嵌入式系统原理

The Principle of Embedded System

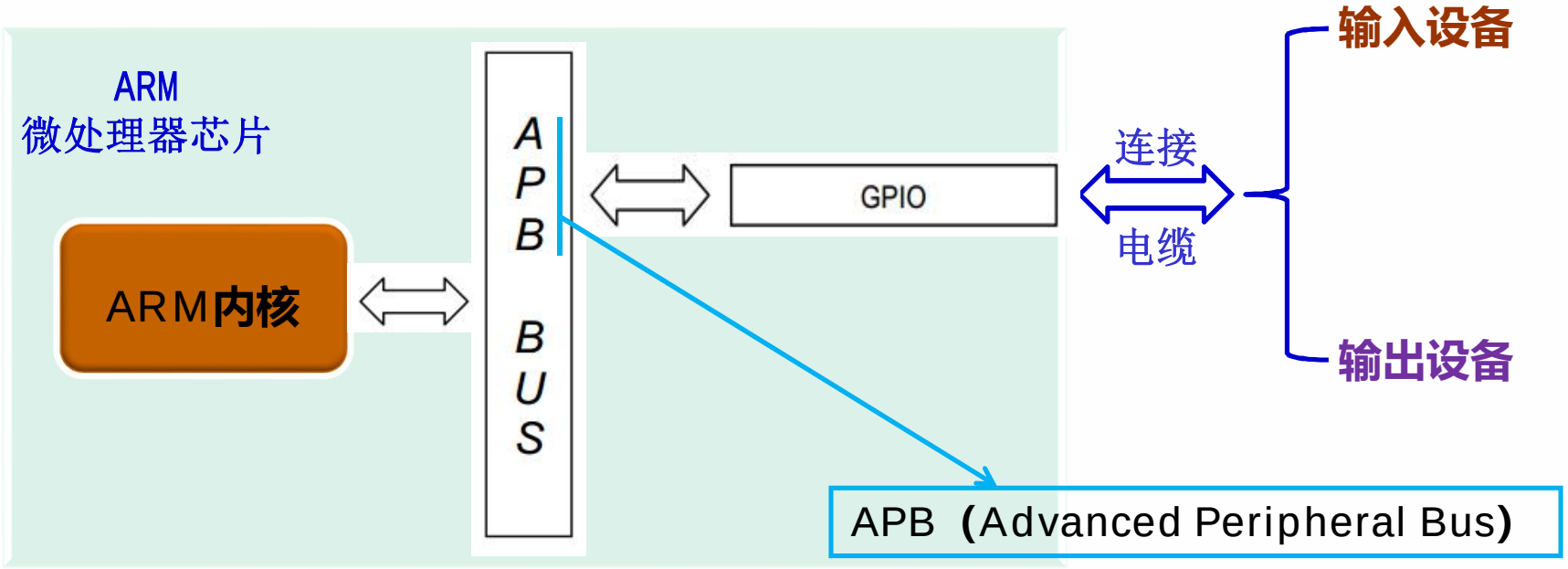
合肥工业大学·计算机与信息学院

2024年3月

- 5.1 通用输入输出端口(GPIO)
- 5.2 中断系统
- 5.3 定时部件
- 5.4 通用异步收发器(UART)*

用途

- 通用输入输出端口（General Purpose I/O，GPIO）是嵌入式系统的重要组成部分。
- 用于连接各类输入、输出设备，实现其与微处理器之间的数据传输。



构成

以S3C2440为例

- 130个GPIO引脚，分为9组：GPA～GPJ。
  - 端口 A（GPA）：23位输出端口
  - 端口 B（GPB）：11 位输入/输出端口
  - 端口 C（GPC）：16 位输入/输出端口
  - 端口 D（GPD）：16 位输入/输出端口
  - 端口 E（GPE）：16 位输入/输出端口
  - 端口 F（GPF）：8 位输入/输出端口
  - 端口 G（GPG）：16 位输入/输出端口
  - 端口 H（GPH）：11位输入/输出端口
  - 端口 J（GPJ）：13 位输入/输出端口
- 每个端口具有多种功能，需要在主程序运行之前，借助对应的控制寄存器进行编程设置。
- 若某个GPIO引脚不用于特定功能，则可以将其设置为普通的输入输出引脚。

端口A（GPA）

- 23个引脚。
- 两种功能：
  - ✓ 普通输出口；
  - ✓ 用于输出外接主存储器的地址信号和存储块的选择信号，以及外接NAND Flash存储器的控制信号。

端口A（GPA）

- 23个引脚。

引脚名称	功能1	功能2
GPA22	输出	nFCE
GPA21	输出	nRSTOUT
GPA20	输出	nFRE
GPA19	输出	nFWE
GPA18	输出	ALE
GPA17	输出	CLE
GPA16	输出	nGCS5
GPA15	输出	nGCS4
GPA14	输出	nGCS3
GPA13	输出	nGCS2
GPA12	输出	nGCS1

引脚名称	功能1	功能2
GPA11	输出	ADDR26
GPA10	输出	ADDR25
GPA9	输出	ADDR24
GPA8	输出	ADDR23
GPA7	输出	ADDR22
GPA6	输出	ADDR21
GPA5	输出	ADDR20
GPA4	输出	ADDR19
GPA3	输出	ADDR18
GPA2	输出	ADDR17
GPA1	输出	ADDR16
GPA0	输出	ADDR0

端口B（GPB）

- 11个引脚。
- 两种功能：
  - 普通输入/输出口；
  - 用于DMA和总线请求信号和应答信号以及各种时钟信号。

端口B（GPB）

- 11个引脚。

引脚名称	功能1	功能2
GPB10	输入/输出	nXDREQ0
GPB9	输入/输出	nXDACK0
GPB8	输入/输出	nXDREQ1
GPB7	输入/输出	nXDACK1
GPB6	输入/输出	nXBREQ
GPB5	输入/输出	nXBACK
GPB4	输入/输出	TCLK0
GPB3	输入/输出	TOUT3
GPB2	输入/输出	TOUT2
GPB1	输入/输出	TOUT1
GPB0	输入/输出	TOUT0

端口C（GPC）

- 16个引脚。
- 两种功能：
  - 普通输入/输出口；
  - 用于LCD显示器的数据信号和控制信号。

端口C（GPC）

- 16个引脚。

引脚名称	功能1	功能2
GPC15	输入/输出	VD7
GPC14	输入/输出	VD6
GPC13	输入/输出	VD5
GPC12	输入/输出	VD4
GPC11	输入/输出	VD3
GPC10	输入/输出	VD2
GPC9	输入/输出	VD1
GPC8	输入/输出	VD0

引脚名称	功能1	功能2
GPC7	输入/输出	LCD_LPCREVB
GPC6	输入/输出	LCD_LPCREV
GPC5	输入/输出	LCD_LPCOE
GPC4	输入/输出	VM
GPC3	输入/输出	VFRAME
GPC2	输入/输出	VLINE
GPC1	输入/输出	VCLK
GPC0	输入/输出	LEND

端口D（GPD）

- 16个引脚。
- 三种功能：
  - 普通输入/输出口；
  - 用于LCD显示器的数据信号。
  - 用于SPI总线的控制信号。

端口D（GPD）

- 16个引脚。

引脚名称	功能1	功能2	功能3
GPD15	输入/输出	VD23	nSS0
GPD14	输入/输出	VD22	nSS1
GPD13	输入/输出	VD21	—
GPD12	输入/输出	VD20	—
GPD11	输入/输出	VD19	—
GPD10	输入/输出	VD18	SPICLK1
GPD9	输入/输出	VD17	SPIMOSI1
GPD8	输入/输出	VD16	SPIMISO1

引脚名称	功能1	功能2	功能3
GPD7	输入/输出	VD15	—
GPD6	输入/输出	VD14	—
GPD5	输入/输出	VD13	—
GPD4	输入/输出	VD12	—
GPD3	输入/输出	VD11	—
GPD2	输入/输出	VD10	—
GPD1	输入/输出	VD9	—
GPD0	输入/输出	VD8	—

端口E（GPE）

- 16个引脚。
- 三种功能：
  - 普通输入/输出口；
  - 用于SPI、I^2C、I^2S总线和SD存储器的控制信号。
  - 用于AC’97音频接口的控制信号。

端口E（GPE）

- 16个引脚。

引脚名称	功能1	功能2	功能3
GPE15	输入/输出	IICSDA	—
GPE14	输入/输出	IICSCL	—
GPE13	输入/输出	SPICLK0	—
GPE12	输入/输出	SPIMOSI0	—
GPE11	输入/输出	SPIMISO0	—
GPE10	输入/输出	SDDAT3	—
GPE9	输入/输出	SDDAT2	—
GPE8	输入/输出	SDDAT1	—
GPE7	输入/输出	SDDAT0	—

引脚名称	功能1	功能2	功能3
GPE6	输入/输出	SDCMD	—
GPE5	输入/输出	SDCLK	—
GPE4	输入/输出	I2SSDO	AC_SDATA_OUT
GPE3	输入/输出	I2SSDI	AC_SDATA_IN
GPE2	输入/输出	CDCLK	AC_nRESET
GPE1	输入/输出	I2SSCLK	AC_BIT_CLK
GPE0	输入/输出	I2SLRCK	AC_SYNC

端口F（GPF）

- 8个引脚。
- 两种功能：
  - 普通输入/输出口；
  - 用于中断请求信号。——常用

端口F（GPF）

- 8个引脚。

引脚名称	功能1	功能2
GPF7	输入/输出	EINT7
GPF6	输入/输出	EINT6
GPF5	输入/输出	EINT5
GPF4	输入/输出	EINT4
GPF3	输入/输出	EINT3
GPF2	输入/输出	EINT2
GPF1	输入/输出	EINT1
GPF0	输入/输出	EINT0

端口G（GPG）

- 16个引脚。
- 三种功能：
  - 普通输入/输出口；
  - 用于中断请求信号。
  - 用于SPI总线、LCD显示器的控制信号。

端口G（GPG）

- 16个引脚。

引脚名称	功能1	功能2	功能3
GPG15	输入/输出	EINT23	—
GPG14	输入/输出	EINT22	—
GPG13	输入/输出	EINT21	—
GPG12	输入/输出	EINT20	—
GPG11	输入/输出	EINT19	TCLK1
GPG10	输入/输出	EINT18	nCTS1
GPG9	输入/输出	EINT17	nRTS1
GPG8	输入/输出	EINT16	—

引脚名称	功能1	功能2	功能3
GPG7	输入/输出	EINT15	SPICLK1
GPG6	输入/输出	EINT14	SPIMOSI1
GPG5	输入/输出	EINT13	SPIMISO1
GPG4	输入/输出	EINT12	LCD_PWREN
GPG3	输入/输出	EINT11	nSS1
GPG2	输入/输出	EINT10	nSS0
GPG1	输入/输出	EINT9	—
GPG0	输入/输出	EINT8	—

端口H（GPH）

- 11个引脚。
- 三种功能：
  - 普通输入/输出口；
  - 用于异步串行接口UART的控制信号；
  - 用于异步串行接口UART的控制信号。（仅两个）

端口H（GPH）

- 11个引脚。

引脚名称	功能1	功能2	功能3
GPH10	输入/输出	CLKOUT1	—
GPH9	输入/输出	CLKOUT0	—
GPH8	输入/输出	UEXTCLK	—
GPH7	输入/输出	RXD2	nCTS1
GPH6	输入/输出	TXD2	nRTS1
GPH5	输入/输出	RXD1	—
GPH4	输入/输出	TXD1	—
GPH3	输入/输出	RXD0	—
GPH2	输入/输出	TXD0	—
GPH1	输入/输出	nRTS0	—
GPH0	输入/输出	nCTS0	—

端口J（GPJ）

- 13个引脚。
- 两种功能：
  - 普通输入/输出口；
  - 用于摄像头接口的数据信号和控制信号。

端口J（GPJ）

- 13个引脚。

引脚名称	功能1	功能2
GPJ12	输入/输出	CAMRESET
GPJ11	输入/输出	CAMCLKOUT
GPJ10	输入/输出	CAMHREF
GPJ9	输入/输出	CAMVSYNC
GPJ8	输入/输出	CAMPCLK
GPJ7	输入/输出	CAMDATA7
GPJ6	输入/输出	CAMDATA6
GPJ5	输入/输出	CAMDATA5
GPJ4	输入/输出	CAMDATA4
GPJ3	输入/输出	CAMDATA3
GPJ2	输入/输出	CAMDATA2
GPJ1	输入/输出	CAMDATA1
GPJ0	输入/输出	CAMDATA0

I/O端口的使用

- 使用每个端口对应的控制寄存器来编程设定。
  - ✓ 每个I/O引脚的功能；
  - ✓ I/O端口的状态（如：输入or输出、数据线是否挂起）。

1. 端口配置寄存器GPACON - GPJCON
2. 端口数据寄存器GPADAT - GPJDAT
3. 端口上拉寄存器GPBUP - GPJUP
4. 驱动能力控制寄存器DSC0-DSC1

端口A的控制寄存器（2个）

寄存器	地址	R/W	描述	复位值
GPACON	0x56000000	R/W	配置端口 A 的引脚	0xFFFFFF
GPADAT	0x56000004	R/W	端口 A 的数据寄存器	-
保留	0x56000008	-	保留	-
保留	0x5600000C	-	保留	-

➢ 端口A的配置寄存器——GPACON

名称	位索引	描述	初始值
GPA22	[22]	0 =输出； 1 = nFCE。	1
GPA21	[21]	0 =输出； 1 = nRSTOUT。	1
GPA20	[20]	0 =输出； 1 = nFRE。	1
GPA19	[19]	0 =输出； 1 = nFWE。	1
GPA18	[18]	0 =输出； 1 = ALE。	1
GPA17	[17]	0 =输出； 1 = CLE。	1
GPA16	[16]	0 =输出； 1 = nGCS5。	1
GPA15	[15]	0 =输出； 1 = nGCS4。	1
GPA14	[14]	0 =输出； 1 = nGCS3。	1
GPA13	[13]	0 =输出； 1 = nGCS2。	1
GPA12	[12]	0 =输出； 1 = nGCS1。	1
GPA11	[11]	0 =输出； 1 = ADDR26。	1

注：nRSTOUT = nRESET & nWDTRST & SW_RESET

➢ 端口A的配置寄存器——GPACON（续）

名称	位索引	描述	初始值
GPA10	[10]	0 =输出； 1 = ADDR25。	1
GPA9	[9]	0 =输出； 1 = ADDR24。	1
GPA8	[8]	0 =输出； 1 = ADDR23。	1
GPA7	[7]	0 =输出； 1 = ADDR22。	1
GPA6	[6]	0 =输出； 1 = ADDR21。	1
GPA5	[5]	0 =输出； 1 = ADDR20。	1
GPA4	[4]	0 =输出； 1 = ADDR19。	1
GPA3	[3]	0 =输出； 1 = ADDR18。	1
GPA2	[2]	0 =输出； 1 = ADDR17。	1
GPA1	[1]	0 =输出； 1 = ADDR16。	1
GPA0	[0]	0 =输出； 1 = ADDR0。	1

➢ 端口A的数据寄存器——GPADAT

名称	位索引	描述	初始值
GPA22～GPA0	[22:0]	存放端口A的数据。 “1”-输出高电平；“0”-输出低电平。	—

端口B的控制寄存器（3个）

寄存器	地址	R/W	描述	复位值
GPBCON	0x56000010	R/W	配置端口 B 的引脚	0x0
GPBDAT	0x56000014	R/W	端口 B 的数据寄存器	-
GPBUP	0x56000018	R/W	端口 B 的上拉使能寄存器	0x0
保留	0x5600001C	-	保留	-

➢ 端口B的配置寄存器——GPBCON

名称	位索引	描述	初始值
GPB10	[21:20]	00=输入； 01=输出； 10=nXDREQ0； 11=保留。	0
GPB9	[19:18]	00=输入； 01=输出； 10=nXDACK0； 11=保留。	0
GPB8	[17:16]	00=输入； 01=输出； 10=nXDREQ1； 11=保留。	0
GPB7	[15:14]	00=输入； 01=输出； 10=nXDACK1； 11=保留。	0
GPB6	[13:12]	00=输入； 01=输出； 10=nXBREQ； 11=保留。	0
GPB5	[11:10]	00=输入； 01=输出； 10=nXBACK； 11=保留。	0
GPB4	[9:8]	00=输入； 01=输出； 10=TCLK0； 11=保留。	0
GPB3	[7:6]	00=输入； 01=输出； 10=TOUT3； 11=保留。	0
GPB2	[5:4]	00=输入； 01=输出； 10=TOUT2； 11=保留。	0
GPB1	[3:2]	00=输入； 01=输出； 10=TOUT1； 11=保留。	0
GPB0	[1:0]	00=输入； 01=输出； 10=TOUT0； 11=保留。	0

➢ 端口B的数据寄存器——GPBDAT

名称	位索引	描述	初始值
GPB10～GPB0	[10:0]	存放端口B的数据。 ● 当配置为输入端口时， 相应位为引脚状态。 ● 当配置为输出端口时， 引脚输出相应位。 ● 当配置为功能引脚， 将读取到未定义值。	—

➢ 端口B的上拉使能寄存器——GPBUP

名称	位索引	描述	初始值
GPB[10:0]	[10:0]	0：设置对应的端口引脚具有上拉功能； 1：设置对应的端口引脚没有上拉功能。	0

★用途：简化嵌入式系统的硬件电路设计，对于需要上拉电路的IO端口，无需设计上拉电阻。

端口C～J的控制寄存器（7×3个）

- 与端口B的控制寄存器类似，区别在于端口的引脚个数不同，而且部分端口有第三功能。
- 详见《S3C2440用户手册》。

端口 C 控制寄存器（GPCCON，GPCDAT，GPCUP）

寄存器	地址	R/W	描述	复位值
GPCCON	0x56000020	R/W	配置端口 C 的引脚	0x0
GPCDAT	0x56000024	R/W	端口 C 的数据寄存器	–
GPCUP	0x56000028	R/W	端口 C 的上拉使能寄存器	0x0
保留	0x5600002C	–	保留	–

端口 D 控制寄存器（GPDCON，GPDDAT，GPDUP）

寄存器	地址	R/W	描述	复位值
GPDCON	0x56000030	R/W	配置端口 D 的引脚	0x0
GPDDAT	0x56000034	R/W	端口 D 的数据寄存器	–
GPDUP	0x56000038	R/W	端口 D 的上拉使能寄存器	0xF000
保留	0x5600003C	–	保留	–

端口 E 控制寄存器（GPECON，GPEDAT，GPEUP）

寄存器	地址	R/W	描述	复位值
GPECON	0x56000040	R/W	配置端口 E 的引脚	0x0
GPEDAT	0x56000044	R/W	端口 E 的数据寄存器	–
GPEUP	0x56000048	R/W	端口 E 的上拉使能寄存器	0x0
保留	0x5600004C	–	保留	–

端口 F 控制寄存器（GPFCON，GPFDAT，GPFUP）

如果 GPF0 至 GPF7 在掉电模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPFCON	0x56000050	R/W	配置端口 F 的引脚	0x0
GPFDAT	0x56000054	R/W	端口 F 的数据寄存器	–
GPFUP	0x56000058	R/W	端口 F 的上拉使能寄存器	0x00
保留	0x5600005C	–	保留	–

端口 G 控制寄存器（GPGCON，GPGDAT，GPGUP）

如果 GPG0 至 GPG7 在睡眠模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPGCON	0x56000060	R/W	配置端口 G 的引脚	0x0
GPGDAT	0x56000064	R/W	端口 G 的数据寄存器	–
GPGUP	0x56000068	R/W	端口 G 的上拉使能寄存器	0xFC00
保留	0x5600006C	–	保留	–

端口 H 控制寄存器（GPHCON，GPHDAT，GPHUP）

寄存器	地址	R/W	描述	复位值
GPHCON	0x56000070	R/W	配置端口 H 的引脚	0x0
GPHDAT	0x56000074	R/W	端口 H 的数据寄存器	–
GPHUP	0x56000078	R/W	端口 H 的上拉使能寄存器	0x000
保留	0x5600007C	–	保留	–

端口 J 控制寄存器（GPJCON，GPJDAT，GPJUP）

寄存器	地址	R/W	描述	复位值
GPJCON	0x560000D0	R/W	配置端口 J 的引脚	0x0
GPJDAT	0x560000D4	R/W	端口 J 的数据寄存器	–
GPJUP	0x560000D8	R/W	端口 J 的上拉使能寄存器	0x0000
保留	0x560000DC	–	保留	–

