第五章 GPIO 模块

1 GPIO 基础

1.1 GPIO 概念

General Purpose Input Output （通用输入/输出）简称为 GPIO，通俗地说，就是一些引脚，可以通过它们输出高低电平或者通过它们读入引脚的状态是高电平或是低电平。I/O 接口是一个微控制器必须具备的最基本外设功能。通常在 ARM 里，所有 I/O 都是通用的，称为 GPIO（General Purpose Input/Output）。

GPIO 口的使用非常广泛。它还可以模拟一些简单的通信接口，如 SPI、I^2C 等。

TM4C123GH6PM 微控制器包括六组 I/O 端口，分别为 PA0-PA7， PB0-PB7， PC0-PC7、PD0-PD7、PE0-PE5、PF0-PF4，这六组 I/O 口共占用 43 个引脚（TM4C123GH6PM 共 64 个引脚）。

GPIO 模块有以下特点：

高度灵活的复用功能，可以配置成普通输入输出或其他外设功能（复用）；

可通过高级外设总线（APB）和高速外设总线（AHB）访问 A-F 端口；

可编程控制的 GPIO 中断；

可进行位的读写操作；

可以用来启动一个 ADC 采样序列或 μDMA 传输；

休眠模式下，引脚状态可被保留；

可编程控制的 GPIO 配置。

1.2 GPIO 的引脚及电气特性

表 1 列出了 TM4C123GH6PM 的 GPIO 引脚工作电压范围。

表 1 GPIO 工作电压范围

参数	参数名称	数值		单位
		最小值	最大值	
VDD	VDD 电压	0	4	V
VDDA	V 电压 b DDA	0	4	V
VBAT	VBAT 电池电压	0	4	V
VBATRMP	VBAT 电池电压斜率时间	0	0.7	V/μs
VIN_GPIO	GPIO 输入电压	-0.3	5.5	V
	PD4, PD5, PB0, PB1 的输入电压（配置为 GPIO）	-0.3	VDD + 0.3	V
IGPIOMAX	每个 PIN 的最大电流值	-	25	mA
TS	未供电情况下最大允许值	-65	150	° C
TJMAX	供电情况下最大允许值	-	150	° C

TM4C123GH6PM 芯片的主电源 VDD 电压一般取标准值 3.3V，当 GPIO 引脚配置为输入功能时，除了 PD4、PD5、PB0 和 PB1 最高可承受 3.6V 电压外，其他所有 GPIO 都可以承受 5V 电压。

特殊的引脚有：

不可屏蔽中断（NMI）引脚：PF0 和 PD7。

被保护的引脚：NMI 引脚和用作 debug 的 4 个引脚 PC0-PC3。

如果要使用特殊的引脚作其他功能，需要先对其进行解锁操作，具体操作见后续保护配置介绍。

要正确使用一个外设，首先要了解其主要功能和硬件结构。

2 GPIO 的功能

嵌入式程序员要更好地使用 GPIO，须先了解其功能配置。因为一般 MCU 的引脚都是有限的，而要利用这些引脚实现更多的功能，生产厂商在设计 MCU 芯片时会将引脚设计为复用模式，例如 TM4C123GH6PM 芯片的 GPIO 引脚除了可以配置为普通的 GPIO 功能以外，还具备其他复用的外设功能，如可配置为 I^2C，SSI，PWM，ADC 等外设功能，具体功能可以通过寄存器进行配置，每个引脚在某个时刻只能设置为其中一种功能，要么为普通 GPIO 功能，要么为复用的外设功能，复位时，默认为普通的 GPIO 功能。

在使用 GPIO 之前，需对引脚进行一定的设置，下面先介绍 GPIO 的功能分类，接着介绍如何配置为相应的功能，本节介绍的配置方法依然是寄存器配置法，事实上库函数也可实现相应功能（其实也是配置寄存器），常用库函数见第 5 节介绍。

2.1 普通的 GPIO 功能

普通的 GPIO 具有数字输入和数字输出功能。几乎每个引脚都可以通过软件设置为数字输入或输出功能，如读取按键的高低电平，即是数字输入功能，点亮 LED 则为数字输出功能。

2.2 复用的外设功能

复用为外设功能时，GPIO 模块的引脚也可以被配置为输入或输出功能，此时的输入和输出功能可分为模拟输入，数字输入和数字输出功能。

（1） 输入功能

输入功能分为模拟输入和数字输入。

模拟输入：当 GPIO 模块复用为模拟比较器 AC 功能时，可将某些引脚设置为该 AC

模块的模拟输入引脚，当复用为模数转换 ADC 模块时，可作为模拟信号的输入引脚。标明为 AINx 的模拟输入功能引脚不能承受+5V 的电压，此时外部模拟电压输入范围一般为 0~3.3V，而其余模拟功能的引脚可以耐压+5V。

数字输入即将外部的高或低电平输入给引脚。

（2） 输出功能

输出功能有**推挽输出**和**开漏输出** 2 种模式。

推挽输出功能时可以直接输出高/低电平。不管 GPIO 模块是被设置为普通 GPIO 功能还是复用外设功能，当引脚被设置为输出功能时，默认为推挽输出。推挽模式不适合作为输入。

开漏结构即漏极开路，漏极没有上拉电阻，相当于普通的三极管的集电极开路（OC），因此只能输出低电平，如果要得到高电平，在实际应用时通常都要外接合适的上拉电阻（通常采用 4.7～10kΩ）；开漏能够方便地实现“线与”逻辑功能，即多个开漏的管脚可以直接并在一起（不需要缓冲隔离）使用，并统一外接一个合适的上拉电阻，就自然形成“逻辑与”关系。

配置开漏功能时，引脚设置为数字输出功能时才有效。当引脚复用为 I^2C 功能时，一般要将数据传输线配置为开漏模式。

弱上/下拉功能是引脚内部接入了上/下拉电阻，因此当引脚为高电平时，其电压是等于系统电压 3.3V 的，而开漏结构，是外接上拉电阻的，上拉的电平可以为系统电压 3.3V，也可以上拉电阻接到外接的 5V 电压上，使得高电平电压输出为 5V。

