ARM-Based Microcontrollers

TM4C123GH6PM GPIO PERIPHERAL

By : Ahmed Magdy

AGENDA

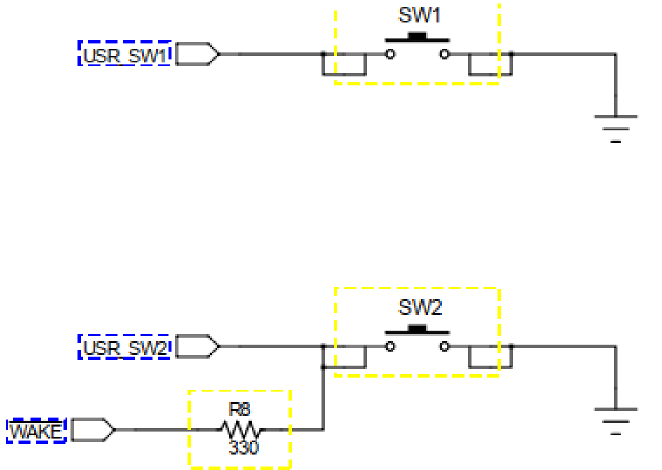
- Launchpad and Microcontroller architecture overview
- How to read the datasheet
- Memory mapped architecture and address space
- GPIO PERIPHERAL HW DESIGN
- GPIO OUTPUT CONFIGURATION
- GPIO INPUT CONFIGURATION
- BIT BANDING & Register Masking
- GPIO PERIPHERAL DRIVER SW DESIGN

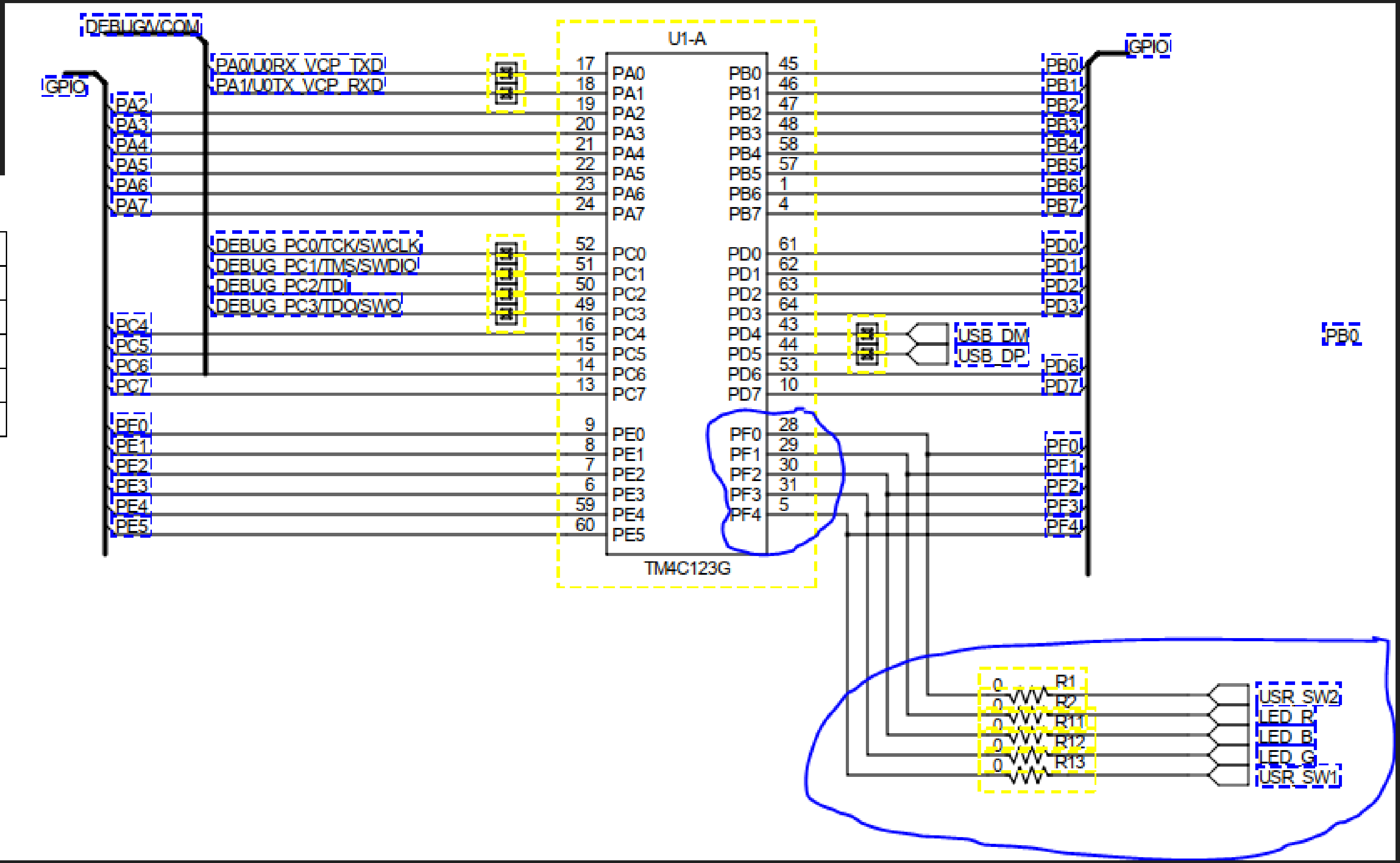
Launchpad and Microcontroller architecture overview