驱动强度控制寄存器（2个）

寄存器	地址	R/W	描述	复位值
DSC0	0x560000C4	R/W	强度控制寄存器 0	0x0
DSC1	0x560000C8	R/W	强度控制寄存器 1	0x0

用途：用于设置ARM微处理器芯片数据总线、地址总线以及部分作为第二/第三功能使用的GPIO引脚的驱动电流大小。

➢ 驱动强度控制寄存器0——DSC0

名称	位索引	描述	初始值
nEN_DSC	[31]	使能驱动强度控制。 0=使能；1=禁止。	0
保留	[30:10]	–	
DSC_ADR	[9:8]	地址总线驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	
DSC_DATA3	[7:6]	DATA[31:24] I/O 驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	
DSC_DATA2	[5:4]	DATA[23:16] I/O 驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	
DSC_DATA1	[3:2]	DATA[15:8] I/O 驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	
DSC_DATA0	[1:0]	DATA[7:0] I/O 驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	

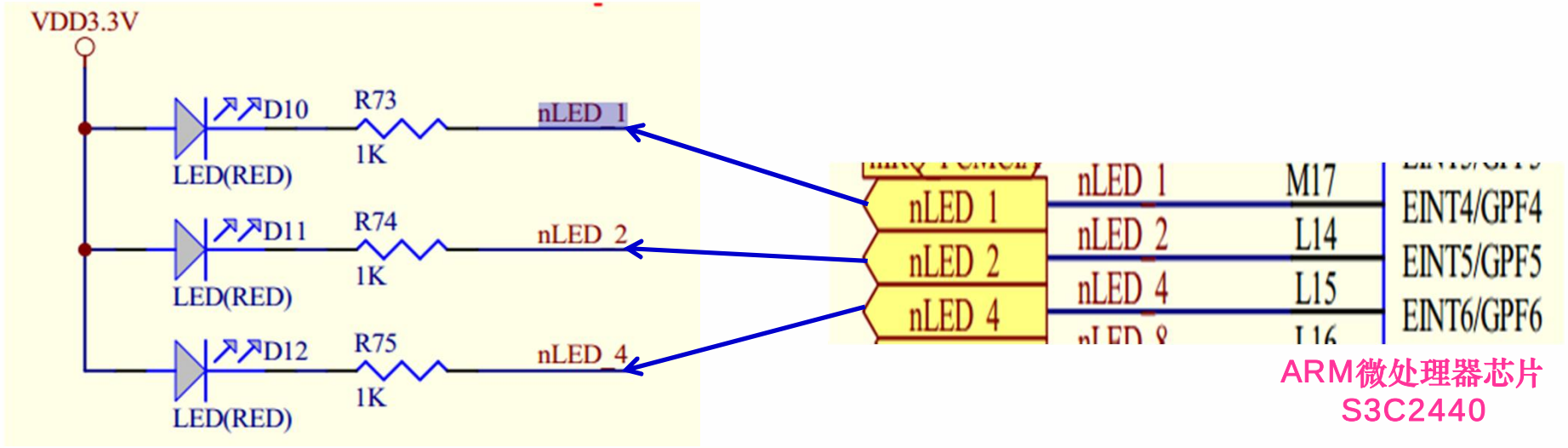
➢ 驱动强度控制寄存器1——DSC1

名称	位索引	描述	初始值
DSC_SCK1	[29:28]	SCLK1驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	00
DSC_SCK0	[27:26]	SCLK0驱动强度。 00=12mA；10=10mA；01=8mA；11= 6mA。	00
DSC_SCKE	[25:24]	SCKE驱动强度。 00=10mA；10=8mA；01=6mA；11= 4mA。	00
DSC_SDR	[23:22]	nSRAS/nSCAS驱动强度。 00=10mA；10=8mA；01=6mA；11= 4mA。	00
DSC_NFC	[21:20]	Nand Flash控制驱动强度(nFCE,nFRE,nFWE,CLE, ALE)。 00=10mA；10=8mA；01=6mA；11= 4mA。	00
DSC_BE	[19:18]	nBE[3:0]驱动强度。 00=10mA；10=8mA；01=6mA；11= 4mA。	00
DSC_WOE	[17:16]	nWE/nOE驱动强度。 00=10mA；10=8mA；01=6mA；11= 4mA。	00
DSC_CSn (n=0～7)	[2n+1:2n]	nGCSn驱动强度。 00=10mA；10=8mA；01=6mA；11= 4mA。	00

举例

- 使用GPIO的F端口点亮发光二极管。

1.硬件原理图



2.端口F寄存器

端口 F 控制寄存器（GPFCON，GPFDAT，GPFUP）

如果 GPF0 至 GPF7 在掉电模式中用于唤醒信号，端口将被设置为中断模式。

寄存器	地址	R/W	描述	复位值
GPFCON	0x56000050	R/W	配置端口 F 的引脚	0x0
GPFDAT	0x56000054	R/W	端口 F 的数据寄存器	–
GPFUP	0x56000058	R/W	端口 F 的上拉使能寄存器	0x00
保留	0x5600005C	–	保留	–

GPFCON	Bit	Description	
GPF7	[15:14]	00 = Input 10 = EINT[7]	01 = Output 11 = Reserved
GPF6	[13:12]	00 = Input 10 = EINT[6]	01 = Output 11 = Reserved
GPF5	[11:10]	00 = Input 10 = EINT[5]	01 = Output 11 = Reserved
GPF4	[9:8]	00 = Input 10 = EINT[4]	01 = Output 11 = Reserved

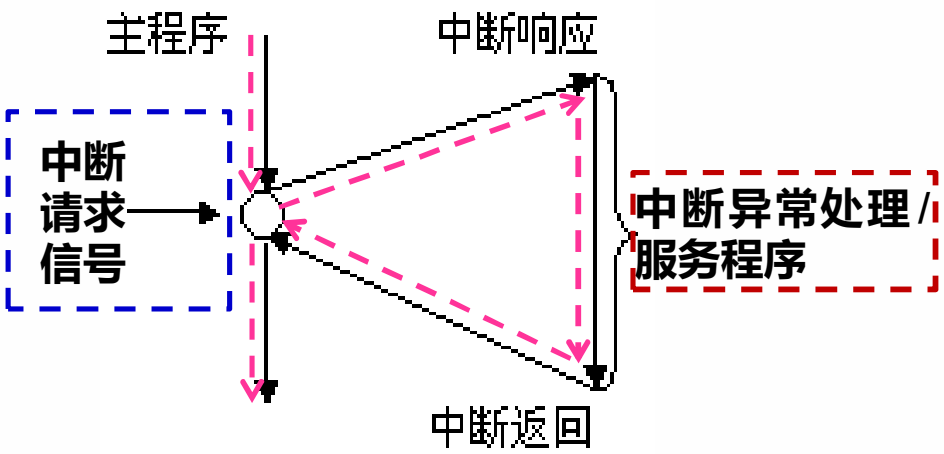
3.汇编代码

```
.text
.global _start
_start:
        LDR R0,=0x56000050          ;端口F的配置寄存器地址
        LDR R1,[R0]
        BIC R1,R1,#0x3F00           ;配置GPF4～6为输出引脚
        ORR R1,R1,#0x1500
        STR R1,[R0]
        LDR R0,=0x56000054          ;端口F的数据寄存器地址
        LDR R1,[R0]
        BIC R1,R1,#0x70             ;GPF4～6输出为低电平
        STR R1,[R0]
WAIT:
        B WAIT
```

- 5.1 通用输入输出端口(GPIO)
- 5.2 中断系统
- 5.3 定时部件
- 5.4 通用异步收发器(UART)*

中断的概念

- 微处理器在执行正常程序的过程中，因某事件发生，收到来自外围部件（设备）的请求信号。
- 若能够响应该信号，则暂停当前程序的正常执行，转去执行针对请求事件的处理操作，待结束后再返回被暂停的程序继续执行。



中断的作用

并行处理

- ✓在外围设备需要传输数据时才产生“中断”，使得微处理器可以与多个外围设备同时工作，提高了微处理器的工作效率。

实时处理

- ✓在实时控制系统中，外围设备提出服务请求的时间是随机的。只有通过中断系统，才能对它们进行快速响应。

故障处理

- ✓系统在运行过程中，常常会出现一些突发性故障，利用中断功能就可以对它们进行实时处理。

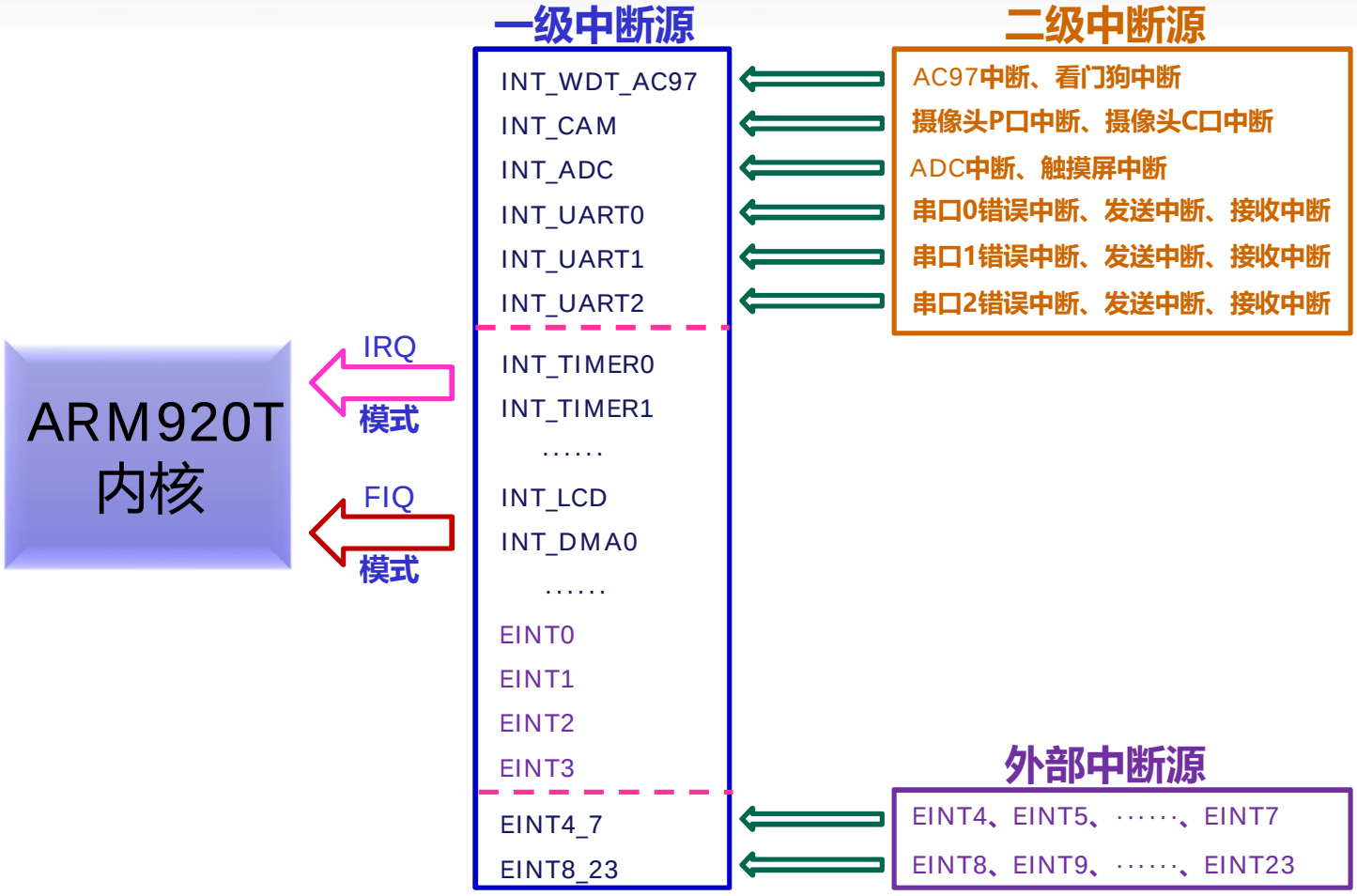
中断系统的构成

以S3C2440为例

- ARM920T微处理器(内核)的7种异常类型中，由芯片外部中断请求引脚和内部外设触发的是外部中断请求（IRQ）和快速中断请求（FIQ）。
- 借助中断控制器，接收并管理60个中断源发出的中断请求信号。

60 个中断源(1 个看门狗 ,5 个定时器，9 个 UART，
24 个外部中断，4 个 DMA，2 个 RTC，2 个 ADC，
1 个 IIC ,2 个 SPI ,1 个 SDI ,2 个 USB ,1 个 LCD ,
1 个电池故障 ,1 个 NAND ,2 个摄像头 ,1 个 AC'97)

ARM920T的中断源



一级中断源 ——32个

- 拥有二级中断源

中断源名称	描　述	仲裁组
INT_ADC	ADC结束中断、触摸屏中断	ARB5
INT_RTC	RTC闹钟中断	ARB5
INT_SPI1	SPI1中断	ARB5
INT_UART0	串口0中断（ERR、RXD、TXD）	ARB5
INT_IIC	I^2C中断	ARB4
INT_USBH	USB主机中断	ARB4
INT_USBD	USB设备中断	ARB4
INT_NFCON	Nand Flash控制中断	ARB4
INT_UART1	串口1中断（ERR、RXD、TXD）	ARB4
INT_SPI0	SPI0中断	ARB4
INT_SDI	SDI中断	ARB3
INT_DMA3	DMA通道3中断	ARB3
INT_DMA2	DMA通道2中断	ARB3
INT_DMA1	DMA通道1中断	ARB3
INT_DMA0	DMA通道0中断	ARB3

对应多个中断源

INT_LCD	LCD中断（INT_FrSyn、INT_FiCnt）(2个）	ARB3
INT_UART2	串口2中断（ERR、RXD、TXD）	ARB2
INT_TIMER4	定时器4中断	ARB2
INT_TIMER3	定时器3中断	ARB2
INT_TIMER2	定时器2中断	ARB2
INT_TIMER1	定时器1中断	ARB2
INT_TIMER0	定时器0中断	ARB2
INT_WDT_AC97	看门狗定时器中断	ARB1
INT_TICK	RTC定时中断	ARB1
nBATT_FLT	电池失效中断	ARB1
INT_CAM	摄像头接口	ARB1
EINT8_23	外部中断8_23（16个）	ARB1
EINT4_7	外部中断4_7（4个）	ARB1
EINT3	外部中断3	ARB0
EINT2	外部中断2	ARB0
EINT1	外部中断1	ARB0
EINT0	外部中断0	ARB0

二级中断源 ——15个

中断源名称	描　述	所属一级中断源
INT_AC97	AC97中断	INT_WDT_AC97
INT_WDT	WDT中断	INT_WDT_AC97
INT_CAM_P	摄像头接口的P口捕获中断	INT_CAM
INT_CAM_C	摄像头接口的C口捕获中断	INT_CAM
INT_ADC_S	ADC中断	INT_ADC
INT_TC	触摸屏中断（开/关）	INT_ADC
INT_ERR2	UART2错误中断	INT_UART2
INT_TXD2	UART2发送中断	INT_UART2
INT_RXD2	UART2接收中断	INT_UART2
INT_ERR1	UART1错误中断	INT_UART1
INT_TXD1	UART1发送中断	INT_UART1
INT_RXD1	UART1接收中断	INT_UART1
INT_ERR0	UART0错误中断	INT_UART0
INT_TXD0	UART0发送中断	INT_UART0
INT_RXD0	UART0接收中断	INT_UART0

中断控制器的功能

1. 外部中断请求信号管理
2. 中断模式设定
3. 中断请求信号标记
4. 中断屏蔽设定
5. 中断优先级管理
6. 中断服务标记

中断管理功能的实现

- 特殊功能寄存器SFR——17个
- SFR的组织

与一级中断源相关的SFR(6个)

- 源挂起寄存器SRCPND
- 中断模式寄存器INTMOD
- 中断屏蔽寄存器INTMSK
- 优先级寄存器PRIORITY
- 中断挂起寄存器INTPND
- 中断偏移寄存器INTOFFSET

与二级中断源相关的SFR(2个)

- 次级源挂起寄存器SUBSRCPND
- 中断次级屏蔽寄存器INTSUBMSK

与外部中断源相关的SFR(9个)

- 外部中断控制寄存器EXTINT0～EXTINT2
- 外部中断滤波寄存器EINTFLT0～EINTFLT3
- 外部中断屏蔽寄存器EINTMASK
- 外部中断挂起寄存器EINTPEND

与一级中断源相关的SFR

一级中断源 ⇒

- 源挂起寄存器SRCPND
- 中断模式寄存器INTMOD
- 中断屏蔽寄存器INTMSK
- 优先级寄存器PRIORITY
- 中断挂起寄存器INTPND
- 中断偏移寄存器INTOFFSET

源挂起寄存器

- 用于标记32个一级中断源的中断请求信号是否触发，即是否有中断等待处理，供中断服务程序读取和判断。
- 每一位都由中断源自动置位，无论该中断源是否被设定位屏蔽，也不受优先级逻辑的影响。
- 为了保证能收到同一中断源的下一次中断请求，通常需要在中断服务程序中人工清除置位。

通过写入数据来清除。写入“1”表示清除，“0”表示保持不变。

寄存器	地址	R/W	描述	复位值
SRCPND	0X4A000000	R/W	指示中断请求状态 0 = 中断未被请求　　1 = 中断源声明了中断请求	0x00000000

与一级中断源相关的SFR

➢ 源挂起寄存器——SRCPND

名称	位索引	描述	初始值
INT_ADC	[31]	确定INT_ADC中断请求。0=没有请求；1=请求。	0
INT_RTC	[30]	确定INT_RTC中断请求。0=没有请求；1=请求。	0
INT_SPI1	[29]	确定INT_SPI1中断请求。0=没有请求；1=请求。	0
INT_UART0	[28]	确定INT_UART0中断请求。0=没有请求；1=请求。	0
INT_IIC	[27]	确定INT_IIC中断请求。0=没有请求；1=请求。	0
INT_USBH	[26]	确定INT_USBH中断请求。0=没有请求；1=请求。	0
INT_USBD	[25]	确定INT_USBD中断请求。0=没有请求；1=请求。	0
INT_NFCON	[24]	确定INT_NFCON中断请求。0=没有请求；1=请求。	0
INT_UART1	[23]	确定INT_UART1中断请求。0=没有请求；1=请求。	0
INT_SPI0	[22]	确定INT_SPI0中断请求。0=没有请求；1=请求。	0
INT_SDI	[21]	确定INT_SDI中断请求。0=没有请求；1=请求。	0
INT_DMA3	[20]	确定INT_DMA3中断请求。0=没有请求；1=请求。	0
INT_DMA2	[19]	确定INT_DMA2中断请求。0=没有请求；1=请求。	0

与一级中断源相关的SFR

➢ 源挂起寄存器——SRCPND（续）

名称	位索引	描述	初始值
INT_DMA1	[18]	确定INT_DMA1中断请求。0=没有请求；1=请求。	0
INT_DMA0	[17]	确定INT_DMA0中断请求。0=没有请求；1=请求。	0
INT_LCD	[16]	确定INT_LCD中断请求。0=没有请求；1=请求。	0
INT_UART2	[15]	确定INT_UART2中断请求。0=没有请求；1=请求。	0
INT_TIMER4	[14]	确定INT_TIMER4中断请求。0=没有请求；1=请求。	0
INT_TIMER3	[13]	确定INT_TIMER3中断请求。0=没有请求；1=请求。	0
INT_TIMER2	[12]	确定INT_TIMER2中断请求。0=没有请求；1=请求。	0
INT_TIMER1	[11]	确定INT_TIMER1中断请求。0=没有请求；1=请求。	0
INT_TIMER0	[10]	确定INT_TIMER0中断请求。0=没有请求；1=请求。	0
INT_WDT_AC97	[9]	确定INT_WDT_AC97中断请求。0=没有请求；1=请求。	0
INT_TICK	[8]	确定INT_TICK中断请求。0=没有请求；1=请求。	0
nBATT_FLT	[7]	确定nBATT_FLT中断请求。0=没有请求；1=请求。	0
INT_CAM	[6]	确定INT_ CAM中断请求。0=没有请求；1=请求。	0