按照以上介绍，将 GPIO 功能分类如下：

图 1 GPIO 的功能分类

2.3 GPIO 内部结构

按照内部结构图 2 来看，输入和输出控制是不同的通路。

数字输入功能均可配置为弱上/下拉功能，不能配置为开漏和推挽模式。

数字输出功能可配置为开漏输出和推挽输出功能。当引脚配置为数字输出功能时，如果没有打开开漏功能，则默认为推挽输出功能。

无论是数字输入还是输出，都可以通过配置寄存器设置引脚的驱动电流大小，如 2mA、4 mA、8 mA 等。没有配置的情况下，默认为 2mA 大小。

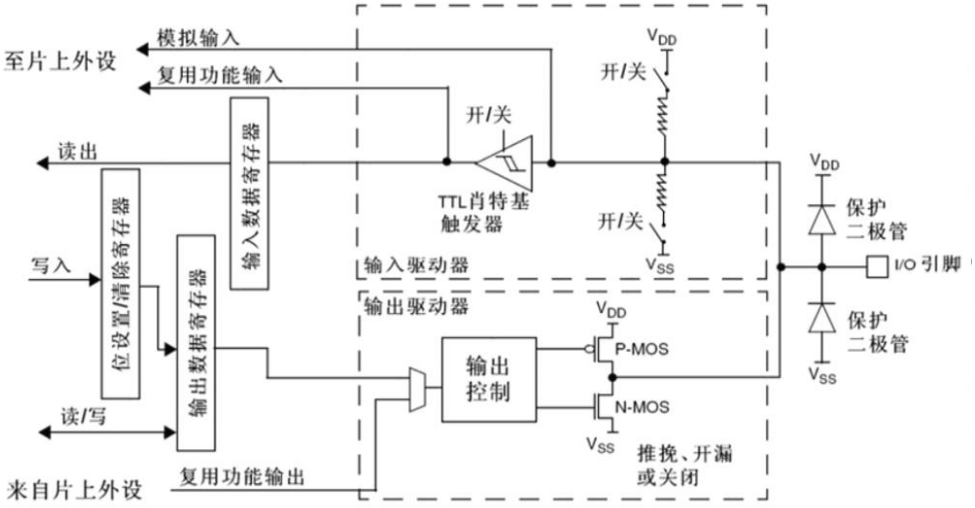

图 2 GPIO 内部结构图

下面详细介绍 GPIO 的内部结构图。TM4C123GH6PM 芯片的 GPIO 引脚具有输入和输出功能，下面分别介绍这几种输入输出模式。

2.3.1 输入功能—数字输入

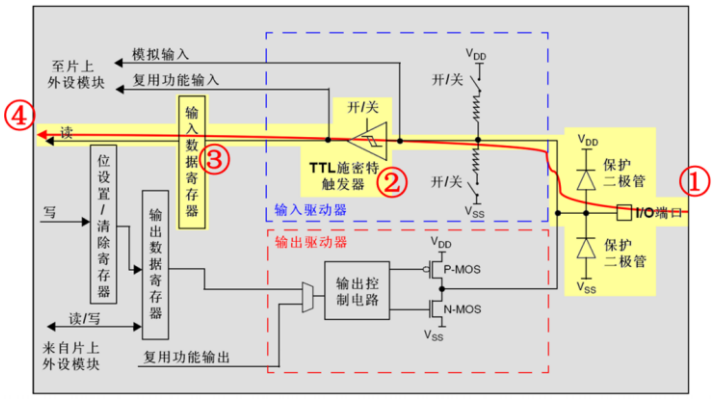

图 3 数字输入功能结构框图

数字输入功能的信号流向如下：

1)外部通过 GPIO 口输入电平，既无上拉也无下拉电阻；

2)传输到施密特触发器(此时施密特触发器为打开状态)；

3)继续传输到输入数据寄存器；

4)CPU 通过读输入数据寄存器实现读取外部输入电平值。

在输入模式下可以读取外部输入电平。

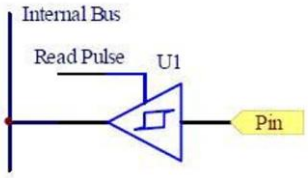

图 4 三态缓冲器

如图 4 所示，输入模式的结构比较简单，就是一个带有施密特触发输入（Schmitt-triggered input）的三态缓冲器（U1），并具有很高的输入等效阻抗。施密特触发输入的作用是能将缓慢变化的或者是畸变的输入脉冲信号整形成比较理想的矩形脉冲信号。执行 GPIO 管脚读操作时，在读脉冲（Read Pulse）的作用下会把管脚（Pin）的当前电平状态读到内部总线上（Internal Bus）。在不执行读操作时，外部管脚与内部总线之间是隔离的。

GPIO 内部具有钳位保护二极管，如图 1 所示。其作用是防止从外部管脚 Pin 输入的电压过高或者过低。VDD 正常供电是 3.3V，如果从 Pin 输入的信号（假设任何输入信号都有一定的内阻）电压超过 VDD 加上二极管的导通压降（假定在 0.6V 左右），则上面的二极管导通，会把多余的电流引到 VDD，而真正输入到内部的信号电压不会超过 3.9V。同理，如果从 Pin 输入的信号电压比 GND 还低，则由于下面的二极管的作用，会把实际输入内部的信号电压钳制在－0.6V 左右。

