通用输入输出2

程晨闻

东南大学电气工程学院

➢ 作业

– 1. 开发板上的TM4C1294芯片是如何获得3.3V的电压的?

DEBUG_VBUS（5V）经过JP1跳线与VBUS相连，经过限流开关，变成+5V；然后经过TPS73733和跳线JP3，变为+3.3V；+3.3V经过JP2，变为MCU3.3，给芯片供电。

– 2. 有哪些方法将开发板上的TM4C1294芯片复位?

按键复位，TARGET_RESET复位。

– 3. 开发板上的四个分别由PN0,PN1,PF4,PF0引脚控制的LED灯亮起来时，LED上面的电流大约是多少?

(3.3 – 0.7)V/330 A= 7.9mA。

– 4. TM4C1294芯片刚上电后，系统时钟的频率是_**16MHz**_。

➢ 作业

– 5. 运行SysCtlDelay(120000000)后，实际延时了多长时间?

这个延时程序里会执行三个空指令，每个指令需要时间1/120M s，参数是120M，因此延时时间是120M*1/120M*3s=3s。

– 6. project0.c程序中SysCtlDelay(ui32SysClock/6); 这段代码是延迟了多长时间? **0.5s。**

– 7. 用memory Browser查看内存，

地址0x00000000的内容是 0x2000036C，这代表什么意义?

地址0x00000004的内容是 0x00000FD9，这代表什么意义?

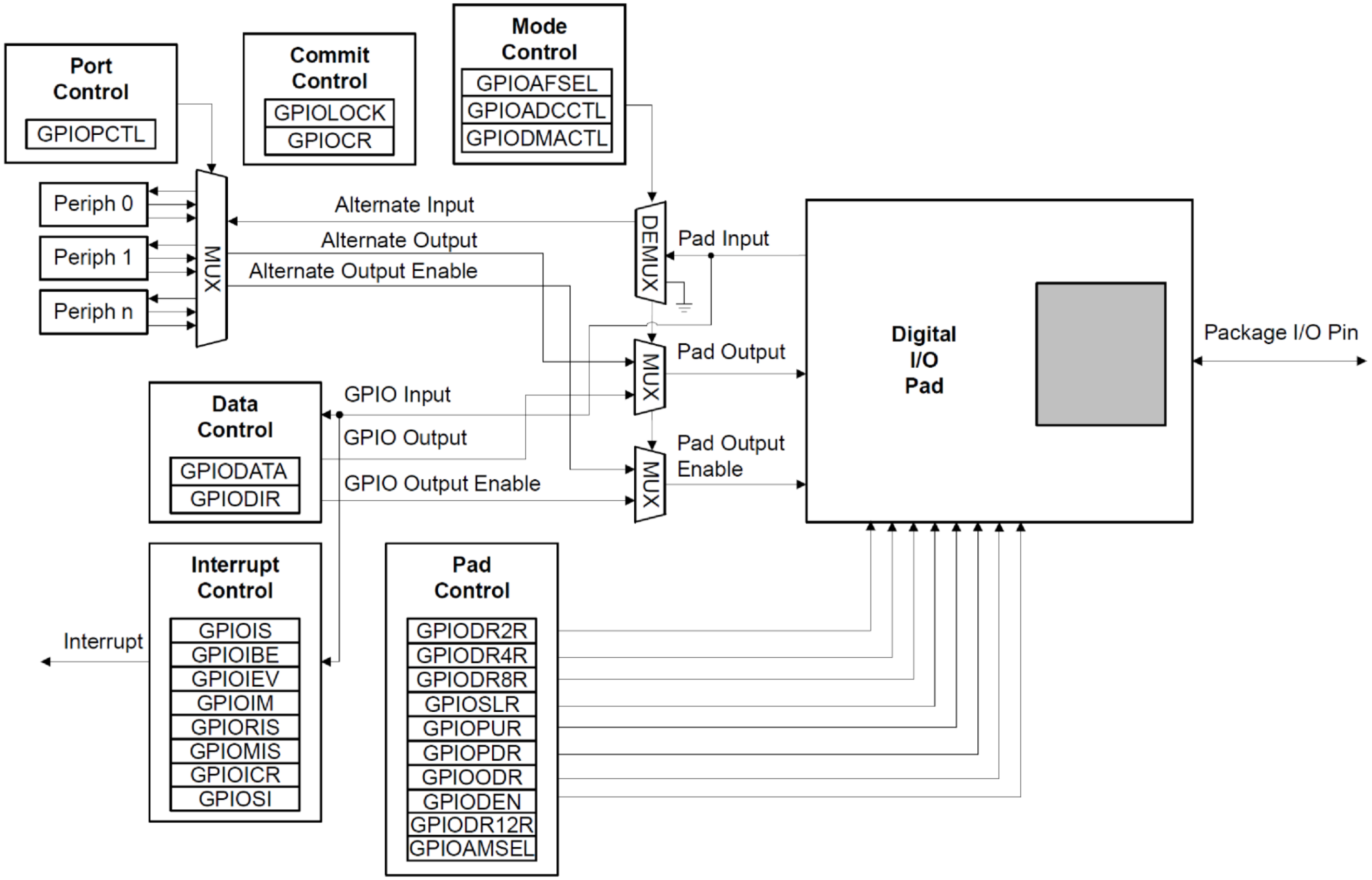
复位后，0x2000036C加载到堆栈指针SP中，0x00000FD9加载到程序计数器PC中。

知识回顾

- **端口（Port）**
  - CPU和外设进行通信的媒介
  - TM4C1294有15个端口，共90个引脚
- **引脚（Pin）**
  - MCU与外界连接的独立导线，隶属于某个端口
  - 可设置为多种功能（多功能复用）
  - 输入方式（浮空输入，上拉输入，下拉输入，模拟输入）
  - 输出方式（推挽输出，开漏输出）