➢ 源挂起寄存器——SRCPND（续）

名称	位索引	描述	初始值
EINT8_23	[5]	确定EINT8_23中断请求。0=没有请求；1=请求。	0
EINT4_7	[4]	确定EINT4_7中断请求。0=没有请求；1=请求。	0
EINT3	[3]	确定EINT3中断请求。0=没有请求；1=请求。	0
EINT2	[2]	确定EINT2中断请求。0=没有请求；1=请求。	0
EINT1	[1]	确定EINT1中断请求。0=没有请求；1=请求。	0
EINT0	[0]	确定EINT0中断请求。0=没有请求；1=请求。	0

与外部中断挂起寄存器(EINTPEND)一同确定EINT4～ENT23中具体哪些信号有效。

中断模式寄存器

- 用于设定32个一级中断源的中断模式。
  - ✓ 若设置为“1”，则相应的中断源按FIQ模式处理；
  - ✓ 若设置为“0”，则相应的中断源按IRQ模式处理。
- 注意：同一个时刻，仅有一个中断源能够在FIQ模式下处理，即仅有一个位可以被置1。因此，应该将最紧迫的中断源设置为FIQ模式。

寄存器	地址	R/W	描述	复位值
INTMOD	0X4A000004	R/W	中断模式寄存器 0 = IRQ 模式　　1 = FIQ 模式	0x00000000

与一级中断源相关的SFR

➢ 中断模式寄存器——INTMOD

名称	位索引	描述	初始值
INT_ADC	[31]	确定INT_ADC中断请求。0=IRQ； 1=FIQ。	0
INT_RTC	[30]	确定INT_RTC中断请求。0=IRQ； 1=FIQ。	0
INT_SPI1	[29]	确定INT_SPI1中断请求。0=IRQ； 1=FIQ。	0
INT_UART0	[28]	确定INT_UART0中断请求。0=IRQ； 1=FIQ。	0
INT_IIC	[27]	确定INT_IIC中断请求。0=IRQ； 1=FIQ。	0
INT_USBH	[26]	确定INT_USBH中断请求。0=IRQ； 1=FIQ。	0
INT_USBD	[25]	确定INT_USBD中断请求。0=IRQ； 1=FIQ。	0
INT_NFCON	[24]	确定INT_NFCON中断请求。0=IRQ； 1=FIQ。	0
INT_UART1	[23]	确定INT_UART1中断请求。0=IRQ； 1=FIQ。	0
INT_SPI0	[22]	确定INT_SPI0中断请求。0=IRQ； 1=FIQ。	0
INT_SDI	[21]	确定INT_SDI中断请求。0=IRQ； 1=FIQ。	0
INT_DMA3	[20]	确定INT_DMA3中断请求。0=IRQ； 1=FIQ。	0
INT_DMA2	[19]	确定INT_DMA2中断请求。0=IRQ； 1=FIQ。	0

中断模式寄存器——INTMOD（续）

名称	位索引	描述	初始值
INT_DMA1	[18]	确定INT_DMA1中断请求。0=IRQ； 1=FIQ。	0
INT_DMA0	[17]	确定INT_DMA0中断请求。0=IRQ； 1=FIQ。	0
INT_LCD	[16]	确定INT_LCD中断请求。0=IRQ； 1=FIQ。	0
INT_UART2	[15]	确定INT_UART2中断请求。0=IRQ； 1=FIQ。	0
INT_TIMER4	[14]	确定INT_TIMER4中断请求。0=IRQ； 1=FIQ。	0
INT_TIMER3	[13]	确定INT_TIMER3中断请求。0=IRQ； 1=FIQ。	0
INT_TIMER2	[12]	确定INT_TIMER2中断请求。0=IRQ； 1=FIQ。	0
INT_TIMER1	[11]	确定INT_TIMER1中断请求。0=IRQ； 1=FIQ。	0
INT_TIMER0	[10]	确定INT_TIMER0中断请求。0=IRQ； 1=FIQ。	0
INT_WDT_AC97	[9]	确定INT_WDT_AC97中断请求。0=IRQ； 1=FIQ。	0
INT_TICK	[8]	确定INT_TICK中断请求。0=IRQ； 1=FIQ。	0
nBATT_FLT	[7]	确定nBATT_FLT中断请求。0=IRQ； 1=FIQ。	0
INT_CAM	[6]	确定INT_ CAM中断请求。0=IRQ； 1=FIQ。	0

➢ 中断模式寄存器——INTMOD（续）

名称	位索引	描述	初始值
EINT8_23	[5]	确定EINT8_23中断请求。0=IRQ； 1=FIQ。	0
EINT4_7	[4]	确定EINT4_7中断请求。0=IRQ； 1=FIQ。	0
EINT3	[3]	确定EINT3中断请求。0=IRQ； 1=FIQ。	0
EINT2	[2]	确定EINT2中断请求。0=IRQ； 1=FIQ。	0
EINT1	[1]	确定EINT1中断请求。0=IRQ； 1=FIQ。	0
EINT0	[0]	确定EINT0中断请求。0=IRQ； 1=FIQ。	0

中断屏蔽寄存器

- 用于设定32个一级中断源是否被允许中断。
  - ✓ 若设置为“1”，则相应的中断源不会被服务；
  - ✓ 若设置为“0”，则相应的中断源可以被服务。

寄存器	地址	R/W	描述	复位值
INTMSK	0X4A000008	R/W	决定屏蔽哪个中断源。被屏蔽的中断源将不会服务 0 = 中断服务可用　　1 = 屏蔽中断服务	0xFFFFFFFF

➢ 中断屏蔽寄存器——INTMSK

名称	位索引	描述	初始值
INT_ADC	[31]	确定INT_ADC中断请求。0=允许中断；1=屏蔽中断。	1
INT_RTC	[30]	确定INT_RTC中断请求。0=允许中断；1=屏蔽中断。	1
INT_SPI1	[29]	确定INT_SPI1中断请求。0=允许中断；1=屏蔽中断。	1
INT_UART0	[28]	确定INT_UART0中断请求。0=允许中断；1=屏蔽中断。	1
INT_IIC	[27]	确定INT_IIC中断请求。0=允许中断；1=屏蔽中断。	1
INT_USBH	[26]	确定INT_USBH中断请求。0=允许中断；1=屏蔽中断。	1
INT_USBD	[25]	确定INT_USBD中断请求。0=允许中断；1=屏蔽中断。	1
INT_NFCON	[24]	确定INT_NFCON中断请求。0=允许中断；1=屏蔽中断。	1
INT_UART1	[23]	确定INT_UART1中断请求。0=允许中断；1=屏蔽中断。	1
INT_SPI0	[22]	确定INT_SPI0中断请求。0=允许中断；1=屏蔽中断。	1
INT_SDI	[21]	确定INT_SDI中断请求。0=允许中断；1=屏蔽中断。	1
INT_DMA3	[20]	确定INT_DMA3中断请求。0=允许中断；1=屏蔽中断。	1
INT_DMA2	[19]	确定INT_DMA2中断请求。0=允许中断；1=屏蔽中断。	1

中断屏蔽寄存器——INTMSK（续）

名称	位索引	描述	初始值
INT_DMA1	[18]	确定INT_DMA1中断请求。0=允许中断；1=屏蔽中断。	1
INT_DMA0	[17]	确定INT_DMA0中断请求。0=允许中断；1=屏蔽中断。	1
INT_LCD	[16]	确定INT_LCD中断请求。0=允许中断；1=屏蔽中断。	1
INT_UART2	[15]	确定INT_UART2中断请求。0=允许中断；1=屏蔽中断。	1
INT_TIMER4	[14]	确定INT_TIMER4中断请求。0=允许中断；1=屏蔽中断。	1
INT_TIMER3	[13]	确定INT_TIMER3中断请求。0=允许中断；1=屏蔽中断。	1
INT_TIMER2	[12]	确定INT_TIMER2中断请求。0=允许中断；1=屏蔽中断。	1
INT_TIMER1	[11]	确定INT_TIMER1中断请求。0=允许中断；1=屏蔽中断。	1
INT_TIMER0	[10]	确定INT_TIMER0中断请求。0=允许中断；1=屏蔽中断。	1
INT_WDT_AC97	[9]	确定INT_WDT_AC97中断请求。0=允许中断；1=屏蔽中断。	1
INT_TICK	[8]	确定INT_TICK中断请求。0=允许中断；1=屏蔽中断。	1
nBATT_FLT	[7]	确定nBATT_FLT中断请求。0=允许中断；1=屏蔽中断。	1
INT_CAM	[6]	确定INT_ CAM中断请求。0=允许中断；1=屏蔽中断。	1

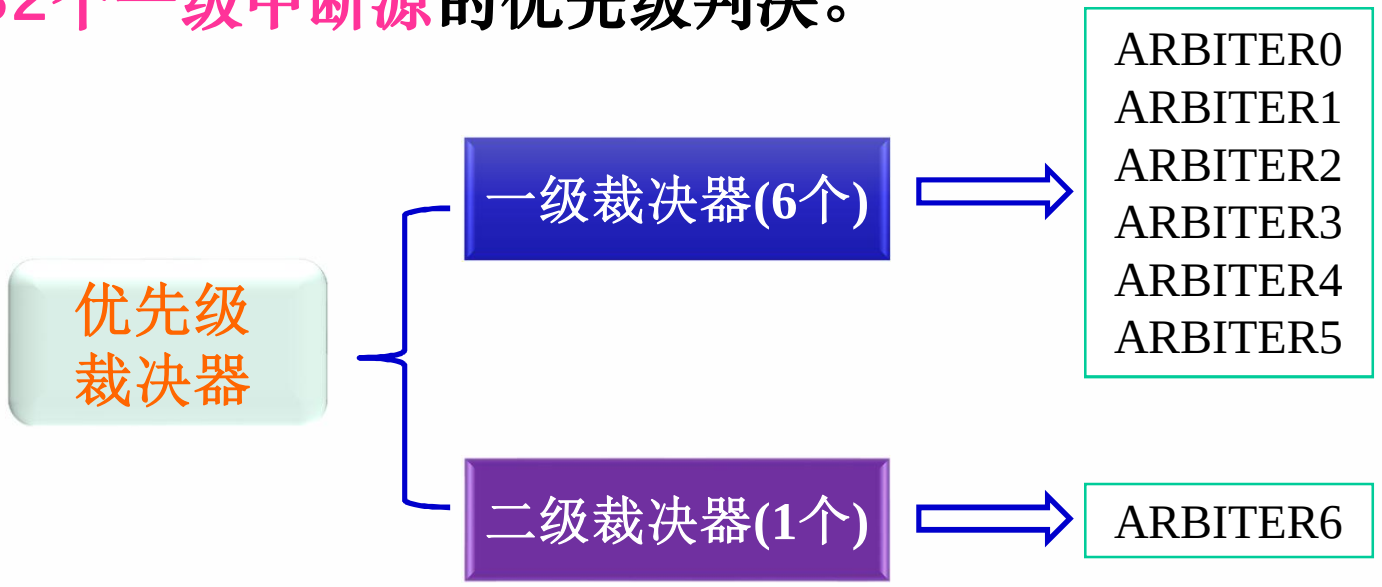
➢ 中断屏蔽寄存器——INTMSK（续）

名称	位索引	描述	初始值
EINT8_23	[5]	确定EINT8_23中断请求。0=允许中断；1=屏蔽中断。	1
EINT4_7	[4]	确定EINT4_7中断请求。0=允许中断；1=屏蔽中断。	1
EINT3	[3]	确定EINT3中断请求。0=允许中断；1=屏蔽中断。	1
EINT2	[2]	确定EINT2中断请求。0=允许中断；1=屏蔽中断。	1
EINT1	[1]	确定EINT1中断请求。0=允许中断；1=屏蔽中断。	1
EINT0	[0]	确定EINT0中断请求。0=允许中断；1=屏蔽中断。	1

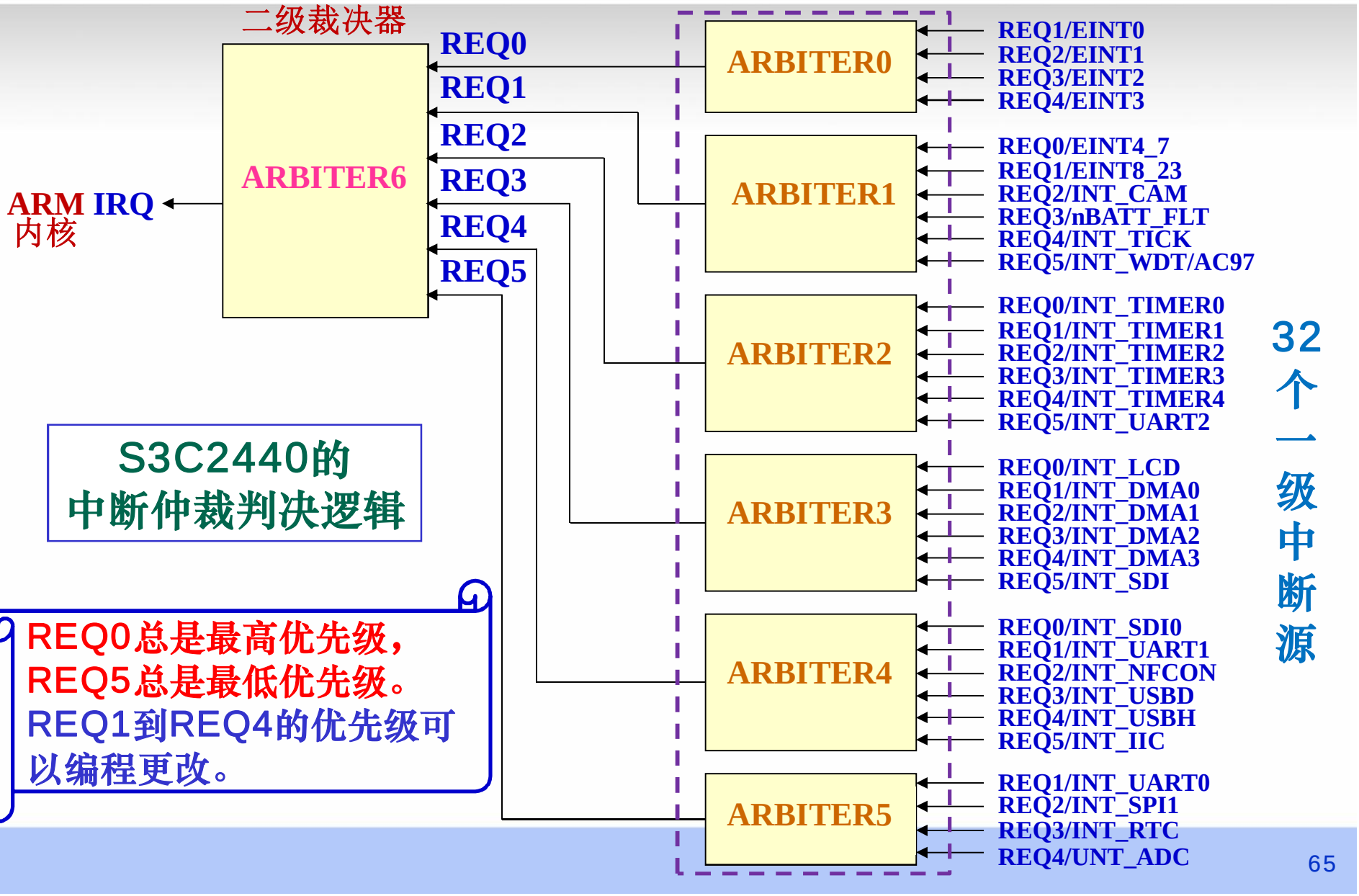
与外部中断屏蔽寄存器(EINTMASK)一同确定EINT4～ENT23中具体哪些信号被屏蔽 or 允许。

中断优先级

- 当多个中断源的中断请求信号同时有效时，需要通过中断优先级来确定对这些中断请求的服务顺序。
- 中断控制器借助优先级裁决器（仲裁判决器）实现对32个一级中断源的优先级判决。



与一级中断源相关的SFR



中断优先级的类型

- 静态优先级。——优先级固定
  - ✓ REQ1～REQ4的优先级一旦通过程序设定好，则固定不再变化。
- 动态优先级。——优先级循环
  - ✓ REQ1～REQ4的优先级通过程序设定好后,只要被中断处理了，则其优先级自动降为最低。

优先级寄存器

- 用于设定32个一级中断源在IRQ模式下的优先级顺序。
- 每个裁决器的优先级分别通过2位ARB_SEL（优先级顺序）和1位ARB_MODE（优先级类型）来设定。

寄存器	地址	R/W	描述	复位值
PRIORITY	0X4A00000C	R/W	IRQ 优先级控制寄存器	0x7F

➢ 优先级寄存器——PRIORITY

名称	位索引	描述	初始值
ARB_SEL6	[20:19]	确定仲裁判决器Arbiter 6的优先级顺序。 00 = REQ 0-1-2-3-4-5；01 = REQ 0-2-3-4-1-5； 10 = REQ 0-3-4-1-2-5；11 = REQ 0-4-1-2-3-5。	00
ARB_SEL5	[18:17]	确定仲裁判决器Arbiter 5的优先级顺序。 00 = REQ 1-2-3-4；01 = REQ 2-3-4-1； 10 = REQ 3-4-1-2；11 = REQ 4-1-2-3。	00
ARB_SEL4	[16:15]	确定仲裁判决器Arbiter 4的优先级顺序。 00 = REQ 0-1-2-3-4-5；01 = REQ 0-2-3-4-1-5； 10 = REQ 0-3-4-1-2-5；11 = REQ 0-4-1-2-3-5。	00
ARB_SEL3	[14:13]	确定仲裁判决器Arbiter 3的优先级顺序。 00 = REQ 0-1-2-3-4-5；01 = REQ 0-2-3-4-1-5； 10 = REQ 0-3-4-1-2-5；11 = REQ 0-4-1-2-3-5。	00
ARB_SEL2	[12:11]	确定仲裁判决器Arbiter 2的优先级顺序。 00 = REQ 0-1-2-3-4-5；01 = REQ 0-2-3-4-1-5； 10 = REQ 0-3-4-1-2-5；11 = REQ 0-4-1-2-3-5。	00
ARB_SEL1	[10:9]	确定仲裁判决器Arbiter 1的优先级顺序。 00 = REQ 0-1-2-3-4-5；01 = REQ 0-2-3-4-1-5； 10 = REQ 0-3-4-1-2-5；11 = REQ 0-4-1-2-3-5。	00

➢ 优先级寄存器——PRIORITY（续）

名称	位索引	描述	初始值
ARB_SEL0	[8:7]	确定仲裁判决器Arbiter 0的优先级顺序。 00 = REQ 1-2-3-4； 01 = REQ 2-3-4-1； 10 = REQ 3-4-1-2； 11 = REQ 4-1-2-3。	00
ARB_MODE6	[6]	确定仲裁判决器Arbiter 6的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE5	[5]	确定仲裁判决器Arbiter 5的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE4	[4]	确定仲裁判决器Arbiter 4的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE3	[3]	确定仲裁判决器Arbiter 3的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE2	[2]	确定仲裁判决器Arbiter 2的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE1	[1]	确定仲裁判决器Arbiter 1的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1
ARB_MODE0	[0]	确定仲裁判决器Arbiter 0的循环优先级顺序。 0 =优先级不循环； 1 = 优先级循环。	1

中断挂起寄存器

- 用于标记32个一级中断源的中断请求是否即将或者正在被微处理器服务。
  - ✓ =1：表明所对应的中断在所有已触发的中断中优先级最高且没有被屏蔽。
- 位于优先级逻辑之后，故同一时刻，只有1位被置1，并向微处理器发出IRQ信号。——★仅对IRQ模式的中断有效。
- 通常需要在中断服务程序中人工清除置位。

通过写入数据来清除。写入“1”表示清除，“0”表示保持不变。

寄存器	地址	R/W	描述	复位值
INTPND	0X4A000010	R/W	指示中断请求状态 0 = 未请求中断　　1 = 中断源已声明中断请求	0x00000000

与一级中断源相关的SFR

➢ 中断挂起寄存器——INTPND

名称	位索引	描述	初始值
INT_ADC	[31]	确定INT_ADC中断请求。0=没有请求；1=请求。	0
INT_RTC	[30]	确定INT_RTC中断请求。0=没有请求；1=请求。	0
INT_SPI1	[29]	确定INT_SPI1中断请求。0=没有请求；1=请求。	0
INT_UART0	[28]	确定INT_UART0中断请求。0=没有请求；1=请求。	0
INT_IIC	[27]	确定INT_IIC中断请求。0=没有请求；1=请求。	0
INT_USBH	[26]	确定INT_USBH中断请求。0=没有请求；1=请求。	0
INT_USBD	[25]	确定INT_USBD中断请求。0=没有请求；1=请求。	0
INT_NFCON	[24]	确定INT_NFCON中断请求。0=没有请求；1=请求。	0
INT_UART1	[23]	确定INT_UART1中断请求。0=没有请求；1=请求。	0
INT_SPI0	[22]	确定INT_SPI0中断请求。0=没有请求；1=请求。	0
INT_SDI	[21]	确定INT_SDI中断请求。0=没有请求；1=请求。	0
INT_DMA3	[20]	确定INT_DMA3中断请求。0=没有请求；1=请求。	0
INT_DMA2	[19]	确定INT_DMA2中断请求。0=没有请求；1=请求。	0