（1）输入上拉模式

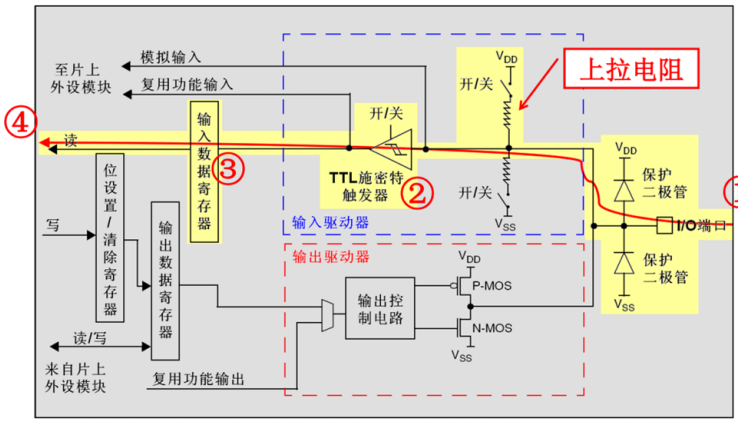

图 5 输入上拉模式结构框图

外部输入通过上拉电阻，施密特触发器存入输入数据寄存器被 CPU 读取。

（2）输入下拉模式

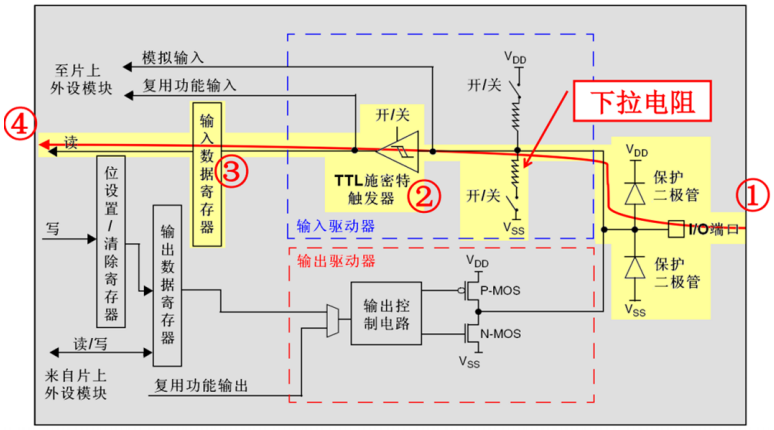

图 6 输入下拉模式结构框图

外部输入通过下拉电阻，施密特触发器存入输入数据寄存器，被 CPU 读取。

2.3.2 输入功能—模拟输入

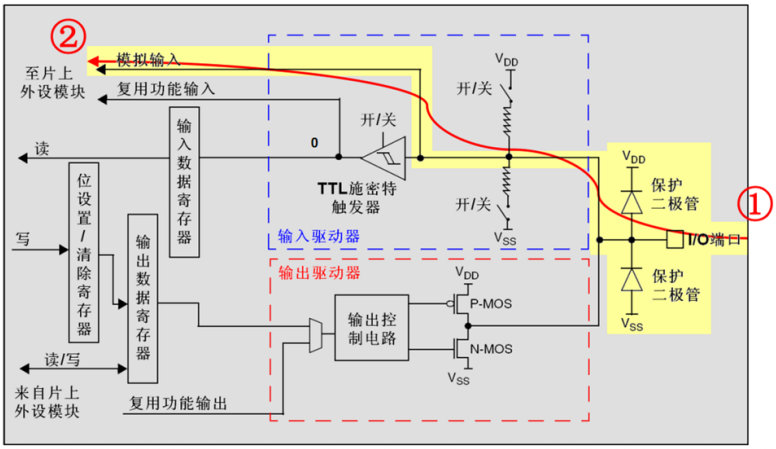

图 7 模拟输入结构框图

上拉电阻和下拉电阻部分均为关闭状态；

施密特触发器为截止状态。

通过模拟输入通道输入到 CPU。

IO 口外部模拟电压输入范围一般为 0~3.3V。

2.3.3 输出功能-开漏输出模式

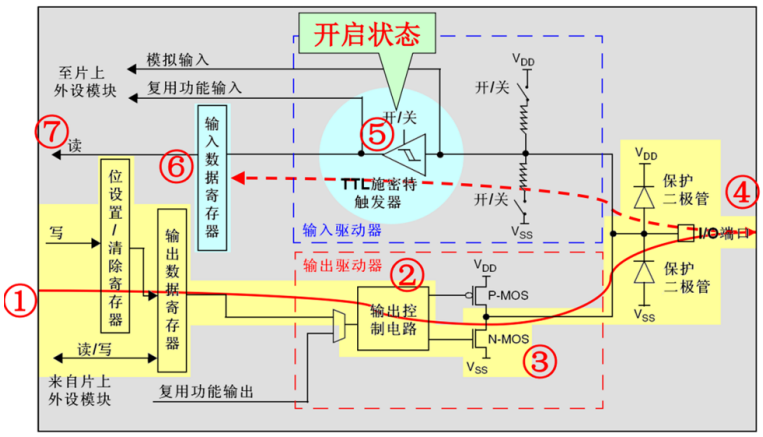

图 8 开漏输出模式结构框图

1) CPU 写入映射到输出数据寄存器；

2) 联通到输出控制电路；

3) 输出控制电路电平输出进入 N-MOS 管

输出控制电路输出为 0：N-MOS 截止，IO 口电平由外部上拉/下拉决定。

输出控制电路输出为 1：N-MOS 开启，IO 口输出低电平。

开漏输出的实际作用就是一个开关，输出“1”时断开、输出“0”时连接到 GND（有一定内阻）。回读功能：读到的是输出锁存器的状态，而不是外部管脚 Pin 的状态。因此开漏输出模式是不能用来输入的。

2.3.4 输出功能-推挽输出模式

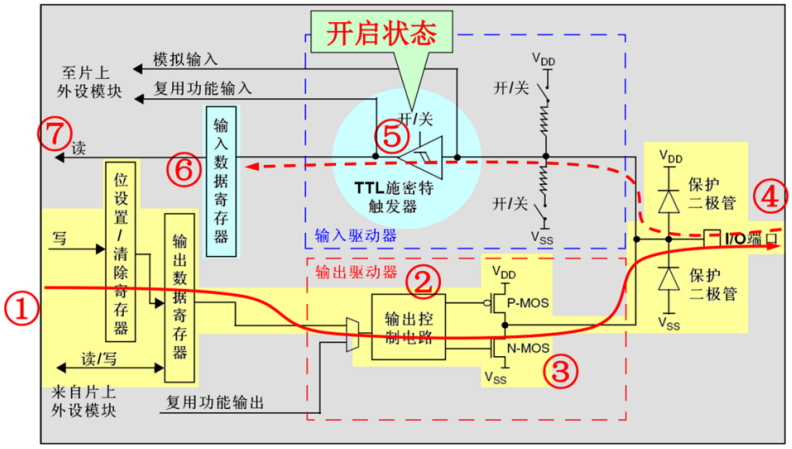

图 9 推挽输出模式结构框图

与开漏输出相比较：输出控制寄存器部分相同，输出驱动器部分加入了 P-MOS 管部分，当输出控制电路输出 1 时：P-MOS 管导通 N-MOS 管截止，被上拉到高电平，IO 口输出为高电平 1，当输出控制电路输出 0 时：P-MOS 管截止 N-MOS 管导通，被下拉到低电