通用输入输出接口（GPIO）

➢ TM4C1294的GPIO



➢ 外设寄存器

– 控制引脚的具体功能

– 外设寄存器可以通过系统总线访问，在系统总线上，有确定的地址

– 端口的控制寄存器组的地址，由基地址和偏移地址组成

– 实际地址为：基地址+偏移地址（各个端口配置寄存器组的基地址不同，具体功能的偏移地址相同）

– 有两条系统总线可以与外设通信，分别是APB和AHB（AHB比APB快）

– hw_memmap.h中，使用宏定义，定义了端口基地址

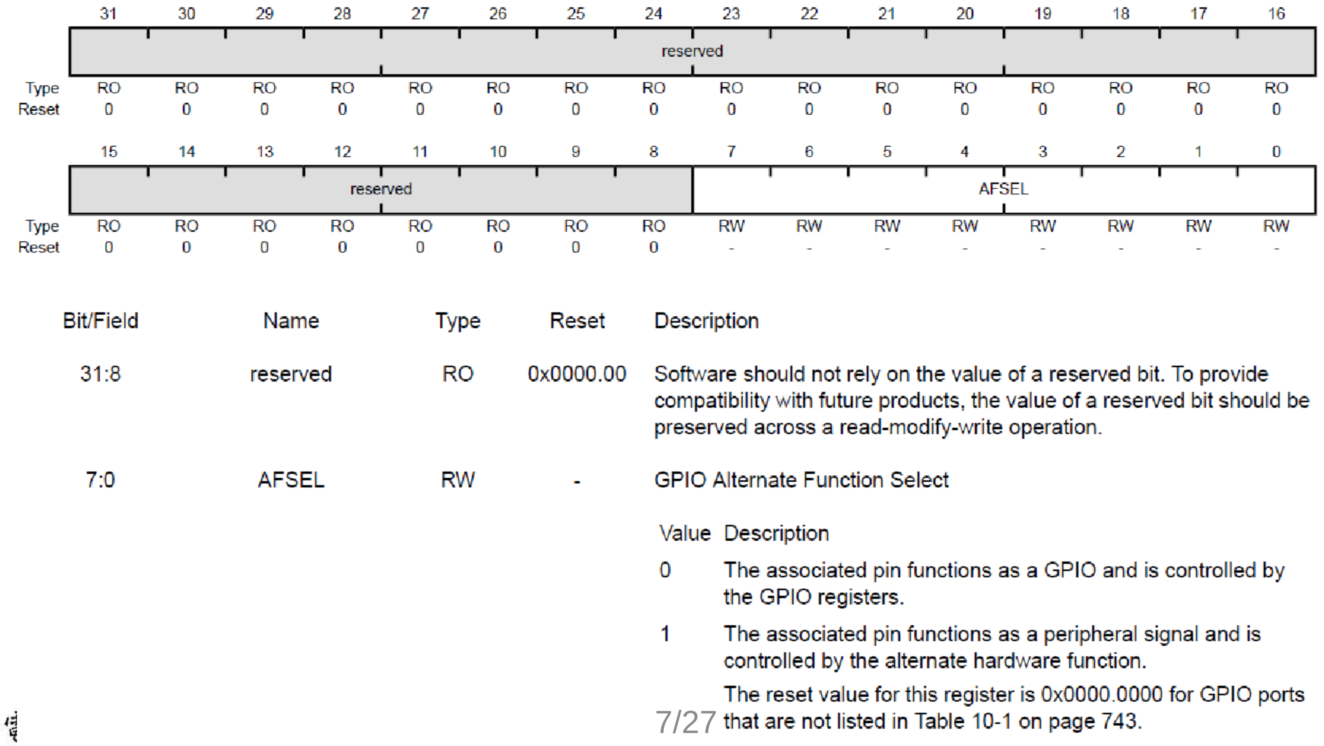
– hw_gpio.h中，定义了偏移地址

➢ 复用选择寄存器GPIOAFSEL

– 只有低八位有实际作用，用于控制该端口的八个引脚是否启用复用功能，每一位控制一个引脚

– 0为不复用，1为复用

– 复用的功能，由GPIOPCTL决定



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								AFSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	AFSEL	RW	-	GPIO Alternate Function Select

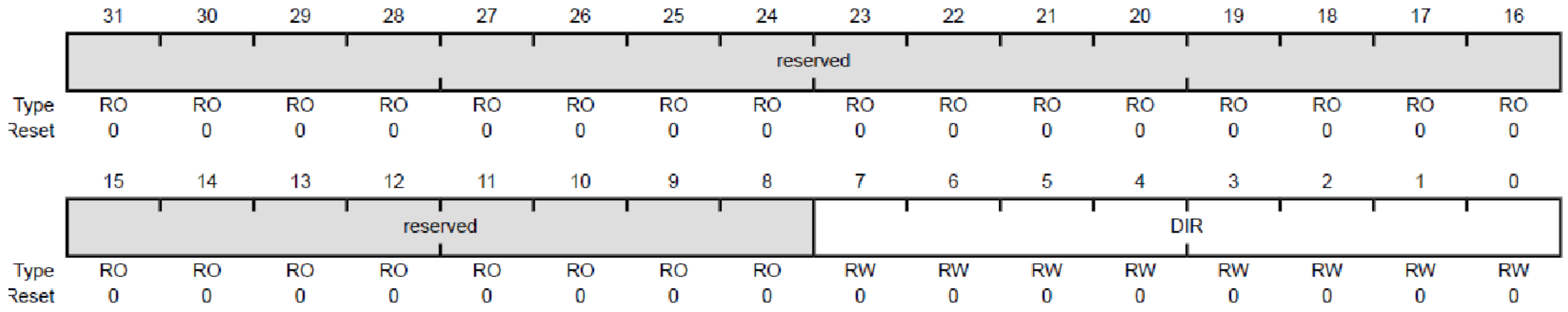
Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 743.

➢ 方向寄存器GPIODIR

– 低8位有效

– 0位输入，1位输出



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	RW	0x00	GPIO Data Direction

Value	Description
0	Corresponding pin is an input.
1	Corresponding pins is an output.