➢ 中断挂起寄存器——INTPND（续）

名称	位索引	描述	初始值
INT_DMA1	[18]	确定INT_DMA1中断请求。0=没有请求；1=请求。	0
INT_DMA0	[17]	确定INT_DMA0中断请求。0=没有请求；1=请求。	0
INT_LCD	[16]	确定INT_LCD中断请求。0=没有请求；1=请求。	0
INT_UART2	[15]	确定INT_UART2中断请求。0=没有请求；1=请求。	0
INT_TIMER4	[14]	确定INT_TIMER4中断请求。0=没有请求；1=请求。	0
INT_TIMER3	[13]	确定INT_TIMER3中断请求。0=没有请求；1=请求。	0
INT_TIMER2	[12]	确定INT_TIMER2中断请求。0=没有请求；1=请求。	0
INT_TIMER1	[11]	确定INT_TIMER1中断请求。0=没有请求；1=请求。	0
INT_TIMER0	[10]	确定INT_TIMER0中断请求。0=没有请求；1=请求。	0
INT_WDT_AC97	[9]	确定INT_WDT_AC97中断请求。0=没有请求；1=请求。	0
INT_TICK	[8]	确定INT_TICK中断请求。0=没有请求；1=请求。	0
nBATT_FLT	[7]	确定nBATT_FLT中断请求。0=没有请求；1=请求。	0
INT_CAM	[6]	确定INT_ CAM中断请求。0=没有请求；1=请求。	0

➢ 中断挂起寄存器——INTPND（续）

名称	位索引	描述	初始值
EINT8_23	[5]	确定EINT8_23中断请求。0=没有请求；1=请求。	0
EINT4_7	[4]	确定EINT4_7中断请求。0=没有请求；1=请求。	0
EINT3	[3]	确定EINT3中断请求。0=没有请求；1=请求。	0
EINT2	[2]	确定EINT2中断请求。0=没有请求；1=请求。	0
EINT1	[1]	确定EINT1中断请求。0=没有请求；1=请求。	0
EINT0	[0]	确定EINT0中断请求。0=没有请求；1=请求。	0

中断偏移寄存器

- 用于**标记**INTPND中置“1”位对应中断源的偏移量，其值代表了即将或者正在被微处理器服务的**中断源号**。
- 一旦**人工清除**了SRCPND和INTPND中的置“1”位，则该寄存器的值会**自动清除**。
- **注意：**与INTPND一样，仅对**IRQ模式**的中断有效。

寄存器	地址	R/W	描述	复位值
INTOFFSET	0x4A000014	R	指示 IRQ 中断请求源	0x00000000

与一级中断源相关的SFR

➢ 中断偏移寄存器——INTOFFSET

中断源	偏移量	中断源	偏移量	中断源	偏移量	中断源	偏移量
INT_ADC	31	INT_UART1	23	INT_UART2	15	nBATT_FLT	7
INT_RTC	30	INT_SPI0	22	INT_TIMER4	14	INT_CAM	6
INT_SPI1	29	INT_SDI	21	INT_TIMER3	13	EINT8_23	5
INT_UART0	28	INT_DMA3	20	INT_TIMER2	12	EINT4_7	4
INT_IIC	27	INT_DMA2	19	INT_TIMER1	11	EINT3	3
INT_USBH	26	INT_DMA1	18	INT_TIMER0	10	EINT2	2
INT_USBD	25	INT_DMA0	17	INT_WDT_AC97	9	EINT1	1
INT_NFCON	24	INT_LCD	16	INT_TICK	8	EINT0	0

与二级中断源相关的SFR

1.次级源挂起寄存器

寄存器	地址	R/W	描述	复位值
SUBSRCPND	0X4A000018	R/W	指示中断请求状态 0 = 未请求中断　　1 = 中断源已声明中断请求	0x00000000

2.中断次级屏蔽寄存器

寄存器	地址	R/W	描述	复位值
INTSUBMSK	0X4A00001C	R/W	决定屏蔽哪个中断源。被屏蔽的中断源将不会服务 0 = 中断服务可用　　1 = 屏蔽中断服务	0xFFFF

与外部中断源相关的SFR

外部中断源
INT0~INT23

外部中断控制寄存器
EXTINT0～EXTINT2

外部中断滤波寄存器
EINTFLT0～EINTFLT3

外部中断屏蔽寄存器
EINTMASK

外部中断挂起寄存器
EINTPEND

共9个

外部中断控制寄存器

- 用于设定外部中断请求信号的触发方式。
  - ✓ 电平触发：低电平触发、高电平触发
  - ✓ 边沿触发：下降沿触发、上升沿触发、双边沿触发

寄存器	地址	R/W	描述	复位值
EXTINT0	0x56000088	R/W	外部中断控制寄存器 0	0x000000
EXTINT1	0x5600008C	R/W	外部中断控制寄存器 1	0x000000
EXTINT2	0x56000090	R/W	外部中断控制寄存器 2	0x000000

与外部中断源相关的SFR

➢ 外部中断控制寄存器——EXTINT0

名称	位索引	描述	初始值
EINTn (n=0~7)	[4n+2:4n]	设置EINTn的信号触发方式 000=低电平；001=高电平；01x=下降沿触发； 10x=上升沿触发；11x=双边沿触发	000

➢ 外部中断控制寄存器——EXTINT1

名称	位索引	描述	初始值
FLTENn (n=8~15)	[4n-29]	EINTn的滤波器使能 0=滤波器禁止；1=滤波器使能	0
EINTn (n=8~15)	[4n-30: 4n-32]	设置EINTn的信号触发方式 000=低电平；001=高电平；01x=下降沿触发； 10x=上升沿触发；11x=双边沿触发	000

➢ 外部中断控制寄存器——EXTINT2

名称	位索引	描述	初始值
FLTENn (n=16~23)	[4n-61]	EINTn的滤波器使能 0=滤波器禁止；1=滤波器使能	0
EINTn (n=16~23)	[4n-62: 4n-64]	设置EINTn的信号触发方式 000=低电平；001=高电平；01x=下降沿触发； 10x=上升沿触发；11x=双边沿触发	000

外部中断滤波寄存器

- 用于设定8个外部中断请求信号（EINT16～EINT23）的滤波器的时钟信号来源和滤波宽度。
- 为了确保微处理器能有效检测到外部中断请求信号，要求信号的有效逻辑电平至少要保持40ns。

寄存器	地址	R/W	描述	复位值
EINTFLT0	0x56000094	R/W	保留	0x000000
EINTFLT1	0x56000098	R/W	保留	0x000000
EINTFLT2	0x5600009C	R/W	外部中断滤波寄存器 2	0x000000
EINTFLT3	0x560000A0	R/W	外部中断滤波寄存器 3	0x000000

➢ 外部中断滤波寄存器——EINTFLT2

名称	位索引	描述	初始值
FLTCLKn (n=16~19)	[8n-121]	设置EINTn的滤波器的时钟信号来源 0=PCLK；1=EXTCLK/OSC_CLK。	0
EINTFLTn (n=16~19)	[8n-122: 8n-128]	设置EINTn的滤波宽度	0x00

➢ 外部中断滤波寄存器——EINTFLT3

名称	位索引	描述	初始值
FLTCLKn (n=20~23)	[8n-153]	设置EINTn的滤波器的时钟信号来源 0=PCLK；1=EXTCLK/OSC_CLK。	0
EINTFLTn (n=20~23)	[8n-154: 8n-160]	设置EINTn的滤波宽度	0x00

外部中断屏蔽寄存器

- 用于设定20个外部中断请求信号（EINT4～EINT23）是否允许中断。

寄存器	地址	R/W	描述	复位值
EINTMASK	0x560000A4	R/W	外部中断屏蔽寄存器	0x000FFFFF

- 外部中断屏蔽寄存器——EINTMASK

名称	位索引	描述	初始值
EINTn (n=20~23)	[n]	设置EINTn的中断允许 0=使能中断；1=禁止中断。	0
EINTn (n=4~19)	[n]	设置EINTn的中断允许 0=使能中断；1=禁止中断。	1
保留	[3:0]	保留	111

外部中断挂起寄存器

- 用于标记20个外部中断请求信号（EINT4～EINT23）是否触发，供中断服务程序读取和判断。

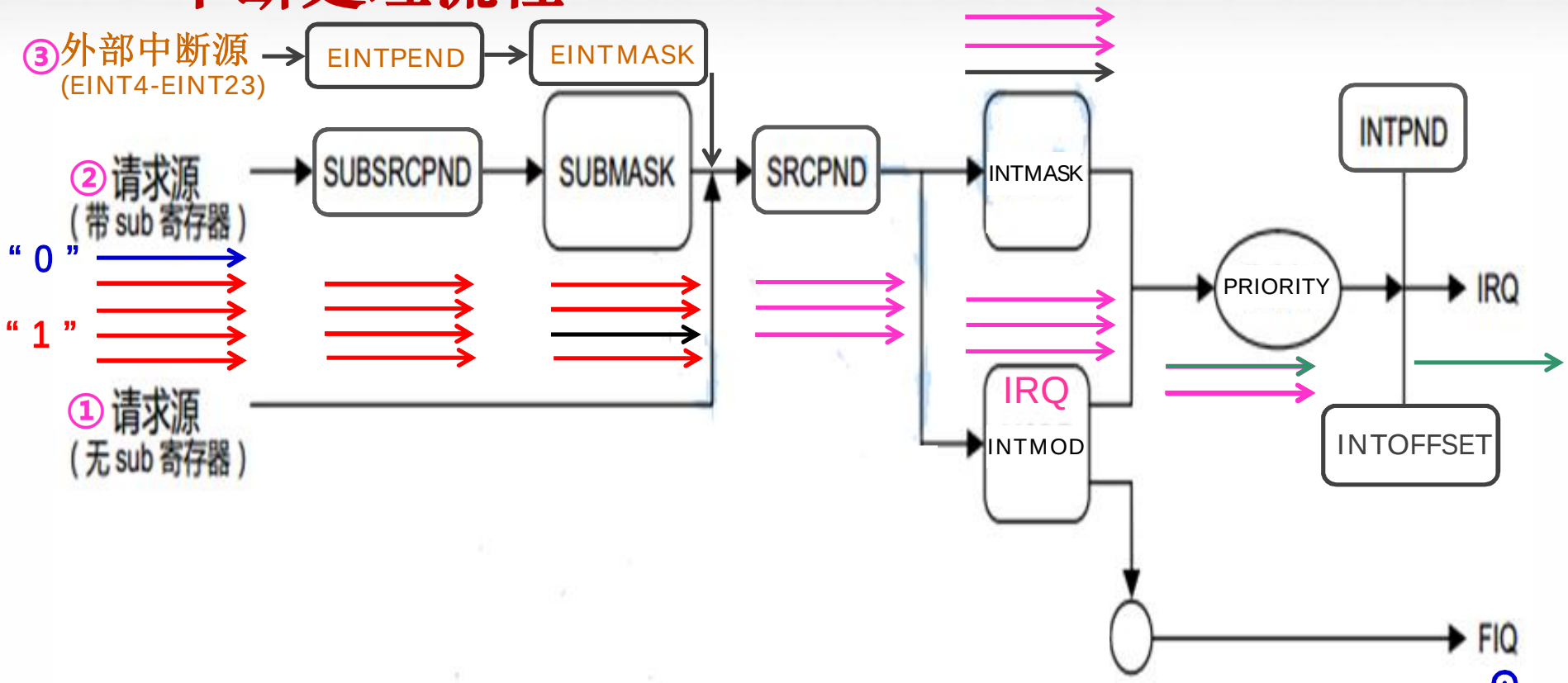
寄存器	地址	R/W	描述	复位值
EINTPEND	0x560000A8	R/W	外部中断挂起寄存器	0x00

- 外部中断挂起寄存器——EINTPEND

名称	位索引	描述	初始值
EINTn (n=4~23)	[n]	表明EINTn引脚中断请求信号是否有效 0=无中断请求；1=有中断请求。	0
保留	[3:0]	保留	000

△注意：对某位写入“1”，可实现对其清零。

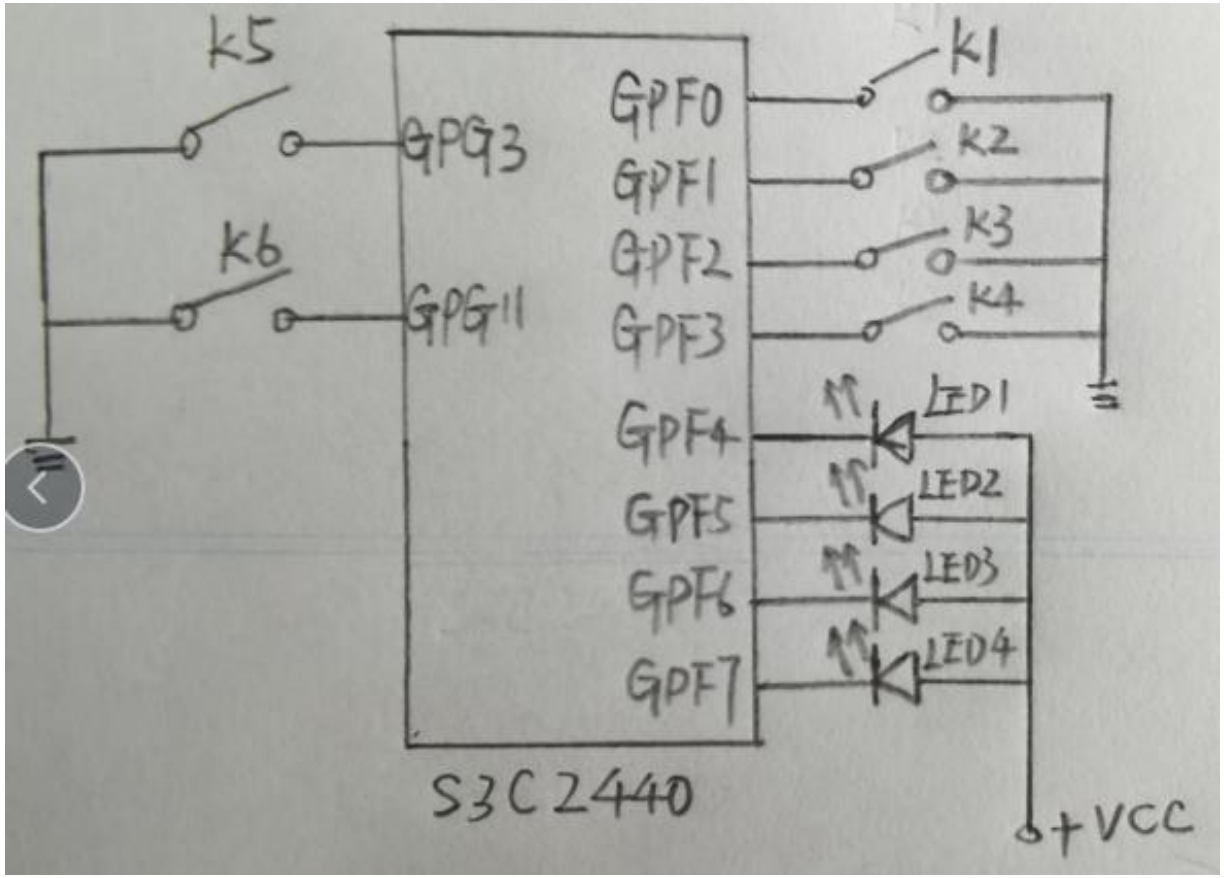
中断处理流程



CPSR：I=0（允许IRQ中断）
F=0（允许FIQ中断）

举例

- 开关通过中断方式点亮发光二极管。



K1按下，LED1亮；
K2按下，LED2亮；
K3按下，LED3亮；
K4按下，LED4亮;
K5按下，LED1和3亮；
K6按下，LED2和3亮。

举例

- 开关通过中断方式点亮发光二极管。

```
   GPFCON EQU 0x56000050
 1 GPFDAT EQU 0x56000054
 2 GPGCON EQU 0x56000060
 3 GPGDAT EQU 0x56000064
 4
 5 SRCPND EQU 0X4A000000
 6 INTMSK EQU 0X4A000008
 7 PRIORITY EQU 0x4A00000C
 8 INTPND EQU 0X4A000010
 9 INTOFFSET EQU 0x4A000014
10 EINTMASK EQU 0x560000a4
11 EINTPEND EQU 0x560000a8
12
13     AREA KEY_LED_INT, CODE, READONLY ;这里表示此区域是只读代码"KEY_LED"
14     EXPORT     _ENTRY ;引入程序入口
15
16 _ENTRY
17
18     b START          ;程序从START开始执行
19     b . ;0x4
20     b . ;0x8
21     b . ;0xc
22     b . ;0x10
23     b . ;0x14
24     b IRQ_HANDLER    ;0x00000018中断入口
25     b . ;0x1c
```

伪指令EQU
将寄存器与端口地址关联

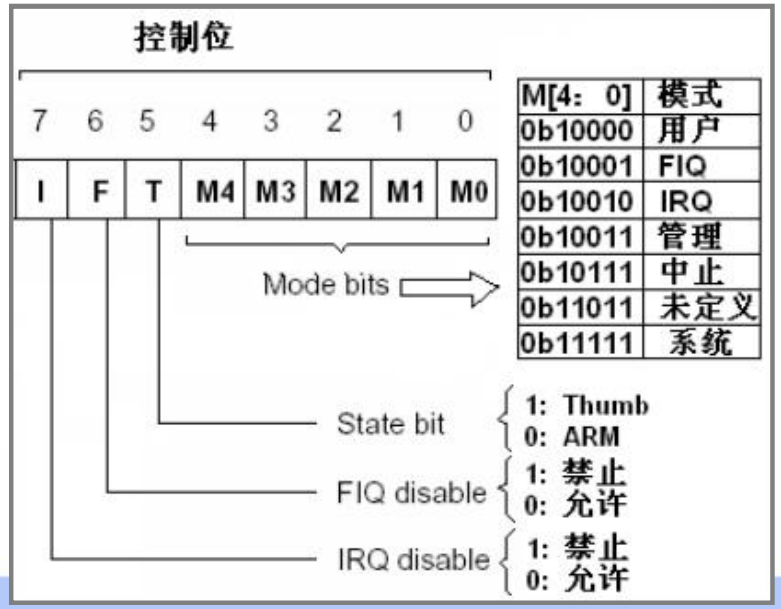
异常向量表
IRQ的异常向量地址0x00000018

主程序

```
28 START
29     ldr sp, =4096          ;初始化当前模式(系统模式)栈指针
30     msr cpsr_c, #0x92      ;初始化配置前，设置当前模式为IRQ中断模式
31     ldr sp, =3072          ;初始化IRQ模式的栈指针
32     bl INIT_CONF           ;跳转到初始化配置
33     msr cpsr_c, #0x5f      ;设置当前模式为系统模式，并开启IRQ中断允许
34
35
36 LOOP
37     ldr r0, =GPFDAT
38     mov r1, #0xff(or #0xf0)    ;将LED熄灭
39     str r1, [r0]
40     b LOOP
```

1 0 0 10010

0 1 0 11111



初始化配置函数

```
42 INIT_CONF
43      ;GPIO的配置，包括K1~K4中断配置和LED1~LED4的配置
44      ldr r0, =GPFCON
45      ldr r1, =0x55aa
46      str r1, [r0]
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

GPFCON	位	描述			
GPF7	[15:14]	00 = 输入	01 = 输出	10 = EINT[7]	11 = 保留
GPF6	[13:12]	00 = 输入	01 = 输出	10 = EINT[6]	11 = 保留
GPF5	[11:10]	00 = 输入	01 = 输出	10 = EINT[5]	11 = 保留
GPF4	[9:8]	00 = 输入	01 = 输出	10 = EINT[4]	11 = 保留
GPF3	[7:6]	00 = 输入	01 = 输出	10 = EINT[3]	11 = 保留
GPF2	[5:4]	00 = 输入	01 = 输出	10 = EINT[2]	11 = 保留
GPF1	[3:2]	00 = 输入	01 = 输出	10 = EINT[1]	11 = 保留
GPF0	[1:0]	00 = 输入	01 = 输出	10 = EINT[0]	11 = 保留

初始化配置函数