平，IO 口输出为低电平 0，同时 IO 口输出的电平可以通过输入电路读取。

在推挽输出模式下，GPIO 还具有回读功能，实现回读功能的是一个简单的三态门。注意：执行回读功能时，读到的是管脚的输出锁存状态，而不是外部管脚 Pin 的状态。

2.3.5 推挽输出和开漏输出的区别

推挽输出：可以输出强高/强低电平，可以连接数字器件。

开漏输出：只能输出强低电平(高电平需要依靠外部上拉电平拉高)，适合做电流型驱动，吸收电流能力较强(20mA 之内)。

2.4 功能配置

GPIO 模块的控制寄存器分组如下：

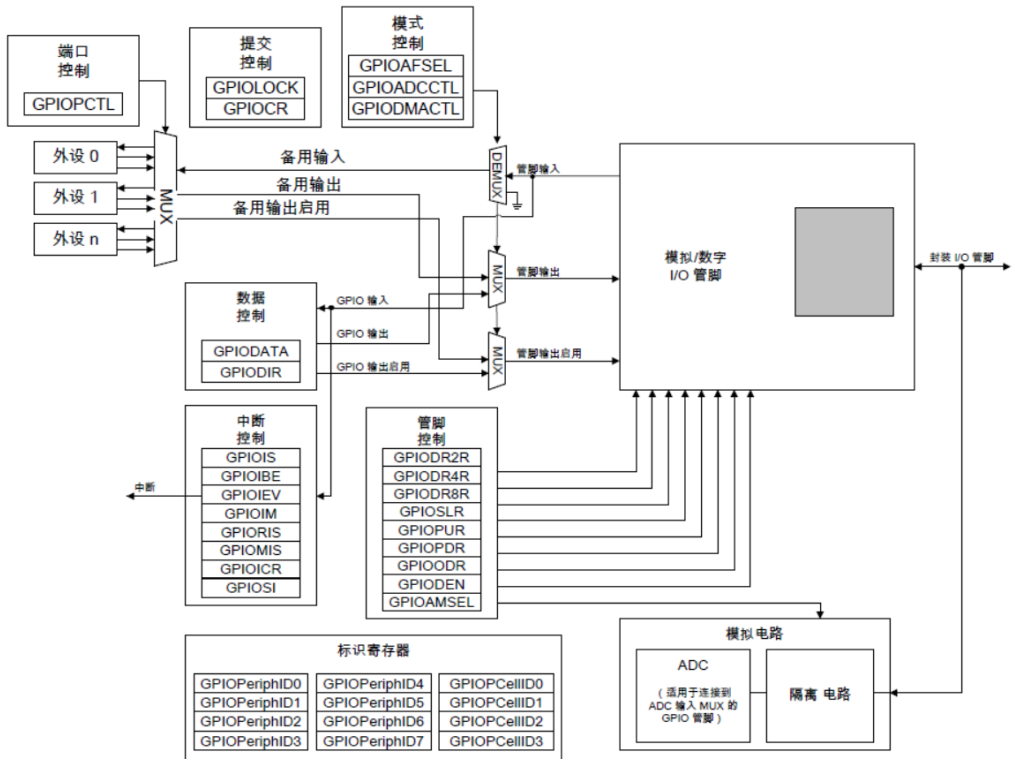

图 10 GPIO 的结构图

GPIO 的功能配置共分为 5 组：

1、 模式配置：用来配置 GPIO 模块是普通 GPIO 功能还是复用的外设功能；

2、 方向配置：用来配置引脚是输入还是输出功能；

3、 引脚配置：配置是否开漏，是否上下拉，驱动电流大小等；

4、 中断配置：设置引脚是否允许产生中断以及中断产生的条件等，将在中断部分介绍。

5、 保护配置：有些特殊引脚具有保护功能，如果需要使用这些引脚，需对其进行解锁操

作。

下面针对这 5 组配置里除了中断配置外的其他 4 组配置进行说明：

2.4.1 模式配置

模式配置即是配置 GPIO 模块是普通 GPIO 功能还是复用的外设功能，默认为普通 GPIO 功能。与模式配置有关的 2 个主要寄存器为模式选择寄存器 GPIOAFSEL 和端口选择寄存器 GPIOPCTL。

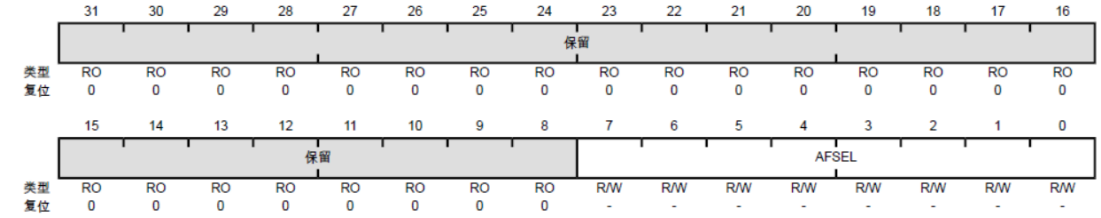

图 11 模式选择寄存器 GPIOAFSEL 的位域图

每个 GPIO（A-F）模块都有一个**模式选择寄存器 GPIOAFSEL**，该寄存器的低 8 位分别设置 8 个引脚的功能。如果某位为 0 则表示该引脚用作普通 GPIO 功能，为 1 则表示该引脚被设置为复用的外设功能。

复用的外设功能，又分为 2 种，一种为数字功能，一种为模拟功能，数字功能取决于 GPIOPCTL 寄存器。

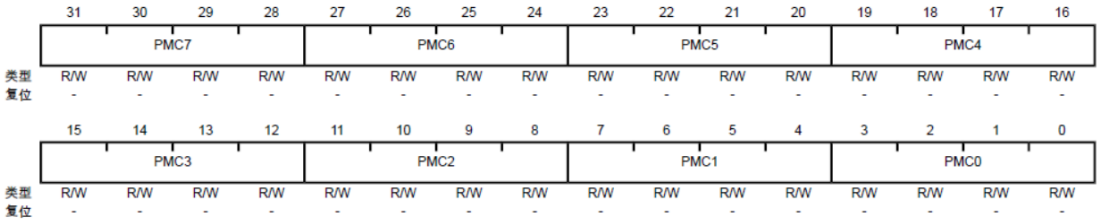

图 12 GPIOPCTL 寄存器的位域图

每个GPIO（A-F）模块都有一个**端口选择寄存器GPIOPCTL，**其中PMC0-PMC7分别代表0-7引脚的外设功能。每个PMCx占4位，可以将某个引脚配置为16种不同的编码来代表不同的外设功能。

表 2 GPIO 引脚的复用外设功能

IO	管脚	模拟功能	数字功能（GPIOPCTL PMCx 位域编码）[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	37	-	U0Rx	-	-	-	-	-	-	-	-	-	-
PA1	38	-	U0Tx	-	-	-	-	-	-	-	-	-	-
PA2	39	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-

例如，要将 PA 模块的 PA0 配置为外设 U0Rx 功能，则需要配置 PA 模块的 **GPIOPCTL 寄存器的 PMC0 域。该域有 4 位，可以代表 15 种数字功能（1-15）**，应将 PMC0 域的编码配置为 0X1（4 位二进制为 0001），代表 PA0 被配置为第 1 种外设功能，即

