EMBEDDED SYSTEMS

UNIT-IV

Microcontroller Fundamentals for Basic Programming

RCEW, Pasupula (V), Nandikotkur Road,
Near Venkayapalli, KURNOOL

What is an Embedded System?

An embedded system is an electronic/electro mechanical system designed to perform a specific function and it is the combination of hardware and software. An embedded system performs only one particular task.

Examples:

- A Washing machine can only wash clothes.
- A Digital camera can only capture the images.
- An Air conditioner can control the temperature in the room.

I/O Pin Multiplexing

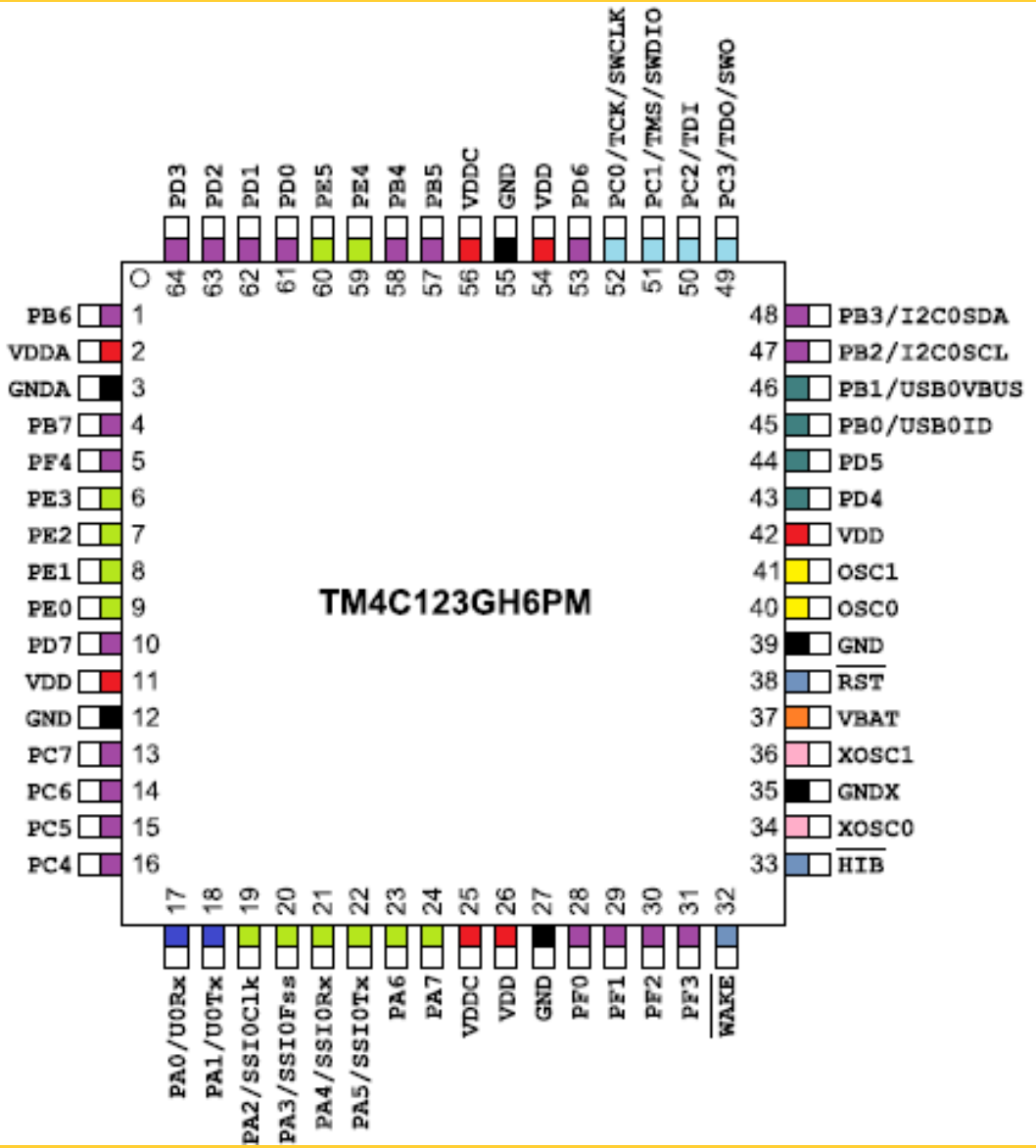
- The Microcontroller (System-On-Chip) has a lot of functionality but a limited number of pins (or pads).
- Even though a single pin can only perform one function at a time, they can be configured internally to perform different functions. This is called **Pin Multiplexing**. i.e. one pin can perform multiple functions.
- The regular function of a pin is to perform parallel I/O. Most of the pins have an alternative function.

I/O Pin Multiplexing

I/O pins on Tiva microcontrollers have a wide range of alternative functions:

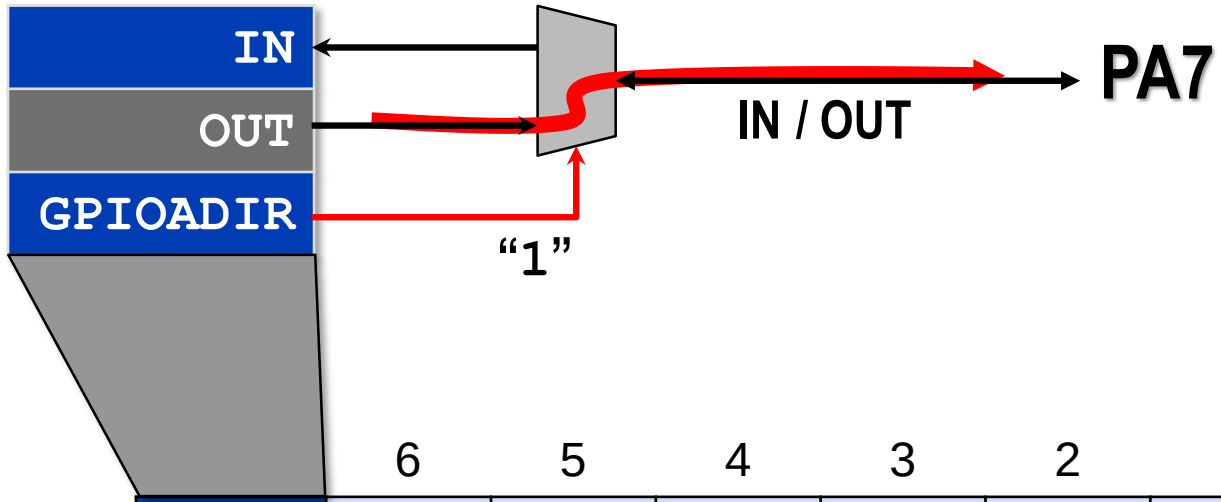
- **UART** - Universal Asynchronous Receiver/Transmitter
- **SPI/SSI** - Serial Peripheral Interface/Synchronous Serial Interface
- **I2C** - Inter-Integrated Circuit
- **Timer** - Periodic Interrupts, Input capture, and Output compare
- **PWM** - Pulse Width Modulation
- **ADC** - Analog to Digital Converter, Measure Analog Signals
- **QEI** - Quadrature Encoder Interface
- **USB** - Universal Serial Bus
- **CAN** - Controller Area Network
- **Ethernet** - High-Speed Network
- **Analog Comparator** - Compare two analog signals

I/O Pin Multiplexing



I/O Pin Multiplexing

GPIOADIR (Pin Direction): Input or Output



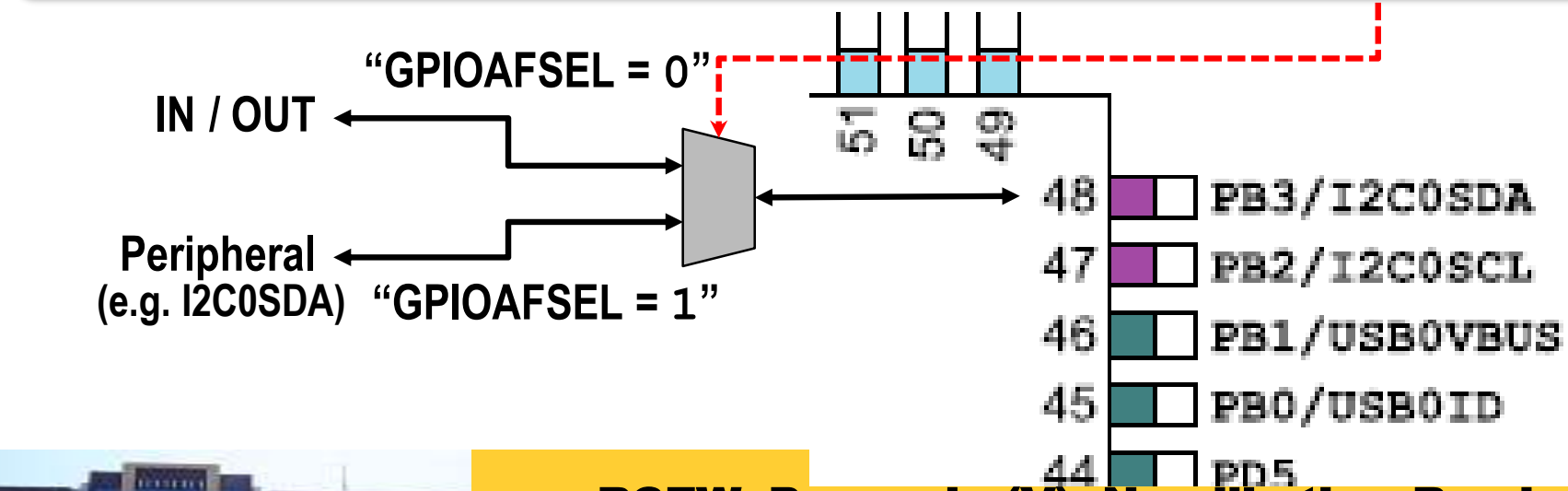
	6	5	4	3	2	1	0
PA IN							
PA OUT							
PA DIR	1						

Each bit in GPIO DIR register selects the direction of the corresponding I/O pin, regardless of the selected function for the pin.