方向寄存器GPIODIR

– 对引脚单独控制，利用“driverlib/gpio.h”中对引脚做的宏定义

```
#define GPIO_PIN_0                 0x00000001  // GPIO pin 0
#define GPIO_PIN_1                 0x00000002  // GPIO pin 1
#define GPIO_PIN_2                 0x00000004  // GPIO pin 2
#define GPIO_PIN_3                 0x00000008  // GPIO pin 3
#define GPIO_PIN_4                 0x00000010  // GPIO pin 4
#define GPIO_PIN_5                 0x00000020  // GPIO pin 5
#define GPIO_PIN_6                 0x00000040  // GPIO pin 6
#define GPIO_PIN_7                 0x00000080  // GPIO pin 7
```

如果要将PORTn的Pin3设置为输出：

HWREG(GPIO_PORTN_BASE + GPIO_O_DIR)= GPIO_PIN_3;

也可以用或操作，控制设置多个引脚

如果要将PORTn的Pin0和Pin3都设置为输出

HWREG(GPIO_PORTN_BASE + GPIO_O_DIR)= GPIO_PIN_0 | GPIO_PIN_3;

端口方向控制函数

- void GPIOPinTypeGPIOInput (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeGPIOOutput (uint32_t ui32Port, uint8_t ui8Pins)
- void GPIOPinTypeGPIOOutputOD (uint32_t ui32Port, uint8_t ui8Pins)
- 这三个函数已经被封装在driverlib库中
- TivaWare提供了这三个函数的源代码，位于C:\ti\TivaWare_C_Series-2.2.0.295\driverlib\gpio.c和同目录下的gpio.h中
- SW-TM4C-DRL-UG-2.2.0.295.pdf文档对函数的使用做了详细说明

通用输入输出接口（GPIO）

➢ 端口方向控制函数

```
void
GPIOPinTypeGPIOInput(uint32_t ui32Port, uint8_t ui8Pins)
{
    // Check the arguments.
    ASSERT(_GPIOBaseValid(ui32Port));
    // Make the pin(s) be inputs.
    GPIODirModeSet(ui32Port, ui8Pins, GPIO_DIR_MODE_IN);
    // Set the pad(s) for standard push-pull operation.
    GPIOPadConfigSet(ui32Port, ui8Pins, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD);
}
```

```
#define GPIO_DIR_MODE_IN        0x00000000  // Pin is a GPIO input
#define GPIO_DIR_MODE_OUT       0x00000001  // Pin is a GPIO output
```

➢ 端口方向控制函数

```
void
GPIODirModeSet(uint32_t ui32Port, uint8_t ui8Pins, uint32_t ui32PinIO)
{
    // Check the arguments.
    ASSERT(_GPIOBaseValid(ui32Port));
    ASSERT((ui32PinIO == GPIO_DIR_MODE_IN) ||
           (ui32PinIO == GPIO_DIR_MODE_OUT) ||
           (ui32PinIO == GPIO_DIR_MODE_HW));
    // Set the pin direction and mode.
    HWREG(ui32Port + GPIO_O_DIR) = ((ui32PinIO & 1) ?
                                (HWREG(ui32Port + GPIO_O_DIR) |
ui8Pins) :
                                (HWREG(ui32Port + GPIO_O_DIR) &
~(ui8Pins)));
    HWREG(ui32Port + GPIO_O_AFSEL) = ((ui32PinIO & 2) ?
                                  (HWREG(ui32Port + GPIO_O_AFSEL) |
                                   ui8Pins) :
                                  (HWREG(ui32Port + GPIO_O_AFSEL) &
                                   ~(ui8Pins)));

}
```