U0Rx 功能。要将 PA2 配置为外设 SSI0Clk 功能，应将 PA 模块的 **GPIOPCTL 寄存器的** PMC2 域的编码配置为 0X2（4 位二进制为 0010）。**更多数字外设功能配置可参考《tm4c1231h6pge 中文数据手册》第 1059 页表 20-5 GPIO 管脚和复用功能。**

当引脚为数字功能时，还需配置数字使能寄存器 GPIODEN。当某位为 0 时，表示该引脚不使能数字功能，当某位为 1 时，表示该引脚使能数字功能。

还有一种模拟功能（模拟比较器和 ADC 功能）的配置，若将模拟选择寄存器 GPIOAMSEL 设置为 1，则该引脚被配置为模拟功能。因模拟功能只有输入功能，没有输出功能，因此此时也应将方向寄存器 GPIODIR 设置为输入。

注意：如果一个引脚用作 ADC 输入功能，则该引脚硬件上是经过了内部隔离电路接入到 ADC 模块的。

2.4.2 方向配置

方向寄存器 GPIODIR 用来设置引脚的方向为输入还是输出。

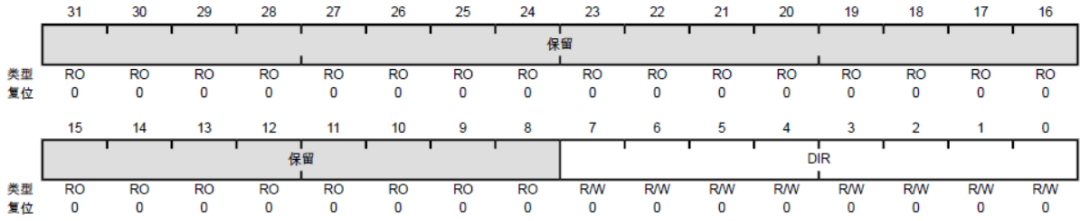

图 13 GPIODIR 寄存器的位域图

每个 GPIO（A-F）模块都有一个方向**寄存器 GPIODIR**，该寄存器的低 8 位分别设置 8 个引脚的方向。如果某位为 0 则表示该引脚用作输入功能，为 1 则表示该引脚被设置为输出功能。复位时默认为 0。

2.4.3 引脚配置

引脚配置分为开漏配置，上下拉配置和驱动电流配置。

1、 开漏配置，由 GPIOODR 寄存器来设置是否打开开漏功能，每个模块有一个 GPIOODR 寄存器，该寄存器的低 8 位用来决定该模块的哪个引脚打开开漏功能。

2、 上/下拉配置，由 GPIOPUR 寄存器配置上拉功能，GPIOPDR 寄存器配置下拉功能，配置方法跟开漏配置类似。

3、 驱动电流配置，由 2mA/4 mA /8 mA 寄存器来配置引脚电流的大小。

以上寄存器的配置比较简单，因此各寄存器的定义请参考数据手册，不再累述。

2.4.4 保护配置

一般情况下，GPIO 各引脚相关寄存器默认复位值为 0，即默认为普通 GPIO 功能，但下表中的引脚除外。它们的复位值如表 2 所示。

表 3 具有非 0 复位值的 GPIO 管脚

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	$I^{21}C0$	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0

这些 GPIO 引脚的功能在复位时是复用外设功能的，如果要将其改为其他功能，需要重新对它们进行配置。其中 PC[3:0]是被保护的，配置前须对其进行解锁操作。

PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

PD7和PF0复位时是默认被配置为GPIO功能的，它们是NMI（不可屏蔽中断）引脚，并且它们被保护锁定了，为了防止软件对它们进行意外编程。如果要把他们配置成其他功能，这些管脚必须进行解锁，与其相关的寄存器为GPIOLOCK和GPIOCR，寄存器请参考中文数据手册620页。

例：PF0解锁

```
#define GPIO_LOCK_KEY           0x4C4F434B
#define GPIO_O_CR               0x00000524   // GPIO Commit
#define GPIO_O_LOCK             0x00000520   // GPIO Lock

HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = 0;
```

注意：解锁前，须设置系统时钟，并使能该端口。

2.4.5 功能配置步骤

一般引脚功能是按照**模式配置—>方向配置—>引脚配置**的步骤对引脚进行配置，当引脚被保护了，又需要修改其功能时，才使用保护配置。以上介绍了配置时所使用的寄存器，而库函数提供了以上相应的每一种功能，库函数将在后续介绍。表 3 为 GPIO 的一些常用的功能配置。

表 4 GPIO 端口配置示例

配置	GPIO 寄存器位值[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
数字输入(GPIO)	0	0	0	1	?	?	X	X	X	X
数字输出(GPIO)	0	1	0	1	?	?	?	?	?	?
开漏输出(GPIO)	0	1	1	1	X	X	?	?	?	?
开漏输入/输出 (I2CSDA)	1	X	1	1	X	X	?	?	?	?
数字输入 (定时器 CCP)	1	X	0	1	?	?	X	X	X	X
数字输出 (定时器 PWM)	1	X	0	1	?	?	?	?	?	?
数字输入/输出 (SSI)	1	X	0	1	?	?	?	?	?	?
数字输入/输出 (UART)	1	X	0	1	?	?	?	?	?	?
模拟输入（比较器）	0	0	0	0	0	0	X	X	X	X
数字输出（比较器）	1	X	0	1	?	?	?	?	?	?

a. X=忽略（无关位）

？代表是0或1由具体情况决定，取决于配置

3 GPIO 的数据读写

为了帮助提高软件的效率，GPIO 端口允许通过地址线的[9:2]位来表示是否允许对数据寄存器 GPIODATA 中的某位进行修改。因为在这种方式下，软件可以在不影响其他引脚状态下只修改独立的 GPIO 引脚。当屏蔽值的某位为 1 时，才对该引脚进行修改，为 0 的位不动。

例 1，将 0xEB 的值写入地址 GPIODATA+0x098 中，因 0x098 的[9:2]位这 8 位数据中只有第 1，2，5 位相应的位为 1（对应 0x098 的[9:0]位中的第 3，4，7 位），表示只允许修改 GPIODATA 的第 1，2，5 位，因此结果如图所示，最后只有第 1，2，5 位被修改了，u 表示写入后数据保持不变。