I/O Pin Multiplexing

	7	6	5	4	3	2	1	0
GPIODATA								
GPIODIR								
GPIOPUR								
GPIOPDR								
GPIODEN								
GPIOAFSEL								

- **Most pins on MCU's are multiplexed to provide you with greater flexibility**
- **Often, two (or more) digital peripherals are connected to the pin** – in this case, some families use PxDIR to select between them, while others have multiple PxSEL registers



I/O Pull Up/Down Resistors

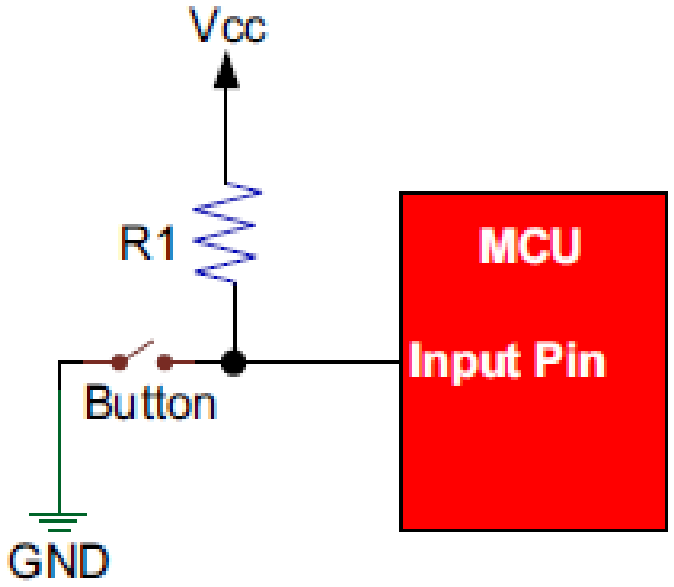


Figure 2-3 Pull-up Resistor for Input Pin

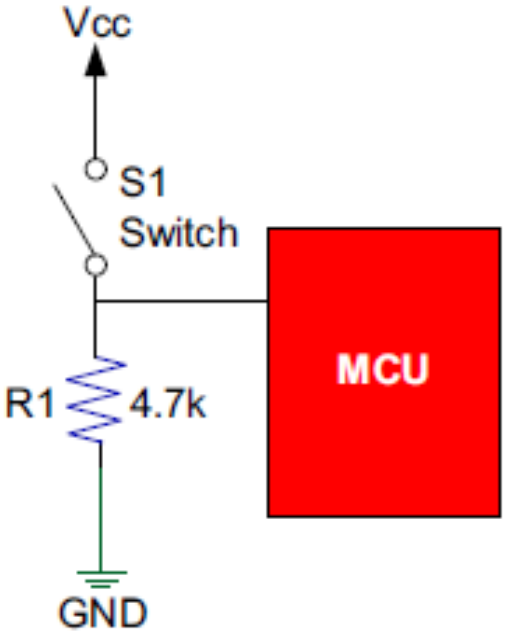


Figure 2-4 Pull-down Resistor for Input Pin

I/O Pull Up/Down Resistors

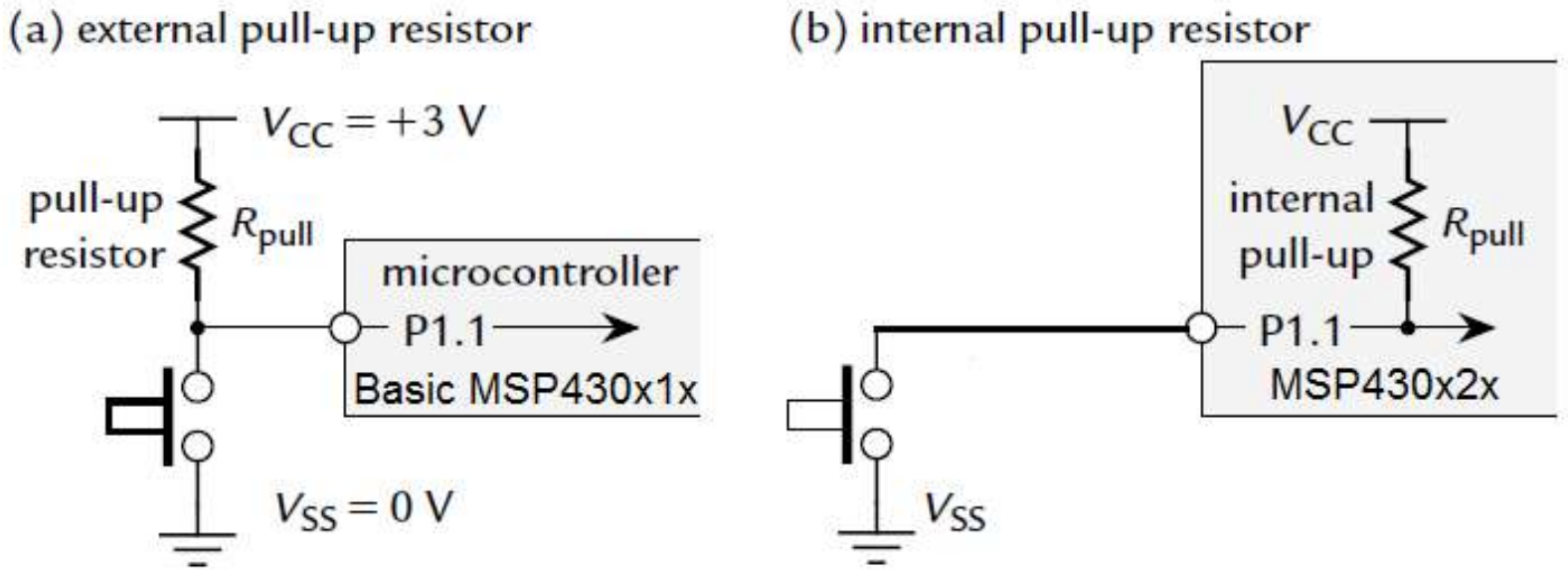
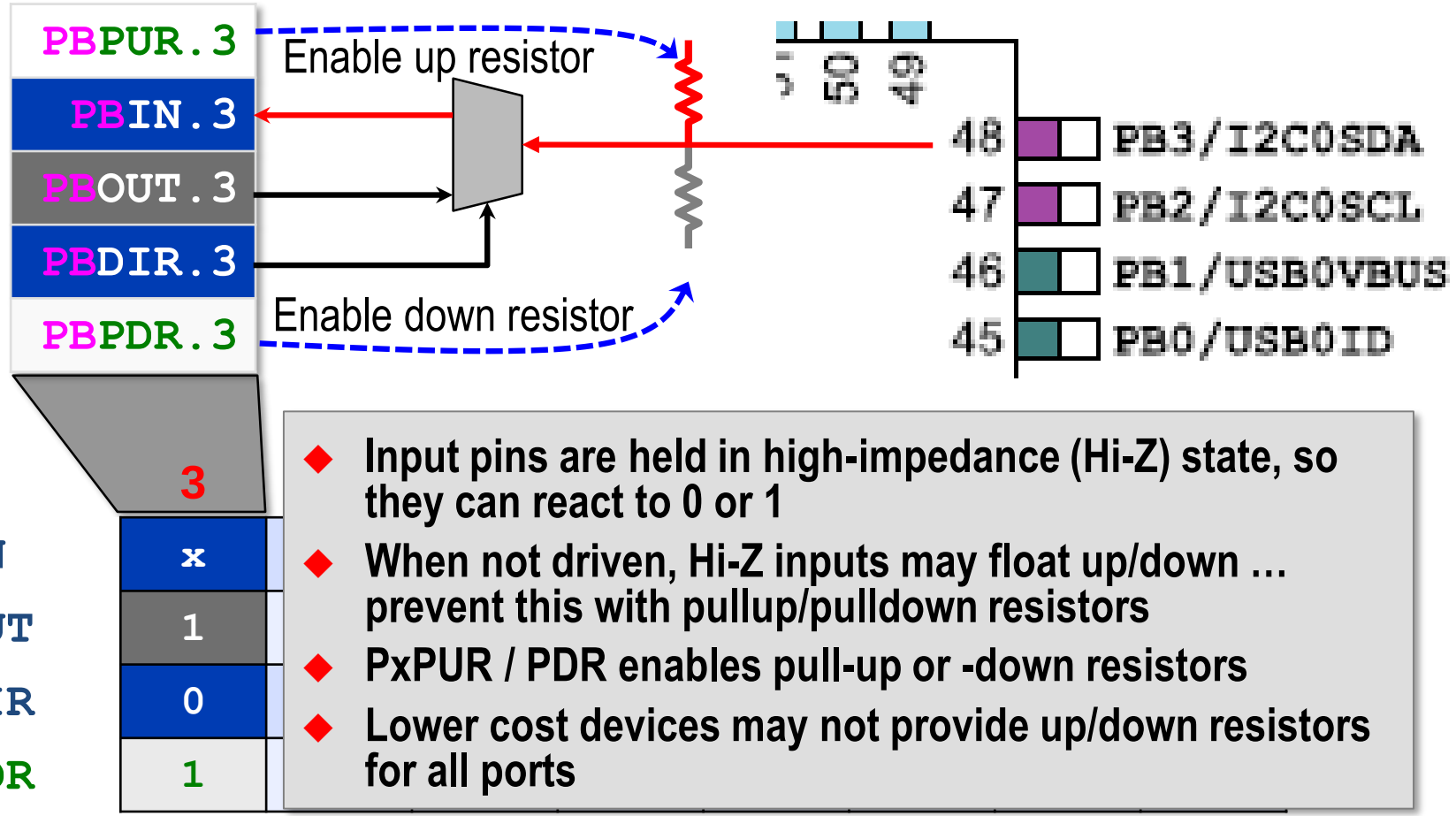


Figure: Standard, active low connection of a push button to an input with an (a) external and (b) internal pull-up resistor R_{pull}.

I/O Pull Up/Down Resistors



- Input pins are held in high-impedance (Hi-Z) state, so they can react to 0 or 1
- When not driven, Hi-Z inputs may float up/down … prevent this with pullup/pulldown resistors
- PxPUR / PDR enables pull-up or -down resistors
- Lower cost devices may not provide up/down resistors for all ports

Each bit in GPIOPUR/GPIOPDR register selects the Pull-up / Pull-down Resister of the corresponding I/O pin.

GPIO Control

TM4C123x Digital I/O Ports (GPIO)

GPIO Module Features:

- There are up to **6** I/O ports **PA- PF** each Port is 8 bit wide
- This GPIO module supports up to 43 programmable input/output pins
- The GPIO pins are flexibly multiplexed. This allows it to be also used as peripheral functions.