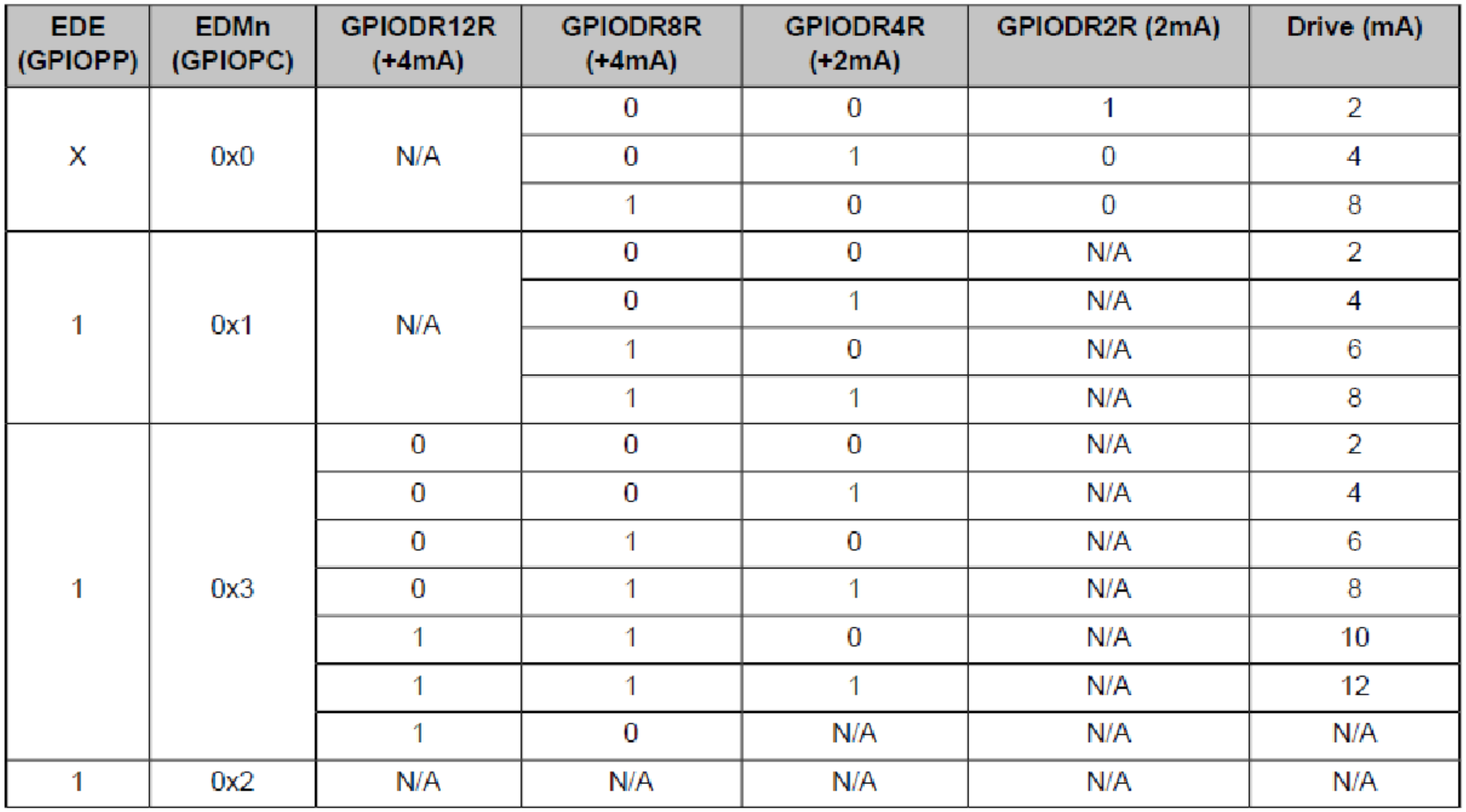
通用输入输出接口（GPIO）

- **输出驱动能力配置寄存器GPIODR2R，GPIODR4R，GPIODR8R，GPIODR12R，GPIOPP，GPIOPC**
  - 这几个寄存器用于控制引脚的最大电流输出能力，可以是2mA、4mA、8mA、10mA或者12mA；
  - GPIODR2R，GPIODR4R，GPIODR8R，GPIODR12R这几个寄存器都是只有**低8位**有实际作用，用于控制该端口的对应的八个引脚的输出能力，每一位控制一个引脚；
  - GPIOPP寄存器只有**1位**有作用，控制是否开启强化输出模式；
  - GPIOPC寄存器的**低16位**有意义，每两位控制一个引脚。

通用输入输出接口（GPIO）

➢ 最大电流输出能力

Table 10-13. GPIO Drive Strength Options



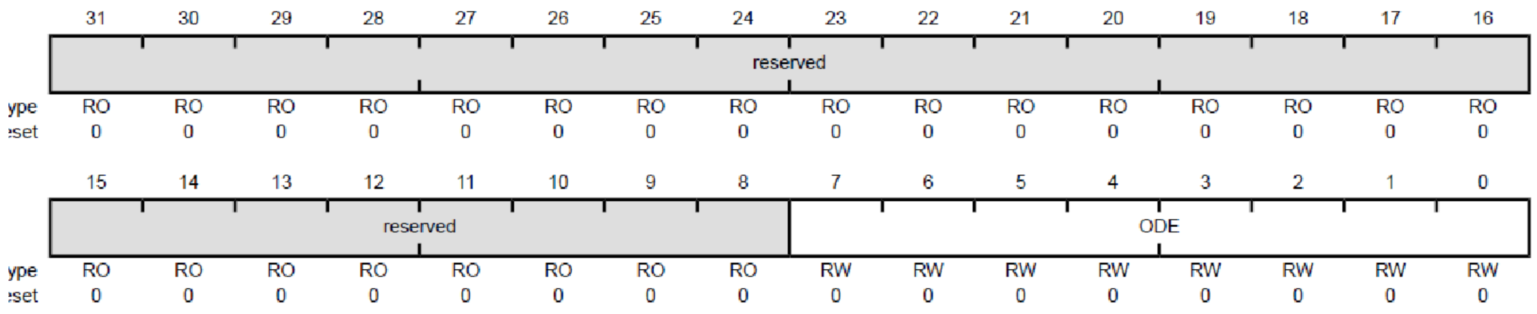
EDE (GPIOPP)	EDMn (GPIOPC)	GPIODR12R (+4mA)	GPIODR8R (+4mA)	GPIODR4R (+2mA)	GPIODR2R (2mA)	Drive (mA)
X	0x0	N/A	0	0	1	2
			0	1	0	4
			1	0	0	8
1	0x1	N/A	0	0	N/A	2
			0	1	N/A	4
			1	0	N/A	6
			1	1	N/A	8
1	0x3	0	0	0	N/A	2
		0	0	1	N/A	4
		0	1	0	N/A	6
		0	1	1	N/A	8
		1	1	0	N/A	10
		1	1	1	N/A	12
		1	0	N/A	N/A	N/A
1	0x2	N/A	N/A	N/A	N/A	N/A

通用输入输出接口（GPIO）

➢ 开漏输出配置寄存器GPIOODR

– 只有低八位有实际作用，用于控制该端口的八个引脚是否启用开漏输出，每一位控制一个引脚

– 0表示**推挽输出**，1表示**开漏输出**



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
ype	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ODE							
ype	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	RW	0x00	Output Pad Open Drain Enable

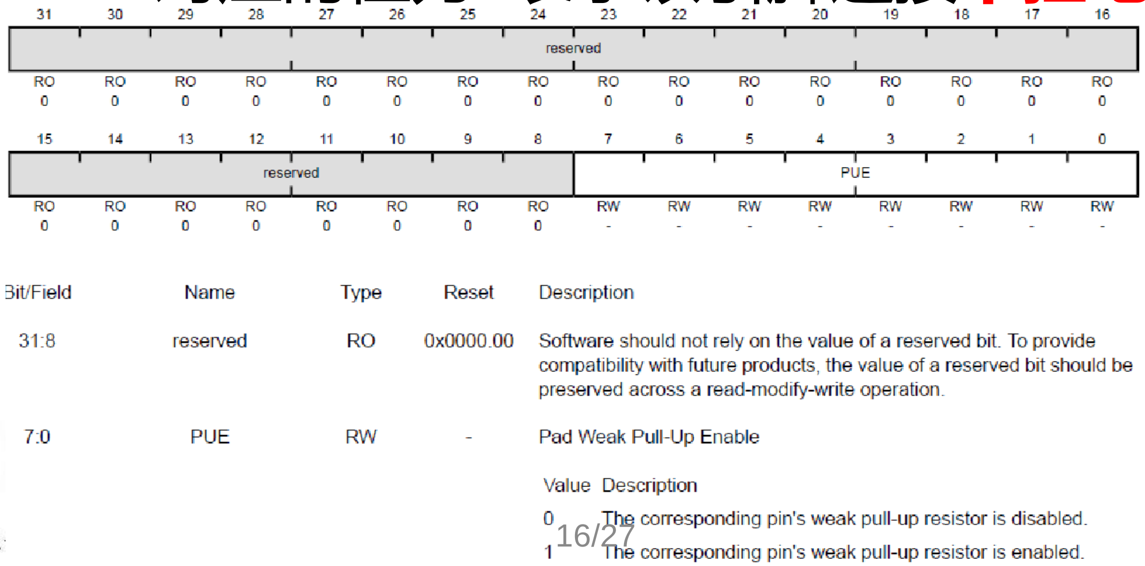
Value	Description
0	The corresponding pin is not configured as open drain.
1	The corresponding pin is configured as open drain.

