Utilizando o Tiva: General Purpose IOs

EMB5629

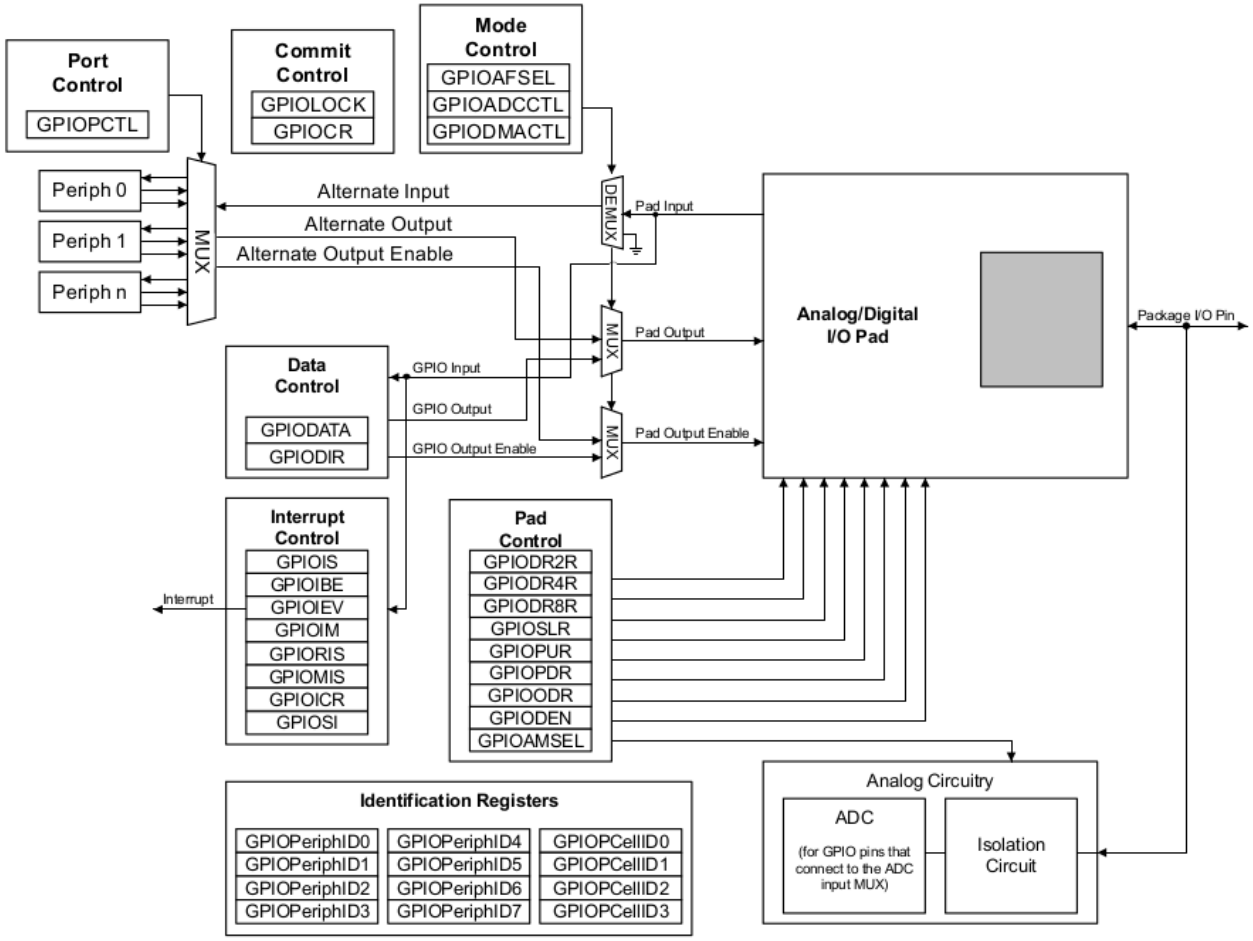
UNIVERSIDADE FEDERAL
DE SANTA CATARINA

General Purpose IO

- Any GPIO can be an interrupt:
  - Edge-triggered on rising, falling or both
  - Level-sensitive on high or low values
- Can directly initiate an ADC sample sequence or µDMA transfer
- Toggle rate up to the CPU clock speed on the Advanced
- High-Performance Bus. ½ CPU clock speed on the Standard.
- 5V tolerant in input configuration
- Programmable Drive Strength (2, 4, 8mA or 8mA with slew rate control)
- Programmable weak pull-up, pull-down, and open drain
- Pin state can be retained during Hibernation mode