- The GPIO pins are 5V-tolerant in input configuration
- Ports A-F are 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
- All individual I/O bits are independently programmable.
- Any combination of input, output, and interrupt conditions is possible.
- Individually configurable pull-up or pull-down resistors
- Read and write access to port-control registers is supported by all instructions.
- Ports can be accessed byte-wise (PA through PF) OR bit wise

GPIO Control

Advanced features of GPIO in Tiva Launchpad:

- Weak pull-up or pull-down resistors
- 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can sink
- 18-mA for high-current applications
- Slew rate control for 8-mA pad drive
- Open drain enables
- Digital input enables

GPIO Control

TM4C123x GPIO Ports

- GPIO = General Purpose Input/Output
- 8-bit I/O ports
- 1 to 6 ports, depending on family and pin-count
- Each pin is individually controllable
- Input pins can generate interrupts
- Controlled by memory-mapped registers / Programming System Registers:
  - GPIODATA
  - GPIODIR
  - GPIOPUR
  - GPIOPDR
  - GPIOAFSEL
  - GPIODEN
  - GPIOPCTL
  - …

	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
I/O Port A	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Programming System Registers

- **Memory-Mapped Registers / Programming System Registers:**

Name	Description
GPIODATA	GPIO Data
GPIODIR	GPIO Direction
GPIOPUR	GPIO Pull-Up Resistor
GPIOPDR	GPIO Pull-Down Resistor
GPIOAFSEL	GPIO Alternate Function Select
GPIODEN	GPIO Digital Enable
GPIOPCTL	GPIO Port Control
.....	

