PWMSEL2 和 PWMSEL4 置位。

匹配寄存器的值如下：

MR0 = 100 (PWM 速率)

MR1 = 41, MR2 = 78 (PWM2 输出)

MR3 = 53, MR4 = 27 (PWM4 输出)

MR5 = 65 (PWM5 输出)

图 25.2 PWM 的示例波形

表 25.1 PWM 触发器的置位和复位输入

PWM 通道	单边沿 PWM(PWMSELn=0)		双边沿 PWM(PWMSELn=1)	
	置位	复位	置位	复位
1	匹配0	匹配1	匹配0[1]	匹配1[1]
2	匹配0	匹配2	匹配1	匹配2
3	匹配0	匹配3	匹配2[2]	匹配3[2]
4	匹配0	匹配4	匹配3	匹配4
5	匹配0	匹配5	匹配4[2]	匹配5[2]
6	匹配0	匹配6	匹配5	匹配6

[1]. 这种情况下与单边沿模式相同，因为匹配 0 是相邻的匹配寄存器。基本上 PWM1 不能用作双边沿输出。

[2]. 通常不建议使用 PWM 通道 3 和通道 5 作为双边沿 PWM 输出，因为这样会减少可用的双边沿 PWM 的个数。使用 PWM2, PWM4 和 PWM6 可得到最多对数的双边沿 PWM 输出。

25.3 管脚描述

表 25.2总结了所有与PWM相关的管脚。

表 25.2 管脚小结

管脚	类型	描述
PWM1	输出	PWM 通道 1 输出。
PWM2	输出	PWM 通道 2 输出。
PWM3	输出	PWM 通道 3 输出。
PWM4	输出	PWM 通道 4 输出。
PWM5	输出	PWM 通道 5 输出。
PWM6	输出	PWM 通道 6 输出。
PCAP0.0 PCAP1[1:0]	输入	捕获输入。如果在捕获管脚上发生跳变事件，那么定时器/计数器的值将载入相应的捕获寄存器，并选择是否产生中断。PWM0 产生一个捕获输入，PWM1 产生两个捕获输入。

25.4 PWM基址

表 25.3 PWM0 和 PWM1 的地址

PWM	基址
0	0xE001 4000
1	0xE001 8000

25.5 寄存器描述

PWM0 和PWM1 功能增加了新的寄存器和寄存器位，见表 25.4。

表 25.4 PWM0 和 PWM1 寄存器映射

名称	描述	访问	复位值[1]	PWM0 地址 &名称	PWM1 地址 &名称
IR	中断寄存器。可以写 IR 来清除中断。可读取 IR 来识别 8 个中断源中哪个被挂起。	R/W	0	0xE001 4000 PWM0IR	0xE001 8000 PWM1IR
TCR	定时器控制寄存器。TCR 用于控制定时器计数器功能。定时器计数器可通过 TCR 禁止或复位。	R/W	0	0xE001 4004 PWM0TCR	0xE001 8004 PWM1TCR
TC	定时器计数器。32 位 TC 每经过 PR+1 个 PCLK 周期加 1。TC 通过 TCR 进行控制。	R/W	0	0xE001 4008 PWM0TC	0xE001 8008 PWM1TC
PR	预分频寄存器。TC 每经过 PR+1 个 PCLK 周期加 1。	R/W	0	0xE001 400C PWM0PR	0xE001 800C PWM1PR

续表 25.4

名称	描述	访问	复位值[1]	PWM0 地址& 名称	PWM1 地址& 名称
PC	预分频计数器。32 位的 PC 是一个计数器，它增加到等于 PR 中保存的值。当到达了 PR 中的值时，TC 加 1。PC 可通过总线接口进行观察和控制。	R/W	0	0xE001 4010 PWM0PC	0xE001 8010 PWM1PC
MCR	匹配控制寄存器。MCR 用于控制在匹配时是否产生中断或复位 TC。	R/W	0	0xE001 4014 PWM0MCR	0xE001 8014 PWM1MCR
MR0	匹配寄存器 0。MR0 可通过 MCR 使能为在它与 TC 匹配时复位 TC、停止 TC 和 PC 和/或产生中断。此外，MR0 和 TC 的匹配将置位所有单边沿模式的 PWM 输出，并置位双边沿模式下的 PWM1。	R/W	0	0xE001 4018 PWM0MR0	0xE001 8018 PWM1MR0
MR1	匹配寄存器 1。MR1 可通过 MCR 使能为在它与 TC 匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR1 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM1，并置位双边沿模式下的 PWM2。	R/W	0	0xE001 401C PWM0MR1	0xE001 801C PWM1MR1
MR2	匹配寄存器 2。MR2 可通过 MCR 使能为在它与 TC 匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR2 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM2，并置位双边沿模式下的 PWM3。	R/W	0	0xE001 4020 PWM0MR2	0xE001 8020 PWM1MR2
MR3	匹配寄存器 3。MR3 可通过 MCR 使能为在它与 TC 匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR3 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM3，并置位双边沿模式下的 PWM4。	R/W	0	0xE001 4024 PWM0MR3	0xE001 8024 PWM1MR3
CCR	捕获控制寄存器。CCR 控制使用捕获输入的哪一个边沿来加载捕获寄存器和是否在捕获发生时产生中断。	R/W	0	0xE001 4028 PWM0CCR	0xE001 8028 PWM1CCR
CR0	捕获寄存器 0。在 CAPn.0 输入上有事件发生时使用 TC 的值来加载 PWMn CR0。	RO	0	0xE001 402C PWM0CR0	-
CR1	捕获寄存器 1。见 CR0。	RO	0	0xE001 4030 PWM0CR1	0xE001 8030 PWM1CR1
MR4	匹配寄存器 4。MR4 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR4 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM4，并置位双边沿模式下的 PWM5。	R/W	0	0xE001 4040 PWM0MR	0xE001 8040 PWM1MR

续表 25.4

名称	描述	访问	复位值[1]	PWM0 地址&名称	PWM1 地址&名称
MR5	匹配寄存器 5。MR5 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR5 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM5，并置位双边沿模式下的 PWM6。	R/W	0	0xE001 4044 PWM0MR	0xE001 8044 PWM1MR
MR6	匹配寄存器 6。MR6 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR6 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM6。	R/W	0	0xE001 4048 PWM0MR	0xE001 8048 PWM1MR
PCR	PWM 控制寄存器。它使能 PWM 输出并将 PWM 通道类型选择为单边沿或双边沿控制。	R/W	0	0xE001 404C PWM0PCR	0xE001 804C PWM1PCR
LER	加载使能寄存器。使能新 PWM 匹配值的使用。	R/W	0	0xE001 4050 PWM0LER	0xE001 8050 PWM1LER
CTCR	计数控制寄存器。CTCR 在定时器和计数器中选择一种模式，在计数器模式中选择要计数的信号或边沿。	R/W	0	0xE001 4070 PWM0CTCR	0xE001 8070 PWM1CTCR

[1]复位值只反映了储存在已使用的位中的数据。它并不包含保留位的内容。

25.5.1 PWM中断寄存器（PWM0IR - 0xE001 4000 和PWM1IR - 0xE001 8000）

PWM中断寄存器包含 11 个位（见表 25.5）。其中 7 个位用于匹配中断，4 个位保留将来之用。如果有中断产生，则PWMIR中对应的位置位，否则该位为 0。向对应的IR位写入 1 会复位中断。写入 0 无效。

表 25.5 PWM 中断寄存器（PWM0IR –地址 0xE001 4000 和 PWM1IR –地址 0xE001 8000）位描述

位	符号	描述	复位值
0	PWMMR0 中断	PWM 匹配通道 0 的中断标志。	0
1	PWMMR1 中断	PWM 匹配通道 1 的中断标志。	0
2	PWMMR2 中断	PWM 匹配通道 2 的中断标志。	0
3	PWMMR3 中断	PWM 匹配通道 3 的中断标志。	0
7:4	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未定义。	0000
8	PWMMR4 中断	PWM 匹配通道 4 的中断标志。	0
9	PWMMR5 中断	PWM 匹配通道 5 的中断标志。	0
10	PWMMR6 中断	PWM 匹配通道 6 的中断标志。	0
15:11	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未定义。	NA

25.5.2 PWM定时器控制寄存器（PWM0TCR -0xE001 4004 和PWM1TCR - 0xE001 8004）

PWM定时器控制寄存器（PWMTCR）用来控制PWM定时器计数器的操作。每个位的功能见表 25.6。

表 25.6 PWM 定时器控制寄存器（PWM0TCR –地址 0xE001 4004 和 PWM1TCR –地址 0xE001 8004）位描述

位	符号	描述	复位值
0	计数器使能	为 1 时，PWM 定时器计数器和 PWM 预分频计数器使能计数。为 0 时，计数器被禁止。	0
1	计数器复位	为 1 时，PWM 定时器计数器和 PWM 预分频计数器在 PCLK 的下一个上升沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
2	-	保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA
3	PWM 使能	为 1 时，PWM 模式使能。PWM 模式将映像寄存器连接到匹配寄存器。只有在 PWMLER 中的相应位置位后发生的匹配 0 事件才会使程序写入匹配寄存器的值生效。需要注意的是，决定 PWM 速率（PWM 匹配寄存器 0-MR0）的 PWM 匹配寄存器必须在使能 PWM 之前设定。否则不会出现匹配事件来使映像寄存器的内容生效。	0
4	主机禁能（仅为 PWM0）	使用主机禁能控制位可使两个PWM同步。主机PWM（PWM0 模块）的主机禁能位控制两个PWM的第二个使能输入，如图 25.1所示。如果分别使用PWM，则主机禁能位应总是为 0，并且使用各个计数器使能位来控制它们。 该位在从机 PWM（PWM1）中不执行任何操作。	0
7:5	-	保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

25.5.3 PWM计数控制寄存器（PWM0CTCR - 0xE001 4070 和PWM1CTCR - 0xE001 8070）

计数控制寄存器（CTCR）用来在定时器模式和计数器模式之间进行选择，并在计数器模式中选择进行计数的管脚和边沿。各个位的功能见表 25.7。

表 25.7 PWM 计数控制寄存器（PWM0CTCR –地址 0xE001 4004 和 PWM1CTCR –地址 0xE001 8004）位描述

位	符号	描述	复位值
1:0	计数器/定时器模式	00：定时器模式：当预分频计数器和预分频寄存器匹配时 TC 加 1	00
		01：计数器模式：通过设置位 3:2 使 TC 在 PCAP 输入信号的上升沿加 1。	
		10：计数器模式：通过设置位 3:2 使 TC 在 PCAP 输入信号的下降沿加 1。	
		11：计数器模式：通过设置位 3:2 使 TC 在 PCAP 输入信号的上升沿和下降沿都加 1。	
3:2	计数输入选择	当寄存器的 1:0 位不为 00 时，这些位选择哪个携带信号的 PCAP 管脚用来使 TC 加 1。 PWM0:00 = PCAP0.0（其它组合保留）。 PWM1:00 = PCAP1.0，01= PCAP1.1（其它组合保留）。	00
7:4	-	保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

[1] PCAP 输入信号频率必须不能超过 PCLK/4。当通过 PCAP 管脚提供 PWM 时钟时，该管脚上信号的高(低）电平持续的时间决不能少于 1/(2×PCLK)。

25.5.4 PWM 匹配控制寄存器（PWM0MCR -0xE001 4014 和 PWM1MCR-0xE001 8014）

PWM匹配控制寄存器用来控制在PWM匹配寄存器与PWM定时器计数器匹配时所执行的操作。每个位的功能见表 25.8。

表 25.8 匹配控制寄存器（PWM0MCR –地址 0xE001 4014 和 PWM1MCR – 地址 0xE000 8014）位描述

位	符号	值	描述	复位值
0	PWMMR0I	1	PWMMR0 引发的中断：PWMMR0 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
1	PWMMR0R	1	PWMMR0 引发的复位：PWMMR0 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
2	PWMMR0S	1	PWMMR0 引发的停止：如果 PWMMR0 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
3	PWMMR1I	1	PWMMR1 引发的中断：PWMMR1 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
4	PWMMR1R	1	PWMMR1 引发的复位：PWMMR1 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
5	PWMMR1S	1	PWMMR1 引发的停止：如果 PWMMR1 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
6	PWMMR2I	1	PWMMR2 引发的中断：PWMMR2 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
7	PWMMR2R	1	PWMMR2 引发的复位：PWMMR2 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
8	PWMMR2S	1	PWMMR2 引发的停止：如果 PWMMR2 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
9	PWMMR3I	1	PWMMR3 引发的中断：PWMMR3 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
10	PWMMR3R	1	PWMMR3 引发的复位：PWMMR3 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
11	PWMMR3S	1	PWMMR3 引发的停止：如果 PWMMR3 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	

续表 25.8

位	符号	值	描述	复位值
12	PWMMR4I	1	PWMMR4 引发的中断：PWMMR4 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
13	PWMMR4R	1	PWMMR4 引发的复位：PWMMR4 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
14	PWMMR4S	1	PWMMR4 引发的停止：如果 PWMMR4 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
15	PWMMR5I	1	PWMMR5 引发的中断：PWMMR5 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
16	PWMMR5R	1	PWMMR5 引发的复位：PWMMR5 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
17	PWMMR5S	1	PWMMR5 引发的停止：如果 PWMMR5 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
18	PWMMR6I	1	PWMMR6 引发的中断：PWMMR6 与 PWMTC 的值相匹配时产生中断。	0
		0	该中断被禁止。	
19	PWMMR6R	1	PWMMR6 引发的复位：PWMMR6 与 PWMTC 的值相匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
20	PWMMR6S	1	PWMMR6 引发的停止：如果 PWMMR6 与 PWMTC 的值相匹配，则 PWMTC 和 PWMPC 停止，PWMTCR[0]被设为 0。	0
		0	该特性被禁止。	
31:21	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

25.5.5 PWM捕获控制寄存器（PWM0CCR - 0xE001 4028 和PWM1CCR - 0xE001 8028）

捕获控制寄存器用来控制在出现一个捕获事件时是否向 4 个捕获寄存器中的一个加载定时器计数器的值，以及决定是否通过捕获事件产生中断。上升沿和下降沿可以同时设置，这将导致在两个边沿上都会发生捕获事件。在以下描述中，“n”代表定时器编号 0 或 1。

注：如果在 CTCR 中选择一个特定的 PCAP 输入用于计数器模式，则该输入在本寄存器中的 3 个位都应该被设置为 000，而其他 2 个 PCAP 输入可以选择进行捕获和/或产生中断。

表 25.9 PWM 捕获控制寄存器（PWM0CCR – 地址 0xE001 4028 和 PWM1CCR – 地址 0xE001 8028）位描述

位	符号	值	描述	复位值
0	在 PCAPn.0 的上升沿进行捕获	0	该特性被禁止。	0
		1	同步采样到的 PCAPn.0 上升沿将使 TC 的值载入 CR0。	
1	在 PCAPn.0 的下降沿进行捕获	0	该特性被禁止。	0
		1	同步采样到的 PCAPn.0 下降沿将使 TC 的值载入 CR0。	
2	发生 PCAPn.0 事件时产生中断	0	该特性被禁止。	0
		1	由 PCAPn.0 事件引发的 CR0 装载将产生中断。	
3	在 PCAPn.1 的上升沿进行捕获[1]	0	该特性被禁止。	0
		1	同步采样到的 PCAPn.1 上升沿将使 TC 的值载入 CR1。	
4	在 PCAPn.1 的下降沿进行捕获	0	该特性被禁止。	0
		1	同步采样到的 PCAPn.1 下降沿将使 TC 的值载入 CR1。	
5	发生 PCAPn.1 事件时产生中断[1]	0	该特性被禁止。	0
		1	由 PCAPn.1 事件引发的 CR1 装载将产生中断。	
31:6	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

[1] 这是用于 PWM0 的保留的管脚。

25.5.6 PWM控制寄存器（PWM0PCR -0xE001 404C和PWM1PCR -0xE001 804C）

PWM控制寄存器用来使能和选择每个PMW通道的类型。每个位的功能详见表 25.10。

表 25.10 PWM 控制寄存器（PWM0PCR–地址 0xE001 404C 和 PWM1PCR–地址 0xE001 804C）位描述

位	符号	值	描述	复位值
1:0	Unused		未使用，始终为 0。	NA
2	PWMSEL2		PWM2 输出单/双边沿模式控制。	0
		1	选择双边沿控制的模式。	
		0	选择单边沿控制的模式。	
3	PWMSEL3	1	PWM3 输出边沿控制。详见 PWMSEL2。	0
4	PWMSEL4	1	PWM4 输出边沿控制。详见 PWMSEL2。	0
5	PWMSEL5	1	PWM5 输出边沿控制。详见 PWMSEL2。	0
6	PWMSEL6	1	PWM6 输出边沿控制。详见 PWMSEL2。	0
8:7	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA
9	PWMENA1		PWM1 输出使能控制。	0
		0	禁止 PWM 输出。	
		1	使能 PWM 输出。	
10	PWMENA2		PWM2 输出使能控制。详见 PWMENA1。	0
11	PWMENA3		PWM3 输出使能控制。详见 PWMENA1。	0
12	PWMENA4		PWM4 输出使能控制。详见 PWMENA1。	0
13	PWMENA5		PWM5 输出使能控制。详见 PWMENA1。	0
14	PWMENA6		PWM6 输出使能控制。详见 PWMENA1。	0
31:15	Unused		未使用，始终为 0。	NA

25.5.7 PWM锁存使能寄存器（PWM0LER -0xE001 4050 和PWM1LER -0xE001 8050）

当 PWM 匹配寄存器用来产生 PWM 时，PWM 锁存使能寄存器用来控制 PWM 匹配寄存器的更新。当定时器处于 PWM 模式时如果软件对 PWM 匹配寄存器地址执行写操作，那么写入的值将在映像寄存器中被保存，但不立即使用。

当 PWM 匹配 0 事件发生时（在 PWM 模式下，通常也会复位定时器），如果对应的锁存使能寄存器位已经置位，那么映像寄存器的内容将传送到实际的匹配寄存器中。此时，新的值将生效并决定下一个 PWM 周期。当发生新值传送时，LER 中的所有位都自动清零。在 PWMLER 中相应位置位和 PWM 匹配 0 事件发生之前，任何写入 PWM 匹配寄存器的值都不会影响 PWM 操作。

例如，当 PWM2 配置为双边沿操作并处于运行中时，用来改变时序的事件的典型序列如下：

- 将新值写入 PWM 匹配 1 寄存器。
- 将新值写入 PWM 匹配 2 寄存器。
- 写 PWMLER，同时置位位 1 和位 2。
- 更改的值将在下一次定时器复位时（当 PWM 匹配 0 事件发生时）生效。

写两个 PWM 匹配寄存器的顺序并不重要，因为写入的值在写 PWMLER 之前都是无效的。这样就确保了两个值可以同时生效（如果要求）。如果需要，也可用同样的方法来修改单个值。

PWMLER中所有位的功能如表 25.11所示。

表 25.11 PWM 锁存使能寄存器（PWM0LER –地址 0xE001 4050 和 PWM1LER – 地址 0xE001 8050）位描述

位	符号	描述	复位值
0	使能 PWM 匹配 0 锁存	将该位置位允许最后写入PWM匹配 0 寄存器的值在由PWM匹配事件引起的下次定时器复位时生效。见25.5.4 节“PWM匹配控制寄存器（PWM0MCR -0xE001 4014 和PWM1MCR- 0xE001 8014）”的描述。	0
1	使能 PWM 匹配 1 锁存	PWMMR1 寄存器更新控制。详见位 0。	0
2	使能 PWM 匹配 2 锁存	PWMMR2 寄存器更新控制。详见位 0。	0
3	使能 PWM 匹配 3 锁存	PWMMR3 寄存器更新控制。详见位 0。	0
4	使能 PWM 匹配 4 锁存	PWMMR4 寄存器更新控制。详见位 0。	0
5	使能 PWM 匹配 5 锁存	PWMMR5 寄存器更新控制。详见位 0。	0
6	使能 PWM 匹配 6 锁存	PWMMR6 寄存器更新控制。详见位 0。	0
7	-	保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

第26章 模数转换器（ADC）

26.1 特性

- 10 位逐次逼近式模数转换器；
- 8 个管脚复用为输入脚；
- 掉电模式；
- 测量范围：0～3V；
- 10 位转换时间≥2.44μs；
- 一个或多个输入的突发（burst）转换模式；
- 可选择由输入跳变或定时器匹配信号触发转换；
- 每个 A/D 通道的独立结果寄存器减少了中断开销。

26.2 描述

A/D 转换器的基本时钟由 APB 时钟（PCLK）提供。每个转换器包含一个可编程的分频器，它可将这个时钟调整为逐次逼近转换所需的 4.5MHz（最大）。完全满足精度要求的转换需要 11 个这样的时钟。

26.3 管脚描述

表 26.1总结了所有与ADC相关的管脚。

表 26.1 A/D 管脚描述

管脚名称	类型	描述
AD0[7:0]	输入	**模拟输入**。A/D 转换器单元可以测量所有这些输入信号上的电压。注意：这些模拟输入始终连接到它们的管脚上，即使管脚复用寄存器将它们设定为端口管脚。通过将这些管脚驱动成端口输出可实现 A/D 转换器的简单自测。 **注意**：尽管ADC管脚被设置成最大可承受 5V的电压（见第 8 章的8.2 “管脚配置”），但ADC模块上的模拟复用管脚所能承受的电压并非如此。大于$V_{DD(3V3)}$/VREF/3.3V(V_{DDA})＋10%的电压不能应用于被选择用作ADC输入的管脚上，否则读取ADC会出错。例如，AD0.0 和AD0.1 用作ADC0 输入管脚，电压分别为AD0.0=4.5V而AD0.1=2.5V，虽然AD0.1 输入管脚的电压在正常范围内，但AD0.0 管脚上过大的电压仍然会导致AD0.1 读取错误。 如果在应用中未使用 A/D 转换器，则与 A/D 输入相关的管脚就可以用作最大可承受 5V 电压的数字 IO 管脚。
VREF	参考电压	**参考电压**。该管脚为 A/D 转换器提供参考电压。
V_{DDA},V_{SSA}	电源	**模拟电源和地**。它们分别与标称为 $V_{DD(3V3)}$和 V_{SS} 的电压相同，但为了降低噪声和出错几率，两者应当隔离。

26.4 寄存器描述

ADC的基址为 0xE003 4000。A/D转换器包含的寄存器如表 26.2所示。

表 26.2 A/D 寄存器

名称	描述	访问	复位值[1]	地址
AD0CR	A/D 控制寄存器。A/D 转换开始前，必须写入 AD0CR 寄存器来选择工作模式。	R/W	0x0000 0001	0xE003 4000
AD0GDR	A/D 全局数据寄存器，它包含最近一次 A/D 转换的结果。	R/W	NA	0xE003 4004
AD0STAT	A/D 状态寄存器。该寄存器包含所有 A/D 通道的 DONE 标记和 OVERRUN 标记，以及 A/D 中断标记。	RO	0	0xE003 4030
AD0INTEN	A/D 中断使能寄存器。该寄存器包含的使能位控制每一个 A/D 通道的 DONE 标记是否用来产生中断。	R/W	0x0000 0100	0xE003 400C
ADDR0	A/D 通道 0 数据寄存器。该寄存器包含在通道 0 上完成的最近一次转换的结果。	R/W	NA	0xE003 4010
ADDR1	A/D 通道 1 数据寄存器。该寄存器包含在通道 1 上完成的最近一次转换的结果。	R/W	NA	0xE003 4014
ADDR2	A/D 通道 2 数据寄存器。该寄存器包含在通道 2 上完成的最近一次转换的结果。	R/W	NA	0xE003 4018
ADDR3	A/D 通道 3 数据寄存器。该寄存器包含在通道 3 上完成的最近一次转换的结果。	R/W	NA	0xE003 401C
ADDR4	A/D 通道 4 数据寄存器。该寄存器包含在通道 4 上完成的最近一次转换的结果。	R/W	NA	0xE003 4020
ADDR5	A/D 通道 5 数据寄存器。该寄存器包含在通道 5 上完成的最近一次转换的结果。	R/W	NA	0xE003 4024
ADDR6	A/D 通道 6 数据寄存器。该寄存器包含在通道 6 上完成的最近一次转换的结果。	R/W	NA	0xE003 4028
ADDR7	A/D 通道 7 数据寄存器。该寄存器包含在通道 7 上完成的最近一次转换的结果。	R/W	NA	0xE003 402C

[1]复位值只反映了储存在所使用位中的数据，并不包含保留位的内容。

26.4.1 A/D控制寄存器（AD0CR - 0xE003 4000）

A/D 控制寄存器中的位可用于选择要转换的 A/D 通道、A/D 时序、A/D 模式和 A/D 启动触发器。

表 26.3 A/D 控制寄存器（AD0CR – 地址 0xE003 4000）位描述

位	符号	值	描述	复位值
7:0	SEL		从 AD0.7:0 中选择采样和转换的输入脚。对于 AD0，位 0 选择管脚 AD0.0，位 7 选择管脚 AD0.7。软件控制模式下，只有一位可被置位。硬件扫描模式下，任何一位都可以置位。全零等效于 0x01。	0x01

