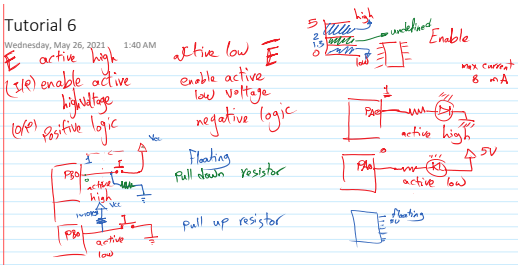
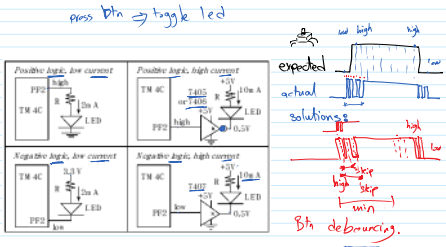
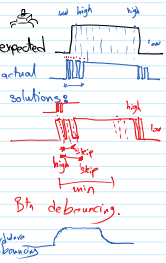
Tutorial 6

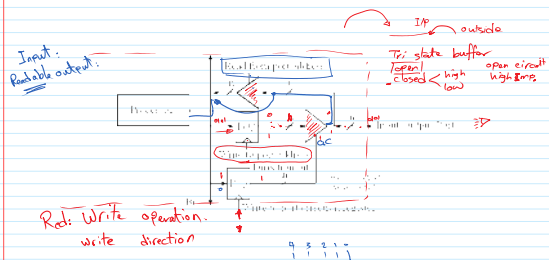
Wednesday, May 26, 2021 1:40 AM

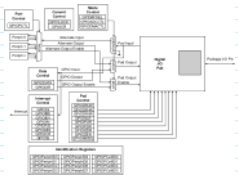


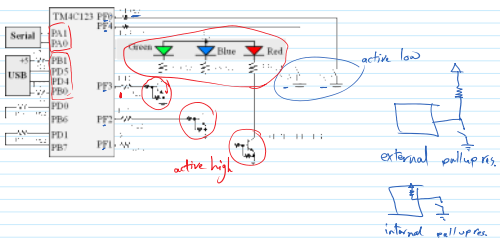
press Btn ⇒ toggle led

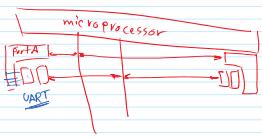










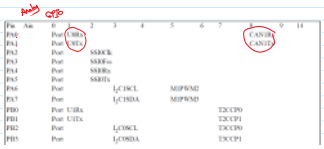


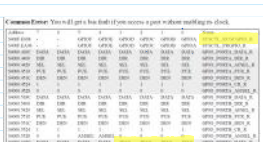
look like Memory access
but they are not memory

Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous devices. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

base addr

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000





- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	RW	0x0000.0000	GPIO Data
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	RW	0x0000.0000	GPIO Interrupt Mask
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	RW	-	GPIO Alternate Function Select
0x500	GPIODR2R	RW	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	RW	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	RW	0x0000.0000	GPIO 8-mA Drive Select
0x50C	GPIOODR	RW	0x0000.0000	GPIO Open Drain Select
0x510	GPIOPUR	RW	-	GPIO Pull-Up Select
0x514	GPIOPDR	RW	0x0000.0000	GPIO Pull-Down Select
0x518	GPIOSLR	RW	0x0000.0000	GPIO Slew Rate Control Select
0x51C	GPIODEN	RW	-	GPIO Digital Enable
0x520	GPIOLOCK	RW	0x0000.0001	GPIO Lock
0x524	GPIOCR	-	-	GPIO Commit
0x528	GPIOAMSEL	RW	0x0000.0000	GPIO Analog Mode Select
0x52C	GPIOPCTL	RW	-	GPIO Port Control
0x530	GPIOADCCTL	RW	0x0000.0000	GPIO ADC Control
0x534	GPIODMACTL	RW	0x0000.0000	GPIO DMA Control
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3

GPIO Alt Fun located at PCTL

Debugging

GPIO_PORTF_GPIODEN_R

0x4005D000 + 0x51C

4 fans JTag PC0—PC3

2 Pin NMI PD7 PF0

base address of PortF + offset DEN Registe

int platform_dependent

0x4005D000 + 0x98 = 0x4005D098

GPIO_PORTF_DATA_R EQU 0x4005D3FC

LDR R0, = GPIO_PORTF_DATA_R

LDR R1, [R0]