Programming System Registers

GPIODATA: The GPIODATA register is the data register. In software control mode, values written in the GPIODATA register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the GPIO Direction (GPIODIR) register.

GPIODIR: The GPIODIR register is the data direction register. Setting a bit in the GPIODIR register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIOPUR: The GPIOPUR register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in GPIOPUR automatically clears the corresponding bit in the GPIO Pull-Down Select (GPIOPDR) register.

GPIOPDR: The GPIOPDR register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in GPIOPDR automatically clears the corresponding bit in the GPIO Pull-Up Select (GPIOPUR) register.

GPIOAFSEL: The GPIOAFSEL register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The GPIO Port Control (GPIOPCTL) register is used to select one of the possible functions.

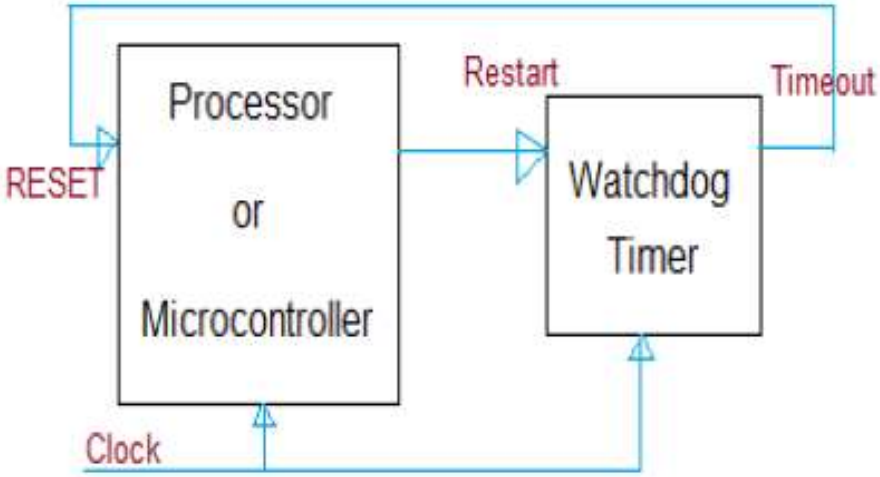
Programming System Registers

GPIODEN: The GPIODEN register is the digital enable register. The digital function of some GPIO Pins are disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use these pins as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

GPIOPCTL: The GPIOPCTL register is used in conjunction with the GPIOAFSEL register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the GPIOAFSEL register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the GPIOAFSEL register, the corresponding GPIO signal is controlled by an associated peripheral. The GPIOPCTL register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition

Watchdog Timer

- Every CPU has a system clock which drives the program counter. In every cycle, the program counter executes instructions stored in the flash memory of a microcontroller.
- These instructions are executed sequentially. There exist possibilities where a remotely installed system may freeze or run into an unplanned situation which may trigger an infinite loop.
- On encountering such situations, system reset or execution of the interrupt subroutine remains the only option. Watchdog timer provides a solution to this.
- A watchdog timer counter enters a counter lapse or timeout after it reaches certain count. Under normal operation, the program running the system continuously resets the watchdog timer.



- When the system enters an infinite loop or stops responding, it fails to reset the watchdog timer. In due time, the watchdog timer enters counter lapse.
- This timeout will trigger a reset signal to the system or call for an interrupt service routine (ISR).

Watchdog Timer

- The TM4C123GH6PM microcontroller has two Watchdog Timer modules, one module is clocked by the system clock (Watchdog Timer 0) and the other (Watchdog Timer 1) is clocked by the PIOSC therefore it requires synchronizers