续表 26.3

位	符号	值	描述	复位值
15:8	CLKDIV		将 APB 时钟（PCLK）进行（CLKDIV 值+1）分频得到 A/D 转换时钟，该时钟必须小于或等于 4.5MHz。典型地，软件将 CLKDIV 编程为最小值来得到 4.5MHz 或稍低于 4.5MHz 的时钟，但某些情况下（例如高阻抗模拟电源）可能需要更低的时钟。	0
16	BURST	0	转换由软件控制，需要 11 个时钟方能完成。	0
		1	A/D 转换器以 CLKS 字段选择的速率重复执行转换，（如果必要）并扫描 SEL 字段中为 1 的位选择的管脚。A/D 转换器启动后第一次转换的是 SEL 字段中为 1 的位中的最低有效位对应的模拟输入，然后是为 1 的更高有效位对应的模拟输入（如果可用）。重复的转换可通过清零该位来终止，但该位被清零时并不会中止正在进行的转换。 **注：**当 BURST=1 时，起始位必须为 000，否则转换不会开始。	
19:17	CLKS		该字段用来选择突发模式下每次转换使用的时钟数和所得 ADDR 转换结果的 LS 位中可确保精度的位的数目，CLKS 可在 11 个时钟（10 位）～4 个时钟（3 位）之间选择。	000
		000	11 个时钟/10 位	
		001	10 个时钟/9 位	
		010	9 个时钟/8 位	
		011	8 个时钟/7 位	
		100	7 个时钟/6 位	
		101	6 个时钟/5 位	
		110	5 个时钟/4 位	
		111	4 个时钟/3 位	
20			保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA
21	PDN	1	A/D 转换器可选。	0
		0	A/D 转换器处于掉电模式。	
23:22	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA
26:24	START		当 BURST 位为 0 时，这些位控制着 A/D 转换是否启动和何时启动：	0
		000	不启动（PDN 清零时使用该值）。	
		001	立即启动转换。	
		010	当寄存器位 27 选择的边沿出现在 P2.10/EINT0 脚时启动转换。	
		011	当寄存器位 27 选择的边沿出现在 P1.27/CAP0.1 脚时启动转换。	
		100	当寄存器位 27 选择的边沿出现在 MAT0.1 时启动转换。	
		101	当寄存器位 27 选择的边沿出现在 MAT0.3 时启动转换。	
		110	当寄存器位 27 选择的边沿出现在 MAT1.0 时启动转换。	
		111	当寄存器位 27 选择的边沿出现在 MAT1.1 时启动转换。	
27	EDGE		该位只有在 START 字段为 010～111 时有效，在这种情况下：	0
		1	在所选 CAP/MAT 信号的下降沿启动转换	
		0	在所选 CAP/MAT 信号的上升沿启动转换	
31:28	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

26.4.2 A/D全局数据寄存器（AD0GDR - 0xE003 4004）

A/D 全局数据寄存器包含最近一次 A/D 转换的结果，其中包含数据、DONE 和 Overrun 标志以及与数据相关的 A/D 通道的数目。

表 26.4 A/D 全局数据寄存器（AD0GDR –地址 0xE003 4004）位描述

位	符号	描述	复位值
5:0	Unused	这些位读出时为 0。专门用于未来的扩展和分辨率更高的 A/D 转换器。	0
15:6	V/V_{REF}	当 DONE 为 1 时，该字段包含一个二进制数小数，用来代表 SEL 字段选中的 Ain 脚的电压，该电压通过分压 V_{DDA} 得到。该字段为 0 表明 Ain 脚的电压小于，等于或接近于 V_{SSA}；该字段为 0x3FF 表明 Ain 脚的电压接近于，等于或大于 V_{REF} 上的电压。	X
23:16	Unused	这些位读出时为 0。它们允许连续 A/D 值的累加，而不需要使用与门屏蔽，使得至少有 256 个值不会溢出到 CHN 字段。	0
26:24	CHN	这些位包含的是 LS 位的转换通道。	X
29:27	Unused	这些位读出为 0。它们用于未来 CHN 字段的扩展，使之兼容可以转换更多通道的 A/D 转换器。	0
30	OVERRUN	Burst 模式下，如果在产生 LS 位结果的转换前一个或多个转换结果被丢失和覆盖，该位置位。在非 FIFO 操作中，该位通过读 ADDR 寄存器清零。	0
31	DONE	A/D 转换结束时该位置位。该位在 ADDR 被读出和 ADCR 被写入时清零。如果 ADCR 在转换过程中被写入，那么该位置位并启动一次新的转换。	0

26.4.3 A/D状态寄存器（ADSTAT - 0xE003 4030）

A/D 状态寄存器允许同时检查所有 A/D 通道的状态。每个 A/D 通道的 ADDRn 寄存器的 DONE 和 OVERRUN 标志都反映在 ADSTAT 中。在 ADSTAT 中同样可以找到中断标记（所有 DONE 标志逻辑或的结果）。

表 26.5 A/D 状态寄存器（ADSTAT – 地址 0xE003 4030）位描述

位	符号	描述	复位值
7:0	Done7:0	这些位反映了每个 A/D 通道的结果寄存器中的 DONE 状态标志。	0
15:8	Overrun7:0	这些位反映了每个 A/D 通道的结果寄存器中的 OVERRUN 状态标志。对 ADSTAT 进行读操作可以同时检查所有 A/D 通道的状态。	0
16	ADINT	该位是 A/D 中断标志。当任何一个 A/D 通道的 Done 标记被声明和使能用于产生 A/D 中断时（通过 ADINTEN 寄存器），该位为 1。	0
31:17	Unused	未使用，始终为 0。	0

26.4.4 A/D中断使能寄存器（ADINTEN - 0xE003 400C）

转换完成时，该寄存器用来控制哪个 A/D 通道产生中断。例如，可能需要通过对某些 A/D 通道连续执行转换来监控传感器。最近一次的结果可根据需要随时由应用程序读出。在这种情况下，某些 A/D 通道转换结束时无需产生中断。

表 26.6 A/D 中断使能寄存器（ADINTEN – 地址 0xE003 400C）位描述

位	符号	描述	复位值
7:0	ADINTEN 7:0	转换完成时，这些位用来控制哪个 A/D 通道在转换结束时产生中断。如果位 0 为 1，则当 A/D 通道 0 的转换结束时产生中断，如果位 1 等于 1，则当 A/D 通道 1 的转换结束时产生中断，如此类推。	0x00
8	ADGINTEN	为 1 时，使能 ADDR 的全局 DONE 标志产生中断。为 0 时，只有个别由 ADINTEN7:0 使能的 A/D 通道才产生中断。	1
31:9	Unused	未使用，始终为 0。	0

26.4.5 A/D数据寄存器（ADDR0～ADDR7 - 0xE003 4010～0xE003 402C）

A/D 数据寄存器保存着 A/D 转换完成时的结果，同时包含指示转换已结束以及转换溢出的标志。

表 26.7 A/D 数据寄存器（ADDR0～ADDR7 –地址 0xE003 4010 到 0xE003 402C）位描述

位	符号	描述	复位值
5:0	Unused	未使用，始终为 0。 这些位读出时总是为 0。专门用于未来的扩展和分辨率更高的 A/D 转换器。	0
15:6	V/V_{REF}	当 DONE 为 1 时，该字段包含一个二进制数小数，用来代表 Ain 管脚的电压。该电压通过分压 V_{REF} 得到。该字段为 0 表明 Ain 脚的电压小于，等于或接近于 V_{REF}；该字段为 0x3FF 表明 Ain 脚的电压接近于，等于或大于 V_{REF} 上的电压。	NA
29:16	Unused	这些位读出时总是为 0。它们允许连续 A/D 值的累加，而不需要使用与门屏蔽，使得至少有 256 个值不会溢出到 CHN 字段。	0
30	OVERRUN	Burst 模式下，如果在产生 LS 位结果的转换前一个或多个转换结果被丢失和覆盖，该位置位。该位可通过读该寄存器清零。	0
31	DONE	A/D 转换结束时该位置位。此位在该寄存器被读出时清零。	0

26.5 操作

26.5.1 硬件触发的转换

如果 ADCR 的 BURST 位为 0 且 START 字段的值包含在 010-111 之内，那么当所选管脚或定时器匹配信号发生跳变时 A/D 转换器启动一次转换。也可选择在 4 个匹配信号中任何一个的指定边沿转换，或者在 2 个捕获/匹配管脚中任何一个的指定边沿转换。将所选端口的管脚状态或所选的匹配信号与 ADCR 的位 27 异或来作为边沿检测逻辑。

26.5.2 中断

当 ADSTAT 寄存器中的 ADINT 位为 1 时，中断被提交到向量中断控制器（VIC）。如果中断被使能（通过 ADINTEN 寄存器）的 A/D 通道中的任何 DONE 位为 1，则 ADINT 置位。软件通过 VIC 中 A/D 转换器对应的中断使能位来控制是否产生中断。DONE 标志必须通过读出正在产生中断的 A/D 通道的结果寄存器来清除。

26.5.3 精度和数字接收器

当A/D转换器用来测量AD0 脚的电压时，并不理会管脚选择寄存器中的管脚设置（见第 9 章的表 9.3“管脚连接模块寄存器映射”），但是通过禁能管脚的数字接收器来选择AD0 功能可以提高转换精度。

第27章 数模转换器（DAC）

27.1 特性

- 10 位数模转换器；
- 电阻串结构；
- 缓冲输出；
- 掉电模式；
- 可调节速度和功率。

27.2 管脚描述

表 27.1总结了所有与DAC相关的管脚。

表 27.1 D/A 管脚描述

管脚	类型	描述
AOUT	输出	**模拟输出**。当 DACR 写入新值后，经过所选的设定时间，该管脚的电压为 VALUE/1024×VREF（相对于 V_{SSA}）。
VREF	参考	**参考电压**。该管脚为 D/A 转换器提供参考电压。
V_{DDA}, V_{SSA}	电源	**模拟电源和地**。它们应当分别与 $V_{DD(3V3)}$和 V_{SS} 的电压相同，但为了降低噪声和出错几率，两者应当隔离。

27.3 寄存器描述（DACR – 0xE006 C000）

该读/写寄存器包含转换成模拟值的数字值和用来平衡性能和功耗的位。位 5:0 保留用于未来分辨率更高的 D/A 转换器。

表 27.2 D/A 转换器寄存器（DACR –地址 0xE006 C000）位描述

位	符号	值	描述	复位值
5:0	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA
15:6	VALUE		当该字段被写入新值后，经过一段所选的设定时间，管脚 Aout 上的电压为 VALUE/1024×VREF（相对于 V_{SSA}）。	0
16	BIAS	0	DAC的最大设定时间为1μs，最大电流为700μA。	0
		1	DAC 的最大设定时间为 2.5μs，最大电流为 350μA。	
31:17	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义。	NA

27.4 操作

PINSEL1 寄存器（见9.6.2 节“管脚功能选择寄存器 1（PINSEL1 – 0xE002 C004）”）的位 21:20 控制DAC的使能及管脚P0.26/AD0.3/AOUT/RXD3 的状态。当这些位的值为 10 时，DAC被上电并激活。

在 Aout 管脚上加载一个不超过 100pF 的电容的情况下，BIAS 位的描述中提到的设定时间是有效的。使用大于该值的阻抗将导致设定的时间超出指定时间。最终的数据手册将提供阻抗和设定时间的关系图。

第28章 实时时钟(RTC)和蓄电池RAM

28.1 特性

- 测量经过的时间以维护日历和时钟的运转；
- 超低功耗设计，以支持电池供电系统；
- 提供秒、分、小时、日期（月）、月、年、星期和日期（年）；
- 专门的 32 kHz 振荡器或来自 APB 时钟的可编程预分频器；
- 专门的电源管脚，可以与电池或者 3.3V 主电源相连；
- 报警输出管脚，帮助将系统从深度掉电模式中唤醒，或者在芯片上所有功能单元（RTC 和蓄电池 RAM 除外）的电源都被移除时使用；
- 周期性中断在时间寄存器任意字段的值递增时产生，并且可以选择小数秒的值（fractional second values）；
- 2k 字节静态 RAM，由 VBAT 供电；
- RTC 和蓄电池 RAM 电源与芯片的其它部分是分离的。

28.2 描述

实时时钟（RTC）是一组计数器，这些计数器用于测量系统上电时间，且在系统掉电时可以自由地运行。RTC 在掉电模式下仅消耗极低的功率。LPC2400 中的 RTC 时钟源可以是单独的 32.768kHz 振荡器，也可以是一个基于 APB 时钟的可编程的预分频器。此外，RTC 通过其自带的电源管脚 VBAT 供电，VBAT 可以与蓄电池相连，也可以与器件其它部分使用的 3.3V 电源管脚相连。

VBAT 管脚只向 RTC 和蓄电池 RAM 供电。这两个功能单元只需很低的功率就可工作，并且可以通过外部的蓄电池供电。当 CPU 和芯片上的其他功能单元停止运行且电源被移除时，RTC 会提供一个报警输出信号，通过该信号外部硬件可以恢复对芯片供电，使其可继续工作。报警输出信号的额定电压大约为 1.8V。

28.3 结构

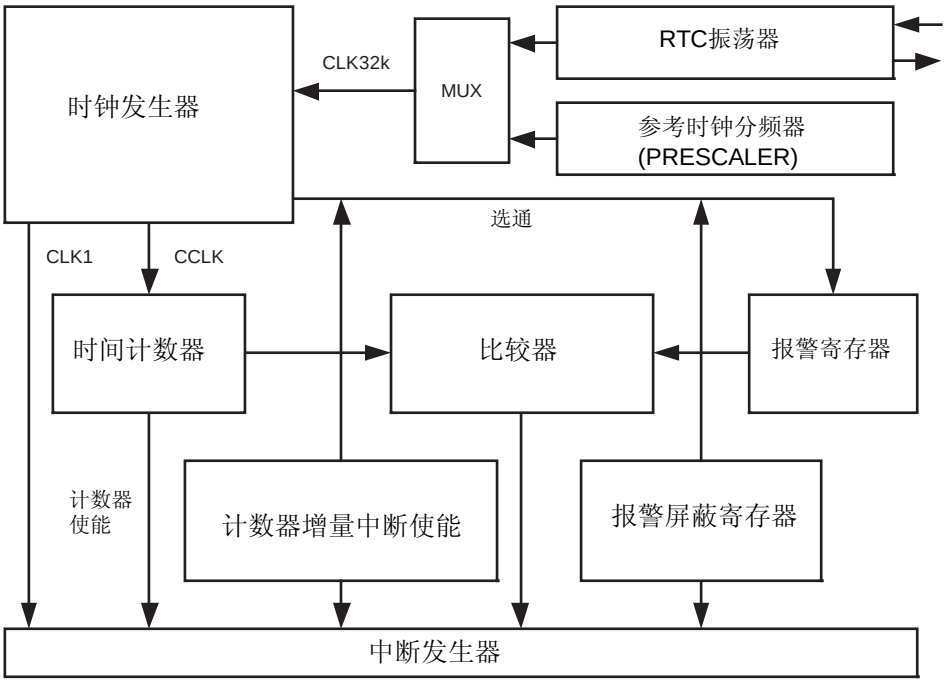


图 28.1 RTC 结构图

28.4 寄存器描述

RTC含有许多寄存器。地址空间根据功能分成 4 组。前 8 个地址供混合(miscellaneous)寄存器组（见28.4.2小节）使用。第二组的 8 个地址供时间计数器组（见28.4.4小节）使用，第三组的 8 个地址供报警寄存器组（见28.5小节）使用。剩下的寄存器控制基准时钟分频器。

实时时钟（RTC）模块包含的寄存器如表 28.1所示。具体的描述见下文。在这些描述中，大多数寄存器的“复位值”一栏，显示的都是“NC”，表示这些寄存器的值不因复位而改变。在从上电到RTC运行这段时间内，软件必须将这些寄存器初始化。

表 28.1 实时时钟寄存器映射

名称	规格	描述	访问	复位值[1]	地址
ILR	2	中断位置寄存器	R/W	*	0xE002 4000
CTC	15	时钟节拍计数器	RO	*	0xE002 4004
CCR	4	时钟控制寄存器	R/W	*	0xE002 4008
CIIR	8	计数器递增中断寄存器	R/W	*	0xE002 400C
AMR	8	报警屏蔽寄存器	R/W	*	0xE002 4010
CTIME0	32	完整时间寄存器 0	RO	*	0xE002 4014
CTIME1	32	完整时间寄存器 1	RO	*	0xE002 4018
CTIME2	32	完整时间寄存器 2	RO	*	0xE002 401C
SEC	6	秒计数器	R/W	*	0xE002 4020
MIN	6	分寄存器	R/W	*	0xE002 4024

续表 28.1

名称	规格	描述	访问	复位值[1]	地址
HOUR	5	小时寄存器	R/W	*	0xE002 4028
DOM	5	日期（月）寄存器	R/W	*	0xE002 402C
DOW	3	星期寄存器	R/W	*	0xE002 4030
DOY	9	日期（年）寄存器	R/W	*	0xE002 4034
MONTH	4	月寄存器	R/W	*	0xE002 4038
YEAR	12	年寄存器	R/W	*	0xE002 403C
CISS	8	次秒级中断的计数器增量选择屏蔽寄存器	R/W	*	0xE002 4040
ALSEC	6	秒报警值	R/W	*	0xE002 4060
ALMIN	6	分报警值	R/W	*	0xE002 4064
ALHOUR	5	小时报警值	R/W	*	0xE002 4068
ALDOM	5	日期（月）报警值	R/W	*	0xE002 406C
ALDOW	3	星期报警值	R/W	*	0xE002 4070
ALDOY	9	日期（年）报警值	R/W	*	0xE002 4074
ALMON	4	月报警值	R/W	*	0xE002 4078
ALYEAR	12	年报警值	R/W	*	0xE002 407C
PREINT	13	预分频值，整数部分	R/W	0	0xE002 4080
PREFRAC	15	预分频值，小数部分	R/W	0	0xE002 4084

[1] RTC 当中除预分频器部分之外的其它寄存器都不受器件复位的影响。如果 RTC 使能，这些寄存器必须通过软件来初始化。复位值仅反映存放在使用位的数据，不包括保留位的内容。

28.4.1 RTC中断

中断的产生由中断位置寄存器(ILR)、计数器递增中断寄存器(CIIR)、报警寄存器和报警屏蔽寄存器(AMR)控制。只有转换到中断状态才能产生中断。ILR 单独使能 CIIR 和 AMR 中断。CIIR 中的每个位都对应一个时间计数器。如果 CIIR 使能用于一个特定的计数器，那么该计数器的值每增加一次就产生一个中断。用户通过使用报警寄存器可以设定产生中断的日期或时间。AMR 提供一个屏蔽报警比较的机制。当所有非屏蔽报警寄存器均与它们对应的时间计数器的值相匹配时，就会产生中断。

如果RTC是在自带的振荡器（RTCX1-2 管脚）下工作，那么RTC中断就可以让微控制器退出掉电模式。如果RTC中断使能用于唤醒系统且出现其选定的事件，那么与XTAL1/2 管脚相关的振荡器唤醒周期开始。要详细了解基于RTC的唤醒过程，请参考本用户手册的4.7.8 小节“中断唤醒寄存器（INTWAKE – 0xE01F C144）”和4.8 小节“唤醒定时器”。

28.4.2 混合寄存器组

表 28.2所示为A[6:2]的 0 到 7 寄存器。详见下文具体的描述。

表 28.2 混合寄存器

名称	规格	描述	访问	地址
ILR	3	中断位置寄存器。该单元的读取结果指示了中断源。向该寄存器的某个位写入 1 可以清除相应的中断。	R/W	0xE002 4000
CTC	15	时钟节拍计数器。表示时钟分频器的值。	RO	0xE002 4004
CCR	4	时钟控制寄存器。控制时钟分频器的操作情况。	R/W	0xE002 4008
CIIR	8	计数器递增中断寄存器。当多个计数器递增时，决定由哪个计数器产生中断。	R/W	0xE002 400C
AMR	8	报警屏蔽寄存器。控制屏蔽哪个报警寄存器。	R/W	0xE002 4010
CTIME0	32	完整时间寄存器 0。	RO	0xE002 4014
CTIME1	32	完整时间寄存器 1。	RO	0xE002 4018
CTIME2	32	完整时间寄存器 2。	RO	0xE002 401C

28.4.2.1 中断位置寄存器（ILR - 0xE002 4000）

中断位置寄存器为 2 位寄存器，它指定哪些模块产生中断（见表 28.3）。向一个位写入 1 会清除相应的中断。写入 0 无效。因而程序员可以读取该寄存器并将读出的值回写到寄存器中以清除检测到的中断。

表 28.3 中断位置寄存器（ILR – 地址 0xE002 4000）位描述

位	符号	描述	复位值
0	RTCCIF	为 1 时，计数器增量中断模块产生中断。向该位写入 1 清除计数器增量中断。	NC
1	RTCALF	为 1 时，报警寄存器产生中断。向该位写入 1 清除报警中断。	NC
2	RTSSF	为 1 时，产生计数器增量次秒级中断。每秒产生的中断次数由 CISS 寄存器决定。	NC
7:2	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义	NA

28.4.2.2 时钟节拍计数器寄存器（CTCR - 0xE002 4004）

本寄存器为只读寄存器。它可通过时钟控制寄存器（CCR）复位为 0。CTC 由时钟分频计数器的位组成。

表 28.4 时钟节拍计数器（CTCR – 地址 0xE002 4004）位描述

位	符号	描述	复位值
14:0	时钟节拍计数器	位于秒计数器之前，CTC每秒计数 32,768 个时钟。由于RTC预分频器的关系，这 32,768 个时间增量的长度可能不相同。详见28.8.1 小节“基准时钟分频器（预分频器）”的描述。	NA
15	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.4.2.3 时钟控制寄存器（CCR - 0xE002 4008）

时钟控制寄存器是一个 4 位寄存器，它控制时钟分频电路的操作。每位的功能详见下表。

表 28.5 时钟控制寄存器（CCR – 地址 0xE002 4008）位描述

位	符号	描述	复位值
0	CLKEN	时钟使能；当该位为 1 时，时间计数器使能。为 0 时，时间计数器都被禁止，这时可对其进行初始化。	NA
1	CTCRST	CTC 复位；为 1 时，时钟节拍计数器复位。在 CCR[1]变为 0 之前，它将一直保持复位状态。	NA
3:2	CTTEST	测试使能；在正常操作中，这些位应该全为 0。	NA
4	CLKSRC	如果该位为 0，时钟节拍计数器(CTC)的时钟源将采用预分频器，这点和NXP早期的嵌入式ARM系列器件是相同的。如果该位为 1，则CTC时钟源将采用与RTCX1 和RTCX2 管脚相连的 32 kHz振荡器（硬件的详细内容请见28.10 节“RTC外部 32kHz振荡器元件的选择”）。	NA
7:5	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义	NA

28.4.2.4 计数器增量中断寄存器（CIIR – 0xE002 400C）

计数器增量中断寄存器（CIIR）可以让计数器在每次计数值增加时产生一次中断。这个中断在中断位置寄存器的位 0(ILR[0])写入 1 之前，一直有效。

表 28.6 计数器增量中断寄存器（CIIR - 0xE002400C）位描述

位	符号	描述	复位值
0	IMSEC	为 1 时，每增加一秒就产生一次中断。	NA
1	IMMIN	为 1 时，每增加一分钟就产生一次中断。	NA
2	IMHOUR	为 1 时，每增加一小时就产生一次中断。	NA
3	IMDOM	为 1 时，月份的日期增加一次就产生一次中断。	NA
4	IMDOW	为 1 时，星期的日期增加一次就产生一次中断。	NA
5	IMDOY	为 1 时，年的日期增加一次就产生一次中断。	NA
6	IMMON	为 1 时，每增加一个月就产生一次中断。	NA
7	IMYEAR	为 1 时，每增加一年就产生一次中断。	NA

28.4.2.5 计数器增量选择屏蔽寄存器(CISS – 0xE002 4040)

CISS 寄存器提供了一种可以从实时时钟那里获取毫秒（millisecond）级周期性 CPU 中断的方法。它可以释放一个通用定时器，也可以在一个周期性中断与另外一个周期性中断之间让 CPU 进入低功耗模式，从而减少了功耗。

我们可以使用一个进位输出信号来产生次秒级中断，该信号能够在时钟节拍计数器的不同计时阶段触发中断。计时范围从 16 个 CTC 计数值（大约为 488 微秒）到 2048 个 CTC 计数值（大约为 62.5 毫秒）。可用的计数值和对应的时间如下表所示。

表 28.7 计数器增量选择屏蔽寄存器(CISS – 地址 0xE002 4040)位描述

位	符号	值	描述	复位值
2:0	SubSecSel	SubSecSel：次级秒选择。该字段用于选择产生次秒级中断的计数值，具体如下：		NC
		000	时钟节拍计数器每计数 16 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 488 微秒就产生一次中断。	
		001	时钟节拍计数器每计数 32 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 977 微秒就产生一次中断。	
		010	时钟节拍计数器每计数 64 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 1.95 毫秒就产生一次中断。	
		011	时钟节拍计数器每计数 128 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 3.9 毫秒就产生一次中断。	
		100	时钟节拍计数器每计数 256 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 7.8 毫秒就产生一次中断。	
		101	时钟节拍计数器每计数 512 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 15.6 毫秒就产生一次中断。	
		110	时钟节拍计数器每计数 1024 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 31.25 毫秒就产生一次中断。	
		111	时钟节拍计数器每计数 2048 次就产生一次中断。当频率为 32.768 kHz 时，大约每隔 62.5 毫秒就产生一次中断。	
6:3	Unused	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。		NA
7	SubSecEna	次秒级中断使能。		NC
		0	禁止次秒级中断。	
		1	使能次秒级中断。	

28.4.2.6 报警屏蔽寄存器（AMR – 0xE002 4010）

报警屏蔽寄存器（AMR）允许用户屏蔽任意报警寄存器。下表列出了 AMR 寄存器的位与报警寄存器位之间的关系。为了实现报警功能，每个非屏蔽的报警寄存器的值必须和对应的时间计数值匹配，这样才能产生中断。中断只在计数器比较结果第一次从不匹配到匹配时产生。向中断位置寄存器（ILR）的位写入 1 会清除相应的中断。如果所有屏蔽位都置位，则报警将被禁止。

表 28.8 报警屏蔽寄存器（AMR –地址 0xE002 4010）位描述

位	功能	描述	复位值
0	AMRSEC	为 1 时，秒计数值不与报警寄存器比较。	NA
1	AMRMIN	为 1 时，分计数值不与报警寄存器比较。	NA
2	AMRHOUR	为 1 时，小时计数值不与报警寄存器比较。	NA
3	AMRDOM	为 1 时，日期（月）计数值不与报警寄存器比较。	NA
4	AMRDOW	为 1 时，星期计数值不与报警寄存器比较。	NA
5	AMRDOY	为 1 时，日期（年）计数值不与报警寄存器比较。	NA
6	AMRMON	为 1 时，月计数值不与报警寄存器比较。	NA
7	AMRYEAR	为 1 时，年计数值不与报警寄存器比较。	NA