➢ 上拉与下拉电阻配置寄存器GPIOPUR，GPIOPDR

– 只有**低八位**有实际作用，用于控制该端口的八个引脚是否连接上拉电阻、下拉电阻，每一位控制一个引脚；

– GPIOPUR 对应的位为**0**表示该引脚**不连接上拉电阻**，GPIOPUR 对应的位为**1**表示该引脚**连接上拉电阻**；

– GPIOPDR对应的位为**0**表示该引脚**不连接下拉电阻**，GPIOPDR 对应的位为**1**表示该引脚连接**下拉电阻**。



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								PUE							
RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

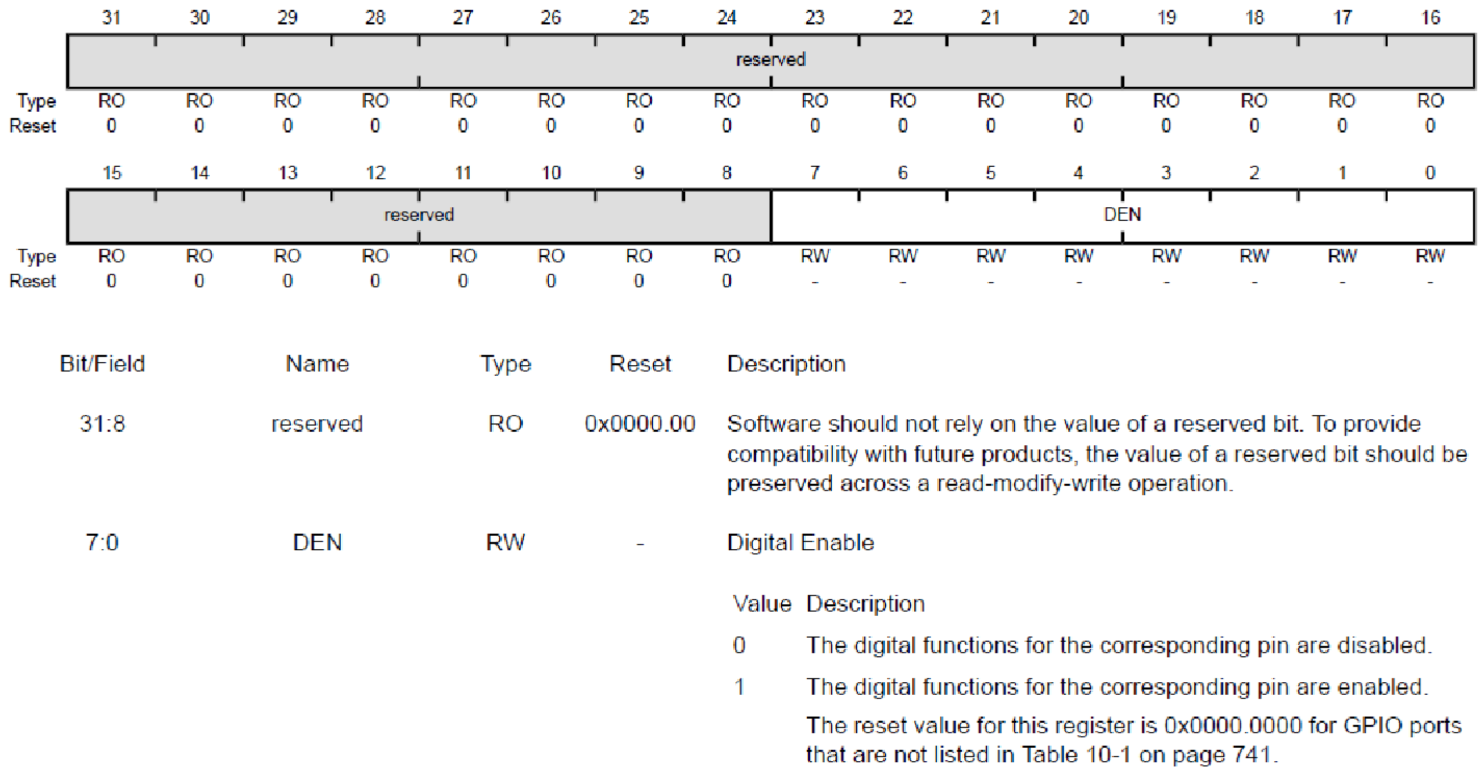
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	RW	-	Pad Weak Pull-Up Enable

Value	Description
0	The corresponding pin's weak pull-up resistor is disabled.
1	The corresponding pin's weak pull-up resistor is enabled.

➢ 数字逻辑功能使能寄存器GPIODEN

– 重置后，引脚自动配置为三态，断开引脚与GPIO模块的连接；

– 如果要使用引脚的**数字逻辑**功能，需要**使能GPIODEN**寄存器中相应的位。



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	RW	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.