- Launchpad Schematic:

Table 2-2. User Switches and RGB LED Signals

GPIO Pin	Pin Function	USB Device
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGD LED (Green)

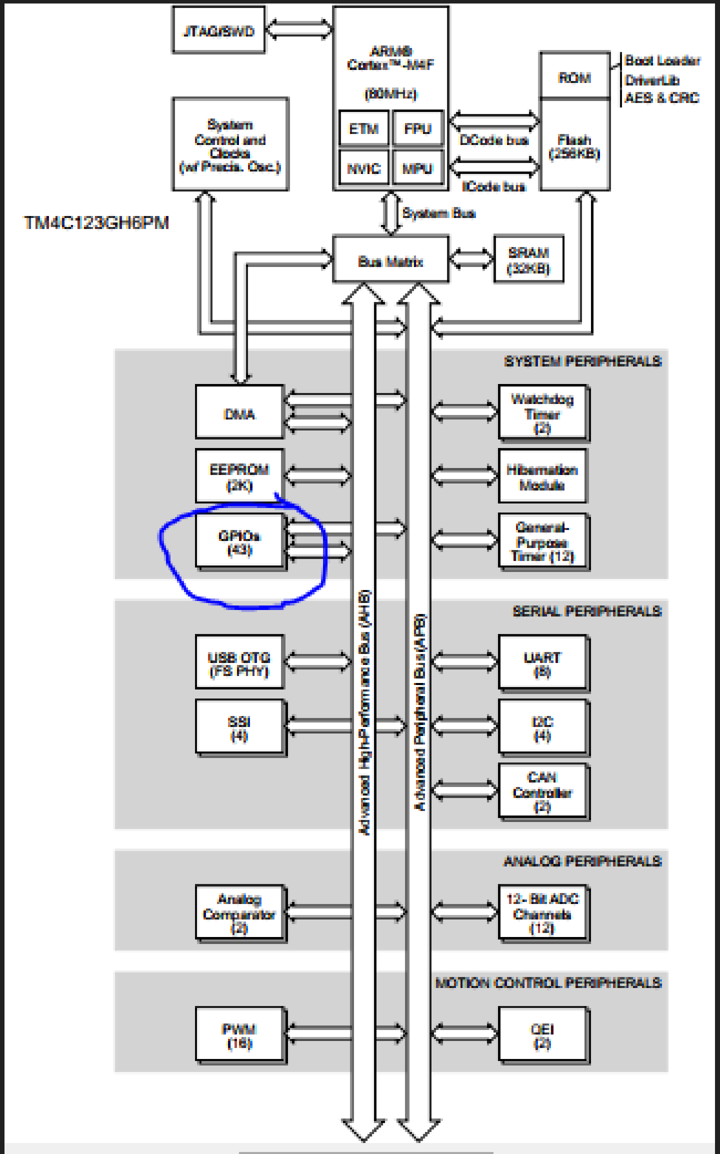




Launchpad and Microcontroller architecture overview

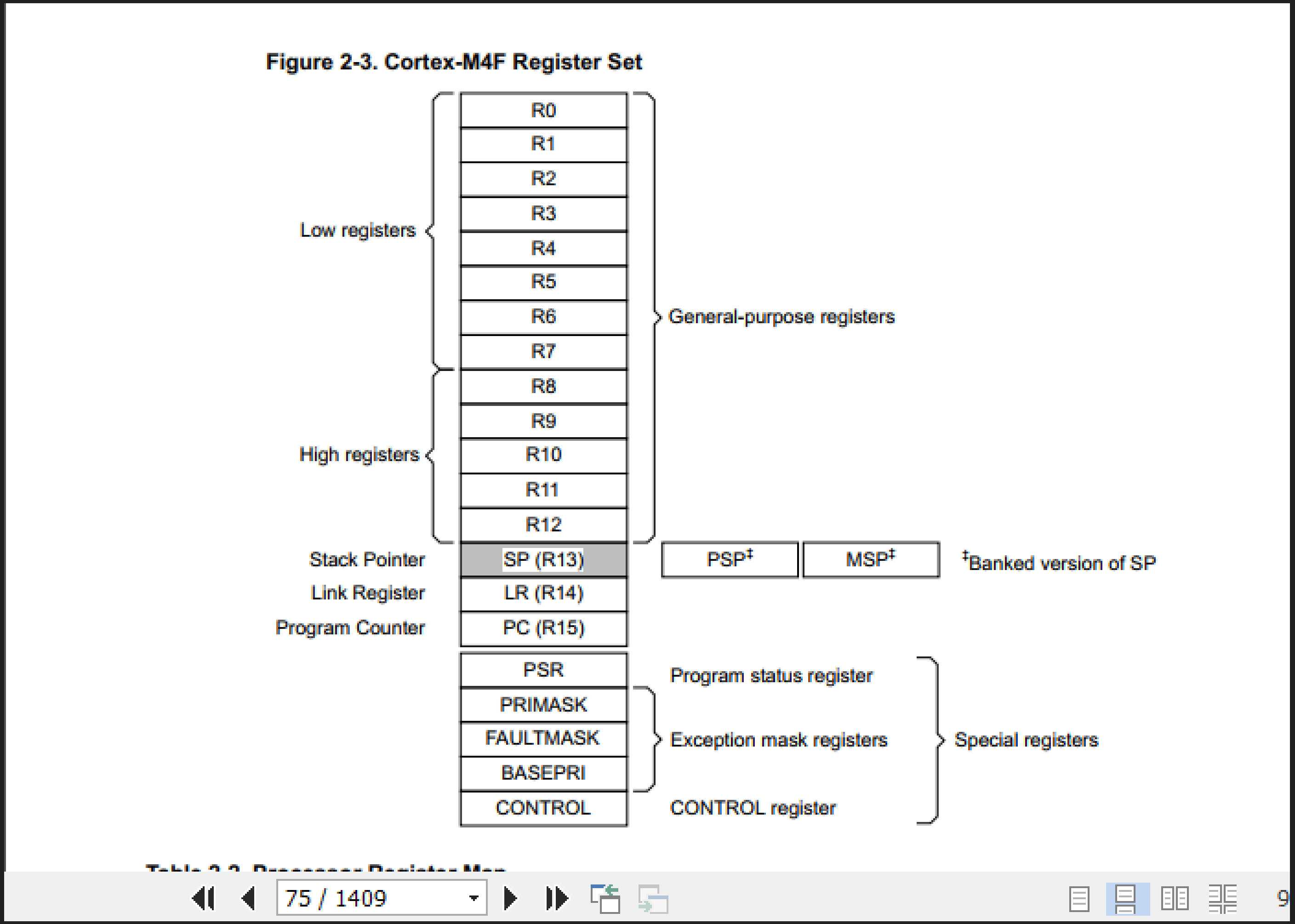
- Microcontroller Overview:

Feature	Description
Core	ARM Cortex-M4F processor core
Performance	80-MHz operation; 100 DMIPS performance
Flash	256 KB single-cycle Flash memory
System SRAM	32 KB single-cycle SRAM
EEPROM	2KB of EEPROM
Internal ROM	Internal ROM loaded with TivaWare™ for C Series software
General-Purpose Input/Output (GPIO)	Six physical GPIO blocks



Launchpad and Microcontroller architecture overview

- Microcontroller Overview:

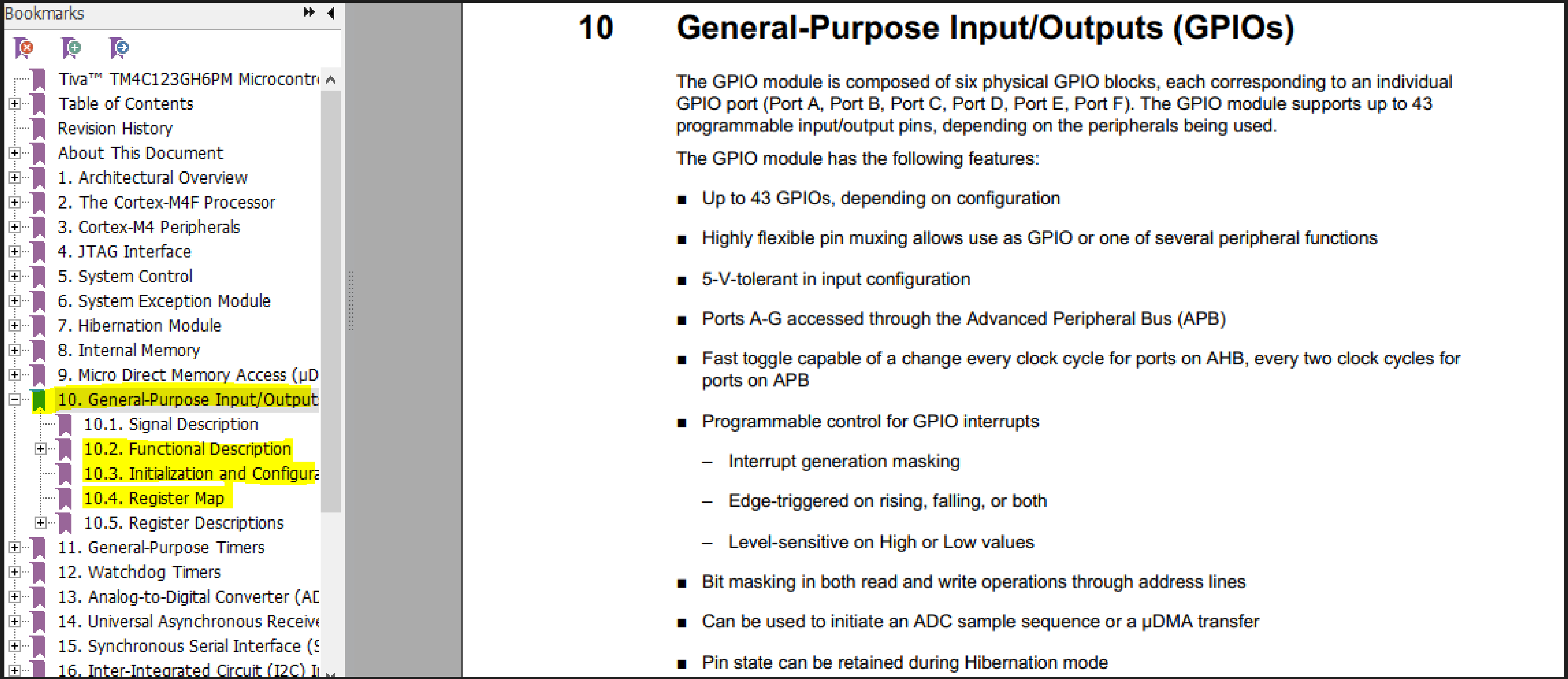


Launchpad and Microcontroller architecture overview

- Microcontroller Overview:

The ARM chips have two buses: APB (Advanced Peripheral Bus) and AHB (Advanced HighPerformance Bus). The AHB bus is much faster than APB. The AHB allows one clock cycle access to the peripherals. The APB is slower and its access time is minimum of 2 clock cycles.

How to read the datasheet


Bookmarks

Tiva™ TM4C123GH6PM Microcontr
Table of Contents
Revision History
About This Document
1. Architectural Overview
2. The Cortex-M4F Processor
3. Cortex-M4 Peripherals
4. JTAG Interface
5. System Control
6. System Exception Module
7. Hibernation Module
8. Internal Memory
9. Micro Direct Memory Access (µD
10. General-Purpose Input/Output
10.1. Signal Description
10.2. Functional Description
10.3. Initialization and Configura
10.4. Register Map
10.5. Register Descriptions
11. General-Purpose Timers
12. Watchdog Timers
13. Analog-to-Digital Converter (AD
14. Universal Asynchronous Receive
15. Synchronous Serial Interface (S
16. Inter-Integrated Circuit (I2C) I

10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of six physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F). The GPIO module supports up to 43 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 43 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Ports A-G accessed through the Advanced Peripheral Bus (APB)
- Fast toggle capable of a change every clock cycle for ports on AHB, every two clock cycles for ports on APB
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence or a µDMA transfer
- Pin state can be retained during Hibernation mode

How to read the datasheet

- Tiva™ TM4C123GH6PM Microcontr
- Table of Contents
- Revision History
- About This Document
- 1. Architectural Overview
- 2. The Cortex-M4F Processor
- 3. Cortex-M4 Peripherals
- 4. JTAG Interface
- 5. System Control
- 6. System Exception Module
- 7. Hibernation Module
- 8. Internal Memory
- 9. Micro Direct Memory Access (µD
- 10. General-Purpose Input/Output
  - 10.1. Signal Description
  - 10.2. Functional Description
  - 10.3. Initialization and Configura
  - 10.4. Register Map
  - 10.5. Register Descriptions
- 11. General-Purpose Timers
- 12. Watchdog Timers

well as the **GPIOPCellID0-GPIOPCellID3** registers.

10.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory 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. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 258). Note that GPIO can only be accessed through the AHB aperture.