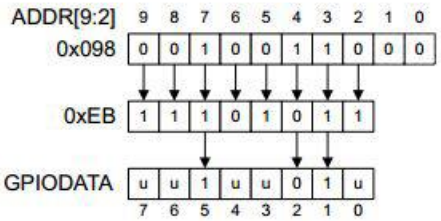

图 14 GPIODATA 写入例子

在读操作过程中，如果与数据位相关联的位被置为 1，那么读取该值，如果该位为 0，读出的数据为 0，而不管其实际值。例如，读取地址 GPIODATA+0x0C4 处的值，结果如图所示。这个例子显示了如何读取 GPIODATA 的第 5，4，0 位。

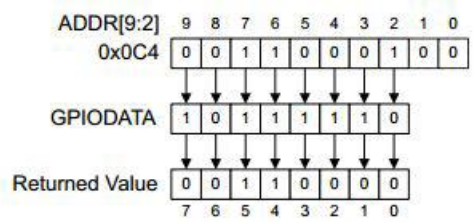

图 15　GPIODATA 读取例子

这些读写操作方式是库函数可以直接实现的。

例，**GPIOPinWrite**(GPIO_PORTA_BASE，GPIO_PIN_4，1<<4);

GPIOPinWrite的函数原型如下：

```
void
GPIOPinWrite(uint32_t ui32Port, uint8_t ui8Pins, uint8_t ui8Val)
{
    // Check the arguments.
    ASSERT(_GPIOBaseValid(ui32Port));
    // Write the pins.
    HWREG(ui32Port + (GPIO_O_DATA + (ui8Pins << 2))) = ui8Val;
}
```

例中调用函数时，GPIO_PIN_4 的值为 0x10，将 1<<4 后变为 0x10，将 0x10 写入地址 GPIO_O_DATA+0x10<<2=GPIO_O_DATA+0x040，写入后，只改变 PA4 的引脚状态为 1，其余位均不变。

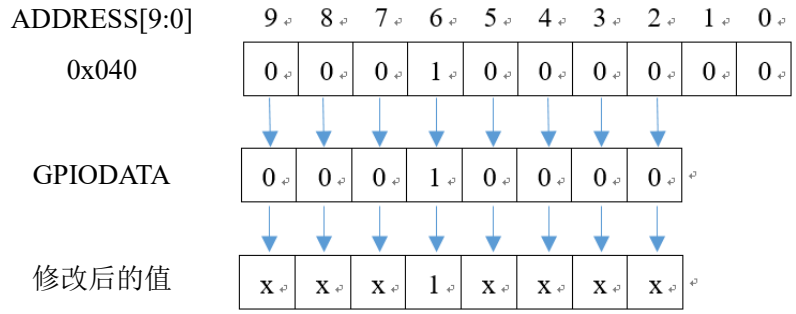

图 16 **GPIOPinWrite 写入示例**

4 GPIO 中断

TM4C123GH6PM的GPIO除了GPIO和复用功能外，还可以用作外部中断源。当某个GPIO口中断功能被使能后，该GPIO口就成为一个外部中断源。

4.1 中断的概念

中断是 CPU 实时地处理外部事件的一种内部机制。当某种外部事件发生时，CPU 的中断系统将迫使 CPU 暂停正在执行的程序，转而去进行中断事件的处理；中断处理完毕

后，又返回被中断的程序处，继续执行下去。

在没有干预的情况下，CPU 一般是在执行一个死循环程序，在封闭状态下自主运行，如果在某一时刻需要响应一个外部事件(比如有按键按下)，这时就会用到外部中断。具体来讲，外部中断就是微控制器的一个引脚发生了电平的变化(比如由高变低)，而通过捕获这个变化，微控制器内部自主运行的程序就会被暂时打断，转而去执行相应的中断处理程序，执行完后又回到原来中断的地方继续执行原来的程序的过程。这个引脚上的电平变化，产生了一个外部中断条件，从而申请了一个外部中断事件，而这个能申请外部中断的引脚就是外部中断的触发引脚。

4.2 中断的触发方式

中断有两种触发方式：电平触发方式和跳沿触发方式。

4.2.1 电平触发方式

电平触发方式可以分为低电平和高电平触发。当外部中断引脚上产生一个高电平或低电平时会触发中断，当外部中断源被设定为低电平触发方式时，在中断服务程序返回之前，外部中断请求输入必须无效(即变为高电平)，否则 CPU 返回主程序后会再次响应中断。所以电平触发方式适合于外部中断以低电平输入而且中断服务程序能清除外部中断请求源(即外部中断输入电平又变为高电平)的情况。

4.2.2 边沿触发方式

边沿触发又叫跳沿触发，分为上升沿和下降沿，即外部中断引脚产生 1 个低电平到高电平（上升沿）的变化或产生 1 个高电平到低电平（下降沿）的变化时触发中断。一般控制器分为这两种边沿触发方式，但 TIVA C 还支持双沿触发，即产生上升沿和下降沿时都会触发中断。

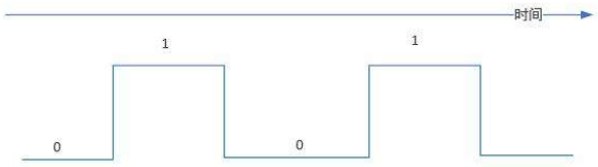

图 17 触发电平图

电平触发：只要产生相应的电平，就会触发中断，例如高电平触发，则每次到“1”，都会触发中断，以此类推。

边沿触发：只要产生相应的边沿，就会触发中断。例如下降沿触发，则只要电平从“1”跳为“0”，就会触发中断，以此类推。

4.3 GPIO 中断

TIVA 所有的 IO 口都可作为中断源(51 单片机只有 2 个端口可以作为外部中断输入)。

每个 GPIO 模块由 7 个寄存器来控制中断。这些寄存器用于选择中断源、极性，以及边沿属性。当一个或多个 GPIO 输入产生一个中断，一个中断输出为整个 GPIO 端口而被发送到中断控制器。对于边沿触发中断，软件必须清除中断以使能下一步中断。对于电平

触发的中断，外部必须保持恒定的电压，直至进入中断服务函数，当电平发生变化时，中断清零。

4.3.1 设置触发方式

GPIO 中断检测寄存器（GPIOIS）：设置电平触发或边沿触发；

GPIO 中断双沿寄存器（GPIOIBE）：设置单边沿或双边沿触发；

GPIO 中断事件寄存器（GPIOIEV）：设置高电平/上升沿触发或低电平/下降沿触发。

4.3.2 使能/禁止中断

GPIO 中断屏蔽（GPIOIM）寄存器：使能/禁止中断。当某位置位时，相应的位产生的中断允许被送到中断控制器。

4.3.3 中断状态查询

GPIO 原始中断状态寄存器（GPIORIS）：当一个管脚上发生中断时 GPIORIS 寄存器被置位。当 GPIO 中断屏蔽 (GPIOIM) 寄存器（请参考中文数据手册 602 页）中的某位置位时，相应的中断被送到中断控制器。读出来的某位为零则表示相应的位未发生中断。