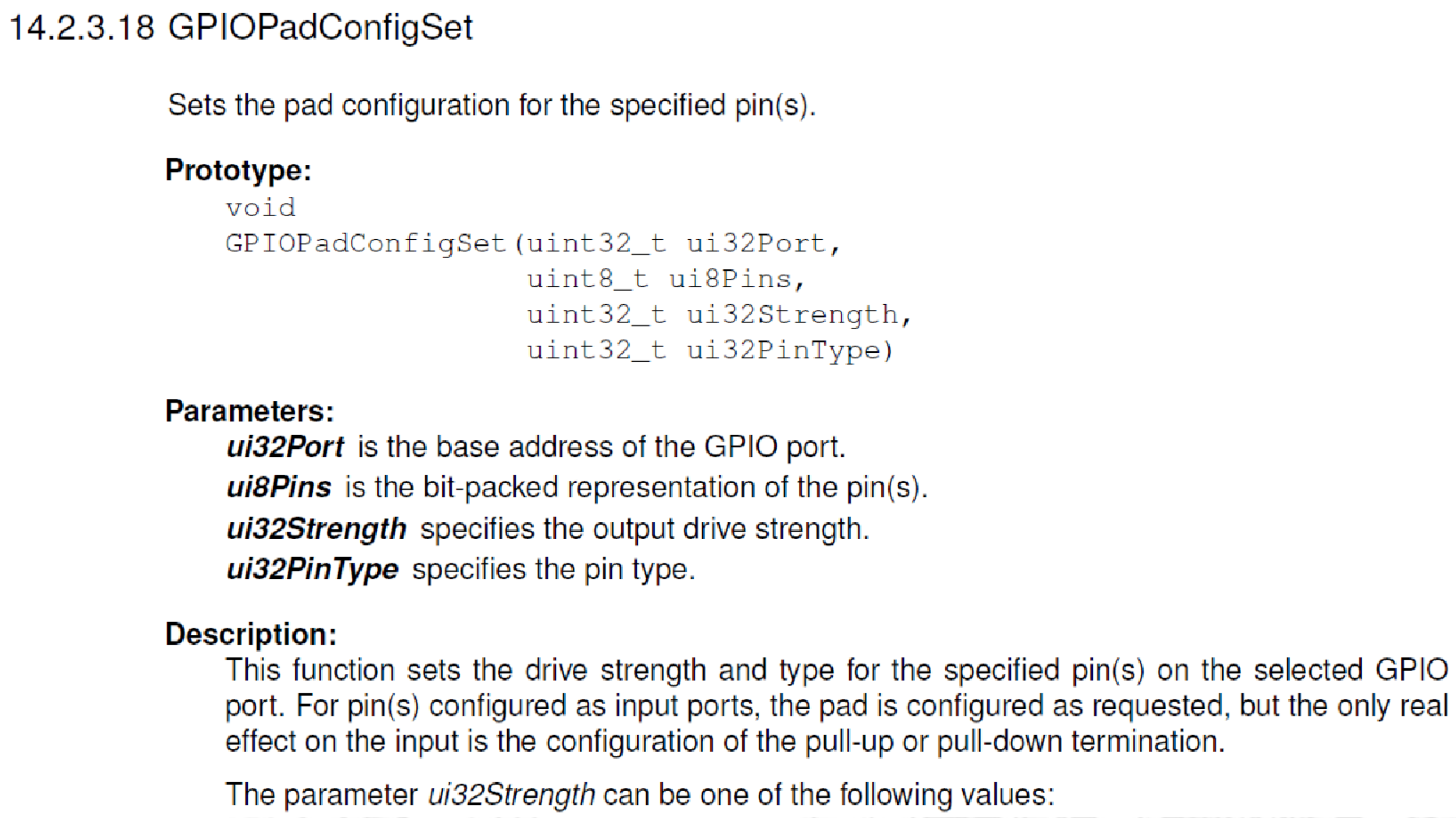
The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 741.

学习内容

- 控制寄存器的访问
- 控制寄存器的功能
- 数据寄存器的功能与操作
- GPIO模块的使能
- GPIO模块的应用1：矩阵键盘
- GPIO模块的应用2：数码管
- 矩阵键盘和数码管的应用

➢ GPIOPadConfigSet函数

– 用于设置引脚的电流输出能力和引脚类型，如上拉、下拉、开漏等



14.2.3.18 GPIOPadConfigSet

Sets the pad configuration for the specified pin(s).