To configure the GPIO pins of a particular port, follow these steps:

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.

Memory mapped architecture and address space

Tiva™ TM4C123GH6PM Microcontr
Table of Contents
Revision History
About This Document
1. Architectural Overview
2. The Cortex-M4F Processor
2.1. Block Diagram
2.2. Overview
2.3. Programming Model
2.4. Memory Model
2.5. Exception Model
2.6. Fault Handling
2.7. Power Management
2.8. Instruction Set Summary
3. Cortex-M4 Peripherals
4. JTAG Interface
5. System Control
6. System Exception Module
7. Hibernation Module
8. Internal Memory
9. Micro Direct Memory Access (μD
10. General-Purpose Input/Output
10.1. Signal Description
10.2. Functional Description
10.3. Initialization and Configura

2.4 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4 GB of addressable memory.

The memory map for the TM4C123GH6PM controller is provided in Table 2-4 on page 92. In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see "Bit-Banding" on page 97).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see "Cortex-M4 Peripherals" on page 122).

Note: Within the memory map, attempts to read or write addresses in reserved spaces result in a bus fault. In addition, attempts to write addresses in the flash range also result in a bus fault.

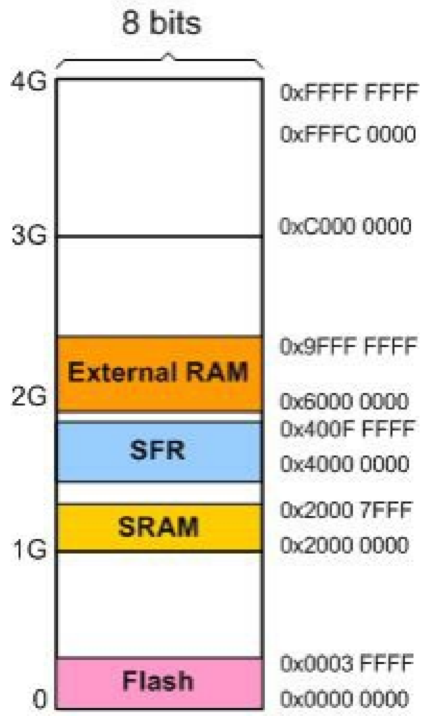
Table 2-4. Memory Map

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

Memory mapped architecture and address space

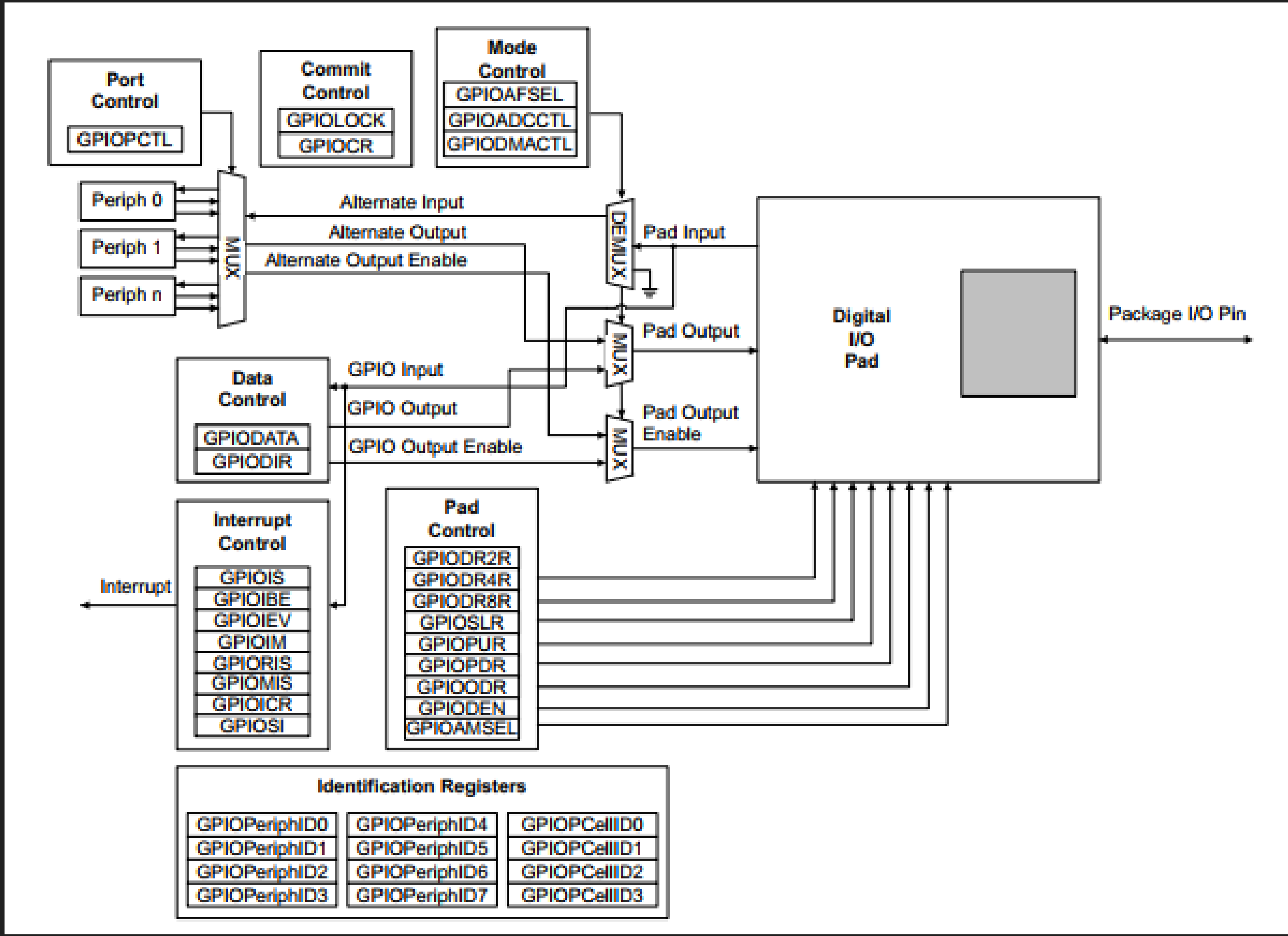
	Allocated size	Allocated address
Flash	256KB	0x00000000 to 0x0003FFFF
SRAM	32KB	0x20000000 to 0x20007FFF
I/O	All the peripherals	0x40000000 to 0x400FFFFF