```
42  INIT_CONF
43      ;GPIO的配置，包括K1~K4中断配置和LED1~LED4的配置
44      ldr r0, =GPFCON
45      ldr r1, =0x55aa
46      str r1, [r0]
47
48      ;K5和K6的配置,配置成中断模式
49      ldr r0, =GPGCON
50      ldr r1, =0x00800080
51      str r1, [r0]
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

GPEGCON	位	描述			
GPG15*	[31:30]	00 = 输入	01 = 输出	10 = EINT[23]	11 = 保留
GPG14*	[29:28]	00 = 输入	01 = 输出	10 = EINT[22]	11 = 保留
GPG13*	[27:26]	00 = 输入	01 = 输出	10 = EINT[21]	11 = 保留
GPG12	[25:24]	00 = 输入	01 = 输出	10 = EINT[20]	11 = 保留
GPG11	[23:22]	00 = 输入	01 = 输出	10 = EINT[19]	11 = TCLK[1]
GPG10	[21:20]	00 = 输入	01 = 输出	10 = EINT[18]	11 = nCTS1
GPG9	[19:18]	00 = 输入	01 = 输出	10 = EINT[17]	11 = nRTS1
GPG8	[17:16]	00 = 输入	01 = 输出	10 = EINT[16]	11 = 保留
GPG7	[15:14]	00 = 输入	01 = 输出	10 = EINT[15]	11 = SPICLK1
GPG6	[13:12]	00 = 输入	01 = 输出	10 = EINT[14]	11 = SPIMOSI1
GPG5	[11:10]	00 = 输入	01 = 输出	10 = EINT[13]	11 = SPIMISO1
GPG4	[9:8]	00 = 输入	01 = 输出	10 = EINT[12]	11 = LCD_PWRDN
GPG3	[7:6]	00 = 输入	01 = 输出	10 = EINT[11]	11 = nSS1
GPG2	[5:4]	00 = 输入	01 = 输出	10 = EINT[10]	11 = nSS0
GPG1	[3:2]	00 = 输入	01 = 输出	10 = EINT[9]	11 = 保留
GPG0	[1:0]	00 = 输入	01 = 输出	10 = EINT[8]	11 = 保留

初始化配置函数

```
42  INIT_CONF
43      ;GPIO的配置， 包括K1~K4中断配置和LED1~LED4的配置
44      ldr r0, =GPFCON
45      ldr r1, =0x55aa
46      str r1, [r0]
47
48      ;K5和K6的配置,配置成中断模式
49      ldr r0, =GPGCON
50      ldr r1, =0x00800080
51      str r1, [r0]
52
53      ;初始化将led都熄灭
54      ldr r0, =GPFDAT
55      mov r1, #0xf0
56      str r1, [r0]
57
58      ;初始化中断屏蔽位
59      ldr r0, =INTMSK
60      ldr r1, = 0xfffffffd0
61      str r1, [r0]
62
63      ;初始化外中断屏蔽位
64      ldr r0, =EINTMASK
65      ldr r1, = 0xf7f7ff ;
66      str r1, [r0]
67
68      mov pc, lr                    ;函数返回
```

INTMSK	位		描述
EINT8_23	[5]	0 = 可服务 √	1 = 屏蔽
EINT4_7	[4]	0 = 可服务	1 = 屏蔽
EINT3	[3]	0 = 可服务 √	1 = 屏蔽
EINT2	[2]	0 = 可服务 √	1 = 屏蔽
EINT1	[1]	0 = 可服务 √	1 = 屏蔽
EINT0	[0]	0 = 可服务 √	1 = 屏蔽

将EINT19和EINT11中断允许

IRQ异常处理函数（中断服务函数）

```
76  IRQ_HANDLER
77      ;返回地址等于中断地址减4
78      sub lr, lr, #4
79      ;保存现场
80      stmfd sp!, {r0-r12,lr}
81      STMFD
82      ;得到offset寄存器的值，它反映了触发中断的中断号
83      ldr r0, =INTOFFSET
84      ldr r2, [r0]
85
86      ;得到LED的状态数据寄存器
87      ldr r0, =GPFDAT
88      ldr r1, [r0]
89
90      ;判断是否触发了INT0，即按键1的中断
91      cmp r2, #0
92      ;是则重新布置LED状态
93      moveq r1, #0xef
94      beq INIT_OVER
95
96      ;INT1，按键2
97      cmp r2, #1
98      moveq r1, #0xdf
99      beq INIT_OVER
100
101     ;INT2，按键3
102     cmp r2, #2
103     moveq r1, #0xbf
104     beq INIT_OVER
```

中断源	偏移量
nBATT_FLT	7
INT_CAM	6
EINT8_23	5
EINT4_7	4
EINT3	3
EINT2	2
EINT1	1
EINT0	0

```
106     ;INT3，按键4
107     cmp r2, #3
108     moveq r1, #0x7f
109     beq INIT_OVER
110
111     ;INT5，它可能是按键5和6发生了中断，按键5和6引脚比较特殊
112     cmp r2, #5
113     ;如果不是，跳转到结束标志
114     bne INIT_OVER
115     ;如果是，就看EINTPEND的值，它必定有一个中断被置1
116     ldr r2, =EINTPEND
117     ldr r3, [r2]
118
119     ;如果是11号引脚发生中断，即按键5发生了中断
120     tst r3, #0x800          ;2^11
121     movne r1, #0xaf
122     bne INIT_OVER
123
124     ; 如果是19号引脚发生中断，即按键6发生了中断
125     tst r3, #0x80000        ;2^19
126     movne r1, #0x9f
127     bne INIT_OVER
```

中断返回函数

```
124 INIT_OVER
125     str r1, [r0]                    ;更新LED状态
126
127     ldr r0, =EINTPEND
128     mov r1, #0xffffffff             ;清除所有中断
129     str r1, [r0]
130
131     ldr r0, =INTPND
132     mov r1, #0xff
133     str r1, [r0]
134
135     ldr r0, =SRCPND
136     mov r1, #0xff
137     str r1, [r0]
138
139     ldmia sp!, {r0-r12,pc}^         ;恢复现场
140      LDMFD
141     END
```

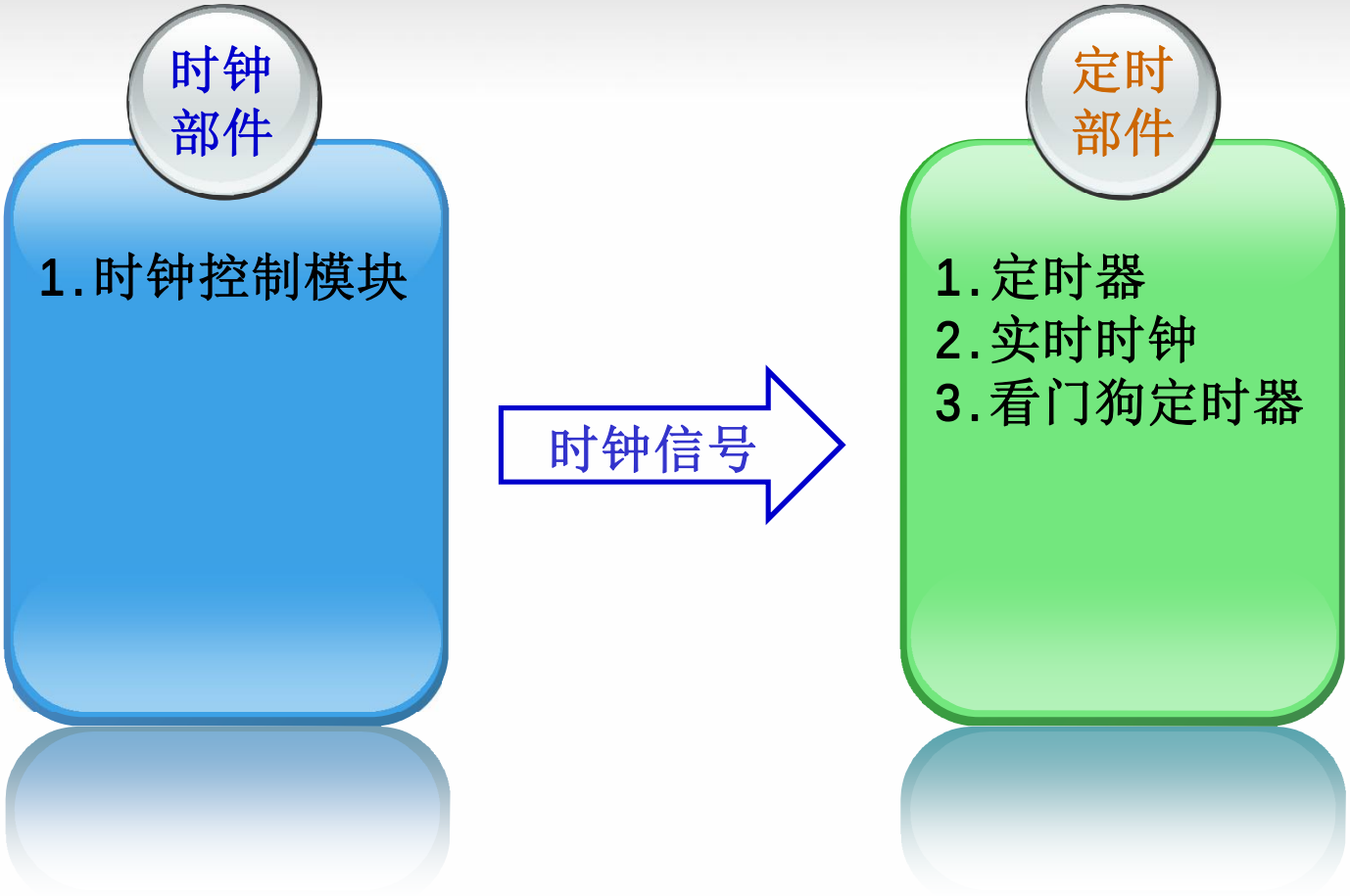
向***挂起寄存器某位写入“1”，实现将对应位清零。

- 5.1 通用输入输出端口(GPIO)
- 5.2 中断系统
- 5.3 定时部件
- 5.4 通用异步收发器(UART)*

作用

- **时钟部件**
  - ✓ 为微处理器工作提供基本的时钟信号，以实现其内部功能电路及外围设备的时序控制。
- **定时部件**
  - ✓ 产生不同周期或特定波形的时钟信号，满足不同的实际应用需求。
  - ✓ 对外部输入信号进行计数。
  - ✓ 为系统提供时间信息，例如年、月、日、星期、时、分、秒等。
  - ✓ 当系统出现故障时，为各控制器提供复位信号。

时钟与定时部件



时钟控制模块

以S3C2440为例

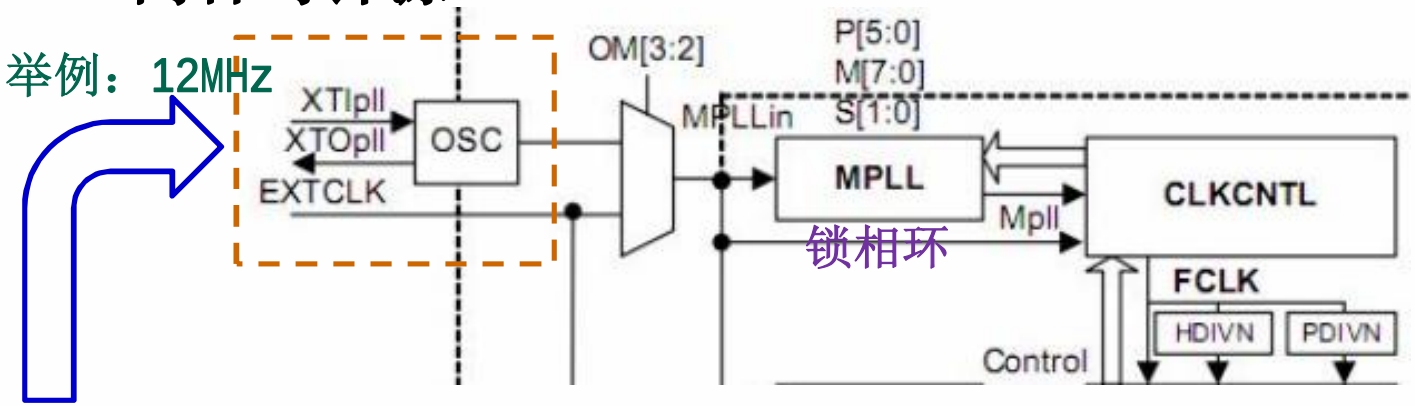
- 三种内部时钟信号

FCLK 内核时钟	HCLK 总线时钟	PCLK I/O接口时钟
• 供微处理器(内核)使用的时钟信号	• 供高性能总线AHB使用的时钟信号	• 供外围总线APB使用的时钟信号

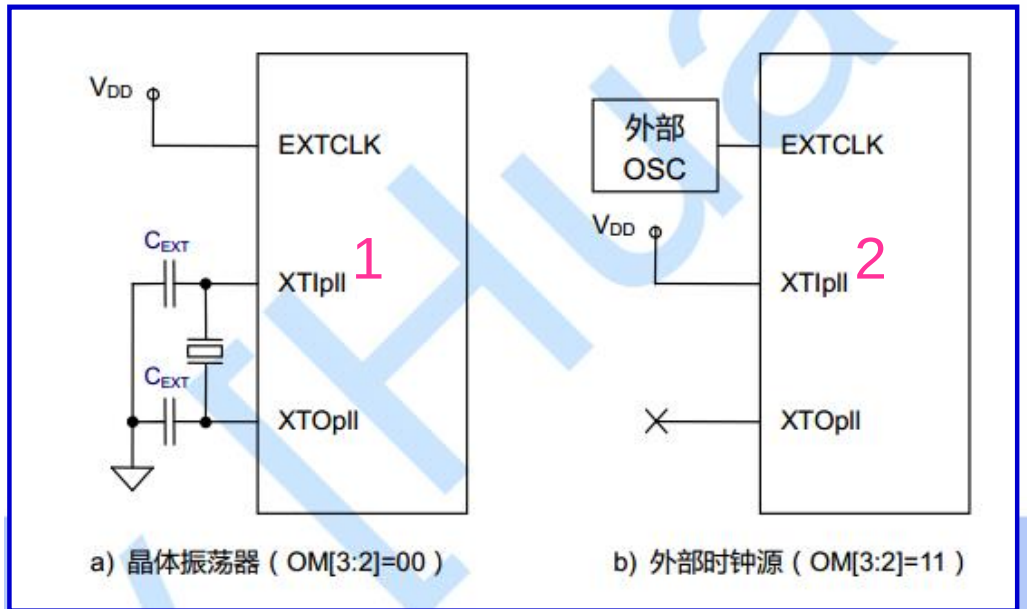
时钟频率：高 → 低

时钟控制模块(续)

两种时钟源



400MHz 100MHz 50MHz
(FCLK) (HCLK) (PCLK)



△时钟频率的计算公式

FCLK= $F_{OUT} = 2 * m * Fin / (p*2^S)$

- Fin：输入时钟源的频率。
- m、p、s：分频控制参数，分别由锁相环配置寄存器MPLLCON中MDIV、PDIV和SDIV的值确定。

时钟控制模块(续)

➢ 分频比例的选择——设定三种时钟信号的比例关系

时钟分频控制（CLKDIVN）寄存器

寄存器	地址	R/W	描述	复位值
CLKDIVN	0x4C000014	R/W	时钟分频控制寄存器	0x00000004

CLKDIVN	位	描述	初始状态
DIVN_UPLL	[3]	UCLK 选择寄存器（UCLK 必须为 48MHz 给 USB） 0：UCLK = UPLL 时钟　　1：UCLK = UPLL 时钟 / 2 当 UPLL 时钟被设置为 48MHz 时，设置为 0 当 UPLL 时钟被设置为 96MHz 时，设置为 1	0
HDIVN	[2:1]	00：HCLK = FCLK/1 01：HCLK = FCLK/2 10：HCLK = FCLK/4　当 CAMDIVN[9] = 0 时 HCLK = FCLK/8　当 CAMDIVN[9] = 1 时 11：HCLK = FCLK/3　当 CAMDIVN[8] = 0 时 HCLK = FCLK/6　当 CAMDIVN[8] = 1 时	00
PDIVN	[0]	0：PCLK 是和 HCLK/1 相同的时钟 1：PCLK 是和 HCLK/2 相同的时钟	0

时钟控制模块(续)

举例：时钟信号频率计算

△已知时钟源的频率为12MHz，MPLLCON中MDIV=92，PDIV=1，SDIV=1，且CLKDIVN中PCLK:HCLK:FCLK设置为1:2:8。试计算FCLK、HCLK和PCLK的频率。

分频参数：m = (MDIV + 8), p = (PDIV + 2), s = SDIV

=92+8　　=1+2　　=1

FCLK＝2*(92+8)*(12000000)/(3×2^1)=400000000=400MHz

HCLK=400/4=100MHz

PCLK=400/8=50MHz

时钟控制模块(续)

举例：相关寄存器的设定程序

```
;设置锁相环配置寄存器MPLLCON
ldr   r0, = MPLLCON
ldr   r1, = ((92 << 12) + (1 << 4) + 1)
str   r1, [r0]

;设置时钟分频控制寄存器CLKDIVN
ldr   r0,=CLKDIVN
ldr   r1,=0x5          ;0x5=0101b
str   r1,[r0]
```

MPLLCON	Bit	
MDIV	[19:12]	Main divider control
PDIV	[9:4]	Pre-divider control
SDIV	[1:0]	Post divider control

CLKDIVN	Bit	Description
DIVN_UPLL	[3]	UCLK select register(UCLK must be 48MHz for USB) 0: UCLK = UPLL clock 1: UCLK = UPLL clock / 2 Set to 0, when UPLL clock is set as 48MHz Set to 1. when UPLL clock is set as 96MHz.
HDIVN	[2:1]	00 : HCLK = FCLK/1. 01 : HCLK = FCLK/2. 10 : HCLK = FCLK/4 when CAMDIVN[9] = 0. HCLK= FCLK/8 when CAMDIVN[9] = 1. 11 : HCLK = FCLK/3 when CAMDIVN[8] = 0. HCLK = FCLK/6 when CAMDIVN[8] = 1.
PDIVN	[0]	0: PCLK has the clock same as the HCLK/1. 1: PCLK has the clock same as the HCLK/2.