Prototype:

```
void
GPIOPadConfigSet(uint32_t ui32Port,
                 uint8_t ui8Pins,
                 uint32_t ui32Strength,
                 uint32_t ui32PinType)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

ui32Strength specifies the output drive strength.

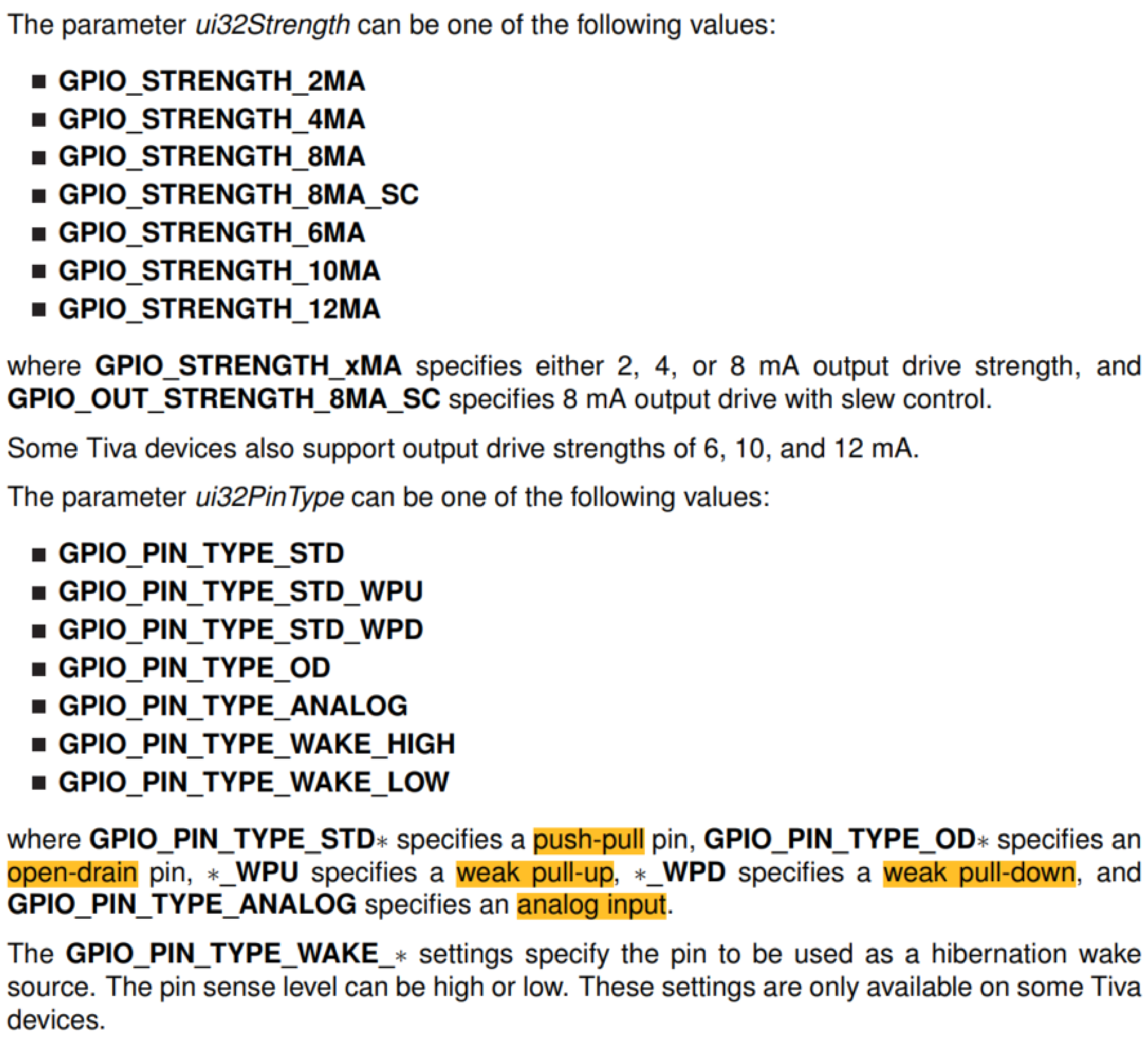
ui32PinType specifies the pin type.

Description:

This function sets the drive strength and type for the specified pin(s) on the selected GPIO port. For pin(s) configured as input ports, the pad is configured as requested, but the only real effect on the input is the configuration of the pull-up or pull-down termination.

The parameter *ui32Strength* can be one of the following values:

通用输入输出接口（GPIO）



The parameter *ui32Strength* can be one of the following values:

- **GPIO_STRENGTH_2MA**
- **GPIO_STRENGTH_4MA**
- **GPIO_STRENGTH_8MA**
- **GPIO_STRENGTH_8MA_SC**
- **GPIO_STRENGTH_6MA**
- **GPIO_STRENGTH_10MA**
- **GPIO_STRENGTH_12MA**

where **GPIO_STRENGTH_xMA** specifies either 2, 4, or 8 mA output drive strength, and **GPIO_OUT_STRENGTH_8MA_SC** specifies 8 mA output drive with slew control.

Some Tiva devices also support output drive strengths of 6, 10, and 12 mA.

The parameter *ui32PinType* can be one of the following values:

- **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_WAKE_HIGH**
- **GPIO_PIN_TYPE_WAKE_LOW**

where **GPIO_PIN_TYPE_STD**∗ specifies a push-pull pin, **GPIO_PIN_TYPE_OD**∗ specifies an open-drain pin, ∗**_WPU** specifies a weak pull-up, ∗**_WPD** specifies a weak pull-down, and **GPIO_PIN_TYPE_ANALOG** specifies an analog input.

The **GPIO_PIN_TYPE_WAKE_**∗ settings specify the pin to be used as a hibernation wake source. The pin sense level can be high or low. These settings are only available on some Tiva devices.

- 在GPIOPinTypeGPIOOutput函数中，调用
  - GPIOPadConfigSet(ui32Port, ui8Pins, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD)

作用是:

- 将指定的端口的引脚，设定为**推挽输出**，最大输出电流为**2mA**

- GPIOPadConfigSet的具体实现方式在gpio.c中

➢ GPIOPadConfigSet函数

– 设置GPIOPC寄存器

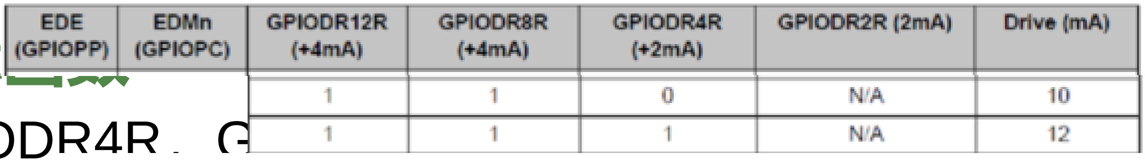
```
#define GPIO_STRENGTH_2MA       0x00000001  // 2mA drive strength
#define GPIO_STRENGTH_4MA       0x00000002  // 4mA drive strength
#define GPIO_STRENGTH_6MA       0x00000065  // 6mA drive strength
#define GPIO_STRENGTH_8MA       0x00000066  // 8mA drive strength
#define GPIO_STRENGTH_10MA      0x00000075  // 10mA drive strength
#define GPIO_STRENGTH_12MA      0x00000077  // 12mA drive strength
```

```
for(ui8Bit = 0; ui8Bit < 8; ui8Bit++)
   {
       if(ui8Pins & (1 << ui8Bit))
       {
           HWREG(ui32Port + GPIO_O_PC) = (HWREG(ui32Port + GPIO_O_PC) &
                                          ~(0x3 << (2 * ui8Bit)));
           HWREG(ui32Port + GPIO_O_PC) |= (((ui32Strength >> 5) & 0x3) <<
                                           (2 * ui8Bit));
       }
 }
```

首先把要操作的引脚，对应的GPIOPC寄存器里的位清零;
如果驱动能力大于等于6mA，则设置该引脚GPIOPC中对应的位为0x03;
如果驱动能力设置为2mA或者4mA，那么设置该引脚GPIOPC中对应的位为0x00。

➢ GPIOPadConfigSet

– 设置GPIODR2R，GPIODR4R，G
寄存器



EDE (GPIOPP)	EDMn (GPIOPC)	GPIODR12R (+4mA)	GPIODR8R (+4mA)	GPIODR4R (+2mA)	GPIODR2R (2mA)	Drive (mA)
		1	1	0	N/A	10
		1	1	1	N/A	12

```
#define GPIO_STRENGTH_10MA      0x00000075
#define GPIO_STRENGTH_12MA      0x00000077
```