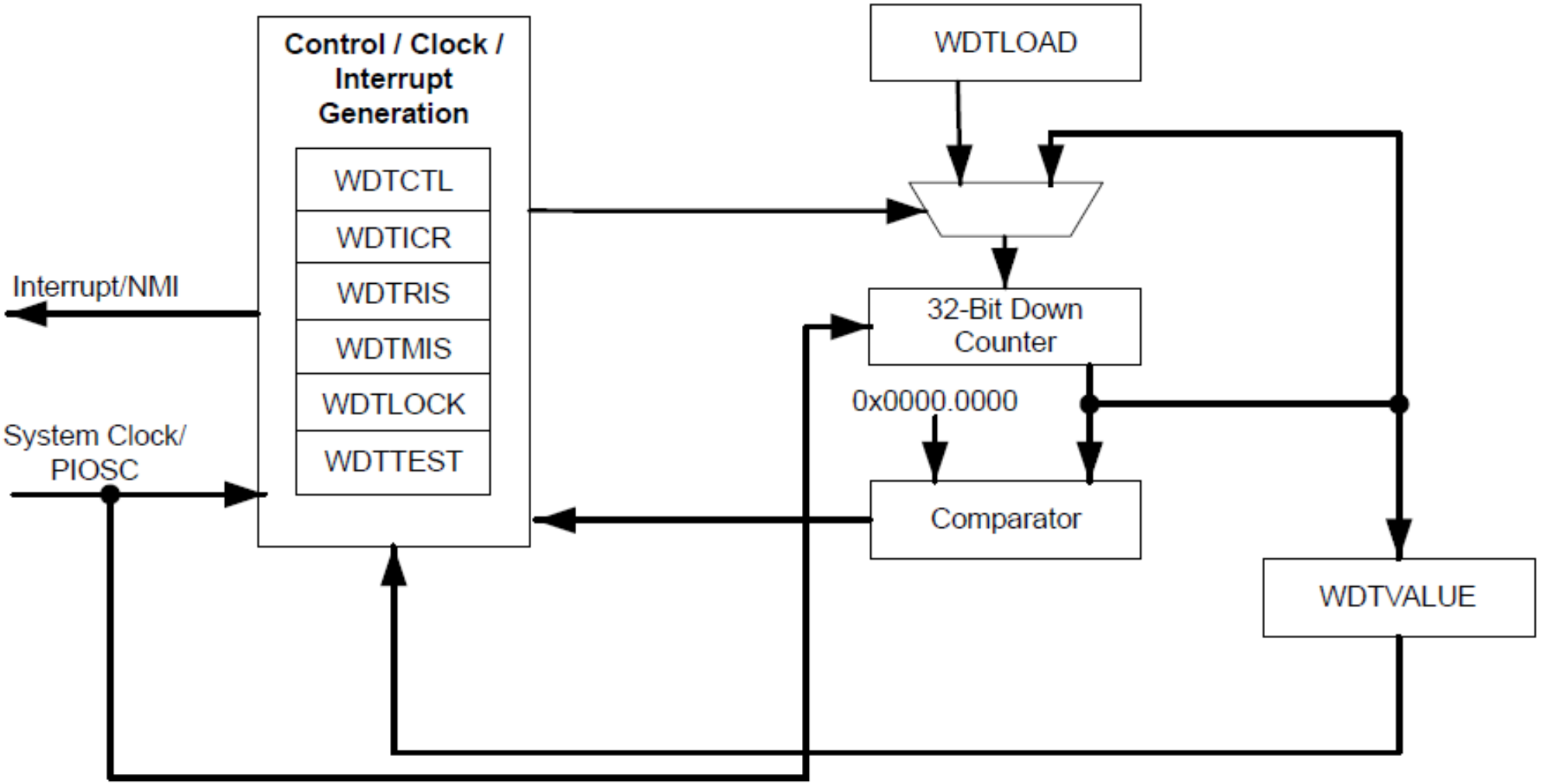
Features of Watchdog Timer in TM4C123GH6PM controller:

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking & optional NMI function
- Lock register protection from runaway software
- Reset generation logic with an enable/disable

- The watchdog timer can be configured to generate an interrupt to the controller on its first time out, and to generate a reset signal on its second time-out.
- Once the watchdog timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

Watchdog Timer

WDT Module Block Diagram



Watchdog Timer

Functional Description:

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. The watchdog interrupt can be programmed to be a Non-Maskable Interrupt (NMI) using the INTTYPE bit in the WDTCTL register. After the first time-out event, the 32-bit counter is re-loaded with the value of the Watchdog Timer Load (WDTLOAD) register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the Watchdog Timer Lock (WDTLOCK) register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the RESEN bit in the WDTCTL register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the WDTLOAD register, and counting resumes from that value.

If WDTLOAD is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to WDTLOAD does not clear an active interrupt. An interrupt must be specifically cleared by writing to the Watchdog Interrupt Clear (WDTICR) register.

The watchdog timer is disabled by default out of reset. To achieve maximum watchdog protection of the device, the watchdog timer can be enabled at the start of the reset vector.

Need Of Low Power For Embedded Systems

Need Of Low Power For Embedded Systems:

It is imperative for an embedded design to be low on its power consumption. Most embedded systems and devices run with a battery. Power demands are increasing rapidly, but battery capacity cannot keep up with its pace. Therefore, a microcontroller that inherently consumes very less power is always encouraging. However, embedded systems engineers usually need to optimize power and performance. Power and performance are inversely proportional to each other.

Low-power embedded design is motivated by the need to run applications for as long as possible while consuming minimum power. In a battery-powered system, this need is magnified. Furthermore, low power implies lower cost of operation and smaller battery size to make applications more mobile. When energy comes at a premium as it does with today's green initiatives, ensuring that an embedded design consumes as little energy as possible is even important for wall-powered applications.

Designing power-efficient applications also ensures less overhead to manage thermal dissipation, and heat generation is controlled at the source by optimizing the power consumed

System Clocks and Control

System Clock:

Clock is like heart of the controller and used to synchronize all the peripherals in the microcontroller. There are multiple clock sources for use in the TIVA microcontroller.

Precision Internal Oscillator (PIOSC): The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a 16-MHz clock with ±1% accuracy with calibration and ±3% accuracy across temperature.

Main Oscillator (MOSC): The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 5 MHz to 25 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 4 MHz to 25 MHz.