Table 2-1: Memory Map in TM4C123GH6PM

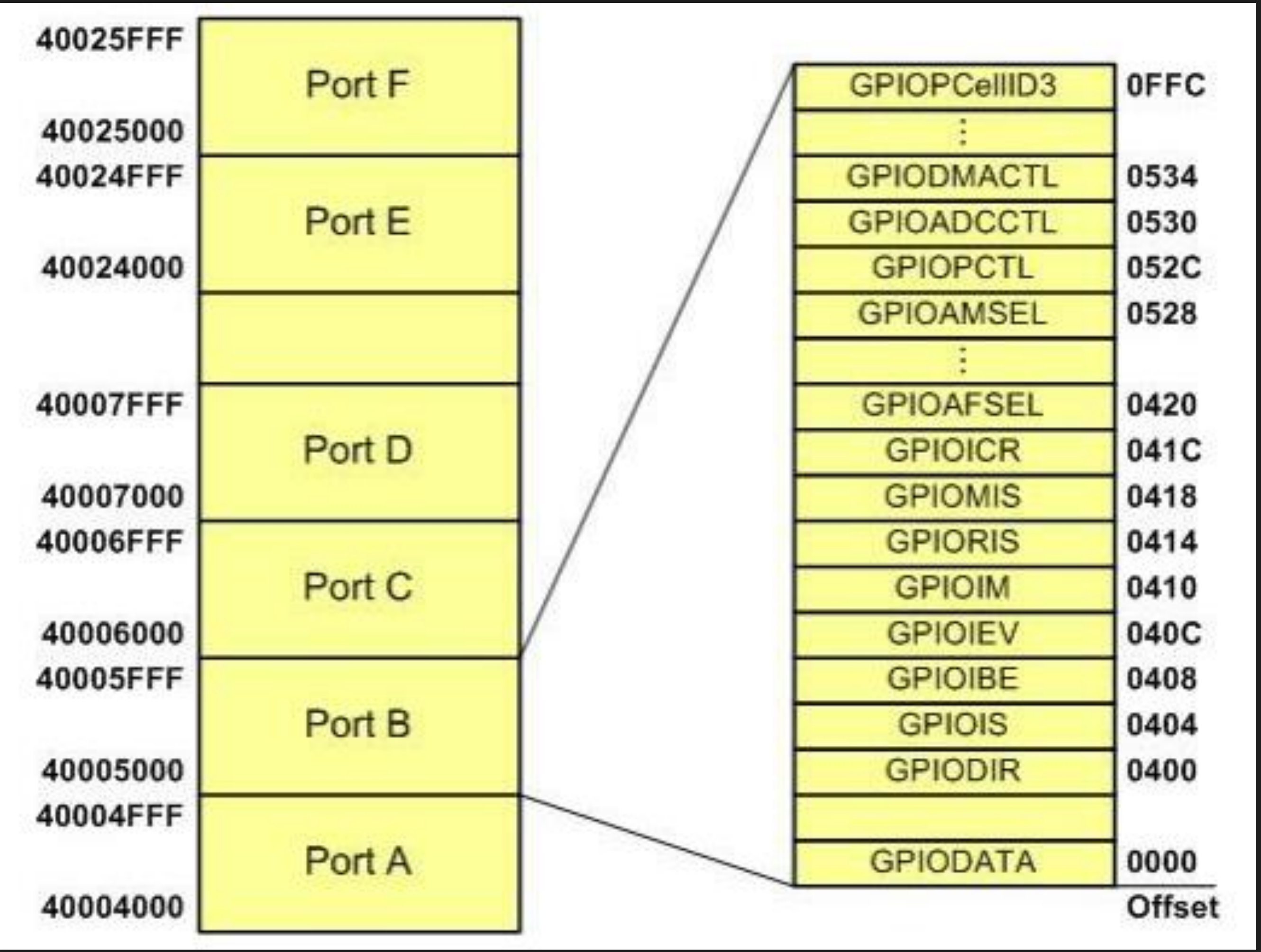


We have 8 bit storage in each location in the 4GB memory

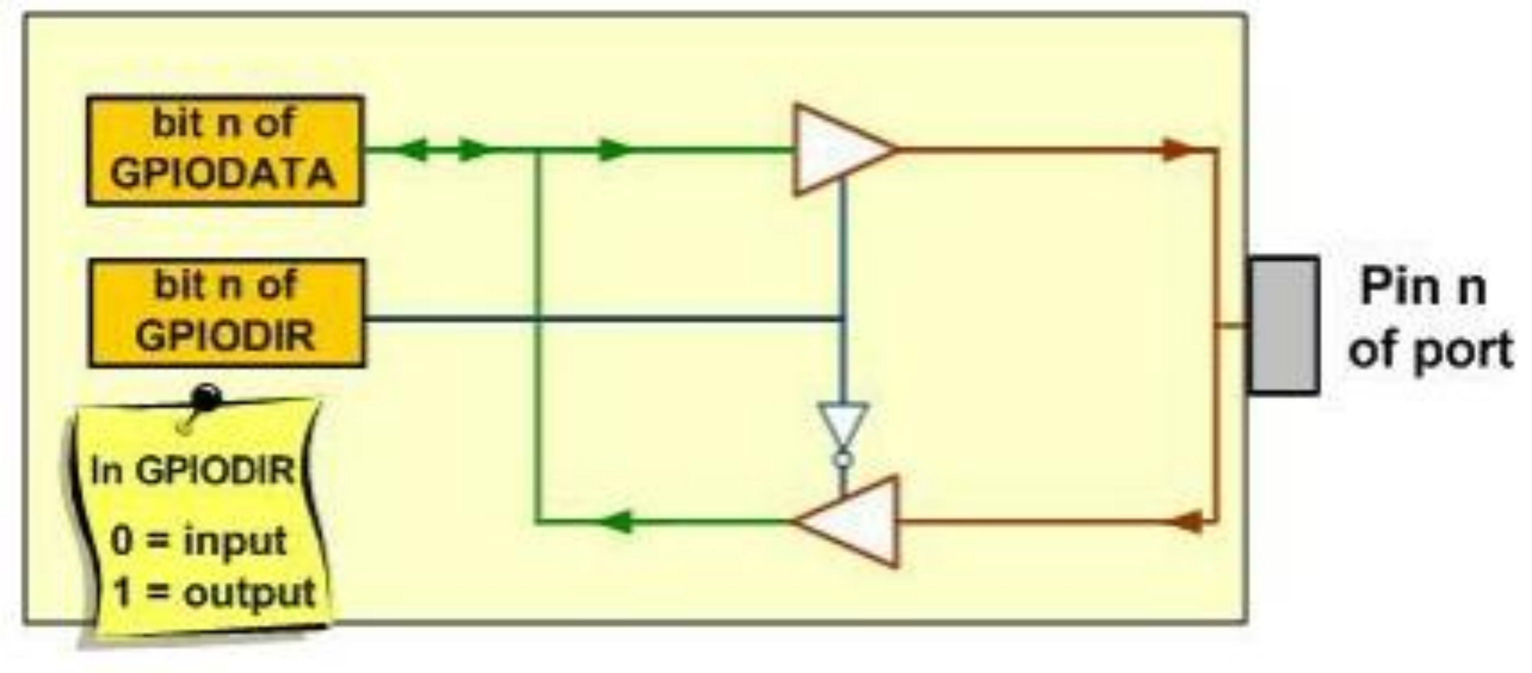
GPIO PERIPHERAL HW DESIGN



GPIO PERIPHERAL HW DESIGN



GPIO PERIPHERAL HW DESIGN



GPIO OUTPUT CONFIGURATION

- **Initialization and Configuration**
  To configure the GPIO pins of a particular port, follow these steps:
  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.
  6. To enable GPIO pins as digital I/Os, set the appropriate DEN bit in the **GPIODEN** register. To enable GPIO pins to their analog function (if available), set the GPIOAMSEL bit in the **GPIOAMSEL** register.