28.4.3 完整时间寄存器

时间计数器的值可以选择以一个完整格式读出，这样程序员只需执行 3 次读操作即可读出所有时间计数器值，每个寄存器都为 32 位，见表 28.9，表 28.10和表 28.11。每个寄存器的最低位分别位于位 0、位 8、位 16 或位 24。

完整时间寄存器为只读寄存器。要更新时间计数器的值，必须对时间计数器寻址。

28.4.3.1 完整时间寄存器 0（CTIME0 - 0xE002 4014）

完整时间寄存器 0 包含的是低位时间，它们是：秒、分、小时和天（星期）。

表 28.9 完整时间寄存器 0（CTIME0 – 地址 0xE002 4014）位描述

位	符号	描述	复位值
5:0	秒	秒值的范围为 0~59。	NA
7:6	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
13:8	分	分值的范围为 0~59。	NA
15:14	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
20:16	小时	小时值的范围为 0~23。	NA
23:21	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
26:24	天（星期）	天数的范围为 0~6。	NA
31:27	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.4.3.2 完整时间寄存器 1（CTIME1 - 0xE002 4018）

完整时间寄存器 1 包含的时间为：日期（月）、月和年。

表 28.10 完整时间寄存器 1（CTIME1 – 地址 0xE002 4018）位描述

位	符号	描述	复位值
4:0	日期（月）	日期（月）；范围为 1~28/ 29/ 30/ 31（取决于具体的月份和是否为闰年）。	NA
7:5	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
11:8	月	月；范围为 1~12。	NA
15:12	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
27:16	年	年；范围为 0~4095。	NA
31:28	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.4.3.3 完整时间寄存器 2（CTIME2 - 0xE002 401C）

完整时间寄存器 2 仅包含的值为：日期（年）。

表 28.11 完整时间寄存器 2（CTIME2 – 地址 0xE002 401C）位描述

位	符号	描述	复位值
11:0	日期（年）	日期（年）；范围为 1~365（闰年为 366）。	NA
31:12	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.4.4 时间计数器组

时间值由 8 个计数器组成，如表 28.12和表 28.13所示。这些计数器可在如表 28.13所示的单元处执行读写操作。

表 28.12 时间计数器的关系和值

计数器	规格	使能	最小值	最大值
秒	6	Clk1（见图 28.1）	0	59
分	6	秒	0	59
小时	5	分	0	23
日期（月）	5	小时	1	28, 29, 30 或 31
星期	3	小时	0	6
日期（年）	9	小时	1	365 或 366（闰年）
月	4	日期（月）	1	12
年	12	月或日期（年）	0	4095

表 28.13 时间计数器寄存器

名称	规格	描述	访问	地址
SEC	6	秒；范围为 0~59。	R/W	0xE002 4020
MIN	6	分；范围为 0~59。	R/W	0xE002 4024
HOUR	5	小时；范围为 0~23。	R/W	0xE002 4028
DOM	5	日期（月）；范围为 1~28,29,30 或 31（取决于具体的月份和是否为闰年）。[1]	R/W	0xE002 402C
DOW	3	星期；范围为 0~6。[1]	R/W	0xE002 4030
DOY	9	日期（年）；范围为 1~365（闰年为 366）。[1]	R/W	0xE002 4034
MONTH	4	月；范围为 1~12。	R/W	0xE002 4038
YEAR	12	年；范围为 0~4095。	R/W	0xE002 403C

[1] 这些值只能在适当的时间间隔处递增且在已定义的溢出点复位。为了使这些值有意义，它们不能进行计算且必须被正确初始化。

28.4.4.1 润年计算

RTC 执行一个简单的位比较，检查年计数器的低 2 位是否为 0。如果为 0，则 RTC 认为这一年为闰年。RTC 将所有能被 4 整除的年份都看作闰年。该算法在计算 1901 年－2099 年都是正确的，但在计算 2100 年却出错，因为 2100 年并不是闰年。闰年对 RTC 的影响只是 2 月份的长度、日期（月）和年的计数值发生了变化。

28.5 报警寄存器组

报警寄存器如下表所示。这些寄存器的值与时间计数器进行比较。如果所有未屏蔽（见28.4.2.6 小节“报警屏蔽寄存器（AMR – 0xE002 4010）”）的报警寄存器的值都与它们对应的时间计数器匹配，那么将会产生一次中断。向中断位置寄存器的位 1（ILR[1]）写入 1 清除中断。

表 28.14 报警寄存器

名称	规格	描述	访问	地址
ALSEC	6	秒报警值	R/W	0xE002 4060
ALMIN	6	分报警值	R/W	0xE002 4064
ALHOUR	5	小时报警值	R/W	0xE002 4068
ALDOM	5	日期（月）报警值	R/W	0xE002 406C
ALDOW	3	日期（星期）报警值	R/W	0xE002 4070
ALDOY	9	日期（年）报警值	R/W	0xE002 4074
ALMON	4	月报警值	R/W	0xE002 4078
ALYEAR	12	年报警值	R/W	0xE002 407C

28.6 报警输出

RTC 含有一个报警输出管脚，用来反映报警寄存器比较的结果和 RTC 中断的情况。该管脚属于 RTC 电源范围，因此只要向 VBAT 供电，它就可以在节电模式使用。由于报警管脚结合了报警和 RTC 中断功能，所以可以向外部提供特定的时间/日期等等，或者是周期间隔。例如，我们可以使用时间（time of day）报警来通知外部电路向 LPC2400 供电，从而将系统从深度掉电模式中唤醒。

28.7 RTC使用注意事项

RTC 时钟源可以是 32.786 kHz 的 RTC 振荡器，也可以是经过基准时钟分频器调节后的 APB 外设时钟（PCLK）。

要使用 RTC，就必须将 VBAT 与管脚 $V_{DD(3V3)}$或独立的电源（外部蓄电池）相连。否则，VBAT 应该接地（与 V_{SS} 连接）。LPC2400 在 VBAT 掉电时不能保持 RTC 的状态，如果时钟源丢失、被中断或改变，LPC2400 也无法维持时间继续计数。

由于 RTC 时钟源为其中一个可用的 APB 时钟（PCLK）或 RTCX1-2 管脚上的 32 kHz 信号，所以所选时钟出现的任何中断都将会导致时间值的偏移。RTC 初始化成一个时间值，或者从 RTC 激活后运行的一段时间内出现了一个错误，这两种情况所带来的变化都就将影响真实的时钟时间。

RTCX1-2 管脚上的信号在任意时刻都可以向 RTC 时钟供电，如果选择 PCLK 作为 RTC 时钟源，则进入掉电模式将会使时间的更新出现误差。同样，将 PCLK 作为 RTC 的时钟源并在系统操作过程中改变时间基准（通过重新配置 PLL、APB 分频器或 RTC 预分频器），时间将会产生某些累加误差。当 RTC 时钟源在 PCLK 和 RTCX 管脚之间进行切换时，也会产生累加时间误差。

一旦选择RTCX1-2 管脚上的 32 kHz信号作为时钟源，RTC无需使用APB时钟（PCLK）就可以完全工作。因此，对功率敏感的应用场合（如使用蓄电池等场合）如果使用RTC，通过使用RTCX1-2 管脚上的信号，以及向PCONP功率控制寄存器（见4.7 小节“功率控制”）的PCRTC位写入 0，将可以减少功耗。

28.8 RTC时钟的产生

RTC 时钟源可以是 32.786 kHz 的 RTC 振荡器，也可以是经过基准时钟分频器调节后的 APB 外设时钟（PCLK）。

28.8.1 基准时钟分频器（预分频器）

当 RTC 时钟源不是由 RTC 振荡器提供，而是来自 APB 外设时钟（PCLK）时，我们就可以使用基准时钟分频器（在下文中称为预分频器）。

预分频器允许从任何大于和等于 65.536 kHz（2×32.768kHz）的外设时钟 PCLK 频率那里产生一个 32.768kHz 的基准时钟。这样，不管外设时钟的速率为多少，RTC 总是以正确的速率运行。通常，预分频器是通过一个包含整数和小数部分的值对外设时钟 PCLK 进行分频。因而不能连续输出一个恒定频率。有些时钟周期比其它周期多 1 个 PCLK，但是每秒钟的计数总数总是 32,768。

基准时钟分频器包含一个 13 位整数计数器和一个 15 位小数计数器。使用这种类型计数器的原因如下：

1. 要获得 LPC2400 所支持的频率，必须使用 13 位整数计数器。可以通过这样计算：频率 160MHz 除以 32,768 再减去 1 等于 4881，余数为 26,624。保存 4881 需要 13 个位，但 13 个位实际可以支持的最高频率为 268.4MHz（32,768×8192）。
2. 余数的最大值为 32,767，需要 15 位来保存。

表 28.15 基准时钟分频寄存器

名称	规格	描述	访问	地址
PREINT	13	预分频值，整数部分	R/W	0xE002 4080
PREFRAC	15	预分频值，小数部分	R/W	0xE002 4084

28.8.2 预分频整数寄存器（PREINT - 0xE002 4080）

这是预分频值的整数部分，通过以下等式计算得到：

PREINT = int (PCLK/32768) – 1。PREINT 的值必须大于或等于 1。

表 28.16 预分频整数寄存器（PREINT - 0xE002 4080）位描述

位	符号	描述	复位值
12:0	预分频整数	包含的是 RTC 预分频值的整数部分。	0
15:13	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.8.3 预分频小数寄存器（PREFRAC - 0xE002 4084）

这是预分频值的小数部分，通过以下等式计算得到：

PREFRAC = PCLK – ((PREINT+1)×32768)

表 28.17 预分频小数寄存器（PREFRAC - 0xE002 4084）位描述

位	符号	描述	复位值
14:0	预分频小数	包含的是 RTC 预分频值的小数部分。	0
15	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

28.8.4 预分频器的使用范例

举一种最简单的情形，假设 PCLK 的频率为 65.537 kHz，那么：

PREINT = int (PCLK/32768) – 1 = 1 且

PREFRAC = PCLK – ([PREINT+1]×32768) = 1

在预分频器的上述设置，每秒钟有 32,767 次 2 个 PCLK 周期计数，1 次 3 个 PCLK 周期计数，加起来每秒刚好为 RTC 提供 32,768 个时钟。

再假设一种比较实际的情形，假设 PCLK 的频率为 10MHz。那么：

PREINT = int (PCLK/32768) – 1 = 304 且

PREFRAC = PCLK – ((PREINT+1)×32768) = 5,760

这时，5,760 个预分频器输出时钟宽度将为 306 (305+1)个 PCLK 周期。余下的时钟宽度为 305 个 PCLK 周期。

采用相似的方法可以将任何高于 65.536 kHz 的 PCLK 频率(每秒钟的周期数必须为偶数）转换成 RTC 的 32 kHz 基准时钟。唯一需要注意的是，如果 PREFRAC 不包含 0，那么每秒当中的 32,768 个时钟长度是不完全相同的，有些时钟会比其它时钟多 1 个 PCLK 周期。虽然较长的脉冲已经尽可能地分配到剩余的脉冲当中，但是在希望直接观察时钟节拍计数器(CTC)的内容的某些应用场合下可能就需要注意这种“抖动（jitter）”了。

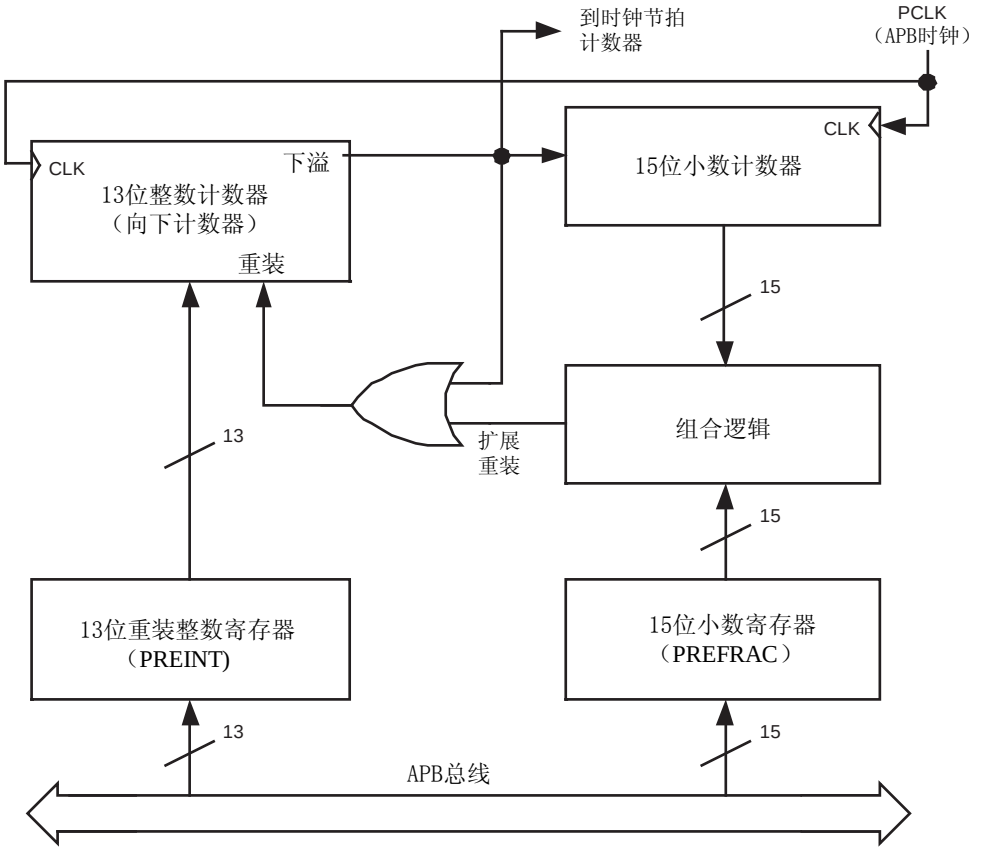


图 28.2 RTC 预分频器结构图

28.8.5 预分频器操作

在图 28.2的预分频器结构图中，标为“组合逻辑”的模块决定了 13 位PREINT计数器的

递减操作何时延长 1 个PCLK。为了使插入的延长周期数目正确并对它们进行合理地分配，组合逻辑将PREFRAC的每个位与15位小数计数器的值结合起来。这种组合关系如表 28.18所示。

例如，如果 PREFRAC 的位 14 为 1（表示小数 1/2），那么 13 位计数器计数的半个周期必须延长。当小数计数器的 LSB 位为 1 时，组合逻辑将使每次计数变化（每当小数计数器的 LSB 位为 1 时）的时间延长 1 个 PCLK，尽量平均分配脉冲宽度。类似地，PREFRAC 的位 13 为 1（表示小数 1/4）时，由 13 位计数器计数的每个 1/4 个周期（每当小数计数器的低 2 位＝10 时）会延长。

表 28.18 整数计数器重装值递增时预分频器的情况

小数计数器	PREFRAC 位														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
--- ---- ---- ---1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- --10	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- -100	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---- 1000	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
--- ---- ---1 0000	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-
--- ---- --10 0000	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
--- ---- -100 0000	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
--- ---- 1000 0000	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
--- ---1 0000 0000	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-
--- --10 0000 0000	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
--- -100 0000 0000	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
--- 1000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
--1 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-
-10 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-
100 0000 0000 0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

28.9 蓄电池RAM

蓄电池 RAM 是位于 APB 总线上的 2k 字节静态 RAM。地址范围为：0xE008 4000 到 0xE008 47FF。蓄电池 RAM 和 RTC 都由 V_{BAT} 管脚供电，两者均位于与芯片其它部分分离的电源区。这样，即使芯片的电源被移除，它们也可以正常工作。

28.10 RTC外部 32kHz振荡器元件的选择

RTC 外部振荡电路如下图所示，由于已经将反馈电阻集成到芯片内，所以外部只有一个晶体、电容 C_{X1} 和 C_{X2} 需要与微控制器连接。

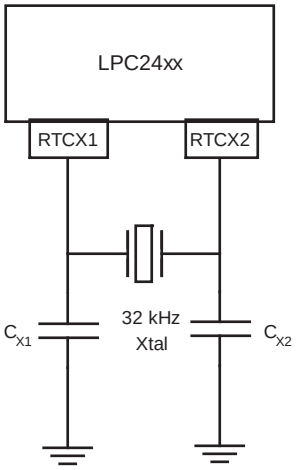


图 28.3 RTC 32 kHz 晶振电路

下表列出了应该使用的晶体参数值。C_L 表示晶体的典型负载电容，它通常由晶体制造商指定。实际的 C_L 值会影响振荡频率。如果使用一个专为不同负载电容而制造的晶体，那么电路的振荡频率与指定的频率将会有点差异（具体取决于晶体的属性）。因此，要得到一个准确的时间基准，建议使用下表中的 C_L 值。表中，外部电容 C_{X1} 和 C_{X2} 的值是由内部寄生电容和 C_L 计算得到的。来自 PCB 和封装的寄生电容不考虑在内。

表 28.19 RTC 外部 32kHz 振荡器 $C_{X1/X2}$ 组件的建议值

晶体负载电容 C_L	R_S 的最大值	外部负载电容 $C_{X1,CX2}$
11pF	< 100 KΩ	18 pF,18 pF
13pF	< 100 KΩ	22 pF, 22 pF
15pF	< 100 KΩ	27 pF,27 pF

第29章 Flash存储器编程固件

29.1 Flash boot装载程序

Boot 装载程序控制复位后的初始化操作，并提供实现 Flash 存储器编程的方法。BOOT 装载器可启动对空片的编程、已编程器件的擦除和再编程以及在系统运行时使用应用程序对 Flash 存储器进行编程。

29.2 特性

- 在系统编程：在系统编程(ISP)是使用 boot 装载程序软件和 UART0 串口对片内 Flash 存储器进行编程和再编程的一种方法。
- 在应用编程：在应用编程(IAP)是按照最终用户的应用代码指示，对片内 Flash 存储器执行擦除和写操作的一种方法。

29.3 应用

Flash boot 装载程序同时提供 ISP 和 IAP 编程接口，可以实现对片内 Flash 存储器的编程。

29.4 描述

Flash boot 装载程序代码在器件上电或复位时执行。装载程序可执行 ISP 命令处理程序或用户应用代码。P2.10 复位后的低电平被当作启动 ISP 命令处理器的外部硬件请求。假定在 **RESET** 管脚上产生上升沿时电源管脚在指定的电平下操作，那么在采样 P2.10 之前最多只允许有 3ms 的时间，并且还要确定是执行用户代码还是 ISP 处理程序。如果 P2.10 采样为低电平且看门狗溢出标志置位，那么启动 ISP 命令处理器的外部硬件请求将被忽略。在没有 ISP 命令处理器的请求（P2.10 复位后采样为高电平）时将搜索有效的用户程序。若找到了有效的用户程序，执行控制权就被转交给用户程序。若没有找到有效的用户程序，就将调用自动波特率程序。

在将管脚 P2.10 作为 ISP 硬件请求时需要特别注意：由于 P2.10 在复位后处于高阻模式，所以要使该管脚的状态稳定，用户需要提供外部硬件（上拉电阻或其它器件）。否则一不小心可能就进入了 ISP 模式。

在上电复位后进入ISP模式时，IRC和PLL被用于产生 14,748MHz的CCLK频率。当ISP 通过用户应用程序来调用时可能不会出现这种情况（见29.9.8 节“重新调用ISP”）。

29.4.1 复位后的存储器映射

Boot 区的 Flash 部分为 8kB 大小，位于片内 Flash 存储器最顶端（从 0x0007 E000 开始 。复位后，整个 boot 区也被映射到片内存储器空间的顶端，即 boot 区位于从地址 0x7FFF E000 开始的存储区。Flash boot 装载程序就从这片存储区运行，而 ISP 和 IAP 程序也均使用部分片内 RAM。至于 RAM 的用法，我们稍后会作介绍。位于片内 Flash 存储器 boot 区的中断向量，它在复位后也被激活，即 boot 区的最低 64 字节也可在从地址 0x0000 0000

开始的存储区中看到。复位向量包含一条跳转指令，通过该指令可以跳转到 Flash boot 装载程序的入口处。

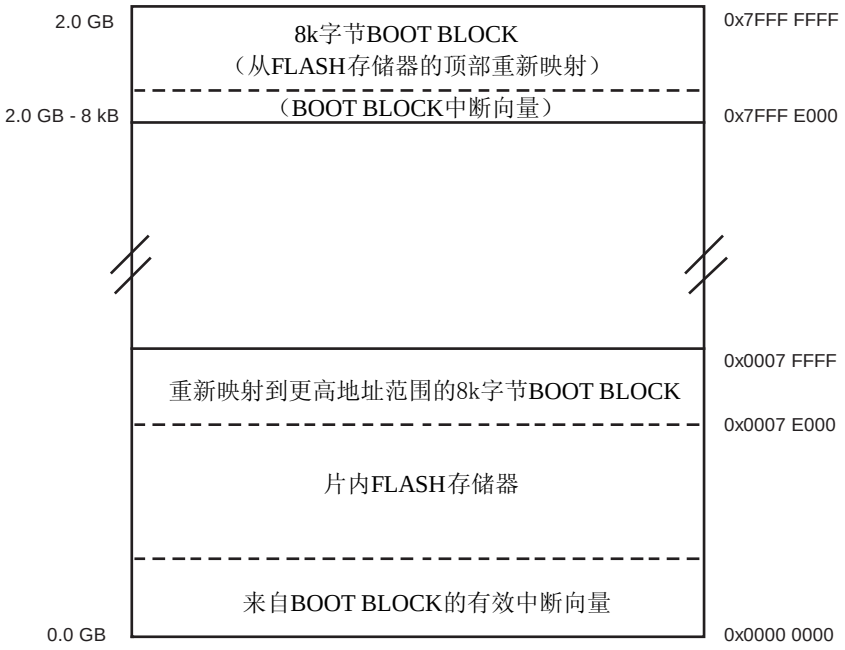


图 29.1 复位后低地址存储器的映射

29.4.1.1 有效用户代码的判定标准

有效用户代码的判定标准:保留的 ARM 中断向量单元（0x0000 0014）应当包含剩余中断向量校验和的 2 的补码，这样就使所有向量的校验和为 0。boot 装载程序代码首先保证 boot 区内的中断向量不发生重叠，然后再计算 Flash 扇区 0 中中断向量的校验和。如果符号差（signature）匹配，那么通过将 0x0000 0000 装入程序计数器，执行控制权便可被转交给用户代码。因而用户 Flash 复位扇区应该包含一条跳转到用户应用程序代码入口的跳转指令。

如果符号差（signature）无效，那么自动波特率程序通过串口 0 可以与主机同步。主机应当发送一个同步字符 ‘?’(0x3F)并等待响应。主机的串口应设定为 8 个数据位、1 个停止位和无奇偶校验。自动波特率程序根据自身的频率测量接收到的同步字符的位时间并对串口波特率发生器进行编程。此外，它还向主机发送一个 ASCII 字符串（“ Synchronized<CR><LF> ”），相应地主机应该将接收到的该字符串（“Synchronized<CR><LF>”）作为响应再发送给自动波特率程序。自动波特率程序通过观察接收到的字符来验证是否同步。如果通过验证，则向主机发送“OK<CR><LF>”字符串。主机应当通过发送部件运行时的晶振频率（单位为 kHz）来响应。例如，如果工作频率为 10MHz，那么主机的响应将为“10000 <CR><LF>”。在自动波特率程序接收到晶振频率后再向主机发送“OK<CR><LF>”字符串。如果没有通过同步验证，那么自动波特率程序再次等待一个同步字符。在用户调用 ISP 的情况下要让自动波特率正确工作，晶振频率应当大于或等于 10MHz。

要详细了解复位、PLL和启动/引导代码之间的相互作用，请参考4.5.2 小节“PLL和启动/引导代码之间的相互作用”。

在接收到晶振频率后，执行初始化并调用ISP命令处理程序。出于安全方面的考虑，在执行Flash编程/擦除操作命令和“Go”命令之前必须执行“Unlock（解锁）”命令。其它命

令不需要执行解锁命令。每个ISP过程都要执行一次“Unlock（解锁）”命令。解锁命令在29.8小节“ISP命令”中介绍。

29.4.2 通信协议

所有 ISP 命令都以单个 ASCII 字符串的形式发送。字符串应当以回车（CR）和/或换行（LF）控制字符作为结束符。多余的<CR>和<LF>均被忽略。所有 ISP 响应都是以<CR><LF>结束的 ASCII 字符串形式发送。数据是以 UU 编码格式发送和接收。

29.4.2.1 ISP命令格式

“命令 参数_0 参数_1 ... 参数_n<CR><LF>”“数据”（只适用于写命令）

29.4.2.2 ISP响应格式

“返回_代码<CR><LF>响应_0<CR><LF>响应_1<CR><LF> ... 响应_n<CR><LF>" "数据"（只适用于读命令）

29.4.2.3 ISP数据格式

数据流采用 UU 编码格式。UU 编码算法将 3 字节二进制数据转换成 4 字节可打印的 ASCII 字符集，而 Hex 格式是将 1 字节二进制数据转换成 2 字节 ASCII Hex 数据，所以 UU 编码的效率要高于十六进制（Hex）格式。发送器在发送校验和之前应当先发送 20 个 UU 编码行。任何 UU 编码行的长度都不应超过 61 个字符（字节），也就是说它可以存放 45 个数据字节。接收器应当将该校验和与接收数据的校验和进行比较。如果校验和匹配，那么接收器应该以“OK<CR><LF>”作为响应，以继续发送操作。如果校验和不匹配，接收器应该以“RESEND<CR><LF>”作为响应。在响应的同时，发送器还应当重新发送字节。

UU 编码的描述见 wotsit 主页。

29.4.2.4 ISP流控制

软件 XON/XOFF 流程控制机制可防止因缓冲区溢出而导致数据丢失这种情况的发生。当数据快速到达时，发送 ASCII 控制字符 DC3(停止)使数据流停止。发送 ASCII 控制字符 DC1(启动)恢复数据流。主机也应支持相同的流控制机制。