PF4 PF3 PF2 PF1 PF0

#define GPIO_PORTF_DATA_R (*((volatile uint32_t *) 0x4005D3FC))

int main() {
GPIO_PORTF_DATA_R = 0xAB;
}

int a = 5;
int* pointer = &a

pointer = 0x4005D3FC
*pointer = 15

Initialization of an I/O port

To initialize an I/O port for general use:

1. Activate the clock for the port in the Run Mode Clock Gating Control Register 2 (**RCGC2**).
2. Unlock the port (**LOCK** = **0x4C4F434B**). This step ~~is only needed for~~ pins **PC0-3**, **PD7** and **PF0** on TM4C123GXL LaunchPad.
3. Disable the analog function of the pin in the Analog Mode Select register (**AMSEL**), because we want to use the pin for digital I/O. If this pin is connected to the ADC or analog comparator, ~~its co~~rresponding bit in **AMSEL** must be set as **1**. In our case, this pin is used as digital I/O, so its corresponding bit must be set as **0**.
4. Clear bits in the port control register (**PCTL**) to select regular digital function. Each GPIO pin needs four bits in its corresponding PCTL register. Not every pin can be configured to every alternative function.
5. Set its direction register (**DIR**). A DIR bit of **0** means input, and **1** means output.
6. Clear bits in the alternate Function Select register (**AFSEL**).
7. Enable digital port in the Digital Enable register (**DEN**).

*Turning on the clock must be first but the other steps can occur in any order.

SYSCTL_RCGCGPIO_R
SYSCTL_PRGPIO_R

Q1. Write an assembly function that initializes port F pins 1, 2, and 3 as Digital Output with initial zero values.

unlocking PC0—PC3 PD7—PF0

```
1  LDR R0,=SYSCTL_RCGCGPIO_R
   LDR R1, [R0]
   ORR R1, #0x20
   STR R1, [R0]
   LDR R0,=SYSCTL_PRGPIO_R      ||  nop
NR LDR R1,[R0]                  ||  nop
   TST R1, #0x20
   beq NR
2  LDR R0,=GPIO_PORTF_lock_R
   LDR R1,=0x1ACF4345B
   STR R1, [R0]
   LDR R0,=GPIO_PORTF_CR_R
   MOV R1, #1
   STR R1, [R0]
3  LDR R0,=GPIO_PORTF_AMSEL_R
   MOV R2,#0x0D
   LDR R1, [R0]
   BIC R1, R2
   STR R1,[R0]
4  LDR R0,=GPIO_PORTF_PCTL_R
   LDR R1,[R0]
   MOV R2,#0xFFF0F
   BIC R1, R2
   STR R1,[R0]
```

SYSCTL_RCGCGPIO_R |= 0x20;
while((SYSCTL_PRGPIO_R&0x20));
// unlocking
GPIO_PORTF_LOCK_R = 0x4C4F434B;
GPIO_PORTF_CR_R = 0x01;
GPIO_PORTF_AMSEL_R &= ~0xD;
GPIO_PORTF_PCTL_R &= ~0xFF0F;
GPIO_PORTF_AFSEL_R &= ~0xD;
GPIO_PORTF_DIR_R |= 0xD
GPIO_PORTF_DEN_R |= 0xD
GPIO_PORTF_DATA_R &= ~0xD

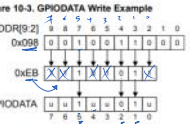
10.2.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 662) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

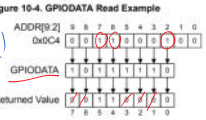
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where ↓ indicates that data is unchanged by the write. This example demonstrates how **GPIODATA** bits 5, 2, and 1 are written.

Figure 10-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 10-4. This example shows how to read **GPIODATA** bits 5, 4, and 0.

Figure 10-4. GPIODATA Read Example



GPIOF → DATA = 0xAB
GPIOF → DEN != 0x02

PF1 DEN =1

11,11 11,11 00
0x3FC

int x = 5;
int y = x + 10;
while ( ) {
y *= 3;
}