GPIO OUTPUT CONFIGURATION

- /* Toggling LEDs in C using registers by addresses */
- /* PORTF data register */
- #define PORTFDAT (*((volatile unsigned int*)0x400253FC))
- /* PORTF data direction register */
- #define PORTFDIR (*((volatile unsigned int*)0x40025400))
- /* PORTF digital enable register */
- #define PORTFDEN (*((volatile unsigned int*)0x4002551C))
- /* run mode clock gating register */
- #define RCGCGPIO (*((volatile unsigned int*)0x400FE608))
- /* coprocessor access control register */
- #define SCB_CPAC (*((volatile unsigned int*)0xE000ED88))
- void delayMs(int n); /* function prototype for delay */

GPIO OUTPUT CONFIGURATION

- int main(void)
- {
- /* enable clock to GPIOF at clock gating register */
- RCGCGPIO |= 0x20;
- /* set PORTF pin3-1 as output pins */
- PORTFDIR = 0x0E;
- /* set PORTF pin3-1 as digital pins */
- PORTFDEN = 0x0E;
- while(1)
- {
- /* write PORTF to turn on all LEDs */
- PORTFDAT = 0x0E;

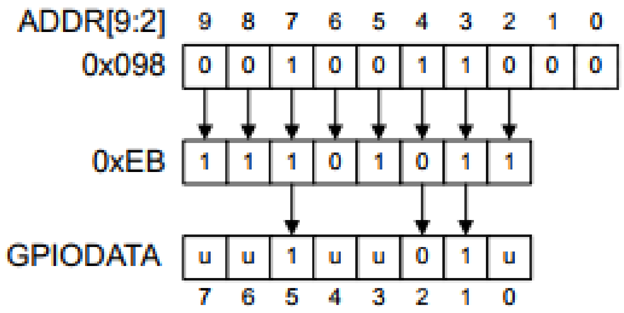
GPIO OUTPUT CONFIGURATION

- delayMs(500);
- /* write PORTF to turn off all LEDs */
- PORTFDAT = 0;
- delayMs(500);
- }
- }
- /* delay n milliseconds (16 MHz CPU clock) */
- void delayMs(int n)
- {
- int i, j;
- for(i = 0 ; i < n; i++)
- for(j = 0; j < 3180; j++)
- {} /* do nothing for 1 ms */
- }

BIT BANDING & Register Masking

For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where u 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.

BIT BANDING & Register Masking

Bit banding case study (For experts only)

You might want to know the reason 256 words (1024 bytes) are set aside for GPIODATA. As shown in the following figure, in order to make the addresses word aligned, bits 1 and 0 are always set to 0. Using bits 2 to 9 of the address, shows the bits that must be changed while writing to the GPIODATA register. For example, writing to address 0x34 (0000110100 in binary) means that pins 0, 2, and 4 of the port must be changed while the other pins remain unchanged. To change all the pins of the port, all bit masks (bits 2 to 9) must be set. This makes the offset address of 0x3FC (0011111111100 in binary).

As an example, writing to address 0x40004310 means that bits 2, 6, and 7 of Port A must be changed since 0x40004000 is the base address of Port A. See the following figure.

GPIO INPUT CONFIGURATION

- /* Read a switch and write it to the LED */
- #include "TM4C123GH6PM.h"
- int main(void)
- {
- unsigned int value;
- SYSCTL->RCGCGPIO |= 0x20; /* enable clock to GPIOF */
- GPIOF->DIR = 0x08; /* set PORTF3 pin as output (LED) pin */
- /* and PORTF4 as input, SW1 is on PORTF4 */
- GPIOF->DEN = 0x18; /* set PORTF pins 4-3 as digital pins */
- GPIOF->PUR = 0x10; /* enable pull up for pin 4 */

GPIO INPUT CONFIGURATION

- while(1)
- {
- value = GPIOF->DATA; /* read data from PORTF */
- value = ~value; /* switch is low active; LED is high active */
- value = value >> 1; /* shift it right to display on green LED */
- GPIOF->DATA = value; /* put it on the green LED */
- }
- }

GPIO PERIPHERAL DRIVER SW DESIGN

- We need the following services:
  1. Set Pin Service
  2. Read Pin Service
  3. Write Port Service
  4. Read Port Service