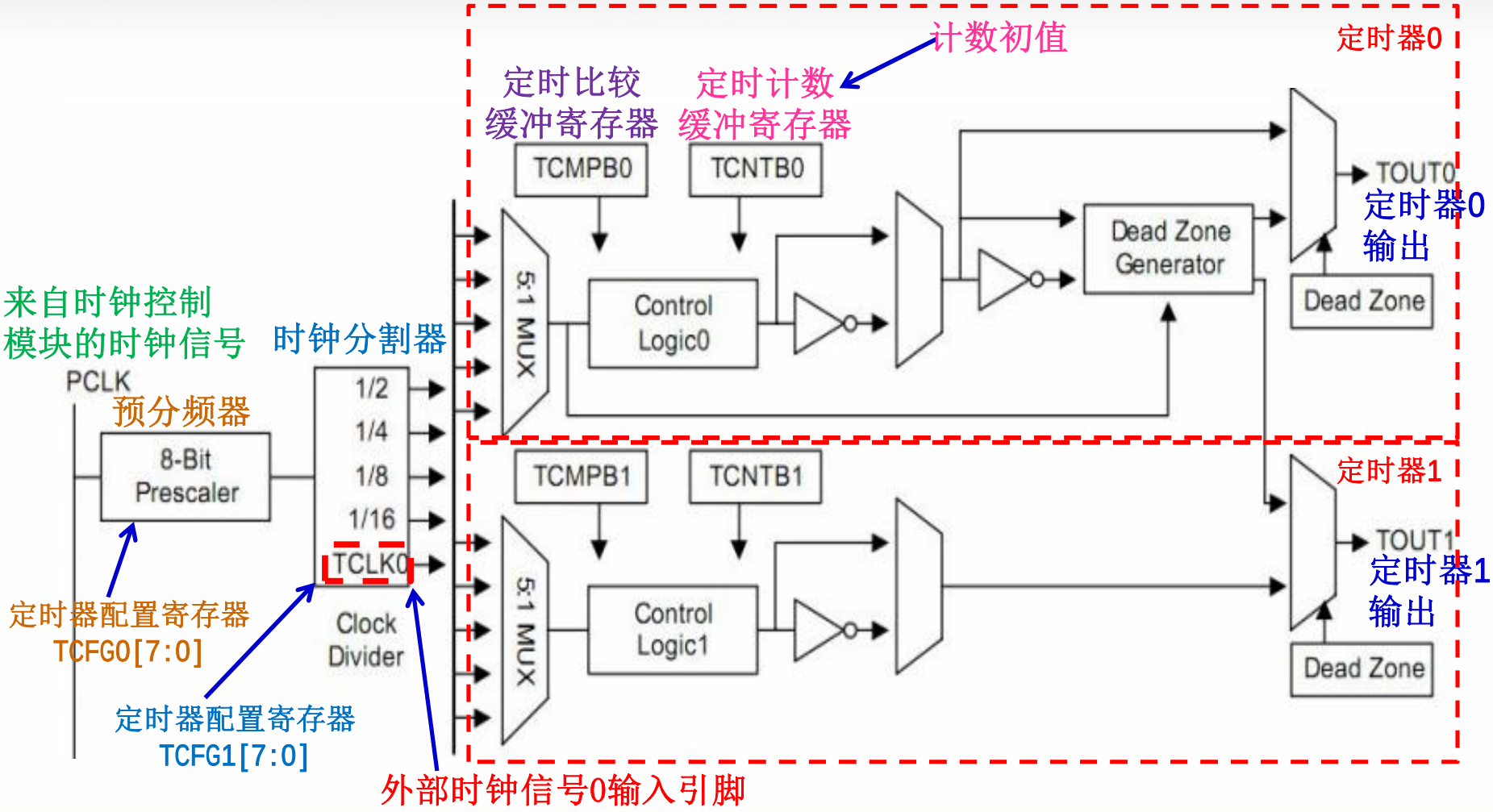
定时器

以S3C2440为例

- 功能：定时、计数、脉宽调制(PWM：Pulse Width Modulation)。
- 5个16位定时器(递减计数器)。
  - 定时器0、1、2和3具有PWM功能。
  - 定时器4是一个无输出引脚的内部定时器。
  - 定时器0还包含用于大电流驱动的死区发生器。
- 每个定时器都包含一个可以生成 5 种不同分频信号(1/2、1/4、1/8、1/16和外部时钟TCLK)的时钟分频器。
- 定时器0和1共用一个8位预分频器，定时器2、3和4共用另外一个的8位预分频器。

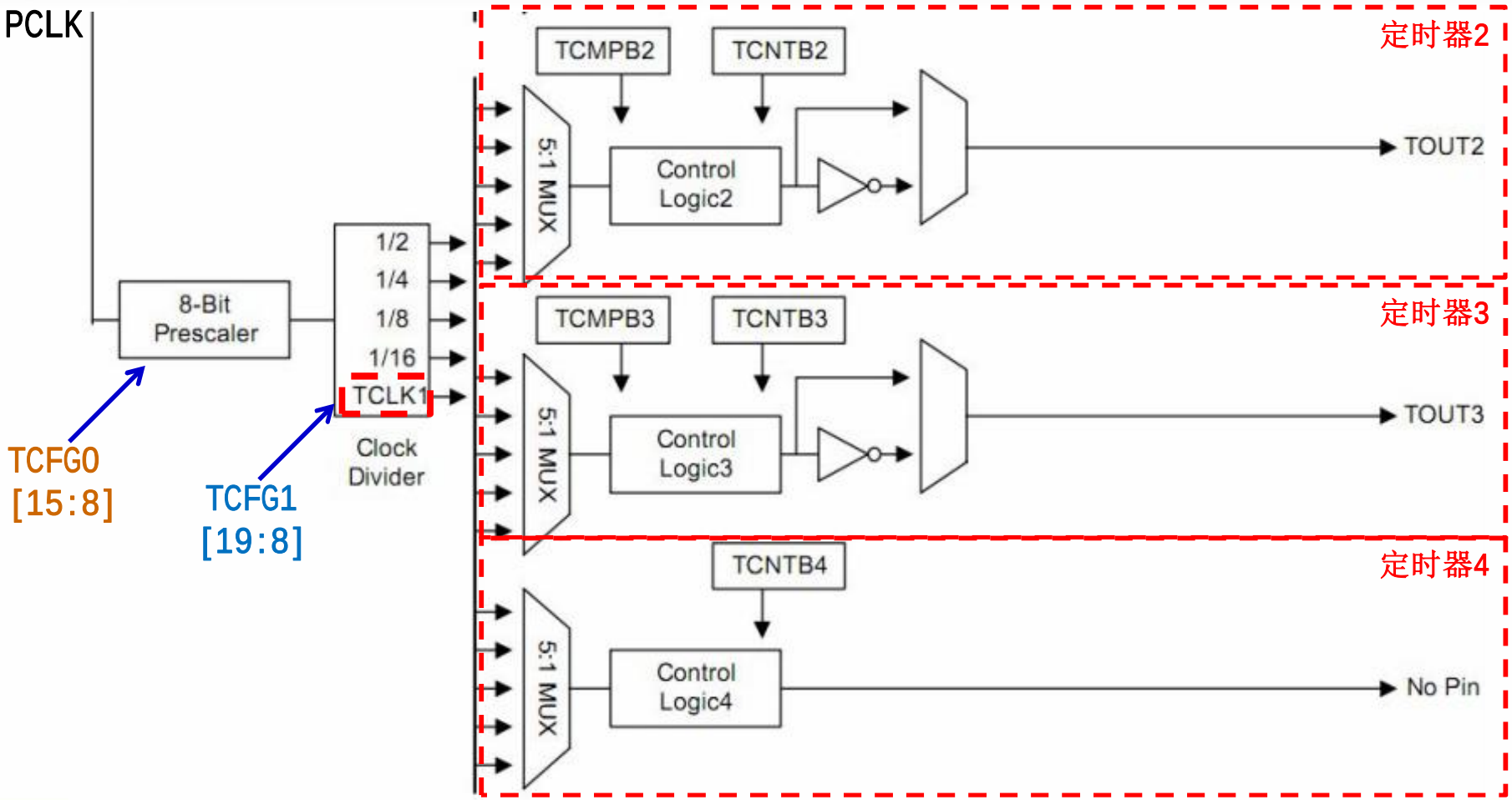

学院 荣誉 责任

定时器的内部结构


计数初值
定时器0
定时比较
缓冲寄存器
定时计数
缓冲寄存器
TCMPB0
TCNTB0
TOUT0
定时器0
输出
Dead Zone
Generator
Dead Zone
5:1 MUX
Control
Logic0
来自时钟控制
模块的时钟信号
时钟分割器
PCLK
预分频器
8-Bit
Prescaler
1/2
1/4
1/8
1/16
TCLK0
Clock
Divider
定时器配置寄存器
TCFG0[7:0]
定时器配置寄存器
TCFG1[7:0]
外部时钟信号0输入引脚
TCMPB1
TCNTB1
定时器1
TOUT1
定时器1
输出
Dead Zone
5:1 MUX
Control
Logic1

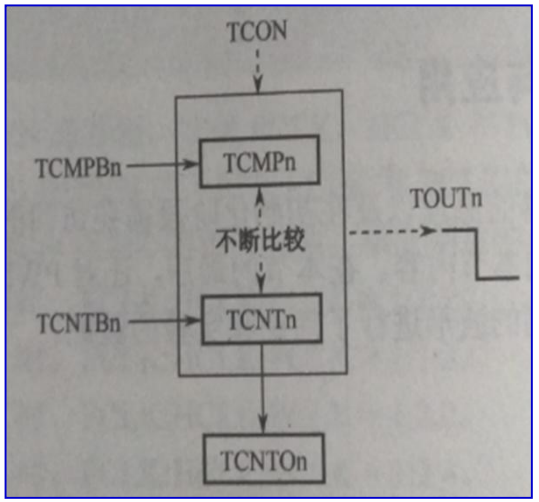

学院 荣誉 责任

定时器的内部结构(续)

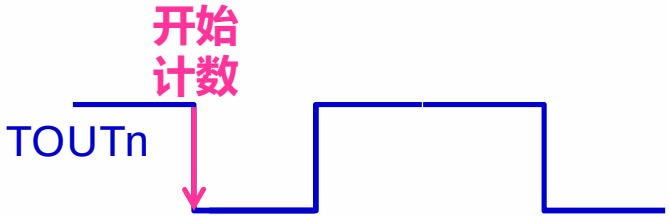

PCLK
8-Bit Prescaler
TCFG0 [15:8]
TCFG1 [19:8]
1/2
1/4
1/8
1/16
TCLK1
Clock Divider
定时器2
TCMPB2
TCNTB2
5:1 MUX
Control Logic2
TOUT2
定时器3
TCMPB3
TCNTB3
5:1 MUX
Control Logic3
TOUT3
定时器4
TCNTB4
5:1 MUX
Control Logic4
No Pin

定时器的工作原理

- 实际执行减1操作的是定时计数寄存器TCNTn。为了防止对其读写影响计数，不允许程序对其直接访问。
  - ✓ 将计数初值写入定时计数缓冲寄存器TCNTBn。每次定时开始时，其值会被复制到TCNTn中。（同时TCMPBn→TCMPn）
  - ✓ 读取TCNTBn，读出的是下次定时操作的重载值，并非TCNTn的当前计数值。
  - ✓ 读取定时计数监视寄存器TCNTOn，可以间接获取TCNTn的动态计数值。

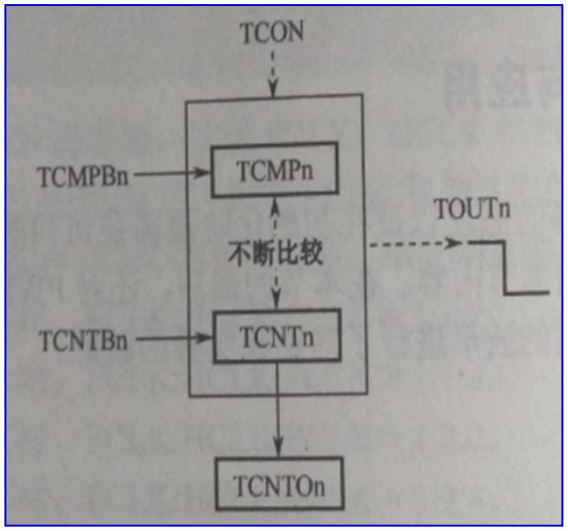


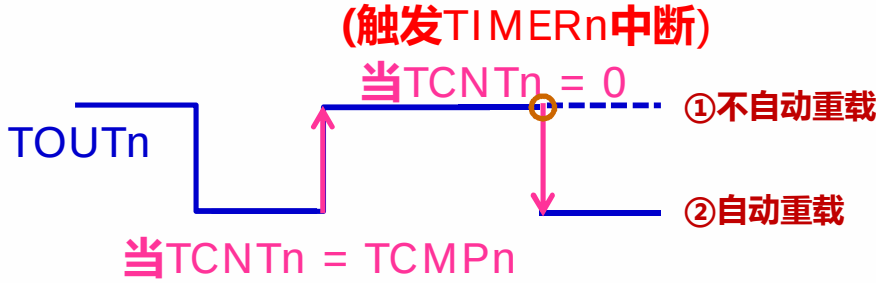
- 启动计数后，TOUTn引脚的输出信号进行第一次反相。



定时器的工作原理(续)

- 当TCNTn等于定时比较寄存器TCMPn时，TOUTn引脚的输出信号进行第二次反相。
- TCNTn减到0时，若没有屏蔽允许，则触发TIMERn中断。并根据定时器控制寄存器TCON的“自动重载”标志，决定是否复制TCNTBn到TCNTn中，以开启下一次定时操作。

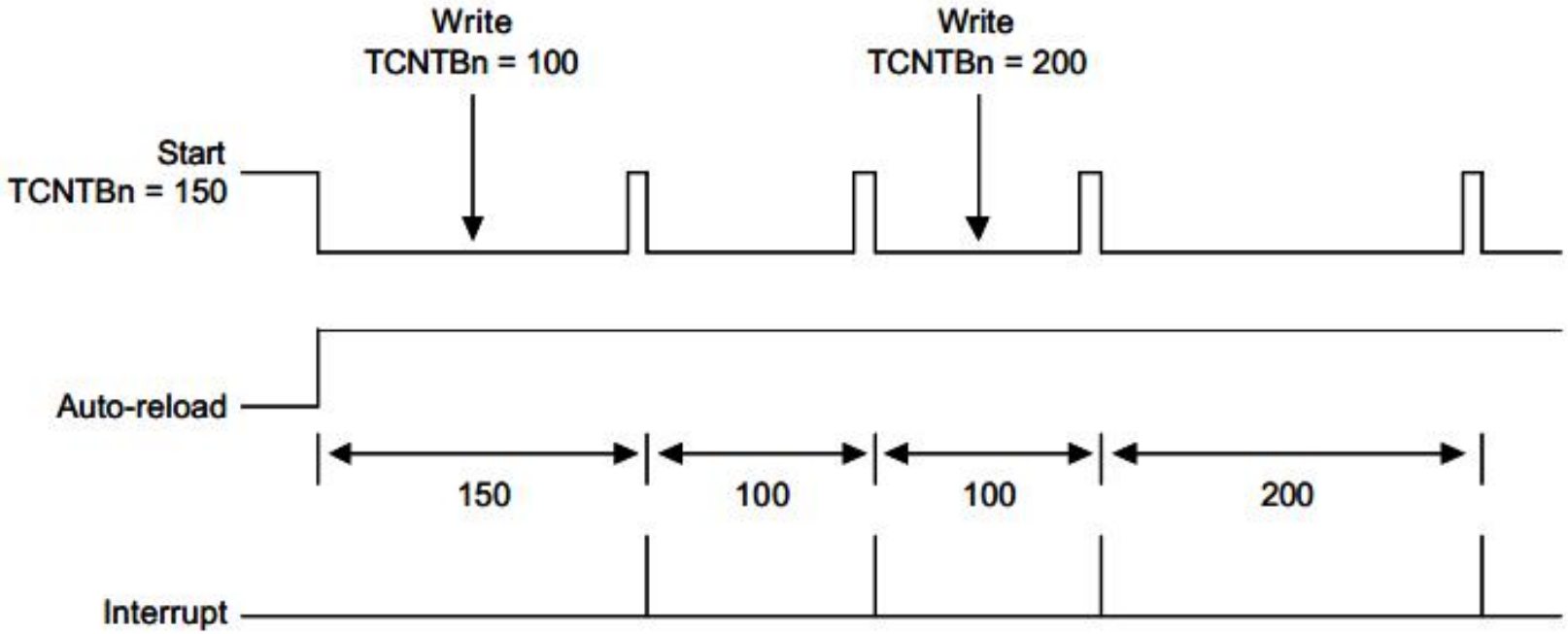




定时器的工作原理(续)

- TCNTBn和TCMPBn有双缓冲功能，故在当前定时操作中设置的新的定时器值，仅当当前定时操作执行完毕后才有效。

——非立即有效



相关寄存器介绍

共用寄存器

定时器配置寄存器0

寄存器	地址	R/W	描述	复位值
TCFG0	0x51000000	R/W	配制两个 8 位预分频器	0x00000000

定时器配置寄存器1

寄存器	地址	R/W	描述	复位值
TCFG1	0x51000004	R/W	5 路多路选择器和 DMA 模式选择寄存器	0x00000000

定时器控制寄存器

寄存器	地址	R/W	描述	复位值
TCON	0x51000008	R/W	定时器控制寄存器	0x00000000

定时器配置寄存器0——TCFG0

名称	位索引	描述	初始值
保留	[31:24]		0x00
死区长度	[23:16]	决定死区段。 死区段持续为 1 的时间等于定时器 0 持续为 1的时间。	0x00
预分频1	[15:8]	决定了定时器 2、3 和 4 的预分频值	0x00
预分频0	[7:0]	决定了定时器 0 和 1 的预分频值	0x00

定时器配置寄存器1——TCFG1

名称	位索引	描述	初始值
保留	[31:24]		0x00
DMA模式	[23:20]	选择 DMA 请求通道。 0000 = 未选择(所有中断)；0001 = 定时器0； 0010 = 定时器 1；0011 = 定时器 2； 0100 = 定时器 3；0101 = 定时器 4；0110 = 保留	0x0
MUX4	[19:16]	选择 PWM 定时器 4 的选通输入(同定时器2)	0x0
MUX3	[15:12]	选择 PWM 定时器 3的选通输入(同定时器2)	0x0
MUX2	[11:8]	选择 PWM 定时器 2 的选通输入 0000 = 1/2；0001 = 1/4；0010 = 1/8；0011 = 1/16； 01xx = 外部 TCLK1	0x0
MUX1	[7:4]	选择 PWM 定时器 1的选通输入(同定时器0)	0x0
MUX0	[3:0]	选择 PWM 定时器 0的选通输入 0000 = 1/2；0001 = 1/4；0010 = 1/8；0011 = 1/16； 01xx = 外部 TCLK0	0x0

定时器控制寄存器——TCON

名称	位索引	描述	初始值
…	[22:8]	设定定时器1～定时器4的参数。(同定时器0)	0x0000
保留	[7:5]		000
死区使能	[4]	决定死区操作 0 = 禁止； 1 = 使能	0
定时器 0 自动重载开/关	[3]	决定定时器 0 的自动重载开启或关闭 0 = 单稳态； 1 = 间隙模式（自动重载）	0
定时器 0 输出变相开/关	[2]	决定定时器 0 的输出变相开启或关闭 0 = 关闭变相； 1 = TOUT0 变换极性	0
定时器 0 手动更新	[1]	决定定时器 0 的手动更新 0 = 无操作； 1 = 更新 TCNTB0 和 TCMPB0	0
定时器 0 启动/停止	[0]	决定定时器 0 的启动或停止 0 = 停止定时器 0； 1 = 启动定时器 0	0

默认关闭变相，TOUTn初始为高电平。

相关寄存器介绍(续)

定时器0的寄存器

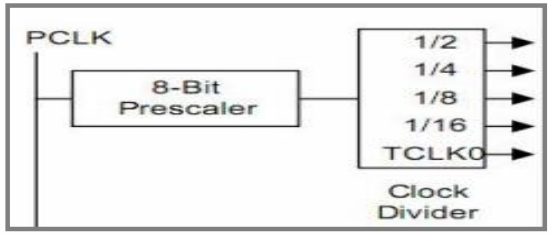
定时器0计数缓冲寄存器

定时器0比较缓冲寄存器

寄存器	地址	R/W	描述	复位值
TCNTB0	0x5100000C	R/W	定时器0计数缓冲寄存器 (即计数初值)	0x00000000
TCMPB0	0x51000010	R/W	定时器0比较缓冲寄存器	0x00000000

计数初值 = 定时器工作频率 ÷ (PWM)定时输出信号频率
= (PWM)定时输出信号周期 ÷ 定时器工作周期

定时器工作频率= PCLK频率 ÷ (预分频值+1) ÷ 分频值
TCLK频率



定时器0计数监视寄存器

寄存器	地址	R/W	描述	复位值
TCNTO0	0x51000014	R	定时器0计数监视寄存器	0x00000000

相关寄存器介绍(续)

定时器1的寄存器

定时器1计数缓冲寄存器

定时器1比较缓冲寄存器

寄存器	地址	R/W	描述	复位值
TCNTB1	0x51000018	R/W	定时器1计数缓冲寄存器	0x00000000
TCMPB1	0x5100001C	R/W	定时器1比较缓冲寄存器	0x00000000

定时器1计数监视寄存器

寄存器	地址	R/W	描述	复位值
TCNTO1	0x51000020	R	定时器1计数监视寄存器	0x00000000

相关寄存器介绍(续)

定时器2的寄存器

定时器2计数缓冲寄存器

定时器2比较缓冲寄存器

寄存器	地址	R/W	描述	复位值
TCNTB2	0x51000024	R/W	定时器2计数缓冲寄存器	0x00000000
TCMPB2	0x51000028	R/W	定时器2比较缓冲寄存器	0x00000000

定时器2计数监视寄存器

寄存器	地址	R/W	描述	复位值
TCNTO2	0x5100002C	R	定时器2计数监视寄存器	0x00000000

相关寄存器介绍(续)

定时器3的寄存器

定时器3计数缓冲寄存器

定时器3比较缓冲寄存器

寄存器	地址	R/W	描述	复位值
TCNTB3	0x51000030	R/W	定时器3计数缓冲寄存器	0x00000000
TCMPB3	0x51000034	R/W	定时器3比较缓冲寄存器	0x00000000