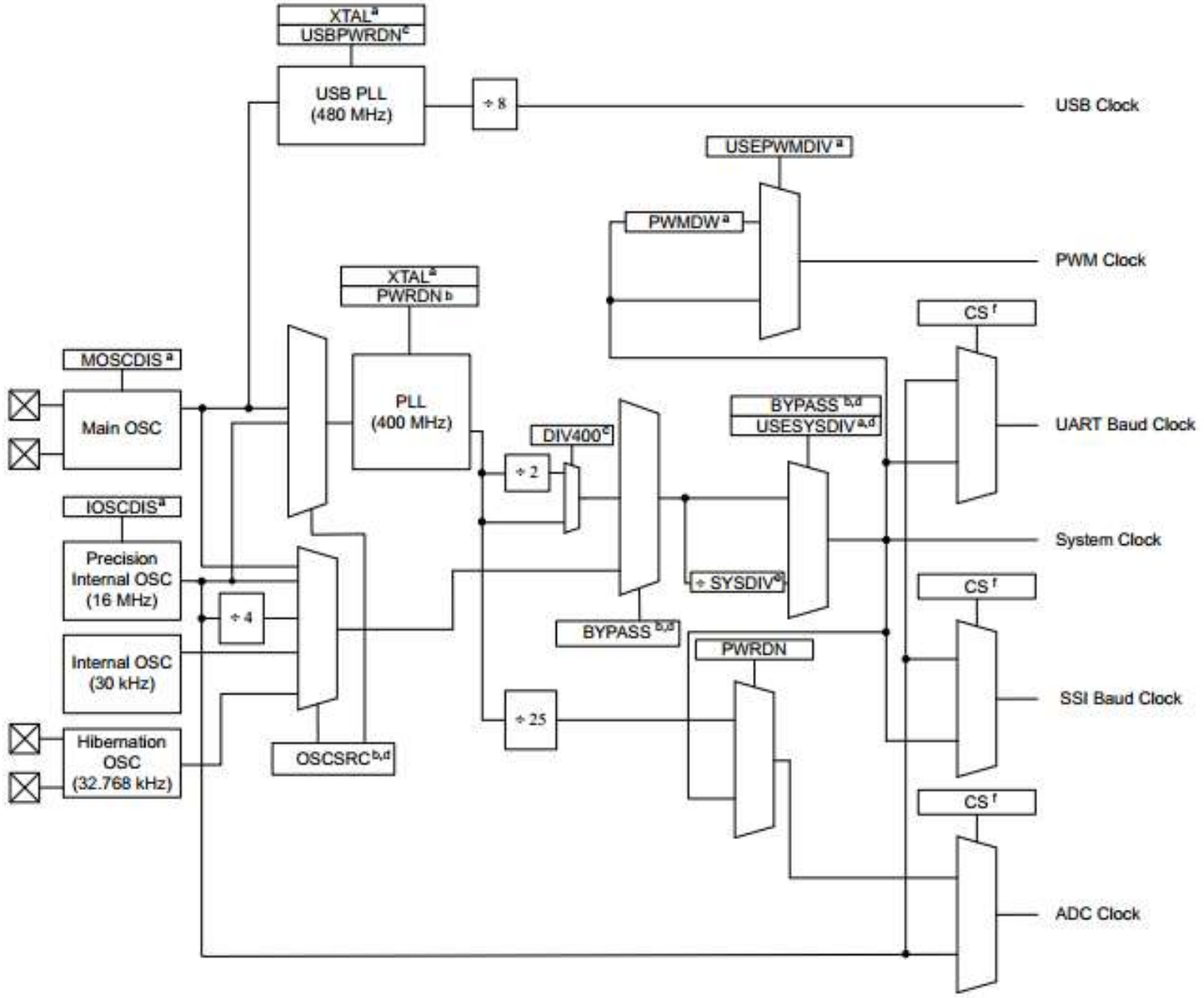
Low-Frequency Internal Oscillator (LFIOSC): The low-frequency internal oscillator is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC to be powered down. In addition, the PIOSC can be powered down while in Deep-Sleep mode.

Hibernation Module Clock Source: The Hibernation module is clocked by a 32.768-kHz oscillator connected to the XOSC0 pin. The 32.768-kHz oscillator can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. The Hibernation module clock source is intended to provide the system with a real-time clock source and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

System Clocks and Control

System Clock:

- The Figure shows the clock tree of TIVA microcontroller.
- From the different clock sources, different clocks are derived. i.e. USB Clock, PWM Clock, UART Baud Clock, System Clock, SSI Baud Clock, ADC Clock



System Clocks and Control

System Control:

For power-savings purposes, the peripheral-specific RCGCx, SCGCx, and DCGCx registers (for example, RCGCWD) control the clock gating logic for that peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively.

There are four levels of operation for the microcontroller defined as:

- Run mode
- Sleep mode
- Deep-Sleep mode
- Hibernate mode

Run Mode: In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the peripheral-specific RCGC registers. The system clock can be any of the available clock sources including the PLL.

Sleep Mode: In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M4F core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode.