29.4.2.5 ISP命令中止

我们可以通过发送ASCII控制字符“ESC”来中止命令。“ISP命令”一节中并没有将该特性作为一个命令进行详述。一旦接收到“escape”代码，ISP命令处理器就会等待一个新的命令。

29.4.2.6 ISP过程中的中断

复位后，位于 Flash boot 扇区内的 boot block 中断向量都有效。

29.4.2.7 IAP过程中的中断

在擦除/编程操作过程中不能访问片内 Flash 存储器。当用户应用程序开始执行时，用户 Flash 区中的中断向量变有效。在调用 Flash 擦除/写 IAP 之前，用户应该禁止中断或确保用户中断向量在 RAM 中有效且中断处理程序位于 RAM 中。IAP 代码不使用或禁止中断。

29.4.2.8 ISP命令处理器使用的RAM

ISP 命令使用片内地址 0x4000 0120 到 0x4000 01FF 范围内的 RAM。用户可以使用该区域，但是一旦复位内容就可能会丢失。Flash 编程命令使用片内 RAM 最顶端的 32 字节。堆栈位于“RAM 顶端－32”的单元处。堆栈的最大使用范围为 256 字节，并且是向下递增的。

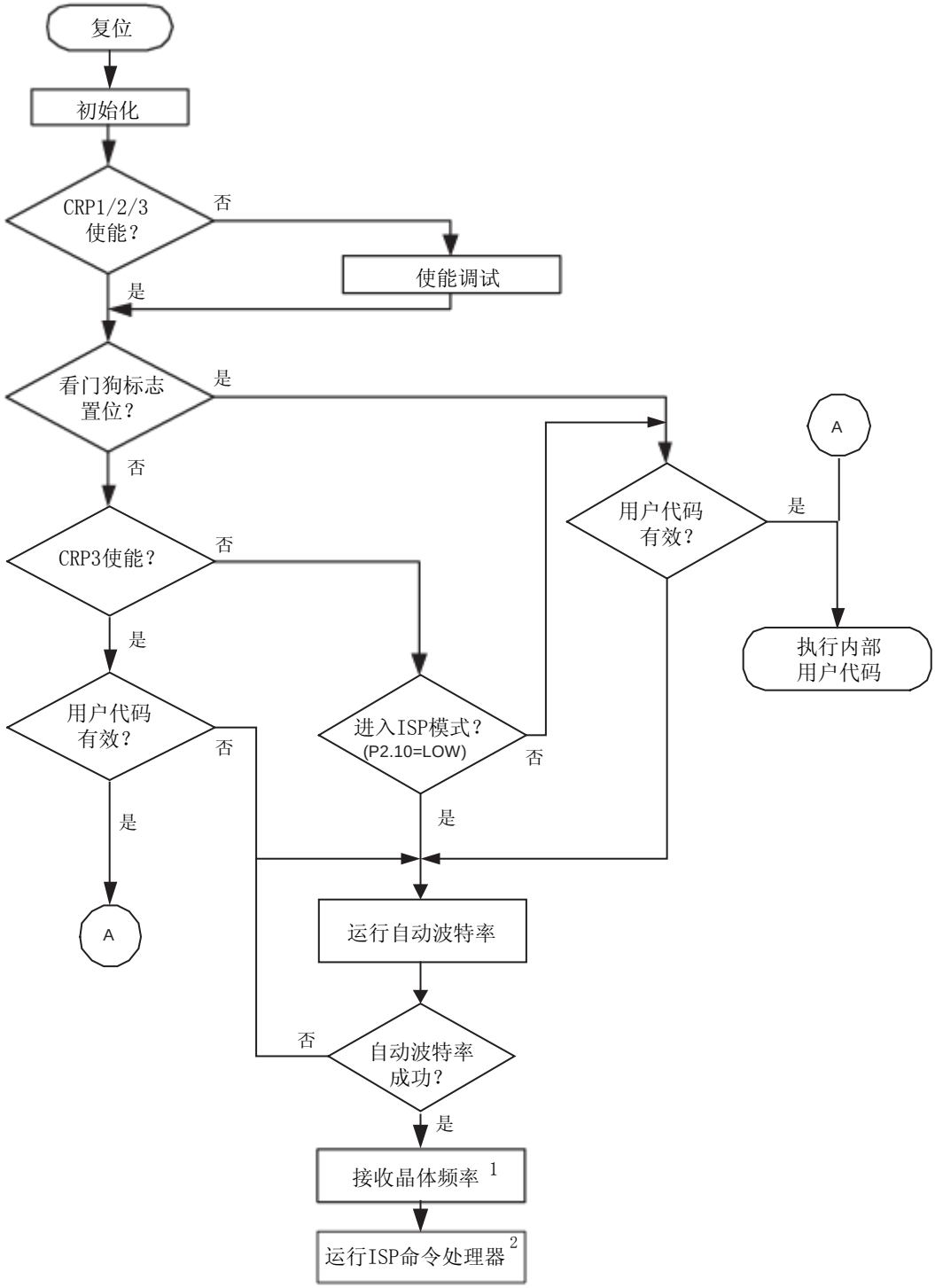
29.4.2.9 IAP命令处理器使用的RAM

Flash 编程命令使用片内 RAM 最顶端的 32 字节。堆栈的最大使用范围为 128 字节，并且是向下递增的。

29.4.2.10 RealMonitor使用的RAM

RealMonitor 使用的片内 RAM 地址范围为 0x4000 0040~0x4000 011F。如果不需要基于 RealMonitor 的调试，用户可使用该区域。Flash boot 装载程序不会初始化 RealMonitor 的堆栈。

29.5 BOOT处理流程图



（1） 关于处理晶体频率的详细内容，请见29.9.8 节“重新调用ISP”。

（2） 关于基于CRP设置的可用的ISP命令的详细内容，请见29.7 节“代码读保护（CRP）”。

图 29.2 Boot 处理流程图

29.6 扇区数

有些 IAP 和 ISP 命令根据“扇区”进行操作并指定扇区数。下表列出了 LPC2400 系列器件的扇区数和存储器地址之间的对应关系。IAP、ISP 和 RealMonitor 程序都位于 boot 区。boot 区位于所有器件中地址 0x0007 E000 到 0x0007 FFFF 处。ISP 和 IAP 命令不允许对 boot 扇区执行写/擦除/运行操作。由于 boot 区的原因，只有 504K 字节 Flash 可供用户程序和数据使用。

表 29.1 LPC2400 器件中的扇区

扇区号	扇区规格（KB）	地址范围
0	4	0x0000 0000 -0x0000 0FFF
1	4	0x0000 1000 -0x0000 1FFF
2	4	0x0000 2000 -0x0000 2FFF
3	4	0x0000 3000 -0x0000 3FFF
4	4	0x0000 4000 -0x0000 4FFF
5	4	0x0000 5000 -0x0000 5FFF
6	4	0x0000 6000 -0x0000 6FFF
7	4	0x0000 7000 -0x0000 7FFF
8	32	0x0000 8000 -0x0000 FFFF
9	32	0x0001 0000 -0x0001 7FFF
10（0x0A）	32	0x0001 8000 -0x0001 FFFF
11（0x0B）	32	0x0002 0000 -0x0002 7FFF
12（0x0C）	32	0x0002 8000 -0x0002 FFFF
13（0x0D）	32	0x0003 0000 -0x0003 7FFF
14（0x0E）	32	0x0003 8000 -0x0003 FFFF
15（0x0F）	32	0x0004 0000 -0x0004 7FFF
16（0x10）	32	0x0004 8000 -0x0004 FFFF
17（0x11）	32	0x0005 0000 -0x0005 7FFF
18（0x12）	32	0x0005 8000 -0x0005 FFFF
19（0x13）	32	0x0006 0000 -0x0006 7FFF
20（0x14）	32	0x0006 8000 -0x0006 FFFF
21（0x15）	32	0x0007 0000 -0x0007 7FFF
22（0x16）	4	0x0007 8000 -0x0007 8FFF
23（0x17）	4	0x0007 9000 -0x0007 9FFF
24（0x18）	4	0x0007 A000 -0x0007 AFFF
25（0x19）	4	0x0007 B000 -0x0007 BFFF
26（0x1A）	4	0x0007 C000 -0x0007 CFFF
27（0x1B）	4	0x0007 D000 – 0x0007 DFFF

29.7 代码读保护（CRP）

代码读保护是一种机制，它允许用户使能系统中的不同安全级别以便访问片内 Flash 和限制 ISP 的使用。如果有需要，通过在 Flash 地址单元 0x0000 01FC 编程特定的格式来调用 CRP。IAP 命令不受代码读保护的影响。

启动 bootloader 版本 3.2，执行 CRP 的 3 个级别。较早的 bootloader 版本只能执行 CRP2。

要点：一旦器件经历完一个电源周期，任何 CRP 变化就变有效。

表 29.2 代码读保护选项

名称	在 0x000001FC 处编程的格式	描述
CRP1	0x12345678	禁止通过 JTAG 管脚访问芯片。该模式允许使用下列 ISP 命令和约束来进行部分 Flash 更新： ● 写 RAM 命令不能访问在 0x40000200 下面的 RAM ● 复制 RAM 到 Flash 命令不能写扇区 0 ● 仅当选择所有扇区用于擦除时，擦除命令才能擦除扇区 0 ● 禁止比较命令 当需要有 CRP 且进行 Flash 字段更新、而不能擦除所有扇区时，可使用该模式。由于比较命令在部分更新的情况下被禁止，因此第二个装载程序应执行校验和机制来验证 Flash 的完整性。
CRP2	0x87654321	禁止通过 JTAG 管脚访问芯片。禁止下列的 ISP 命令： ● 读存储器 ● 写 RAM ● 运行（Go） ● 复制 RAM 到 Flash ● 比较 当使能 CRP2 时，ISP 擦除命令仅允许擦除所有用户扇区。
CRP3	0x43218765	禁止通过 JTAG 管脚访问芯片。如果有效的用户代码在 Flash 扇区 0 中出现时，ISP 入口通过拉低 P2.10 被禁能。 该模式有效地禁止了 ISP 忽略使用 P2.10 管脚。由用户的应用程序决定使用 IAP 调用来提供所需的 Flash 更新机制，还是通过 UART0 重新调用 ISP 命令来使能 Flash 更新。 **注意：当调用该模式时，不再对器件执行更多的厂商测试。**

表 29.3 代码读保护硬件/软件相互作用

CRP 选项	用户代码有效	复位时的 P2.10 管脚	JTAG 使能	LPC2400 进入 ISP 模式	在 ISP 模式中的部分 Flash 更新
No	No	X	Yes	Yes	Yes
No	Yes	高	Yes	No	NA
No	Yes	低	Yes	Yes	Yes
CRP1	Yes	高	No	No	NA
CRP1	Yes	低	No	Yes	Yes
CRP2	Yes	高	No	Yes	NA
CRP2	Yes	低	No	Yes	No
CRP3	Yes	高	No	No	NA
CRP3	Yes	低	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

如果使能 CRP 模式并允许通过 ISP 访问芯片，那么不支持或受到限定的 ISP 命令将通过返回代码 CODE_READ_PROTECTION_ENABLED 被终止。

29.8 ISP命令

下面的命令是 ISP 命令处理器所接受的命令。每个命令都支持具体的状态代码。当接收到未定义命令时，命令处理程序会发送返回代码 INVALID_COMMAND。命令和返回代码均为 ASCII 格式。

只有当接收到的 ISP 命令执行完毕后，ISP 命令处理器才会发送 CMD_SUCCESS，这时主机才能发送新的 ISP 命令。但“设置波特率”、“写 RAM”、“读存储器”和“运行”命令除外。

表 29.4 ISP 命令汇总

ISP 命令	用法	描述
解锁	U <解锁代码>	见表 29.5
设置波特率	B <波特率> <停止位>	见表 29.6
回声	A <设定>	见表 29.8
写 RAM	W <起始地址> <字节数>	见表 29.9
读存储器	R <地址> <字节数>	见表 29.10
准备写操作的扇区	P <起始扇区号> <结束扇区号>	见表 29.11
将 RAM 内容复制到 Flash	C <Flash 地址> <RAM 地址> <字节数>	见表 29.12
运行	G <地址> <模式>	见表 29.13
擦除扇区	E <起始扇区号> <结束扇区号>	见表 29.14
扇区查空	I <起始扇区号> <结束扇区号>	见表 29.15
读器件 ID	J	见表 29.16
读 Boot 代码版本	K	见表 29.18
比较	M <地址 1> <地址 2> <字节数>	见表 29.19

29.8.1 解锁 <解锁代码>

表 29.5 ISP 解锁命令

命令	U
输入	解锁代码：23130_{10}
返回代码	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
描述	该命令用于解锁 Flash 写、擦除和运行命令。
举例	“U 23130<CR><LF>”解锁 Flash 写/擦除&运行命令。

29.8.2 设置波特率<波特率><停止位>

表 29.6 ISP 设置波特率命令

命令	B
输入	波特率: 9600 | 19200 | 38400 | 57600 | 115200 | 230400 停止位: 1 | 2
返回代码	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
描述	该命令用于改变波特率。新的波特率在命令处理程序发送 CMD_SUCCESS 返回代码之后生效。
举例	“B 57600 1<CR><LF>”设置串口波特率 57600bps 和 1 个停止位。

表 29.7 ISP 波特率和 CCLK 频率的关系（单位:MHz）

ISP 波特率和 CCLK 频率	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456[1]	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

[1] ISP 入口复位后使用片内 IRC 和 PLL 在 CCLK=14.748MHz 的频率下运行器件。

29.8.3 回声<设定>

表 29.8 ISP 回声命令

命令	A
输入	设定：打开=1 | 关闭=0
返回代码	CMD_SUCCESS | PARAM_ERROR
描述	回声命令的默认设定是打开。当打开时，ISP 命令处理器将接收到的串行数据发送回主机。
举例	"A 0<CR><LF>" 回声关闭。

29.8.4 写RAM<起始地址><字节数>

主机应当仅在接收到 CMD_SUCCESS 返回代码后才发送数据。主机应当在发送 20 个 UU 编码行之后发送校验和。校验和是通过增加原始数据（UU 编码前）字节产生，并在发送完 20 个 UU 编码行后重新进行设置。任何 UU 编码行的长度不应超过 61 个字符（字节），即它可以存放 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和应该按照实际发送的字节数进行计算。ISP 命令处理器将它与接收字节的校验和进行比较。如果校验和匹配，那么 ISP 命令处理器将以“OK<CR><LF>”作为响应，以继续发送操作。如果校验和不匹配，接收器将以“RESEND<CR><LF>”作为响应。在响应的同时发送器还应该重新发送字节。

表 29.9 ISP 写 RAM 命令

命令	W
输入	**起始地址**：数据字节执行写操作处的 RAM 地址，该地址应当以字为边界。 **字节数**：写入的字节数。计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS | ADDR_ERROR (地址不是以字为边界) | ADDR_NOT_MAPPED | COUNT_ERROR (字节计数值不是 4 的倍数) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	该命令用于将数据下载到 RAM。数据应当为 UU 编码格式。该命令在代码读保护使能时停止使用。
举例	"W 1073742336 4<CR><LF>"向地址 0x4000 0200 写入 4 个字节数据。

29.8.5 读存储器<地址><字节数>

数据流之后紧跟着的是命令成功返回代码。校验和在发送完 20 个 UU 编码行之后发送。校验和通过增加原始数据（UU 编码前）字节产生，并在发送完 20 个编码行后重新进行设置。任何 UU 编码行的长度都不应超过 61 个字符（字节），即它可以存放 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和应该按照实际发送的字节数进行计算。主机将它与接收字节的校验和进行比较。如果校验和匹配，那么主机将以“OK<CR><LF>”作为响应，继续发送操作。如果校验和不匹配，主机将以“RESEND<CR><LF>”作为响应。在响应的同时，ISP 命令处理器应该重新发送数据。

表 29.10 ISP 读存储器命令

命令	R
输入	**起始地址**：读取数据字节处的地址，该地址应当以字为边界。 **字节数**：读取的字节数。计数值应该为 4 的倍数。
返回代码	CMD_SUCCESS，后面是<实际数据（UU 编码）>| ADDR_ERROR (地址不是以字为边界) | ADDR_NOT_MAPPED | COUNT_ERROR (字节计数值不是 4 的倍数) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	该命令用于读出 RAM 或 Flash 存储器的数据。该命令在代码读保护使能时停止使用。
举例	"R 1073741824 4<CR><LF>"从地址 0x4000 0000 读出 4 个字节数据。

29.8.6 准备写操作的扇区<起始扇区号> <结束扇区号>

该命令使 Flash 写/擦除操作分成两个步骤处理。

表 29.11 ISP 准备写操作的扇区命令

命令	P
输入	**起始扇区号** **结束扇区号**：应当大于或等于起始扇区号。
返回代码	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
描述	该命令必须在执行“将 RAM 内容复制到 Flash”或“擦除扇区”命令之前执行。这两个命令的成功执行会导致相关的扇区再次被保护。该命令不能用于 boot 区。要准备单个扇区，可将起始和结束扇区号设置为相同值。
举例	"P 0 0<CR><LF>"准备 Flash 扇区 0。

29.8.7 将RAM内容复制到Flash <Flash地址> <RAM地址> <字节数>

表 29.12 ISP 复制命令

命令	C
输入	**Flash 地址(DST)**：要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节。 **RAM 地址(SRC)**：读取数据字节的源 RAM 地址。 **字节数**：写入字节的数目。应当为 256 | 512 | 1024 | 4096

续表 29.12

命令	C
返回代码	CMD_SUCCESS | SRC_ADDR_ERROR (地址不以字为边界) | DST_ADDR_ERROR (地址边界错误) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (字节计数值不是 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	该命令用于编程 Flash 存储器。“准备写操作的扇区”应该先于该命令出现。受影响的扇区应当先通过调用“准备写操作的扇区”命令准备。当成功执行复制命令后，受影响的扇区将自动再次受到保护。该命令不能写 boot 扇区。该命令在代码读保护使能时停止使用。
举例	"C 0 1073774592 512<CR><LF>"将 RAM 地址 0x4000 8000 开始的 512 字节复制到 Flash 地址 0。

29.8.8 运行<地址><模式>

表 29.13 ISP 运行命令

命令	G
输入	**地址：**代码执行起始的 Flash 或 RAM 地址。该地址应当以字为边界。 **模式：**T（执行 Thumb 模式下的程序）|A（执行 ARM 模式下的程序）。
返回代码	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	该命令用于执行位于 RAM 或 Flash 存储器当中的程序。一旦成功执行该命令，就有可能不再返回 ISP 命令处理程序。该命令在代码读保护使能时停止使用。
举例	“G 0 A<CR><LF>”跳转到 ARM 模式下的地址 0x0000 0000 处。

29.8.9 擦除扇区<起始扇区号><结束扇区号>

表 29.14 ISP 擦除扇区命令

命令	E
输入	**起始扇区号** **结束扇区号：**应当大于等于起始扇区号。

续表 29.14

命令	E
返回代码	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。boot 区不能由该命令擦除。当代码读保护使能时，该命令只允许擦除所有用户扇区。
举例	"E 2 3<CR><LF>"擦除 Flash 扇区 2 和 3。

29.8.10 扇区查空<起始扇区号><结束扇区号>

表 29.15 ISP 扇区查空命令

命令	I
输入	**起始扇区号：** **结束扇区号：**应当大于或等于起始扇区号。
返回代码	CMD_SUCCESS | SECTOR_NOT_BLANK (后跟<第一个非空字的偏移量> <非空字的内容>) | INVALID_SECTOR | PARAM_ERROR |
描述	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。 **在第一个 64 字节重新映射到 Flash 的 boot 区时，对扇区 0 进行查空总是失败。**
举例	"I 2 3<CR><LF>"对 Flash 扇区 2 和 3 进行查空。

29.8.11 读器件ID号

表 29.16 ISP 读器件 ID 命令

命令	J
输入	无
返回代码	CMD_SUCCESS后跟ASCII格式的器件ID号（见表 29.17 “LPC2400 器件ID号”）。
描述	该命令用于读取器件的 ID 号。

表 29.17 LPC2400 器件 ID 号

器件	ASCII/dec 编码	Hex 编码
2468	100925237	0x0603 FF35

29.8.12 读Boot代码版本号

表 29.18 ISP 读取 Boot 代码版本号命令

命令	K
输入	无。
返回代码	CMD_SUCCESS 后跟 2 字节 ASCII 格式的 boot 代码版本号。将其解释为<字节 1 (主)>.<字节 0 (次)>。
描述	该命令用于读取 boot 代码版本号。

29.8.13 比较<地址 1><地址 2><字节数>

表 29.19 ISP 比较命令

命令	M
输入	**地址 1(DST)：**要比较的数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **地址 2(SRC)：**要比较的数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **字节数：**待比较的字节数；计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS | (源和目标数据相同) COMPARE_ERROR | (后跟第一个不匹配字节的地址) COUNT_ERROR (字节数不是 4 的倍数) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR |
描述	该命令用来比较两个地址单元的存储器内容。**如果源地址和目标地址包含从地址 0 开始的第一个 64 字节的任意字节时，比较结果可能不正确。第一个 64 字节重新映射到 Flash 的 boot 区。**
举例	"M 8192 1073741824 4<CR><LF>"将 RAM 地址 0x4000 0000 开始的 4 个字节与 Flash 地址 0x2000 开始的 4 个字节进行比较。

29.8.14 ISP返回代码

表 29.20 ISP 返回代码汇总

返回代码	符号	描述
0	CMD_SUCCESS	命令被成功执行。只有当主机发出的命令被成功执行完毕后，才由 ISP 处理器发送。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址没有以字为边界。
3	DST_ADDR_ERROR	目标地址的边界错误。
4	SRC_ADDR_NOT_MAPPED	源地址没有在存储器映射中被映射。计数值必须考虑可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有在存储器映射中被映射。计数值必须考虑到可用性。
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值。
7	INVALID_SECTOR	扇区号无效或结束扇区号小于起始扇区号。
8	SECTOR_NOT_BLANK	扇区非空。

续表 29.20

返回代码	符号	描述
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区的命令未执行。
10	COMPARE_ERROR	源和目标数据不相等。
11	BUSY	Flash 编程硬件接口忙。
12	PARAM_ERROR	参数不足或无效参数。
13	ADDR_ERROR	地址没有以字为边界。
14	ADDR_NOT_MAPPED	地址没有在存储器映射中被映射。计数值必须考虑可用性。
15	CMD_LOCKED	命令被锁定。
16	INVALID_CODE	解锁代码无效。
17	INVALID_BAUD_RATE	无效波特率设定。
18	INVALID_STOP_BIT	无效停止位设定。
19	CODE_READ_PROTECTION_ENABLED	内核读保护使能。

29.9 IAP命令

对于在应用编程来说，IAP程序应当通过寄存器r0 中的字指针指向包含命令代码和参数的存储器(RAM)来调用IAP程序。IAP命令的结果返回到寄存器r1 所指向的结果表（result table）。用户可通过传递寄存器r0 和r1 中的相同指针来重新使用命令表以获取结果。参数表应当大到足够存放所有的结果以防结果的数目大于参数的数目。参数传递的过程如图29.3所示。参数和结果的数目根据IAP命令而有所不同。参数的最大数目为 5，被传递给“将RAM内容复制到Flash”命令。结果的最大数目为 2，由“扇区查空”命令返回。命令处理程序在接收到一个未定义的命令时发送状态代码INVALID_COMMAND。IAP程序是thumb代码，位于地址 0x7FFF FFF0 处。

IAP 功能可用下面的 C 代码来调用。

定义 IAP 程序的入口地址。由于 IAP 地址的第 0 位是 1，因此，当程序计数器转移到该地址时会引起 Thumb 指令集的变化。

```
#define    IAP_LOCATION 0x7ffffff1
```

定义数据结构或指针，将 IAP 命令表和结果表传递给 IAP 函数：

```
unsigned    long     command[5];
unsigned    long     result[2];
```

或者：

```
unsigned    long     * command;
unsigned    long     * result;
command = (unsigned    long    *)    0x….
result = (unsigned    long    *)    0x….
```

定义函数类型指针，函数包含 2 个参数，无返回值。注意：IAP 将函数结果和 R1 中的表格基址一同返回。

```
typedef  void (*IAP)  (unsigned  int [ ] , unsigned  int [ ]);
IAP  iap_entry;
```

设置函数指针：

```
iap_entry=(IAP)  IAP_LOCATION;
```

使用下面的语句来调用 IAP。

```
iap_entry  (command , result);
```

使用 ADS（ARM 开发套件）中 ARM 连接器所支持的符号定义文件（symbol definition file）可以进一步简化 IAP 的调用过程。此外，用户还可使用汇编程序来调用 IAP 程序。

下面的符号定义可用于连接 IAP 程序和用户应用程序。

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

根据 ARM 规范（ARM Thumb 过程调用标准 SWS ESPC 0002 A-05），r0, r1, r2 和 r3 寄存器能够传递多达 4 个参数。其它参数通过堆栈传递。多达 4 个参数可以返回 r0, r1, r2 和 r3 寄存器。其它参数通过存储器间接返回。某些 IAP 调用需要的参数多于 4 个。如果使用 ARM 建议的机制来传递/返回参数，则有可能因为不同厂商所提供的 C 编译器不同而产生问题。使用建议的参数传递机制便可以降低这种风险。

在写或擦除操作过程中不能访问 Flash 存储器。执行 Flash 写/擦除操作的 IAP 命令使用片内 RAM 顶端的 32 个字节空间。如果应用程序中允许 IAP Flash 编程，那么用户程序不应使用该空间。

表 29.21 IAP 命令汇总

ISP 命令	命令代码	描述
准备写操作扇区	50_{10}	见表 29.22
将 RAM 内容复制到 Flash	51_{10}	见表 29.23
擦除扇区	52_{10}	见表 29.24
扇区查空	53_{10}	见表 29.25
读器件 ID	54_{10}	见表 29.26
读 boot 代码版本	55_{10}	见表 29.27
比较	56_{10}	见表 29.28
重新调用 ISP	57_{10}	见表 29.29