General Purpose IO

Figure 10-2. Analog/Digital I/O Pads



Mapa de Memória

Table 2-4. Memory Map

Start	End	Description	For details, see page ...
Memory			
0x0000.0000	0x0003.FFFF	On-chip Flash	540
0x0004.0000	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.7FFF	Bit-banded on-chip SRAM	525
0x2000.8000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x220F.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	525
0x2210.0000	0x3FFF.FFFF	Reserved	-
Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	776
0x4000.1000	0x4000.1FFF	Watchdog timer 1	776
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	658
0x4000.5000	0x4000.5FFF	GPIO Port B	658

Escrevendo em um registrador

```
#define ESC_REG(x)
(*((volatile uint32_t *)(x)))
```

GPIO Address Masking

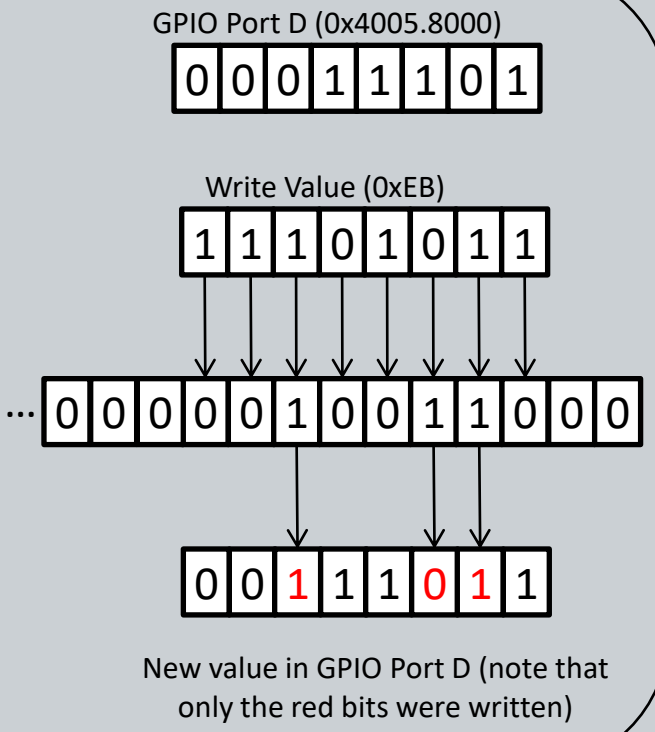
Each GPIO port has a base address. You can write an 8-bit value directly to this base address and all eight pins are modified. If you want to modify specific bits, you can use a bit-mask to indicate which bits are to be modified. This is done in hardware by mapping each GPIO port to 256 addresses. Bits 9:2 of the address bus are used as the bit mask.

The register we want to change is GPIO Port D (0x4005.8000)
Current contents of the register is:

The value we will write is 0xEB:

Instead of writing to GPIO Port D directly, write to 0x4005.8098. Bits 9:2 (shown here) become a bit-mask for the value you write.

Only the bits marked as “1” in the bit-mask are changed.




```
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_5|GPIO_PIN_2|GPIO_PIN_1, 0xEB);
```

Note: you specify base address, bit mask, and value to write.
The GIPOPinWrite() function determines the correct address for the mask.

Critical Function GPIO Protection

- Six pins on the device are protected against accidental programming:
  - PC3,2,1 & 0: JTAG/SWD
  - PD7 & PF0: NMI

◆Any write to the following registers for these pins will not be stored unless the GPIOLOCK register has been unlocked:

- GPIO Alternate Function Select register
- GPIO Pull Up or Pull Down select registers
- GPIO Digital Enable register

◆The following sequence will unlock the GPIOLOCK register for PF0 using direct register programming:

◆

```
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;
```

◆Reading the GPIOLOCK register returns it to lock status

GPIOs Pins and Alternate Functions

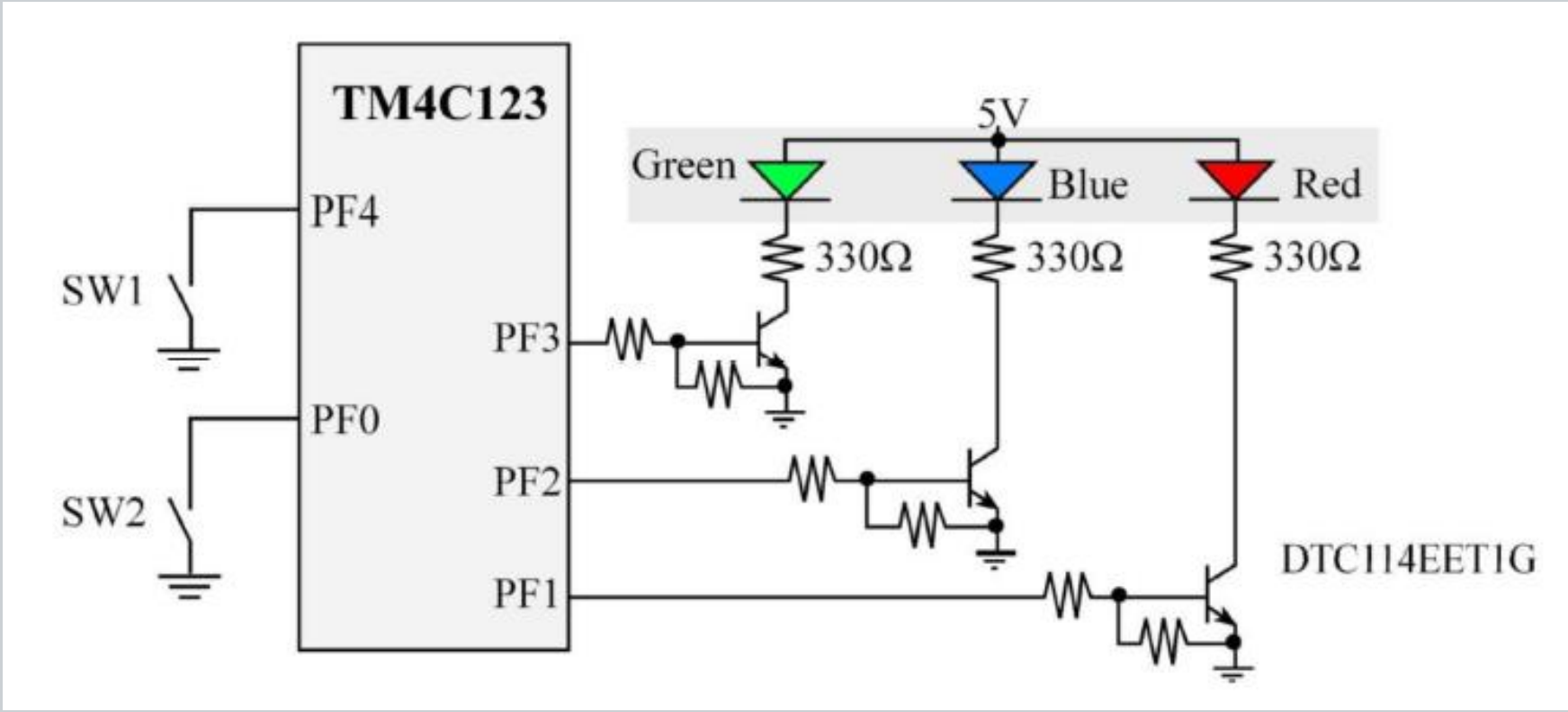
- Table 10-2 Page 650

Table 10-2. GPIO Pins and Alternate Functions (64LQFP)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	CAN1Rx	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	CAN1Tx	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-
PB0	45	USB0ID	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-

Primeira aplicação

- Acender um led quando o botão estiver pressionado.
- LED RGB – PF1, PF2 e PF3.
- Led Verde!


TM4C123
PF4
PF0
PF3
PF2
PF1
SW1
SW2
5V
Green
Blue
Red
330Ω
330Ω
330Ω
DTC114EET1G

Initialization and Configuration

- Page 656 – 10.3 Initialization and Config.

1. Enable the clock to the port by setting the appropriate bits in the **RCGCGPIO** register (see page 340). In addition, the **SCGCGPIO** and **DCGCGPIO** registers can be programmed in the same manner to enable clocking in Sleep and Deep-Sleep modes.
2. Set the direction of the GPIO port pins by programming the **GPIODIR** register. A write of a 1 indicates output and a write of a 0 indicates input.
3. Configure the **GPIOAFSEL** register to program each bit as a GPIO or alternate pin. If an alternate pin is chosen for a bit, then the `PMCx` field must be programmed in the **GPIOPCTL** register for the specific peripheral required. There are also two registers, **GPIOADCCTL** and **GPIODMACTL**, which can be used to program a GPIO pin as a ADC or µDMA trigger, respectively.
4. Set the drive strength for each of the pins through the **GPIODR2R**, **GPIODR4R**, and **GPIODR8R** registers.
5. Program each pad in the port to have either pull-up, pull-down, or open drain functionality through the **GPIOPUR**, **GPIOPDR**, **GPIOODR** register. Slew rate may also be programmed, if needed, through the **GPIOSLR** register.

Anderson Wedderhoff Spengler

E-mail: anderson.spengler@ufsc.br

Telefone: +55 (48) 3721 7489

UNIVERSIDADE FEDERAL
DE SANTA CATARINA