```
HWREG(ui32Port + GPIO_O_DR2R) = ((ui32Strength & 1) ?
                                  (HWREG(ui32Port + GPIO_O_DR2R) |
                                   ui8Pins) :
                                  (HWREG(ui32Port + GPIO_O_DR2R) &
                                   ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_DR4R) = ((ui32Strength & 2) ?
                                  (HWREG(ui32Port + GPIO_O_DR4R) |
                                   ui8Pins) :
                                  (HWREG(ui32Port + GPIO_O_DR4R) &
                                   ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_DR8R) = ((ui32Strength & 4) ?
                                  (HWREG(ui32Port + GPIO_O_DR8R) |
                                   ui8Pins) :
                                  (HWREG(ui32Port + GPIO_O_DR8R) &
                                   ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_DR12R) = ((ui32Strength & 0x10) ?
                                   (HWREG(ui32Port + GPIO_O_DR12R) |
                                    ui8Pins) :
                                   (HWREG(ui32Port + GPIO_O_DR12R) &
                                    ~(ui8Pins)));
```

➢ GPIOPadConfigSet函数

– 设置GPIOOD，GPIOPUR，GPIOPDR，GPIODEN这几个寄存器

```
#define GPIO_PIN_TYPE_STD       0x00000008  // Push-pull
#define GPIO_PIN_TYPE_STD_WPU   0x0000000A  // Push-pull with weak pull-up
#define GPIO_PIN_TYPE_STD_WPD   0x0000000C  // Push-pull with weak pull-down
#define GPIO_PIN_TYPE_OD        0x00000009  // Open-drain
#define GPIO_PIN_TYPE_ANALOG    0x00000000  // Analog comparator
#define GPIO_PIN_TYPE_WAKE_HIGH 0x00000208  // Hibernate wake, high
#define GPIO_PIN_TYPE_WAKE_LOW  0x00000108  // Hibernate wake, low

HWREG(ui32Port + GPIO_O_ODR) = ((ui32PinType & 1) ?
                                (HWREG(ui32Port + GPIO_O_ODR) | ui8Pins) :
                                (HWREG(ui32Port + GPIO_O_ODR) & ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_PUR) = ((ui32PinType & 2) ?
                                (HWREG(ui32Port + GPIO_O_PUR) | ui8Pins) :
                                (HWREG(ui32Port + GPIO_O_PUR) & ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_PDR) = ((ui32PinType & 4) ?
                                (HWREG(ui32Port + GPIO_O_PDR) | ui8Pins) :
                                (HWREG(ui32Port + GPIO_O_PDR) & ~(ui8Pins)));
HWREG(ui32Port + GPIO_O_DEN) = ((ui32PinType & 8) ?
                                (HWREG(ui32Port + GPIO_O_DEN) | ui8Pins) :
                                (HWREG(ui32Port + GPIO_O_DEN) & ~(ui8Pins)));
```

➢ 数据寄存器GPIODATA

- 低八位有意义，代表了每个引脚的电平
- 如果GPIODIR寄存器配置为输出，那么这个寄存器相应的位的电平，会直接传递到对应的引脚上。
- 如果GPIODIR寄存器配置为输入，那么这个寄存器相应的位的电平，代表对应的引脚上的电平。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 749 for examples of reads and writes.

数据寄存器的功能与操作

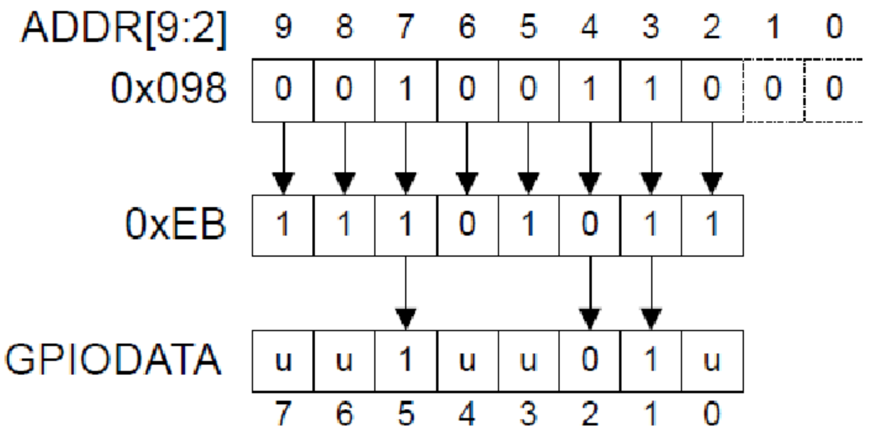
➢ GPIODATA的位带操作

– 使用总线的[9:2]位作为屏蔽位

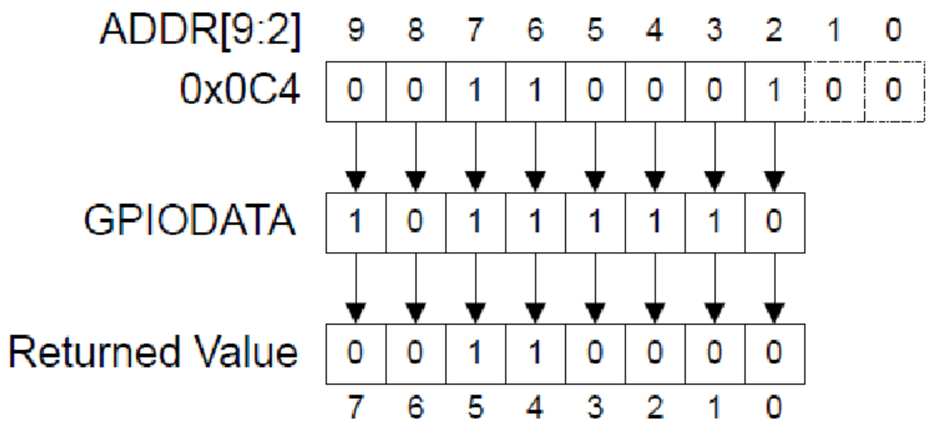
Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	RW	0x0000.0000	GPIO Data	759
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction	760
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense	761
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges	762
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event	763

中间空出了一定的地址空间

写



读



ADDR[9:2] 9 8 7 6 5 4 3 2 1 0
0x0C4 0 0 1 1 0 0 0 1 0 0
GPIODATA 1 0 1 1 1 1 1 0
Returned Value 0 0 1 1 0 0 0 0
7 6 5 4 3 2 1 0

谢谢！