GPIO 中断屏蔽状态寄存器（GPIOMIS）：如果该寄存器中的某个位置位，则说明相应的中断已经发送到中断控制器。如果某位清零，表示没有产生中断，或者中断被屏蔽。

换言之，当 GPIO 引脚产生了中断条件，则查询原始中断状态寄存器（**GPIORIS**），相应位为 1。如果它未被屏蔽（GPIOIM 设置为 1），则 CPU 会执行中断，查询**中断屏蔽状态寄存器（GPIOMIS），相应位为 1**，如果它被屏蔽了，则 CPU 不会执行中断，查询**中断屏蔽状态寄存器（GPIOMIS），相应位为 0。**

4.3.4 中断清除

GPIO 中断清除寄存器（GPIOICR）：对于边沿触发中断，向 GPIOICR 寄存器的 IC 位写 1 即可将 GPIORIS和 GPIOMIS 寄存器中相应的位清零。

4.3.5 中断配置示例

表 5 GPIO 中断配置例子

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value[a]							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge 1=level	X	X	X	X	X	0	X	X
GPIOIBE	0=single edge 1=both edges	X	X	X	X	X	0	X	X
GPIOIEV	0=Low level, or falling edge 1=High level, or rising edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0

a. X=Ignored (don't care bit)

该例子表示某端口的第 2 个引脚的中断触发方式为单边沿上升沿触发，并且该中断是使能的，未被屏蔽。

4.3.5 中断控制相关函数

1）GPIOIntTypeSet　　设置中断类型，即触发类型及触发方式

2）GPIOIntEnable　　GPIO 中断使能，使能某个端口的某个引脚的中断

3）GPIOIntRegister　　注册一个中断

4）GPIOIntClear　　清除中断

4.3.6 中断的配置步骤

中断的配置有 2 种方式，步骤如下：

1、设置中断类型

如：GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_FALLING_EDGE);　//设置 PF4 中断类型

2、GPIO 引脚中断使能

如：GPIOIntEnable(GPIO_PORTF_BASE,GPIO_PIN_4);　　//使能 PF4 中断

3、打开中断，注册中断服务函数

第三步有差异，有 2 种方法：

1、利用 GPIOIntRegister 函数注册中断，中断服务函数可以直接放在 main 文件中，只需将中断服务函数名作为 GPIOIntRegister 的参数注册即可。

如：GPIOIntRegister(GPIO_PORTF_BASE, Key2IntHandler);　//为 PF 注册一个中断处理句柄，该函数中其实也使能了 PORTF 中断。

2、常规设置

（1）main 中利用 IntEnable（）函数打 PORTF 开外设中断。如：

IntEnable(INT_GPIOF_TM4C123);

等同于

```
HWREG(NVIC_EN0) |= 0x40000000;          //使能 IRQ30，即 PORTF 中断
```

（2）将中断服务函数放在 startup_ccs.c 函数中，将相应的向量表中的默认的中断服务函数名修改成当前中断服务函数名。在 startup_ccs.c 的最前面，声明该中断服务函数。

5 GPIO 相关常用库函数

SysCtlPeripheralEnable　外设使能

GPIOPinTypeGPIOOutput　引脚设置为输出功能

GPIOPinTypeGPIOInput　引脚设置为输入功能

GPIODirModeSet　　设置引脚的方向和工作模式

GPIOPinConfigure　　设置 GPIO 引脚的复用功能

GPIOPadConfigSet　　设置引脚的驱动配置，如驱动电流、上拉下拉等。

GPIOPinWrite　　GPIO 写操作

GPIOPinRead　　GPIO 读操作

1）**SysCtlPeripheralEnable**　　外设使能

如：SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);使能 PA 引脚。

2) **函数 GPIOPinTypeGPIOOutput**

功能：将引脚配置为 GPIO 输出。

原型：void GPIOPinTypeGPIOOutput(uint32_t ui32Port， uint8_t ui8Pins)；

参数：ui32Port 为 GPIO 端口的基地址，ui8Pins 代表引脚位。

3) **函数** GPIOPinTypeGPIOInput

功能：将引脚配置为 GPIO 输入。

原型：void GPIOPinTypeGPIOIntput(uint32_t ui32Port， uint8_t ui8Pins)；

参数：ui32Port 为 GPIO 端口的基地址，ui8Pins 代表引脚位。

4) **函数 GPIODirModeSet**

功能：设置引脚的方向和模式。

原型：void GPIODirModeSet(uint32_t ui32Port，uint8_t ui8Pins，uint32_t ui32PinIO)；

参数：ui32Port 为端口地址，ui8Pins 代表引脚位，ui32PinIO 引脚方向或模式。

描述：该函数通过软件控制，将所选端口的特定引脚配置为输入或输出；或者配置引脚为外设功能。ui32PinIO 参数为以下值的枚举类型之一： GPIO_DIR_MODE_IN ，

GPIO_DIR_MODE_OUT，GPIO_DIR_MODE_HW。

GPIO_DIR_MODE_IN：表示引脚被编程为软件控制输入。

GPIO_DIR_MODE_OUT：表示引脚被配置为软件控制输出。

GPIO_DIR_MODE_HW：表示引脚为外设功能。

5) 函数 GPIOPinConfigure

功能：配置 GPIO 引脚的复用功能。

原型：void GPIOPinConfigure(uint32_t ui32PinConfig)；

参数：ui32PinConfig 是引脚配置值，其值只能为 GPIO_Pxx_xxx。

描述：该函数配置引脚复用，选择一个特定的 GPIO 引脚为某一外设功能。一次只能选择一个外设功能，并且每一个外设功能只能与单个 GPIO 引脚有关（尽管实际可以有很多功能与多个 GPIO 引脚有关）。为完整的配置一个引脚，GPIOPinType∗()函数也应该被调用。