定时器3计数监视寄存器

寄存器	地址	R/W	描述	复位值
TCNTO3	0x51000038	R	定时器3计数监视寄存器	0x00000000

相关寄存器介绍(续)

定时器4的寄存器

定时器4计数缓冲寄存器

寄存器	地址	R/W	描述	复位值
TCNTB4	0x5100003C	R/W	定时器4计数缓冲寄存器	0x00000000

定时器4计数监视寄存器

寄存器	地址	R/W	描述	复位值
TCNTO4	0x51000040	R	定时器4计数监视寄存器	0x00000000

定时器0计数缓冲寄存器——TCNTB0

名称	位索引	描述	初始值
定时器 0 计数缓冲寄存器	[15:0]	设置定时器 0 的计数缓冲器的值	0x0000

(0～65535)

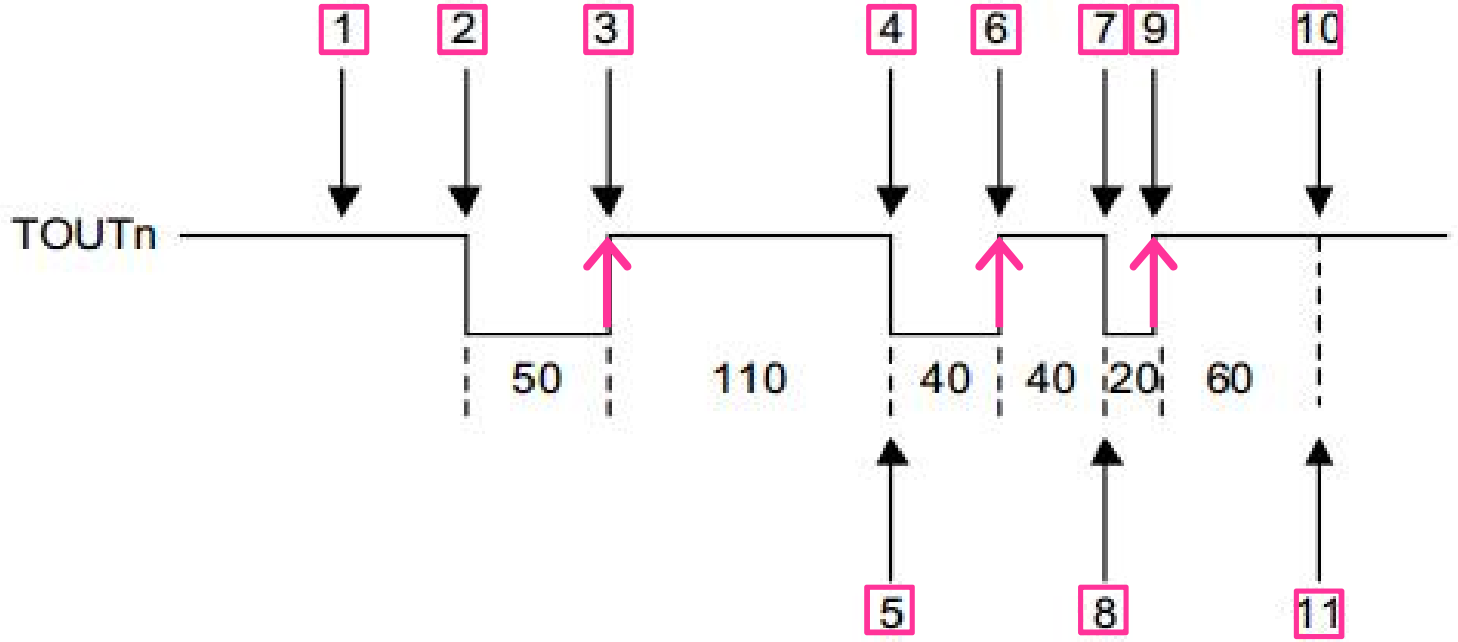
定时器0比较缓冲寄存器——TCMPB0

名称	位索引	描述	初始值
定时器 0 比较缓冲寄存器	[15:0]	设置定时器 0 的比较缓冲器的值	0x0000

定时器0计数监视寄存器——TCNTO0

名称	位索引	描述	初始值
定时器 0 计数监视寄存器	[15:0]	定时器 0 的计数监视值	0x0000

定时器操作实例1



定时器操作实例1

1.设置自动重载位=1，打开计数值自动重载；将160和110分别写入TCNTBn和TCMPBn；设置手动更新位=1，实现向TCNTn和TCMPn写入相应数值；设置反相位=0(不反相)；将80和40写入TCNTBn和TCMPBn，作为第二次重载的计数值。

2.设置启动位，启动定时器，同时TOUTn由高变低。定时器在等待一个微小时间后启动减1计数。此时，手动更新=0(无操作)，反相开关=0(不反相)，自动重载开关=1(自动重载)。

3.当TCNTn的值与TCMPn的值相同时，定时器输出引脚TOUTn的逻辑电平从低电平变为高电平。

4.当TCNTn的值减到0时，发出中断请求，同时TCNTBn的值被放入一个暂存器中。待下一个计数时刻，暂存器中的值(TCNTBn)被重载进TCNTn中。

5.在中断异常处理程序(中断服务程序)中，将80和60分别写入TCNTBn和TCMPBn，为下一次定时做好准备。

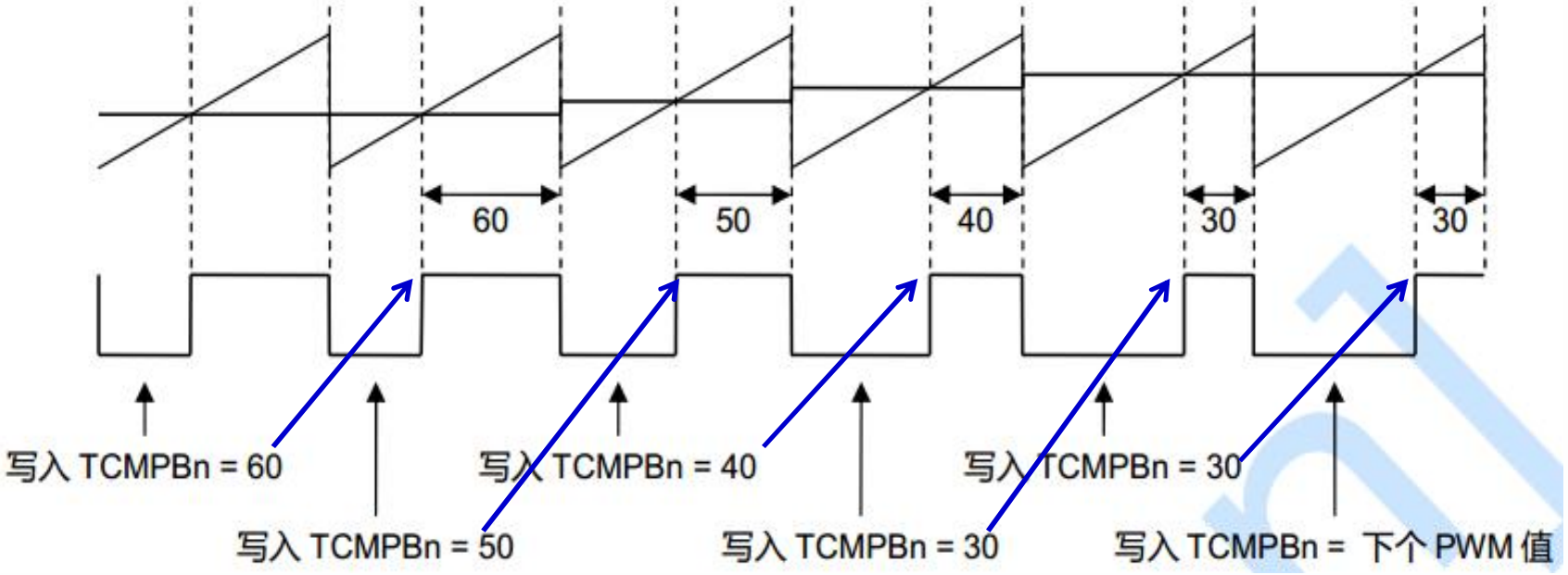
6.当TCNTn的值与TCMPn的值相同时，定时器输出引脚TOUTn的逻辑电平从低电平变为高电平。

定时器操作实例1

7.当TCNTn的值减到0时，发出中断请求，同时TCNTBn的值被放入一个暂存器中。待下一个计数时刻，暂存器中的值(TCNTBn)被重载进TCNTn中。
8.在中断异常处理程序(中断服务程序)中，禁止自动重载和中断请求允许，以关闭定时器。
9.当TCNTn的值与TCMPn的值相同时，定时器输出引脚TOUTn的逻辑电平从低电平变为高电平。
10.当TCNTn的值减到0时，由于自动重载已经禁止，因此TCNTn不会被自动重载计数值，定时器停止。
11.同样地，由于中断请求已被禁止，因此中断也不会触发。

定时器操作实例2

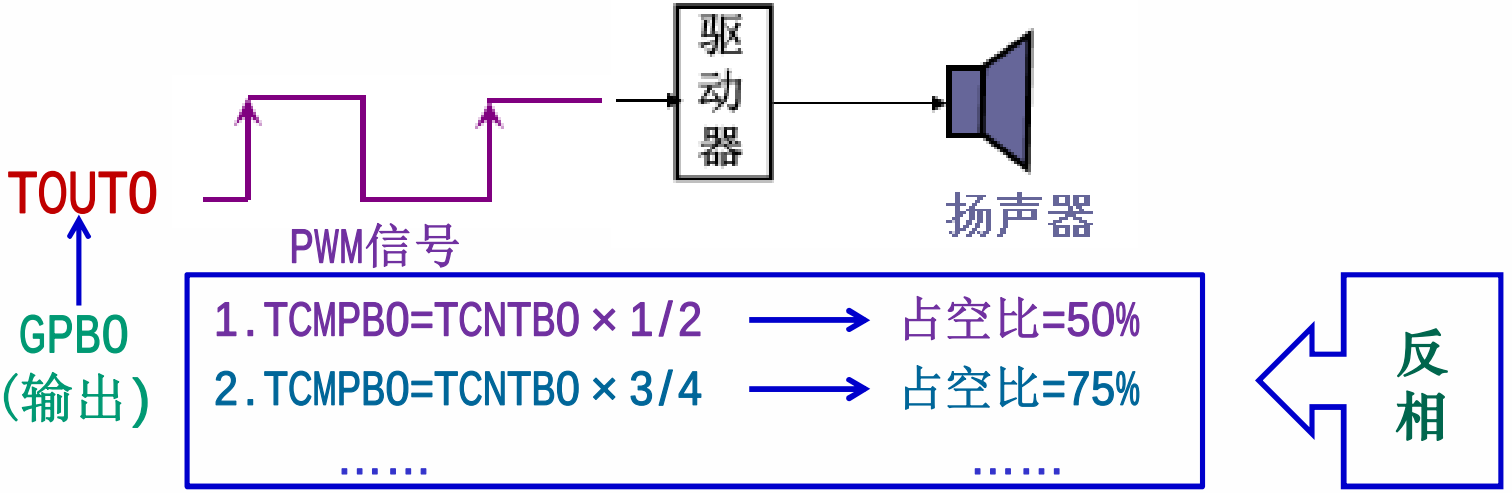
- 利用TCMPBn实现脉宽调制PWM



★某次定时（计数）中断中写入的计数参数，要在下下次定时启动时才有效。

定时器应用实例

- 利用定时器0的PWM功能实现扬声器(蜂鸣器)发声。



//1.配置GPB0为定时器0输出，并保持端口B的其它引脚输入输出不变。

GPB0	[1:0]	00 = 输入	01 = 输出	10 = TOUT0	11 = 保留

```
LDR R0,=0x56000010          ;端口B的配置寄存器GPBCON地址
LDR R1,[R0]
BIC R1,R1,#0x03             ;将寄存器的[1:0]=00
ORR R1,R1,#0x02             ;将寄存器的[1:0]=10
STR R1,[R0]
```

定时器应用实例(续)

//2.配置TCFG0和TCFG1，写入预分频和分频系数，以获取所需频率的定时器0输出信号TOUT0。

Prescaler 0	[7:0]	该 8 位决定了定时器 0 和 1 的预分频值

```
LDR R0,=0x51000000          ;定时器配置寄存器TCFG0地址
LDR R1,[R0]
BIC R1,R1,#0xFF             ;将寄存器的[7:0]=0x00
ORR R1,R1,#0x63             ;将寄存器的[7:0]=0x63(十进制99)
STR R1,[R0]
```

MUX 0	[3:0]	选择 PWM 定时器 0 的选通输入 0000 = 1/2	0001 = 1/4	0010 = 1/8	0011 = 1/16
		01xx = 外部 TCLK1			

```
LDR R0,=0x51000004          ;定时器配置寄存器TCFG1地址
LDR R1,[R0]
BIC R1,R1,#0x0F             ;将寄存器的[3:0]=0x0
ORR R1,R1,#0x02             ;将寄存器的[3:0]=0x2  （二进制0010）
STR R1,[R0]
```

定时器应用实例(续)

//3.配置TCNTB0和TCMPB0，写入计数初值和决定脉宽的比较值。

定时器 0 计数缓冲寄存器	[15:0]	设置定时器 0 的计数缓冲器的值

```
LDR R0,=0x5100000C      ;定时器0计数缓冲寄存器TCNTB0地址
MOV R1,#62500           ;将计算得到的计数初值写入TCNTB0
STR R1,[R0]
```

定时器 0 比较缓冲寄存器	[15:0]	设置定时器 0 的比较缓冲器的值

```
LDR R0,=0x51000010      ;定时器0比较缓冲寄存器TCMPB0地址
MOV R1,R1,LSR #1        ;将TCNTB0/2写入TCMPB0，设置PWM信号占空比50%
STR R1,[R0]
```

定时器应用实例(续)

//4.配置TCON，设置自动重载，并启动定时器0。

定时器 0 自动重载开/关	[3]	决定定时器 0 的自动重载开启或关闭 0 = 单稳态　　1 = 间隙模式（自动重载）
定时器 0 输出变相开/关	[2]	决定定时器 0 的输出变相开启或关闭 0 = 关闭变相　　1 = TOUT0 变换极性
定时器 0 手动更新（注释）	[1]	决定定时器 0 的手动更新 0 = 无操作　　1 = 更新 TCNTB0 和 TCMPB0
定时器 0 启动/停止	[0]	决定定时器 0 的启动或停止 0 = 停止定时器 0　　1 = 启动定时器 0

```
LDR R0,=0x51000008      ;定时器控制寄存器TCON地址
LDR R1,[R0]
ORR R1,R1,#0x02         ;TCON[1]=1：通过设置手动更新，实现首次计数值
                        ;的加载(从TCNTBn→TCNTn，从TCMPBn→TCMPn)。
STR R1,[R0]
BIC R1,R1,#0x02         ;TCON[1]=0：清除手动更新，下一次才能写入。
STR R1,[R0]
ORR R1,R1,#0x0D         ;TCON[3]=1：自动重载；TCON[2]=1：反相；
                        ;TCON[0]=1：启动定时器0。
STR R1,[R0]
```

TCON[2]：输出是否变相，取决于实际硬件电路需要

定时器应用实例(续)

//5.配置中断模式寄存器INTMOD，设置普通中断；配置中断屏蔽寄存器INTMSK，允许定时器0中断。

INT_TIMER0	[10]	0 = IRQ	1 = FIQ

INT_TIMER0	[10]	0 = 可服务	1 = 屏蔽

//6.在定时器0计数值达到0并触发中断后，在IRQ异常处理程序（中断服务程序）中读取中断挂起寄存器INTPND，判断是否是定时器0的中断。若是，根据需要修改计数初值或比较值，并写入TCNTBn和TCMPBn。

INT_TIMER0	[10]	0 = 未请求	1 = 请求

实时时钟

以S3C2440为例

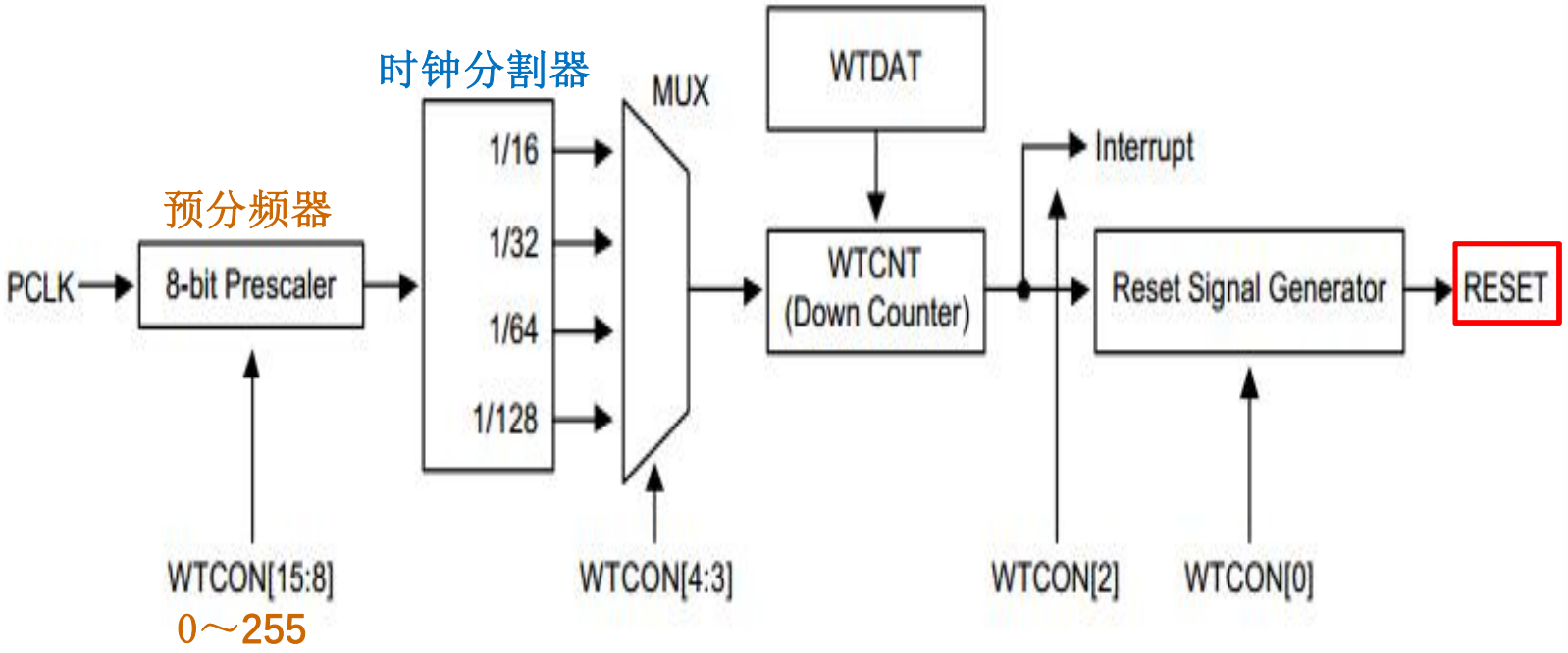
- RTC(Real Time Clock)用于提供年、月、日、时、分、秒、星期等实时时间信息。
  - ✓ 实时时间信息以8位BCD码表示，微处理器可以利用STRB/LDRB指令从相应的寄存器中读取。
- 通过微处理器的XTIrtc和XTOrtc引脚，引入32.768KHz的外部晶振。
- 在系统电源关闭的情况下，可以依靠备用电池工作。
- 在掉电或正常操作模式下，能够在被设定的时间内发出报警中断信号——闹钟功能。

看门狗定时器

以S3C2440为例

- WDT(WatchDog Timer)是一种用于当噪音或系统错误引起故障时，恢复微处理器操作的定时器。
- 本质上是一个16位内部定时器。当计数器值为0(即超时)时，通过触发中断服务，激活128个PCLK时钟周期的内部复位信号。
- 可用作一个普通的带中断请求(INT_WDT)的16位定时器。
- 只能使用PCLK时钟信号作为源输入时钟信号，且没有对外的输出信号引脚。
- 系统处于嵌入式ICE调试模式时，WDT必须无操作(自动关闭)。