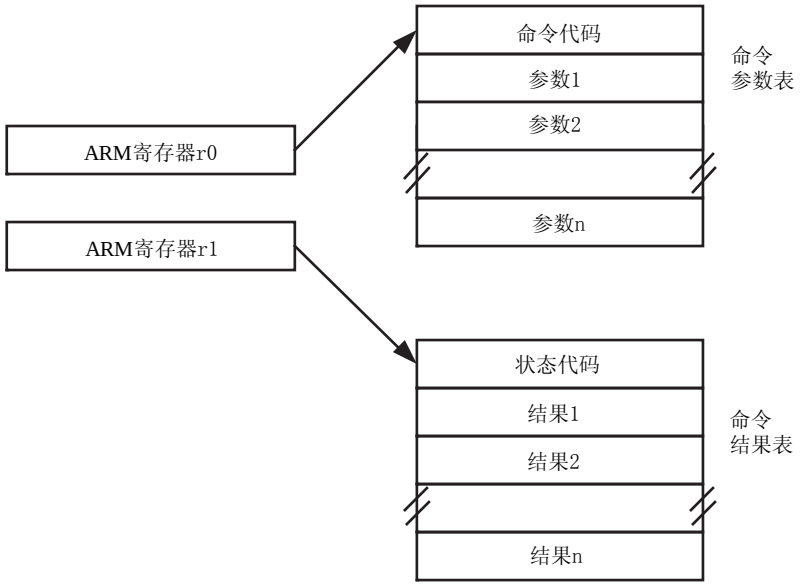


图 29.3 IAP 参数传递

29.9.1 准备写操作扇区

该命令使 Flash 写/擦除操作分两步执行。

表 29.22 IAP 准备写操作扇区命令

命令	准备写操作扇区
输入	**命令代码：50_{10}** **参数 0**：起始扇区号 **参数 1**：结束扇区号（应当大于或等于起始扇区号）。
返回代码	CMD_SUCCESS | BUSY | INVALID_SECTOR
结果	无
描述	该命令必须在执行“将 RAM 内容复制到 Flash”或“擦除扇区”命令之前执行。这两个命令的成功执行会导致相关的扇区再次被保护。该命令不能用于 boot 扇区。要准备单个扇区，可将起始和结束扇区号设置为相同值。

29.9.2 将RAM内容复制到Flash

表 29.23 IAP 将 RAM 内容复制到 Flash 命令

命令	将 RAM 内容复制到 Flash
输入	**命令代码：51_{10}** **参数 0 (DST)**：要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节。 **参数 1 (SRC)**：读取数据字节的源 RAM 地址。该地址应当以字为边界。 **参数 2**：写入字节的数目。应当为 256 | 512 | 1024 | 4096。 **参数 3**：系统时钟频率（CCLK）(单位：kHz)

续表 29.23

命令	将 RAM 内容复制到 Flash
返回代码	CMD_SUCCESS | SRC_ADDR_ERROR (地址不以字为边界) | DST_ADDR_ERROR (地址边界错误) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (字节计数值不是 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY |
结果	无
描述	该命令用于编程 Flash 存储器。受影响的扇区应先通过调用“准备写操作扇区”命令来准备好。当成功执行复制命令后，扇区将自动受到保护。该命令不能写 boot 扇区。

29.9.3 擦除扇区

表 29.24 IAP 擦除扇区命令

命令	擦除扇区
输入	**命令代码：52_{10}** **参数 0：**起始扇区号 **参数 1：**结束扇区号（应当大于或等于起始扇区号）。 **参数 2：**系统时钟频率（CCLK）(单位：kHz)
返回代码	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
结果	无
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。boot 扇区不能由该命令擦除。要擦除单个扇区可将“起始”和“结束”扇区号设定为相同值。

29.9.4 扇区查空

表 29.25 IAP 扇区查空命令

命令	扇区查空
输入	**命令代码：53_{10}** **参数 0：**起始扇区号 **参数 1：**结束扇区号（应当大于或等于起始扇区号）。
返回代码	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR |
结果	**结果 0：**返回代码为 SECTOR_NOT_BLANK 时第一个非空的字单元的偏移量 **结果 1：**非空的字单元的内容
描述	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。要查空单个扇区可将“起始”和“结束”扇区号设定为相同值。

29.9.5 读器件ID号

表 29.26 IAP 读器件 ID 命令

命令	读器件 ID 号
输入	**命令代码：54_{10}** **参数**：无
返回代码	CMD_SUCCESS |
结果	**结果 0**：器件 ID 号。
描述	该命令用于读取器件的 ID 号。

29.9.6 读取Boot代码版本号

表 29.27 IAP 读取 Boot 代码版本号命令

命令	读取 Boot 代码版本
输入	**命令代码：55_{10}** **参数**：无
返回代码	CMD_SUCCESS |
结果	**结果 0**：2 字节 ASCII 格式的 boot 代码版本号。将其解释为<字节 1 (主)>.<字节 0 (次)>。
描述	该命令用于读取 boot 代码版本号。

29.9.7 比较<地址 1> <地址 2><字节数>

表 29.28 IAP 比较命令

命令	比较
输入	**命令代码：56_{10}** **参数 0 (DST)**：要被比较的数据字节的 Flash 或 RAM 起始地址。该地址应当以字为边界。 **参数 1 (SRC)**：要被比较的数据字节的 Flash 或 RAM 起始地址。该地址应当以字为边界。 **参数 2**：待比较的字节数。计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (字节数不是 4 的倍数) | ADDR_ERROR | ADDR_NOT_MAPPED
结果	**结果 0**：当状态代码为 COMPARE_ERROR 时第一个不匹配字节的偏移地址
描述	该命令用来比较两个地址单元的存储器内容。**当源或目标地址包含从地址 0 开始的前 64 字节中的任意一个时，比较的结果不一定正确。前 64 字节重新映射到 RAM。**

29.9.8 重新调用ISP

表 29.29 重新调用 ISP

命令	比较
输入	命令代码：57_{10}
返回代码	无
结果	无
描述	该命令用来调用ISP模式中的引导装载程序。它映射boot向量，设置PCLK=CCLK/4，配置UART0 管脚Rx和Tx，重新设置TIMER1 并复位U0FDR（见16.3.12 节）。当一个有效的用户程序在内部Flash存储器出现且不可访问P2.10 管脚时，可使用该命令来强制ISP模式。该命令不禁止PLL，因此当器件在PLL以外运行时该命令可能调用引导装载程序。在这种情况下，ISP程序应在自动波特率握手后传送CCLK（晶体或PLL输出取决于时钟源选择，见4.4.1 节）频率。 另一个选项是禁止 PLL 并在调用 IAP 之前选择 IRC 作为时钟源。在这种情况下，由 ISP 发送的频率被忽略，且 IRC 和 PLL 被用来产生 CCLK=14.748MHz 的频率。

29.9.9 IAP状态代码

表 29.30 IAP 状态代码汇总

返回代码	符号	描述
0	CMD_SUCCESS	命令被成功执行。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址不是以字为边界。
3	DST_ADDR_ERROR	目标地址的边界错误。
4	SRC_ADDR_NOT_MAPPED	源地址没有映射到存储器映射中。计数值必须考虑其可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有映射到存储器映射中。计数值必须考虑其可用性。
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值。
7	INVALID_SECTOR	扇区号无效。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	“准备写操作扇区”命令未执行。
10	COMPARE_ERROR	源和目标数据不相同。
11	BUSY	Flash 编程硬件接口忙

29.10 JTAG Flash编程接口

调试工具可以将部分Flash映像写入RAM，然后根据正确的偏移重复执行IAP命令“将RAM内容复制到Flash”。

第30章 通用DMA控制器（GPDMA）

30.1 简介

通用 DMA 控制 器（GPDMA）是一个遵循 AMBA AHB 的外设，它允许选择的 LPC2400 外设支持 DMA。

30.2 GPDMA的特性

- 2 个 DMA 通道。每个通道可支持一个单向传输；
- GPDMA 提供 16 根外设 DMA 请求线。某些请求线连接到支持 DMA 的外设：SD/MMC、2 个 SSP 和 I2S 接口；
- 单次 DMA 和突发 DMA 请求信号。每个连接到 GPDMA 的外设可以提交一个突发 DMA 请求或一个单次 DMA 请求。DMA 突发大小（burst size）通过编程 GPDMA 来设置；
- 存储器到存储器、存储器到外设、外设到存储器和外设到外设的传输；
- 通过使用链表来支持分散或聚集的 DMA。这就意味者源区和目标区不一定要占用连续的存储区；
- 硬件 DMA 通道优先级。每个 DMA 通道含有一个特定的硬件优先级。DMA 通道 0 的优先级最高，通道 1 的优先级最低。如果 2 个通道的请求同时有效，则先服务优先级最高的通道；
- AHB 从机 DMA 编程接口。通过在 AHB 从机接口上写 DMA 控制寄存器来编程 GPDMA；
- 一个传输数据的 AHB 总线主机。这个接口在 DMA 请求有效时传输数据；
- 32 位的 AHB 主机总线宽度；
- 源或目标区的递增寻址或非递增寻址；
- DMA 突发大小可编程。编程 DMA 突发大小可以提高传输数据的效率。通常，突发大小被设置成外设 FIFO 大小的一半；
- 每个通道包含一个内部 4 个字的 FIFO；
- 支持 8、16 和 32 位宽的传输；
- 支持大端和小端模式。复位后 GPDMA 默认为小端模式；
- DMA 操作完成或出现 DMA 错误时可以产生一个处理器中断；
- 中断屏蔽。可以屏蔽 DMA 错误中断请求和 DMA 终端计数中断请求；
- 原始中断状态。DMA 错误和 DMA 计数的原始中断状态可以在屏蔽之前被读出；
- 测试寄存器可以用在模块和集成系统级别的测试中；
- 专门识别 GPDMA 的标识寄存器。操作系统可以使用这些寄存器来对自身进行配置。

30.3 功能概述

本章描述了 GPDMA 的主要功能模块。包含下面几节：

- GPDMA 功能描述
- 系统注意事项
- 系统连接
- 和基于存储器管理单元的系统共同使用

30.3.1 GPDMA功能概述

GPDMA 使能外设到存储器、存储器到外设、外设到外设和存储器到存储器的传输。每个 DMA 流都可以为单个源和目标提供单向串行 DMA 传输。例如，一个双向端口就需要一个发送流，一个接收流。源和目标区可以是存储区或外设，可以通过 AHB 主机进行访问。

图 30.1所示为GPDMA的方框图。

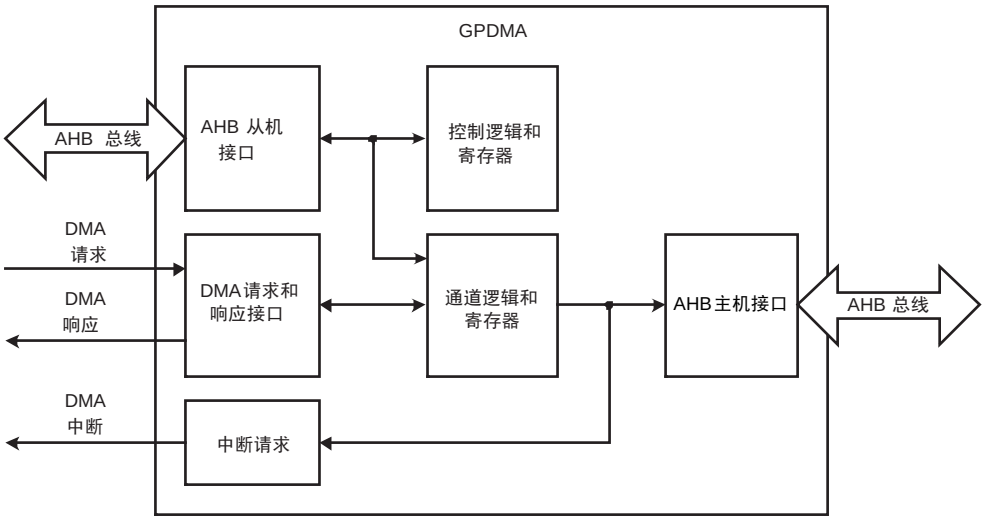


图 30.1 GPDMA 方框图

GPDMA 的功能在下面几节中描述：

- AHB 从机接口
- 控制逻辑和寄存器组
- DMA 请求和响应接口
- 通道逻辑和通道寄存器组
- 中断请求
- AHB 主机接口
- 通道硬件
- DMA 请求优先级

30.3.1.1 AHB从机接口

GPDMA AHB 从机编程总线上的所有传输都是 32 位宽。

30.3.1.2 控制逻辑和寄存器组

寄存器块存放着写入的或通过 AHB 接口读出的数据。

30.3.1.3 DMA请求和响应接口

有关 DMA 请求和响应接口的信息请见 DMA 接口的描述。

30.3.1.4 通道逻辑和通道寄存器组

通道逻辑和通道寄存器组包括每个 DMA 通道所需的寄存器和逻辑。

30.3.1.5 中断请求

中断请求向 ARM 处理器产生中断。

30.3.1.6 AHB主机接口

GPDMA包含一个完整的AHB主机，图 30.2的例子显示了GPDMA与一个系统的连接。

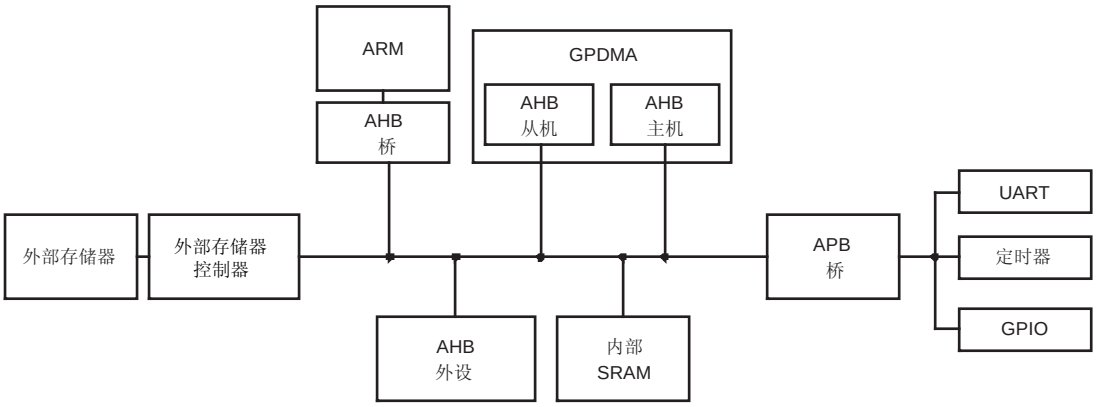


图 30.2 一个系统中的 GPDMA 的例子

AHB 主机能够处理各种类型的 AHB 传输，包括：

- 分离、重试和从机的错误响应。如果一个外设执行一次分离或重试传输，那么 GPDMA 停止并等待直到传输结束。
- 锁定每个流的源和目的传输。
- 为每个流的传输设置保护位。

30.3.1.7 总线和传输宽度

AHB 总线的物理宽度是 32 位。源和目标传输的宽度可以不同，可以相同，也可以比物理总线宽度更窄。GPDMA 根据需要打包（pack）或拆分（unpack）数据。

30.3.1.8 字节顺序特性

GPDMA 可以处理小端和大端寻址。你可以独立设置每个 AHB 主机的字节顺序（endian）。

在内部，GPDMA 将所有的数据当作一个字节流来处理，而不是 16 位或 32 位的数据量。这就意味着当执行源和目的传输字节顺序不同的混合字节顺序传输活动时可以观察到 32 位数据内的字节交换。

注意：如果不需要字节交换，那么请避免在源和目标地址之间使用不同的字节顺序。

表 30.1所示为不同的源和目标组合的字节顺序特性。

表 30.1 字节顺序特性

源 字节顺序	目标 字节顺序	源宽度	目标宽度	源传输编号 /字节通道	源数据	目标传输编 号/字节通道	目标数据
小端	小端	8	8	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21 43 65 87	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21212121 43434343 65656565 87878787
小端	小端	8	16	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21 43 65 87	1/[15:0] 2/[31:16]	43214321 87658765
小端	小端	8	32	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21 43 65 87	1/[31:0]	87654321
小端	小端	16	8	1/[7:0] 1/[15:8] 2/[23:16] 2/[31:24]	21 43 65 87	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21212121 43434343 65656565 87878787
小端	小端	16	16	1/[7:0] 1/[15:8] 2/[23:16] 2/[31:24]	21 43 65 87	1/[15:0] 2/[31:16]	43214321 87658765
小端	小端	16	32	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21 43 65 87	1/[31:0]	87654321
小端	小端	32	8	1/[7:0] 1/[15:8] 2/[23:16] 2/[31:24]	21 43 65 87	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21212121 43434343 65656565 87878787
小端	小端	32	16	1/[7:0] 1/[15:8] 2/[23:16] 2/[31:24]	21 43 65 87	1/[15:0] 2/[31:16]	43214321 87658765

续表 30.1

源 字节顺序	目标 字节顺序	源宽度	目标宽度	源传输编号 /字节通道	源数据	目标传输编 号/字节通道	目标数据
小端	小端	32	32	1/[7:0] 2/[15:8] 3/[23:16] 4/[31:24]	21 43 65 87	1/[31:0]	87654321
大端	大端	8	8	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12121212 34343434 56565656 78787878
大端	大端	8	16	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[15:0] 2/[31:16]	12341234 56785678
大端	大端	8	32	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:0]	12345678
大端	大端	16	8	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12121212 34343434 56565656 78787878
大端	大端	16	16	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[15:0] 2/[31:16]	12341234 56785678
大端	大端	16	32	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:0]	12345678
大端	大端	32	8	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12121212 34343434 56565656 78787878
大端	大端	32	16	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[15:0] 2/[31:16]	12341234 56785678
大端	大端	32	32	1/[31:24] 2/[23:16] 3/[15:8] 4/[7:0]	12 34 56 78	1/[31:0]	12345678

30.3.1.9 错误条件

DMA 传输过程中的错误在传输过程中由外设在 AHB 总线上声明一个错误响应直接标记出来。在当前的传输结束后，GPDMA 自动禁能 DMA 流，可以选择向 CPU 产生一个错误中断。这个错误中断可以被屏蔽。

30.3.1.10 通道硬件

每个流都有专门的硬件通道（包括源和目标控制器、FIFO）支持。这就比只带有一个硬件通道（该通道由几个 DMA 流共用）的 DMA 控制器获得更好的等待时间，而且简化了控制逻辑。

30.3.1.11 DMA请求优先级

DMA 通道的优先级是固定的。DMA 通道 0 的优先级最高，DMA 通道 1 的优先级最低。

如果 GPDMA 正在传输低优先级通道的数据，而高优先级的通道又变得有效，则它将先完成低优先级通道授权给主机接口的所有传输，然后再转去传输高优先级通道的数据。最差情况下这是一个 4 字长的过程。GPDMA 的通道 1 这样设计以便它不能长期占用 AHB 总线。如果通道 1 变为有效，在 4 个可编程大小的传输（与传输的大小无关）之后，GPDMA 放弃总线控制（大约一个总线周期）。这就允许其他 AHB 主机访问总线。

建议存储器到存储器的传输使用低优先级的通道。否则会阻止其他（低优先级的）AHB 总线主机在 GPDMA 存储器到存储器传输的过程中访问总线。

30.3.1.12 中断产生

一个组合中断输出是单个 GPDMA 中断请求相或的结果，中断输出连接到 LPC2400 中断控制器。

30.3.1.13 DMA传输结束指示

用以下方式来指示 DMA 传输的结束：

1. 如果 GPDMA 正在执行流控制，则传输计数到达 0 时表明传输结束，或者
2. 如果外设正在执行流控制，外设设置 DMA 最后一个字请求输入（DMACLSREQ）或 DMA 最后一个突发请求输入（DMALBREQ）来指示传输结束。

从表 30.2“DMA连接”中看出，SSP0、SSP1 和I2S既不使用DMA最后一个字请求输入，也不使用DMA最后一个突发请求输入。因此，如果SSP0、SSP1 和I2S正在执行流控制，就没有DMA传输结束指示。

30.3.2 DMA系统连接

GPDMA和支持它的外设器件的连接取决于在这些外设中实现的DMA功能。表 30.2所示为支持DMA的外设所使用的DMA请求编号。

表 30.2 DMA 连接

外设功能	DMA 单次请求输入	DMA 突发请求输入	DMA 最后一个字请求输入	DMA 最后一个突发请求输入
SSP0 Tx	0	0	-	-
SSP0 Rx	1	1	-	-
SSP1 Tx	2	2	-	-
SSP1 Rx	3	3	-	-
SD/MMC	4	4	4	4
I2S 通道 0	-	5	-	-
I2S 通道 1	-	6	-	-

30.4 编程模型

本章描述了 GPDMA 寄存器，并提供了编程微控制器所需的详细信息。本章包含以下几节：

- 编程模型的相关信息
- 编程 GPDMA
- GPDMA 寄存器小结
- 寄存器描述
- 地址的产生
- 分散/聚集
- 中断请求
- GPDMA 数据流

30.5 编程模型的相关信息

GPDMA 使能外设到存储器、存储器到外设、外设到外设和存储器到存储器的传输。配置每个 DMA 流来为单个源和目标提供单向 DMA 传输。源区和目标区可以是支持 GPDMA 的一个存储区或一个外设，它们必须可以通过 AHB1 来访问。

30.6 编程GPDMA

以下内容适用于 GPDMA 使用的寄存器：

- 保留或不使用的地址单元不允许访问，因为这样会导致器件不可预知的操作。
- 寄存器的保留位或不使用的位必须写入 0，读出时被忽略，在相关文本中特别说明的除外。
- 系统复位或上电复位后所有的寄存器都为逻辑 0，在相关文本中特别说明的除外。
- 除非在相关文本中有特别说明，否则所有的寄存器都支持读和写访问。写操作是对寄存器的内容进行更新，读操作会返回寄存器的内容。
- 本文档中定义的所有寄存器只能使用字读取和字写入来访问（即 32 位访问），在相关文本中特别说明的除外。

30.6.1 使能GPDMA

使GPDMA置位DMACConfiguration寄存器的DMA使能位（请参考30.8.13 节“配置寄存器（DMACConfiguration - 0xFFE0 4030）”）。

30.6.2 禁能GPDMA

执行以下操作来禁能 GPDMA：

- 读取DMACEnbldChns寄存器，确保所有的DMA通道已经被禁能。如果所有通道都有效，请参考30.6.4 节“禁能DMA通道”。
- 通过清零DMACConfiguration寄存器的DMA使能位来禁能GPDMA（请参考30.9.6 节“通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）”）。

30.6.3 使能DMA通道

使能DMA通道置位相应DMA通道配置寄存器的通道使能位（请参考30.9.6 节“通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）”）。

注意：通道在使能前必须被完整地初始化。另外，在通道使能之前，你还必须置位 GPDMA 的使能位。

30.6.4 禁能DMA通道

你可以用以下方法来禁能 DMA 通道：

- 直接写通道使能位。如果使用这个方法，FIFO 中所有未处理的数据都会丢失。
- 使 Active 位、Halt 位与通道使能（Channel Enable）位共同使用。
- 等到传输结束，然后通道被自动禁止。

30.6.5 禁能一个FIFO中没有损失数据的DMA通道

执行以下操作来禁能一个 FIFO 中没有损失数据的 DMA 通道：

1. 置位相应通道配置寄存器的Halt位（请参考30.9.6 节“通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）”）。这会造成后面的DMA请求被忽略。
2. 查询相应通道配置寄存器的 Active 位，直至它的值到达 0。这一位指示了通道中是否有要传输的数据。
3. 清除相应通道配置寄存器中的通道使能位。

30.6.6 设置一个新的DMA传输

执行以下操作来设置一个新的 DMA 传输：

1. 如果某个通道不能留出来供 DMA 传输使用，则：
   - 读取DMACEnabldChns寄存器，找出无效的通道（请参考30.8.8 节“使能通道寄存器（DMACEnbldChns - 0xFFE0 401C）”）。
   - 选择具有所需优先级的无效通道。
2. 编程 GPDMA。

30.6.7 禁能一个DMA通道，丢失FIFO中的数据

清除相应通道配置寄存器中的通道使能位（请参考30.9.6 节“通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）”）。如果当前有一个AHB传输正在执行，则传输完成后再禁能通道。FIFO中的所有数据全部丢失。

30.6.8 终止一个DMA传输

置位相应 DMA 通道配置寄存器中的 Halt 位。当前的源请求被服务。任何其他的源 DMA 请求被忽略，直到清零 Halt 位。

30.6.9 编程DMA通道

执行以下操作来编程一个 DMA 通道：

1. 选择一个具有所需优先级的空闲 DMA 通道。DMA 通道 0 的优先级最高，DMA 通道 1 的优先级最低。
2. 通过写DMACIntTCClr寄存器（请参考30.8.3 节“中断终端计数清除寄存器（DMACIntClear - 0xFFE0 4008）”）和DMACIntErrClr寄存器（请参考30.8.5 节“中断错误清除寄存器（DMACIntErrClr - 0xFFE0 4010）”）清除所使用通道上的挂起中断。先前的通道操作可能会使中断有效。
3. 向 DMACCxSrcAddr 寄存器（请参考 30.9.1 节“通道源地址寄存器（DMACC0SrcAddr - 0xFFE0 4100，DMACC1SrcAddr - 0xFFE0 4120）”）写入源地址。
4. 向 DMACCxDestAddr 寄存器（请参考 30.9.2 节“通道目标地址寄存器（DMACC0DestAddr - 0xFFE0 4104，DMACC1DestAddr - 0xFFE0 4124）”）写入目标地址。
5. 向DMACCxLLI寄存器（请参考30.9.3 节“通道链表项寄存器（DMACC0LLI - 0xFFE0 4108，DMACC1LLI - 0xFFE0 4128）”）写入下个链表项（LLI）的地址。如果传输只有一个数据包组成，则必须向这个寄存器写入 0。
6. 向DMACCxControl寄存器（请参考30.9.4 节“通道控制寄存器（DMACC0Control - 0xFFE0 410C，DMACC0Control - 0xFFE0 412C）”）写入控制信息。
7. 向 DMACCxConfiguration 寄存器（请参考 30.9.6 节“通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）”）写入通道配置信息。如果使能位被置位，则DMA通道自动使能。