6) **函数 GPIOPadConfigSet**

功能： 为引脚设置特定配置。

原型： void GPIOPadConfigSet(uint32_t ui32Port，uint8_t ui8Pins，

uint32_t ui32Strength，uint32_t ui32PinType)；

参数： ui32Port 为 GPIO 端口地址，ui8Pins 代表引脚位，ui32Strength 指定输出驱动力，ui32PinType 指定引脚类型。

描述： 该函数为所选 GPIO 端口的特定引脚配置驱动力和类型。

参数 ui32Strength 取以下值之一：GPIO_STRENGTH_2MA、GPIO_STRENGTH_4MA、GPIO_STRENGTH_8MA、GPIO_STRENGTH_8MA_SC，GPIO_STRENGTH_xMA 表示 2，4 或 8mA 的输出驱动力，GPIO_OUT_STRENGTH_8MA_SC 表示 8mA 带回转控制的输出驱动力。

参数 ui32PinType 取以下值之一： GPIO_PIN_TYPE_STD 、GPIO_PIN_TYPE_STD_WPU 、GPIO_PIN_TYPE_STD_WPD 、GPIO_PIN_TYPE_OD 、GPIO_PIN_TYPE_ANALOG ， GPIO_PIN_TYPE_STD 表示推挽式引脚，GPIO_PIN_TYPE_OD 表示开漏引脚，*_WPU 表示弱上拉，*_WPD 表示弱下拉，GPIO_PIN_TYPE_ANALOG 表示模拟输入。

7) **函数 GPIOPinWrite**

功能：向引脚写入一个值。

原型：void GPIOPinWrite(uint32_t ui32Port，uint8_t ui8Pins，uint8_t ui8Val)；

参数：ui32Port 为 GPIO 端口基地址，ui8Pins 代表引脚位，ui8Val 为写入引脚的值。

注意：对配置为输入的引脚写入没有影响。

8) **函数 GPIOPinRead**

原型：void GPIOPinWrite(uint32_t ui32Port，uint8_t ui8Pins)；

参数：ui32Port 为 GPIO 端口基地址，ui8Pins 代表引脚位。

注意：对配置为输入的引脚写入没有影响。

6 GPIO 配置步骤

GPIO 模块可以通过两个不同的总线访问，他们分别是传统的高级外设总线 (APB)和高级快速总线(AHB)，但是 AHB 提供了比 APB 更好的访问性能。但是这两种访问方式只能选择一种使用。

为指定 GPIO 端口哪个总线进行访问，通过设置 GPIOHBCTL 寄存器（请参考 221 页）中相应的位来控制，默认情况下为 APB 总线方式访问。

要完整配置 GPIO 引脚，请按以下步骤操作：

1. **启用端口时钟**（请参阅 289 页），通过调用 SysCtlPeripheralEnable（）函数实现；

2. **配置引脚模式**，通过调用 GPIOPinConfigure（）函数实现。

3. **设置引脚方向**，通过调用 GPIOPinTypeGPIOOutput（）或 GPIOPinTypeGPIOInput（）实现；注意这两个函数已默认**配置了引脚的驱动强度为 2mA、且打开了数字功能。**

4. **配置中断（可选）**，包括中断类型、事件和中断屏蔽等，通过调用中断相关函数实现；

5. **保护配置**（可选）。

7 例程

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_ints.h"
#include "inc/hw_types.h"
#include "inc/hw_nvic.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
//this code is the handler function for key
void KeyIntHandler()
{
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3,(0x08^GPIOPinRead(GPIO_PORTF_BA
SE,GPIO_PIN_3)));
    GPIOIntClear(GPIO_PORTF_BASE,GPIO_INT_PIN_4);
}

//this code is the main function
int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_4|SYSCTL_USE_OSC|SYSCTL_XTAL_16MHZ|SYSCTL_
OSC_MAIN);//设置系统时钟
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);//使能F模块
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,GPIO_PIN_3); //设置PF3为输出模式
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_4); //将PF4设置为输入模式
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);//设置PF4为弱上拉模式，驱动电流为2mA。
//  GPIOIntRegister(GPIO_PORTF_BASE, KeyIntHandler);//为F模块注册一个中断服务
函数，并且使能F模块中断
    GPIOIntTypeSet(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_FALLING_EDGE); //设置PF4
为下降沿触发
    GPIOIntEnable(GPIO_PORTF_BASE,GPIO_PIN_4); //使能PF4中断
    IntEnable(INT_GPIOF_TM4C123);//INT_GPIOF_TM4C123参数宏定义在
inc/hw_ints.h中，值为46；IntEnable函数在interrupt.c中
//  HWREG(NVIC_EN0) |= 0x40000000;        //与上一句话的作用等同，使能IRQ30，即
PORTF中断
    while(1)
    {
    }
```

}

该例程中，PF4 引脚作为输入引脚，外接按键，PF3 作为输出引脚，控制 LED 灯。PF4 引脚设置为中断，下降沿触发，当有按键按下时，产生下降沿，就会产生中断，此时中断服务函数中会将 PF3 的值进行取反。

因此，该例程实现的功能是：按一下 SW1 键，绿灯亮，再按一下 SW1 键，灯灭。如此循环。

8 预习习题

1、TM4C123GH6PM 有多少 GPIO 口？哪几个 GPIO 口只能承受 3.6V 电压？哪几个 GPIO 口是受保护的？为什么 PF0 在使用之前需要解锁？哪几个 GPIO 口是非零复位值的？

2、GPIOA 的中断向量地址是多少？

3、预习 GPIO 模块，观看视频实验三（上）和实验三（下）、TivaWare 时钟以及 GPIO 的介绍(下) ，并完成 workshop 实验 lab3，实验结果截图。

4、练习德研开发板的 DEMO 程序中的 LEDBink 文件夹下的 Blink 程序，及 LCD 文件夹下的 12864 程序，学习使用 GPIO 控制 LED 灯及 LCD 的方法。（德研开发板的 DEMO 程序是用 KEIL 开发软件写的，只需将 user 文件夹下的程序加到 CCS 工程中即可。）实验现象和结果拍照。

9 课堂任务

使用德研开发板，分别利用查询和中断 2 种方式实现按键控制的程序，实现 K3 按键控制 D2 灯亮，K4 按键控制 D2 灯灭。

TIVA C 开发板的电路图见文档《TM4C LaunchPad Workshop 实验练习步骤指南.pdf》.

德研板的电路图见文档《DY-TivaPB1 V3.0.PDF》和《DY-TivaPB2 V3.0.PDF》.

10 课后任务

设计综合任务中的 GPIO 模块任务，即综合任务中与按键和 LCD 显示有关的部分，要求拟出具体题目，课堂上完成。

参考网址：https://blog.csdn.net/techexchangeischeap/article/details/72569999