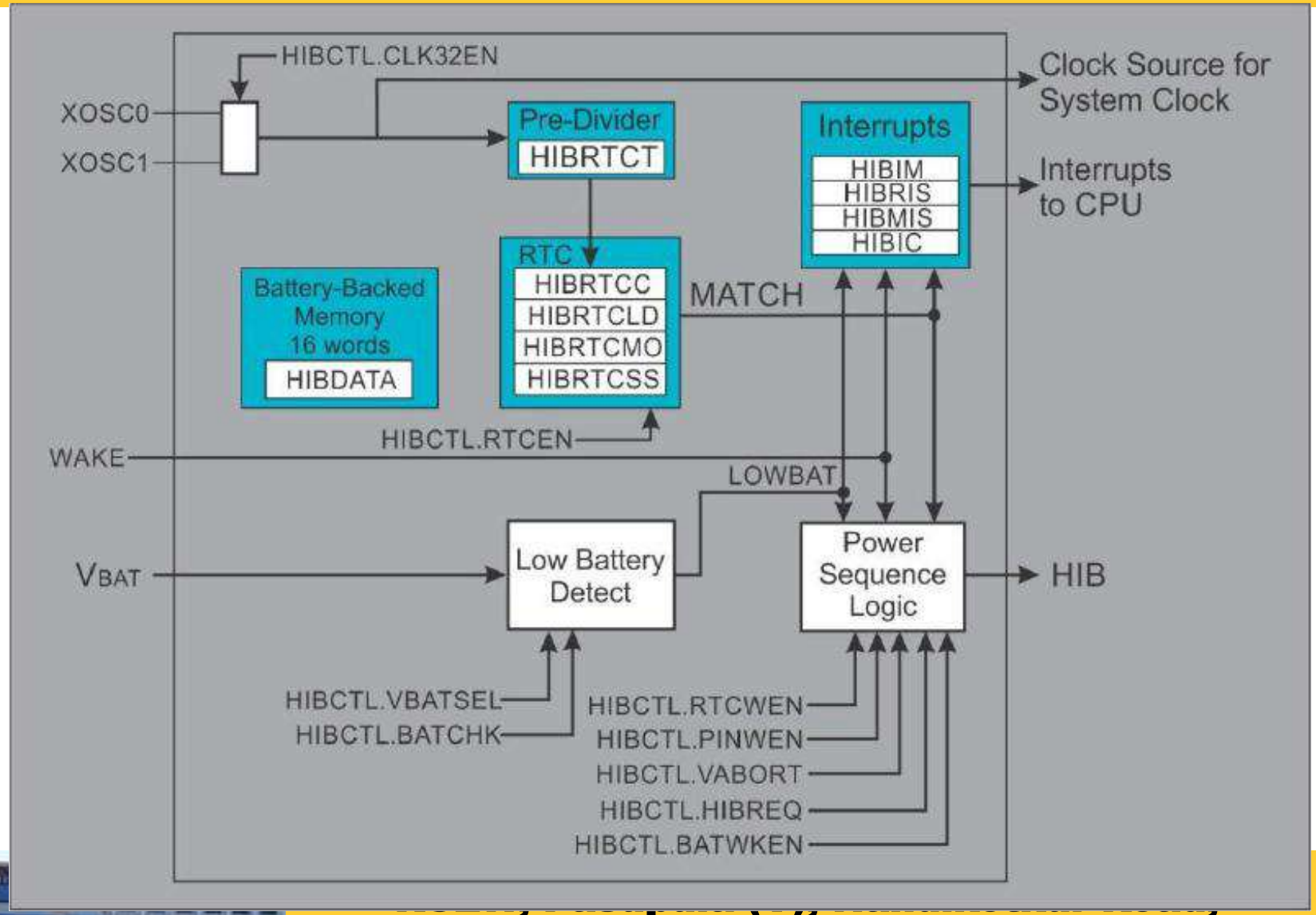
System Clocks and Control

System Control:

Deep-Sleep Mode: In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Deep-Sleep mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first setting the SLEEPDEEP bit in the System Control (SYSCTRL) register and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode.

Hibernate Mode: In this mode, the power supplies are turned off to the main part of the microcontroller and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the microcontroller back to Run mode.

Hibernation Module On TM4C



Hibernation Module On TM4C

This module manages to remove and restore power to the microcontroller and its associated peripherals. This provides a means for reducing system power consumption. When the processor and peripherals are idle, power can be completely removed if the Hibernation module is only the one powered.

- **To achieve this, the Hibernation (HiB) Module is added with following features:**
  - (i) A Real-Time Clock (RTC) to be used for wake events
  - (ii) A battery backed SRAM for storing and restoring processor state. The SRAM consists of 16 32-bit word memory.

The Hibernation module of TM4C123GH6PM provides two mechanisms for power control:

- The first mechanism uses internal switches to control power to the Cortex-M4F.
- The second mechanism controls the power to the microcontroller with a control signal (HIB) that signals an external voltage regulator to turn on or off.
- The Hibernation module power source is determined dynamically.

Hibernate mode can be entered through one of two ways:

- The user initiates hibernation by setting the HIBREQ bit in the Hibernation Control (HIBCTL)
- register. Power is arbitrarily removed from VDD while a valid VBAT is applied

Active Vs Standby Current Consumption

Power Modes:

- There are six power modes in which TM4C123GH6PM operates.
- They are Run, Sleep, Deep Sleep, Hibernate with VDD3ON, Hibernate with RTC, and Hibernate without RTC.
- To understand all these modes and compare them, it is necessary to analyze them under a condition. Let us consider that the device is operating at 40 MHz system clock with PLL.

Mode → / Parameter ↓	Run Mode	Sleep Mode	Deep Sleep Mode	Hibernation (VDD3ON)	Hibernation (RTC)	Hibernation (no RTC)
I_{DD}	32 mA	10 mA	1.05 mA	5 µA	1.7 µA	1.6 µA
V_{DD}	3.3 V	3.3 V	3.3 V	3.3 V	0 V	0 V
V_{BAT}	N.A.	N.A.	N.A.	3 V	3 V	3 V
System Clock	40 MHz with PLL	40 MHz with PLL	30 kHz	Off	Off	Off
Core	Powered On	Powered On	Powered On	Off	Off	Off
	Clocked	Not Clocked	Not Clocked	Not Clocked	Not Clocked	Not Clocked
Peripherals	All On	All Off	All Off	All Off	All Off	All Off
Code	while{1}	N.A.	N.A.	N.A.	N.A.	N.A.