30.7 GPDMA寄存器小结

GPDMA寄存器如表 30.3所示。

表 30.3 GPDMA 寄存器映射

名称	描述	访问	复位值	地址
通用寄存器				
DMACIntStatus	中断状态寄存器	RO	0x0	0xFFE0 4000
DMACIntTCStatus	中断终端计数状态寄存器	RO	0x0	0xFFE0 4004
DMACIntTCClear	中断终端计数清零寄存器	WO	-	0xFFE0 4008
DMACIntErrorStatus	中断错误状态寄存器	RO	0x0	0xFFE0 400C
DMACIntErrClr	中断错误清除寄存器	WO	-	0xFFE0 4010
DMACRawIntTCStatus	原始中断终端计数状态寄存器	RO	-	0xFFE0 4014
DMACRawIntErrorStatus	原始错误中断状态寄存器	RO	-	0xFFE0 4018
DMACEnbldChns	使能通道寄存器	RO	0x0	0xFFE0 401C
DMACSoftBReq	软件突发请求寄存器	R/W	0x0000	0xFFE0 4020
DMACSoftSReq	软件单次请求寄存器	R/W	0x0000	0xFFE0 4024
DMACSoftLBReq	软件最后一个突发请求寄存器	R/W	0x0000	0xFFE0 4028
DMACSoftLSReq	软件最后一个单次请求寄存器	R/W	0x0000	0xFFE0 402C
DMACConfiguration	配置寄存器	R/W	0x0000 0000	0xFFE0 4030
DMACSync	同步寄存器	R/W	0x0000	0xFFE0 4034
通道 0 寄存器				
DMACC0SrcAddr	通道 0 源地址寄存器	R/W	0x0000 0000	0xFFE0 4100
DMACC0DestAddr	通道 0 目标地址寄存器	R/W	0x0000 0000	0xFFE0 4104
DMACC0LLI	通道 0 链表项寄存器	R/W	0x0000 0000	0xFFE0 4108
DMACC0Control	通道 0 控制寄存器	R/W	0x0000 0000	0xFFE0 410C
DMACC0Configuration	通道 0 配置寄存器	R/W	0x00000[1]	0xFFE0 4110
通道 1 寄存器				
DMACC1SrcAddr	通道 1 源地址寄存器	R/W	0x0000 0000	0xFFE0 4120
DMACC1DestAddr	通道 1 目标地址寄存器	R/W	0x0000 0000	0xFFE0 4124
DMACC1LLI	通道 1 链表项寄存器	R/W	0x0000 0000	0xFFE0 4128
DMACC1Control	通道 1 控制寄存器	R/W	0x0000 0000	0xFFE0 412C
DMACC1Configuration	通道 1 配置寄存器	R/W	0x00000[1]	0xFFE0 4130

[1] 位 17 是只读位。

30.8 寄存器描述

本节描述了 GPDMA 的寄存器。

30.8.1 中断状态寄存器（DMACIntStatus - 0xFFE0 4000）

DMACIntStatus是一个只读寄存器，它显示了屏蔽后中断的状态。寄存器中为高的位表明某个特定的DMA通道中断请求有效。请求可以由错误产生，也可以是终端计数中断请

求。表 30.4所示为DMACIntStatus寄存器的位分配。

表 30.4 中断状态寄存器（DMACIntStatus – 地址 0xFFE0 4000）位描述

位	符号	描述	复位值
0	IntStatus0	屏蔽后通道 0 中断的状态。	0
1	IntStatus1	屏蔽后通道 1 中断的状态。	0
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.2 中断终端计数状态寄存器（DMACIntTCStatus - 0xFFE0 4004)

DMACIntTCStatus是一个只读寄存器，指示了屏蔽后终端计数的状态。表 30.5所示为DMACIntTCStatus寄存器的位分配。

表 30.5 中断终端计数状态寄存器（DMACIntTCStatus – 地址 0xFFE0 4004）位描述

位	符号	描述	复位值
0	IntTCStatus0	通道 0 终端计数中断请求的状态。	0
1	IntTCStatus1	通道 1 终端计数中断请求的状态。	0
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.3 中断终端计数清除寄存器（DMACIntClear - 0xFFE0 4008）

DMACIntTCClear寄存器只可写，它清除一个终端计数中断请求。当写这个寄存器时，每个设置为 1 的数据位会使状态寄存器中相应的位清零。数据位为 0 对状态寄存器中相应的位无影响。表 30.6所示为DMACIntTCClear寄存器的位分配。

表 30.6 中断终端计数清除寄存器（DMACIntClear – 地址 0xFFE0 4008）位描述

位	符号	描述	复位值
0	IntTCClear0	写入 1 来清除通道 0 的终端计数中断请求（IntTCStatus0）。	-
1	IntTCClear1	写入 1 来清除通道 1 的终端计数中断请求（IntTCStatus1）。	-
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.4 中断错误状态寄存器（DMACIntErrorStatus - 0xFFE0 400C）

DMACIntErrorStatus是一个只读寄存器，它指示了屏蔽后错误请求的状态。表 30.7所示为DMACIntErrorStatus寄存器的位分配。

表 30.7 中断错误状态寄存器（DMACIntErrorStatus – 地址 0xFFE0 400C）位描述

位	符号	描述	复位值
0	IntErrorStatus0	通道 0 的中断错误状态。	0x0
1	IntErrorStatus1	通道 1 的中断错误状态。	0x0
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.5 中断错误清除寄存器（DMACIntErrClr - 0xFFE0 4010）

DMACIntErrClr是一个只写寄存器，它清除错误中断求。写这个寄存器时，每个值为 1 的数据位会使状态寄存器中相应的位清零。值为 0 的数据位对状态寄存器中相应的位无影响。表 30.8所示为DMACIntErrClr寄存器的位分配。

表 30.8 中断错误清除寄存器（DMACIntErrClr – 地址 0xFFE0 4010）位描述

位	符号	描述	复位值
0	IntErrClr0	写入 1 来清除通道 0 的错误中断请求（IntErrorStatus0）。	-
1	IntErrClr1	写入 1 来清除通道 1 的错误中断请求（IntErrorStatus1）。	-
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.6 原始中断终端计数状态寄存器（DMACRawIntTCStatus - 0xFFE0 4014）

DMACRawIntTCStatus是一个只读寄存器，它指示在屏蔽前哪个DMA通道正在请求传输结束（终端计数中断）。值为 1 的位表明屏蔽前终端计数中断请求有效。表 30.9所示为DMACRawIntTCStatus寄存器的位分配。

表 30.9 原始中断终端计数状态寄存器（DMACRawIntTCStatus – 地址 0xFFE0 4014）位描述

位	符号	描述	复位值
0	RawIntCStatus0	屏蔽前通道 0 的终端计数中断的状态。	-
1	RawIntTCStatus1	屏蔽前通道 1 的终端计数中断的状态。	-
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.7 原始错误中断状态寄存器（DMACRawIntErrorStatus - 0xFFE0 4018）

DMACRawIntErrorStatus是一个只读寄存器，它指示屏蔽前哪个DMA通道正在请求一个错误中断。值为 1 的位表明屏蔽前错误中断请求有效。表 30.10所示为DMACRawIntErrorStatus寄存器的位分配。

表 30.10 原始错误中断状态寄存器（DMACRawIntErrorStatus – 地址 0xFFE0 4018）位描述

位	符号	描述	复位值
0	RawIntErrorStatus0	屏蔽前通道 0 的错误中断的状态。	-
1	RawIntErrorStatus1	屏蔽前通道 1 的错误中断的状态。	-
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.8 使能通道寄存器（DMACEnbldChns - 0xFFE0 401C）

DMACEnbldChns是一个只读寄存器，它指示了哪个DMA通道被使能（由DMACCxConfiguration寄存器的使能位来指示）。值为 1 的位表明相应的DMA通道被使能。DMA传输结束时相应的位被清零。表 30.11所示为DMACEnbldChns寄存器的位分配。

表 30.11 使能通道寄存器（DMACEnbldChns – 地址 0xFFE0 401C）位描述

位	符号	描述	复位值
0	EnabledChannels0	通道 0 的使能状态。	0
1	EnabledChannels1	通道 1 的使能状态。	0
31:2	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.9 软件突发请求寄存器（DMACSoftBReq - 0xFFE0 4020）

DMACSoftBReq是一个可读/写寄存器，它使能软件产生DMA突发请求。每个源的DMA请求可以通过向相应的寄存器位写 1 来产生。传输结束时寄存器位被清零。向这个寄存器写入 0 不会产生任何影响。读取寄存器来指示哪个源正在请求DMA突发传输。请求可以由外设或软件请求寄存器产生。表 30.12所示为DMACSoftBReq寄存器的位分配。

表 30.12 软件突发请求寄存器（DMACSoftBReq – 地址 0xFFE0 4020）位描述

位	符号	描述	复位值
0	SoftBReqSSP0Tx	SSP0 TX 的软件突发请求标志。	0
1	SoftBReqSSP0Rx	SSP0 RX 的软件突发请求标志。	0
2	SoftBReqSSP1Tx	SSP1 TX 的软件突发请求标志。	0
3	SoftBReqSSP1Rx	SSP1 RX 的软件突发请求标志。	0
4	SoftBReqSDMMC	SD/MMC 的软件突发请求标志。	0
31:5	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

注意：建议软件和硬件外设请求不要同时使用。

30.8.10 软件单次请求寄存器（DMACSoftSReq - 0xFFE0 4024）

DMACSoftSReq是一个可读/写寄存器，它使能软件产生DMA单次请求。每个源的DMA请求可以通过向相应的寄存器位写 1 来产生。传输结束时寄存器位被清零。向这个寄存器写入 0 不会产生任何影响。读取寄存器来指示哪个源正在请求单次DMA传输。请求可以由外设或软件请求寄存器产生。表 30.13所示为DMACSoftSReq寄存器的位分配。

表 30.13 软件单次请求寄存器（DMACSoftSReq – 地址 0xFFE0 4024）位描述

位	符号	描述	复位值
0	SoftReqSSP0Tx	SSP0 TX 的单次软件请求标志。	0
1	SoftReqSSP0Rx	SSP0 RX 的单次软件请求标志。	0
2	SoftReqSSP1Tx	SSP1 TX 的单次软件请求标志。	0
3	SoftReqSSP1Rx	SSP1 RX 的单次软件请求标志。	0
4	SoftReqSDMMC	SD/MMC 的单次软件请求标志。	0
5	SoftSReqI2S0	I2S0 的单次软件请求标志。	0
6	SoftSReqI2S1	I2S1 的单次软件请求标志。	0
31:7	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.11 软件最后一个突发请求寄存器（DMACSoftLBreq - 0xFFE0 4028）

DMACSoftLBReq是一个可读/写的寄存器，它使能软件产生DMA最后一个突发请求。每个源的DMA请求可以通过向相应的寄存器位写 1 来产生。传输结束时寄存器位被清零。向这个寄存器写入 0 不会产生任何影响。读取寄存器来指示哪个源正在请求最后一个突发DMA传输。请求可以由外设或软件请求寄存器产生。表 30.14所示为DMACSoftLBReq寄存器的位分配。

表 30.14 软件最后一个突发请求寄存器（DMACSoftLBReq – 地址 0xFFE0 4028）位描述

位	符号	描述	复位值
3:0	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA
4	SoftLBReqSDMMC	SD/MMC 的软件最后一个突发请求的标志。	0
31:5	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.12 软件最后一个单次请求寄存器（DMACSoftLSReq - 0xFFE0 402C）

DMACSoftLSReq是一个可读/写寄存器，它使能软件产生DMA最后一个单次请求。每个源的DMA请求可以通过向相应的寄存器位写入 1 来产生。传输结束时寄存器位被清零。向这个寄存器写入 0 不会产生任何影响。读取寄存器来指示哪个源正在请求最后一个单次DMA传输。请求可以由外设或软件请求寄存器产生。表 30.15所示为DMACSoftLSReq寄存器的位分配。

表 30.15 软件最后一个单次请求寄存器（DMACSoftLSReq – 地址 0xFFE0 402C）位描述

位	符号	描述	复位值
3:0	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA
4	SoftLSReqSDMMC	SD/MMC 的软件最后一个单次请求的标志。	0
31:5	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.13 配置寄存器（DMACConfiguration - 0xFFE0 4030）

DMACConfiguration是一个可读/写的寄存器，它配置GPDMA的操作。AHB主机接口的字节顺序通过写寄存器的M位来改变。复位时AHB主机接口设置成小端模式。表 30.16所示为DMACConfiguration寄存器的位分配。

表 30.16 配置寄存器（DMACConfiguration – 地址 0xFFE0 4030）位描述

位	符号	值	描述	复位值
0	E		GPDMA 使能：	0
		0	禁能。禁止 GPDMA 降低功耗。	
		1	使能。	
1	M		AHB 主机字节顺序配置：	0
		0	小端模式。	
		1	大端模式。	
31:2	-	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.8.14 同步寄存器（DMACSync - 0xFFE0 4034）

DMACSync 是一个可读/写的寄存器，它使能或禁能 DMA 请求信号的同步逻辑。DMA 请求信号由 DMACBREQ[15:0]、DMACSREQ[15:0]、DMACLBREQ[15:0]和 DMACLSREQ[15:0]组成。将一个位设为 0 来使能某组 DMA 请求的同步逻辑。将一个位设为 1 来禁能某组 DMA 请求的同步逻辑。这个寄存器被复位为 0，同步逻辑使能。

表 30.17所示为DMACSync寄存器的位分配。

表 30.17 同步寄存器（DMACSync – 地址 0xFFE0 4034）位描述

位	符号	描述	复位值
15:0	DMACSync	使能或禁能 DMA 请求信号的 DMA 同步逻辑。该位为低时表明 DMACBREQ[15:0] 、 DMACSREQ[15:0] 、 DMACLBREQ[15:0] 和 DMACLSREQ[15:0]请求信号的同步逻辑被使能；该位为高时表明同步逻辑被禁能。	0x0000
31:6	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA

30.9 通道寄存器

通道寄存器用来编程 2 个 DMA 通道。这些寄存器包括：

- 2 个 DMACCxSrcAddr 寄存器
- 2 个 DMACCxDestAddr 寄存器
- 2 个 DMACCxLLI 寄存器
- 2 个 DMACCxControl 寄存器
- 2 个 DMACCxConfiguration 寄存器

当执行分散/聚集 DMA 时，前 4 个寄存器自动更新。

30.9.1 通道源地址寄存器（DMACC0SrcAddr - 0xFFE0 4100，DMACC1SrcAddr - 0xFFE0 4120）

2 个可读/写的 DMACCxSrcAddr 寄存器包含当前要传输数据的源地址（字节对齐）。每个寄存器在相应的通道使能之前由软件直接编程。当 DMA 通道使能时，这个寄存器被更新：

- 跟随源地址的增加而更新。
- 当一个完整的数据包传输结束时源地址紧跟在链表之后。

在通道有效时读取这个寄存器不能提供有用的信息。这是因为那时软件已经处理了读取的值，通道可能已经继续向前执行了。规定只在通道停止后再读取寄存器，这样寄存器显示的是读出的最后一项的源地址。

注意：源和目标地址必须与源和目标宽度对齐。

表 30.18所示为DMACCxSrcAddr寄存器的位分配。

表 30.18 通道源地址寄存器（DMACC0SrcAddr – 地址 0xFFE0 4100，DMACC1SrcAddr – 地址 0xFFE0 4120）位描述

位	符号	描述	复位值
31:0	SrcAddr	DMA 源地址。	0x0000 0000

30.9.2 通道目标地址寄存器（DMACC0DestAddr - 0xFFE0 4104，DMACC1DestAddr - 0xFFE0 4124）

2 个可读/写的DMACCxDestAddr寄存器包含当前要传输数据的目标地址（字节对齐）。每个寄存器在相应的通道使能之前由软件直接编程。当DMA通道使能时，这个寄存器的内容跟随目标地址的增加而更新，并在一个完整的数据包传输结束后其值代表的地址紧跟在链表之后。在通道有效时读取这个寄存器不能提供有用的信息。这是因为那时软件已经处理了读取的值，通道可能已经继续向前执行了。规定只在通道停止后再读取寄存器，这样寄存器显示的是读出的最后一项的目标地址。表 30.19所示为DMACCxDestAddr寄存器的位分配。

表 30.19 通道目标地址寄存器（DMACC0DestAddr – 地址 0xFFE0 4104，DMACC1DestAddr –地址 0xFFE0 4124）位描述

位	符号	描述	复位值
31:0	DestAddr	DMA 目标地址。	0x0000 0000

30.9.3 通道链表项寄存器（DMACC0LLI - 0xFFE0 4108，DMACC1LLI - 0xFFE0 4128）

两个可读/写的 DMACCxLLI 寄存器包含下个链表项（LLI）的字对齐地址。如果 LLI 为 0，则当前的 LLI 是链的最后一项，当与之相关的所有 DMA 传输结束后 DMA 通道被禁能。

注意：在 DMA 通道使能时编程这个寄存器会产生无法预测的副作用。

表 30.20所示为DMACCxLLI寄存器的位分配。

表 30.20 通道链表项寄存器（DMACC0LLI – 地址 0xFFE0 4108，DMACC1LLI – 地址 0xFFE0 4128）位描述

位	符号	描述	复位值
0	保留	保留，读出为 0，其值不能被修改。	NA
1	R	保留，必须写入 0，读出时屏蔽。	0
31:2	LLI	链表项。下个 LLI 地址的位[31:2]。地址位[1:0]为 0。	0

注意：为了使某些系统的 LLI 装载效率更高，LLI 数据结构可以变成 4 字对齐。

30.9.4 通道控制寄存器（DMACC0Control - 0xFFE0 410C，DMACC0Control - 0xFFE0 412C）

2 个可读/写的DMACCxControl寄存器包含传输大小、突发大小和传输宽度等DMA通道的控制信息。每个寄存器在相应的DMA通道使能之前由软件直接编程。当通道被使能时，在一个完整的数据包传输结束后这个寄存器的值代表的地址紧跟在链表之后。在通道有效时读取这个寄存器不能提供有用的信息。这是因为那时软件已经处理了读取的值，通道可能已经继续向前执行了。规定只在通道停止后再读取寄存器。表 30.21所示为DMACCxControl寄存器的位分配。

表 30.21 通道控制寄存器（DMACC0Control – 地址 0xFFE0 410C，DMACC1Control – 地址 0xFFE0 412C）位描述

位	符号	描述	复位值
11:0	TransferSize	传输大小。当 GPDMA 是流控制器时，写这个字段来设置传输的大小。读取这个字段来指示在目标总线上已完成的传输数量。在通道有效时读取这个寄存器不能提供有用的信息，因为那时软件已经处理了读出的值，通道可能已经继续向前执行了。规定该字段读出的值只在通道先被使能然后又被禁能时才使用。如果 GPDMA 不是流控制器，则不使用传输大小值。	0
14:12	SBSize	源突发大小。该字段指明组成一个源突发的传输数量。这个值必须设置成源外设的突发大小，如果源是存储器，这个值就设置成存储器边界大小。当源外设的 DMACBREQ 信号变为有效时，突发大小为传输的数据量。	0
17:15	DBsize	目标突发大小。该字段指明组成一个目标突发传输请求的传输数量。这个值必须设置成目标外设的突发大小，如果目标是存储器，这个值就设置成存储器边界大小。当目标外设的 DMACBREQ 信号变为有效时，突发大小为传输的数据量。	0
20:18	SWidth	源传输宽度。不允许传输宽度大于 AHB 主机总线宽度。源和目标宽度可以不同。硬件会根据需要自动打包和拆分数据。	0
23:21	DWidth	目标传输宽度。不支持传输宽度大于 AHB 主机总线宽度。源和目标宽度可以不同。硬件会根据需要自动打包和拆分数据。	0
25:24	-	保留，用户软件不应该向保留位写入 1。从保留位读出的值未定义。	NA
26	SI	源增加。该位置位时每次传输后源地址都增加。	0
27	DI	目标增加。该位置位时每次传输后目标地址都增加。	0
30:28	Prot	保护。	0
31	I	终端计数中断使能位。它控制着是否希望当前 LLI 触发终端计数中断。	0

表 30.22所示为 3 位DBSize或SBSize字段的值及其对应的突发大小。

表 30.22 源或目标突发大小

DBSize 或 SBSize 的位值	源或目标突发传输请求大小
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

表 30.23所示为 3 位SWidth或DWidth字段的值及其对应的传输宽度。

表 30.23 源或目标传输宽度

DBWidth 或 SBWidth 的位值	源或目标突发传输请求大小
000	字节（8 位）
001	半字（16 位）
010	字（32 位）
011 和 1xxx	保留

30.9.5 保护和访问信息

当传输发生时，AHB访问信息被提供给源和目标外设。传输信息通过编程DMA通道（DMACCxControl寄存器的Prot位和DMACCxConfiguration寄存器的Lock位）来提供。这两位由软件编程，如果需要，外设可以使用这个信息。表 30.24提供了 3 个位的信息，并给出了 3 个保护位的作用。

表 30.24 保护位

DMACC1Control	值	描述	复位值
28		特权或用户。该位控制 AHB HPROT[1]信号。指明是用户模式还是特权模式中的访问。	0
	0	用户模式。	
	1	特权模式。	
29		可缓冲或不可缓冲。该位指示访问是可缓冲的。例如，该位可以向 AMBA 桥指示在源总线上读操作可以在零等待状态中完成，无需等待目标总线的仲裁和等待从机接收数据。该位控制着 AHB HPROT[2]信号。 指示访问是可缓冲的或不可缓冲的：	0
	0	不可缓冲。	
	1	可缓冲。	
30		可缓存或不可缓存。该位指示访问是可缓存的。例如，该位可以用来向 AMBA 桥指示当它看见第一个 8 个读突发时它可以在目标总线上传输整个 8 个读突发，而不是通过一次一个的方式来传输。该位控制 AHB HPROT[3]信号。 指示访问是可缓存的或不可缓存的：	0
	0	不可缓存。	
	1	可缓存。	

30.9.6 通道配置寄存器（DMACC0Configuration - 0xFFE0 4110，DMACC1Configuration - 0xFFE0 4130）

这 2 个DMACCxConfiguration寄存器是可读/写的，但位[17]除外，它是只读的。这两个寄存器用来配置DMA通道。当请求一个新的LLI时寄存器不能更新。表 30.25所示为DMACCxConfiguration寄存器的位分配。

表 30.25 通道配置寄存器（DMACC0Configuration – 地址 0xFFE0 4110 和 DMACC1Configuration – 地址 0xFFE0 4130）位描述

位	符号	值	描述	复位值
0	E		通道使能位状态也可以通过读取 DMACEnbldChns 寄存器获得。 通道通过置位该位来使能。 一个通道也可以通过清零使能位被禁能。这会造成在当前的 AHB 传输（如果有传输正在执行）结束后禁能通道。相应的通道 FIFO 中的所有数据都被丢失。通过置位通道使能位来重启通道会产生不可预知的影响，通道必须被完全地重新使能。 当到达最后的 LLI 时或碰到通道错误时，通道被禁能，通道使能位被清零。 如果某个通道必须被禁能而又不丢失 FIFO 中的数据，则必须置位 Halt 位，使得后面的 DMA 请求被忽略。然后必须查询 Active 位直至它达到 0，表示 FIFO 中已经没有数据了。最后清除通道使能位。 通道使能－读该位指示通道当前是被使能还是被禁能。	0
		0	通道禁能。	
		1	通道使能。	
4:1	SrcPeripheral		源外设。这个值选择 DMA 源请求外设。如果传输的源是存储器，则这个字段被忽略。	0
		0000	SSP0 Tx	
		0001	SSP0 Rx	
		0010	SSP1 Tx	
		0011	SSP1 Rx	
		0100	SD/MMC	
		0101	I2S 通道 0	
		0110	I2S 通道 1	
		0111 或 1xxx	这些值保留，不应该被使用。	
5	-	-	保留，不能修改，读时被屏蔽。	NA
9:6	DestPeripheral		目标外设。这个值选择 DMA 目标请求外设。如果传输的目标是存储器，则忽略这个字段。请参考各个值的 SrcPeripheral 符号描述。	0
10	-	-	保留，不能修改，读时被屏蔽。	NA
13:11	FlowCntrl		流控制和传输类型。这个值指示了流控制器和传输类型。流控制器可以是 GPDMA、源外设或目标外设。传输类型可以是存储器到存储器、存储器到外设、外设到存储器或外设到外设。	0
14	IE		中断错误屏蔽。该位清零时将屏蔽相关通道的错误中断。	0
15	ITC		终端计数中断屏蔽。该位清零时将屏蔽相关通道的终端计数中断。	0

续表 30.25

位	符号	值	描述	复位值
16	L		锁定。该位置位时使能锁定的传输。	0
17	A		Active。该位可以和 Halt 以及通道使能位一起使用来彻底地禁能一个 DMA 通道。写这个位无任何影响。	
		0	通道的 FIFO 中没有数据。	
		1	通道 FIFO 中有数据。	
18	H		Halt。通道 FIFO 的内容被取走。这个值可以和 Active 以及通道使能位一起使用来彻底地禁能一个 DMA 通道。	0
		0	使能 DMA 请求。	
		1	忽略后面的源 DMA 请求。	
31:19	-		保留，不能修改，读时被屏蔽。	NA

30.9.7 锁定控制

通过编程 DMACCxConfiguration 寄存器的位 16 来置位锁定位。

当突发产生时，在突发过程中 AHB 仲裁器不允许剥夺主机的授权，直至锁定被取消。GPDMA 可以在一个单次突发（例如一个长源取数突发或一个长目标输出数据突发）时被锁定。在先是一个源取数突发、后面紧跟一个目标输出数据的情况下，GPDMA 通常不会连续被锁定。