看门狗定时器的内部结构



F_watchdog = PCLK / (Prescaler value + 1) / Division_factor

相关寄存器介绍

看门狗定时器控制寄存器

寄存器	地址	R/W	描述	复位值
WTCON	0x53000000	R/W	看门狗定时器控制寄存器	0x8021

看门狗定时器数据寄存器

寄存器	地址	R/W	描述	复位值
WTDAT	0x53000004	R/W	看门狗定时器数据寄存器	0x8000

看门狗定时器计数寄存器

寄存器	地址	R/W	描述	复位值
WTCNT	0x53000008	R/W	看门狗定时器计数寄存器	0x8000

➢ 看门狗定时器控制寄存器——WTCON

名称	位索引	描述	初始值
预分频	[15:8]	预分频值(0到255)	0x80
保留	[7:6]	要求工作时这两位必须为00	00
看门狗定时器	[5]	看门狗定时器的使能或禁止位 0 = 禁止；1 = 使能	1
时钟选择	[4:3]	设定时钟分割系数 00 = 1/16；01 = 1/32；10 = 1/64；11 = 1/128	00
中断使能/禁止	[2]	中断的使能或禁止位 0 = 禁止；1 = 使能	0
保留	[1]	要求工作时此位必须为0	0
复位使能/禁止	[0]	看门狗定时器复位输出的使能或禁止位 0 = 禁止看门狗定时器的复位功能； 1 = 看门狗超时时发出 S3C2440A 复位信号	1

➢ 看门狗定时器数据寄存器——WTDAT

名称	位索引	描述	初始值
计数重载值	[15:0]	看门狗定时器重载的计数值(0～65535)	0x8000

△计数值=所需时间间隔÷T_watchdog = 所需时间间隔×F_watchdog

➢ 看门狗定时器计数寄存器——WTCNT

名称	位索引	描述	初始值
计数值	[15:0]	看门狗定时器的当前计数值	0x8000

看门狗定时器的注意事项

- 看门狗定时器的中断与复位
  - ✓ 看门狗定时器中断是看门狗定时器作为普通定时器应用时，向微处理器发出内部中断请求；
  - ✓ 看门狗定时器复位是让微处理器复位，相当于重新启动所有程序。
- WTDAT存放WDT的计数初值，相当于普通定时器的TCNTBn。
- WTCNT是WDT的减1计数器，相当于普通定时器的TCNTn(程序不可直接访问)。
- 不同于普通定时器，WTDAT的值在WDT初始使能时，不能自动装载到WTCNT中，所以必须要给WTCNT设定一个初始值。

看门狗定时器程序实例

- 与定时器程序类似，此处略。

- 5.1 通用输入输出端口(GPIO)
- 5.2 中断系统
- 5.3 定时部件
- 5.4 通用异步收发器(UART)*

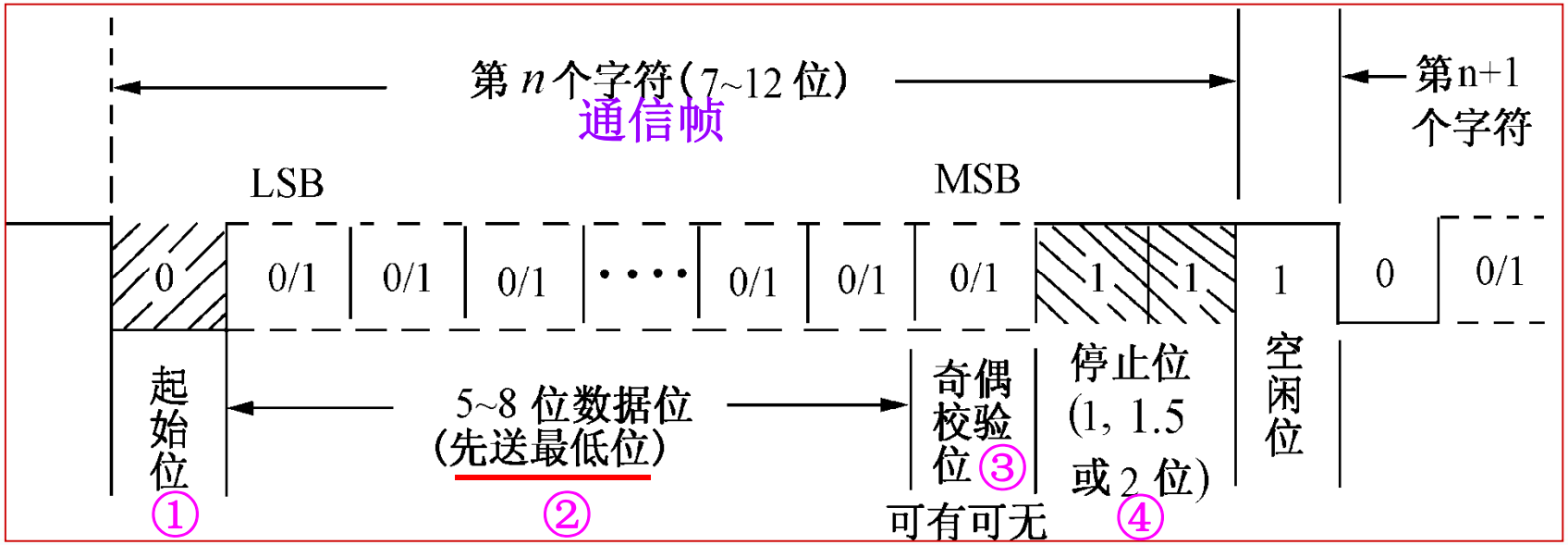
实际应用时，结合教材、微处理器手册自行学习！

有关串行通信的基本概念

异步串行通信

- 以一个字符为传输单位。通信中两个字符间的时间间隔是不固定的，但在同一个字符中的两个相邻位间的时间间隔是固定的。——收、发双方使用同频独立时钟源。
- 异步通信的每个字符（通信帧）由四部分组成。

通信帧格式



有关串行通信的基本概念

- 波特率（Baud Rate）
  - ✧定义：单位时间（每秒）内传送二进制数据的位数。
  - ✧单位：波特（位/秒，bps）
  - ✧计算方法：波特率=数据传送速率(字符/秒)×字符位数
    - 【举例】一个串行字符由1个起始位、7个数据位、1个奇偶校验位和1个停止位组成，每秒传送120个字符，则数据传送的波特率为：

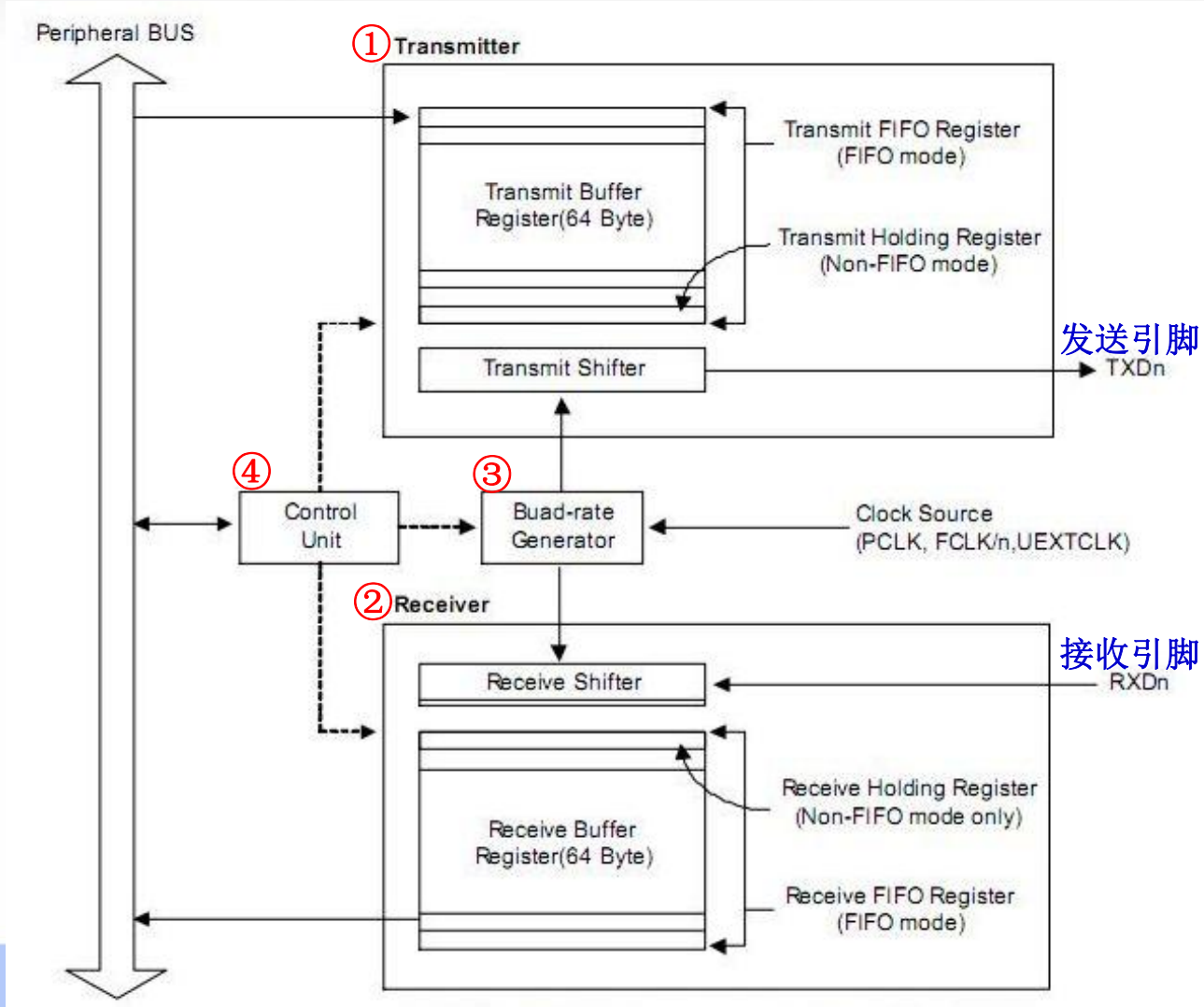
120字符/秒×10=1200位/秒=1200波特（即每位占用0.833毫秒）

UART(Universal Asynchronous Receiver-Transmitter)

以S3C2440为例

- 提供三个独立的通道0、1、2。
- 最大波特率115.2Kbps——使用系统时钟。
- 通过产生中断或DMA请求来传输数据。
- 通信帧：一个或两个停止位，5位、6位、7位、8位数据位和奇偶校验位。

UART通道的内部结构



①发送器

②接收器

③波特率发生器

④控制单元

UART的操作

- 数据发送和接收
- 波特率产生
- 中断/DMA请求的产生
- 自动流控制
- 回送模式
- 红外模式

UART的接口寄存器

控制类

UART线性控制寄存器	UART控制寄存器	UART FIFO控制寄存器	UART MODEM控制寄存器
• ULCON0	• UCON0	• UFCON0	• UMCON0
• ULCON1	• UCON1	• UFCON1	• UMCON1
• ULCON2	• UCON2	• UFCON2	

状态类

UART接收/发送状态寄存器	UART错误状态寄存器	UART FIFO状态寄存器	UART MODEM状态寄存器
• UTRSTAT0	• UERSTAT0	• UFSTAT0	• UMSTAT0
• UTRSTAT1	• UERSTAT1	• UFSTAT1	• UMSTAT1
• UTRSTAT2	• UERSTAT2	• UFSTAT2	

数据缓冲类

UART发送缓冲寄存器	UART接收缓冲寄存器
• UTXH0	• URXH0
• UTXH1	• URXH1
• UTXH2	• URXH2

波特率

UART波特率分频寄存器
• UBRDIV0
• UBRDIV1
• UBRDIV2

UART线性控制寄存器——ULCONn

名称	位索引	描述	初始值
Reserved	[7]	保留。	0
Infrared Mode	[6]	确定是否使用红外模式。 0 = 正常（普通）模式操作；1 = 红外接收/发送模式。	0
Parity Mode	[5:3]	在UART发送/接收操作中，确定奇偶校验产生和检验的类型。 0xx = 无奇偶校验；100 = 奇校验；101 = 偶校验； 110 = 固定/检验奇偶校验为1； 111 = 固定/检验奇偶校验为0。	000
Number of Stop Bit	[2]	确定用于结束帧信号的停止位的个数。 0 = 1位停止位；1 = 2位停止位。	0
Word Length	[1:0]	确定发送或接收每帧数据的位数。 00 = 5位；01 = 6位；10 = 7位；11 = 8位。	00

UART控制寄存器——UCONn

名称	位索引	描述	初始值
FCLK divider	[15:12]	确定UART时钟源选为FCLK/n时的分频器值。 ● n由UCON0[15:12]，UCON1[15:12]，UCON2[14:12]来决定。 ● UCON2[15]是FCLK/n 时钟使能位：0 =禁止；1 =使能。 ● 设置n从7到21，使用UCON0[15:12],设置n从22到36，使用UCON1[15:12],设置n从37到43，使用UCON2[14:12]。	0000
Clock Selection	[11:10]	选择波特率所用的时钟源。 00,10 = PCLK；01 = UEXTCLE；11 = FCLK/n。	00
Tx Interrupt Type	[9]	确定发送中断请求信号的类型。 0 = 边沿触发；1 = 电平触发。	0
Rx Interrupt Type	[8]	确定接收中断请求信号的类型。 0 = 边沿触发；1 = 电平触发。	0
Rx TimeOut Enable	[7]	确定接收超时使能。 0 = 不使能；1 = 使能。	0
Rx Error UART Status Interrupt Enable	[6]	确定接收错误状态使能。 0 = 不使能；1 = 使能。	0

UART控制寄存器——UCONn(续)

名称	位索引	描述	初始值
Loopback Mode	[5]	确定是否采用回送模式。 0 = 正常操作模式；1 = 回送模式。	0
Send Break Signal	[4]	确定通信中断信号。 0 = 正常操作；1 = 发送通信中断信号。	0
Transmit Mode	[3:2]	确定用哪种方式写数据到UART发送缓冲寄存器。 00 = 无效；01 = 中断请求模式； 10 = DMA0请求（UART0）或DMA3（UART2）； 11 = DMA1请求（UART1）。	00
Receive Mode	[1:0]	确定用哪种方式读UART接收缓冲寄存器中的数据。 00 = 无效；01 = 中断请求模式； 10 = DMA0请求（UART0）或DMA3（UART2）； 11 = DMA1请求（UART1）。	00

UART FIFO控制寄存器——UFCONn

名称	位索引	描述	初始值
Tx FIFO Trigger Leve	[7:6]	确定发送FIFO寄存器的触发选择。 00 = 空；01 = 16字节；10 = 32字节；11 = 48字节。	00
Rx FIFO Trigger Leve	[5:4]	确定接收FIFO寄存器的触发选择。 00 = 1字节；01 = 8字节；10 = 16字节；11 = 32字节。	00
Reserved	[3]	保留。	0
Tx FIFO Reset	[2]	确定复位FIFO之后自动清除。 0 = 正常；1 = 复位清除发送FIFO。	0
Rx FIFO Reset	[1]	确定复位FIFO之后自动清除。 0 = 正常；1 = 复位清除接收FIFO。	0
FIFO Enable	[0]	确定FIFO使能。 0 = 不使能；1 = 使能。	0

UART 接收/发送状态寄存器——UTRSTATn

名称	位索引	描述	初始值
Transmitter empty	[2]	当发送缓冲寄存器中没有有效数据，并且发送移位寄存器为空时，该位自动置为1。 0 = 非空；1 = 发送器空（发送缓冲和移位寄存器）。	1
Transmit buffer empty	[1]	当发送缓冲寄存器为空时，该位自动置为1。该位为0时，发送缓冲寄存器非空。 0 = 发送缓冲器非空；1 = 发送缓冲器空。	1
Receive buffer data ready	[0]	当接收缓冲寄存器接收到一个数据时，该位自动置为1。该位为0时，接收缓冲寄存器空。 0 = 接收缓冲器空；1 = 接收缓冲器非空。	0

UART 错误状态寄存器——UERSTATn

名称	位索引	描述	初始值
Break Detect	[3]	当接收到断点信号时，该位自动置为1。 0 = 未接收到断点中断；1 = 接收到断点(已请求中断)。	0
Frame Error	[2]	当接收出现帧错误时，该位自动置为1。 0 = 无帧错误；1 = 帧错误(已请求中断)。	0
Parity Error	[1]	当接收出现奇偶校验错误时，该位自动置为1。 0 = 无奇偶校验错误；1 = 奇偶校验错误(已请求中断)。	0
Overrun Error	[0]	当接收出现溢出错误时，该位自动置为1。 0 = 无溢出错误；1 = 溢出错误(已请求中断)。	0

UART FIFO状态寄存器——UFSTATn

名称	位索引	描述	初始值
Reserved	[15]	保留。	0
Tx FIFO Full	[14]	发送期间，若发送FIFO满，该位自动置为1。 0 = 0字节 ≤ Tx FIFO 数据 ≤ 63字节； 1 = 发送FIFO满。	0
Tx FIFO Count	[13：8]	发送FIFO中的数据个数。	000000
Reserved	[7]	保留。	0
Rx FIFO Full	[6]	接收期间，若接收FIFO满，该位自动置为1。 0 = 0字节 ≤ Rx FIFO 数据 ≤ 63字节； 1 =接收FIFO满。	0
Rx FIFO Count	[5:0]	接收FIFO中的数据个数。	000000

UART 发送缓冲寄存器——UTXHn

名称	位索引	描述	初始值
TXDATAn	[7:0]	UARTn要发送的8位数据。	—

UART 接收缓冲寄存器——URXHn

名称	位索引	描述	初始值
RXDATAn	[7:0]	UARTn接收到的8位数据。	—

UART 波特率分频寄存器——UBRDIVn

名称	位索引	描述	初始值
UBRDIVn	[15:0]	确定波特率分频值，UBRDIVn > 0。 使用UEXTCLK 作为源时钟时，可以设置为‘0’。	—

波特率除数的计算方法

- UBRDIVn = (int) (源时钟 ÷ (波特率 × 16)) –1
  - ✓ 源时钟：PCLK、FCLK/n或UEXTCLK。
  - ✓ UBRDIVn的范围应该是从1到2^{16}-1。当使用小于PCKL的外部时钟UEXTCLK时应该设置为0。
- 举例：
  - ✓ 若波特率为115200bps，且UART的源时钟是40MHz，则

UBRDIVn =(int)(40000000 ÷ (115200 × 16)) - 1

= (int)(21.7) - 1

= 21 - 1

= 20

UART实例

➢使用UART0通道自收自发。波特率115200bps，使用PCLK时钟且频率为50MHz。

- 程序如下：

```
#define ULCON0 (*(volatile unsigned *)0x50000000)          //UART线性控制寄存器
#define UCON0 (*(volatile unsigned *)0x50000004)           //UART控制寄存器
#define UFCON0 (*(volatile unsigned *)0x50000008)          //FIFO控制寄存器
#define UTRSTAT0 (*(volatile unsigned*)0x50000010)         //UART状态寄存器
#define UBRDIV0 (*(volatile unsigned *)0x50000028)         //波特率分频寄存器
#define UTXH0 (*(volatile unsigned *)0x50000020)           //发送缓冲寄存器
#define URXH0 (*(volatile unsigned *)0x50000024)           //接收缓冲寄存器
```

UART实例(续)

```
int TSmain()
{
    char buf;
    ULCON0 &=0xFFFFFF00;
    ULCON0 |=0x03;                          //1 位起始位，8 位数据位
    UCON0 =0x0805;                          //串口时钟PCLK，中断方式接收/发送
    UBRDIV0 =0x1A;                          //波特率115200
    while(1)
    {
        if(UTRSTAT0 & 0x01)                 //接收是否完毕，=1 接收结束
        {
            buf=URXH0;                      //读取数据
            while(!(UTRSTAT0 & 0x04));      //是否允许发送，=1 允许
            UTXH0=buf;
        }
    }
    return 0;
}
```

UART实例(续)

```
//主函数.
AREA |DATA|,CODE,READONLY
ENTRY
ldr r13,=0x1000
IMPORT TSmain      //IMPORT伪指令，告诉编译器标号TSmain来自外部文件。
b TSmain
END
```

The End!

本章要点

- 熟悉通用IO端口的功能和特点。掌握使用其实现基本数据输入输出的方法，包括相关特殊功能寄存器控制字的分析方法及使用汇编指令实现控制的方法。
- 弄清中断控制器的功能及相关特殊功能寄存器的用途。重点掌握普通外部中断的处理流程及使用汇编程序实现中断初始化和中断处理及中断返回的方法。
- 围绕S3C2440微处理器芯片，弄清其内部定时器的构成及工作过程。能够根据定时需求，初步设计定时器相关参数并使用汇编指令配置相关特殊功能寄存器。
- 理解看门狗定时器的工作原理。