在一些情况下，GPDMA 会在先是源传输、后面紧跟着目标传输的情况下被锁定。这可能是 GPDMA 的内部条件允许它在源取数后继续执行目标输出数据。

30.9.8 流控制和传输类型

表 30.26列出了 3 个流控制和传输类型位的位值。

表 30.26 流控制和传输类型位

位值	传输类型	控制器
000	存储器到存储器	DMA
001	存储器到外设	DMA
010	外设到存储器	DMA
011	源外设到目标外设	DMA
100	源外设到目标外设	目标外设
101	存储器到外设	外设
110	外设到存储器	外设
111	源外设到目标外设	源外设

30.10 地址的产生

地址的产生可以是递增或非递增（不支持地址环回）。突发不能超出 1kB 的地址边界。

30.11 分散/聚集

使用链表来支持分散/聚集。这就意味着源和目标区不需要占用连续的存储器空间。不需要分散/聚集的地方 DMACCxLLI 寄存器必须被设置成 0。

源和目标数据区由一连串的链表来定义。每个链表项（LLI）控制着一个数据块的传输，然后选择装载另一个 LLI 来继续 DMA 操作或停止 DMA 流。第一个 LLI 被编程到 GPDMA。

一个 LLI 描述的传输数据（指的是数据包）通常需要一个或多个 DMA 突发（到每个源和目标）。

30.11.1 链表项

一个链表项（LLI）由 4 个字组成。这些字按照以下顺序来排列：

1. DMACCxSrcAddr
2. DMACCxDestAddr
3. DMACCxLLI
4. DMACCxControl

注意：DMACCxConfiguration DMA 通道配置寄存器不是链表项的一部分。

30.11.2 编程GPDMA用于分散/聚集DMA

执行下列操作来编程 GPDMA，使之用于分散/聚集 DMA：

1. 写 LLI 来完成到存储器的 DMA 传输。每个链表项包含 4 个字：
   - 源地址。
   - 目标地址。
   - 指向下个 LLI 的指针。
   - 控制字。

   最后一个 LLI 有自己的链表字指针（设为 0）。LLI 必须存放在 GPDMA 可以访问的存储器（即 AHB1 SRAM 和外部存储器）。
2. 选择一个具有所需优先级的空闲 DMA 通道。DMA 通道 0 的优先级最高，DMA 通道 1 的优先级最低。
3. 将第一个链表项写入 GPDMA 的相关通道（先前写入存储器）。
4. 向通道配置寄存器写入通道配置信息并置位通道使能位。然后，随着每个链表项的装载，GPDMA 传输第一个及其后面的数据包。
5. 每个 LLI 结束时可以产生中断，由 DMACCxControl 寄存器的终端计数位决定。如果这个位被置位，则在相应的 LLI 结束时产生中断。然后，执行中断请求服务，置位 DMACIntTCClear 寄存器的相应位来清除中断。

30.11.3 分散/聚集DMA的例子

图 30.3所示为一个LLI的例子。一个矩形存储器的内容传输到一个外设。每行数据的地址以十六进制的形式在图的左手边给出。描述传输的LLI存放在从地址 0x20000 开始的连续区域。

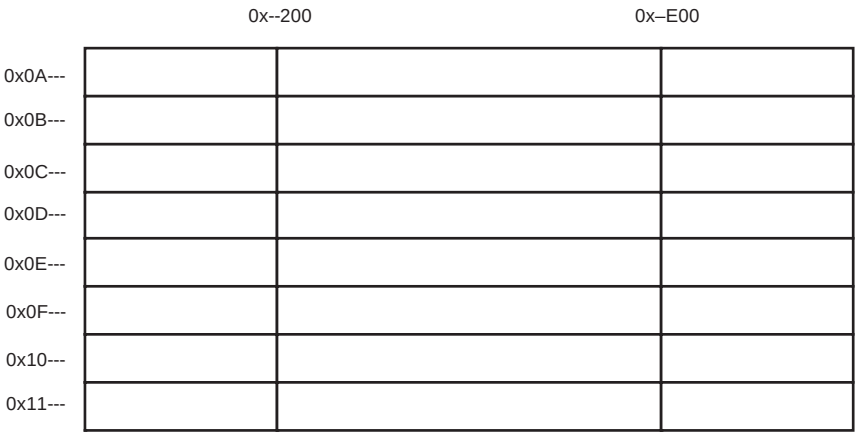


图 30.3 LLI 的例子

第一个 LLI，存放在 0x20000，定义了要传输的第一个数据块，这个数据块是存放在 0x0A200 和 0x0AE00 之间的数据。

- 源起始地址 0x0A200。
- 目标地址设为目标外设地址。
- 传输宽度：1 个字（32 位）。
- 传输大小：3072 个字节（0xC00）。
- 源和地址突发大小：16 个传输。
- 下个 LLI 地址：0x20010。

第二个 LLI，存放在 0x20010，描述了要传输的下个数据块：

- 源起始地址 0x0B200。
- 目标地址设为目标外设地址。
- 传输宽度：1 个字（32 位）。
- 传输大小：3072 个字节（0xC00）。
- 源和目标突发大小：16 个传输。
- 下个 LLI 地址：0x20020。

一个描述符链就形成了，每个描述符都指向链中的下一个描述符。为了初始化 DMA 流，第一个 LLI（位于 0x20000）编程到 GPDMA。当第一个数据包被传输后，下个 LLI 就自动被装载。

最后一个 LLI 存放在 0x20070，它包含：

- 源起始地址 0x11200。
- 目标地址设为目标外设地址。
- 传输宽度：1 个字（32 位）。
- 传输大小：3072 个字节（0xC00）。
- 源和目标突发大小：16 个传输。
- 下个 LLI 地址：0x0。

由于下个 LLI 地址被设为 0，因此，这是最后一个描述符，而且，DMA 通道在传输完

最后这个数据项后被禁能。这时，通道也可能设置成产生中断来向 ARM 处理器指明通道可以被重新编程了。

30.12 中断请求

当遇到一个 AHB 错误，或者在当前 LLI 对应的所有数据都被传输到目标之后传输结束（终端计数）时，可以产生中断请求。中断可以通过编程 DMACCxControl 和 DMACCxConfiguration 通道寄存器的相应位被屏蔽。还提供了中断状态寄存器，该寄存器将中断屏蔽之前（DMACRawIntTCStatus 和 DMACRawInt ErrorStatus）和中断屏蔽之后（DMACIntTCStatus 和 DMACIntErrorStatus）所有 DMA 通道的中断请求组合起来。DMACIntStatus 寄存器将 DMACIntTCStatus 和 DMACIntErrorStatus 请求组合到一个寄存器中，使得中断源能很快被找到。写 DMACIntTCClear 或 the DMACIntErrClr 寄存器，使某个位置高可以选择性地清除中断。

30.12.1 硬件中断序列流

当一个 DMA 中断请求出现时，中断服务程序需要：

1. 读取 DMACIntStatus 寄存器来确定哪个通道产生了中断。如果多个请求有效，建议先检查最高优先级的通道。
2. 读取 DMACIntTCStatus 寄存器来判断中断是否由于传输的结束而产生（终端计数）。位为高时表明传输结束。
3. 读取 DMACIntErrorStatus 寄存器来判断中断是否是因为错误的出现而导致的。位为高时表明出现了错误。
4. 处理中断请求。
5. 对于一个终端计数中断，可以向 DMACIntTCClr 寄存器的相关位写入 1；对于一个错误引起的中断，可以向 DMACIntErrClr 寄存器的相关位写入 1 来清除中断请求。

30.12.2 中断查询序列流

当 GPDMA 中断请求信号被屏蔽、在中断控制器中被禁能或在处理器中被禁能时，使用中断查询序列流。当查询 GPDMA 时，你必须：

1. 读取 DMACIntStatus 寄存器。如果寄存器中的位都不为高，则重复执行这一步骤，否则转去执行步骤 2。如果多个请求有效，建议先检查最高优先级的通道。
2. 读取 DMACIntTCStatus 寄存器来判断中断是否是因为传输的结束而产生的（终端计数）。位为高时表明传输结束。
3. 处理中断请求。
4. 对于一个终端计数中断，可以向 DMACIntTCClr 寄存器的相关位写入 1；对于一个错误引起的中断，可以向 DMACIntErrClr 寄存器的相关位写入 1 来清除中断请求。

30.13 GPDMA数据流

本节描述了 4 个被允许的传输类型的 GPDMA 数据流序列：

- 存储器到外设。

- 外设到存储器。
- 存储器到存储器。
- 外设到外设。

每个传输类型都可以将外设或 GPDMA 作为流控制器，因此，有 8 种可能的控制情况。表 30.27所示为每种类型的传输使用的请求信号。

表 30.27 DMA 请求信号的使用

传输方向	请求发生器	流控制器
存储器到外设	外设	GPDMA
存储器到外设	外设	外设
外设到存储器	外设	GPDMA
外设到存储器	外设	外设
存储器到存储器	GPDMA	GPDMA
源外设到目标外设	源外设和目标外设	源外设
源外设到目标外设	源外设和目标外设	目标外设
源外设到目标外设	源外设和目标外设	GPDMA

30.13.1 外设到存储器或存储器到外设DMA流

对于外设到存储器或存储器到外设的 DMA 流，会出现以下序列：

1. 编程和使能 DMA 通道。
2. 等待一个 DMA 请求。
3. 当下列情况出现时，GPDMA 开始传输数据：
   - DMA 请求变得有效。
   - DMA 流具有最高挂起优先级。
   - GPDMA 是 AHB 总线的总线主机。
4. 如果在传输数据时出错，则产生一个错误中断，禁能 DMA 流，流序列终止。
5. 如果 GPDMA 正在执行流控制，则传输计数减 1。
6. 如果传输已经结束（当 GPDMA 正在执行流控制时通过传输计数到达 0 来指示；当外设正在执行流控制时通过外设发送一个 DMA 请求来指示）：
   - GPDMA 用一个 DMA 应答来响应
   - 产生终端计数中断（这个中断可以被屏蔽）。
   - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCxLLI 和 DMACCxControl 寄存器，并返回到步骤 2。但是，如果 DMACCxLLI 为 0，则 DMA 流被禁能，流序列终止。

30.13.2 外设到外设DMA流

对于外设到外设的 DMA 流，会出现以下序列：

1. 编程和使能 DMA 通道。
2. 等待一个源 DMA 请求。
3. 当下列情况出现时，GPDMA 开始传输数据：
   - DMA 请求变得有效。
   - DMA 流具有最高的挂起优先级。

- GPDMA 是 AHB 总线的总线主机。

4. 如果在传输数据时出错，则产生一个错误中断，然后结束。
5. 如果 GPDMA 正在执行流控制，则传输计数减 1。
6. 如果传输已经结束（当 GPDMA 正在执行流控制时通过传输计数到达 0 来指示；当外设正在执行流控制时通过外设发送一个 DMA 请求来指示）：
   - GPDMA 用一个对源外设的 DMA 应答来响应。
   - 忽略后面的源 DMA 请求。
7. 当目标 DMA 请求变得有效，并且 GPDMA FIFO 中有数据时，将数据传输到目标外设。
8. 如果传输数据时出错，则产生一个错误中断，禁能 DMA 流，流序列终止。
9. 如果传输已经结束，则当 GPDMA 正在执行流控制时通过传输计数到达 0 来指示；当外设正在执行流控制时通过发送一个 DMA 请求来指示。执行以下操作：
   - GPDMA 用一个对目标外设的 DMA 应答来响应。
   - 产生终端计数中断（这个中断可以被屏蔽）。
   - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCxLLI 和 DMACCxControl 寄存器，并返回到步骤 2。但是，如果 DMACCxLLI 为 0，则 DMA 流被禁能，流序列终止。

30.13.3 存储器到存储器DMA流

对于存储器到存储器的 DMA 流，会出现以下序列：

1. 编程和使能 DMA 通道。
2. 只要 DMA 通道具有最高挂起优先级并且 GPDMA 获得 AHB 总线的控制权，就传输数据。
3. 如果传输数据时出现错误，则产生错误中断并禁能 DMA 流。
4. 传输计数减 1。
5. 如果计数已经到达 0：
   - 产生一个终端计数中断（这个中断可以被屏蔽）。
   - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCxLLI 和 DMACCxControl 寄存器并返回步骤 2。但是，如果 DMACCxLLI 为 0，则 DMA 流被禁能，流序列终止。

注意：存储器到存储器的传输应该编程为低通道优先级，否则在存储器到存储器传输结束之前其他 DMA 通道都不能访问总线，或者其他 AHB 主机不能执行任何传输。

30.14 流控制

控制包长度的外设就被称为流控制器。流控制器通常是 GPDMA，包长度由软件在 DMA 通道使能之前编程。如果在 DMA 通道使能时包长度不可知，则源外设或目标外设可以用作流控制器。

对于知道包长度的简单或低性能外设（即，当外设是流控制器时）来说，指示一次传输结束的简单方法是外设产生一个中断，使能处理器重新编程 DMA 通道。

如果一个外设配置用作流控制器，则 DMACCxControl 寄存器的传输大小值就被忽略。

当执行 DMA 传输时：

1. GPDMA 向外设产生一个应答来指示传输已经结束。
2. 产生一个 TC 中断（如果这个中断被使能）。
3. GPDMA 继续执行下个 LLI。

第31章 EmbeddedICE

31.1 特性

- 通过软件调试器启动调试会话，不需要目标资源；
- 允许软件调试器通过 JTAG（联合测试行动小组）端口直接与内核进行对话；
- 在 ARM7TDMI-S 内核中直接插入指令；
- 通过插入不同类型的指令可对 ARM7TDMI-S 内核或系统状态进行检查、保存或修改；
- 允许指令在低调试速度或高系统速度下执行。

31.2 应用

EmbeddedICE 逻辑提供片内调试支持。目标系统调试需要一个运行调试软件的主机和一个 EmbeddedICE 协议转换器。EmbeddedICE 协议转换器将远程调试协议命令转换成访问目标系统的 ARM7TDMI-S 内核所需的 JTAG 数据。

31.3 描述

ARM7TDMI-S 调试结构使用现有的 JTAG[3] 端口来访问内核。供产品测试用的内核周围的扫描链在调试状态下重新用来捕获数据总线的信息并向内核或存储器插入新的信息。在 ARM7TDMI-S 中有两个 JTAG 类型的扫描链。一个 JTAG 类型的测试访问端口控制器控制扫描链。除了扫描链之外，调试结构使用位于 ARM7TDMI-S 核内部的 EmbeddedICE 逻辑。EmbeddedICE 使用自身的扫描链向 ARM7TDMI-S 内核插入观察点和断点。EmbeddedICE 逻辑包含 2 个实时观察点寄存器和 1 个控制和状态寄存器。这两个观察点寄存器或其中的一个可编程为暂停 ARM7TDMI-S 内核。当编程到 EmbeddedICE 逻辑中的值与当前出现在地址总线、数据总线和某些控制信号上的值匹配时，内核的运行将暂停。可以屏蔽任何位使其不会影响比较操作。观察点寄存器可以配置为观察点（即对于数据的访问）或断点（指令取指）。观察点和断点可以按照下面的方式进行组合：

- 在停止 ARM7TDMI-S 内核之前，必须满足 2 个观察点的条件。CHAIN 的功能要求在暂停内核之前满足两个连续的条件。例如，将第一个断点设定为在访问外设时触发，而第二个断点在执行任务切换的代码段时触发。当断点触发时，与任务切换有关的信息准备就绪以备检查。
- 观察点可以配置为在一段地址范围内对观察点有效。RANGE 功能允许实现一个组合的断点，例如在访问存储器最低 256 字节但不访问最低 32 字节时产生断点。

ARM7TDMI-S 内核有一个内置的调试通信通道功能。调试通信通道允许运行在目标上的程序与主调试器或其他独立的主机进行通信而不中断程序流程或甚至不用进入调试状态。ARM7TDMI-S 内核上运行的程序将调试通信通道作为协处理器 14 进行访问。调试通信通道允许 JTAG 端口发送和接收数据，但不影响正常的程序流程。调试通信通道数据和控制寄存器映射到 EmbeddedICE 逻辑中的地址。

- 更多详细信息请参考 IEEE 标准 1149.1 – 1990 标准测试访问端口和边界扫描结构。

31.4 管脚描述

表 31.1 EmbeddedICE 管脚描述

管脚名称	类型	描述
TMS	输入	**测试模式选择**。TMS 管脚选择 TAP 状态机中的下一个状态。
TCK	输入	**测试时钟**。该管脚允许 TMS 和 TDI 管脚上数据输入的移位。它是一个上升沿触发时钟，由 TMS 和 TCK 信号定义器件的内部状态。
TDI	输入	**测试数据输入**。移位寄存器的串行数据输入端。
TDO	输出	**测试数据输出**。移位寄存器的串行数据输出端。器件中的数据在 TCK 信号的下降沿输出。
nTRST	输入	**测试复位**。nTRST 管脚可用于复位 EmbeddedICE 逻辑中的测试逻辑。
RTCK	输出	**返回的测试时钟**。叠加到 JTAG 端口的额外信号。基于 ARM7TDMI-S 处理器内核进行设计时需要该信号。Multi-ICE（ARM 的开发系统）使用该信号来保持与低或宽范围时钟频率的目标系统的同步。详见“Multi-ICE 系统设计注意事项应用笔记注 72（ARM DAI 0072A）”。 板设计者可能需要连接一个弱偏置电阻到该管脚，正如下面描述的一样。

31.5 JTAG功能选择

JTAG 端口可以用来调试或边界扫描。DBGEN 管脚的状态决定了哪种功能可用。当 DBGEN＝0 时，JTAG 端口可以用于边界扫描。当 DBGEN＝1 时，JTAG 端口可以用来调试。

31.6 寄存器描述

EmbeddedICE逻辑包含 16 个寄存器，如下面的表 31.2所示。ARM7TDMI-S调试结构在ARM公司出版的“ARM7TDMI-S(rev 4)技术参考手册”（ARM DDI 0234A）中有详细描述。

表 31.2 EmbeddedICE 逻辑寄存器

名称	宽度	描述	地址
调试控制	6	强制调试状态，禁止中断	00000
调试状态	5	调试状态	00001
调试通信控制寄存器	32	调试通信控制寄存器	00100
调试通信数据寄存器	32	调试通信数据寄存器	00101
观察点 0 地址值	32	保存观察点 0 地址值	01000
观察点 0 地址屏蔽	32	保存观察点 0 地址屏蔽	01001
观察点 0 数据值	32	保存观察点 0 数据值	01010
观察点 0 数据屏蔽	32	保存观察点 0 数据屏蔽	01011
观察点 0 控制值	9	保存观察点 0 控制值	01100
观察点 0 控制屏蔽	8	保存观察点 0 控制屏蔽	01101
观察点 1 地址值	32	保存观察点 1 地址值	10000
观察点 1 地址屏蔽	32	保存观察点 1 地址屏蔽	10001
观察点 1 数据值	32	保存观察点 1 数据值	10010
观察点 1 数据屏蔽	32	保存观察点 1 数据屏蔽	10011

续表 31.2

名称	宽度	描述	地址
观察点 1 控制值	9	保存观察点 1 控制值	10100
观察点 1 控制屏蔽	8	保存观察点 1 控制屏蔽	10101

31.7 方框图

调试环境的方框图如下面的图 31.1所示。

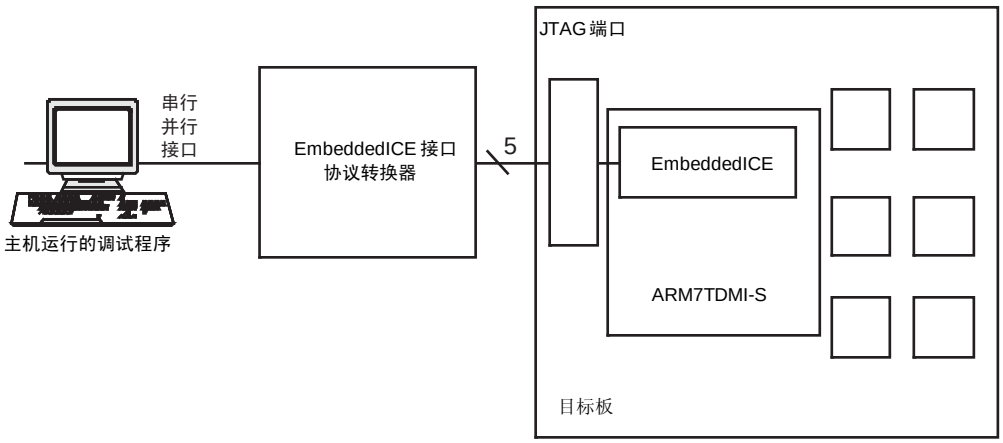


图 31.1 EmbeddedICE 调试环境方框图

第32章 嵌入式跟踪宏单元（ETM）

32.1 特性

- 精确跟踪 ARM 内核正在执行的指令；
- 1 个外部触发输入；
- 10 线接口；
- 所有寄存器都通过 JTAG 接口编程；
- 不使用跟踪时不消耗功率；
- 支持 THUMB 指令集。

32.2 应用

由于微控制器带有大量的片内存储器，因此不可能简单地通过观察外部管脚来确定处理器核是如何运行的。ETM 对深嵌入处理器内核提供了实时跟踪能力。它向一个跟踪端口输出处理器执行的信息。软件调试器允许使用 JTAG 接口对 ETM 进行配置并以用户容易理解的格式显示捕获到的跟踪信息。

32.3 描述

ETM 直接连接到 ARM 内核而不是主 AMBA 系统总线。它将跟踪信息压缩并通过一个窄带跟踪端口输出。外部跟踪端口分析仪在软件调试器的控制下捕获跟踪信息。跟踪端口可以广播指令跟踪信息。指令跟踪(或 PC 跟踪)显示了处理器的执行流程并提供所有已执行指令的列表。指令跟踪被显著压缩为广播分支地址和一套用于指示流水线状态的状态信号。跟踪信息的产生可通过选择触发源进行控制。触发源包括地址比较器、计数器和序列发生器。由于跟踪信息被压缩，因此软件调试器需要一个执行代码的静态映像。由于这个限制，自修改代码无法被跟踪。

32.3.1 ETM配置

ETM 宏单元使用下面的标准配置。

表 32.1 ETM 配置

资源数/类型	Small[1]
地址比较器对	1
数据比较器	0（不支持数据跟踪）
存储器映射译码器	4
计数器	1
时序发生器	No
外部输入	2
外部输出	0
FIFOFULL 信号	Yes（未连接）

续表 32.1

资源数/类型	Small[1]
FIFO 深度	10 字节
跟踪包宽度	4/8

[1] 详见 ARM 文档“嵌入式跟踪宏单元规范”（ARM IHI 0014E）。

32.4 管脚描述

表 32.2 ETM 管脚描述

管脚名称	类型	描述
TRACECLK	输出	**跟踪时钟**。跟踪时钟信号为跟踪端口提供时钟。PIPESTAT[2:0], TRACESYNC 和 TRACEPKT[3:0]信号以跟踪时钟的上升沿为参照。该时钟并不由 ETM 模块产生，而是由系统时钟得来的。该时钟应当为跟踪数据信号提供足够的保持时间。支持半速率时钟模式。跟踪数据信号应当在 TRACECLK 的时钟相位作用下移出。详见“ETM7 技术参考手册”(ARM DDI 0158B)的图 3.14 和图 3.15，例如相对于时钟的跟踪数据的半速率时钟和移位的实现电路。而 TRACECLK 时序请参阅“嵌入式跟踪宏单元规范”(ARM IHI 0014E)。
PIPESTAT[2:0]	输出	**流水线状态**。流水线状态信号提供处理器流水线执行阶段中所发生状况的指示。
TRACESYNC	输出	**跟踪同步**。跟踪同步信号用来指示一组跟踪包中的第一个包。并且只在任何分支地址的第一个包出现时声明为高电平。
TRACEPKT[3:0]	输出	**跟踪包**。跟踪包信号用来输出流水线状态相关的打包的地址和数据信息。所有包的长度都为 8 位。一个包需要两个周期输出，在第一个周期中，Packet[3:0]为输出，第二个周期 Packet[7:4]为输出。
EXTIN[0]	输入	**外部触发输入。**

32.5 寄存器描述

ETM包含 29 个寄存器，如表 32.3所示。有关它们的详细描述见ARM有限公司出版的ARM IHI 0014E文档。

表 32.3 ETM 寄存器

名称	描述	访问	寄存器编码
ETM 控制	控制 ETM 的一般操作。	R/W	000 0000
ETM 配置代码	允许调试器读取每种资源类型的数目。	RO	000 0001
触发事件	保存控制事件。	WO	000 0010
存储映射译码控制	8 位寄存器，用来静态配置存储器映射译码器。	WO	000 0011
ETM 状态	保存挂起的溢出状态位。	RO	000 0100
系统配置	保存使用 SYSOPT 总线的配置信息。	RO	000 0101
跟踪使能控制 3	保存跟踪开启/关闭地址。	WO	000 0110
跟踪使能控制 2	保存比较的地址。	WO	000 0111
跟踪使能事件	保存使能的事件。	WO	000 1000
跟踪使能控制 1	保存包含和排除的区域。	WO	000 1001
FIFOFULL 区域	保存包含和排除的区域。	WO	000 1010

续表 32.3

名称	描述	访问	寄存器编码
FIFOFULL 级别	保存一个级别，低于该级别就认为 FIFO 已满。	WO	000 1011
ViewData 事件	保存使能的事件。	WO	000 1100
ViewData 控制 1	保存包含/排除的区域。	WO	000 1101
ViewData 控制 2	保存包含/排除的区域。	WO	000 1110
ViewData 控制 3	保存包含/排除的区域。	WO	000 1111
地址比较器 1～16	保存比较的地址。	WO	001 xxxx
地址访问类型 1～16	保存访问的类型和规格。	WO	010 xxxx
保留	-	-	000 xxxx
保留	-	-	100 xxxx
初始计数值 1～4	保存计数器的初始值。	WO	101 00xx
计数器使能 1～4	保存计数器时钟使能控制和事件。	WO	101 01xx
计数器重装 1～4	保存计数器重装事件。	WO	101 10xx
计数值 1～4	保存当前的计数器值。	RO	101 11xx
序列发生器状态和控制	保存下一个状态触发的事件。	-	110 00xx
外部输出 1～4	保存每个输出的控制事件。	WO	110 10xx
保留	-	-	110 11xx
保留	-	-	111 0xxx
保留	-	-	111 1xxx

32.6 复用管脚的复位状态

LPC2400 的ETM管脚功能可以复用为GPIO、PWM、UART和CAN功能。为了使用跟踪特性，这个管脚必须配置为选择跟踪功能。详情请参考第 9 章“管脚连接”。

32.7 方框图

ETM调试环境的方框图如图 32.1所示。

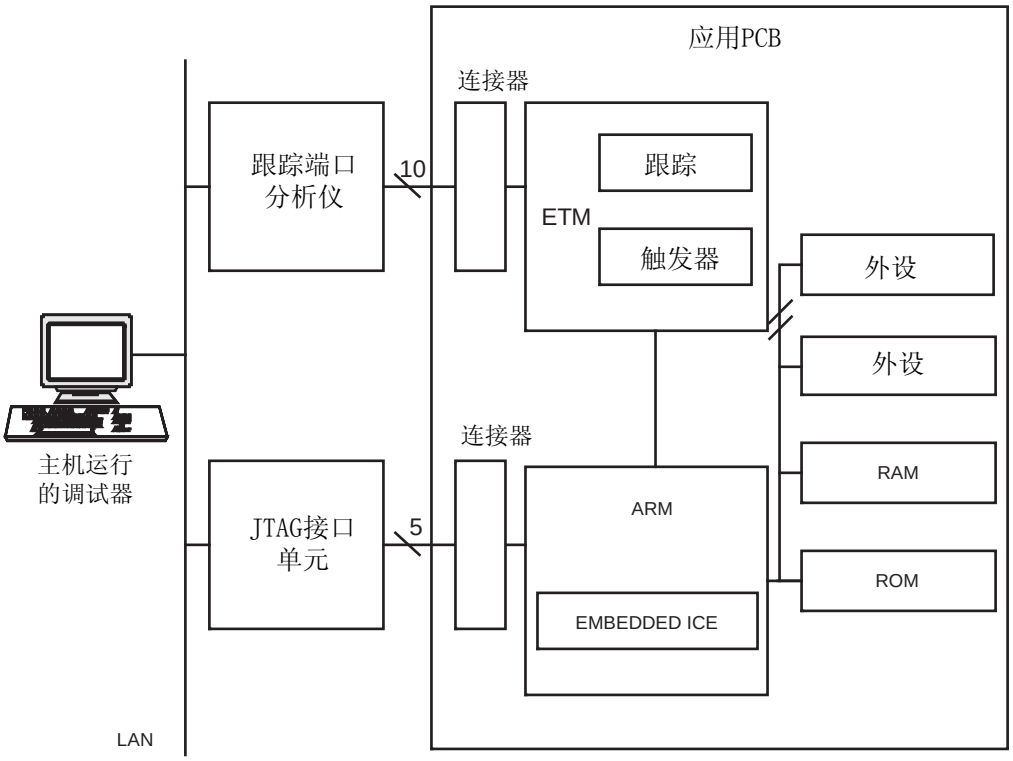


图 32.1 ETM 调试环境方框图

第33章 RealMonitor

33.1 特性

说明：RealMonitor 是一个可配置的软件模块，它由 ARM 公司开发，可以提供实时调试。本章所描述的信息摘自 ARM 文档 *RealMonitor 目标集成指南（ARM DUI 0142A）*。这些信息应用于 RealMonitor 软件的特定配置，RealMonitor 软件编程到 LPC2400 器件的片内 ROM 引导存储器。

- 允许用户在不暂停或复位系统的情况下，与当前运行的系统建立调试会话。
- 在其它用户应用代码调试过程中，允许用户的实时中断代码连续执行。

33.2 应用

实时调试。

33.3 描述

RealMonitor 是一个轻巧的调试监视器，它允许在用户调试前台应用程序时对中断进行服务。它通过 DCC（调试通信通道）与主机进行通信，DCC 位于 EmbeddedICE 逻辑当中。RealMonitor 比 ARM 系统中传统的调试方法更具优势。传统的调试方法包括：

- Angel（基于目标的调试监视器）
- Multi-ICE 或其它 JTAG 单元和 EmbeddedICE 逻辑（基于硬件的调试方案）

尽管这两种方法都提供了健壮的调试环境，但并不适合作为一个轻巧的实时监视器。

Angel 设计成装载和调试可以运行在各种不同模式下的应用程序，它通过不同的连接与调试主机进行通信（例如串口或者以太网）。Angel 要求保存和恢复所有的处理器上下文，这样做的结果是使中断产生延迟。Angel 作为一个全功能的基于目标的调试器，对于执行实时监视来说显得过于笨重了。

Multi-ICE 是一种硬件调试方案。它使用内置在大多数 ARM 处理器中的 EmbeddedICE 单元来执行操作。为了执行访问存储器或处理器寄存器这样的调试任务，Multi-ICE 必须使内核进入调试状态。当处理器处于调试状态时（这段时间可能长达数百万个周期），正常的程序执行被挂起，中断也无法执行。

RealMonitor结合了Angel和Multi-ICE的特性和机制来提供必需的服务和功能。此外，它还包含了Multi-ICE的通信机制（使用JTAG的DCC）和类似Angel的保存和恢复处理器上下文。RealMonitor被预先编程到片内ROM存储器中（引导扇区）。当RealMonitor使能时，它允许用户在部分应用程序继续运行时进行观察和调试。详见33.4 节“如何使能RealMonitor”的内容。

33.3.1 RealMonitor部件

如图 33.1所示，RealMonitor分成两个功能部件：

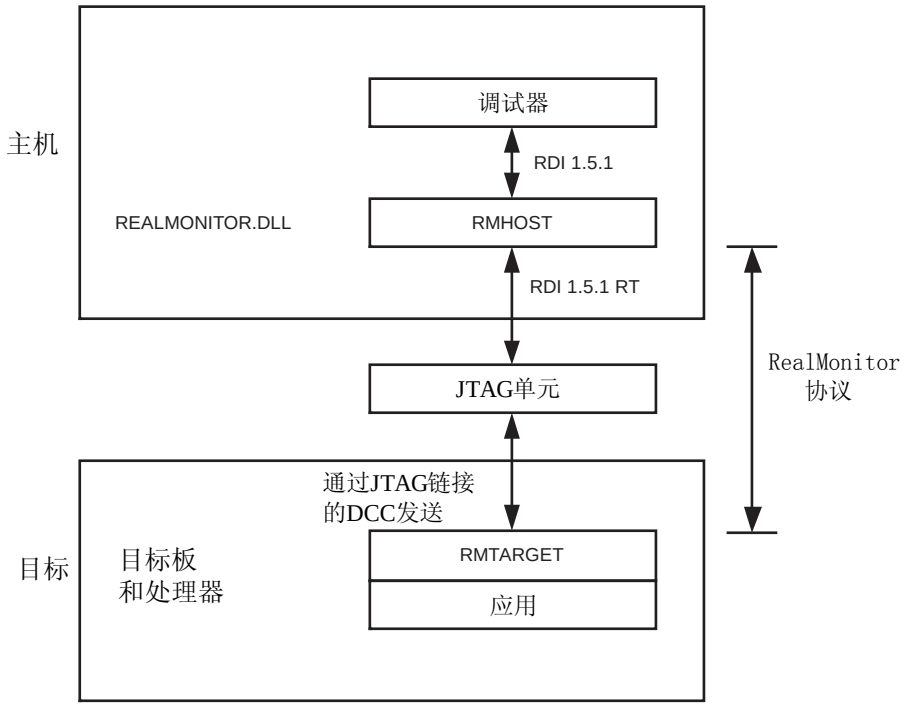


图 33.1 RealMonitor 部件

33.3.1.1 RMHost

RMHost 位于调试器和 JTAG 单元之间。RMHost 控制器和 RealMonitor.dll 将通用远程调试接口（RDI）请求从调试器转换为 JTAG 单元的仅适用于 DCC 的 RDI 信息。有关主机 RealMonitor 集成应用调试的完整信息请参考*“ARM RMHost 用户指南”（ARM DUI 0137A）*。

33.3.1.2 RMTarget

这部分预先编程到片内 ROM 存储器（引导扇区）并在目标硬件上运行。它使用 EmbeddedICE 逻辑并通过 DCC 与主机进行通信。有关 RMTarget 功能的详细信息见 *RealMonitor 目标集成指南（ARM DUI 0142A）*。

33.3.2 RealMonitor是如何工作的

通常情况下，RealMonitor作为一个状态机，如图 33.2所示。为了响应主机接收到的包，或者由于目标板上的异步事件，RealMonitor在运行和停止状态之间进行切换。RMTarget一次只支持一个断点、观察点、停止或半主机SWI的触发。不提供嵌套事件的保存和恢复。因此，如果用户应用程序因为一个断点而停止，而在IRQ处理程序中产生了另一个断点，那么RealMonitor进入应急状态（panic state）。RealMonitor进入该状态后将不执行任何调试。

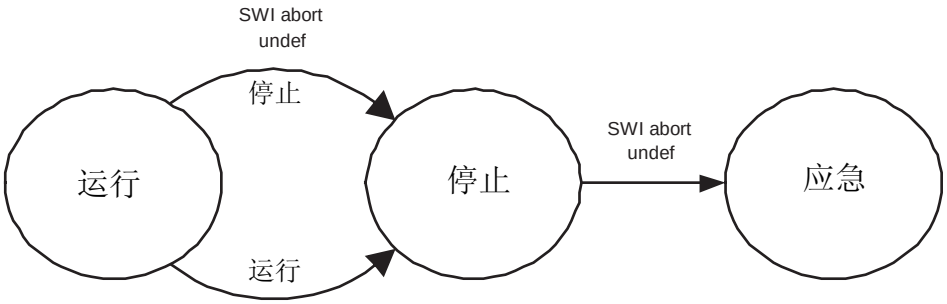


图 33.2 作为状态机的 RealMonitor

一个运行在主计算机上的调试器，例如ARM eXtended调试 器（AXD）或其它 RealMonitor 调试器可以连接到目标，实现命令的发送和数据的接收。主机和目标之间的通信如图 33.1所示。

RealMonitor 的目标部件 RMTarget 与主机部件 RMHost 通过调试通信通道（DCC）进行通信。DCC 是一个可靠的链路，它通过 JTAG 连接来传递数据。

当用户应用程序正在运行时，RMTarget 通常使用 DCC 所产生的 IRQ。这意味着，如果用户应用程序也打算使用 IRQ，那么它必须将 DCC 产生的中断传递给 RealMonitor。

为了实现不间断的调试，处理器中的 EmbeddedICE-RT 逻辑在到达一个断点时产生一个预取指中止异常或在接收到观察点时产生一个数据中止异常。这些异常由 RealMonitor 异常处理程序进行处理，并由调试器将事件告知用户。这样用户应用程序可以在不停止处理器的情况下继续运行。RealMonitor 认为用户应用程序包含下面两部分：

- 一个连续运行的前台应用程序，通常处于用户、系统或 SVC 模式。
- 一个后台应用程序，包含用户系统中特定事件触发的中断和异常处理程序，包括：
  - －IRQ 或 FIQ。
  - －用户前台应用程序产生的数据和预取指中止，表示正在调试的应用程序有错。这两种情况都会通知主机并停止用户应用程序。
  - －由用户前台程序中的未定义指令所导致的未定义异常，表示在调试程序时出现错误。RealMonitor 停止用户程序直至从主机接收到一个“Go”包为止。

当一个不是由用户应用程序处理的异常发生时，执行下面的操作：

- RealMonitor 进入查询 DCC 的一个循环。如果 DCC 读缓冲区已满，则控制权传递给 rm_ReceiveData()（RealMonitor 内部函数）。如果 DCC 写缓冲区空闲，则控制权传递给 rm_TransmitData()（RealMonitor 内部函数）。如果没有别的事要做，函数返回调用程序。上述比较的顺序使读 DCC 的优先级高于写通信链路。
- RealMonitor 停止前台应用程序。如果 IRQ 和 FIQ 在前台程序停止时被应用使能，那么 IRQ 和 FIQ 可以继续被服务。

33.4 如何使能RealMonitor

必须执行下列步骤来使能 RealMonitor。在这一节的末尾给出了执行所有步骤的例程。

33.4.1 增加堆栈

用户必须确保在 RealMonitor 所使用的每一个处理器模式下的应用程序中都建立了堆栈。对于每一种模式，RealMonitor 都要求一个固定字数目的堆栈空间。用户必须为 RealMonitor 和应用程序提供足够的堆栈空间。

RealMonitor 对堆栈有以下要求：

表 33.1 RealMonitor 的堆栈要求

处理器模式	RealMonitor 堆栈使用（字节）
未定义	48
预取指中止	16
数据中止	16
IRQ	8

33.4.2 IRQ模式

该模式下的堆栈是必不可少的。RealMonitor 用两个字保存中断处理程序的入口。在嵌套中断使能前将它们释放。

33.4.3 未定义模式

该模式的堆栈也是必要的。RealMonitor 在处理一个未定义指令异常时使用 12 个字。

33.4.4 SVC模式

RealMonitor 不使用该堆栈。

33.4.5 预取指中止模式

RealMonitor 使用 4 个字保存预取指中止中断处理程序的入口。

33.4.6 数据中止模式

RealMonitor 使用 4 个字保存数据中止中断处理程序的入口。

33.4.7 用户/系统模式

RealMonitor 不使用该堆栈。

33.4.8 FIQ模式

RealMonitor 不使用该堆栈。

33.4.9 处理异常

这一节描述了 RealMonitor 和用户应用程序共用异常处理程序的重要性。

33.4.9.1 RealMonitor异常处理

为了正常工作，RealMonitor必须能够截获特定的中断和异常。图 33.3所示为如何由RealMonitor自身声明异常或与应用程序共用异常。如果用户应用程序要求共用异常，那么它必须提供函数（例如app_IRQDispatch()）。根据异常的特性，该处理程序可以：

- 将控制权传递给 RealMonitor 处理程序，例如 rm_irqhandler2()。

- 为应用程序自身声明异常，例如 app_IRQHandler()。

一个自身不带异常处理程序的应用程序可以安装 RealMonitor 低级异常处理程序，该程序直接指向处理器的向量表。irq 处理程序必须得到向量中断控制器的地址。最简单的方法是在向量表中写一条转移指令，转移指令的目标地址是相应 RealMonitor 异常处理程序的起始地址。

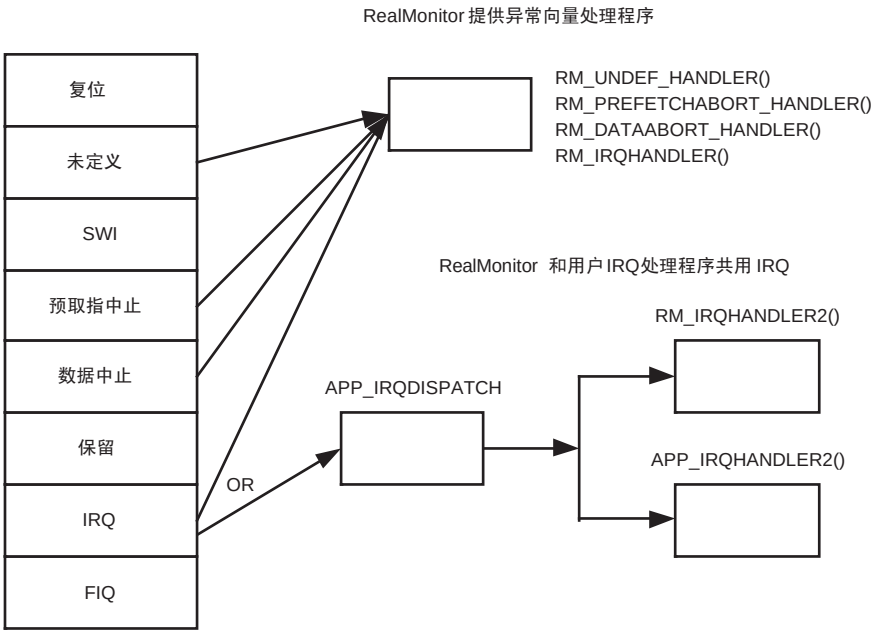


图 33.3 异常处理程序

33.4.10 RMTarget初始化

当处理器处于特权模式并且 IRQ 禁止时，用户必须在应用程序的初始化代码中加入一行代码来调用 rm_init_entry()。

33.4.11 例程

下面的例子显示了如何建立堆栈、VIC、初始化 RealMonitor 以及共用非向量中断：

```
IMPORT rm_init_entry
IMPORT rm_prefetchabort_handler
IMPORT rm_dataabort_handler
IMPORT rm_irqhandler2
IMPORT rm_undef_handler
IMPORT User_Entry；用户应用程序的入口
CODE32
ENTRY
；定义向量表。指令连接器将代码放置到地址 0x0000 0000


AREA exception_table, CODE
```

```
        LDR pc, Reset_Address
        LDR pc, Undefined_Address
        LDR pc, SWI_Address
        LDR pc, Prefetch_Address
        LDR pc, Abort_Address
        NOP                         ; 在此处插入用户代码有效签名
        LDR pc, [pc, # -0x120]      ; 从 VIC 装载 IRQ 向量
        LDR PC, FIQ_Address

Reset_Address DCD __init                        ;复位入口
Undefined_Address DCD rm_undef_handler          ;由 RealMonitor 提供
SWI_Address DCD 0                               ;用户可将 SWI 处理程序的地址放置在此处
Prefetch_Address DCD rm_prefetchabort_handler   ;由 RealMonitor 提供
Abort_Address DCD rm_dataabort_handler          ;由 RealMonitor 提供
FIQ_Address DCD 0                               ;用户可将 FIQ 处理程序的地址放置在此处

AREA init_code, CODE

ram_end EQU 0x4000xxxx                          ; 片内 RAM 的顶端.
__init
; /*********************************************************************
; *为不同的处理模式建立堆栈指针。堆栈向下增加。
; *********************************************************************/
        LDR r2, =ram_end            ; 获得 RAM 的顶端地址
        MRS r0, CPSR                ; 保存当前处理器模式

        ; 初始化未定义模式堆栈，供 RealMonitor 使用
        BIC r1, r0, #0x1f
        ORR r1, r1, #0x1b
        MSR CPSR_c, r1
        ; 为 Falsh 编程程序保留顶端 32 字节，请参考 Flash 存储器系统和编程一章的内容。
        SUB sp,r2,#0x1F

        ; 初始化中止模式堆栈，供 RealMonitor 使用。
        BIC r1, r0, #0x1f
```

```
        ORR r1, r1, #0x17
        MSR CPSR_c, r1
        ; 为未定义模式堆栈保留 64 字节。
        SUB sp,r2,#0x5F

        ; 初始化 IRQ 模式堆栈，供 RealMonitor 和用户程序使用
        BIC r1, r0, #0x1f
        ORR r1, r1, #0x12
        MSR CPSR_c, r1
        ; 为中止模式堆栈保留 32 字节。
        SUB sp,r2,#0x7F

        ; 返回初始模式。
        MSR CPSR_c, r0

        ; 初始化用户应用程序堆栈。
        ; 为 IRQ 模式保留 256 字节。
        SUB sp,r2,#0x17F
; /*********************************************************************
; *  建立向量中断控制器。DCC Rx 和 Tx 中断产生非向量 IRQ 请求。
; * rm_init_entry 可察觉 VIC 并使能 DBGCommRX 和 DBGCommTx 中断。
; *  默认的向量地址寄存器编程为本例上面提到的非向量 app_irqDispatch 的地址。
; *  用户可在此处建立向量 IRQ 或 FIQ。
; *********************************************************************/
        VICBaseAddr EQU 0xFFFFF000          ; VIC 基地址
        VICDefVectAddrOffset EQU 0x34

        LDR r0, =VICBaseAddr
        LDR r1, =app_irqDispatch
        STR r1, [r0,#VICDefVectAddrOffset]

        BL rm_init_entry                    ; 初始化 RealMonitor
        ; 使能 ARM 处理器中的 FIQ 和 IRQ。
        MRS r1, CPSR                        ; 获取 CPSR
        BIC r1, r1, #0xC0                   ; 使能 IRQ 和 FIQ
```

```
        MSR CPSR_c, r1                          ; 更新 CPSR
; /*********************************************************************
; * 获取用户程序入口地址。
; *********************************************************************/
        LDR lr, =User_Entry
        MOV pc, lr
; /*********************************************************************
; * 非向量 irq 处理程序 (app_irqDispatch)
; *********************************************************************/

AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch

        ; 使能中断嵌套
        STMFD sp!, {r12,r14}
        MRS r12, spsr                   ; 将 SPSR 保存到 r12
        MSR cpsr_c,0x1F                 ; 重新使能 IRQ，进入系统模式

; 如果要求共用非向量中断，用户应当在此处插入代码。每个非向量共用 irq 处理程序都必须使用下
; 列代码返回被中断的指令处。
;       MSR cpsr_c, #0x52               ; 禁止 irq，进入 IRQ 模式
;       MSR spsr, r12                   ; 从 r12 恢复 SPSR
;       STMFD sp!, {r0}
;       LDR r0, =VICBaseAddr
;       STR r1, [r0,#VICVectAddrOffset] ; 应答。非向量 irq 已经执行完毕。
;       LDMFD sp!, {r12,r14,r0}         ; 恢复寄存器。
;       SUBS pc, r14, #4                ; 返回到被中断的指令。

        ; 用户中断没有发生，因此调用 rm_irqhandler2。该处理程序没有察觉到 VIC 中断优先级硬件，.
        ; 因此使 rm_irqhandler2 返回到此处。

        STMFD sp!, {ip,pc}
        LDR pc, rm_irqhandler2
        ;rm_irqhandler2 返回到此处。
```

```
MSR cpsr_c, #0x52                ; 禁止 irq，进入 IRQ 模式
MSR spsr, r12                    ; 将 SPSR 从 r12 恢复
STMFD sp!, {r0}
LDR r0, =VICBaseAddr
STR r1, [r0,#VICVectAddrOffset]  ; 应答。非向量 irq 已经执行完毕。
LDMFD sp!, {r12,r14,r0}          ; 恢复寄存器。
SUBS pc, r14, #4                 ; 返回到被中断的指令。
END
```

33.5 RealMonitor建立选项

RealMonitor 使用下列选项建立：

RM_OPT_DATALOGGING=FALSE

该选项使能或禁止在非 RealMonitor（第三方）通道上发送任何从目标到主机的包。

RM_OPT_STOPSTART=TRUE

该选项使能或禁止对所有停止和启动调试特性的支持。

RM_OPT_SOFTBREAKPOINT=TRUE

该选项使能或禁止对软件断点的支持。

RM_OPT_HARDBREAKPOINT=TRUE

在带有 EmbeddedICE-RT 的内核上使能。该器件使用带有 EmbeddedICE-RT 的 ARM-7TDMI-S Rev 4 内核。

RM_OPT_HARDWATCHPOINT=TRUE

在带有 EmbeddedICE-RT 的内核上使能。该器件使用带有 EmbeddedICE-RT 的 ARM-7TDMI-S Rev 4 内核。

RM_OPT_SEMIHOSTING=FALSE

该选项使能或禁止对 SWI 半主机 (semi-hosting)的支持。Semi-hosting 提供运行在 ARM 目标板上的代码，这些代码具有运行 ARM 调试器的主机的一些功能。这些功能包括键盘输入、屏幕输出和磁盘 I/O 等等。

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

当 RealMonitor 停止时，该选项决定是否将 FIQ 模式寄存器保存到寄存器块。

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

使能或禁止对 8/16/32 位读/写的支持。

RM_OPT_EXECUTECODE=FALSE

使能或禁止对“执行代码”缓冲区的执行代码的支持。该代码必须先下载。

RM_OPT_GETPC=TRUE

该选项使能或禁止对 RealMonitor GetPC 包的支持。当中断模式中使用了 RealMonitor 时，它可用于代码剖析。

RM_EXECUTECODE_SIZE=NA

“执行代码”缓冲器大小。参见 RM_OPT_EXECUTECODE 选项。

RM_OPT_GATHER_STATISTICS=FALSE

该选项使能或禁止收集有关 RealMonitor 内部操作统计的代码。

RM_DEBUG=FALSE

该选项使能或禁止在 RealMonitor 中额外的调试和错误检测代码。

RM_OPT_BUILDIDENTIFIER=FALSE

该选项决定是否将一个“建立标识符”建立到 RMTarget 的性能表中。性能表保存在 ROM 中。

RM_OPT_SDM_INFO=FALSE

SDM 向调试工具提供关于应用板和处理器的额外信息。

RM_OPT_MEMORYMAP=FALSE

该选项决定板的存储器映射是否建立到目标当中，并通过性能表得到。

RM_OPT_USE_INTERRUPTS=TRUE

该选项指定是否为中断驱动模式或查询模式建立 RMTarget。

RM_FIFOSIZE=NA

该选项指定数据记录 FIFO 缓冲区的大小（以字为单位）。

CHAIN_VECTORS=FALSE

该选项允许 RMTarget 通过 µHAL（ARM 硬件抽象 API）支持向量链。

第34章 补充信息

34.1 缩写

表 34.1 缩写

首字母缩写	描述
ADC	模数转换器
AHB	先进的高性能总线
AMBA	先进的微控制器总线结构
APB	先进的外设总线
BLS	字节定位选择（Byte Lane select）
BOD	掉电检测
CAN	控制器局域网
DAC	数模转换器
DCC	调试通信通道
DMA	直接存储器存取
DSP	数字信号处理
EOP	数据包结束
ETM	嵌入式跟踪宏单元
GPIO	通用输入/输出
JTAG	联合测试行动组
MII	媒体独立接口
PHY	物理层
PLL	锁相环
PWM	脉宽调制器
RMII	简化的媒体独立接口
SE0	单端 0
SPI	串行外设接口
SSI	同步串行接口
SSP	同步串行端口
TTL	晶体管-晶体管逻辑
UART	通用异步接收器/发送器
USB	通用串行总线

附录A 版本信息

修订版本	修订日期	描述
Rev 1.0	2007 年 2 月	原始版本